# Real-Time Multiprocessor Locks with Nesting: Optimizing the Common Case*

Catherine E. Nemitz Tanya Amert James H. Anderson

**Abstract**

In prior work on multiprocessor real-time locking protocols, only protocols within the RNLP family support unrestricted lock nesting while guaranteeing asymptotically optimal priority-inversion blocking bounds. However, these protocols support nesting at the expense of increasing the cost of processing non-nested lock requests, which tend to be the common case in practice. To remedy this situation, a new *fast-path mechanism* is presented herein that extends prior RNLP variants by ensuring that non-nested requests are processed efficiently. This mechanism yields overhead and blocking costs for such requests that are nearly identical to those seen in the most efficient single-resource locking protocols. In experiments, the proposed fast-path mechanism enabled observed blocking times for non-nested requests that were up to 18 times lower than under an existing RNLP variant and improved schedulability over that variant and a simple group lock.

## Additional Graphs

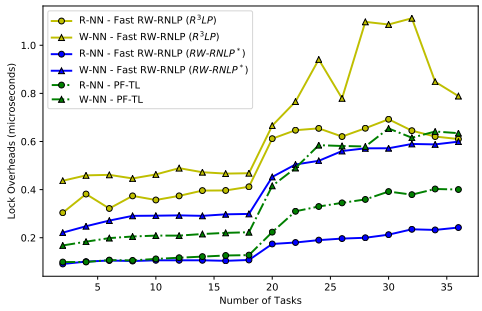
Here we include the results from our protocol testing and schedulability study corresponding to the paper of this title, which can be found online.[1]

To evaluate both variants of our new protocol, we explored the tradeoffs between overhead and blocking via user-space experiments under a variety of scenarios. In our experiments, we varied the number of tasks ($n$), nesting depth ($\mathbb{D} = |D_i|$), critical-section length ($L_i$), probability of a request being nested (rather than non-nested), and probability of a request being a read request (rather than a write request). Each task was pinned to a single core, and for task counts of up to 18, all tasks were assigned to the same socket. To simulate behavior that would generate the worst-case lock overhead and blocking times, each task was configured to issue lock and unlock calls 10,000 times, as fast as possible. Each such lock-unlock call pair corresponded to a single request that was randomly chosen to be nested (or non-nested) and a read (or a write) given the scenario's parameters. In our graphs, we plot these worst-case values, which were obtained by computing the $99^{\text{th}}$ percentile of all recorded results in order to filter out any spurious measurements.

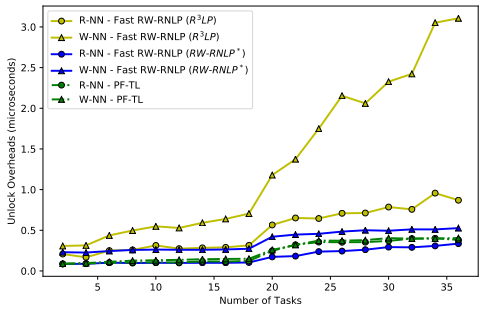
We also conducted a large-scale overhead-aware schedulability study to explore the impact of the worst-case acquisition delays on system schedulability. We used SchedCAT, an open-source real-time schedulability test toolkit, to randomly generate task systems, implement blocking bound computations, and check for schedulability on an 18-core platform with global EDF scheduling. We generated task sets with specified system utilizations, task periods lengths, and per-task utilizations. Tasks were chosen to issue a single request with a given probability, and the probability of a given request being a read (as opposed to a write) or nested (as opposed to non-nested) was specified. For each generated task system, we computed the impact of blocking on each request given the constraints discussed above. We did not consider most types of overhead, such as migration overhead, release blocking, and other overhead sources that impact each scheme similarly. However, we did account for the overhead incurred by using a specific locking protocol; we applied the appropriate overhead values for nested and non-nested read and write requests based on the prevoiusly described experiments. We chose the maximum values of lock and unlock overhead from relevant scenarios.

*Work supported by NSF grants CNS 1409175, CPS 1446631, CNS 1563845, and CNS 1717589, AFOSR grant FA9550-14-1-0161, ARO grant W911NF-14-1-0499, ARO grant W911NF-17-1-0294, and funding from General Motors. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGS 1650116. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

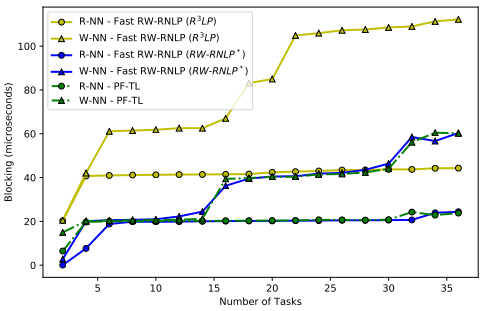
[1]See paper online: `http://www.cs.unc.edu/~anderson/papers.html`.




(a) Lock overhead.

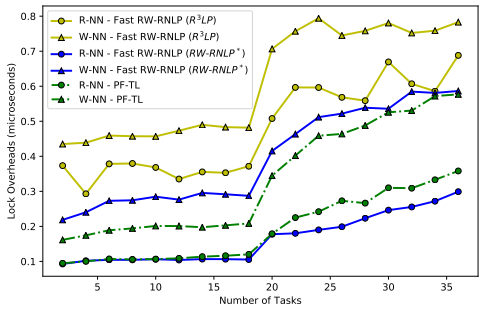



(b) Unlock overhead.

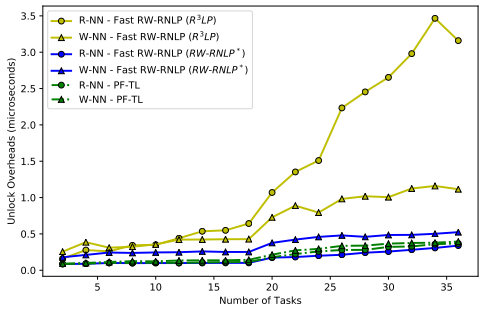



(c) Blocking.

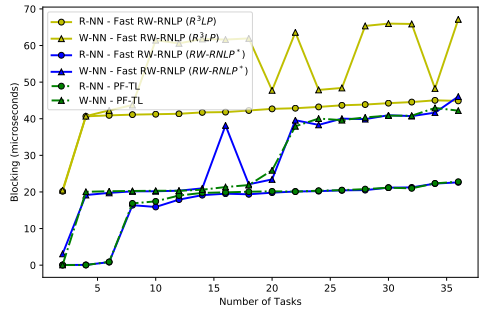
Figure 1: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the PF-TL and the fast RW-RNLP variants. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2.




(a) Lock overhead.

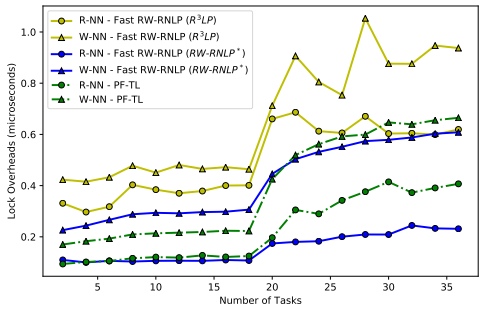



(b) Unlock overhead.

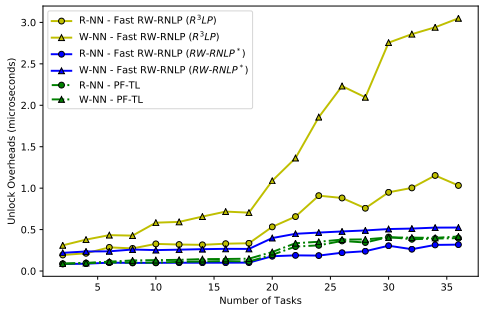



(c) Blocking.

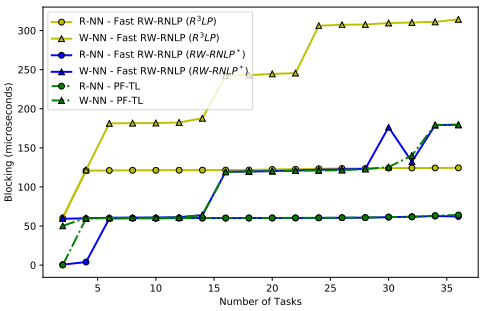
Figure 2: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the PF-TL and the fast RW-RNLP variants. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8.




(a) Lock overhead.

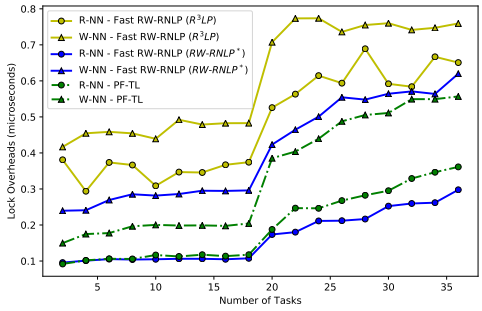



(b) Unlock overhead.

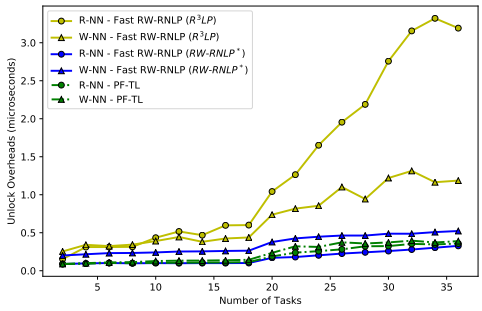



(c) Blocking.

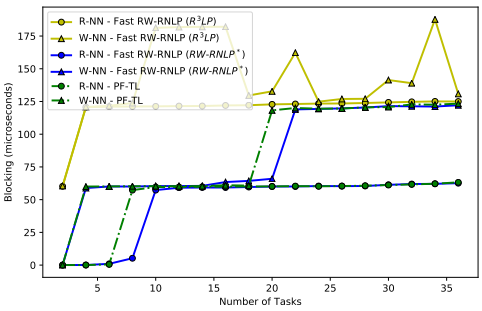
Figure 3: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the PF-TL and the fast RW-RNLP variants. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2.




(a) Lock overhead.

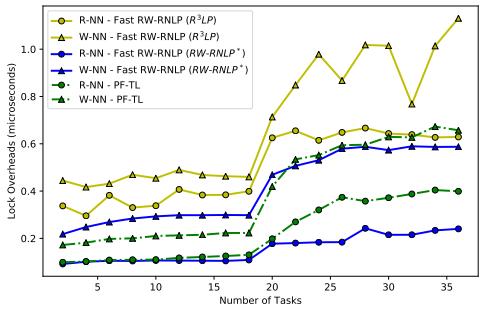



(b) Unlock overhead.

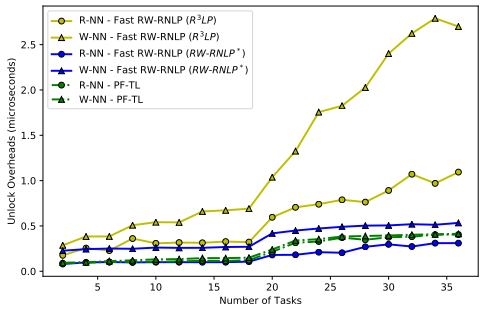



(c) Blocking.

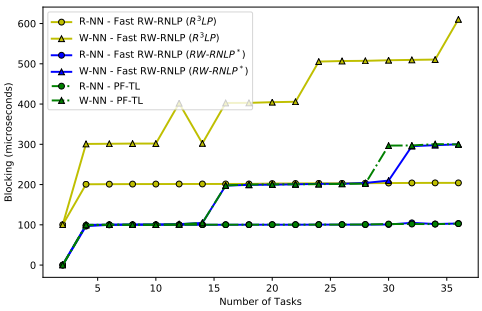
Figure 4: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the PF-TL and the fast RW-RNLP variants. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8.




(a) Lock overhead.

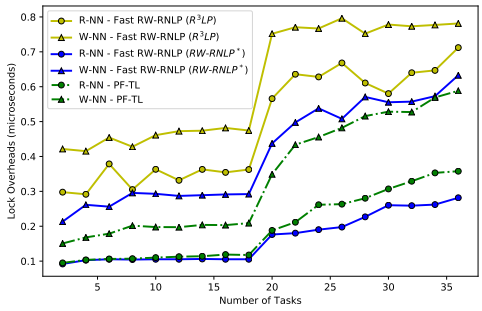



(b) Unlock overhead.

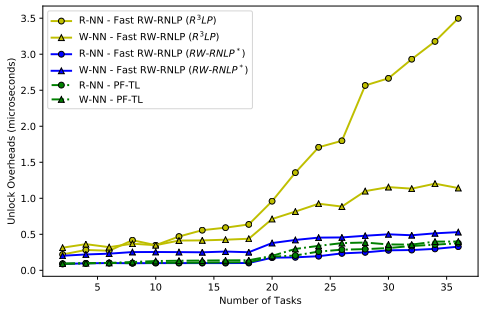



(c) Blocking.

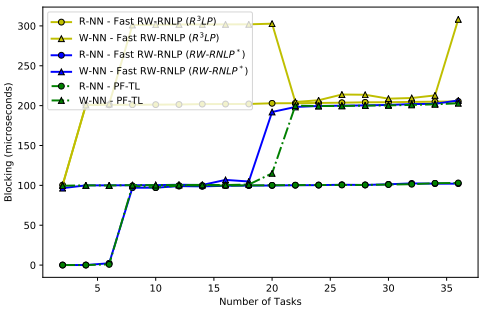
Figure 5: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the PF-TL and the fast RW-RNLP variants. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2.




(a) Lock overhead.

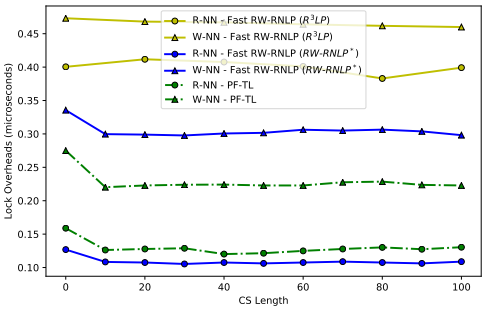



(b) Unlock overhead.

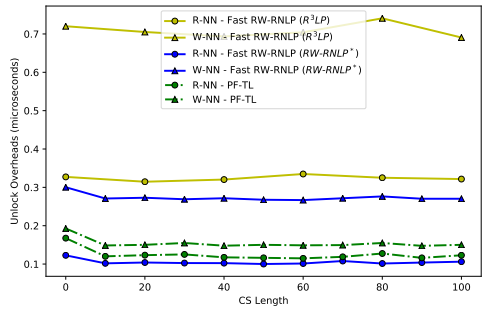



(c) Blocking.

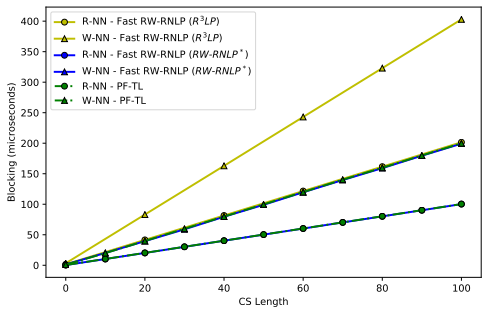
Figure 6: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the PF-TL and the fast RW-RNLP variants. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8.




(a) Lock overhead.

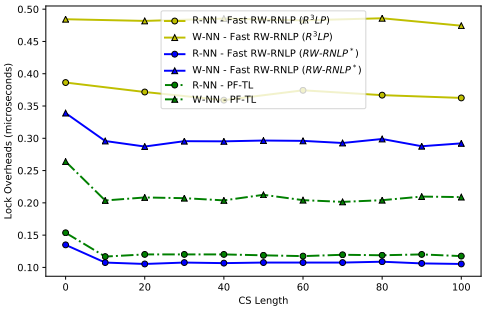



(b) Unlock overhead.

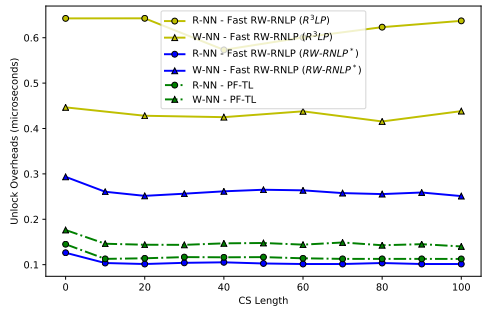



(c) Blocking.

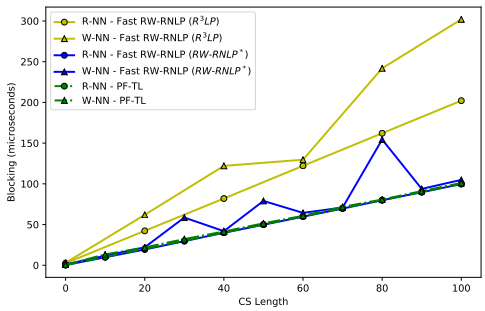
Figure 7: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the PF-TL and the fast RW-RNLP variants. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$,and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2.




(a) Lock overhead.

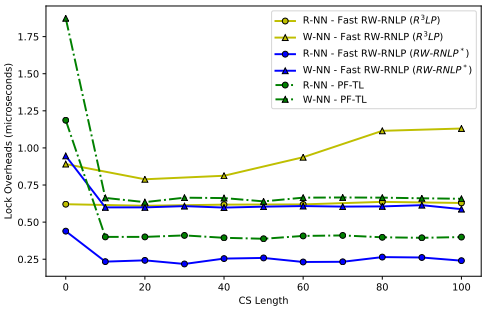



(b) Unlock overhead.

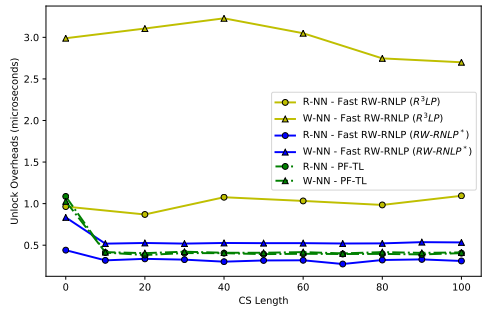



(c) Blocking.

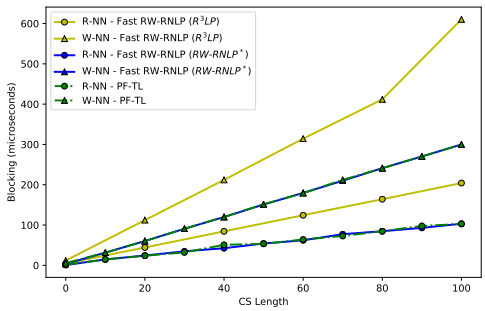
Figure 8: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the PF-TL and the fast RW-RNLP variants. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$,and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8.




(a) Lock overhead.

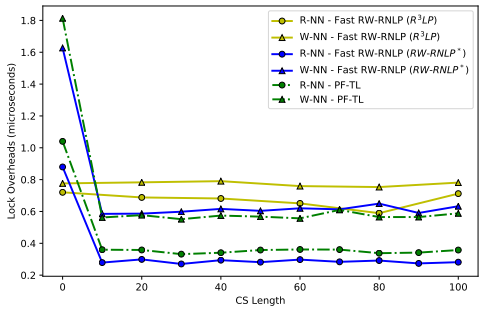



(b) Unlock overhead.

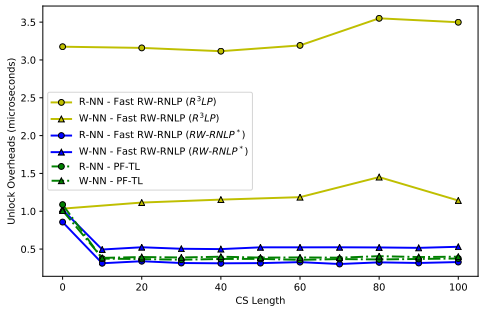



(c) Blocking.

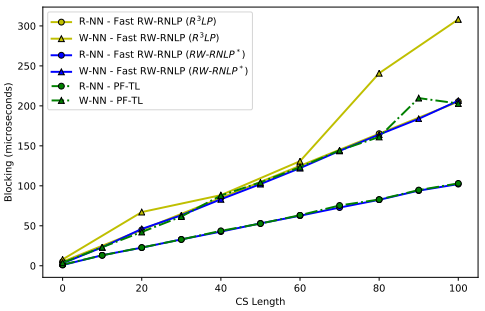
Figure 9: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the PF-TL and the fast RW-RNLP variants. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$,and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2.




(a) Lock overhead.

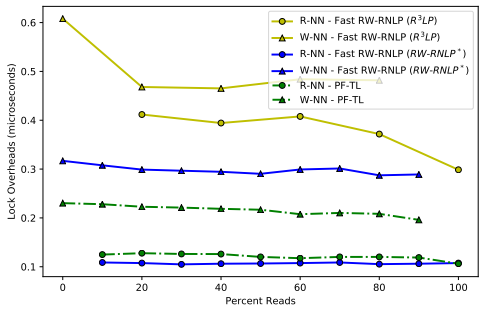



(b) Unlock overhead.

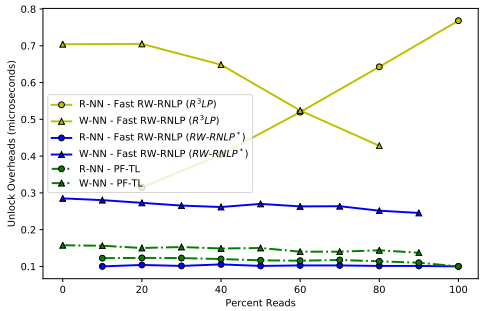



(c) Blocking.

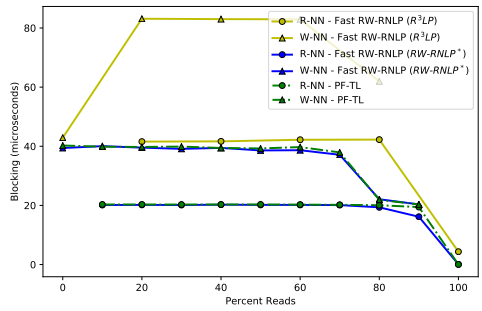
Figure 10: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the PF-TL and the fast RW-RNLP variants. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$,and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8.




(a) Lock overhead.

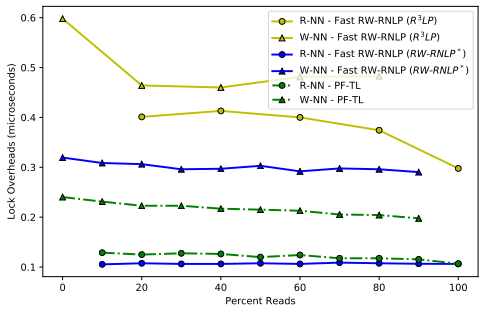



(b) Unlock overhead.

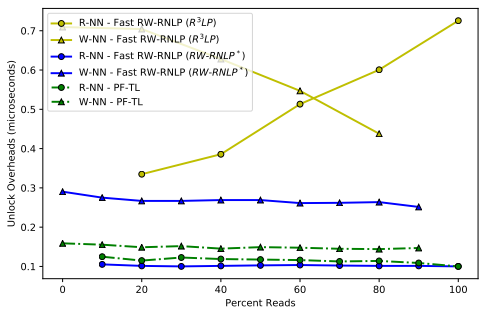



(c) Blocking.

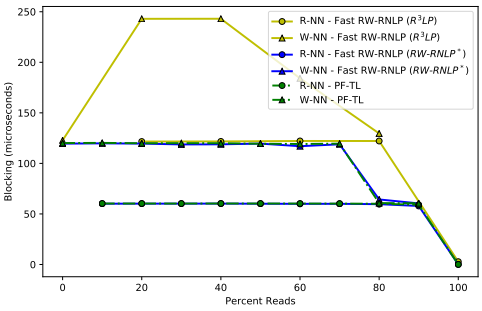
Figure 11: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the PF-TL and the fast RW-RNLP variants. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown.




(a) Lock overhead.

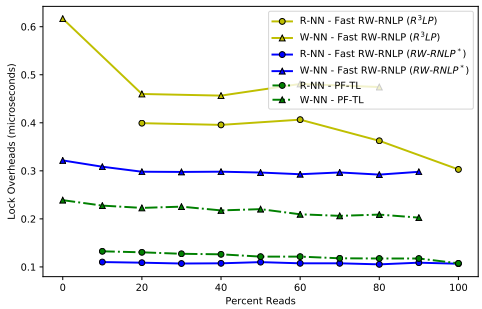



(b) Unlock overhead.

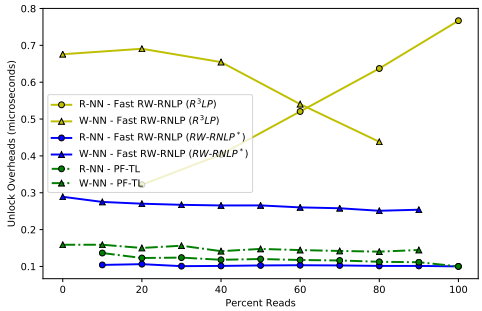



(c) Blocking.

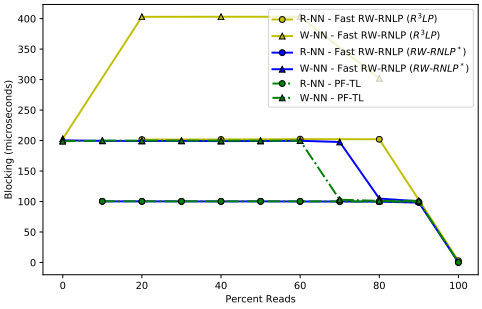
Figure 12: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the PF-TL and the fast RW-RNLP variants. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown.




(a) Lock overhead.

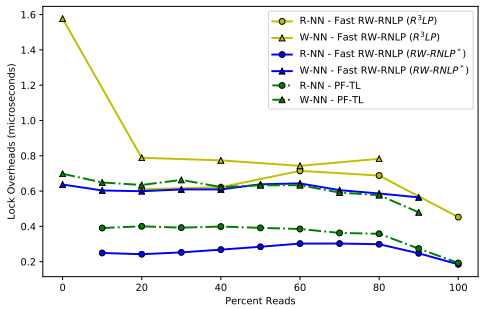



(b) Unlock overhead.

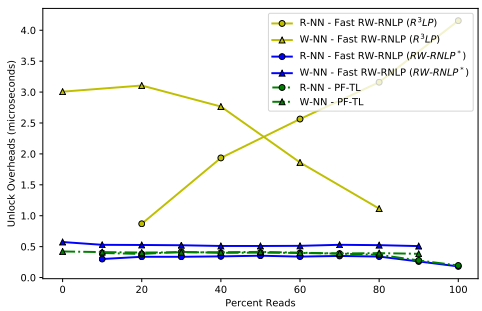



(c) Blocking.

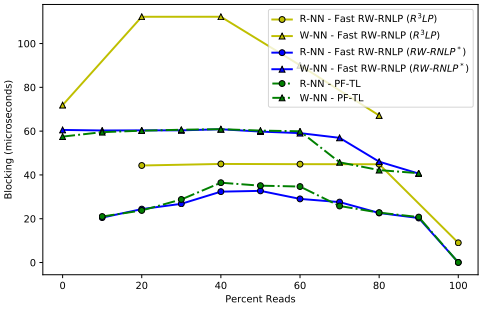
Figure 13: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the PF-TL and the fast RW-RNLP variants. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown.




(a) Lock overhead.

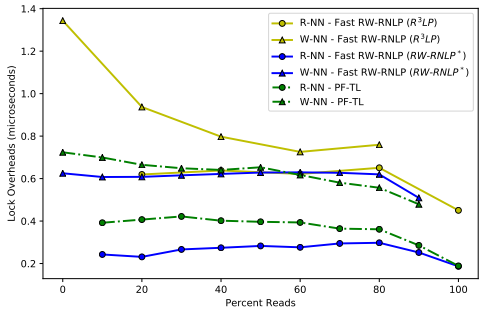



(b) Unlock overhead.

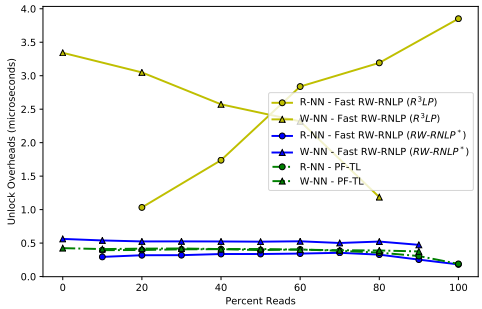



(c) Blocking.

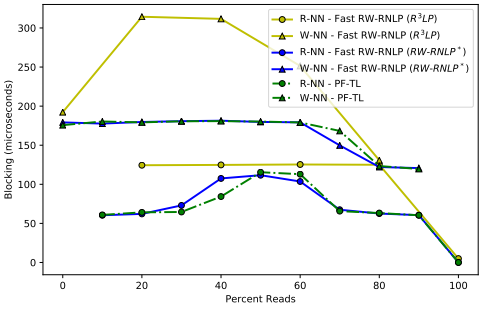
Figure 14: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the PF-TL and the fast RW-RNLP variants. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown.




(a) Lock overhead.

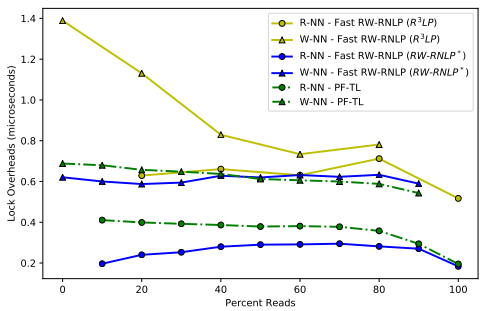



(b) Unlock overhead.

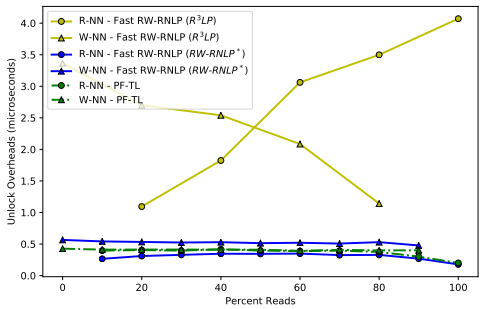



(c) Blocking.

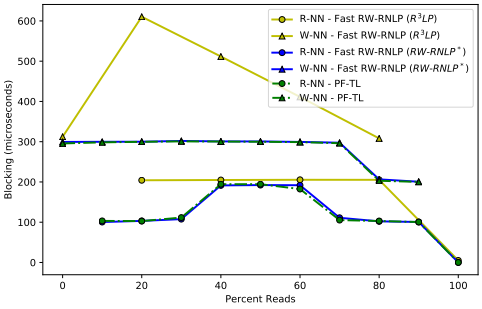
Figure 15: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the PF-TL and the fast RW-RNLP variants. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown.




(a) Lock overhead.

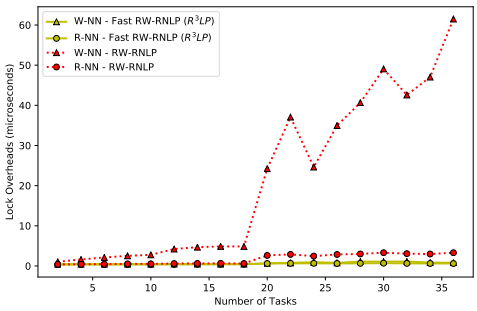



(b) Unlock overhead.

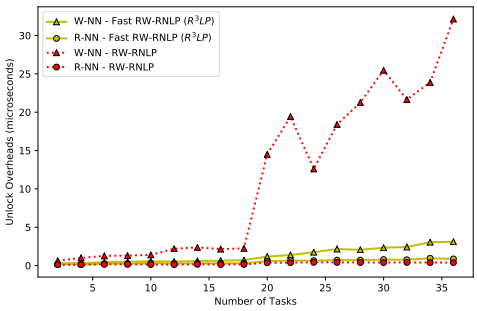



(c) Blocking.

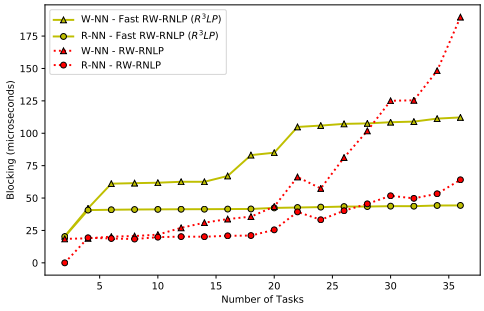
Figure 16: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the PF-TL and the fast RW-RNLP variants. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown.




(a) Lock overhead.

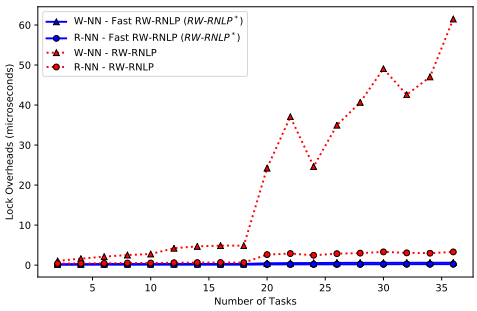



(b) Unlock overhead.

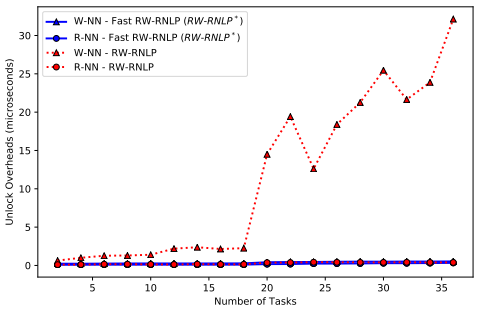



(c) Blocking.

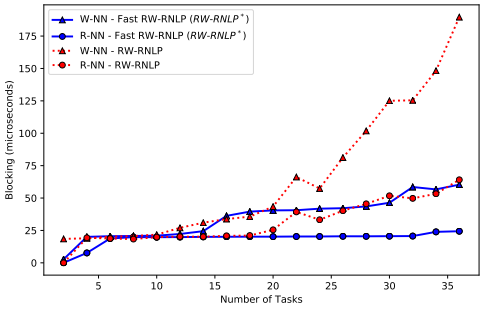
Figure 17: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

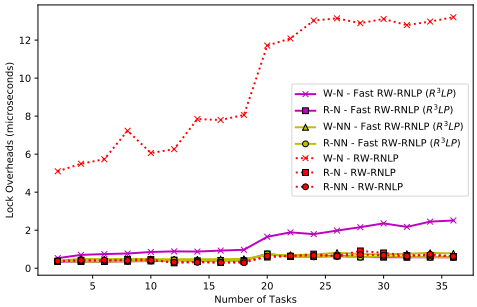



(b) Unlock overhead.

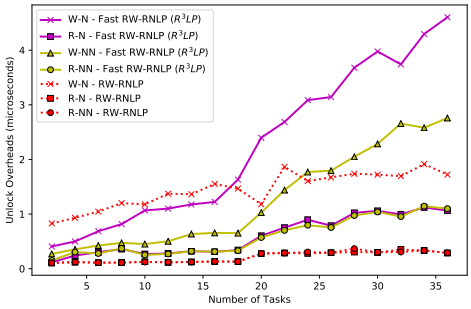



(c) Blocking.

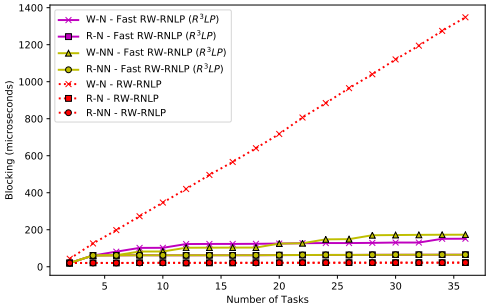
Figure 18: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

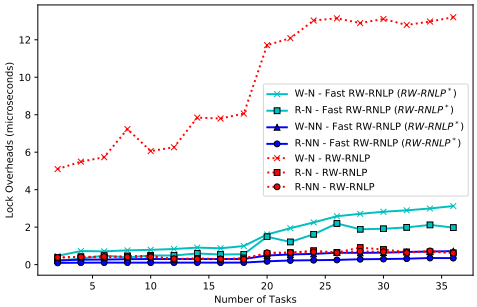



(b) Unlock overhead.

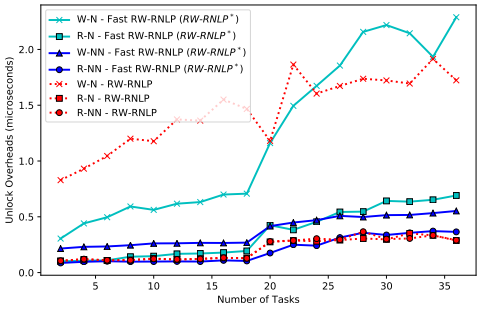



(c) Blocking.

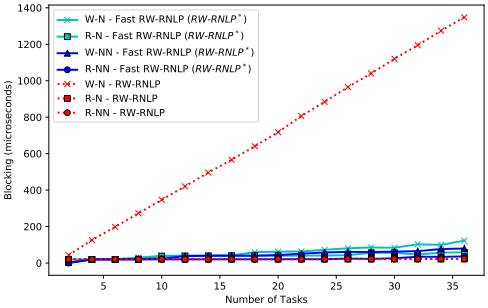
Figure 19: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

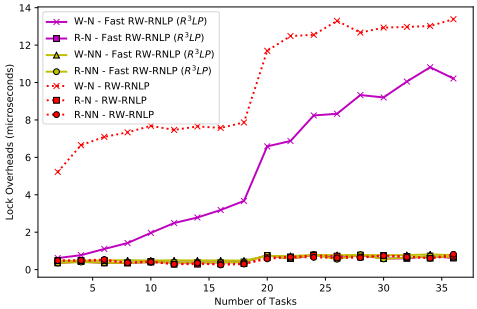



(b) Unlock overhead.

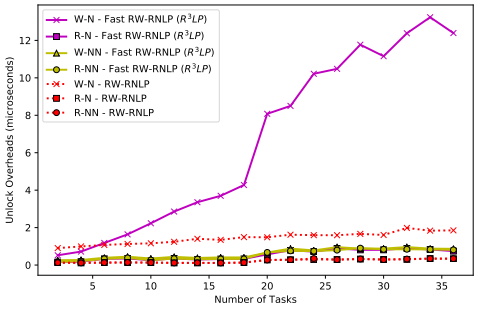



(c) Blocking.

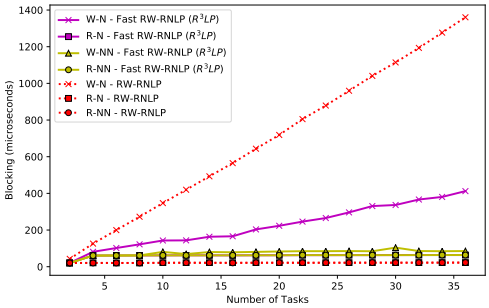
Figure 20: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

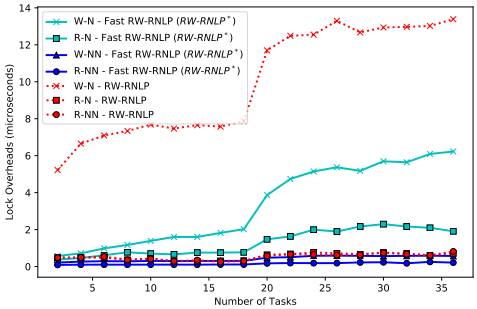



(b) Unlock overhead.

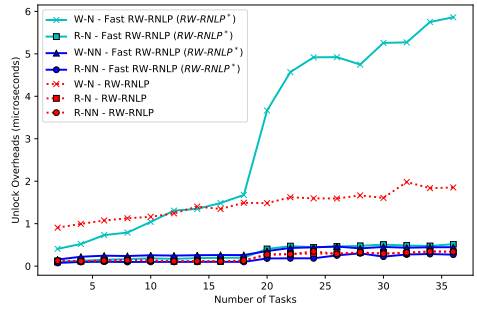



(c) Blocking.

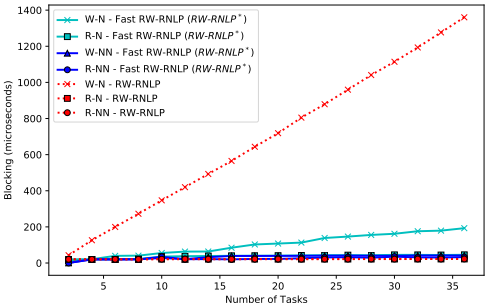
Figure 21: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

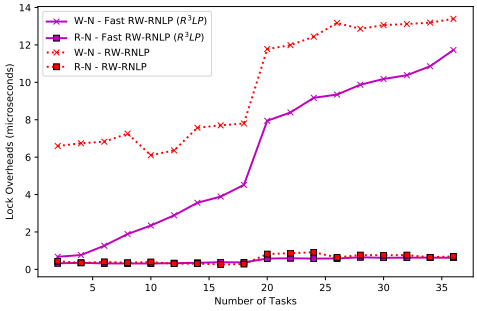



(b) Unlock overhead.

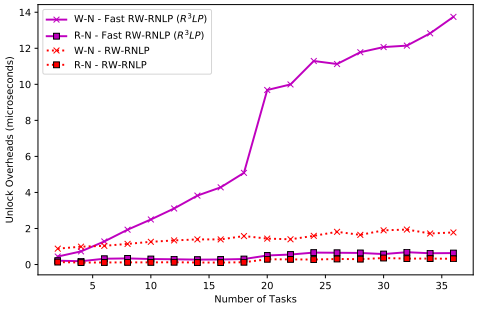



(c) Blocking.

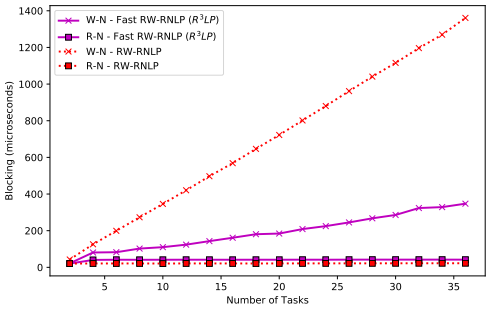
Figure 22: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

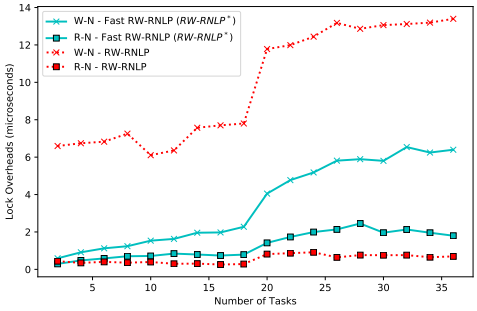



(b) Unlock overhead.

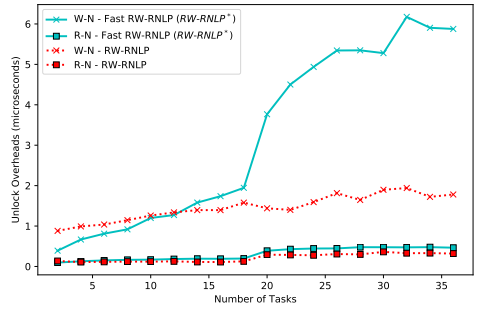



(c) Blocking.

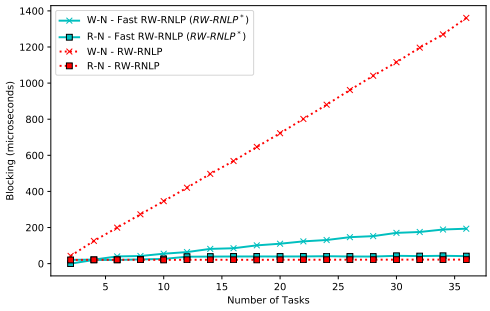
Figure 23: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

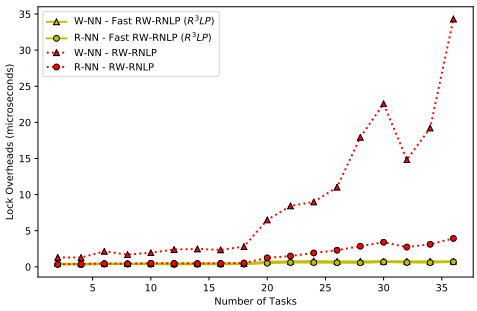



(b) Unlock overhead.

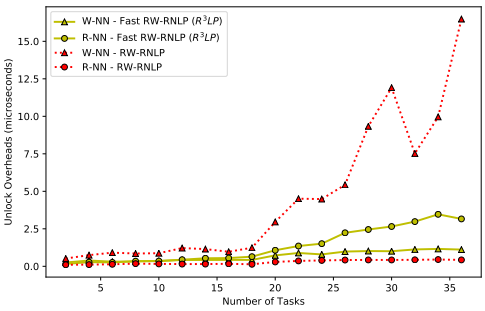



(c) Blocking.

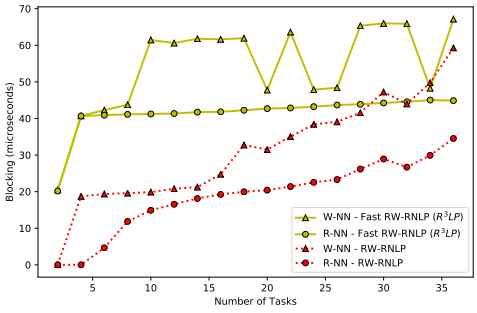
Figure 24: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

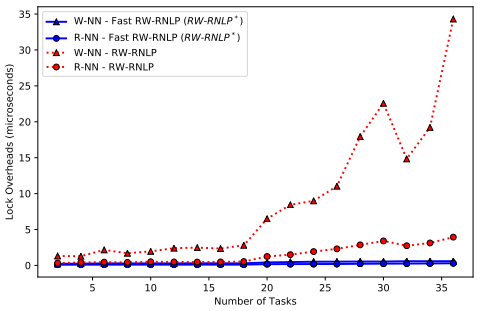



(b) Unlock overhead.

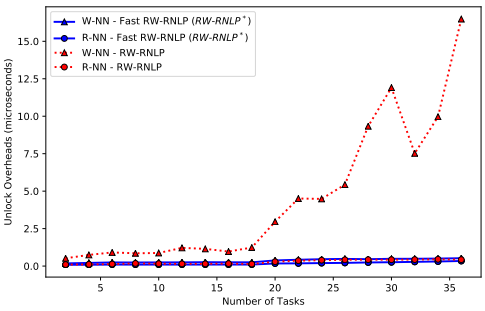



(c) Blocking.

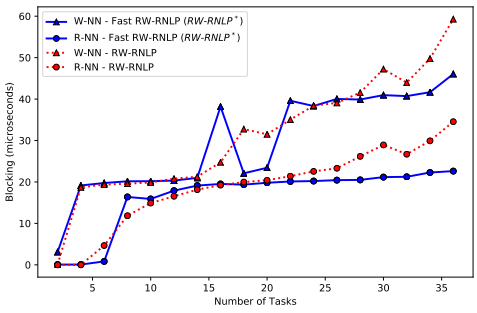
Figure 25: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

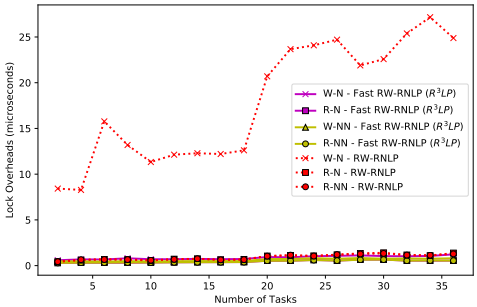



(b) Unlock overhead.

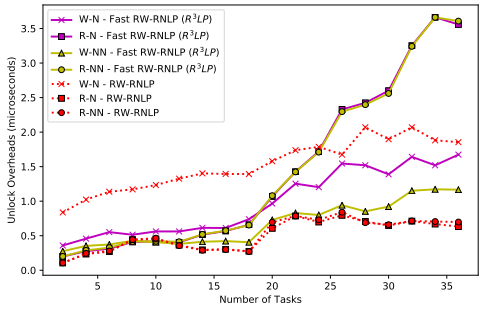



(c) Blocking.

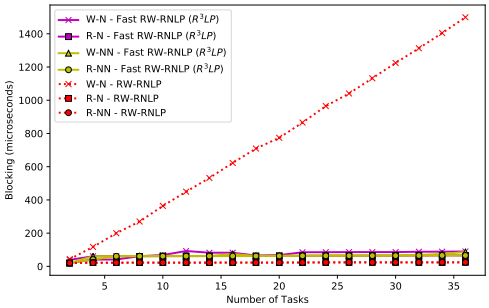
Figure 26: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

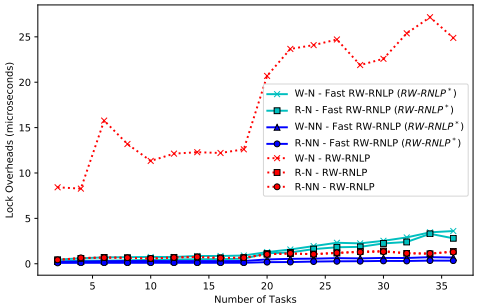



(b) Unlock overhead.

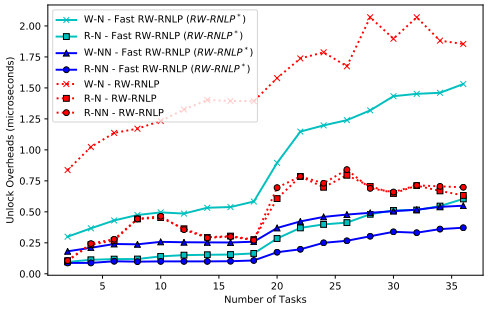



(c) Blocking.

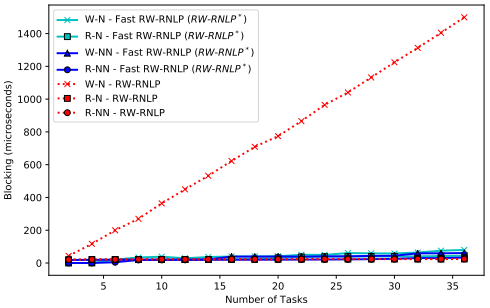
Figure 27: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

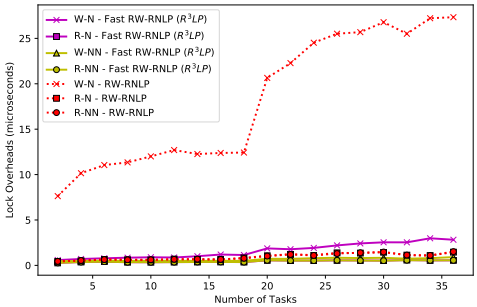



(b) Unlock overhead.

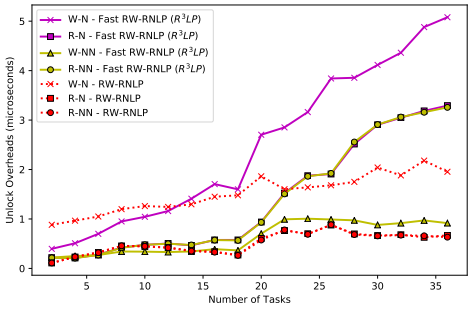



(c) Blocking.

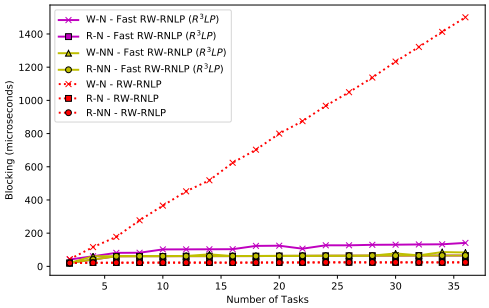
Figure 28: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

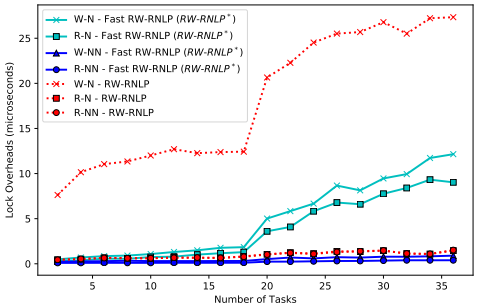



(b) Unlock overhead.

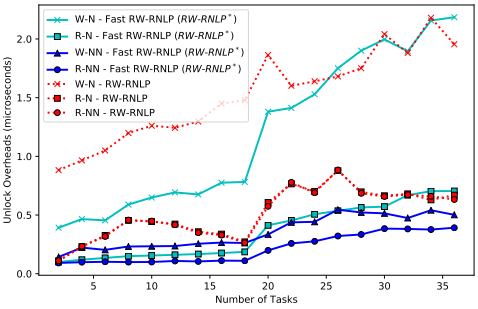



(c) Blocking.

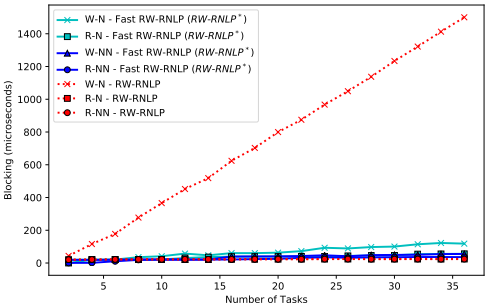
Figure 29: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

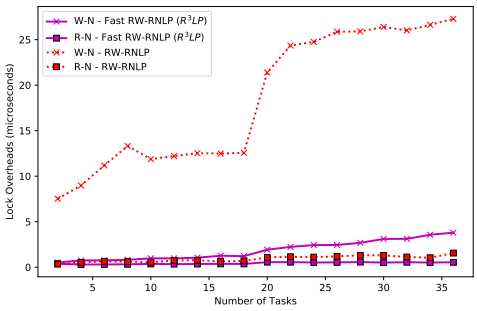



(b) Unlock overhead.

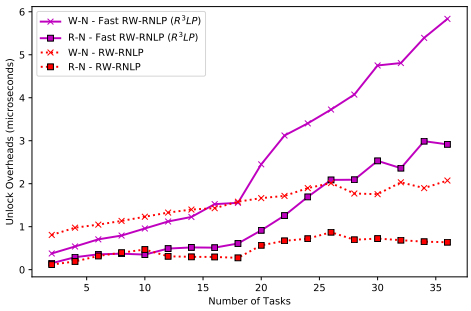



(c) Blocking.

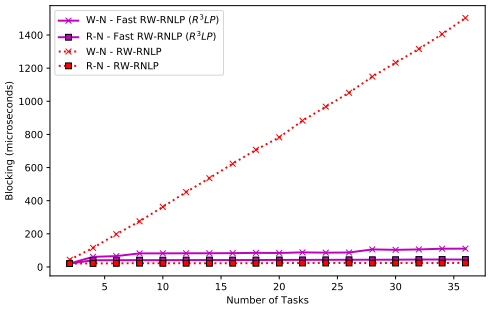
Figure 30: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

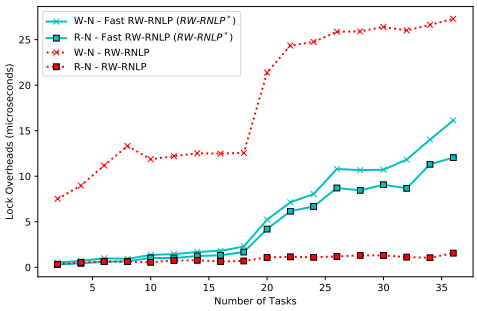



(b) Unlock overhead.

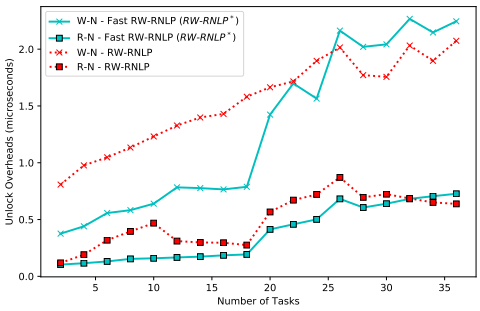



(c) Blocking.

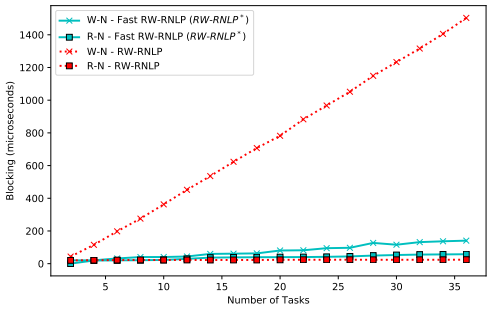
Figure 31: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

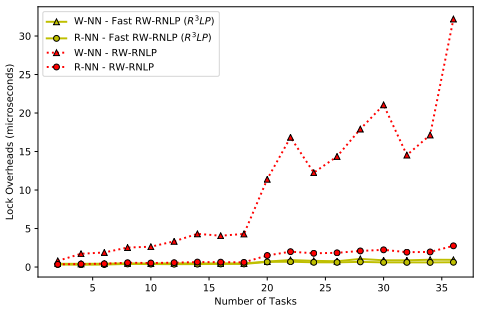



(b) Unlock overhead.

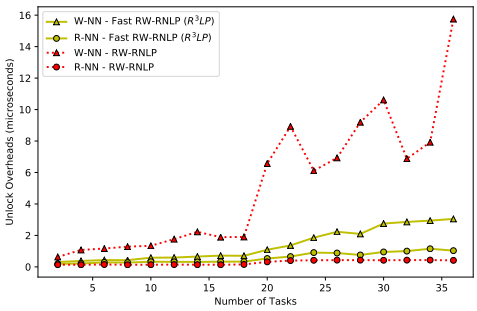



(c) Blocking.

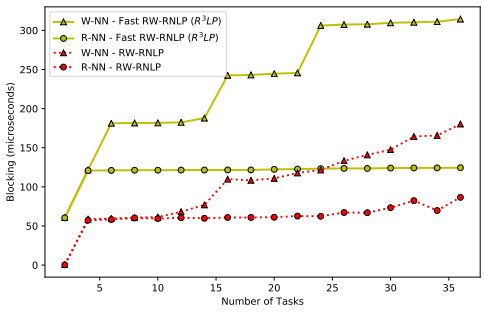
Figure 32: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

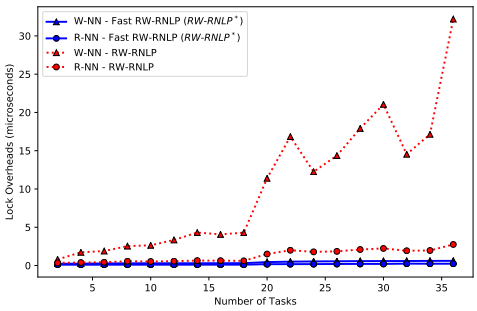



(b) Unlock overhead.

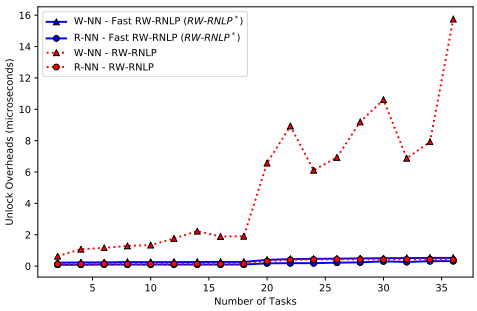



(c) Blocking.

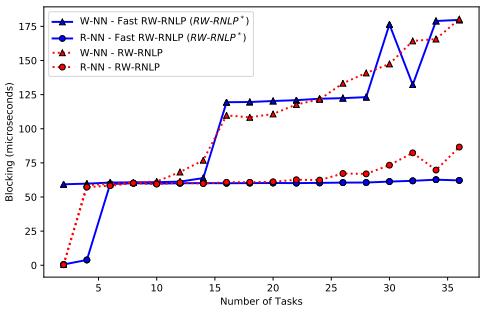
Figure 33: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

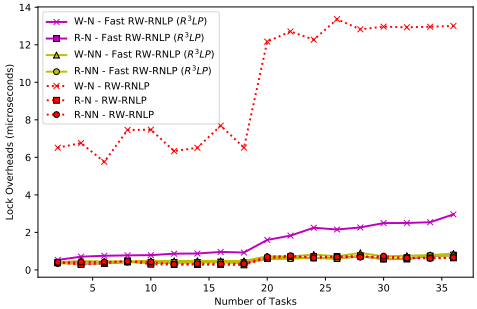



(b) Unlock overhead.

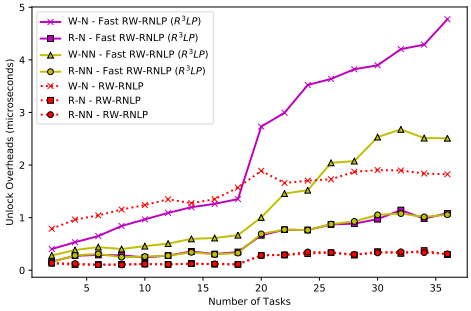



(c) Blocking.

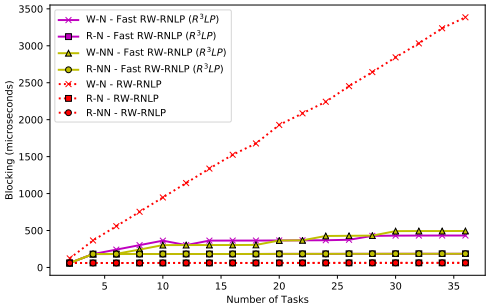
Figure 34: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

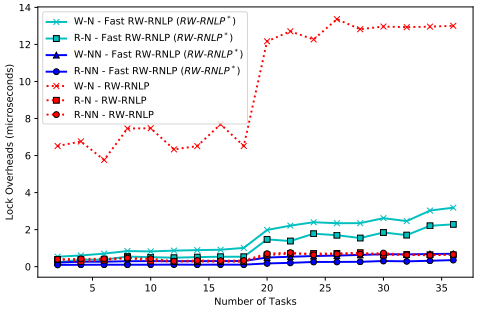



(b) Unlock overhead.

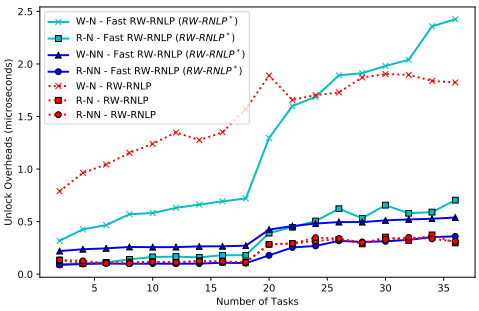



(c) Blocking.

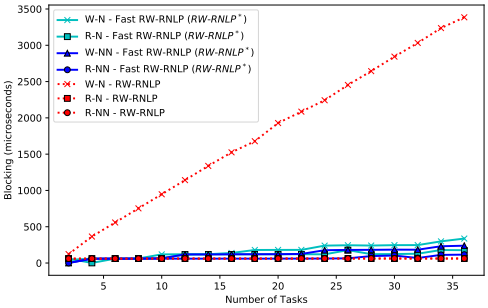
Figure 35: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

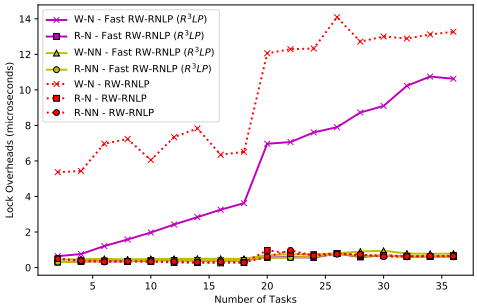



(b) Unlock overhead.

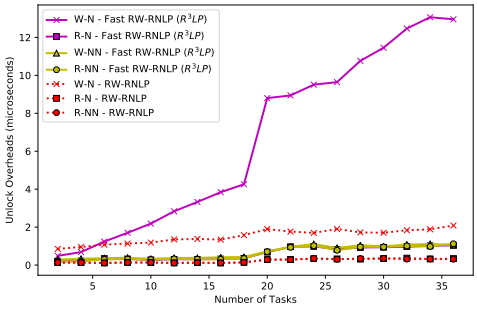



(c) Blocking.

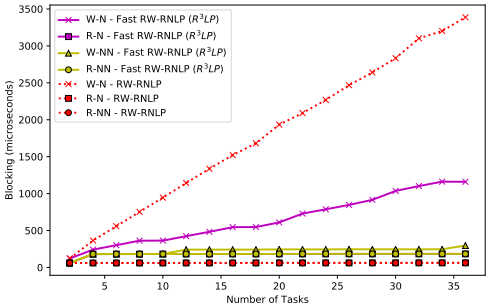
Figure 36: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

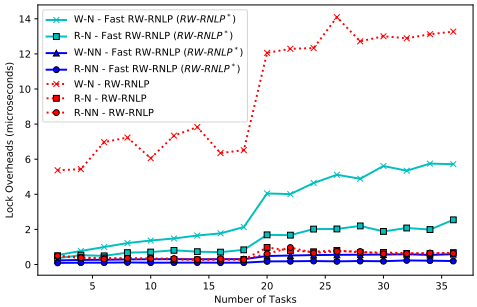



(b) Unlock overhead.

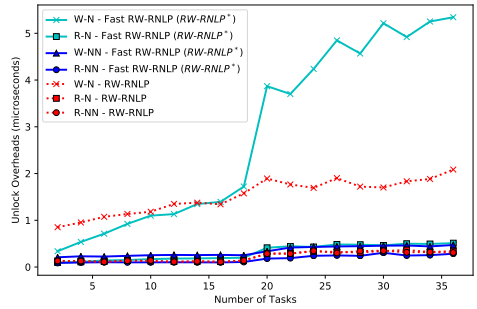



(c) Blocking.

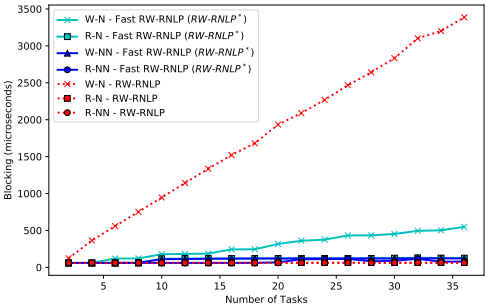
Figure 37: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

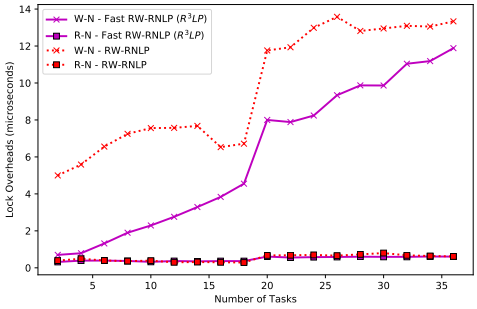



(b) Unlock overhead.

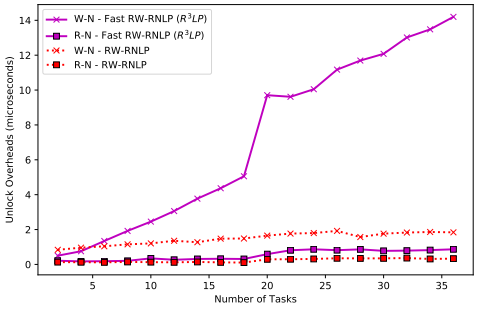



(c) Blocking.

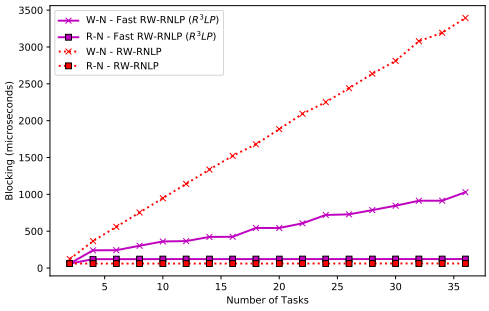
Figure 38: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

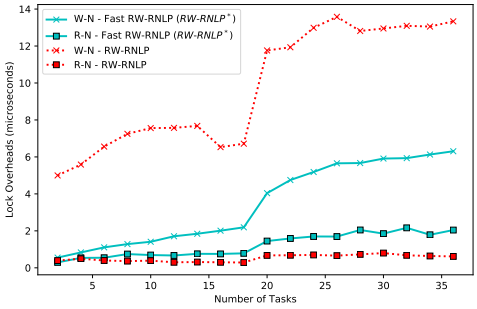



(b) Unlock overhead.

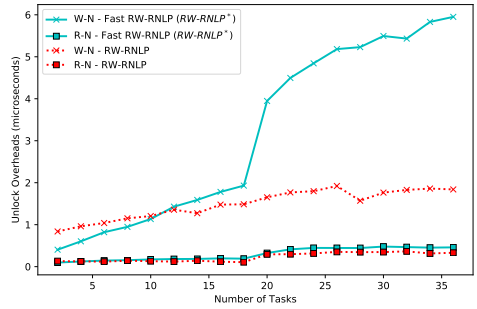



(c) Blocking.

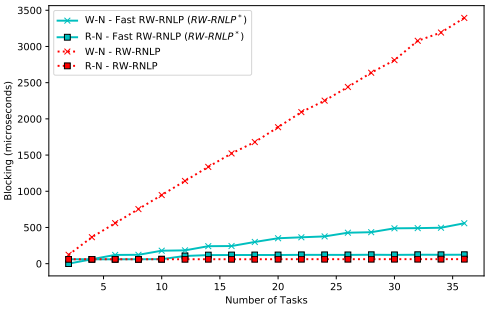
Figure 39: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

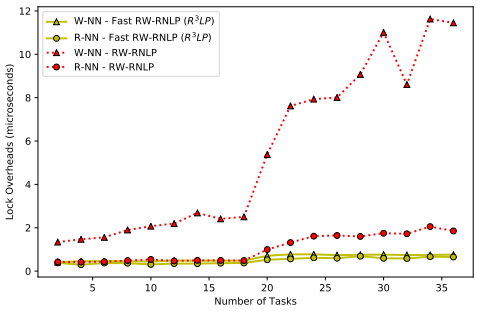



(b) Unlock overhead.

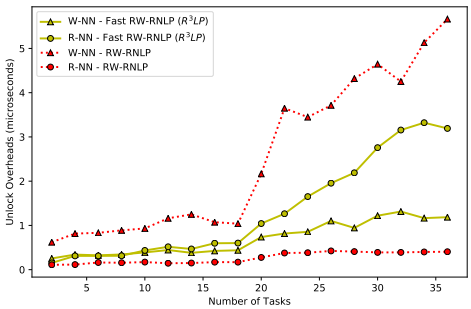



(c) Blocking.

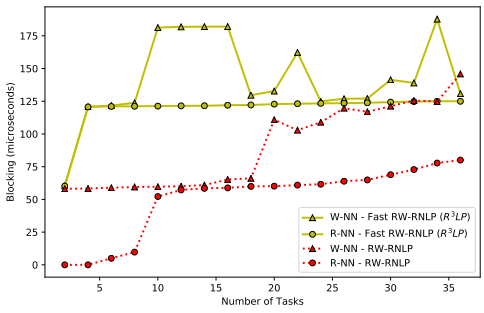
Figure 40: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

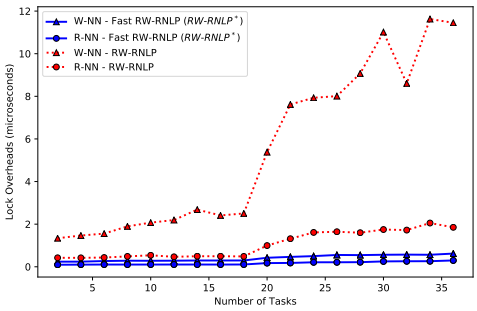



(b) Unlock overhead.

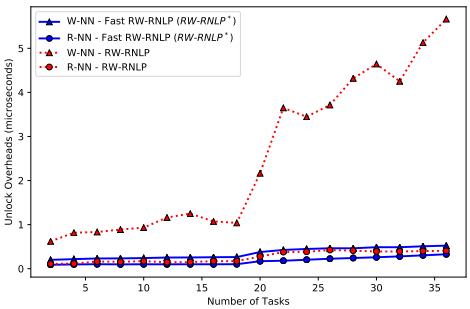



(c) Blocking.

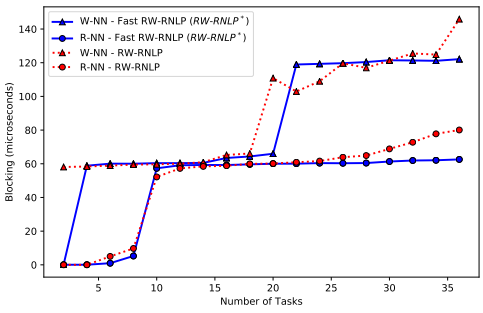
Figure 41: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

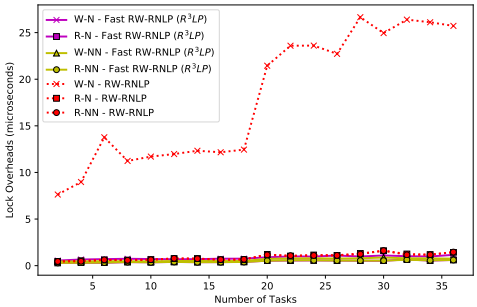



(b) Unlock overhead.

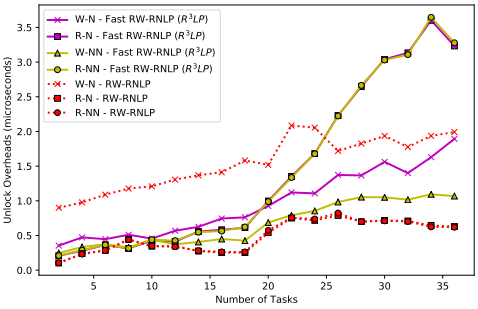



(c) Blocking.

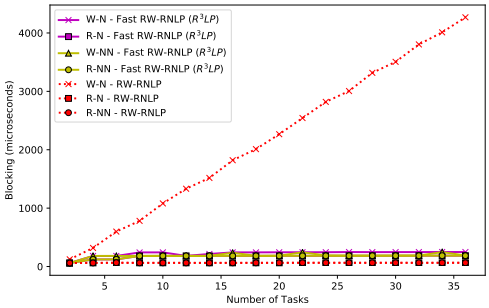
Figure 42: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

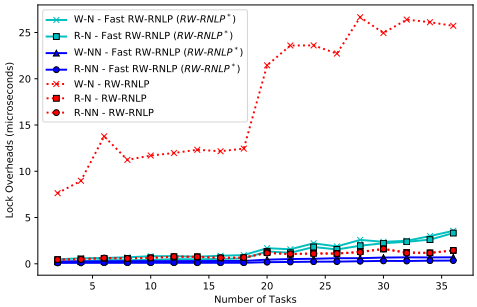



(b) Unlock overhead.

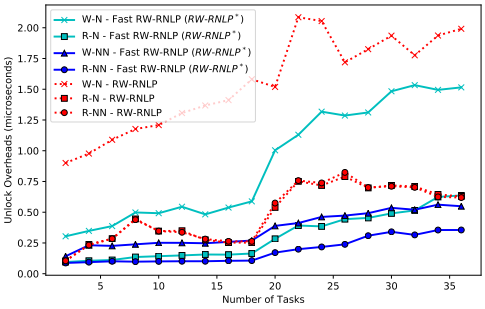



(c) Blocking.

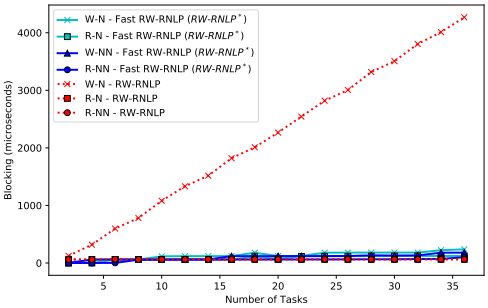
Figure 43: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

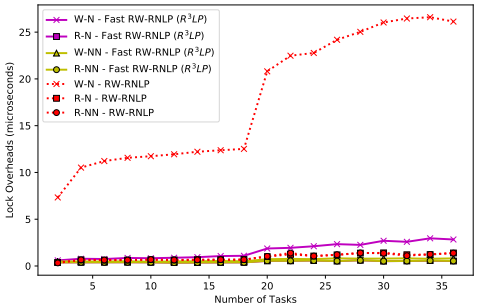



(b) Unlock overhead.

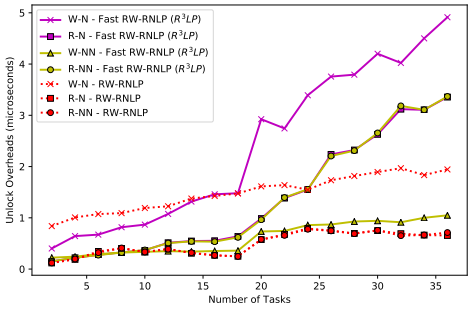



(c) Blocking.

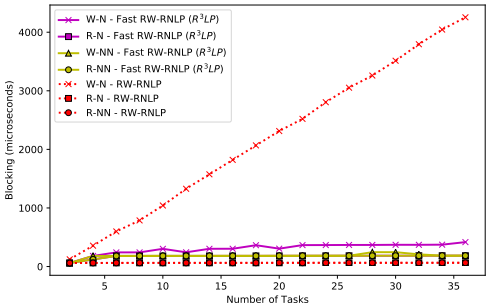
Figure 44: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

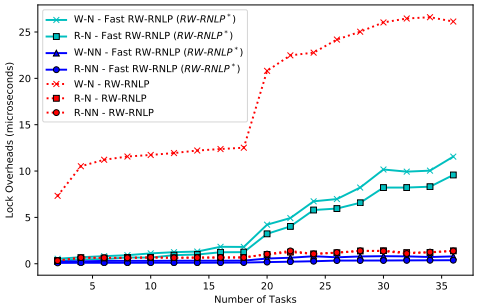



(b) Unlock overhead.

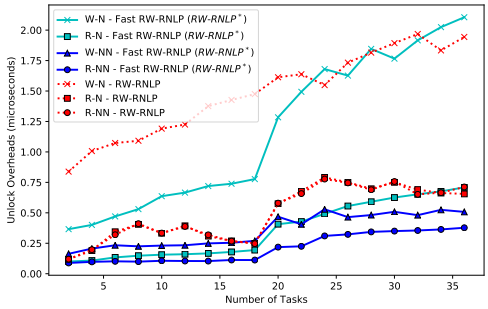



(c) Blocking.

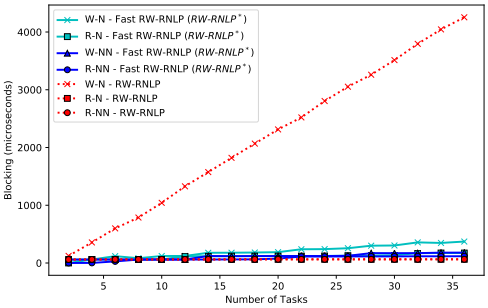
Figure 45: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

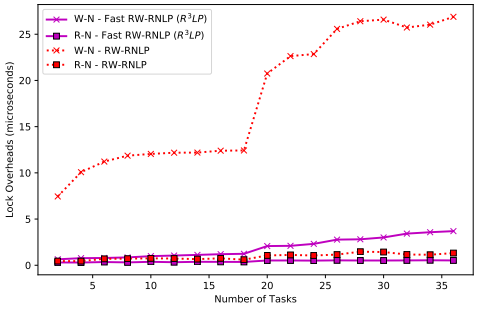



(b) Unlock overhead.

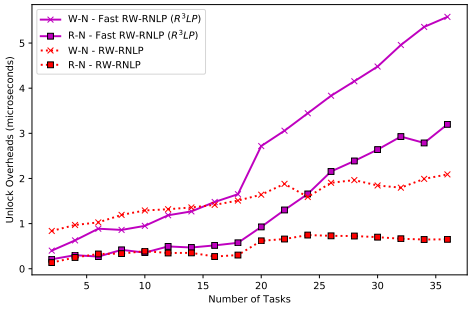



(c) Blocking.

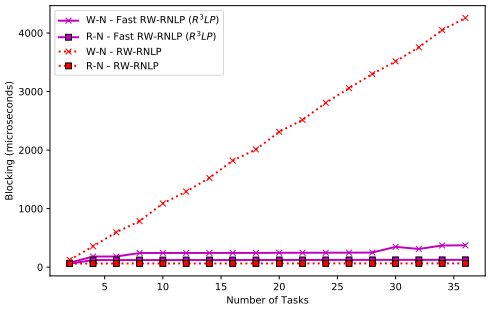
Figure 46: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

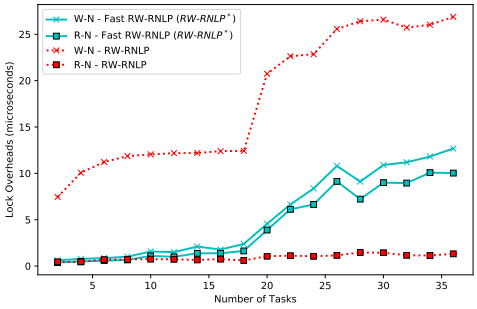



(b) Unlock overhead.

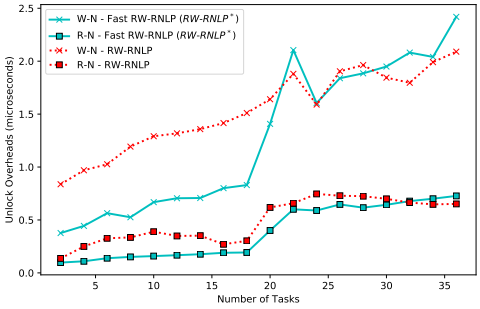



(c) Blocking.

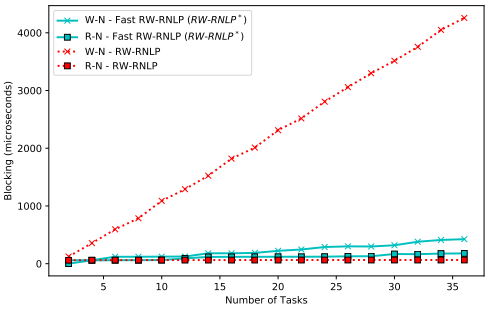
Figure 47: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

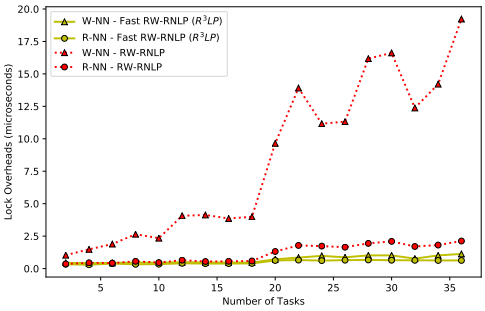



(b) Unlock overhead.

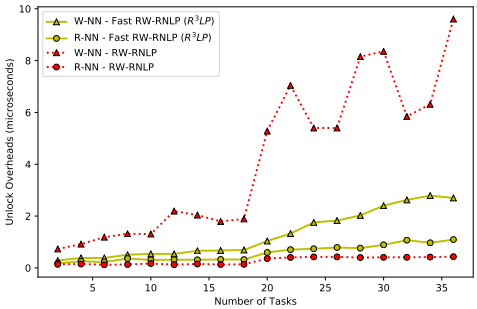



(c) Blocking.

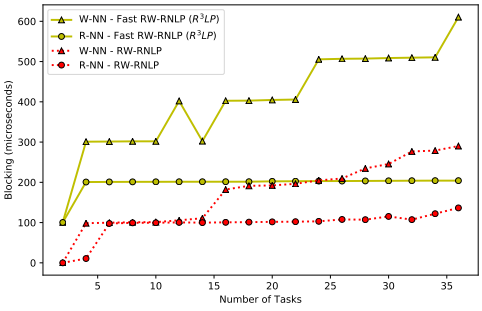
Figure 48: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

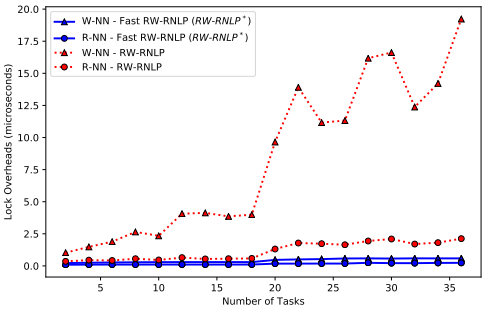



(b) Unlock overhead.

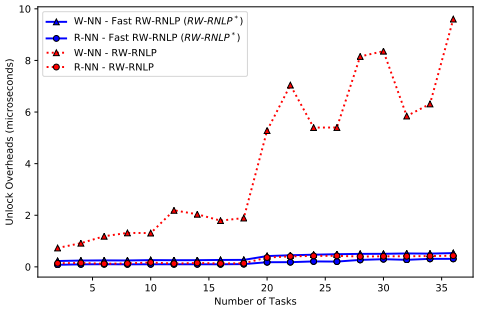



(c) Blocking.

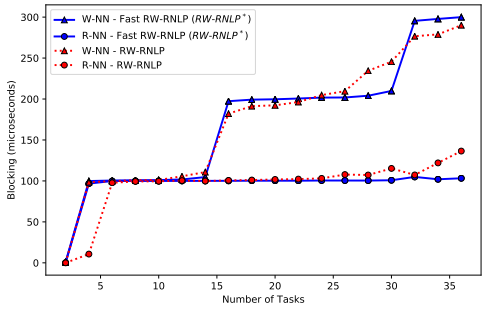
Figure 49: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

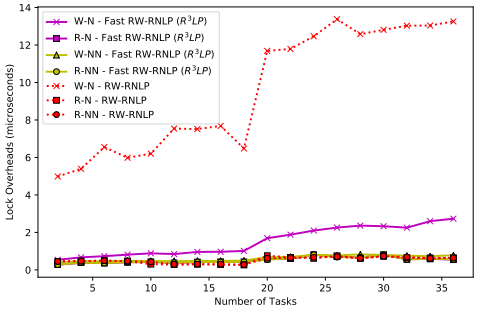



(b) Unlock overhead.

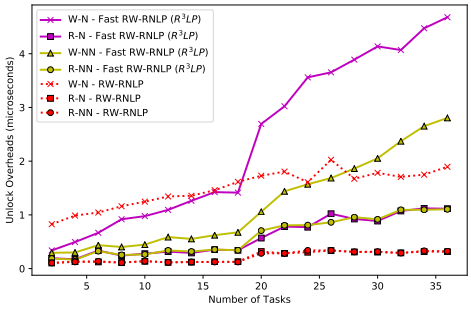



(c) Blocking.

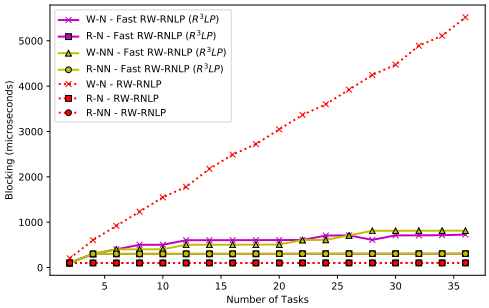
Figure 50: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

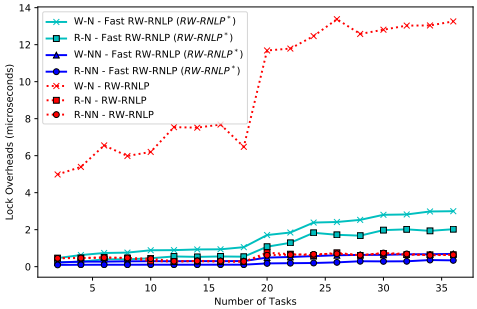



(b) Unlock overhead.

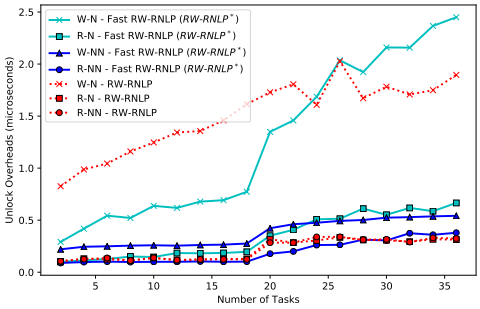



(c) Blocking.

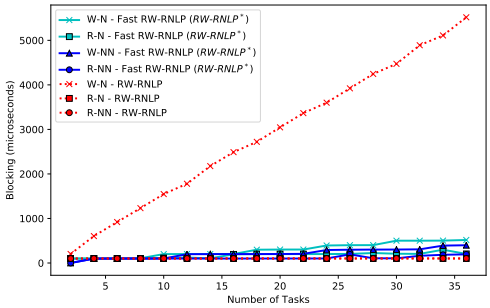
Figure 51: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

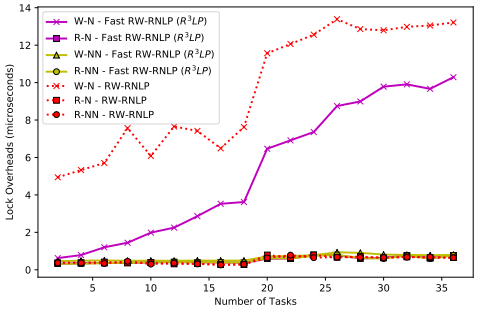



(b) Unlock overhead.

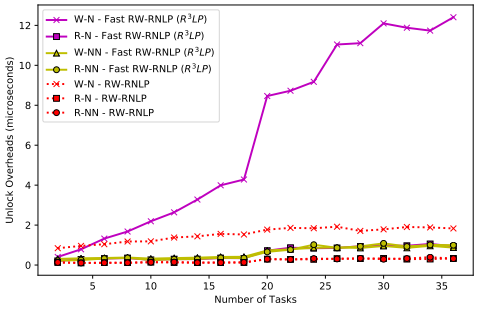



(c) Blocking.

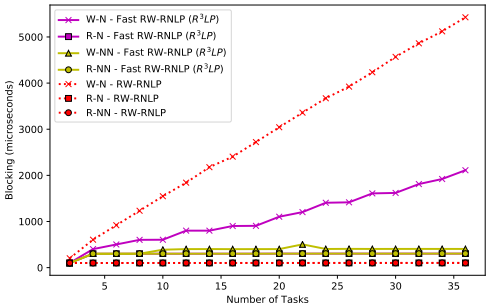
Figure 52: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

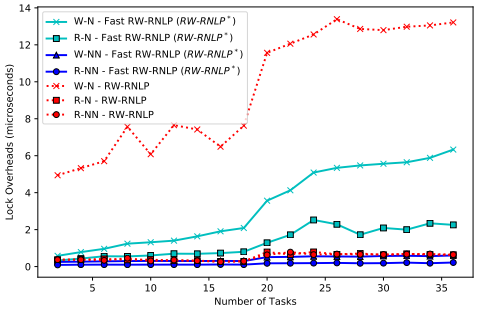



(b) Unlock overhead.

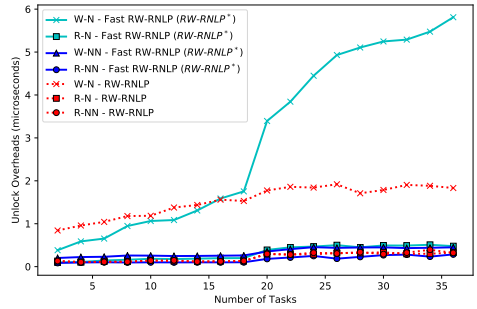



(c) Blocking.

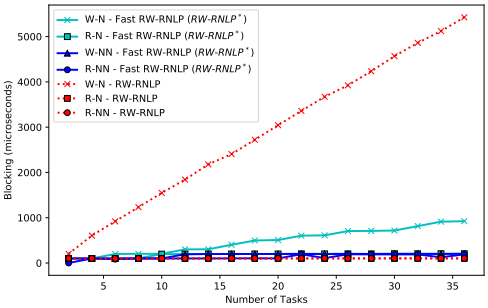
Figure 53: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

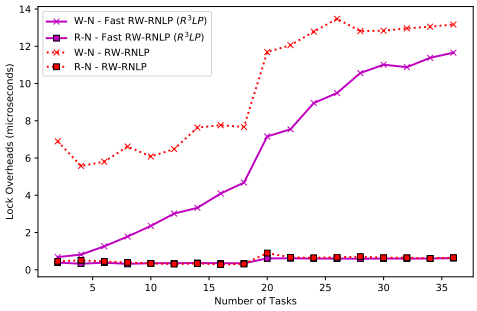



(b) Unlock overhead.

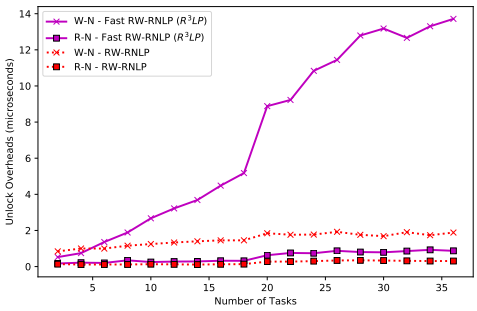



(c) Blocking.

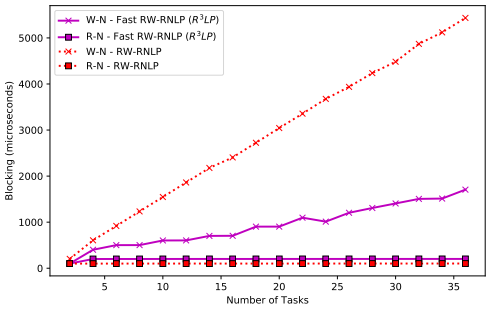
Figure 54: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

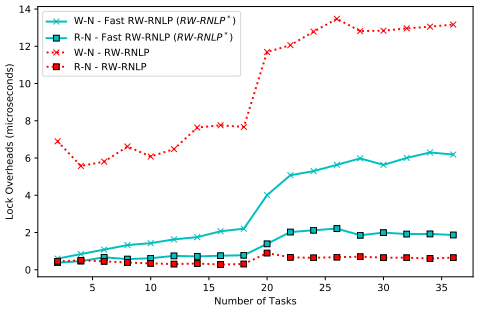



(b) Unlock overhead.

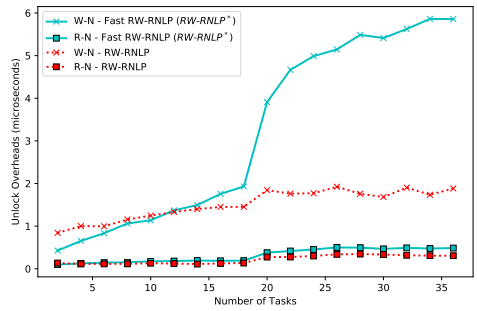



(c) Blocking.

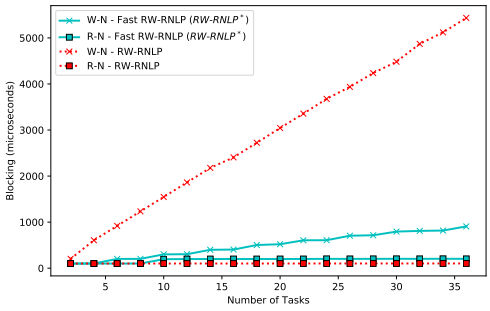
Figure 55: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

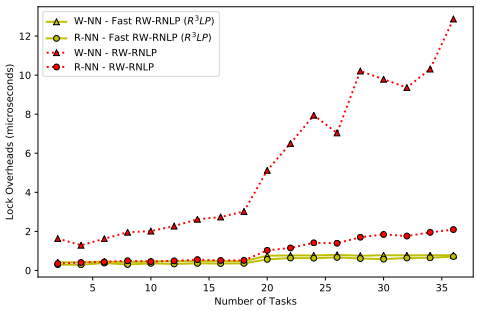



(b) Unlock overhead.

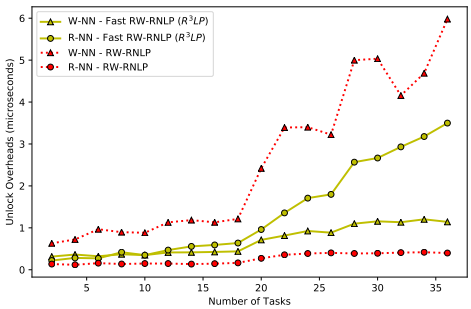



(c) Blocking.

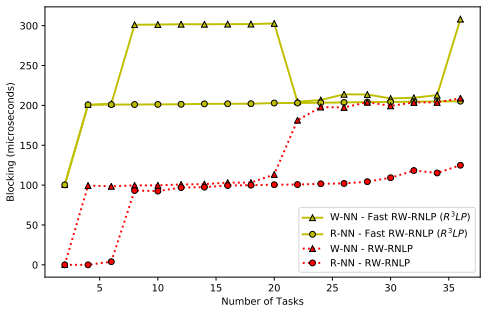
Figure 56: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

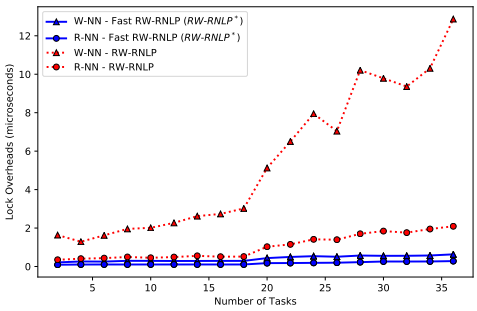



(b) Unlock overhead.

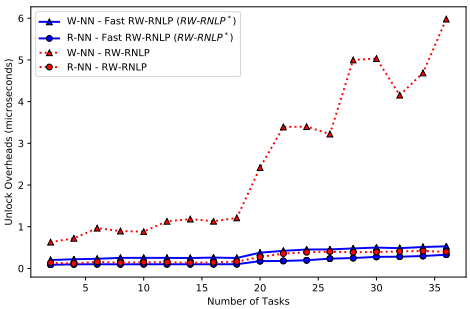



(c) Blocking.

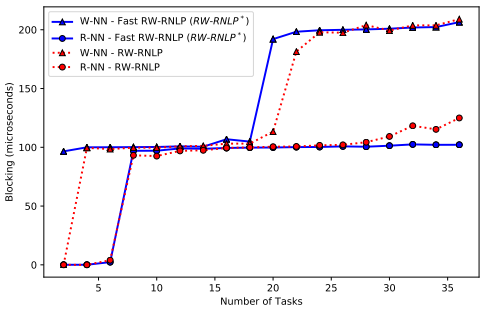
Figure 57: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

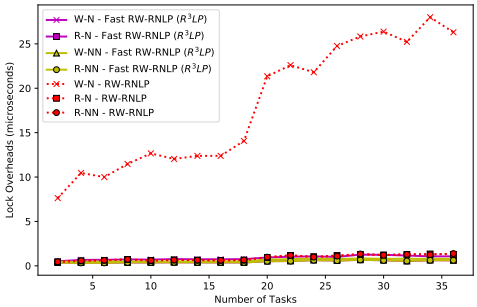



(b) Unlock overhead.

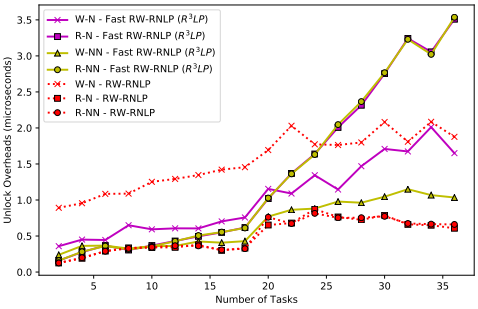



(c) Blocking.

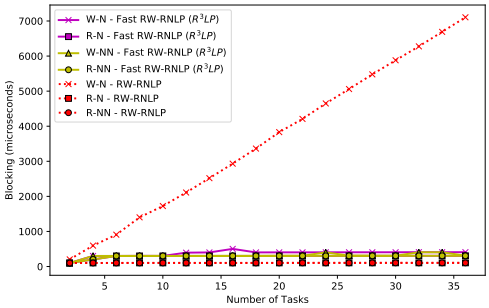
Figure 58: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

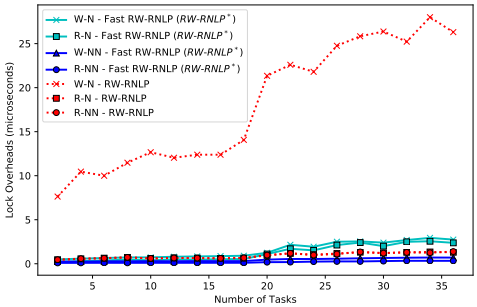



(b) Unlock overhead.

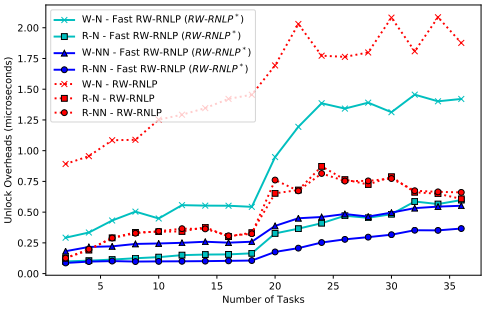



(c) Blocking.

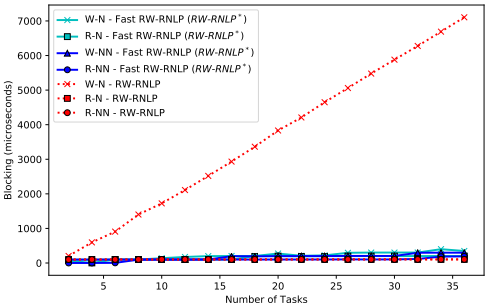
Figure 59: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

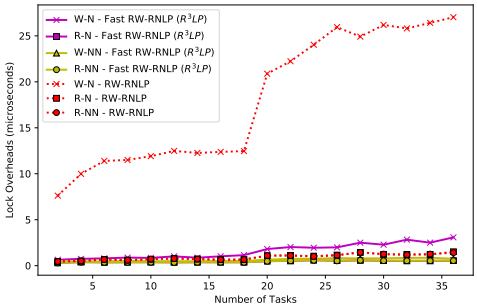



(b) Unlock overhead.

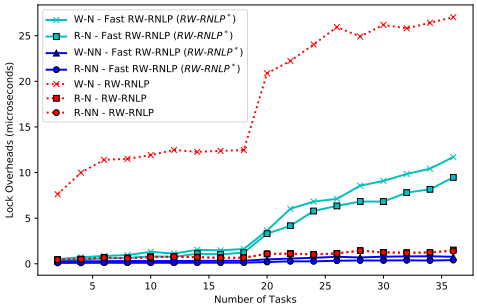



(c) Blocking.

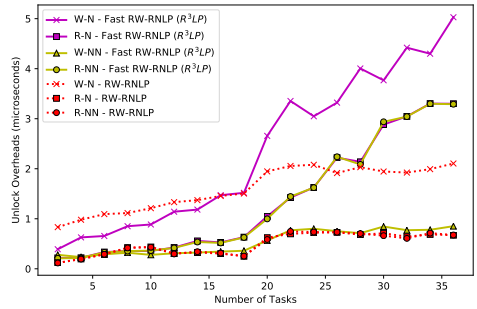
Figure 60: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.



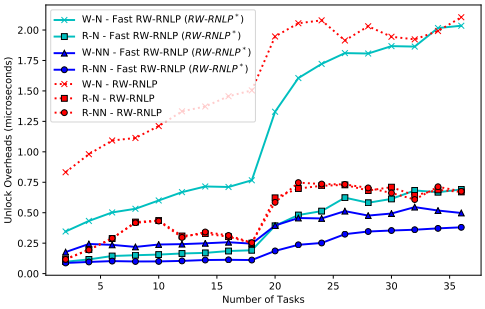

(a) Lock overhead.



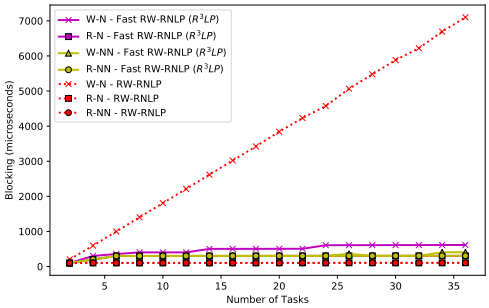

(a) Lock overhead.



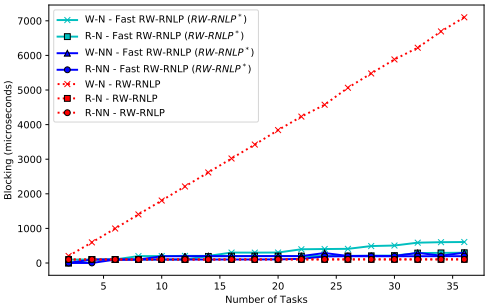

(b) Unlock overhead.




(b) Unlock overhead.



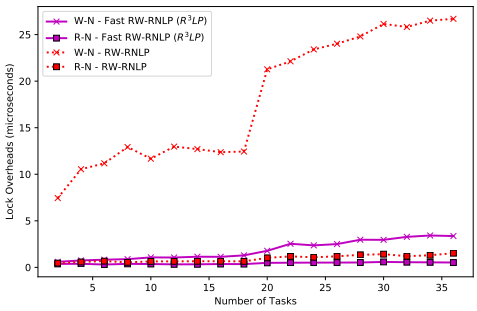

(c) Blocking.



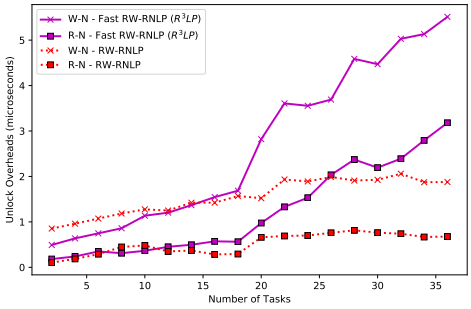

(c) Blocking.

Figure 61: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.

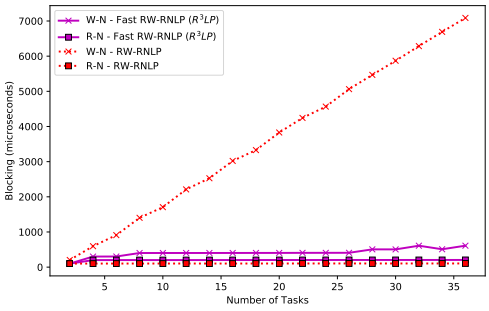
Figure 62: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

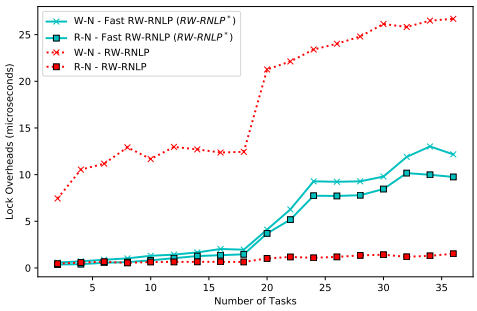



(b) Unlock overhead.

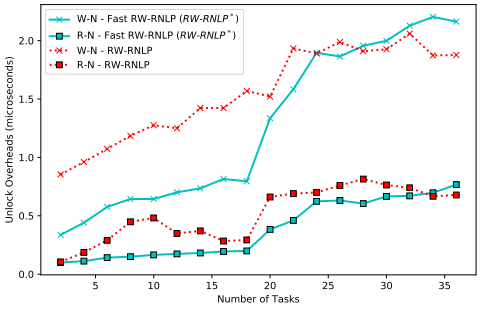



(c) Blocking.

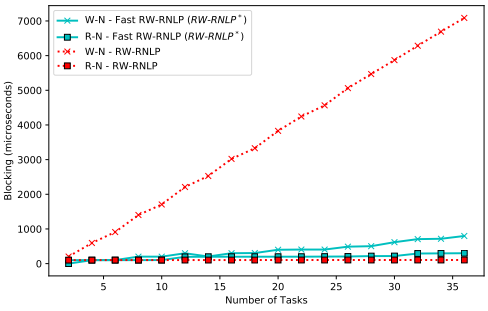
Figure 63: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

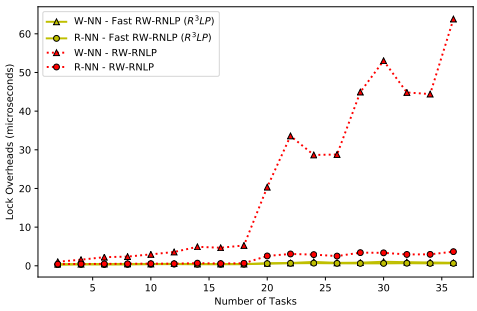



(b) Unlock overhead.

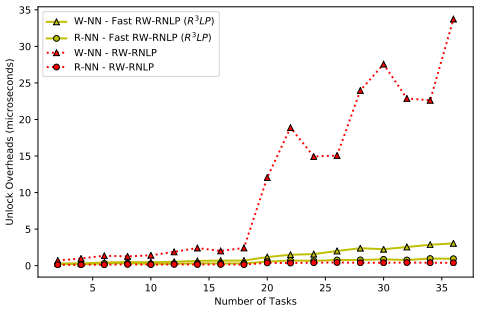



(c) Blocking.

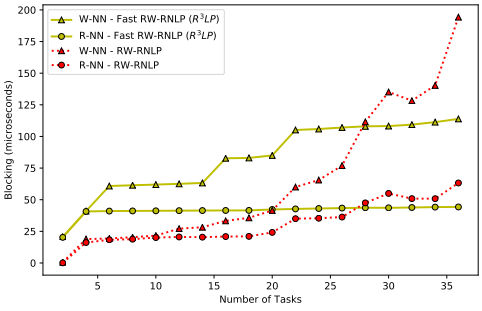
Figure 64: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

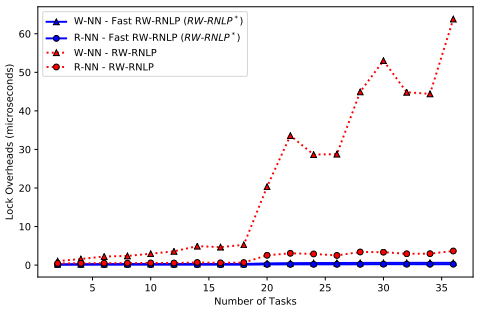



(b) Unlock overhead.

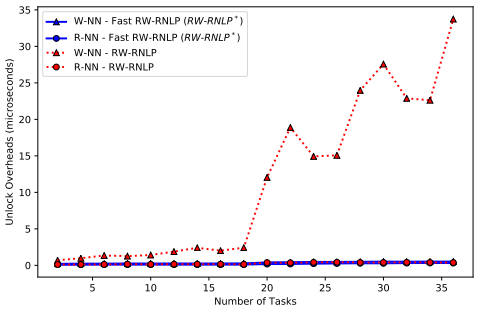



(c) Blocking.

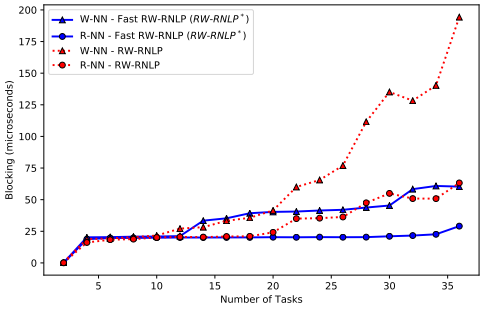
Figure 65: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

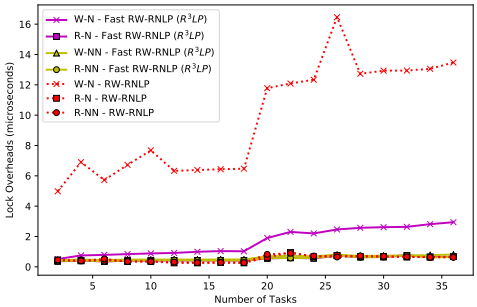



(b) Unlock overhead.

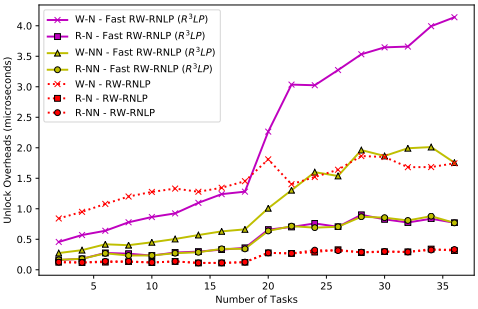



(c) Blocking.

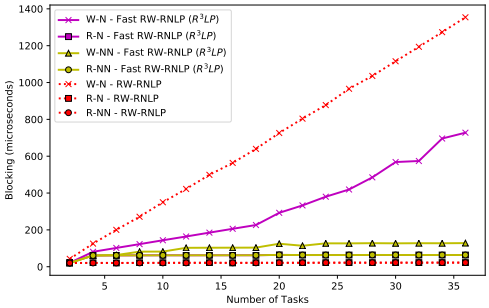
Figure 66: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

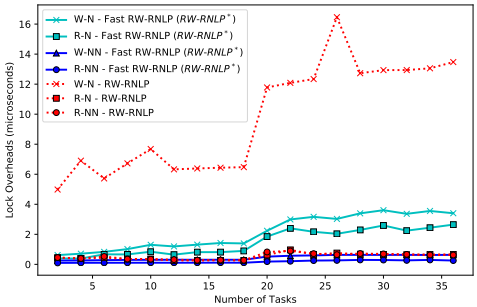



(b) Unlock overhead.

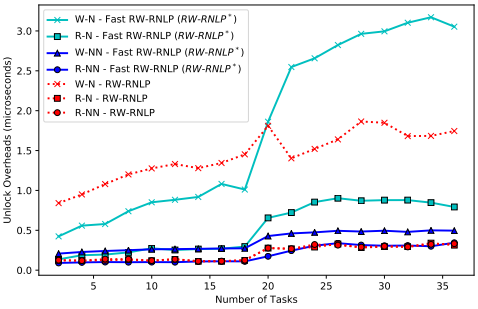



(c) Blocking.

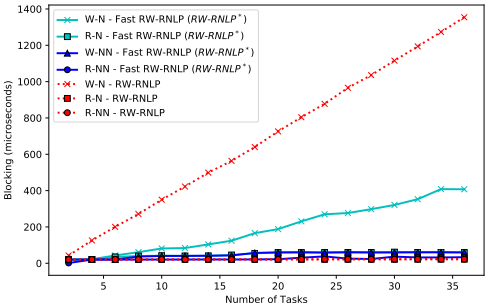
Figure 67: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

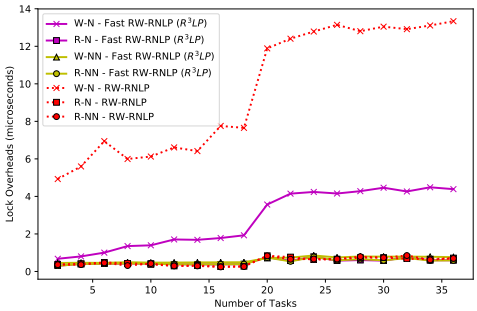



(b) Unlock overhead.

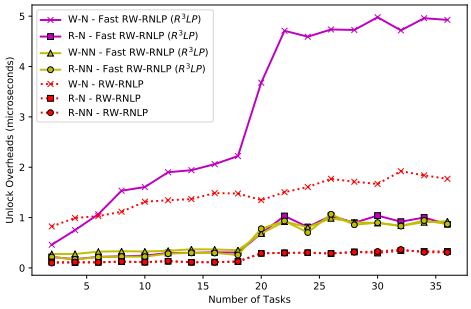



(c) Blocking.

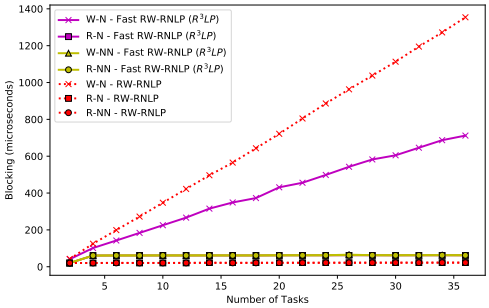
Figure 68: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

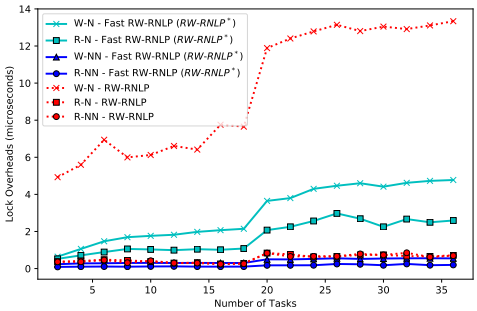



(b) Unlock overhead.

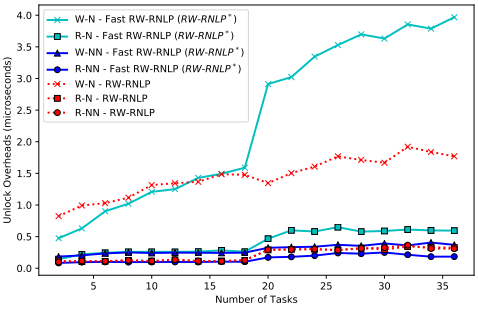



(c) Blocking.

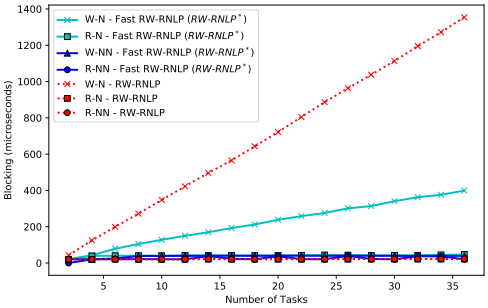
Figure 69: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

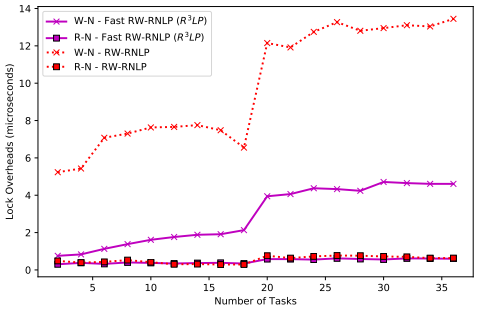



(b) Unlock overhead.

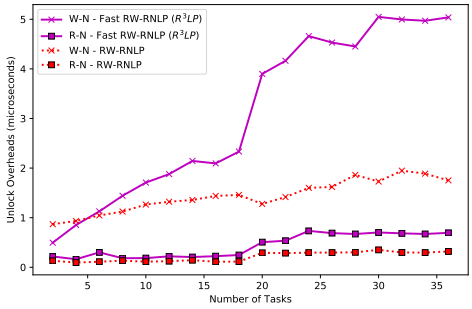



(c) Blocking.

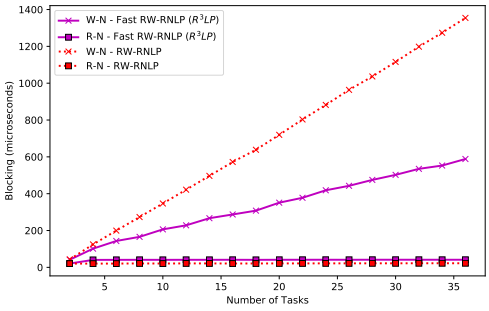
Figure 70: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

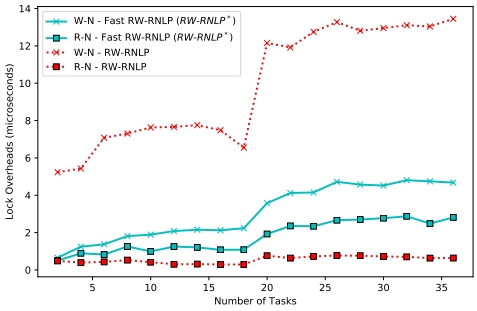



(b) Unlock overhead.

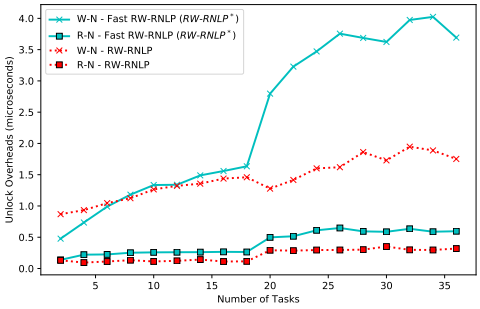



(c) Blocking.

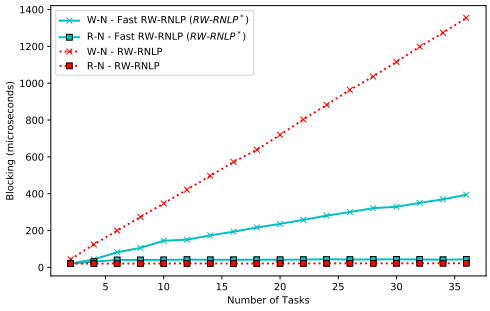
Figure 71: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

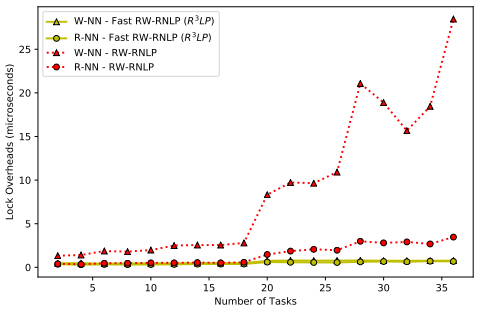



(b) Unlock overhead.

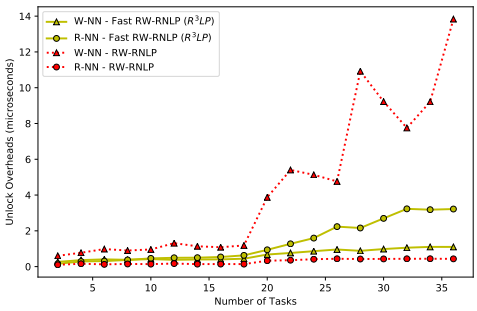



(c) Blocking.

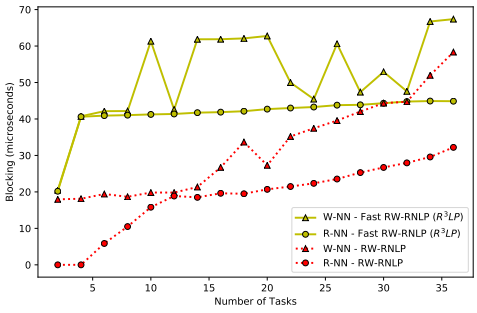
Figure 72: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

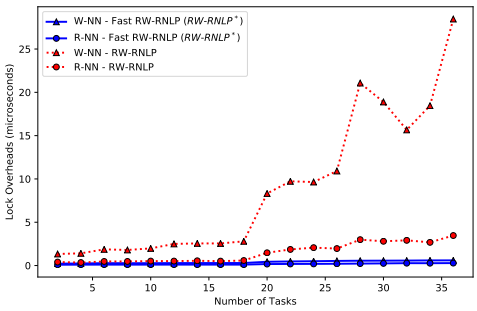



(b) Unlock overhead.

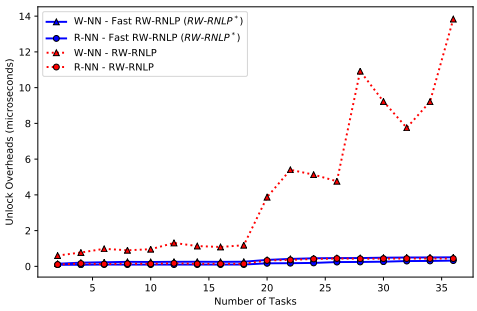



(c) Blocking.

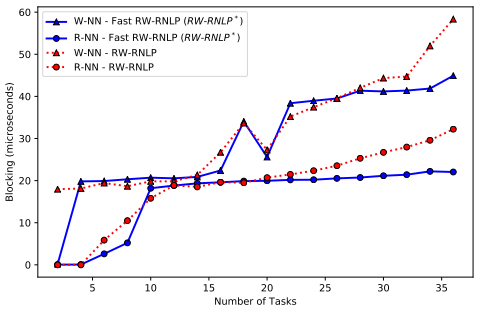
Figure 73: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

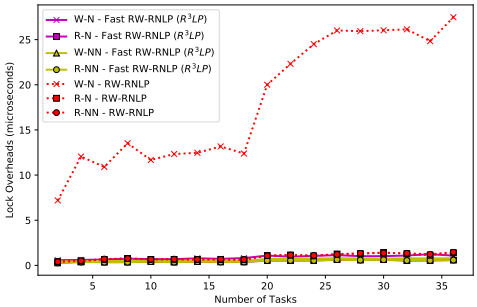



(b) Unlock overhead.

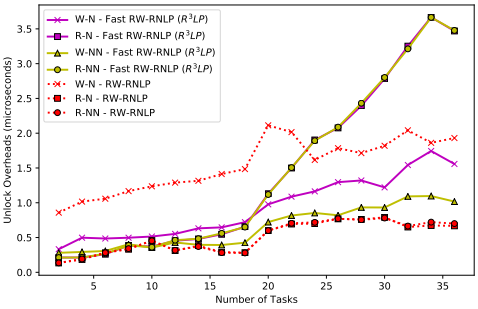



(c) Blocking.

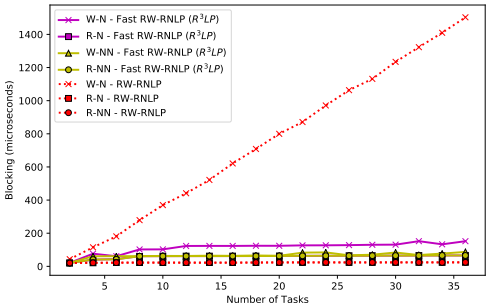
Figure 74: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

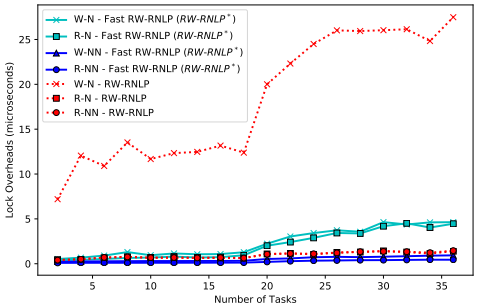



(b) Unlock overhead.

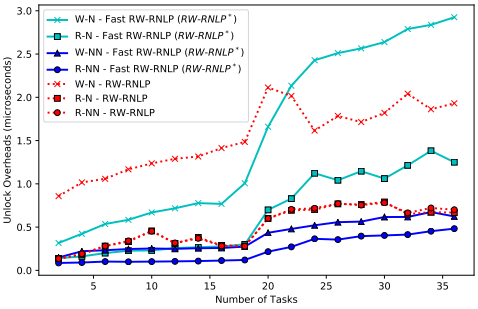



(c) Blocking.

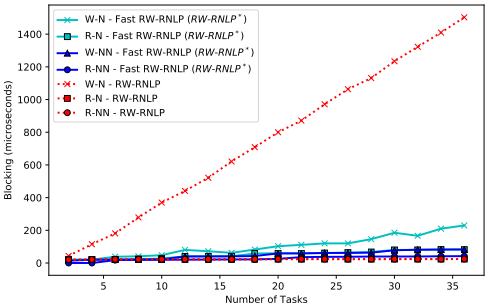
Figure 75: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

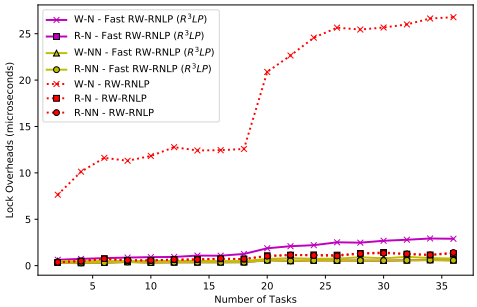



(b) Unlock overhead.

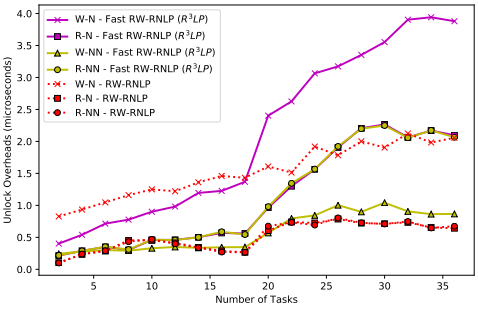



(c) Blocking.

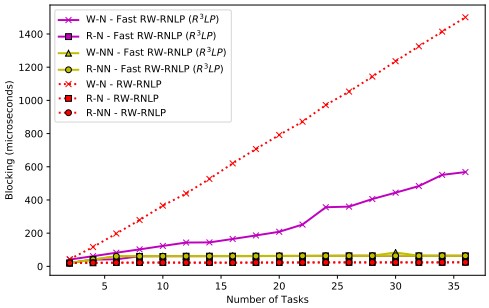
Figure 76: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

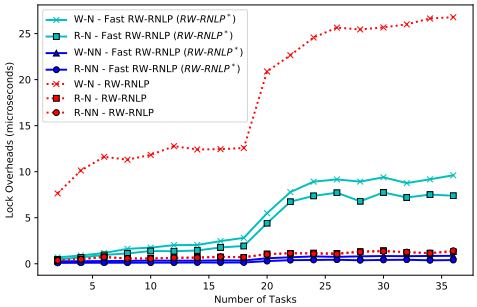



(b) Unlock overhead.

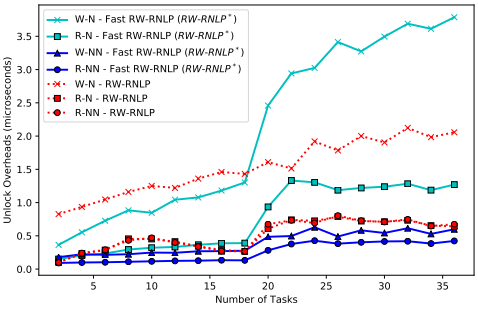



(c) Blocking.

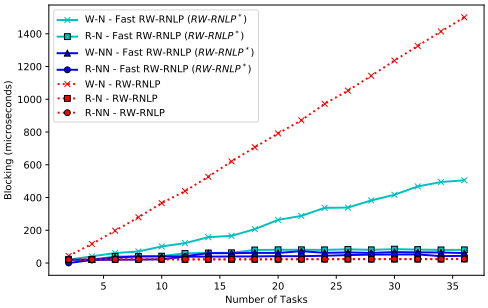
Figure 77: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

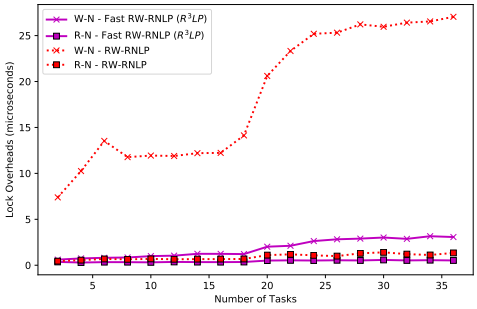



(b) Unlock overhead.

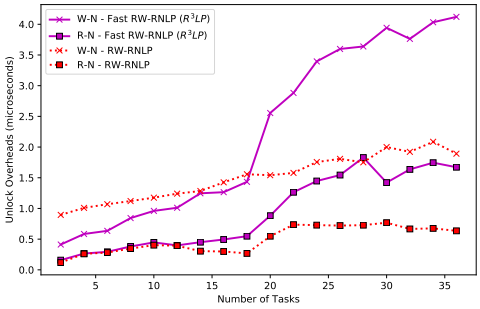



(c) Blocking.

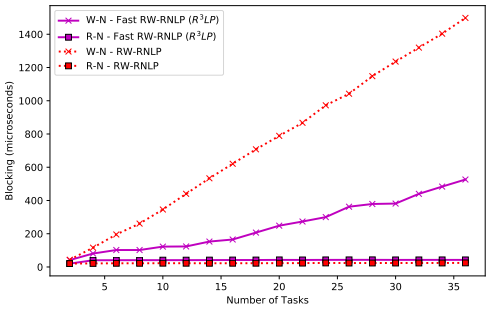
Figure 78: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

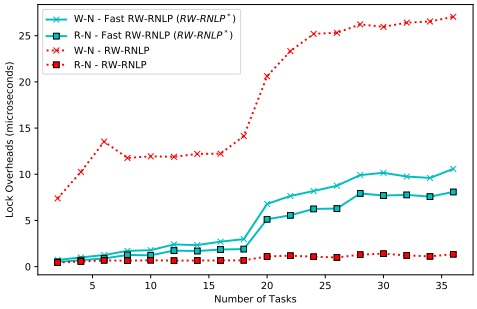



(b) Unlock overhead.

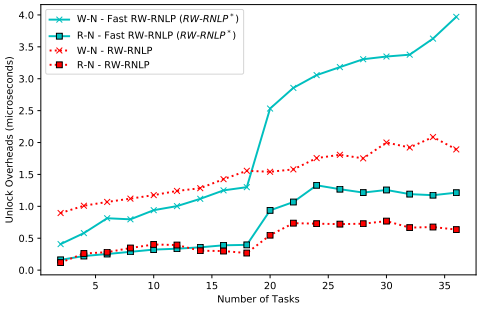



(c) Blocking.

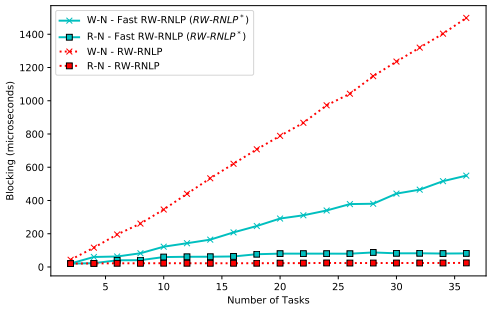
Figure 79: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

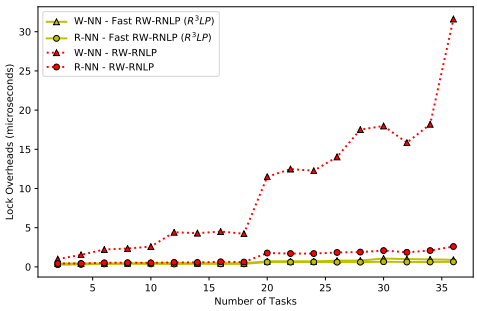



(b) Unlock overhead.

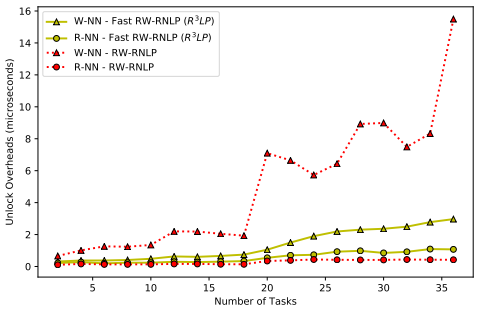



(c) Blocking.

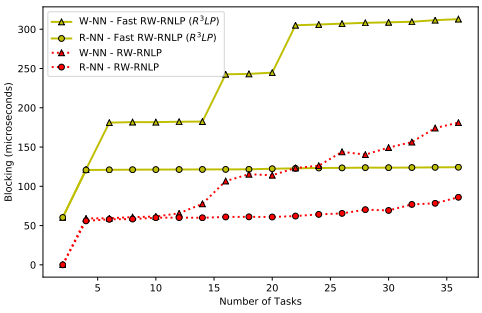
Figure 80: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

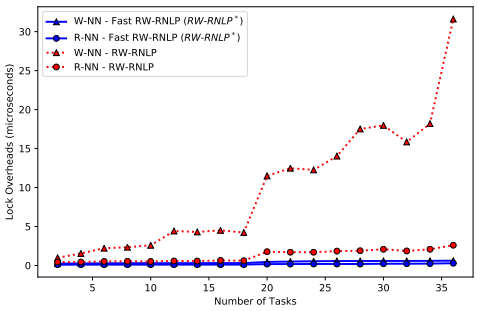



(b) Unlock overhead.

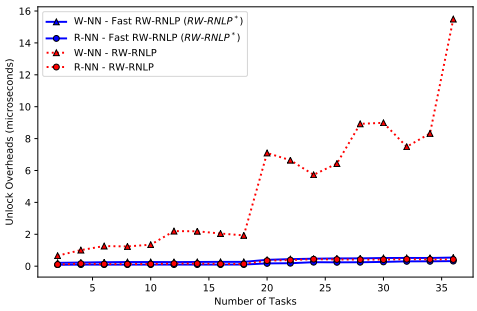



(c) Blocking.

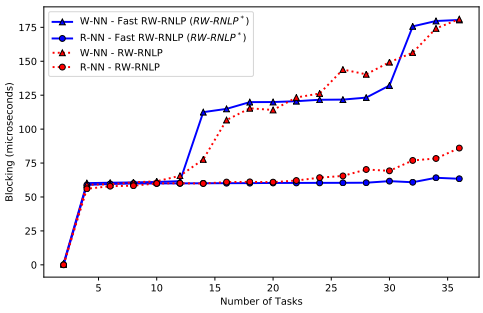
Figure 81: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

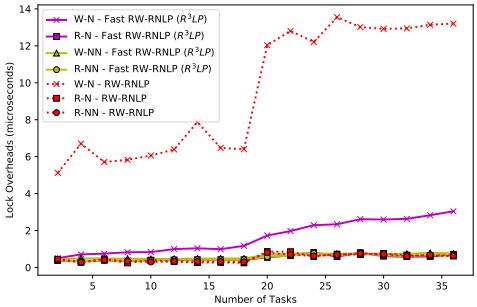



(b) Unlock overhead.

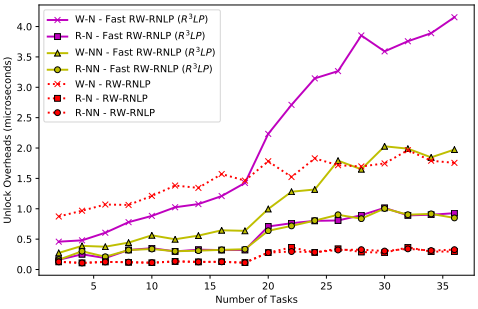



(c) Blocking.

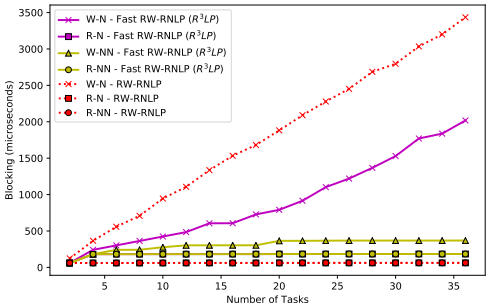
Figure 82: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

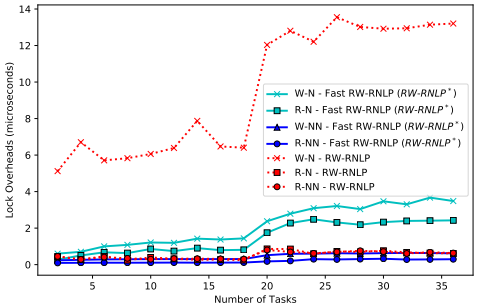



(b) Unlock overhead.

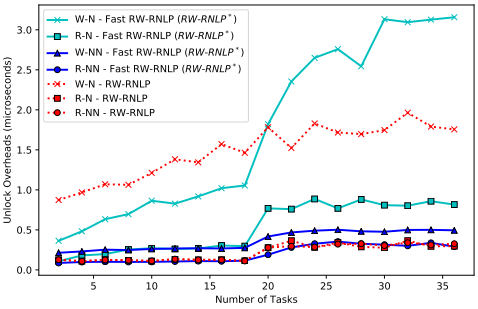



(c) Blocking.

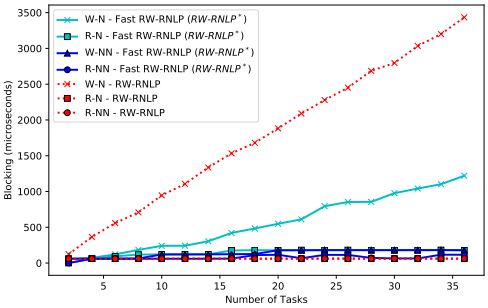
Figure 83: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

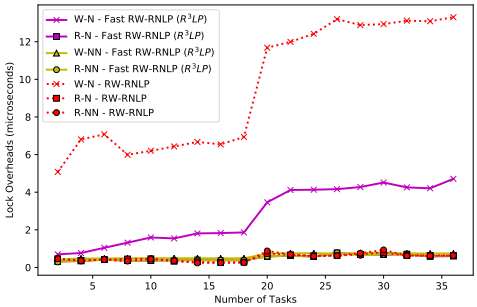



(b) Unlock overhead.

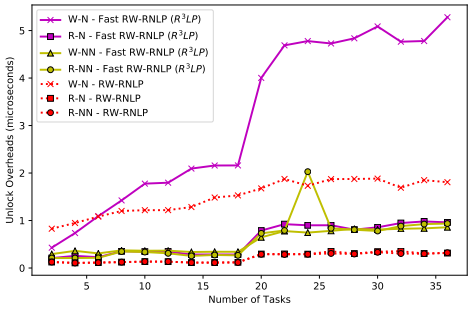



(c) Blocking.

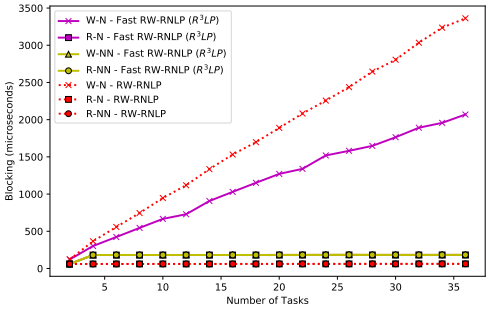
Figure 84: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

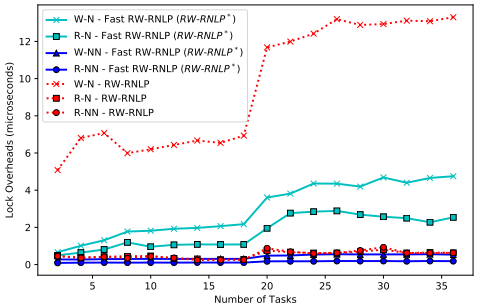



(b) Unlock overhead.

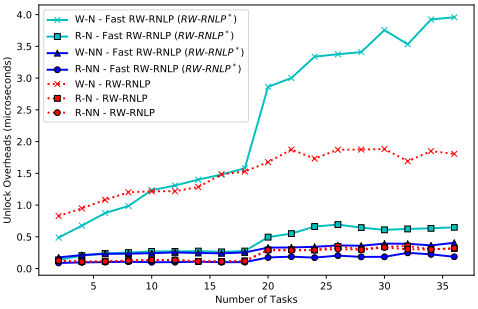



(c) Blocking.

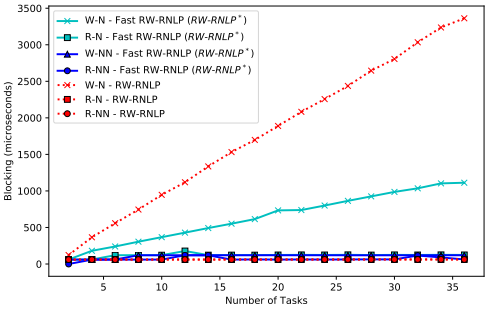
Figure 85: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

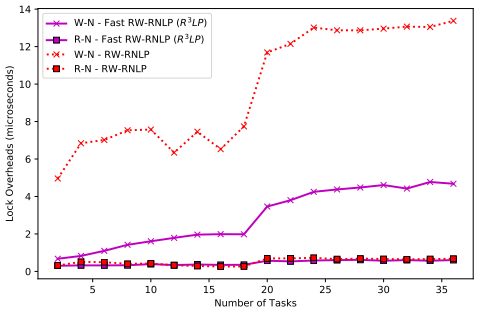



(b) Unlock overhead.

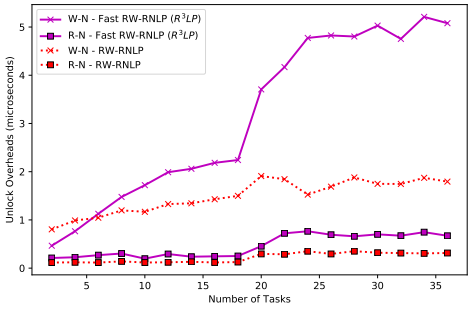



(c) Blocking.

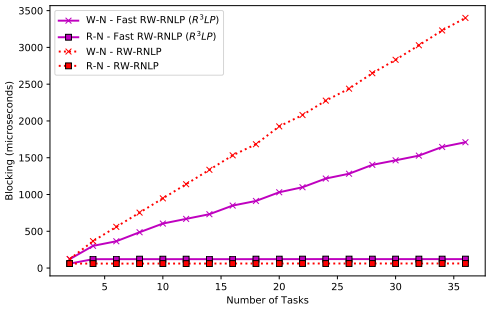
Figure 86: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

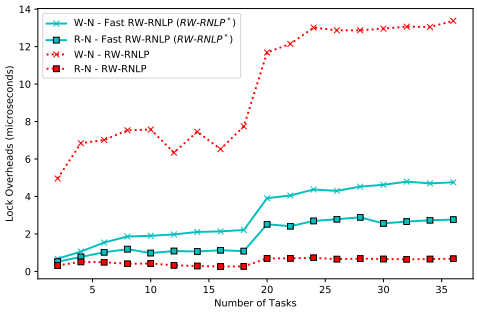



(b) Unlock overhead.

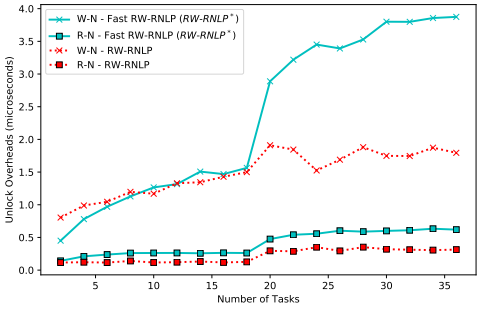



(c) Blocking.

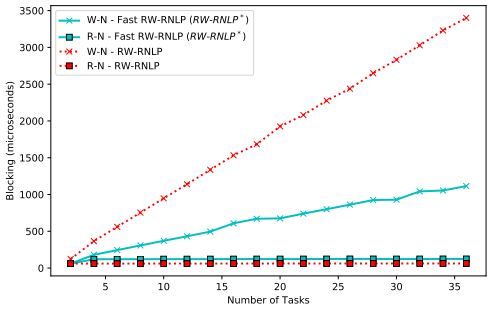
Figure 87: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

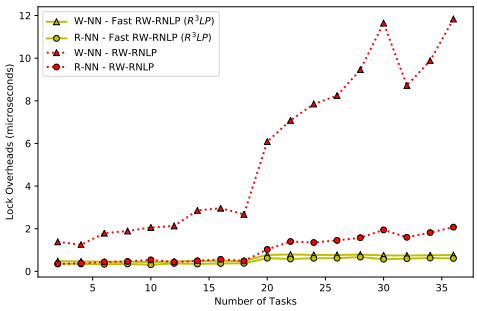



(b) Unlock overhead.

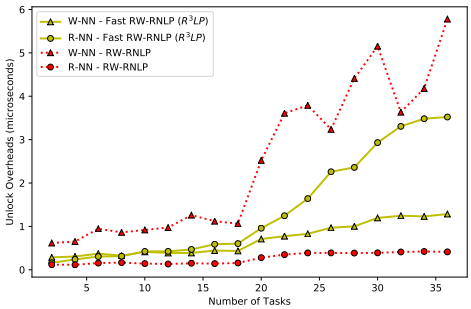



(c) Blocking.

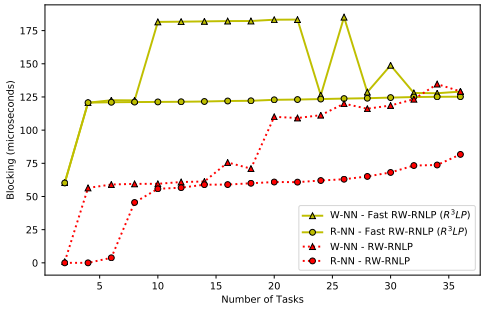
Figure 88: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

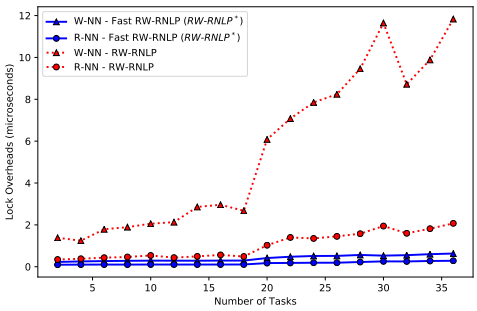



(b) Unlock overhead.

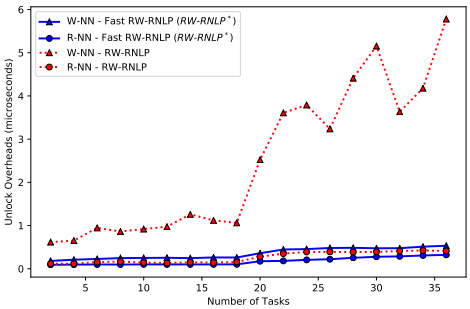



(c) Blocking.

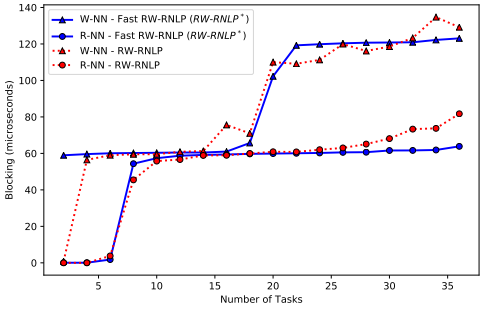
Figure 89: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

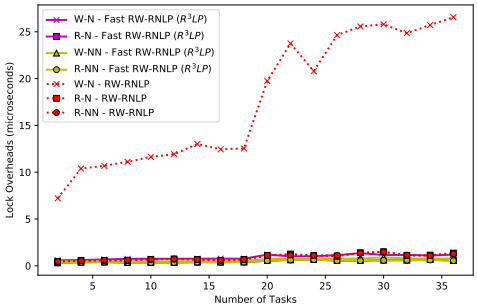



(b) Unlock overhead.

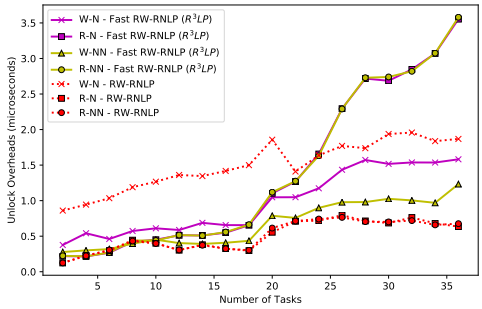



(c) Blocking.

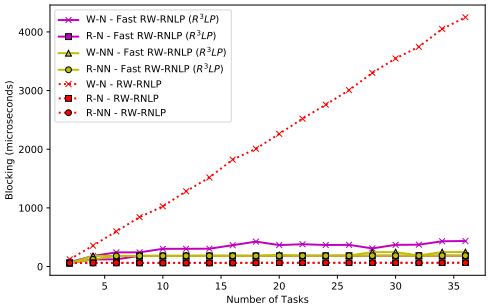
Figure 90: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

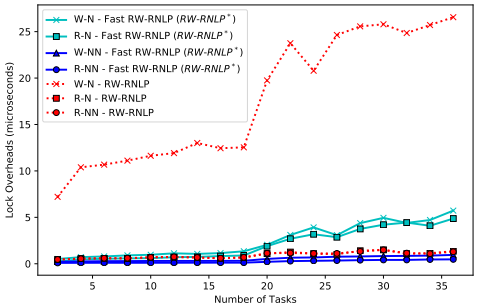



(b) Unlock overhead.

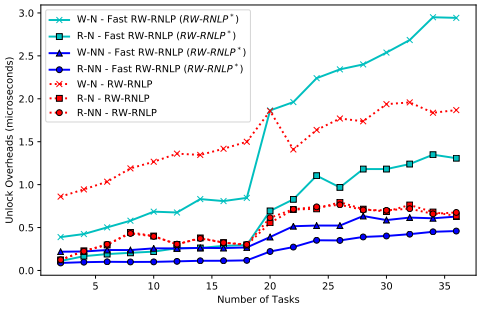



(c) Blocking.

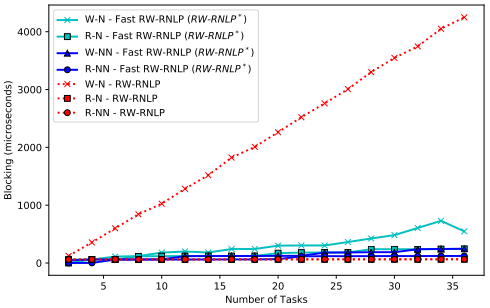
Figure 91: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

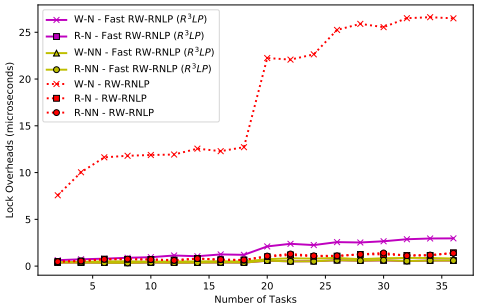



(b) Unlock overhead.

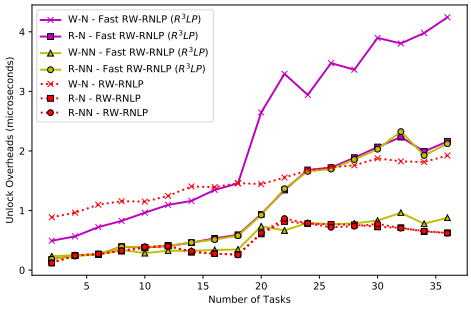



(c) Blocking.

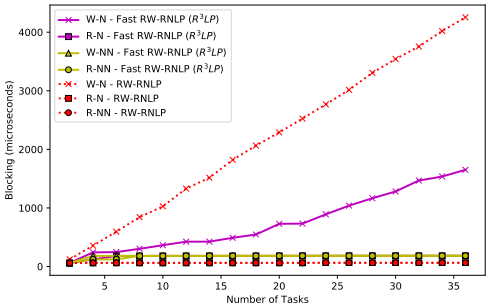
Figure 92: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

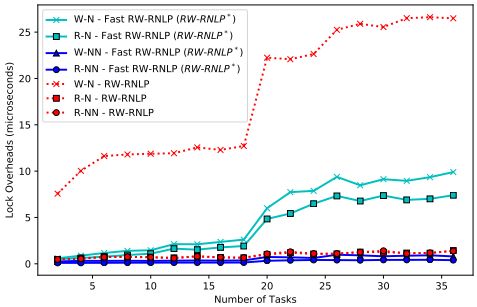



(b) Unlock overhead.

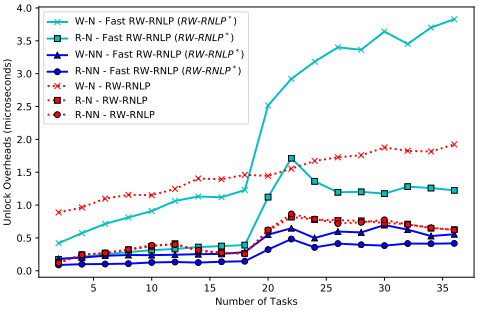



(c) Blocking.

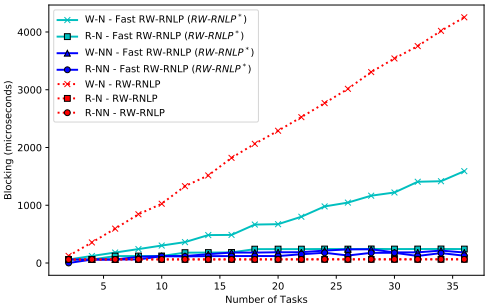
Figure 93: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

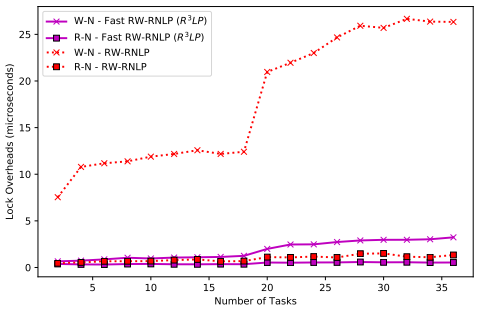



(b) Unlock overhead.

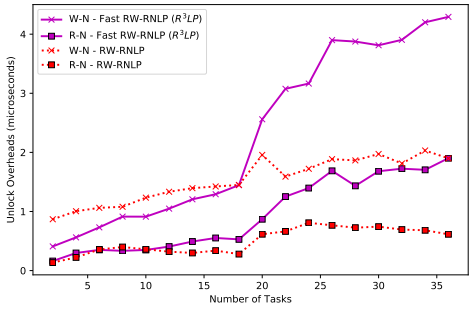



(c) Blocking.

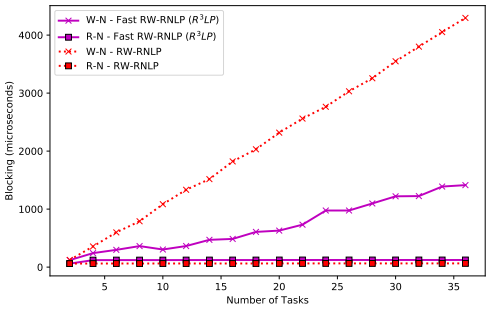
Figure 94: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

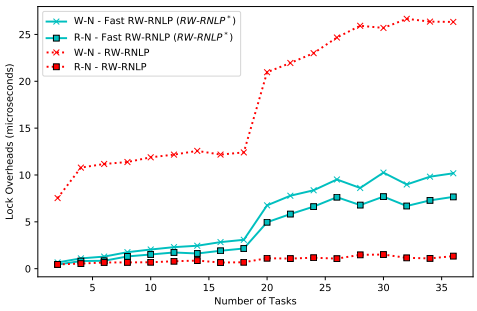



(b) Unlock overhead.

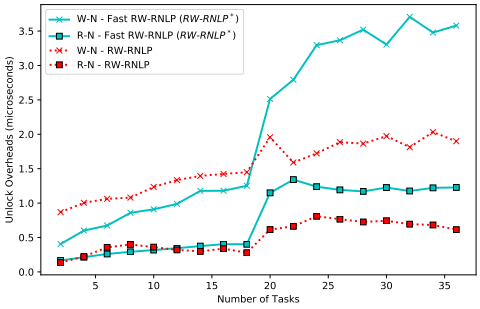



(c) Blocking.

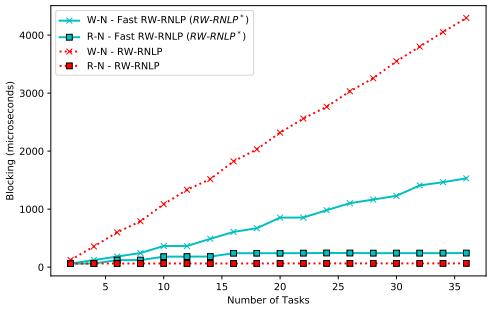
Figure 95: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

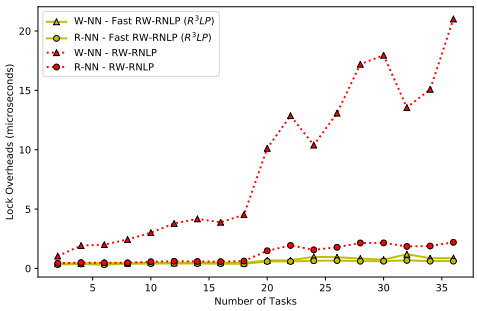



(b) Unlock overhead.

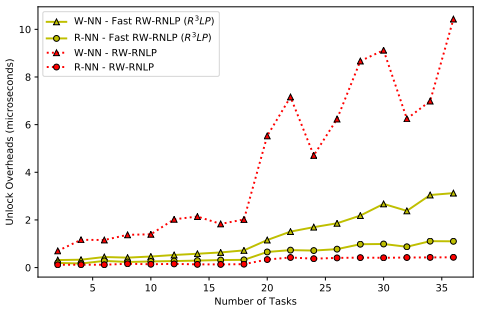



(c) Blocking.

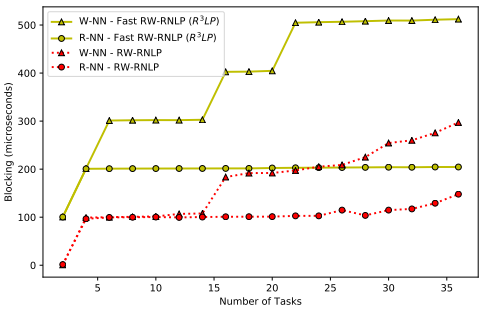
Figure 96: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

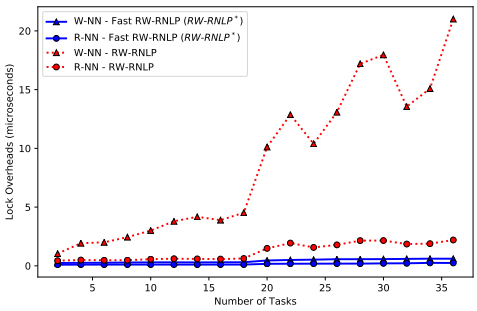



(b) Unlock overhead.

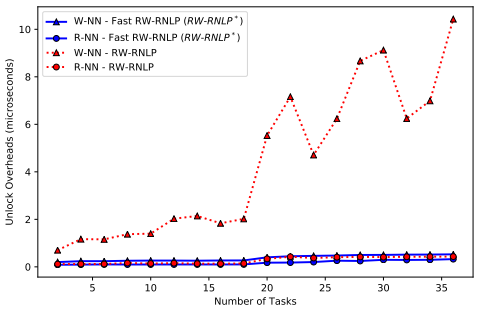



(c) Blocking.

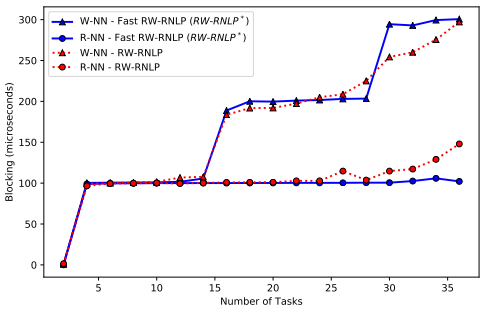
Figure 97: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

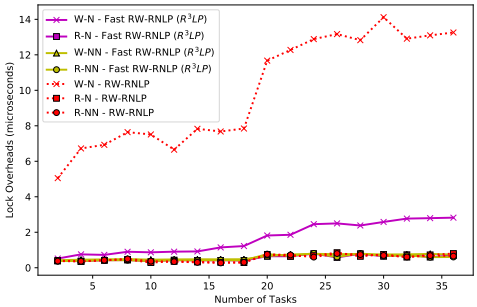



(b) Unlock overhead.

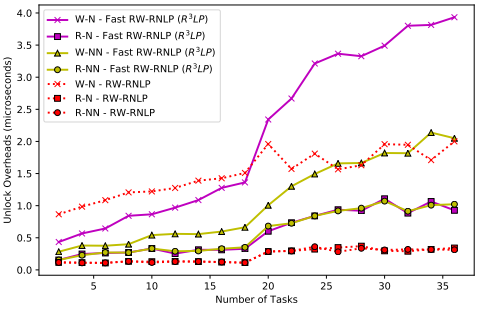



(c) Blocking.

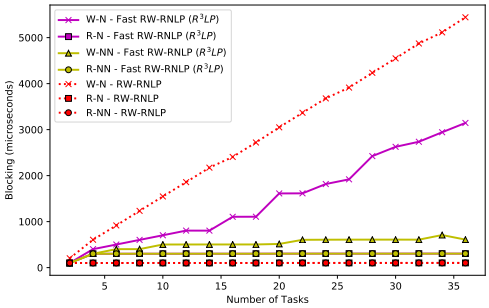
Figure 98: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

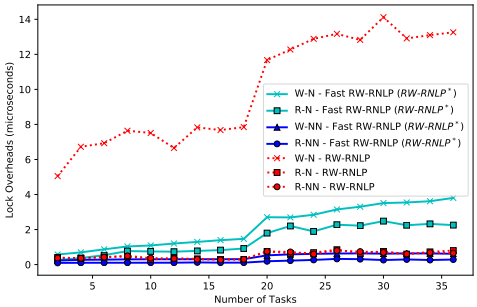



(b) Unlock overhead.

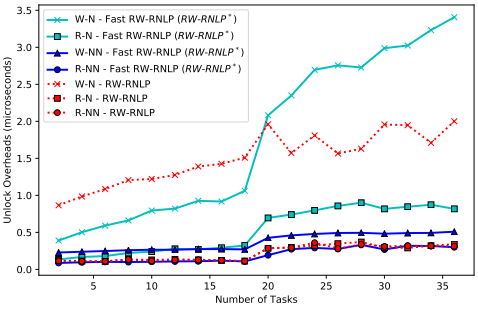



(c) Blocking.

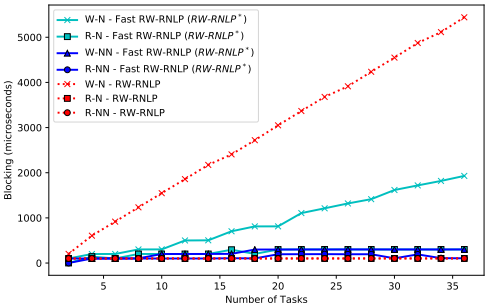
Figure 99: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

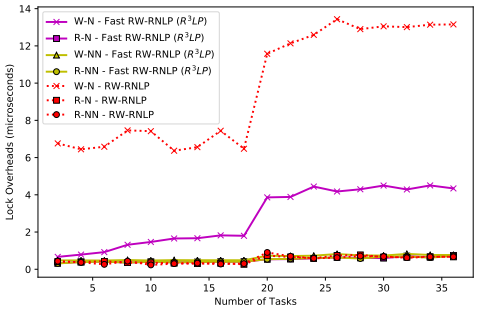



(b) Unlock overhead.

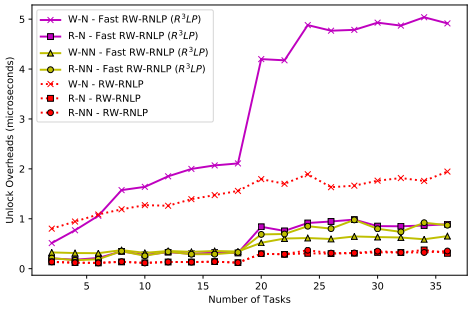



(c) Blocking.

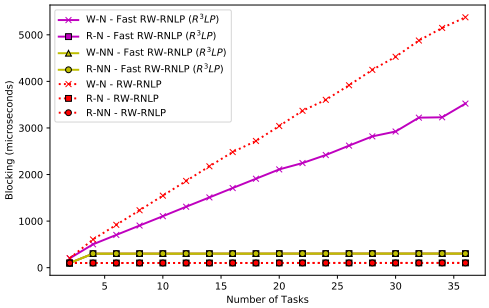
Figure 100: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

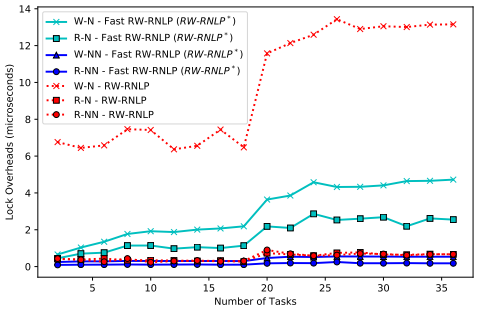



(b) Unlock overhead.

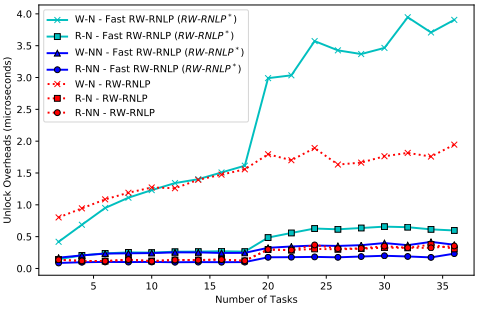



(c) Blocking.

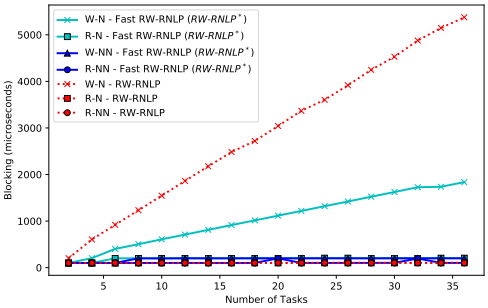
Figure 101: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

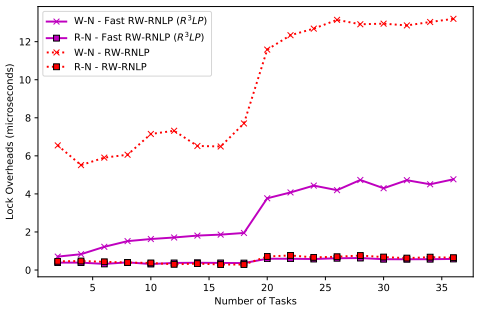



(b) Unlock overhead.

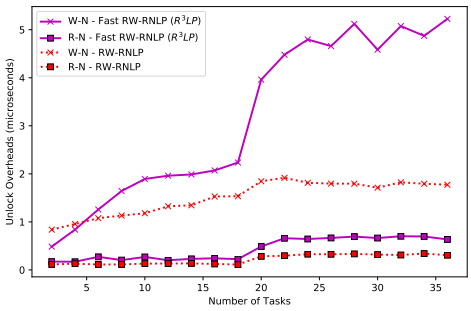



(c) Blocking.

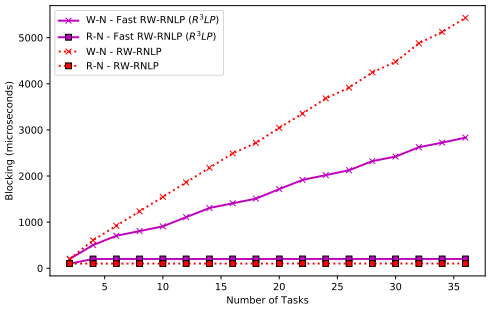
Figure 102: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

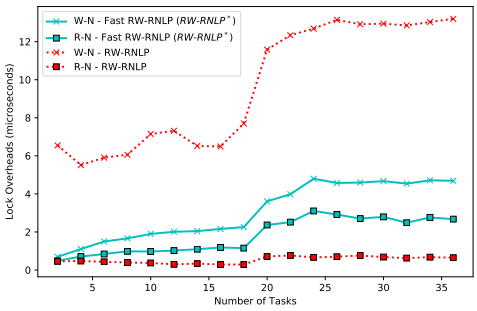



(b) Unlock overhead.

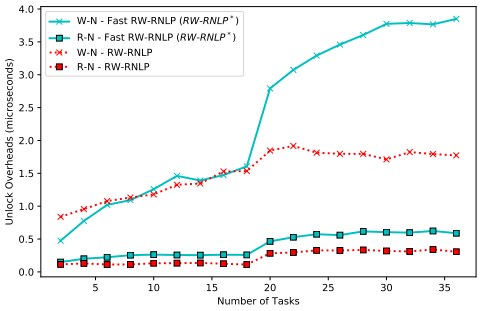



(c) Blocking.

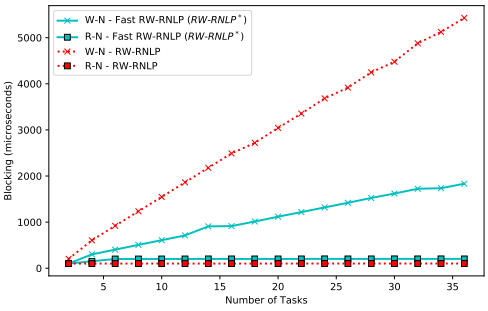
Figure 103: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

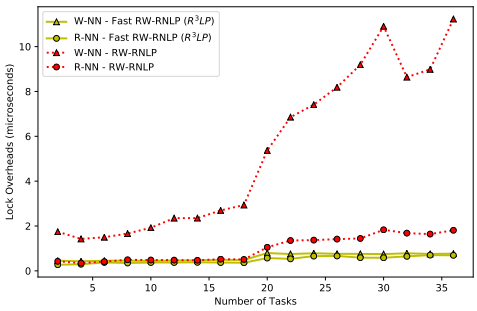



(b) Unlock overhead.

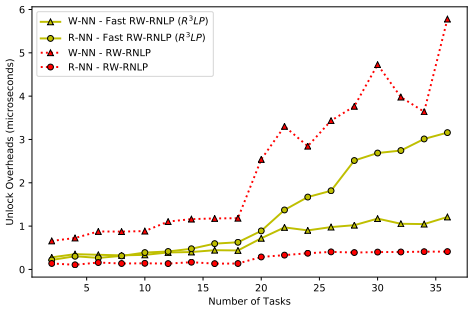



(c) Blocking.

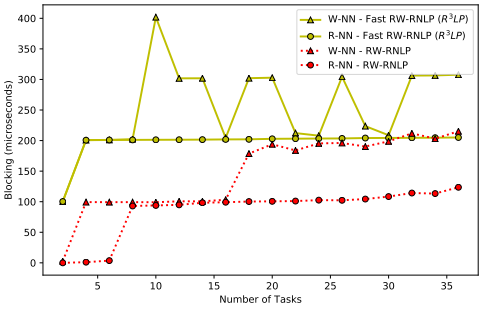
Figure 104: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

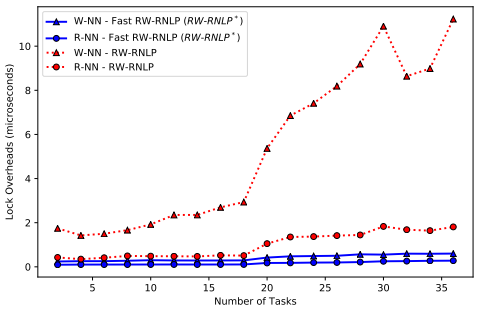



(b) Unlock overhead.

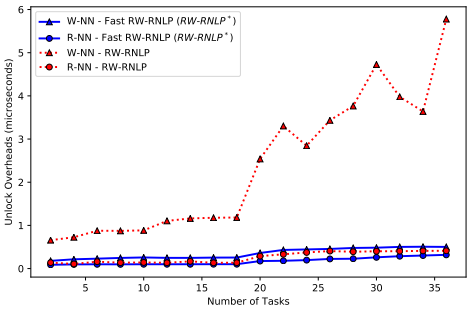



(c) Blocking.

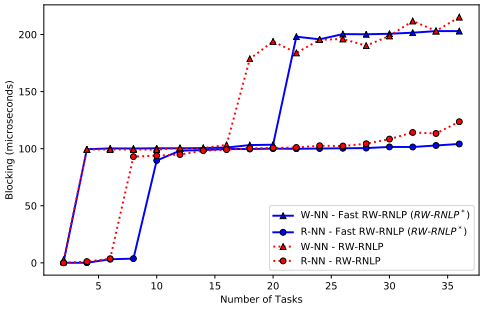
Figure 105: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

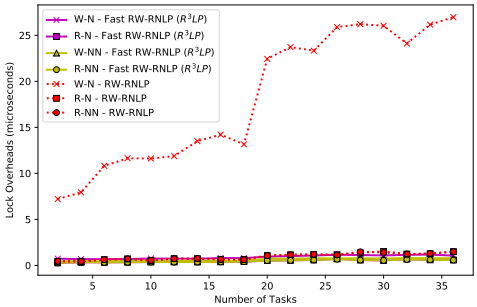



(b) Unlock overhead.

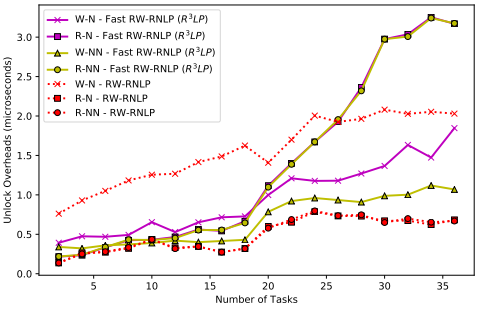



(c) Blocking.

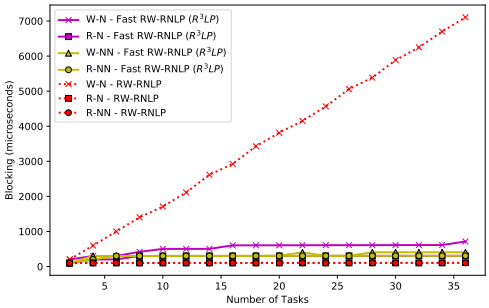
Figure 106: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

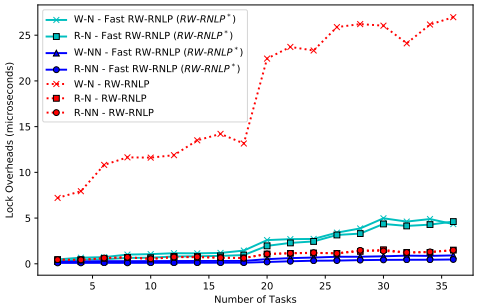



(b) Unlock overhead.

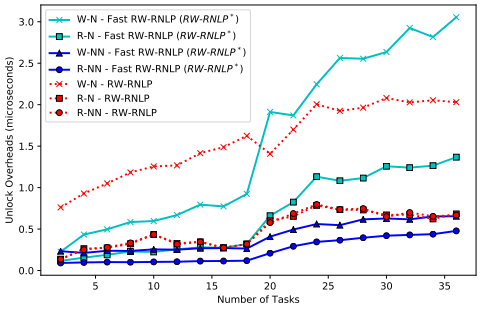



(c) Blocking.

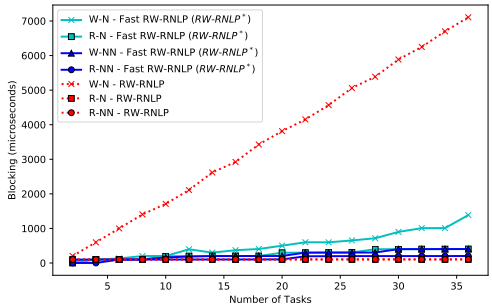
Figure 107: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

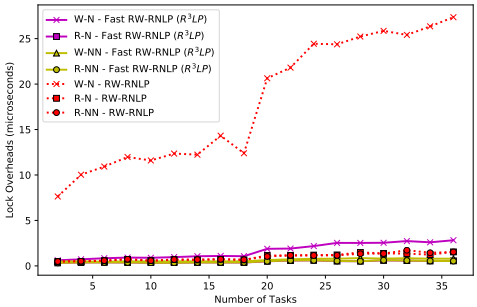



(b) Unlock overhead.

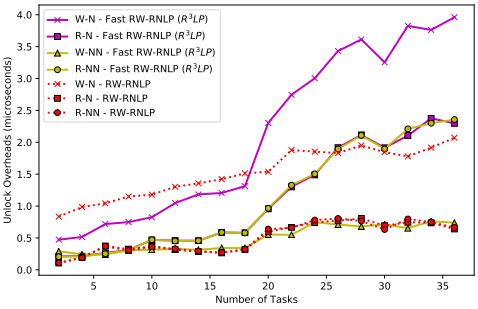



(c) Blocking.

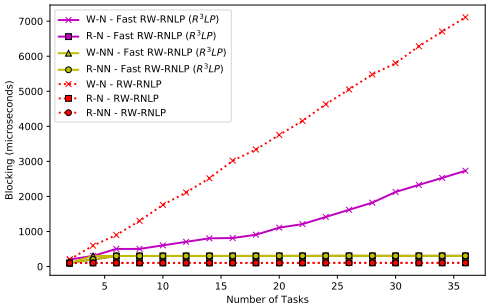
Figure 108: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

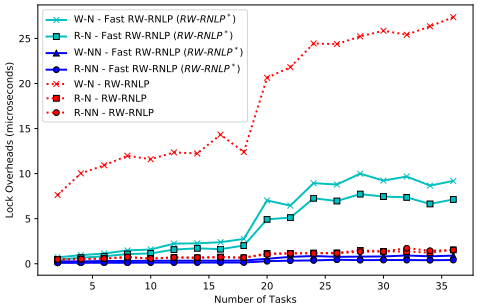



(b) Unlock overhead.

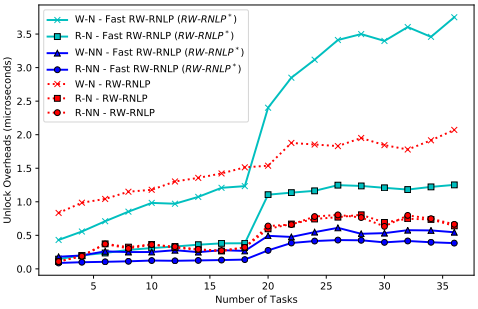



(c) Blocking.

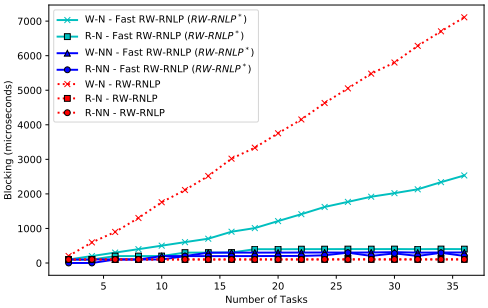
Figure 109: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

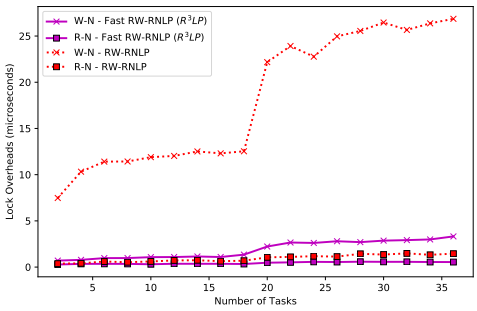



(b) Unlock overhead.

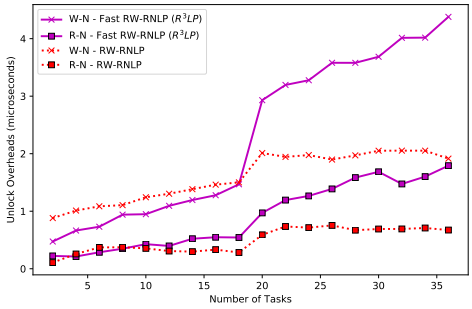



(c) Blocking.

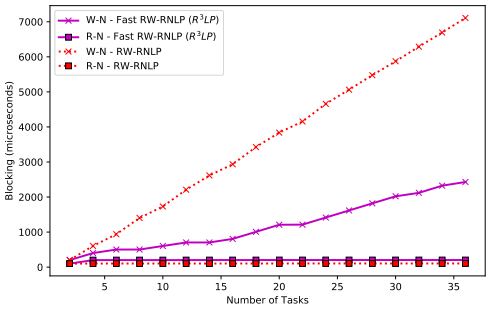
Figure 110: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

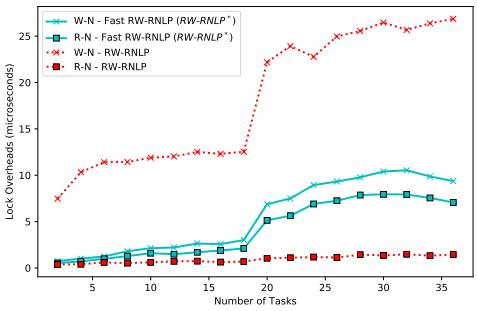



(b) Unlock overhead.

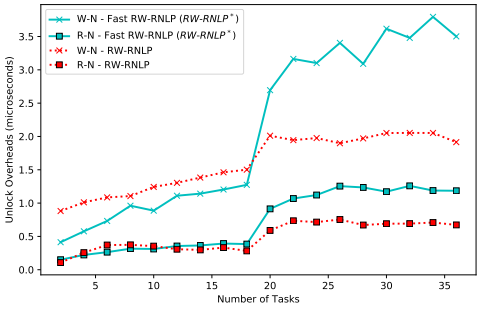



(c) Blocking.

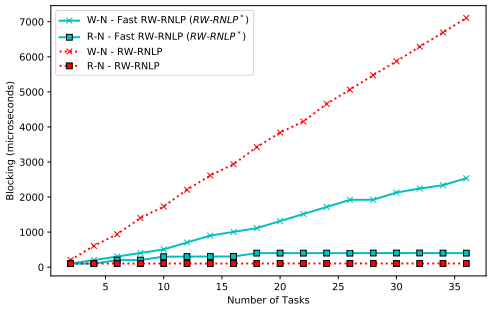
Figure 111: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

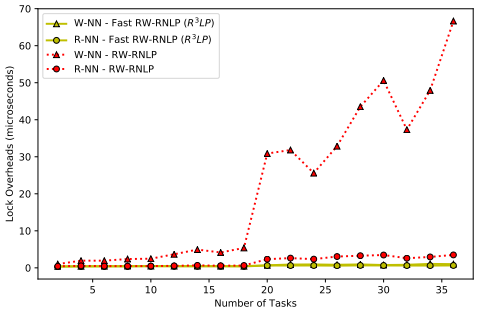



(b) Unlock overhead.

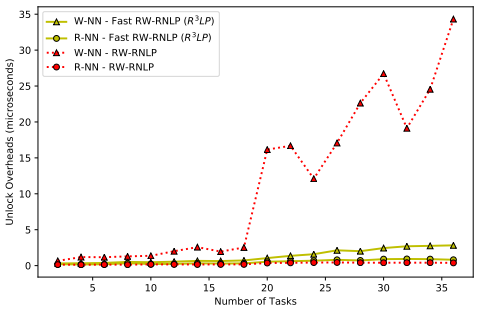



(c) Blocking.

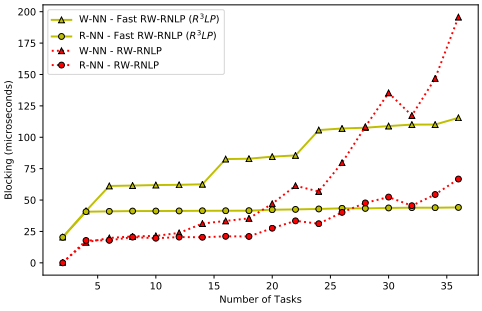
Figure 112: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

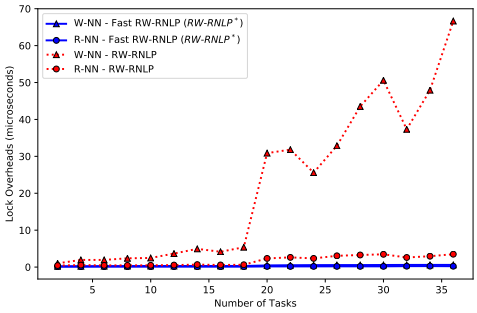



(b) Unlock overhead.

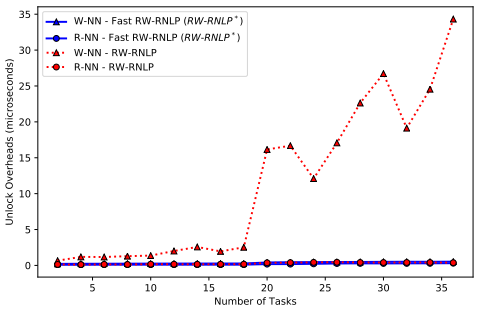



(c) Blocking.

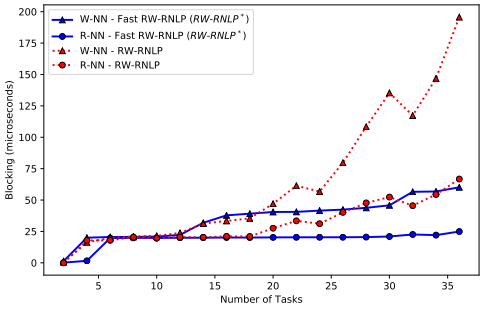
Figure 113: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

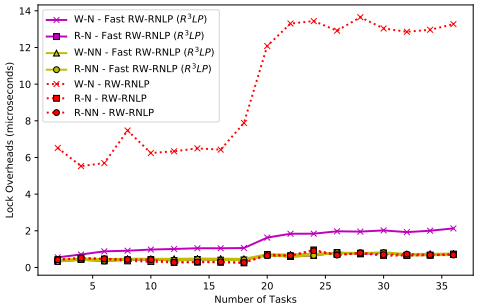



(b) Unlock overhead.

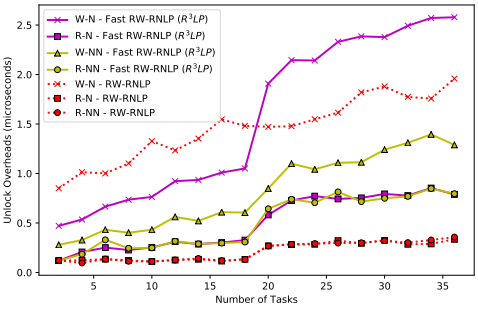



(c) Blocking.

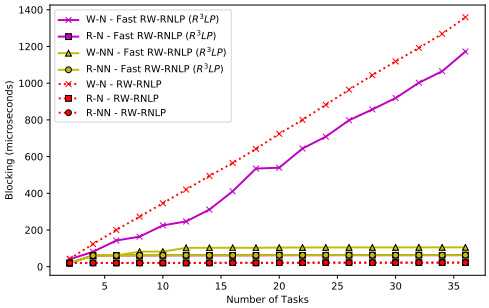
Figure 114: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

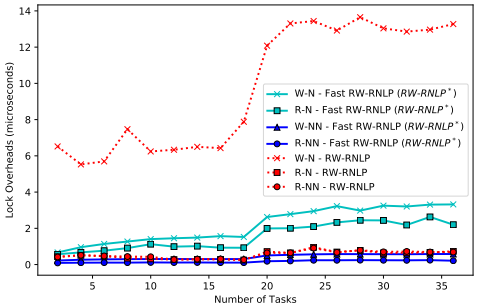



(b) Unlock overhead.

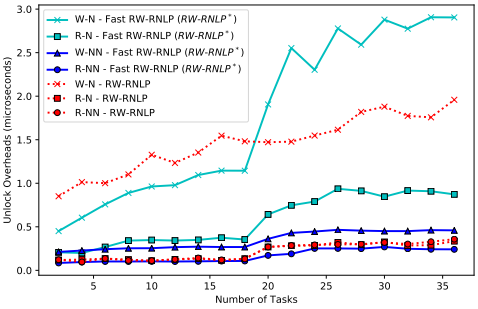



(c) Blocking.

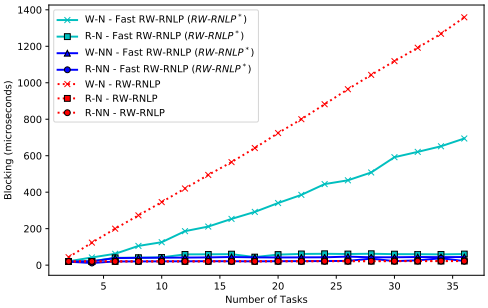
Figure 115: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

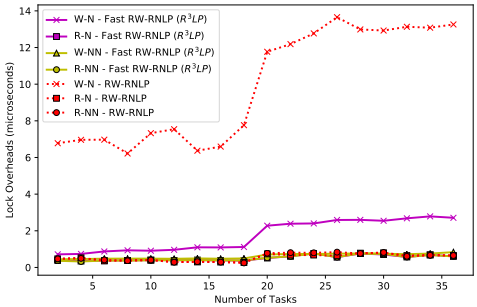



(b) Unlock overhead.

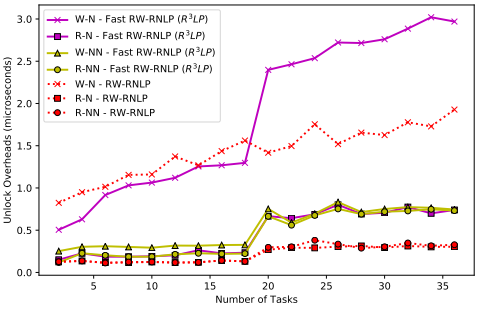



(c) Blocking.

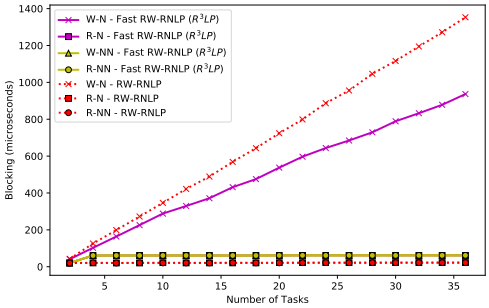
Figure 116: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

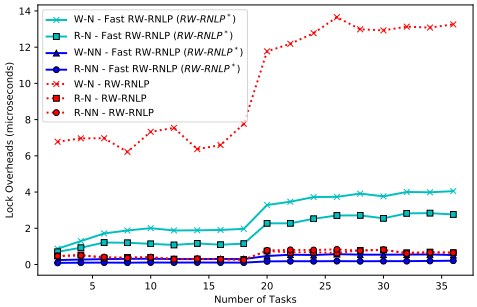



(b) Unlock overhead.

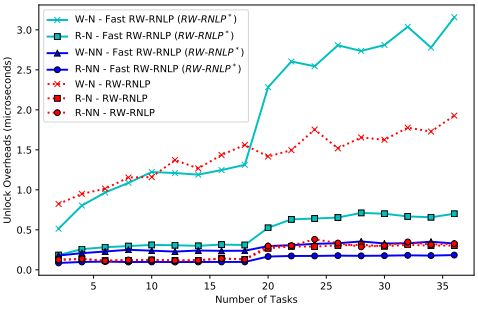



(c) Blocking.

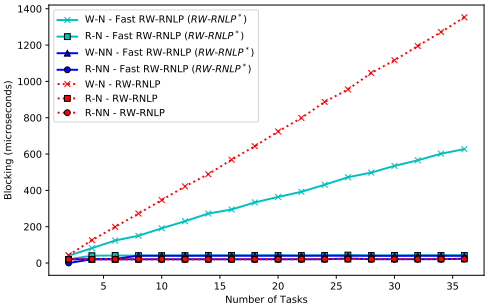
Figure 117: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

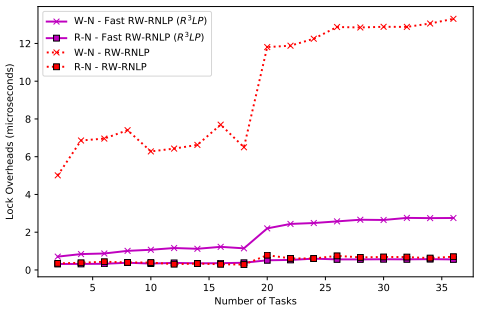



(b) Unlock overhead.

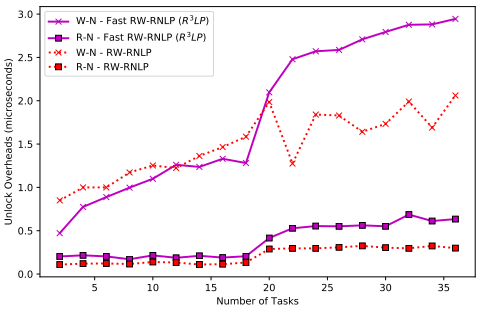



(c) Blocking.

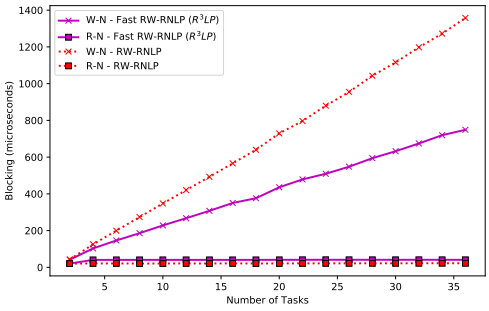
Figure 118: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

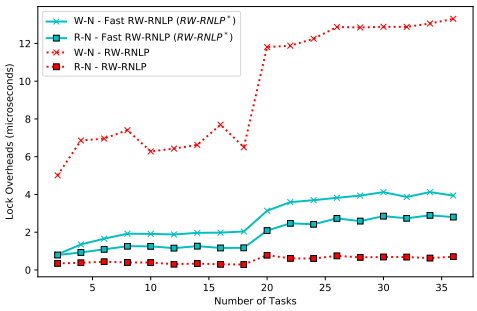



(b) Unlock overhead.

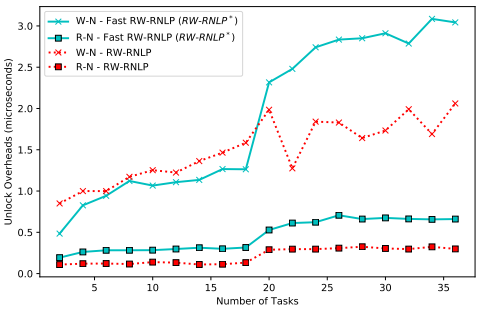



(c) Blocking.

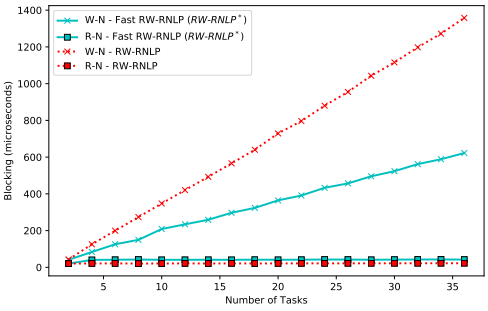
Figure 119: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

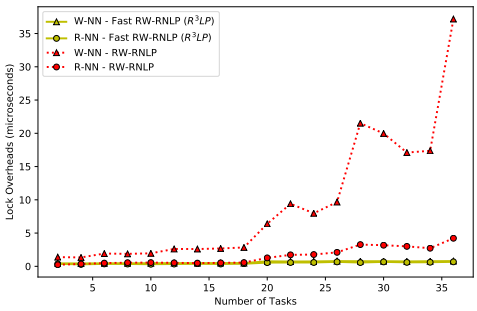



(b) Unlock overhead.

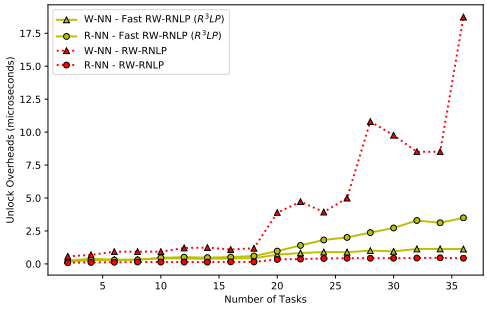



(c) Blocking.

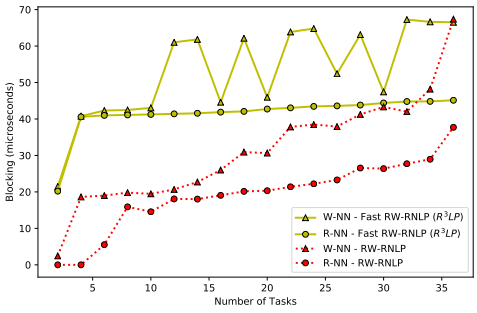
Figure 120: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

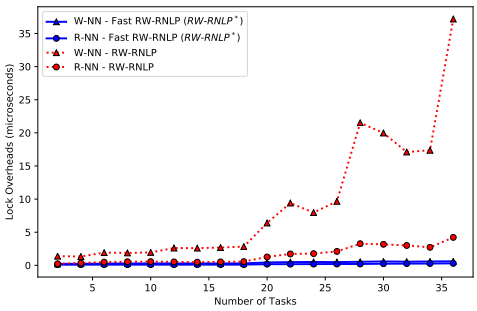



(b) Unlock overhead.

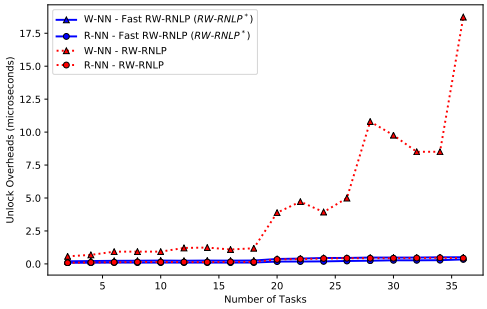



(c) Blocking.

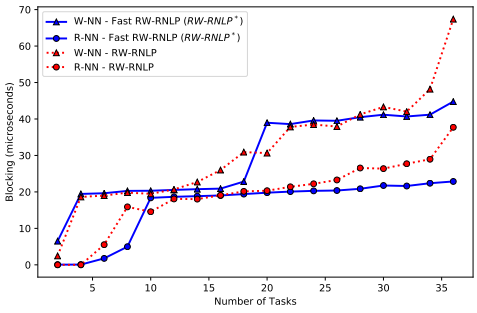
Figure 121: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

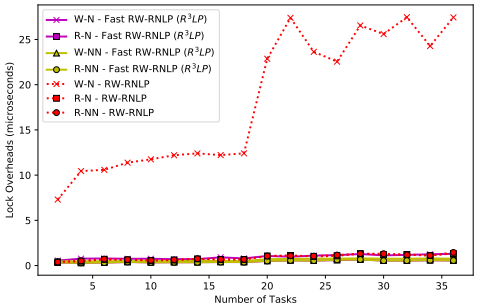



(b) Unlock overhead.

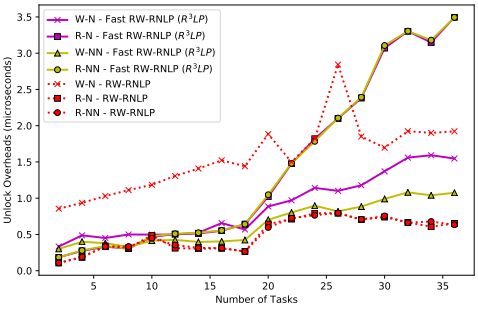



(c) Blocking.

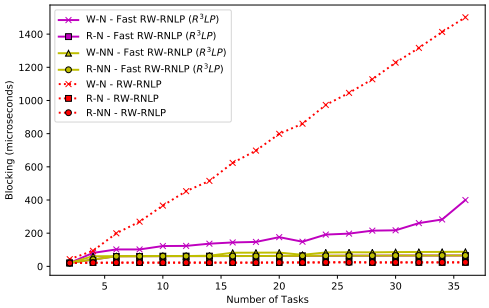
Figure 122: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

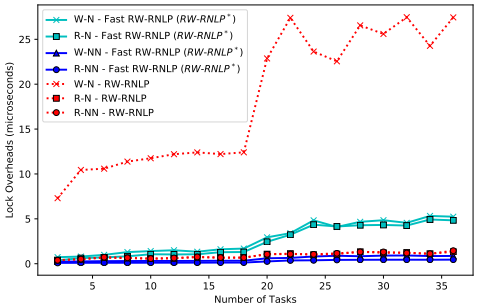



(b) Unlock overhead.

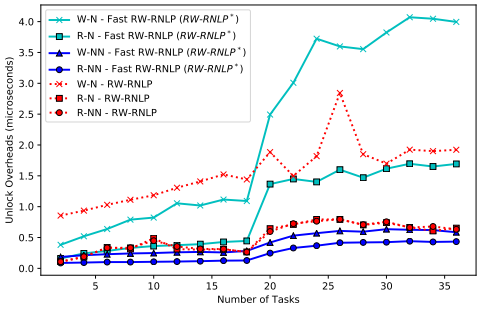



(c) Blocking.

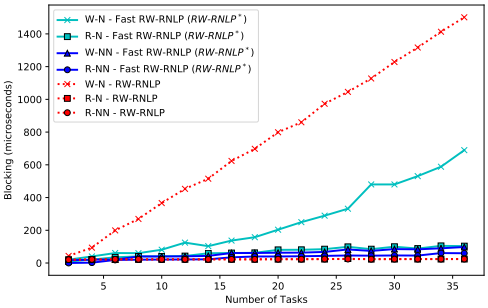
Figure 123: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

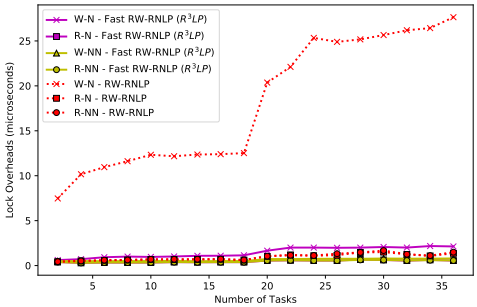



(b) Unlock overhead.

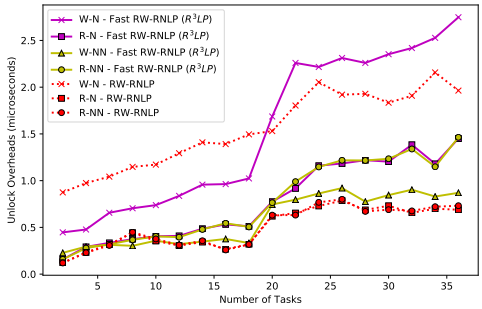



(c) Blocking.

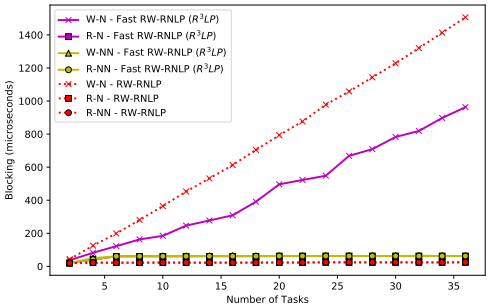
Figure 124: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

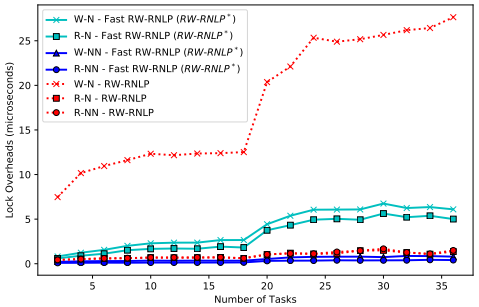



(b) Unlock overhead.

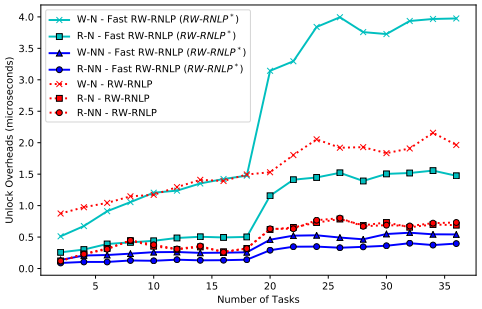



(c) Blocking.

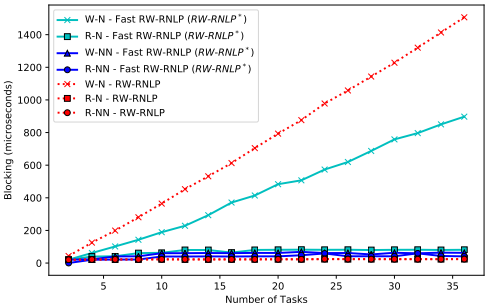
Figure 125: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

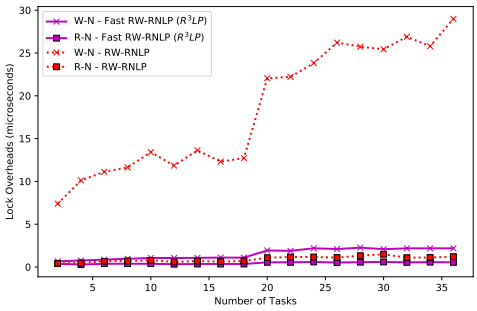



(b) Unlock overhead.

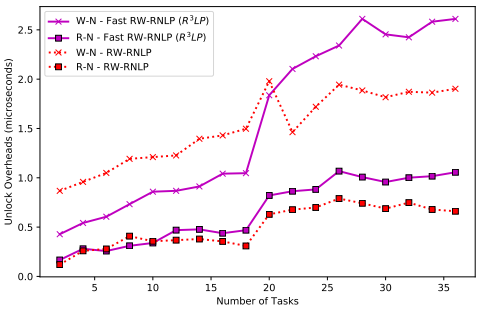



(c) Blocking.

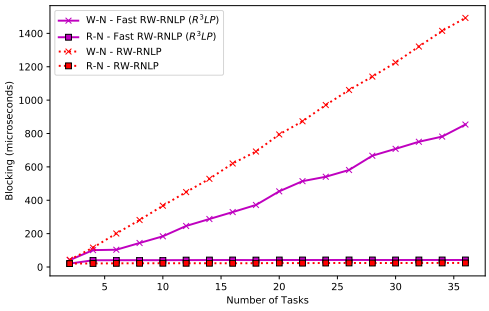
Figure 126: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

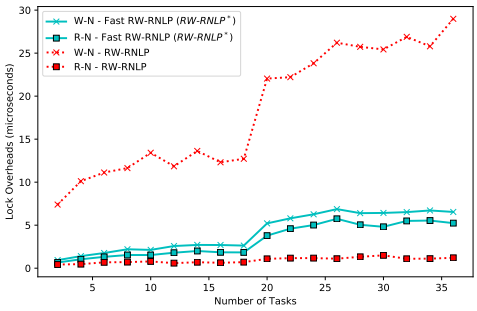



(b) Unlock overhead.

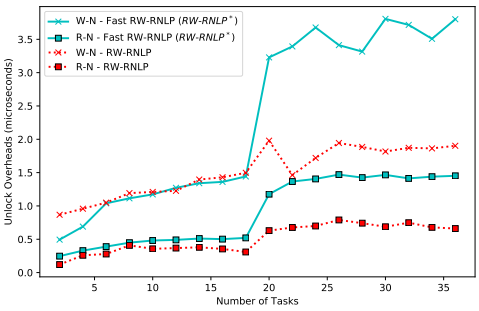



(c) Blocking.

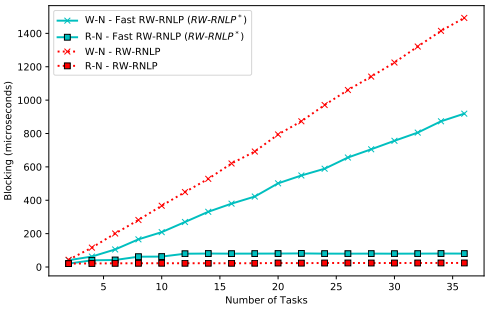
Figure 127: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

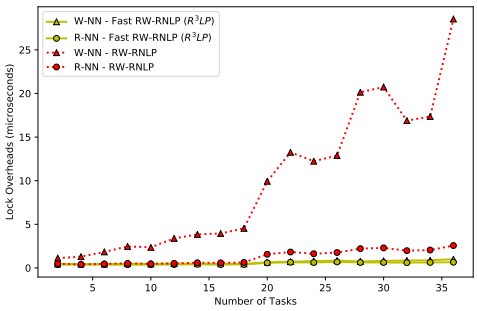



(b) Unlock overhead.

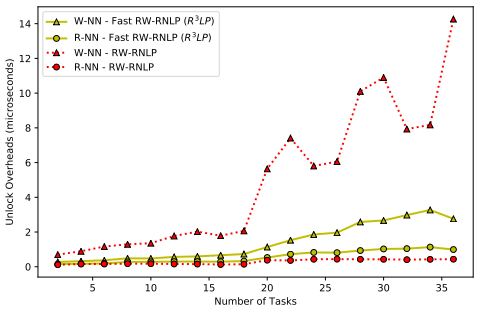



(c) Blocking.

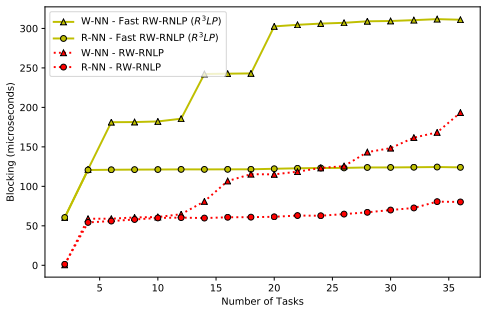
Figure 128: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

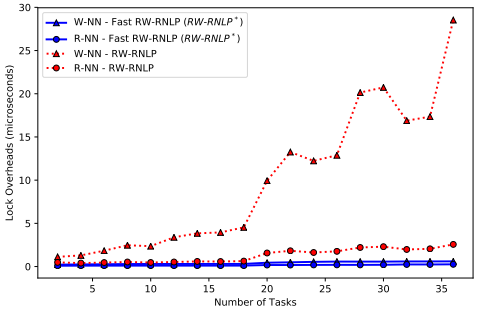



(b) Unlock overhead.

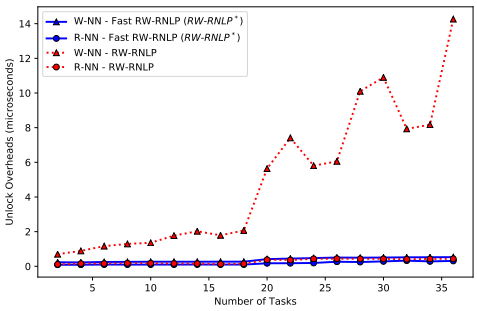



(c) Blocking.

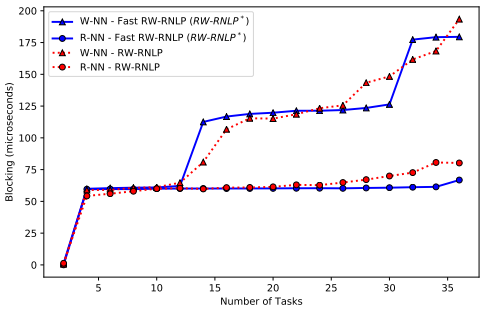
Figure 129: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

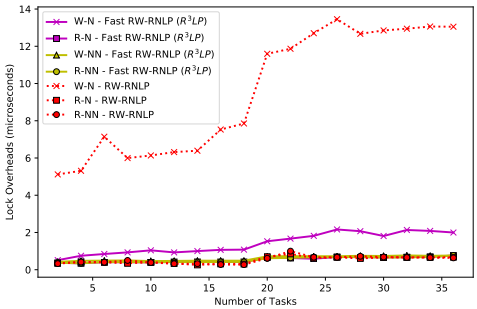



(b) Unlock overhead.

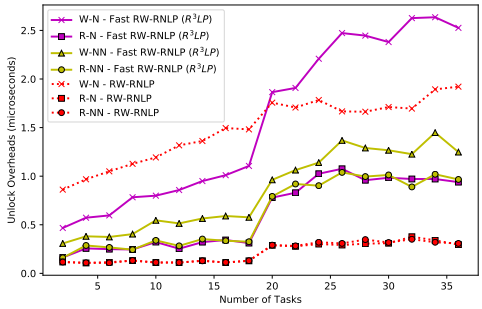



(c) Blocking.

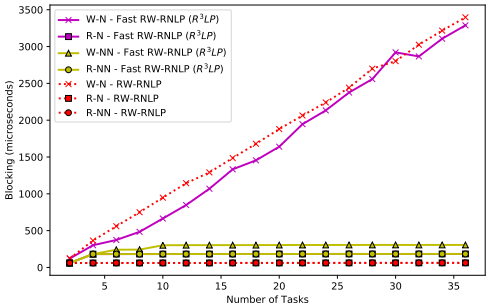
Figure 130: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

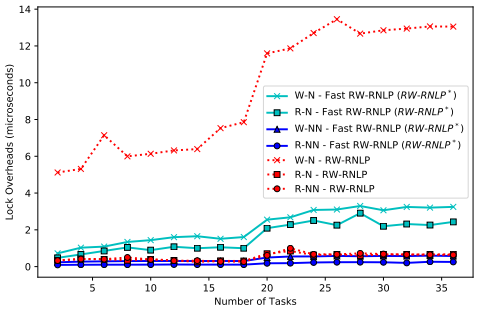



(b) Unlock overhead.

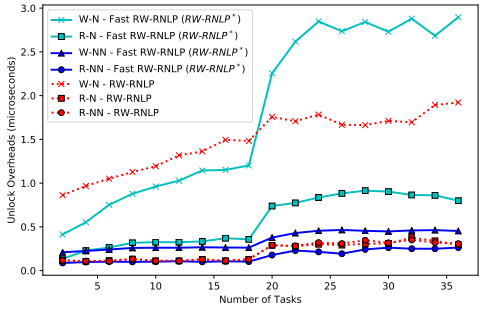



(c) Blocking.

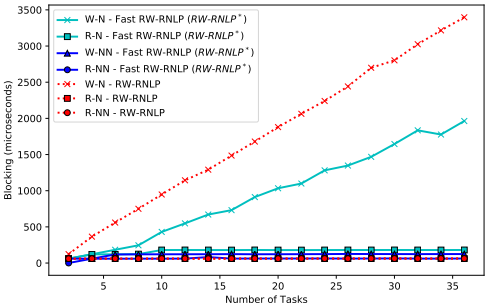
Figure 131: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

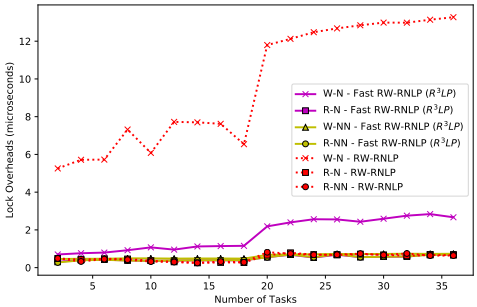



(b) Unlock overhead.

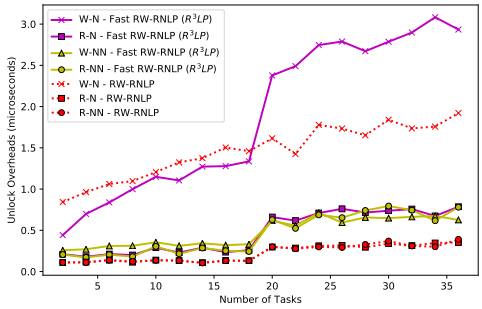



(c) Blocking.

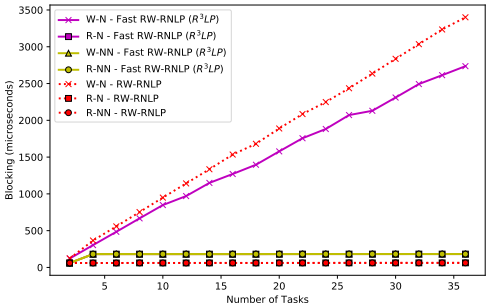
Figure 132: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

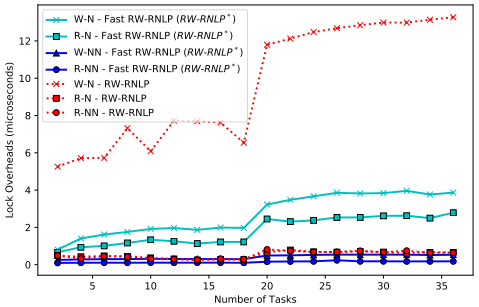



(b) Unlock overhead.

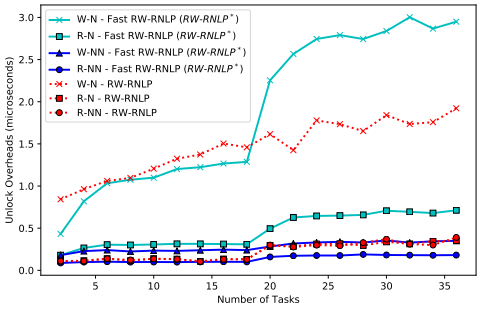



(c) Blocking.

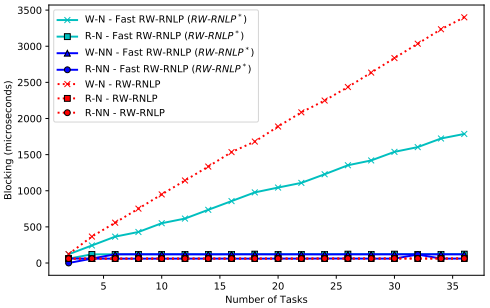
Figure 133: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

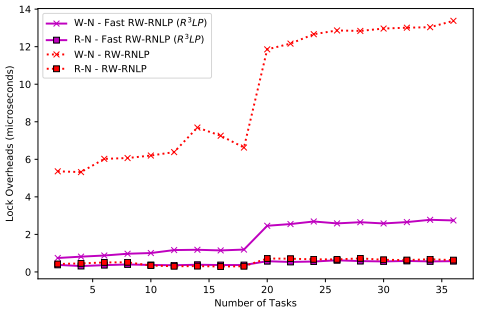



(b) Unlock overhead.

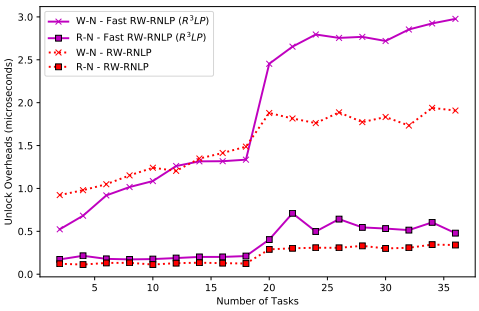



(c) Blocking.

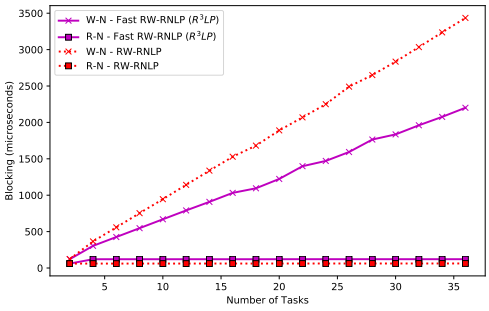
Figure 134: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

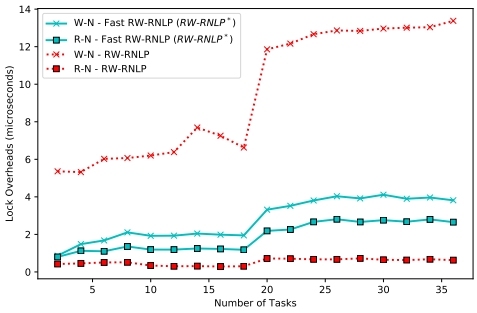



(b) Unlock overhead.

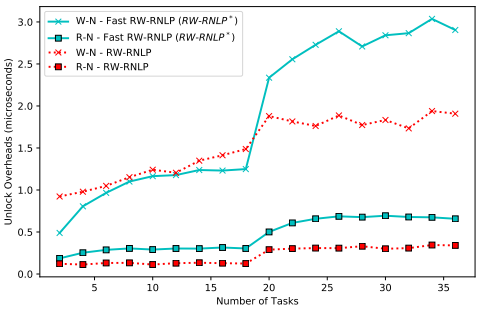



(c) Blocking.

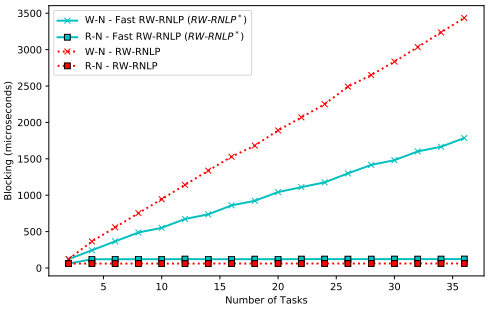
Figure 135: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

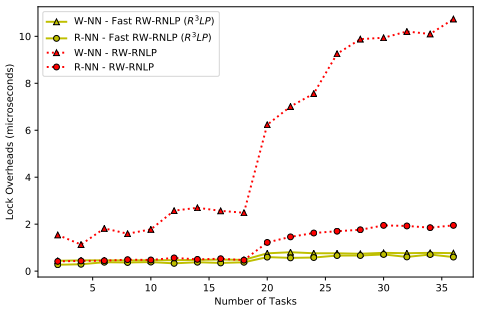



(b) Unlock overhead.

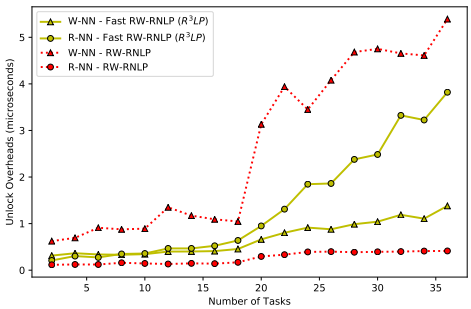



(c) Blocking.

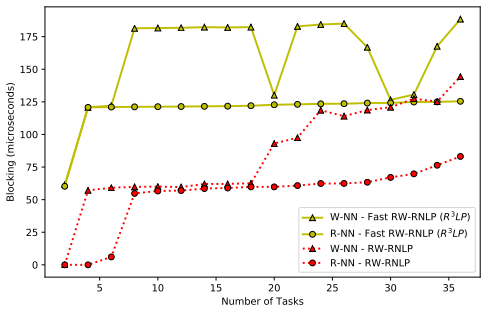
Figure 136: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

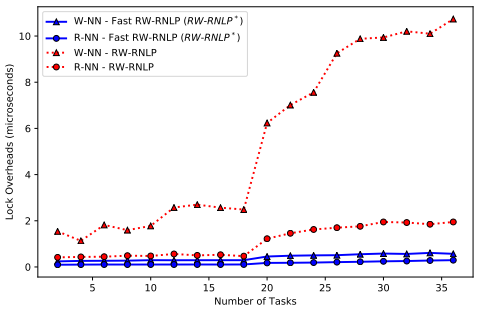



(b) Unlock overhead.

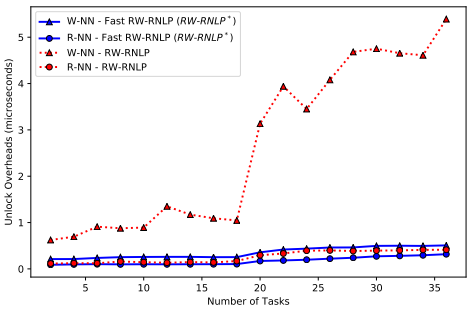



(c) Blocking.

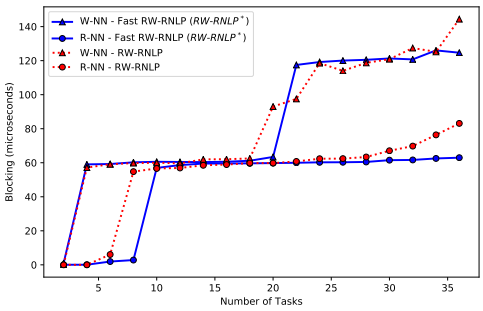
Figure 137: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

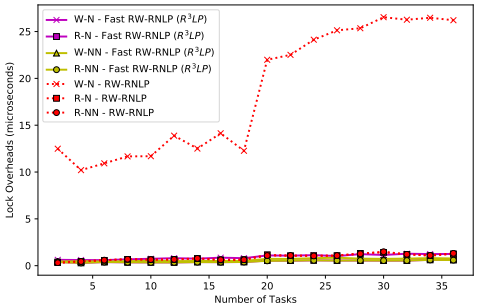



(b) Unlock overhead.

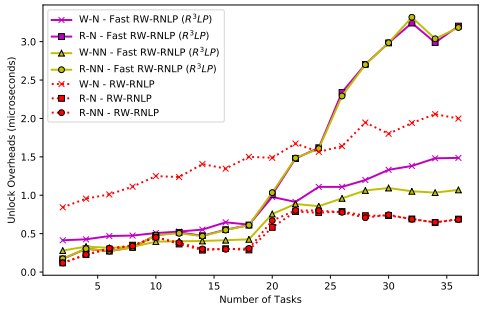



(c) Blocking.

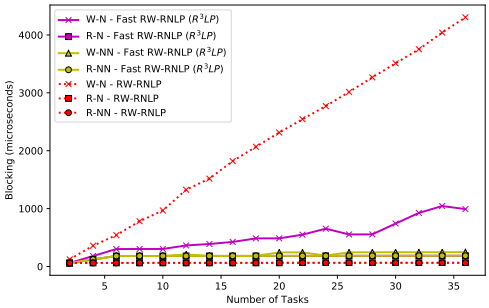
Figure 138: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

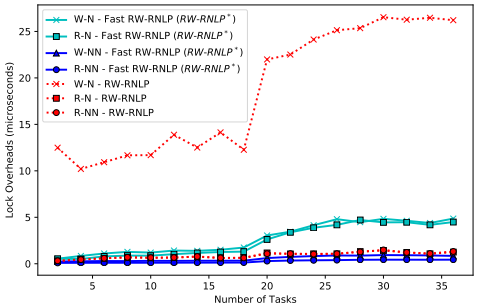



(b) Unlock overhead.

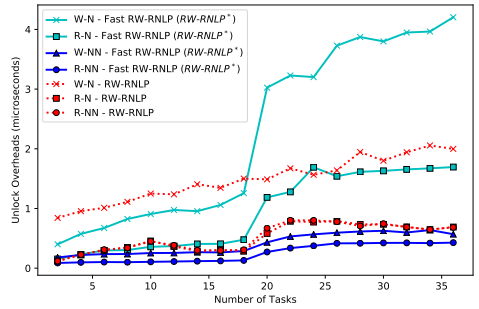



(c) Blocking.

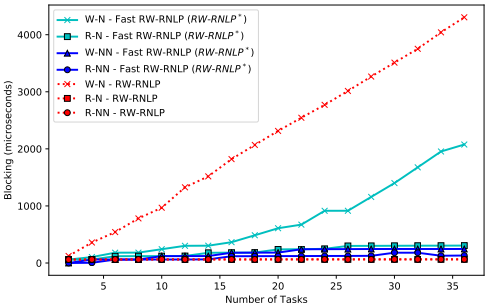
Figure 139: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

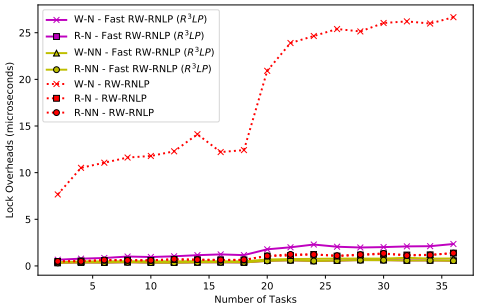



(b) Unlock overhead.

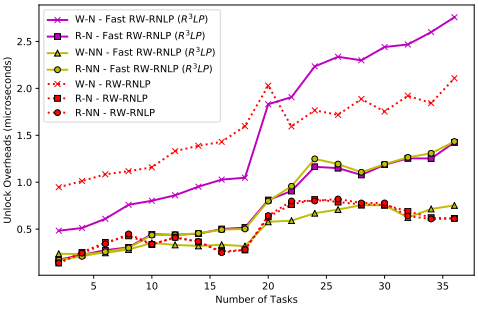



(c) Blocking.

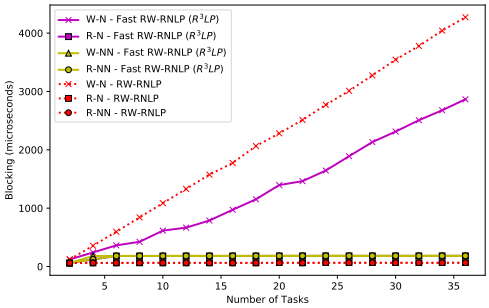
Figure 140: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

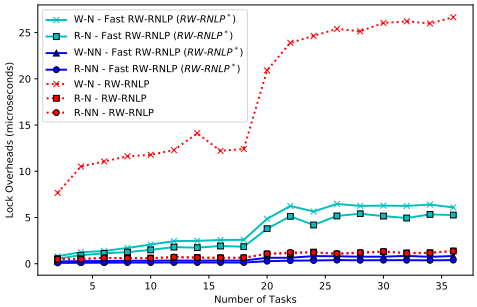



(b) Unlock overhead.

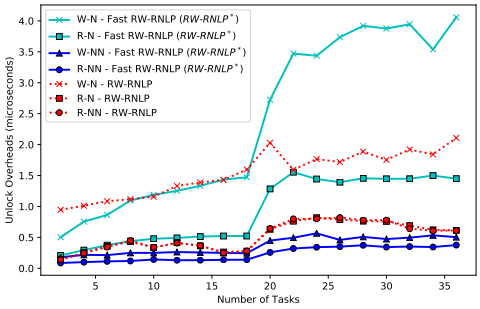



(c) Blocking.

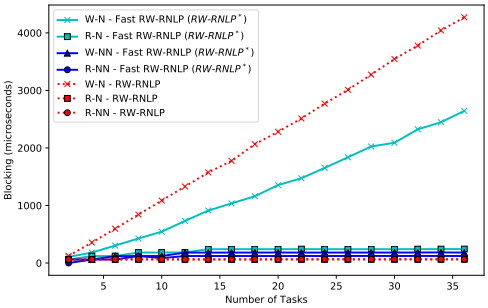
Figure 141: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

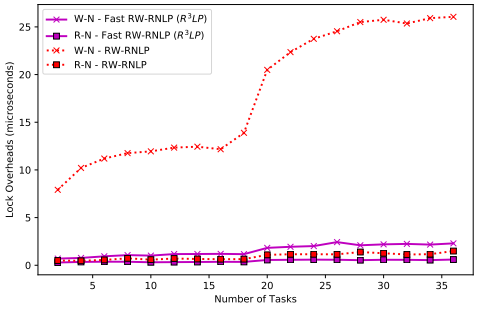



(b) Unlock overhead.

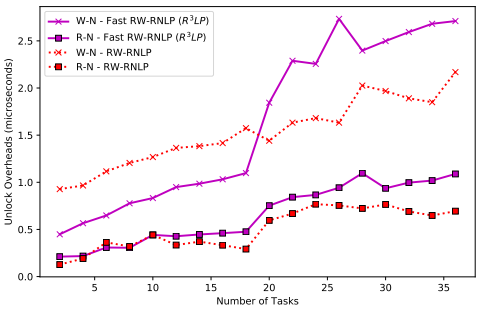



(c) Blocking.

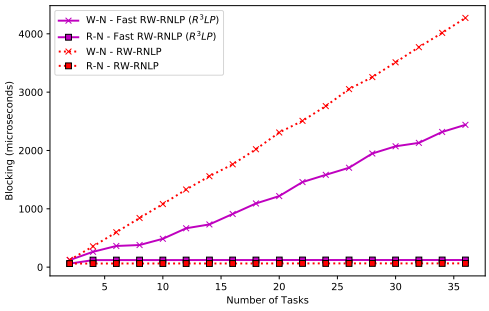
Figure 142: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

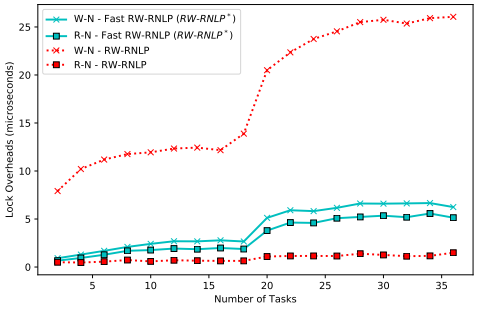



(b) Unlock overhead.

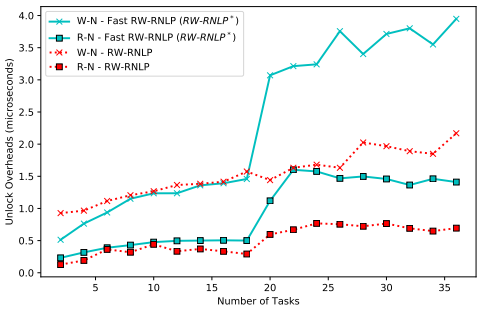



(c) Blocking.

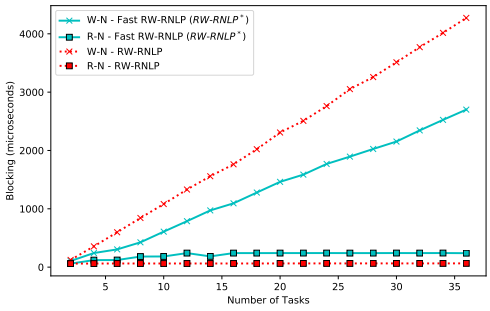
Figure 143: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

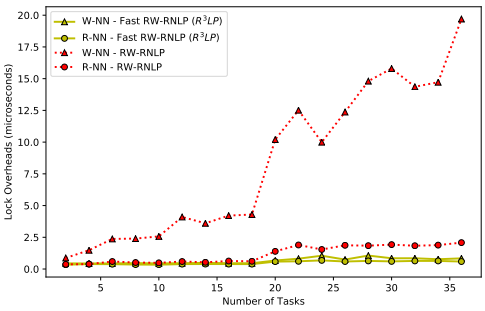



(b) Unlock overhead.

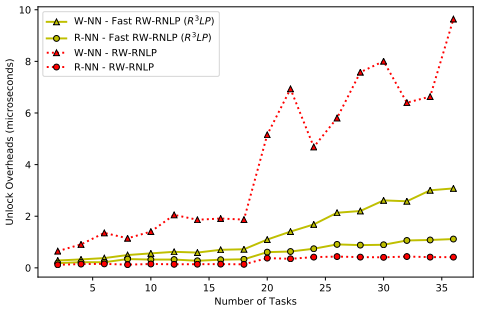



(c) Blocking.

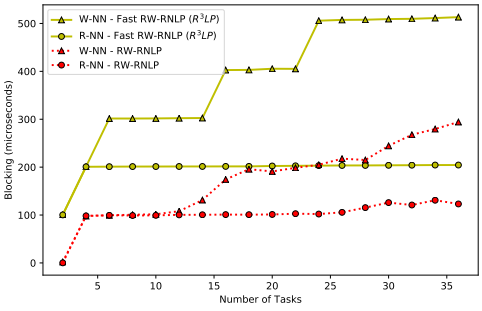
Figure 144: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

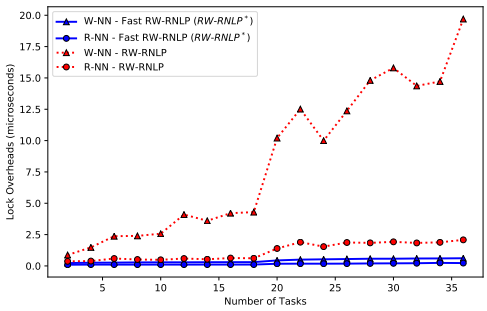



(b) Unlock overhead.

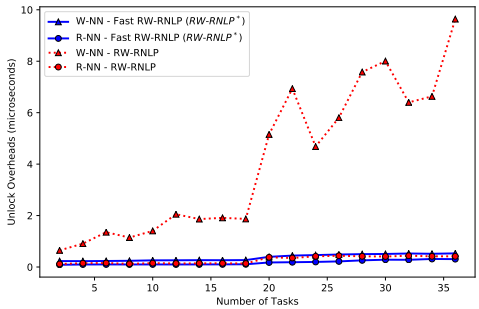



(c) Blocking.

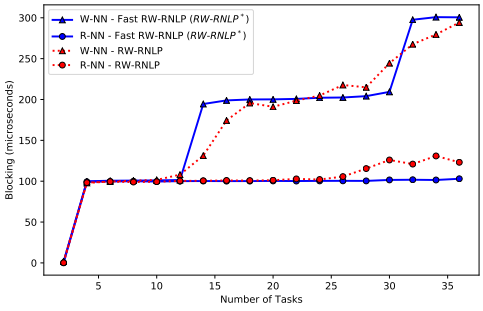
Figure 145: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

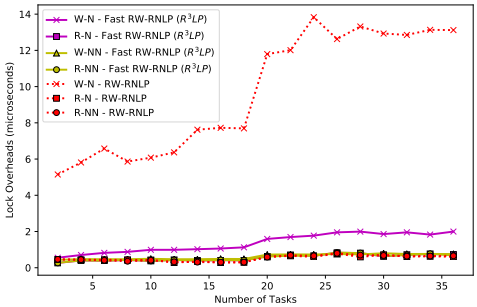



(b) Unlock overhead.

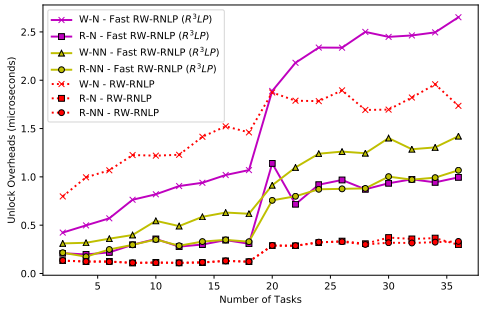



(c) Blocking.

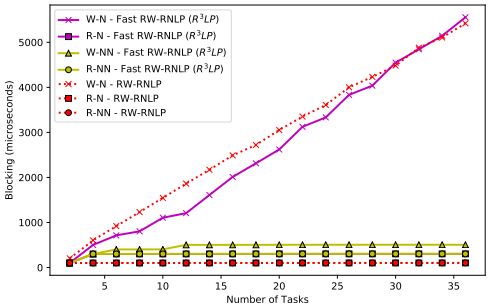
Figure 146: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

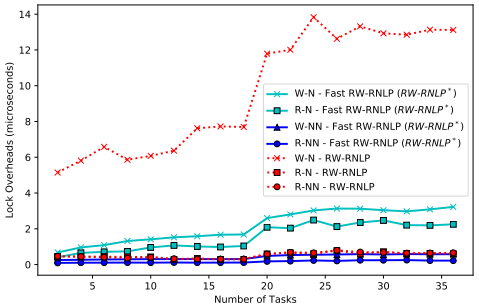



(b) Unlock overhead.

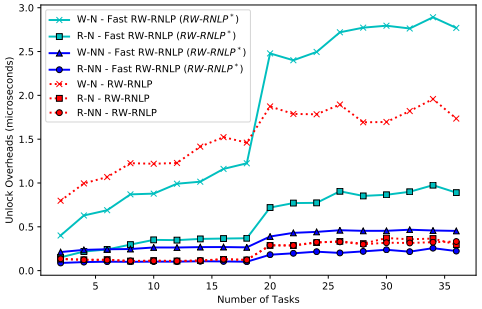



(c) Blocking.

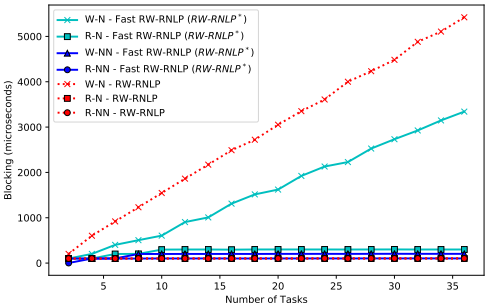
Figure 147: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

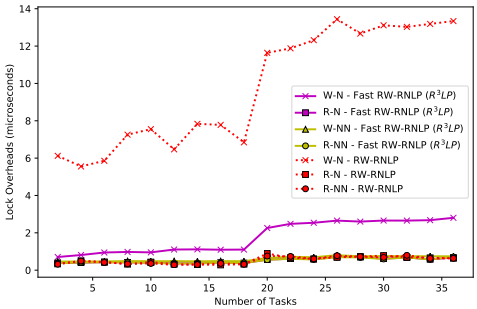



(b) Unlock overhead.

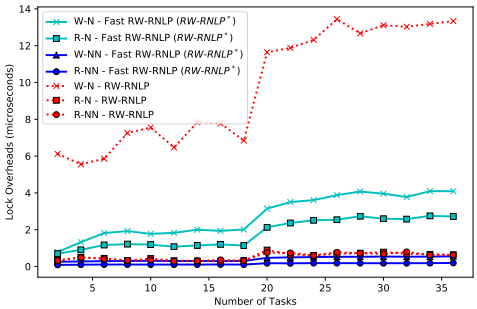



(c) Blocking.

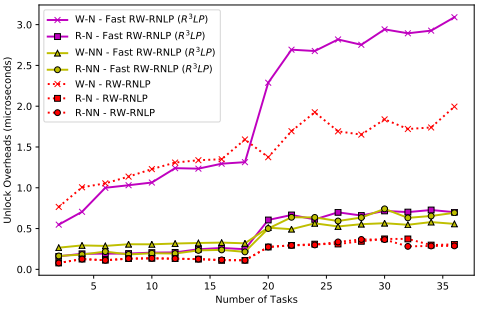
Figure 148: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.



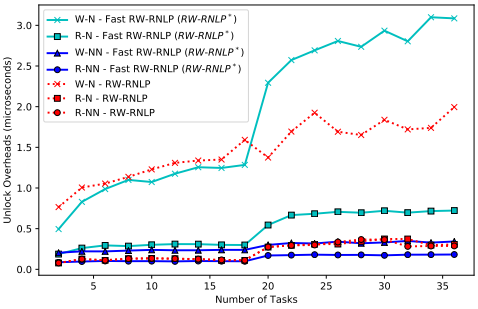

(a) Lock overhead.



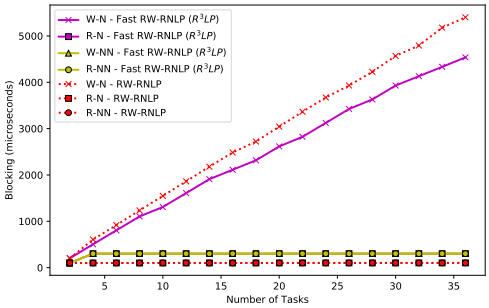

(a) Lock overhead.



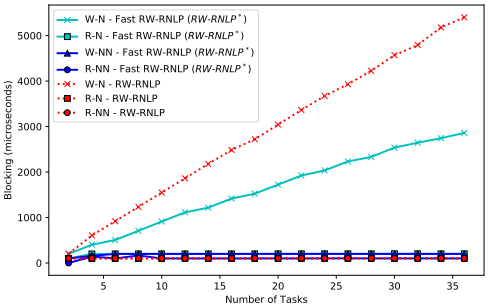

(b) Unlock overhead.




(b) Unlock overhead.



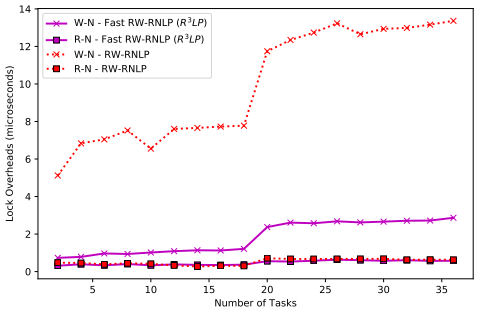

(c) Blocking.



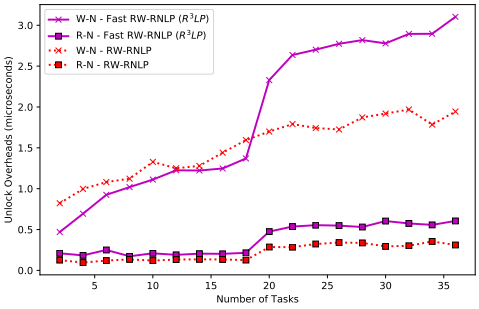

(c) Blocking.

Figure 149: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.

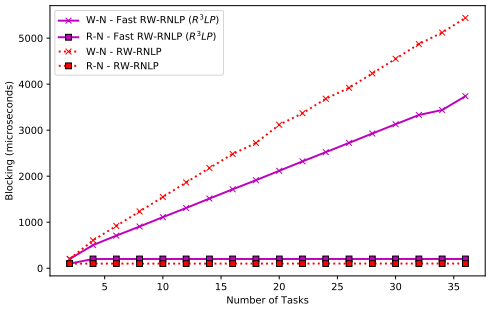
Figure 150: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

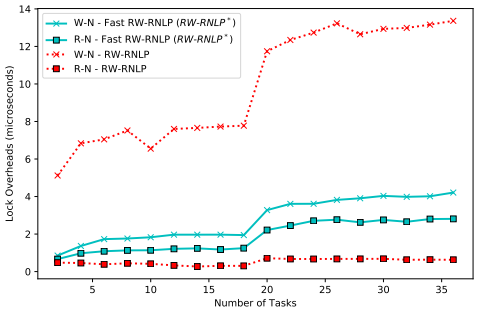



(b) Unlock overhead.

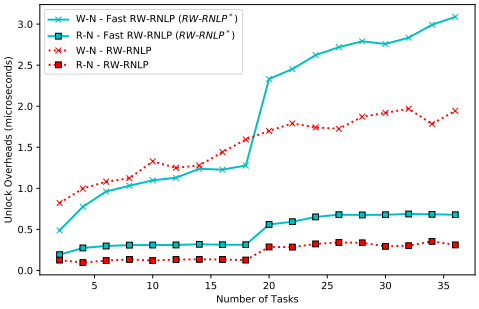



(c) Blocking.

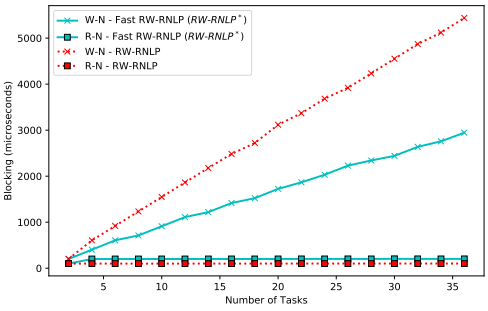
Figure 151: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

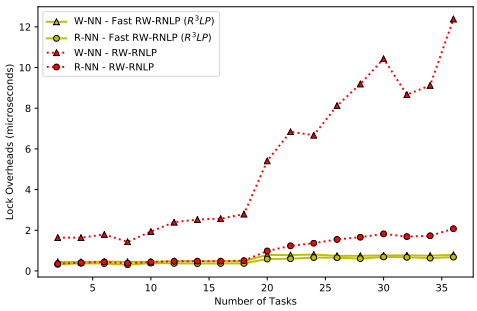



(b) Unlock overhead.

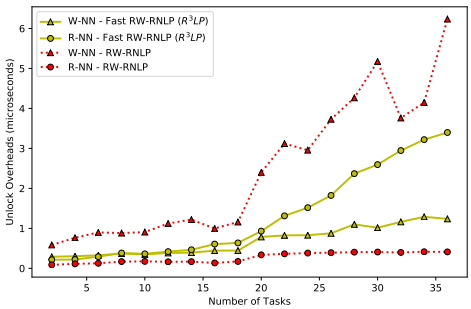



(c) Blocking.

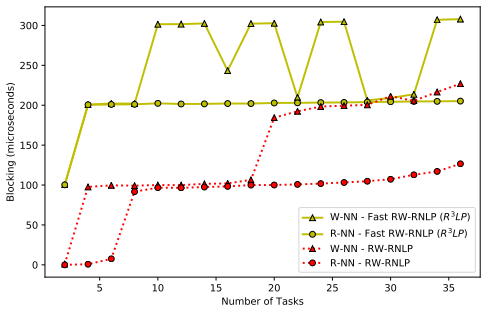
Figure 152: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

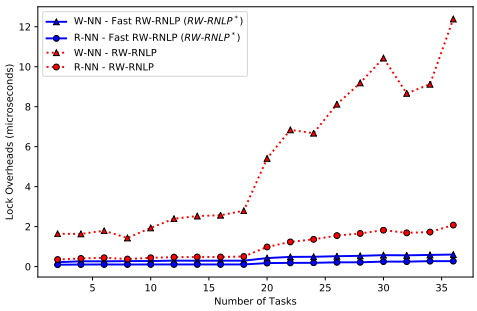



(b) Unlock overhead.

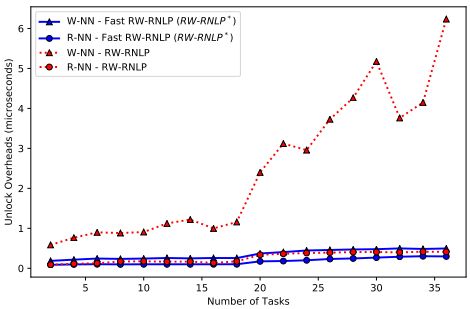



(c) Blocking.

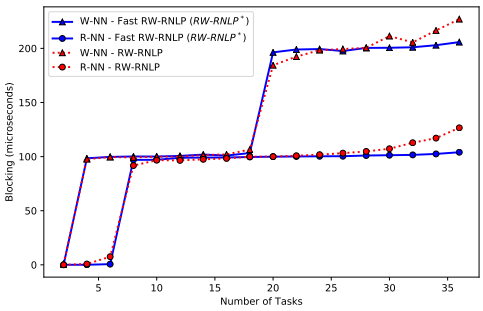
Figure 153: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

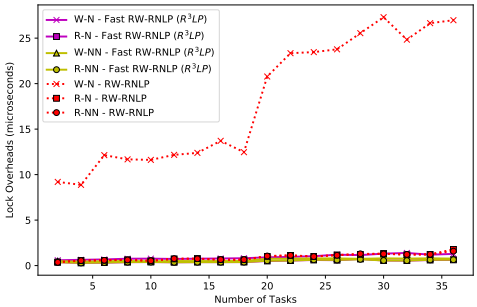



(b) Unlock overhead.

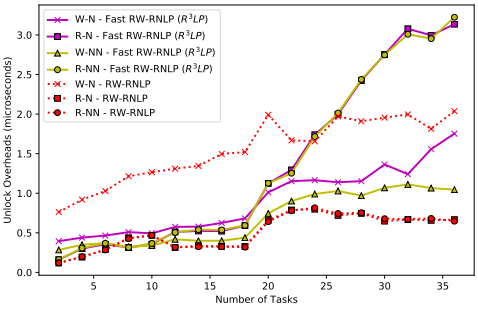



(c) Blocking.

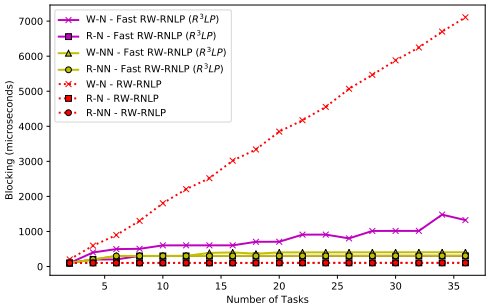
Figure 154: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

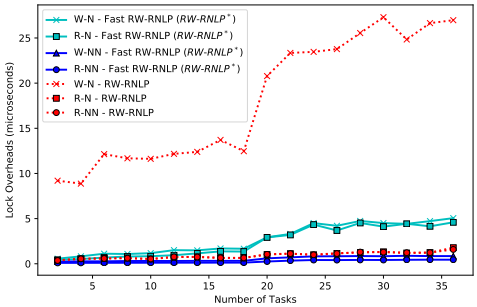



(b) Unlock overhead.

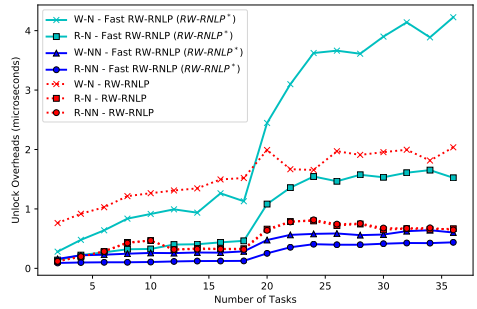



(c) Blocking.

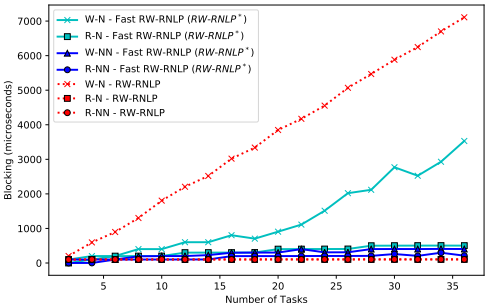
Figure 155: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

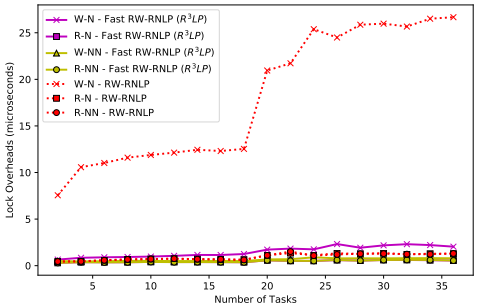



(b) Unlock overhead.

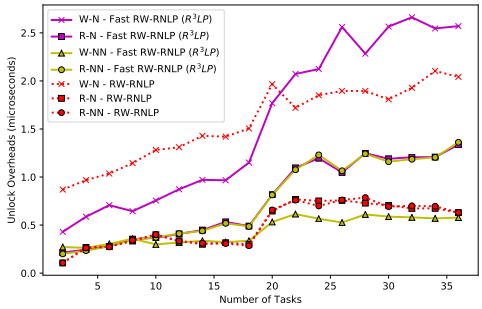



(c) Blocking.

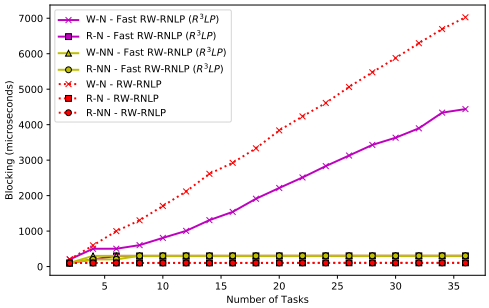
Figure 156: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

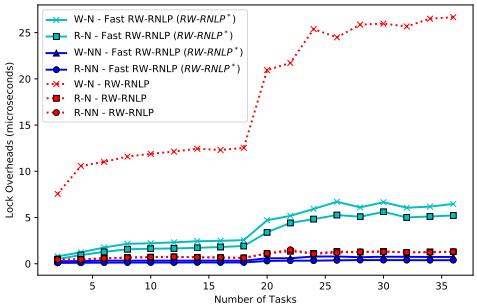



(b) Unlock overhead.

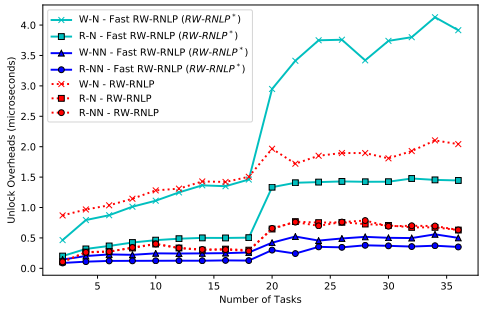



(c) Blocking.

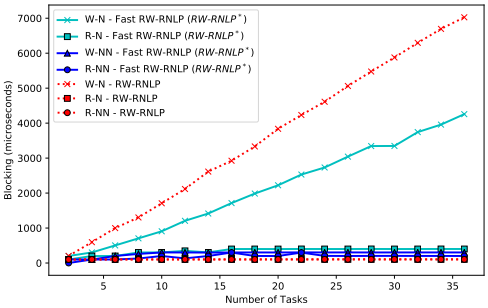
Figure 157: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

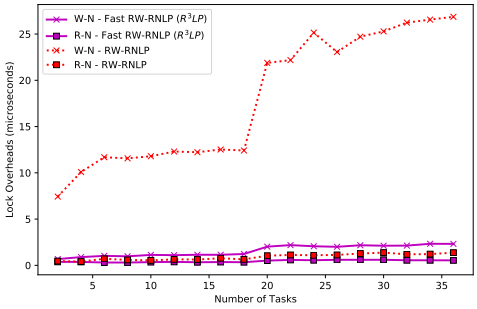



(b) Unlock overhead.

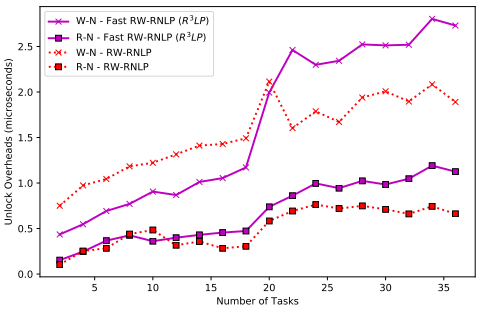



(c) Blocking.

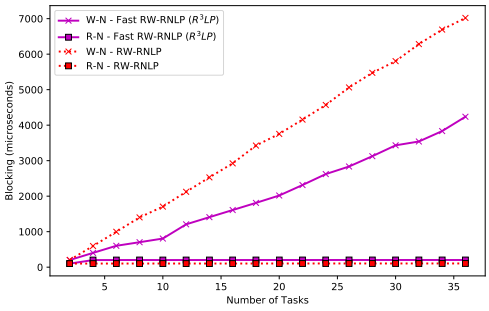
Figure 158: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

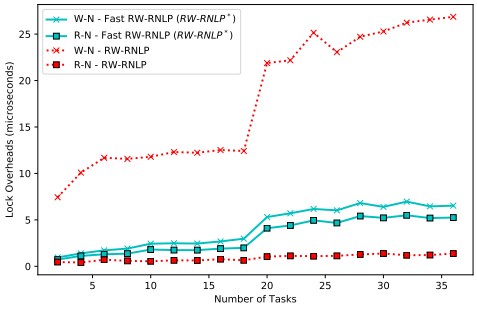



(b) Unlock overhead.

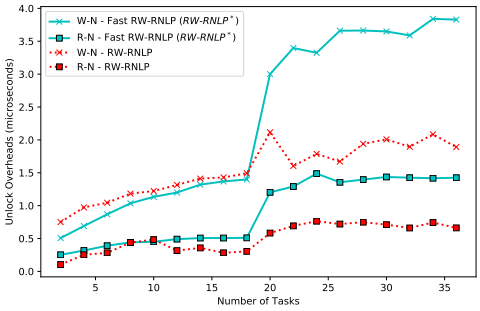



(c) Blocking.

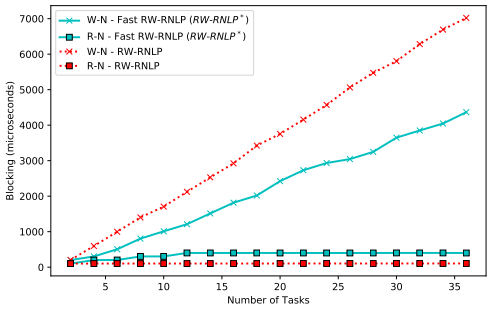
Figure 159: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

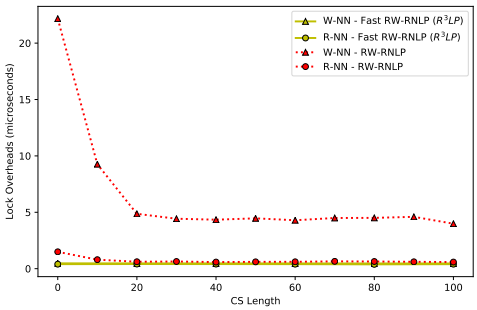



(b) Unlock overhead.

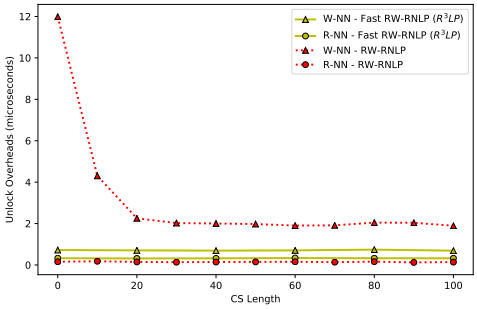



(c) Blocking.

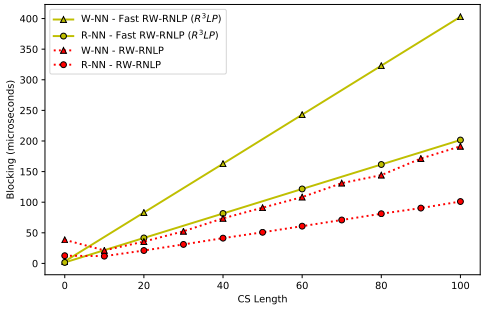
Figure 160: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

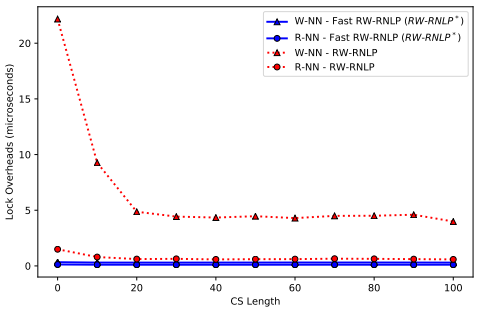



(b) Unlock overhead.

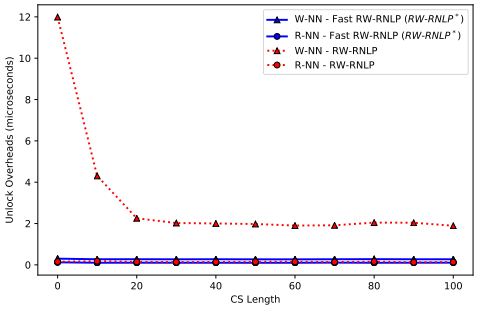



(c) Blocking.

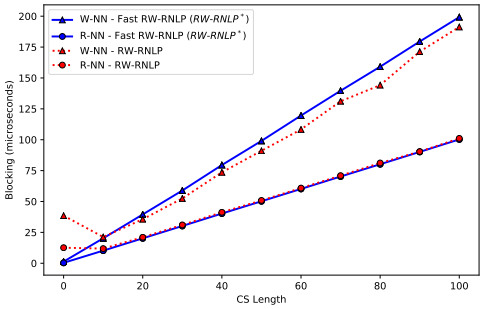
Figure 161: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

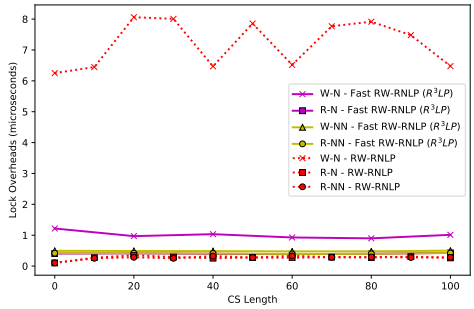



(b) Unlock overhead.

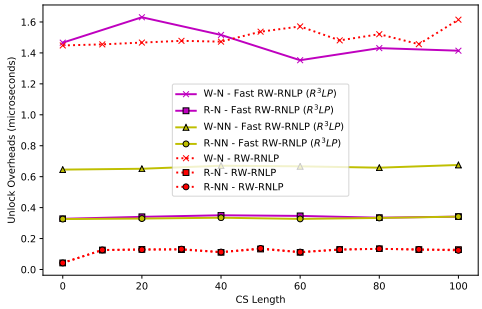



(c) Blocking.

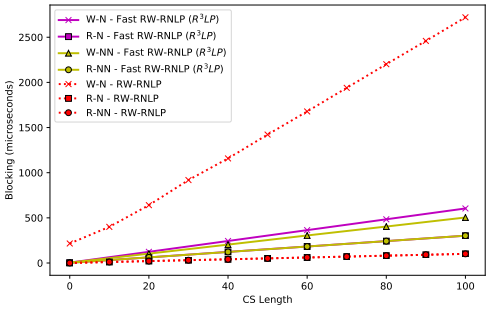
Figure 162: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

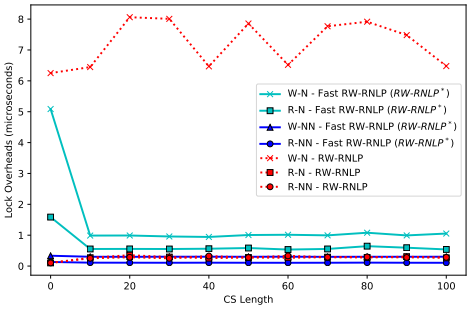



(b) Unlock overhead.

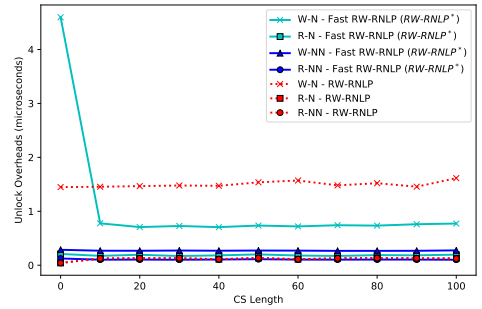



(c) Blocking.

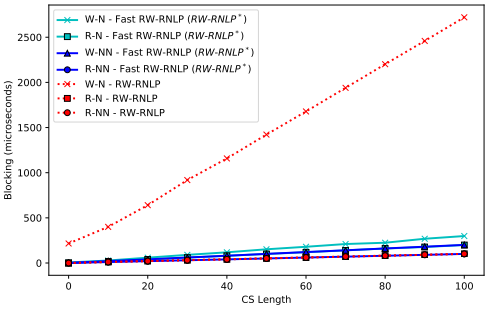
Figure 163: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

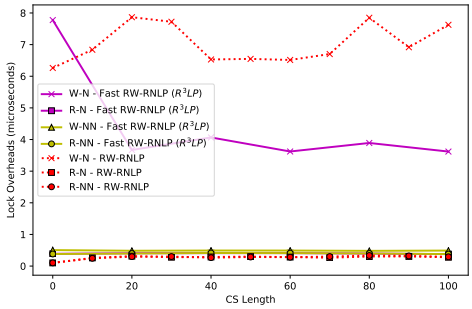



(b) Unlock overhead.

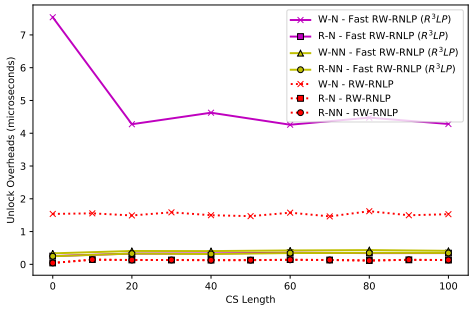



(c) Blocking.

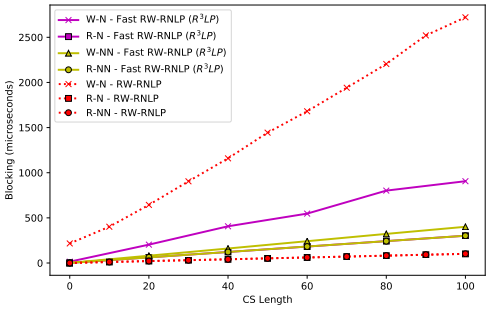
Figure 164: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

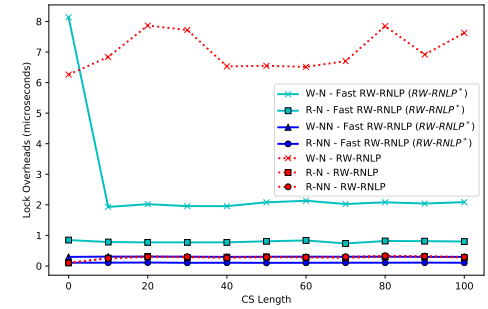



(b) Unlock overhead.

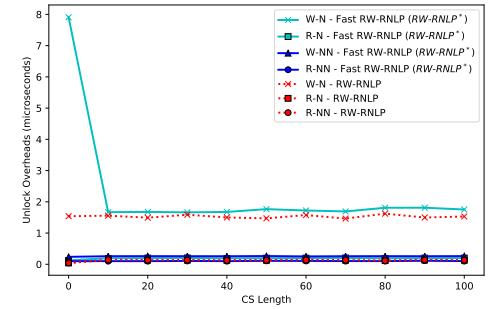



(c) Blocking.

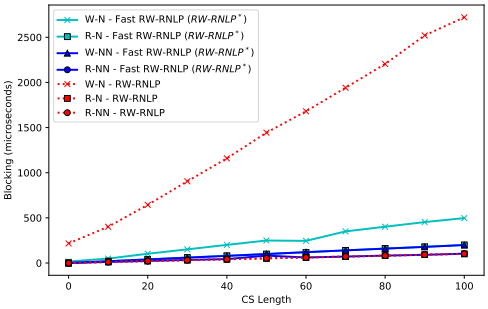
Figure 165: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

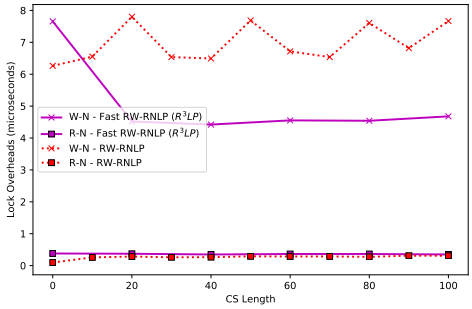



(b) Unlock overhead.

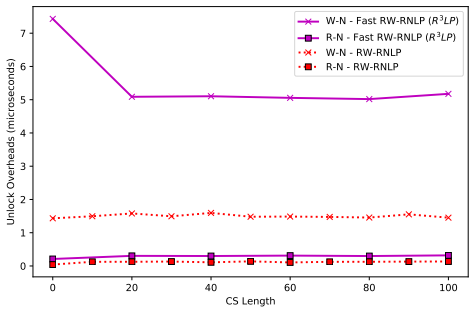



(c) Blocking.

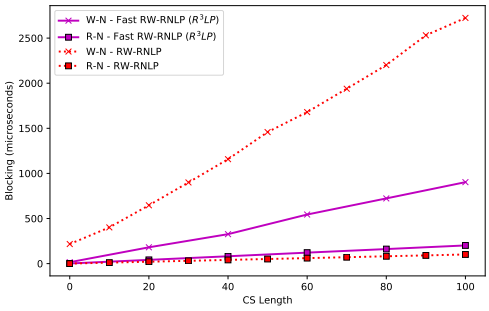
Figure 166: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

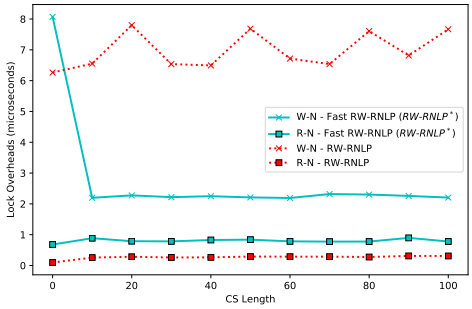



(b) Unlock overhead.

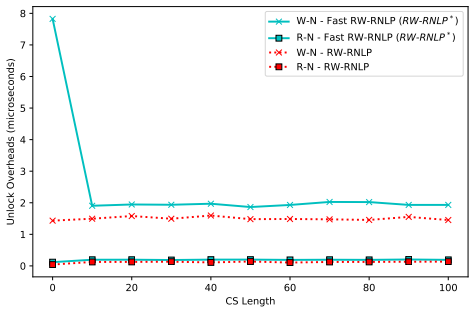



(c) Blocking.

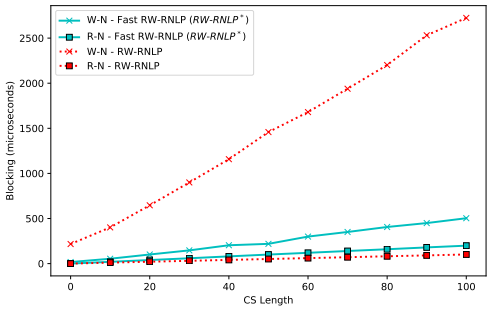
Figure 167: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

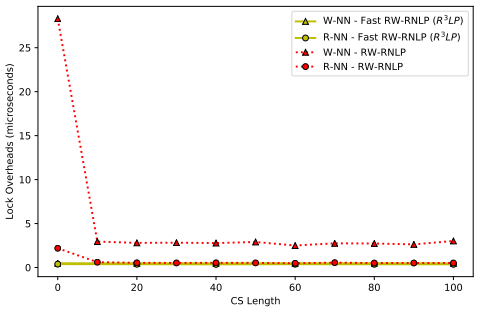



(b) Unlock overhead.

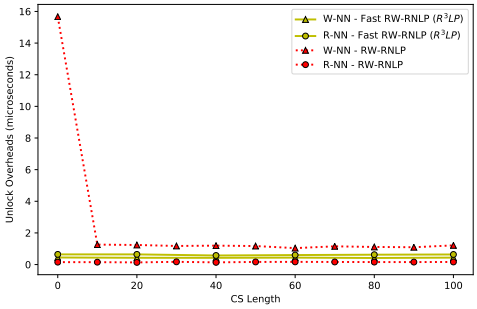



(c) Blocking.

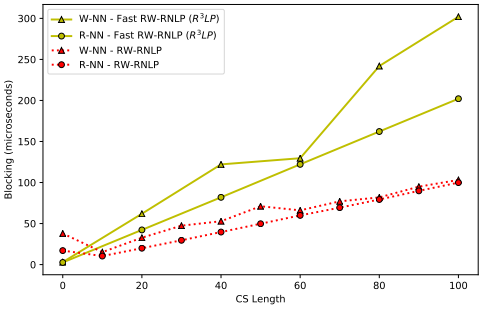
Figure 168: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

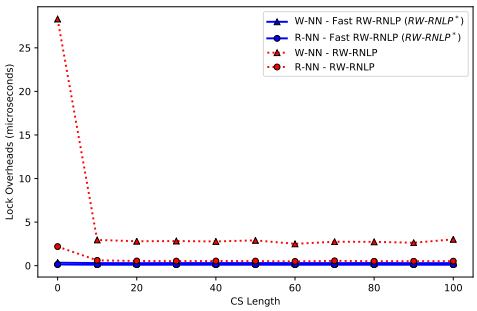



(b) Unlock overhead.

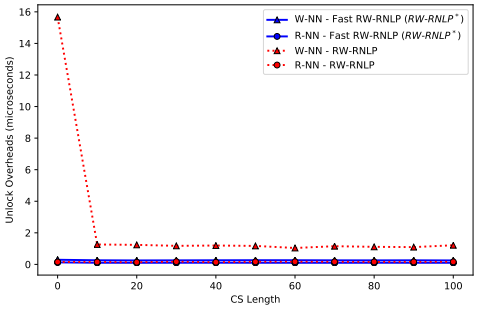



(c) Blocking.

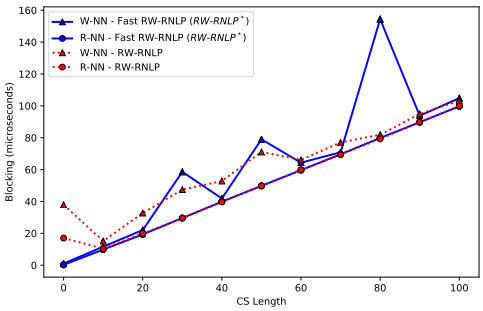
Figure 169: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

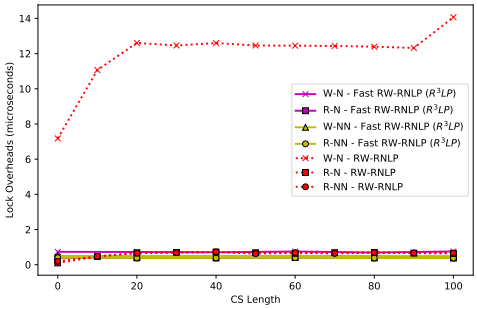



(b) Unlock overhead.

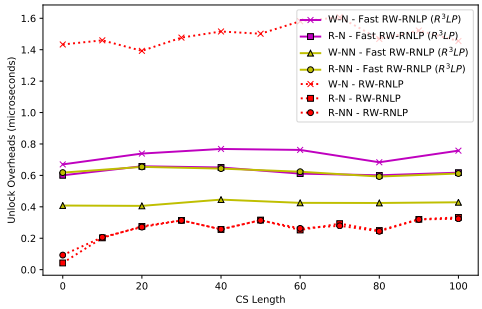



(c) Blocking.

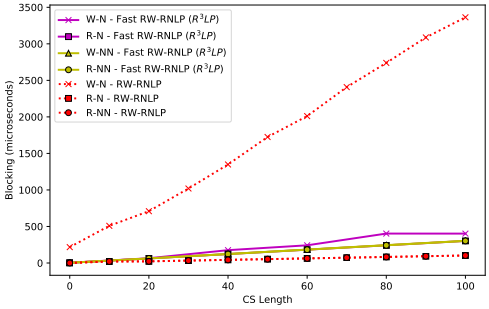
Figure 170: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

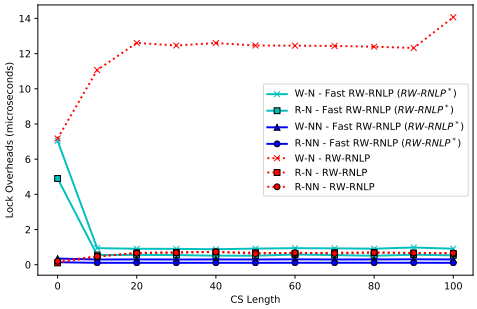



(b) Unlock overhead.

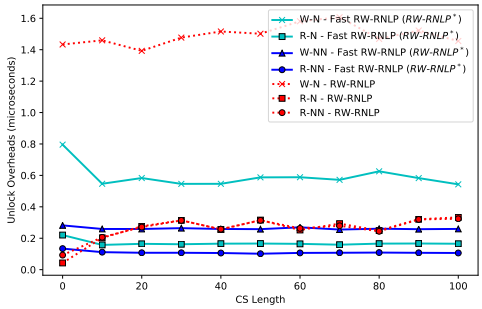



(c) Blocking.

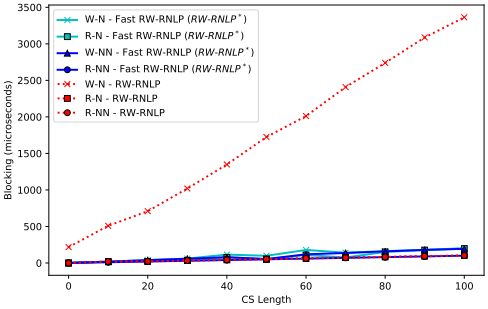
Figure 171: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

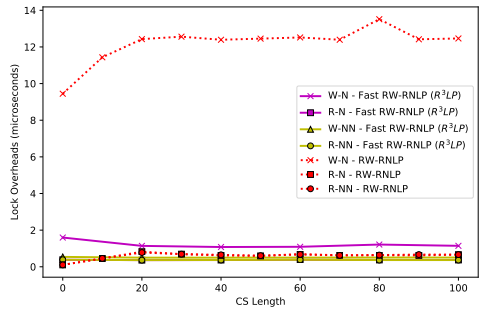



(b) Unlock overhead.

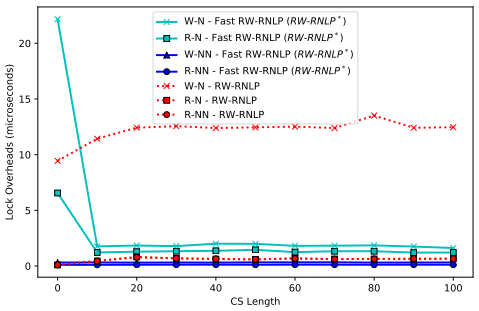



(c) Blocking.

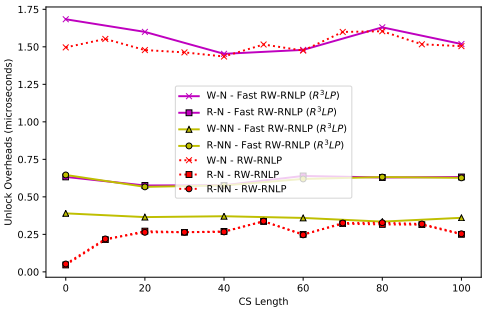
Figure 172: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.



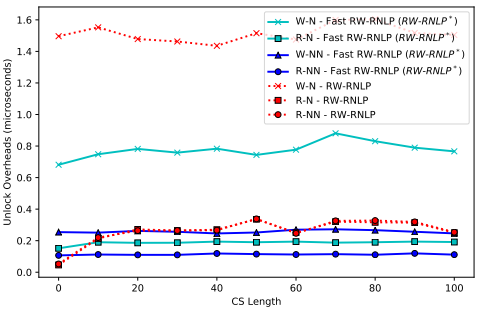

(a) Lock overhead.



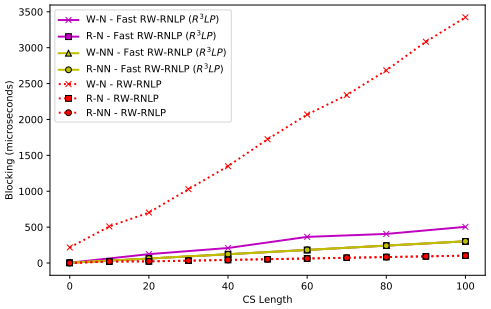

(a) Lock overhead.



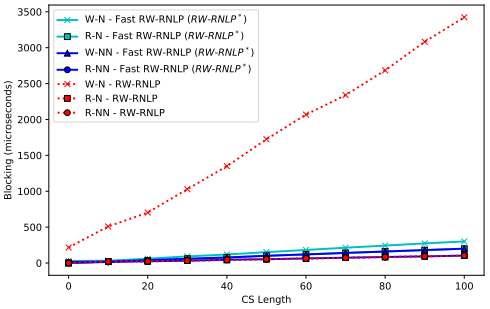

(b) Unlock overhead.




(b) Unlock overhead.



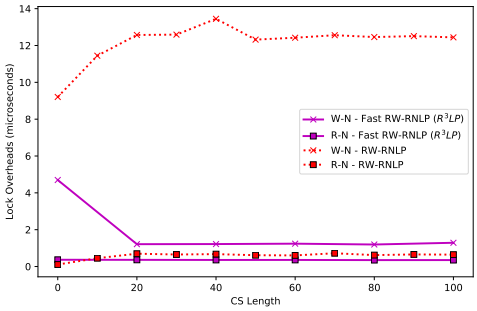

(c) Blocking.



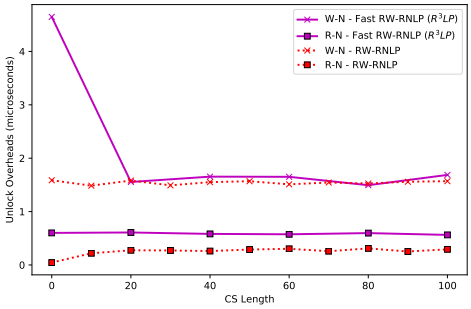

(c) Blocking.

Figure 173: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.

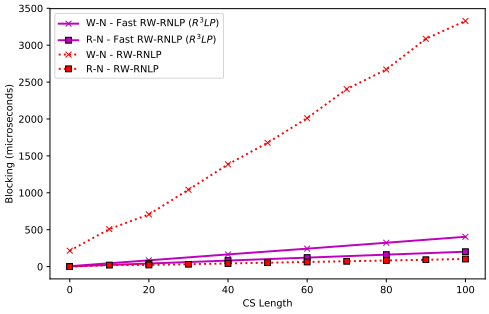
Figure 174: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

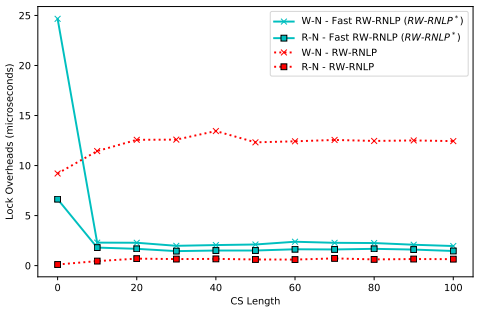



(b) Unlock overhead.

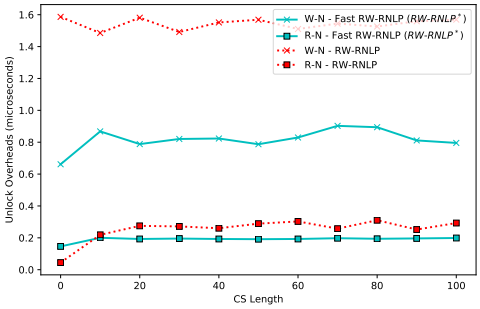



(c) Blocking.

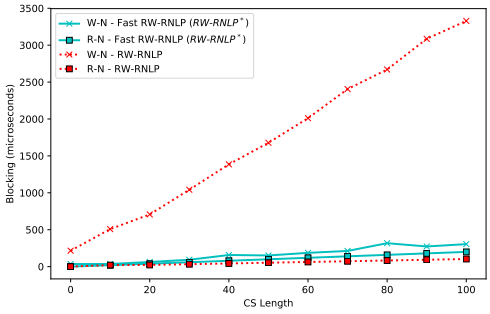
Figure 175: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

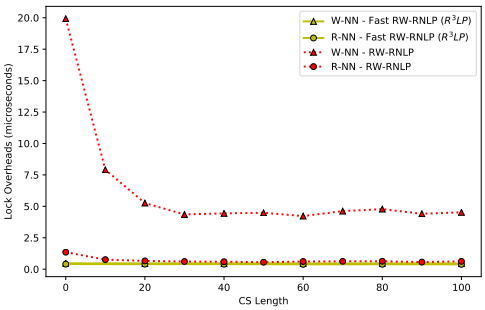



(b) Unlock overhead.

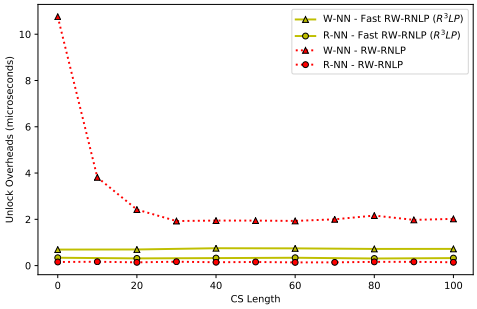



(c) Blocking.

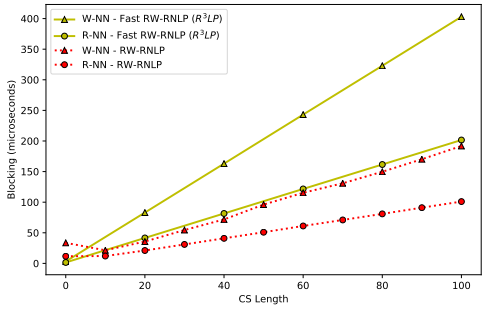
Figure 176: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

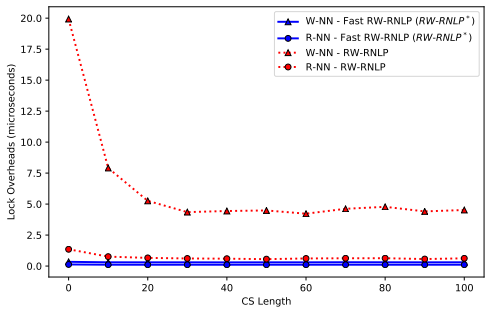



(b) Unlock overhead.

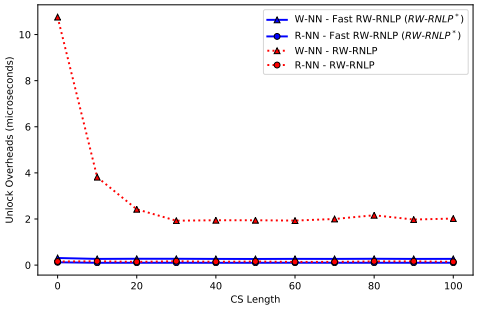



(c) Blocking.

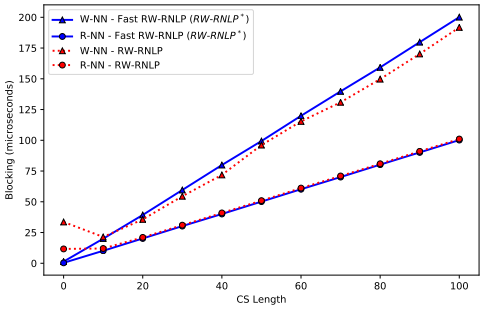
Figure 177: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

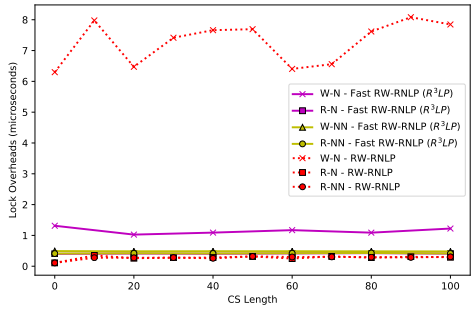



(b) Unlock overhead.

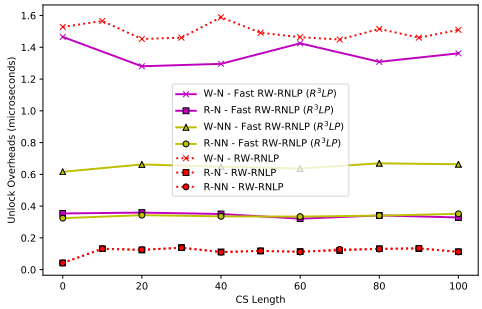



(c) Blocking.

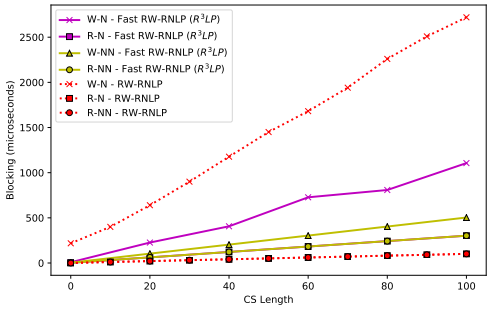
Figure 178: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

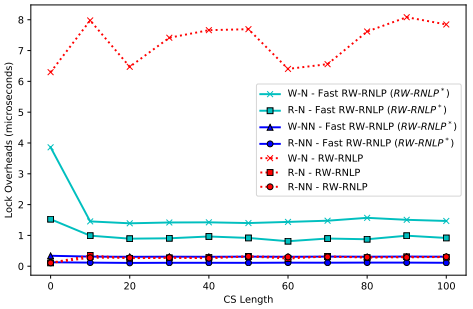



(b) Unlock overhead.

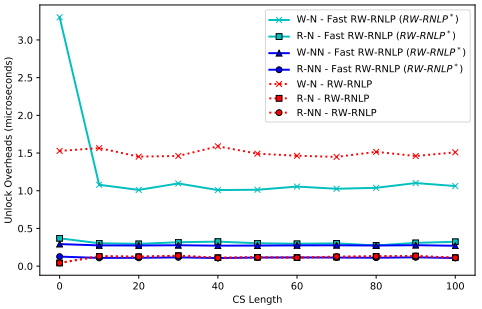



(c) Blocking.

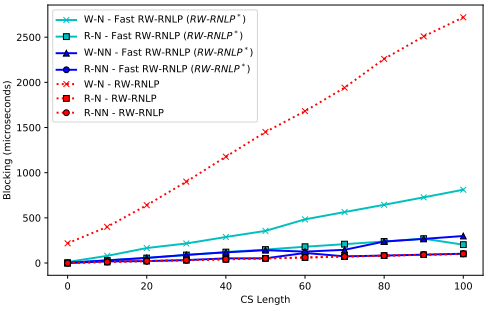
Figure 179: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

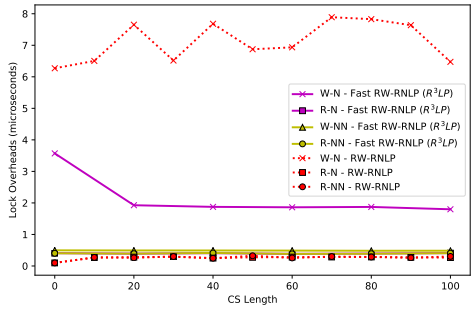



(b) Unlock overhead.

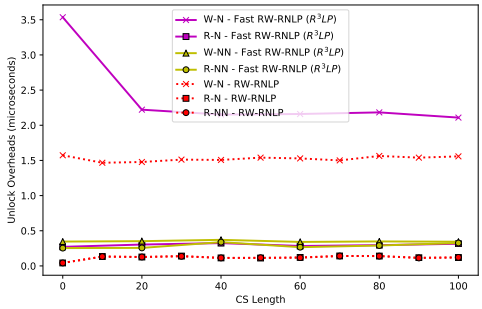



(c) Blocking.

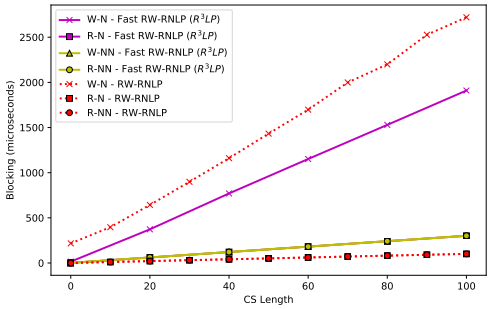
Figure 180: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

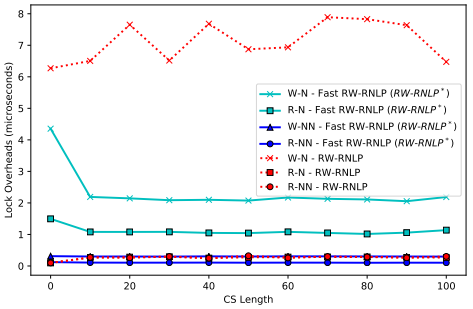



(b) Unlock overhead.

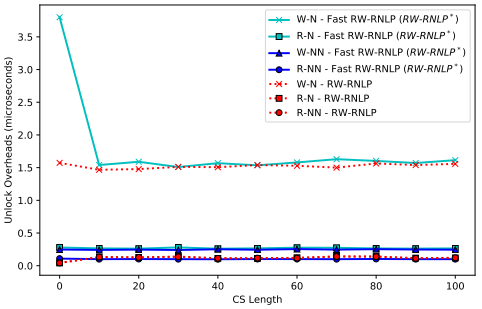



(c) Blocking.

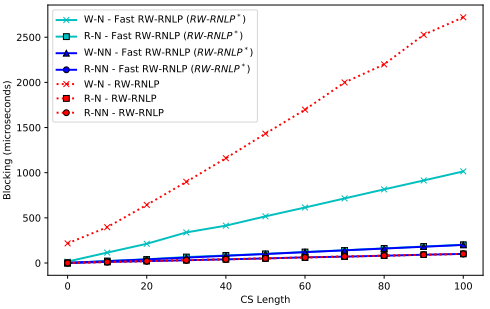
Figure 181: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

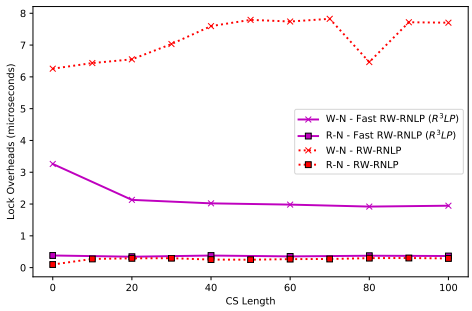



(b) Unlock overhead.

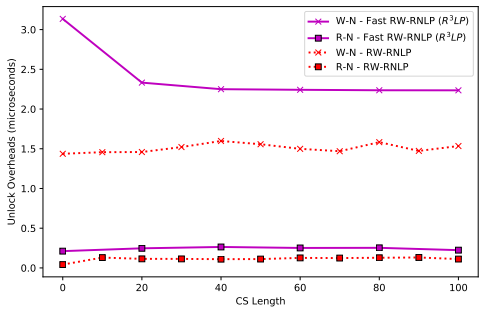



(c) Blocking.

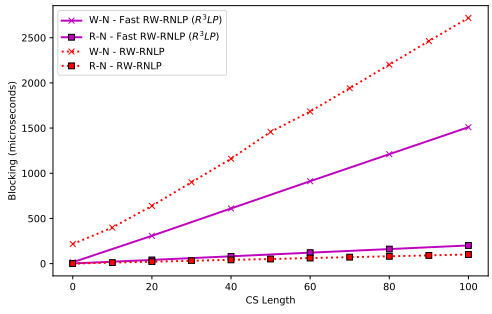
Figure 182: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

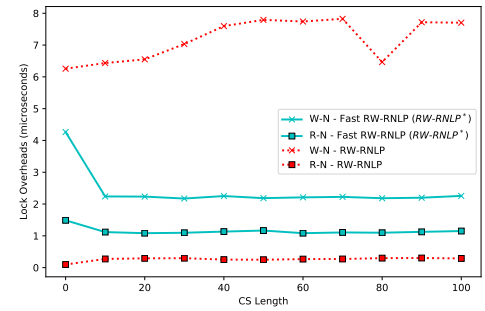



(b) Unlock overhead.

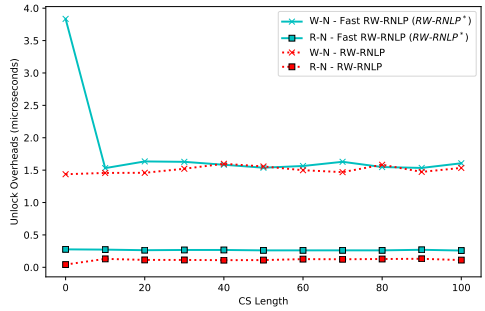



(c) Blocking.

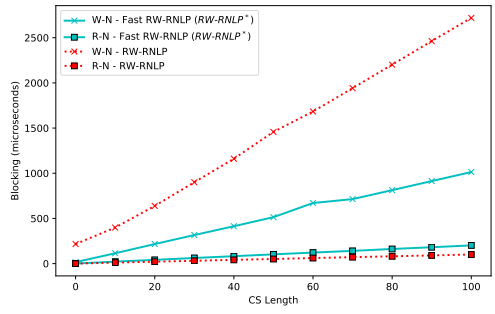
Figure 183: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

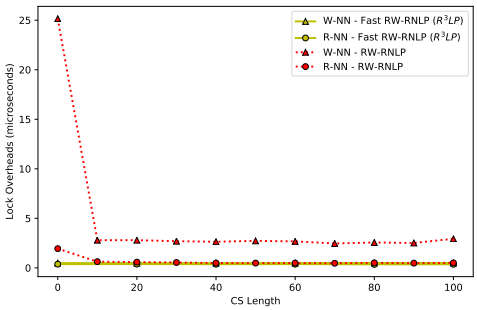



(b) Unlock overhead.

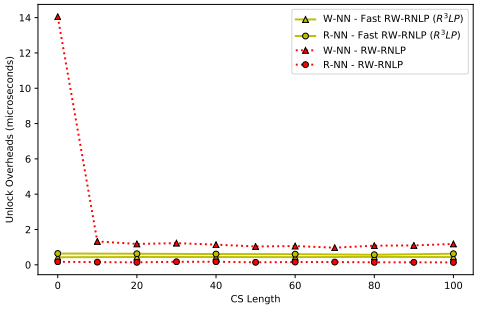



(c) Blocking.

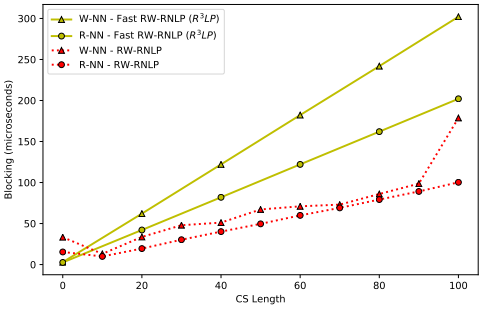
Figure 184: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

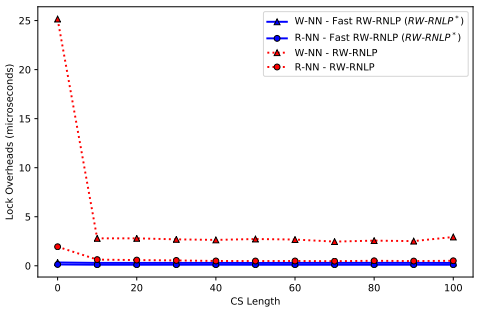



(b) Unlock overhead.

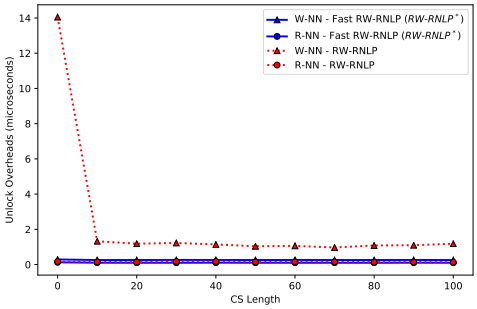



(c) Blocking.

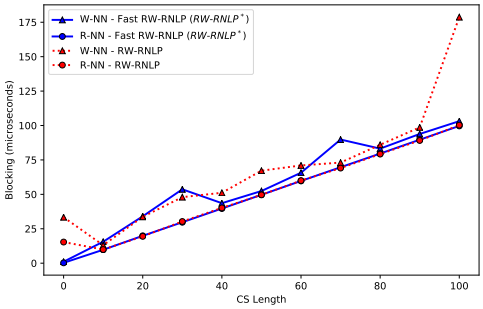
Figure 185: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

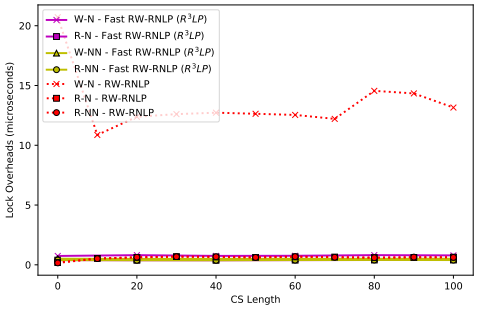



(b) Unlock overhead.

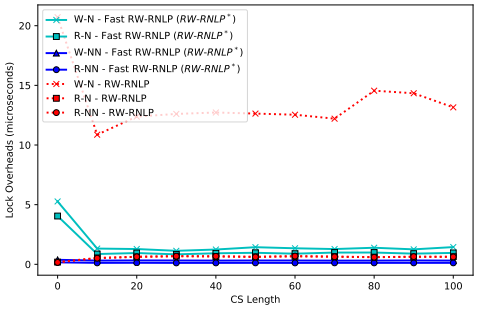



(c) Blocking.

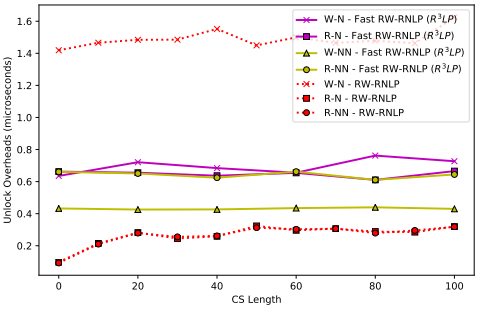
Figure 186: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 0.



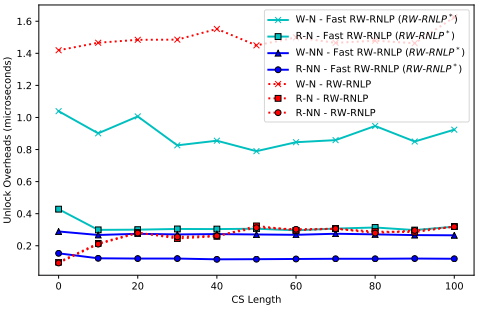

(a) Lock overhead.



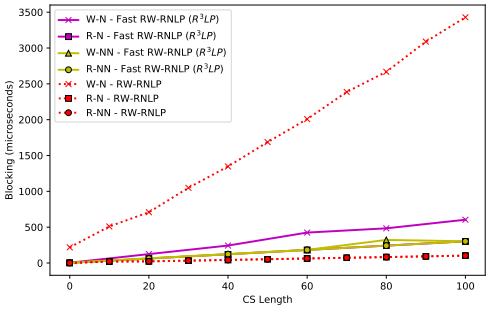

(a) Lock overhead.



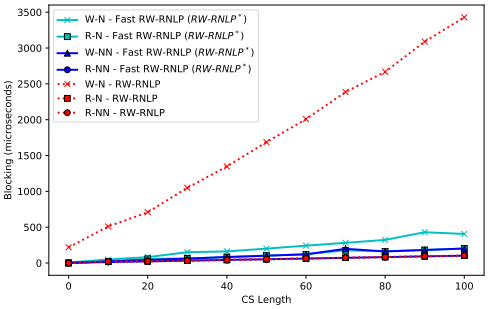

(b) Unlock overhead.




(b) Unlock overhead.



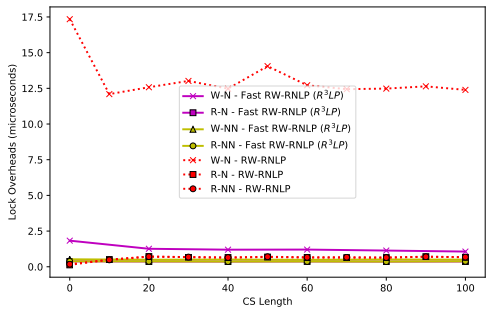

(c) Blocking.



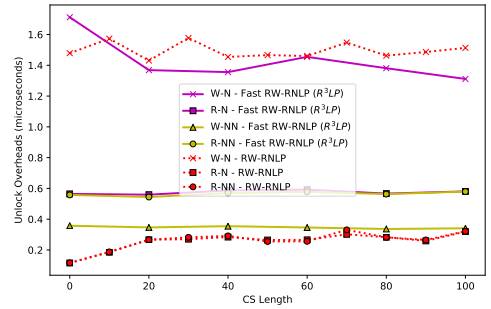

(c) Blocking.

Figure 187: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.

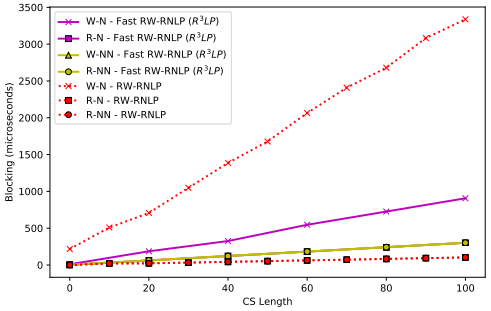
Figure 188: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

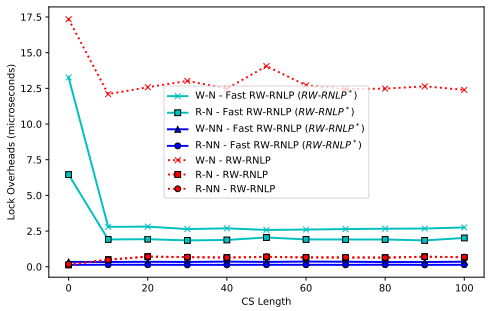



(b) Unlock overhead.

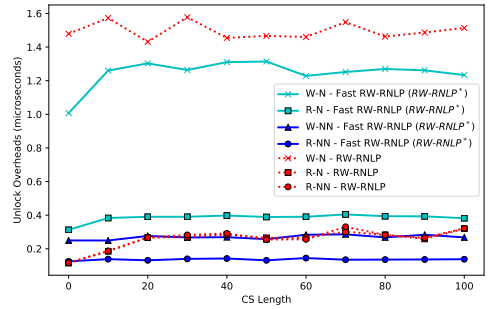



(c) Blocking.

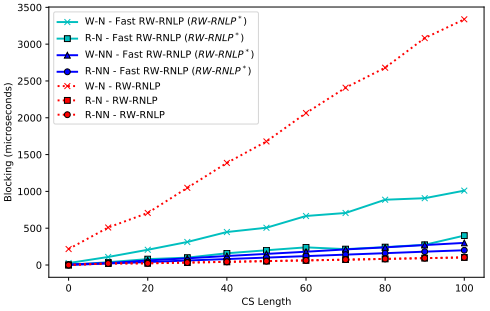
Figure 189: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

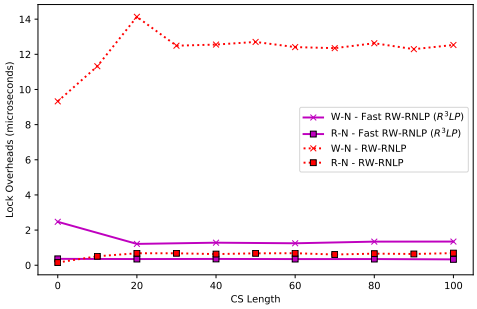



(b) Unlock overhead.

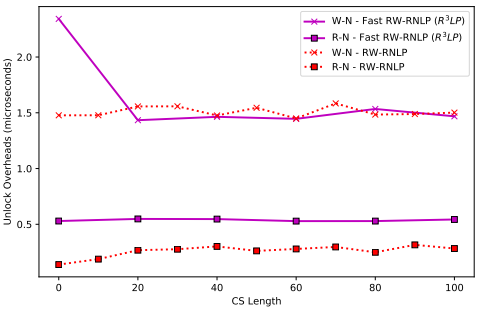



(c) Blocking.

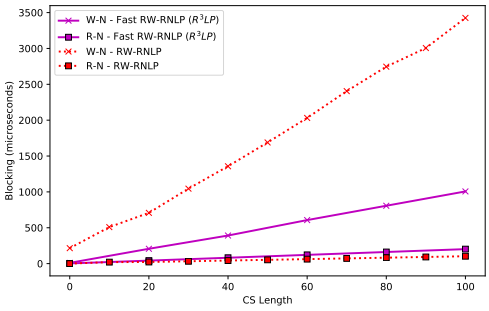
Figure 190: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

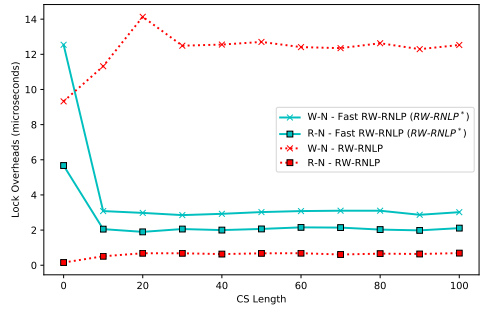



(b) Unlock overhead.

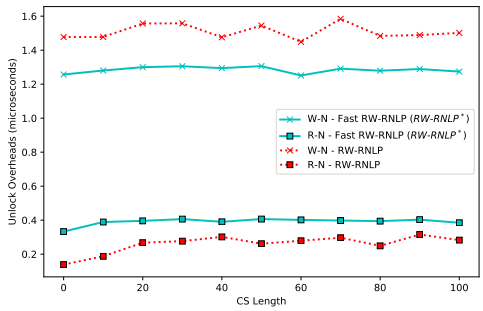



(c) Blocking.

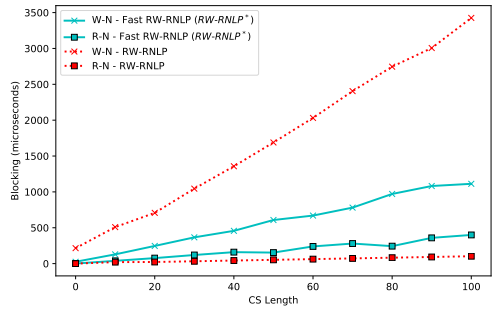
Figure 191: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

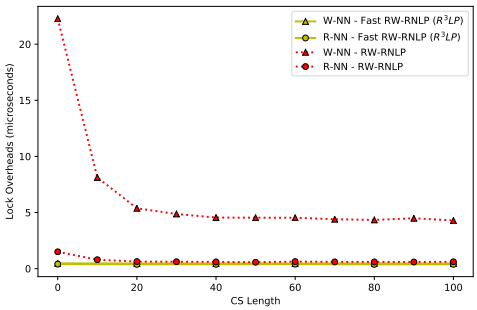



(b) Unlock overhead.

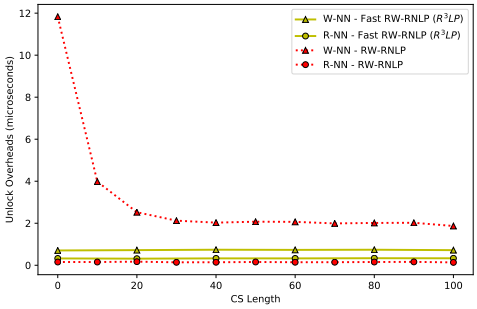



(c) Blocking.

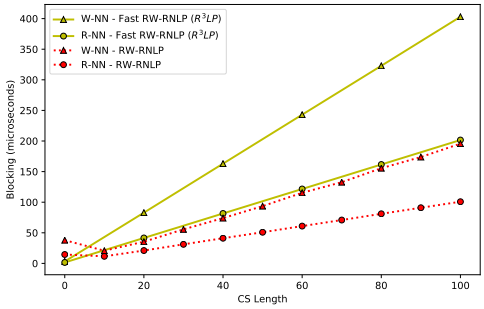
Figure 192: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

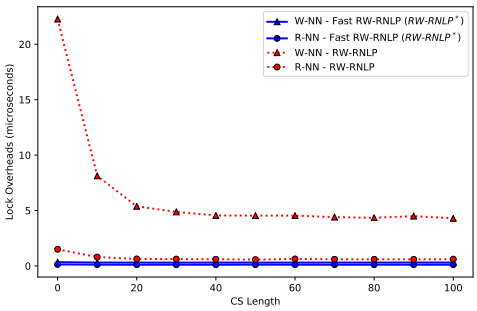



(b) Unlock overhead.

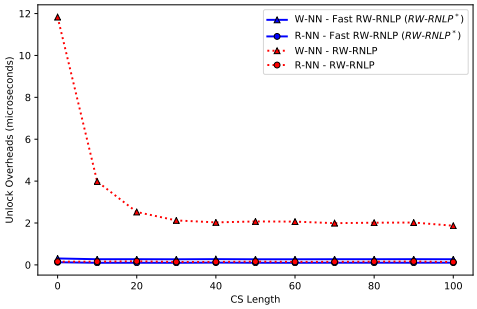



(c) Blocking.

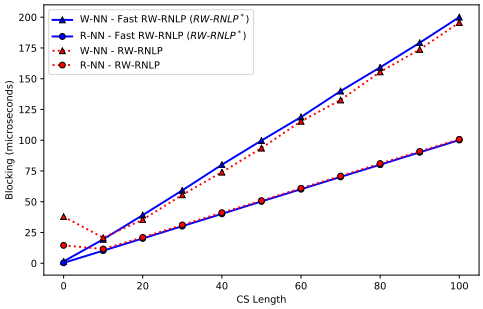
Figure 193: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

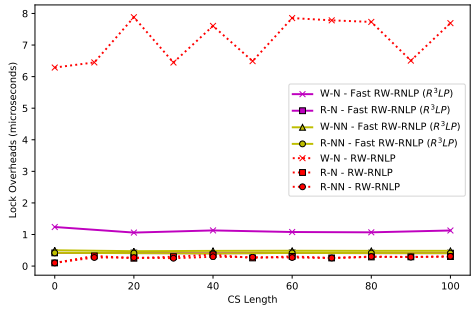



(b) Unlock overhead.

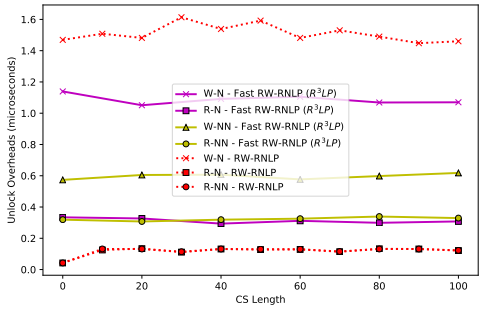



(c) Blocking.

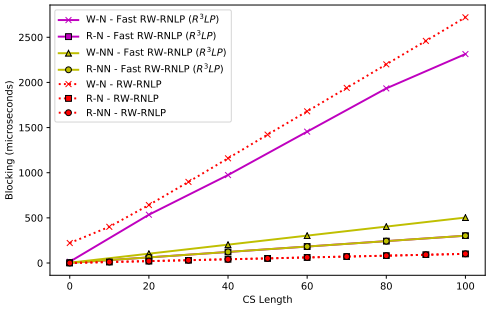
Figure 194: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

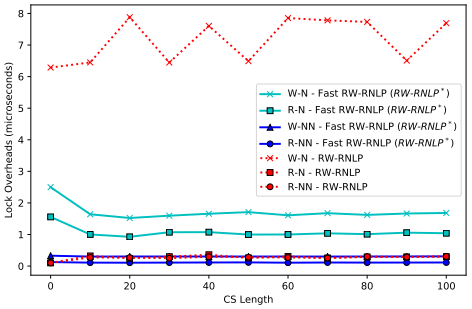



(b) Unlock overhead.

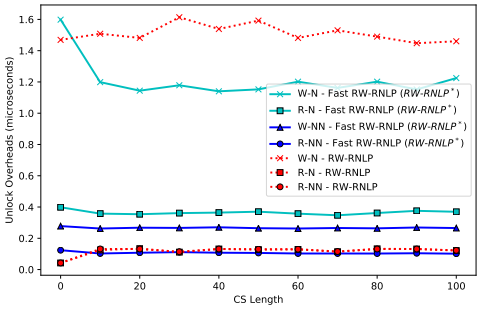



(c) Blocking.

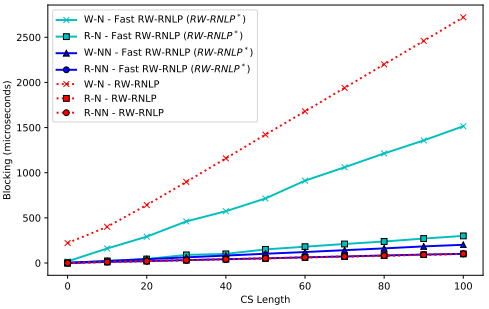
Figure 195: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

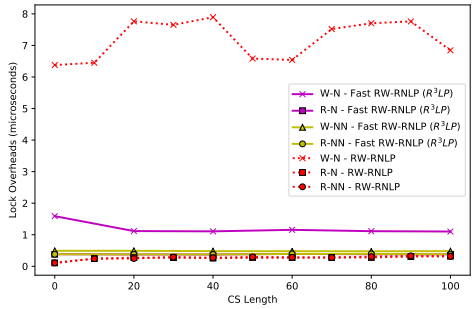



(b) Unlock overhead.

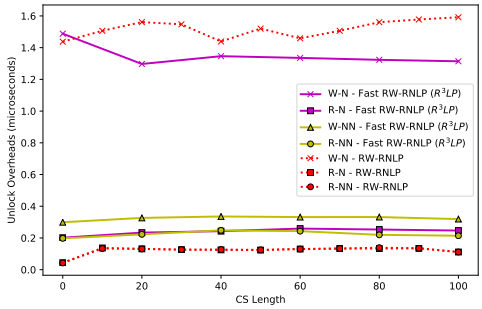



(c) Blocking.

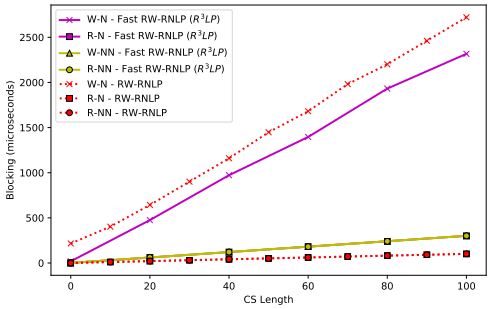
Figure 196: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

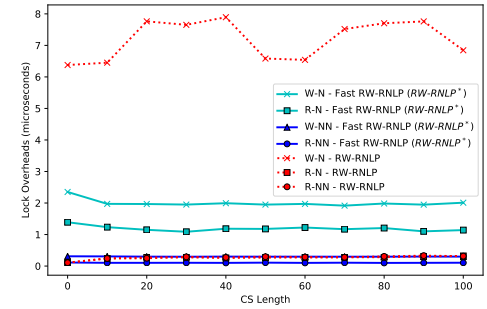



(b) Unlock overhead.

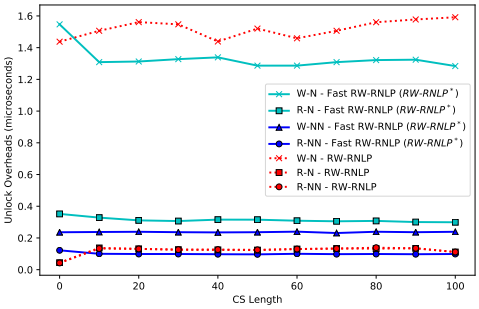



(c) Blocking.

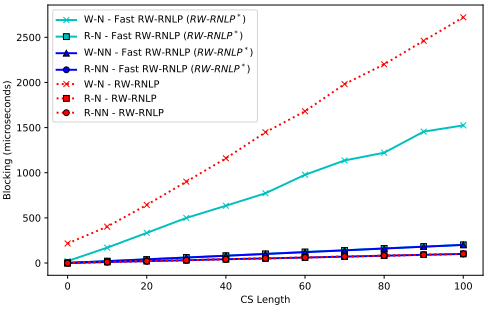
Figure 197: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

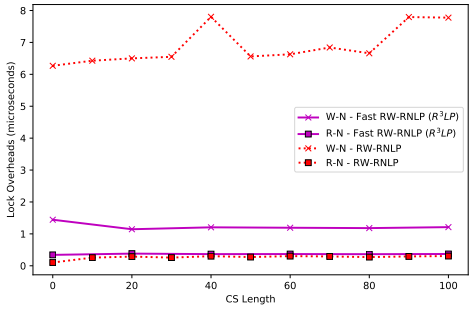



(b) Unlock overhead.

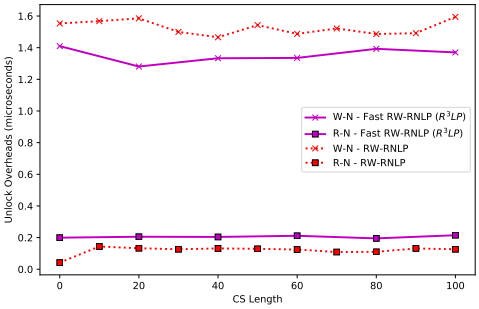



(c) Blocking.

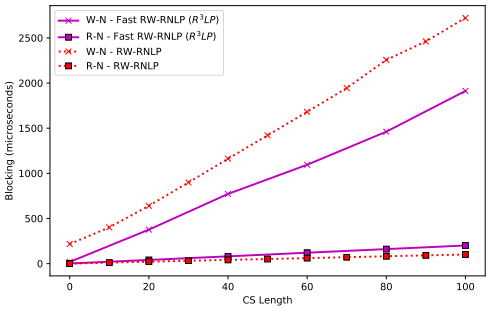
Figure 198: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

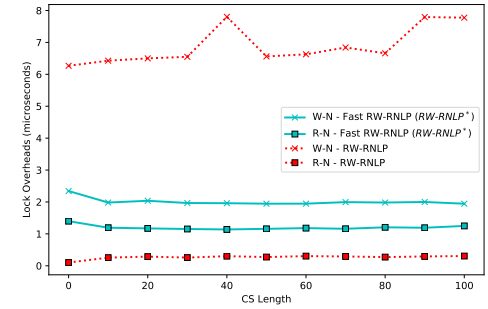



(b) Unlock overhead.

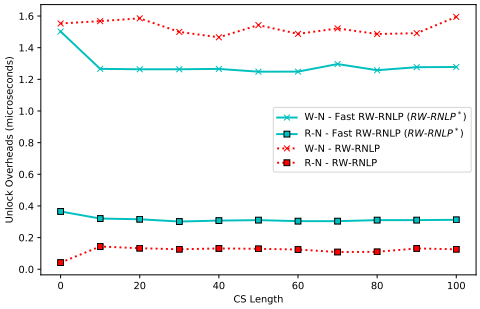



(c) Blocking.

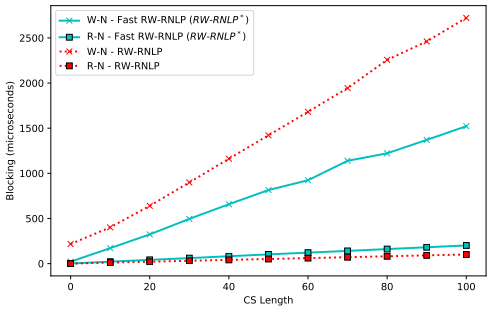
Figure 199: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

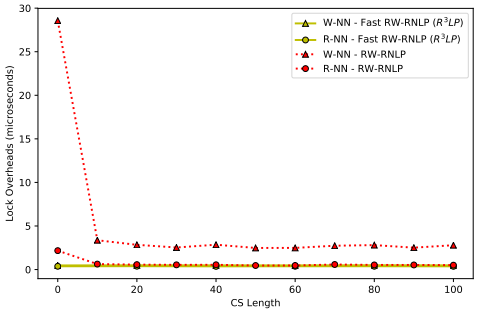



(b) Unlock overhead.

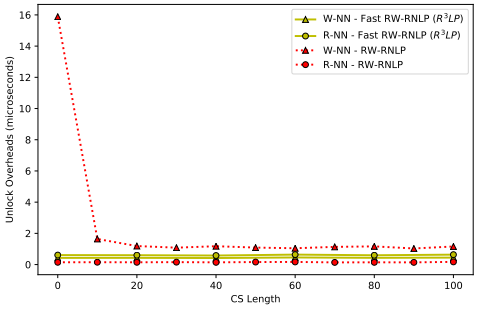



(c) Blocking.

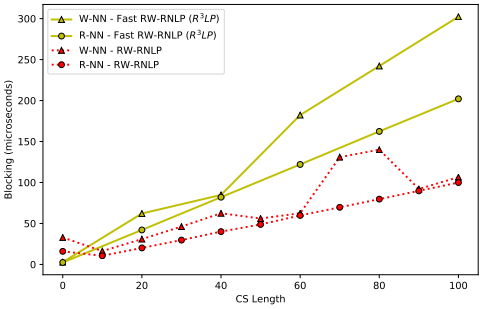
Figure 200: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

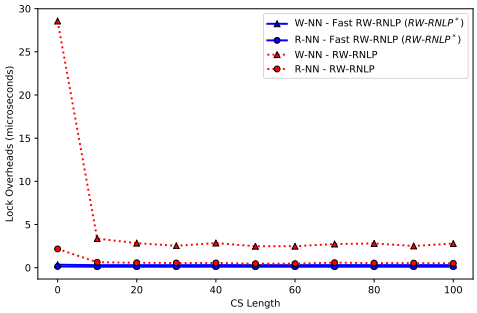



(b) Unlock overhead.

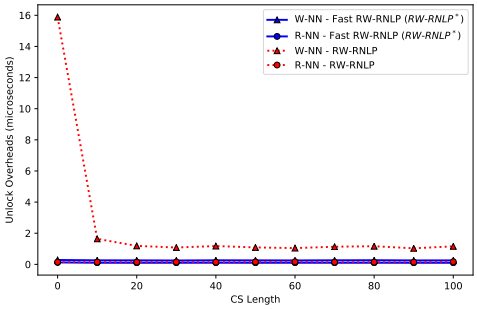



(c) Blocking.

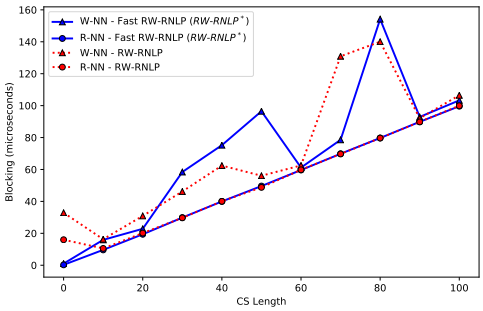
Figure 201: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

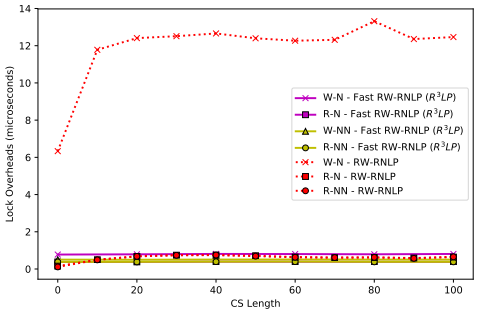



(b) Unlock overhead.

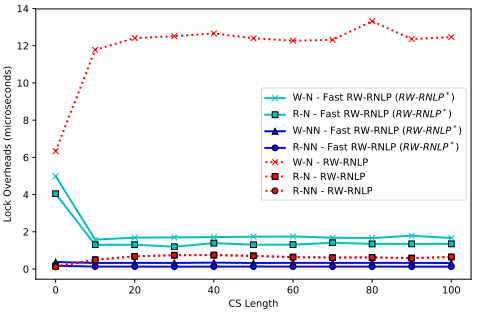



(c) Blocking.

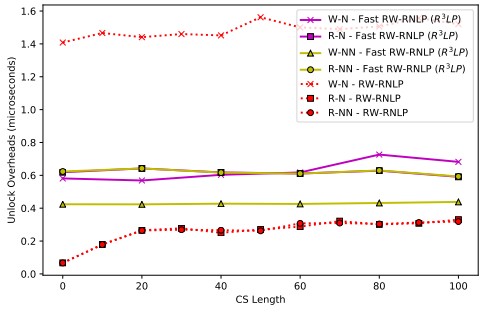
Figure 202: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 0.



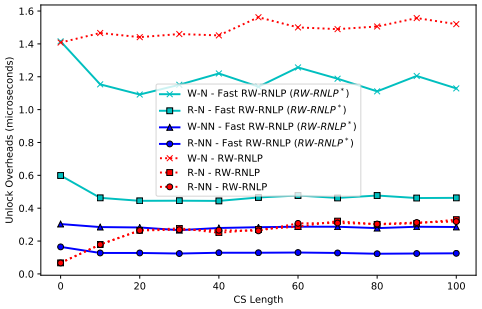

(a) Lock overhead.



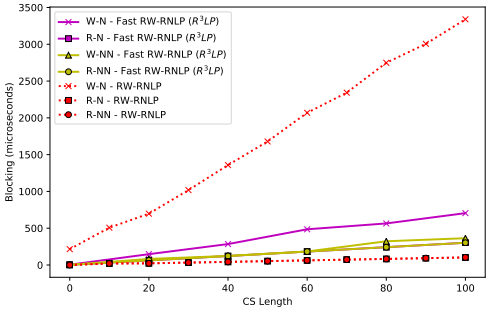

(a) Lock overhead.



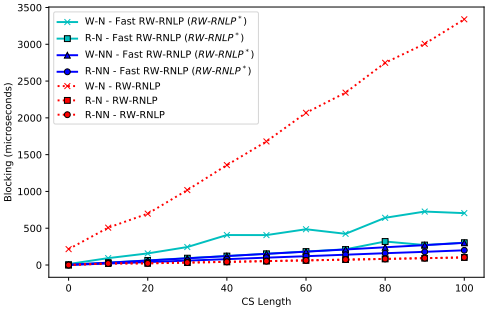

(b) Unlock overhead.




(b) Unlock overhead.



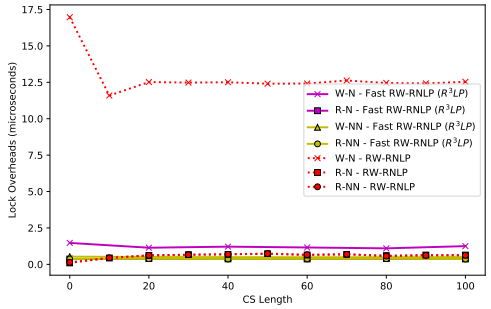

(c) Blocking.



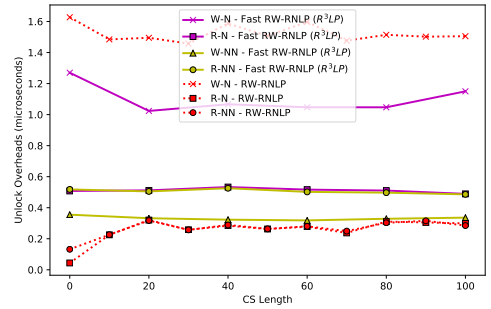

(c) Blocking.

Figure 203: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.

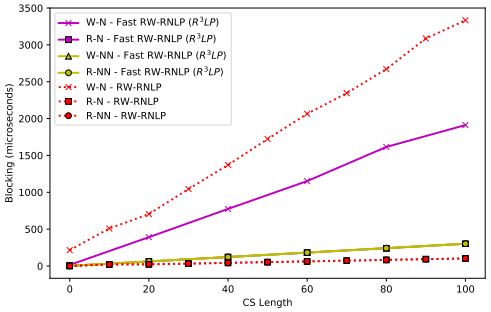
Figure 204: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

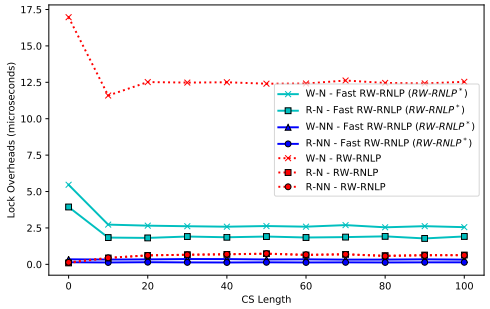



(b) Unlock overhead.

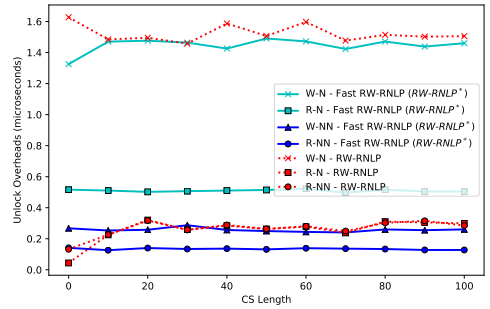



(c) Blocking.

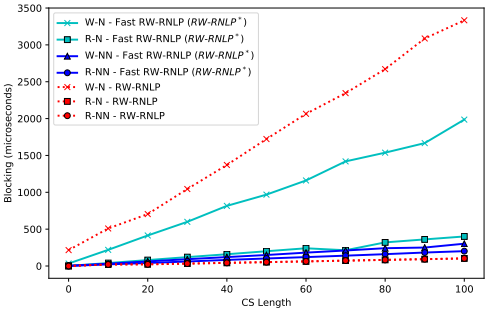
Figure 205: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

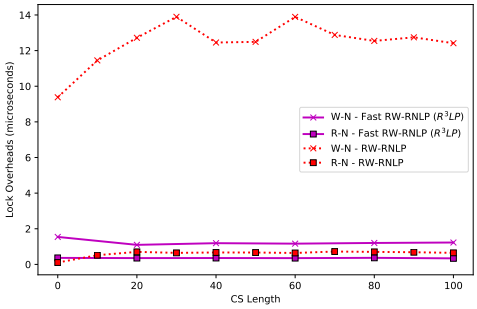



(b) Unlock overhead.

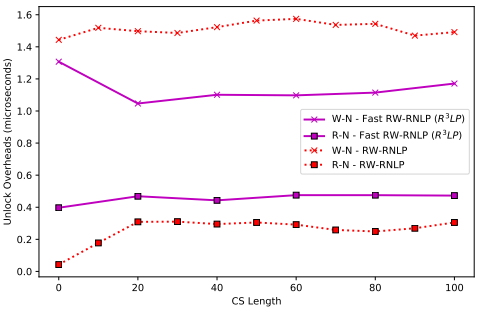



(c) Blocking.

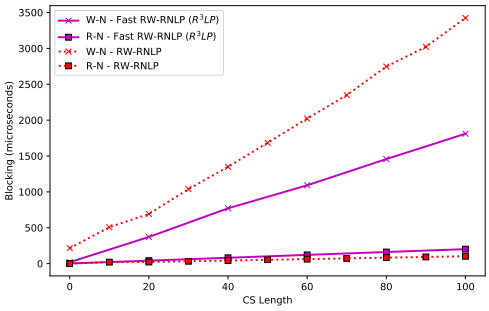
Figure 206: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

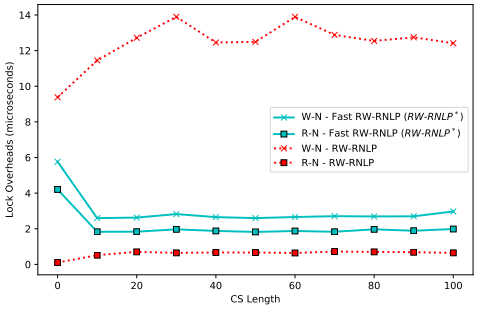



(b) Unlock overhead.

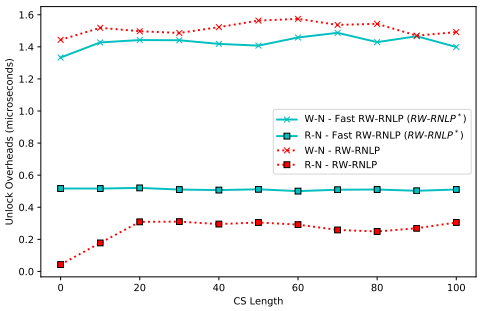



(c) Blocking.

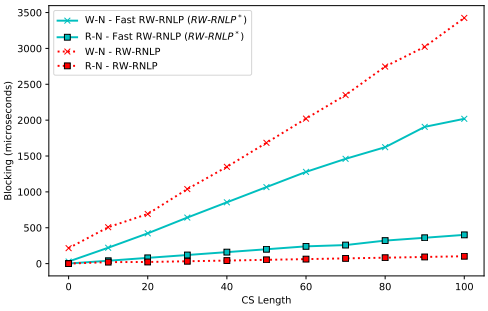
Figure 207: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

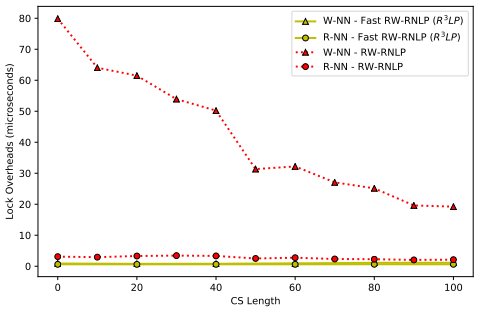



(b) Unlock overhead.

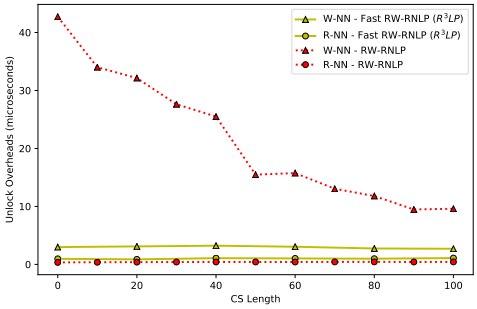



(c) Blocking.

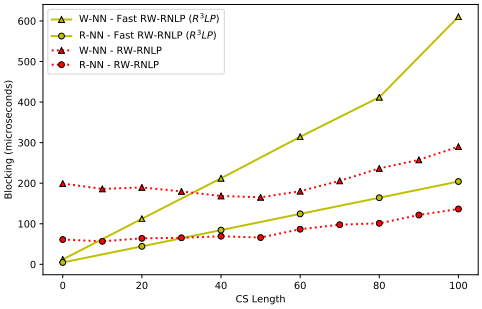
Figure 208: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

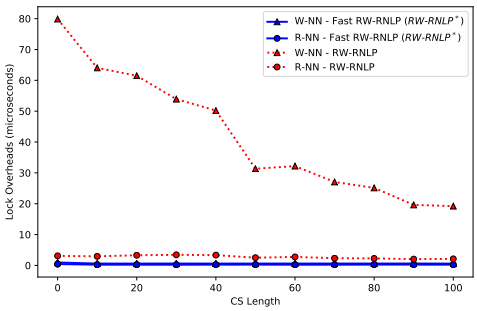



(b) Unlock overhead.

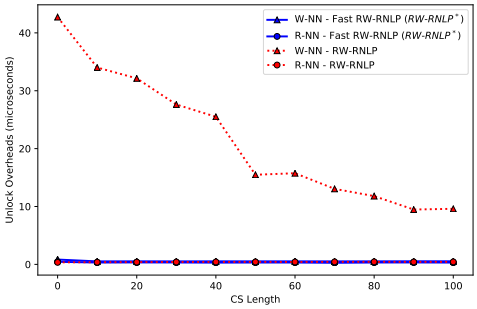



(c) Blocking.

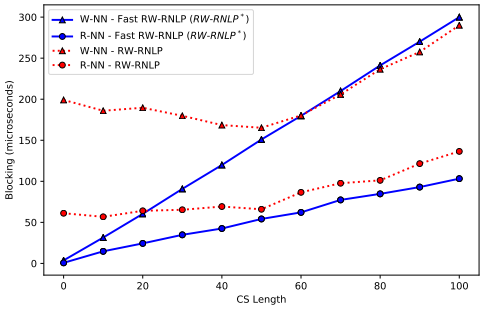
Figure 209: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

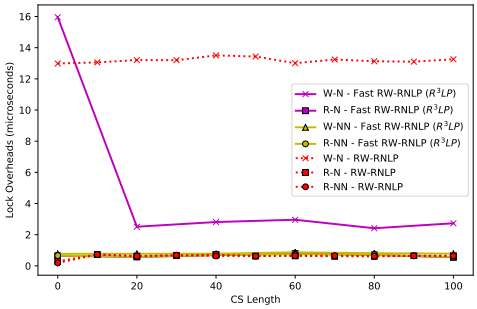



(b) Unlock overhead.

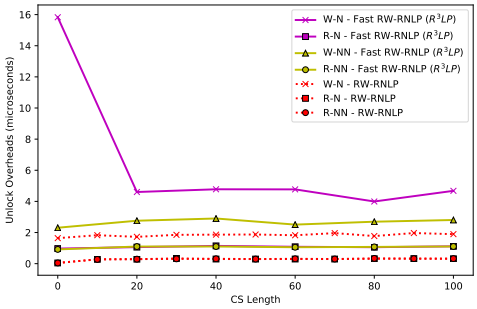



(c) Blocking.

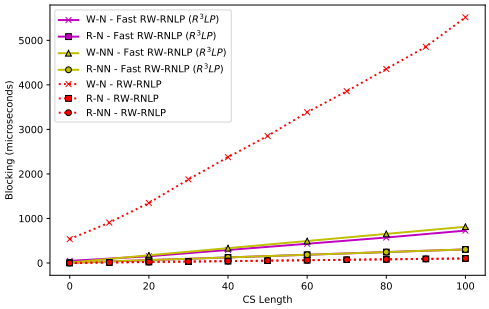
Figure 210: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

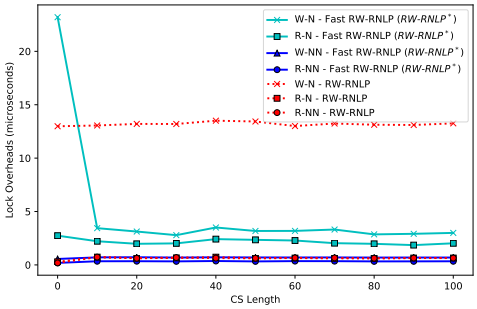



(b) Unlock overhead.

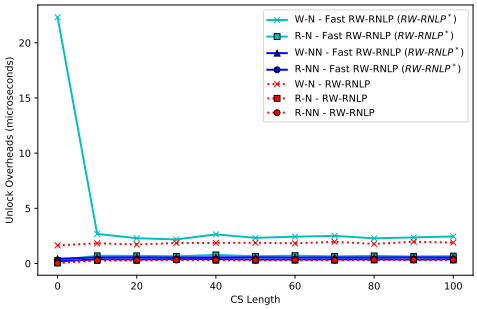



(c) Blocking.

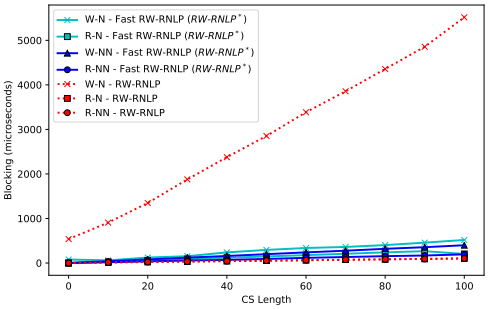
Figure 211: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

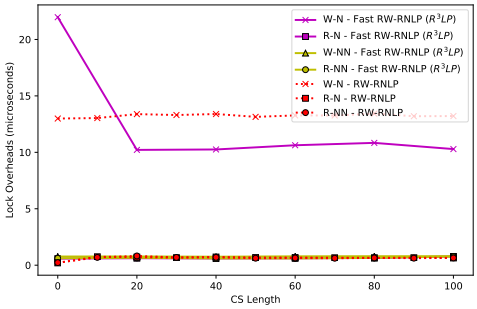



(b) Unlock overhead.

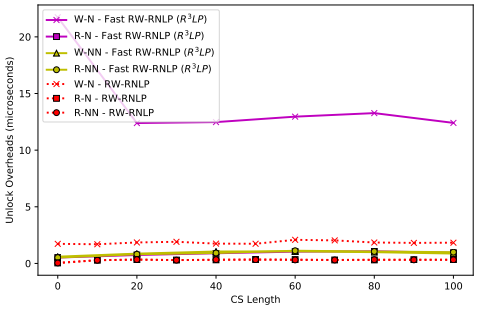



(c) Blocking.

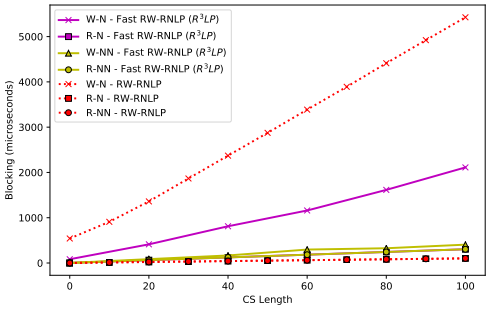
Figure 212: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

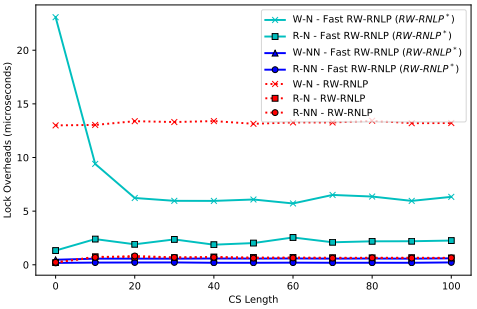



(b) Unlock overhead.

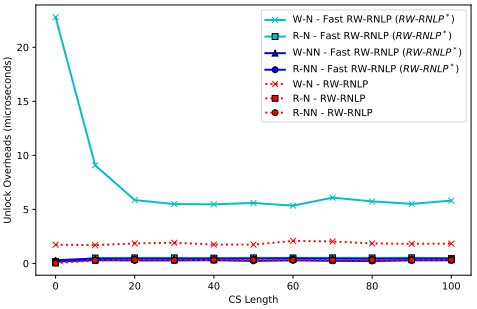



(c) Blocking.

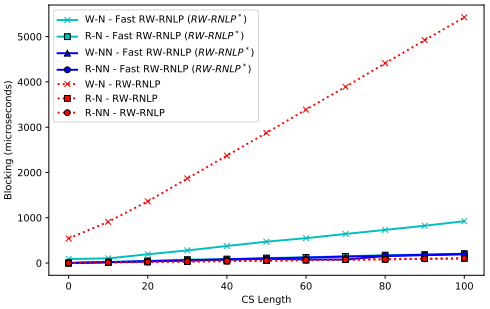
Figure 213: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

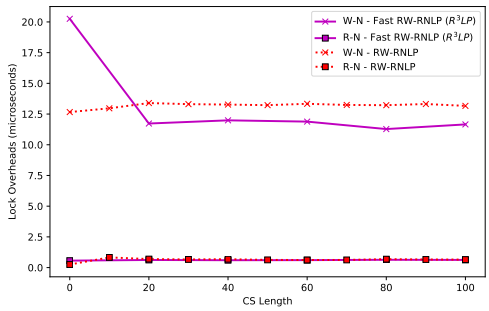



(b) Unlock overhead.

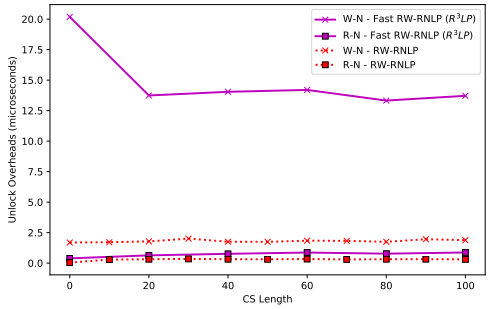



(c) Blocking.

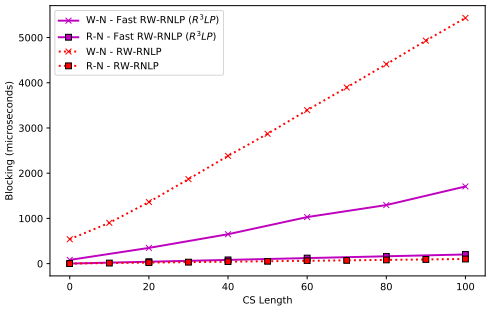
Figure 214: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

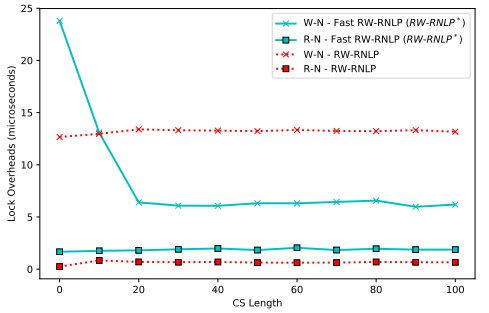



(b) Unlock overhead.

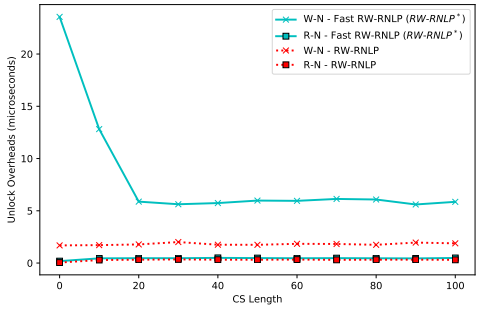



(c) Blocking.

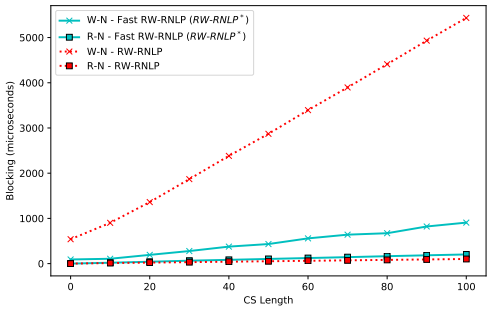
Figure 215: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

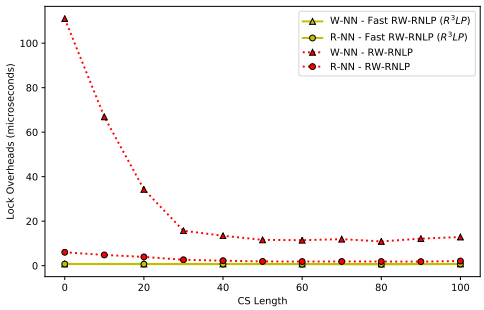



(b) Unlock overhead.

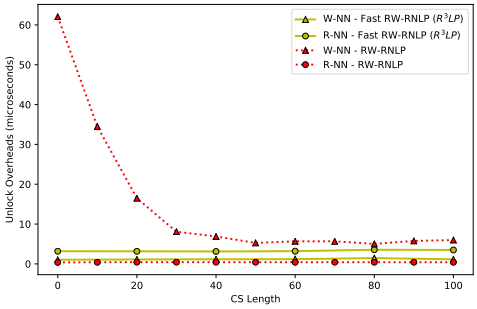



(c) Blocking.

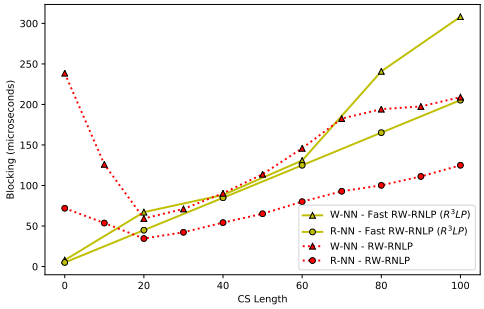
Figure 216: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

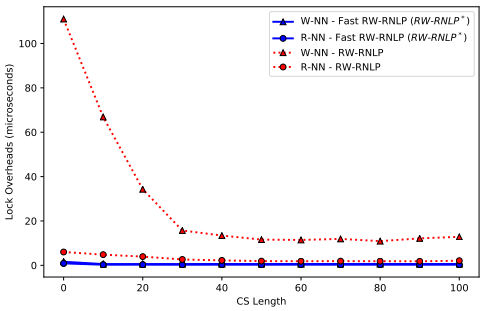



(b) Unlock overhead.

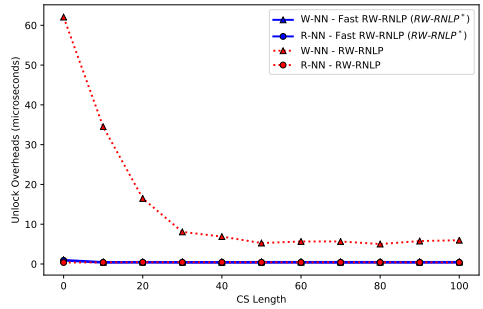



(c) Blocking.

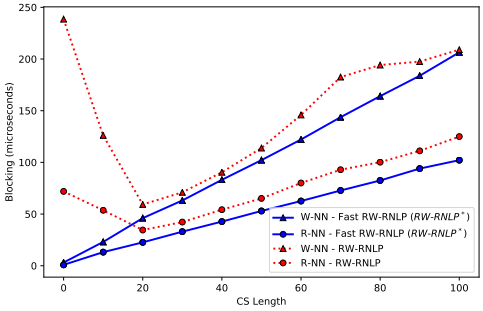
Figure 217: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

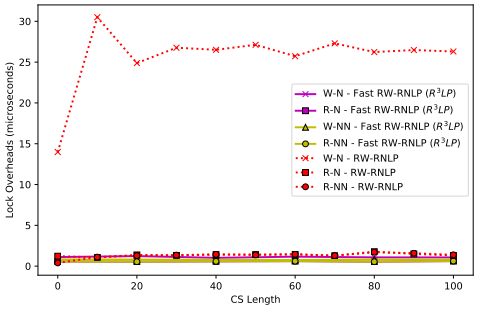



(b) Unlock overhead.

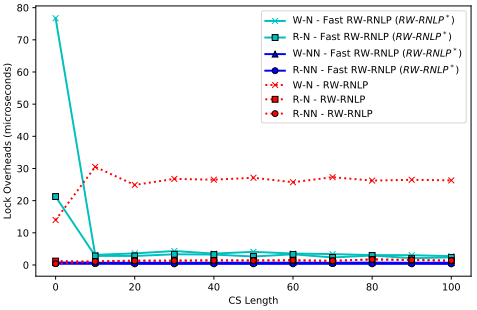



(c) Blocking.

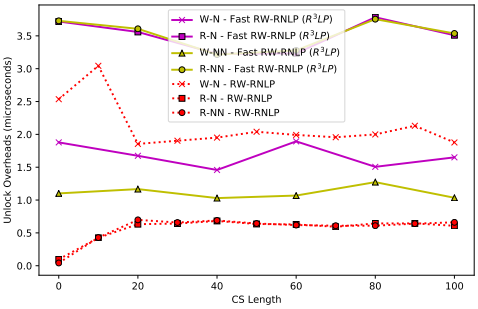
Figure 218: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 0.



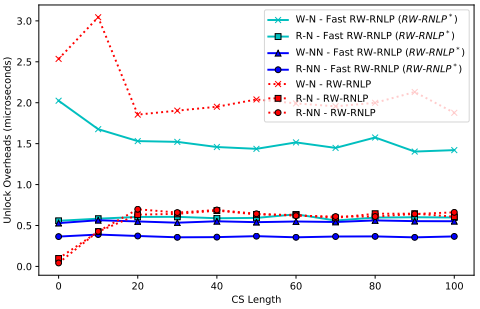

(a) Lock overhead.



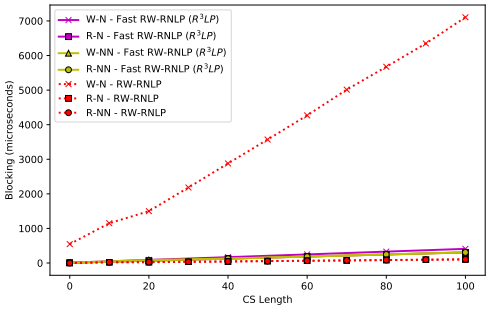

(a) Lock overhead.



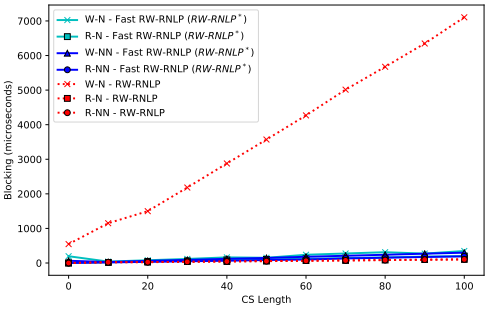

(b) Unlock overhead.




(b) Unlock overhead.



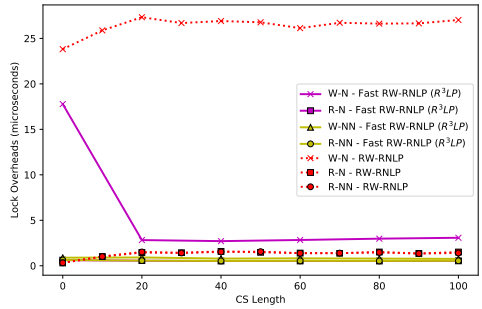

(c) Blocking.



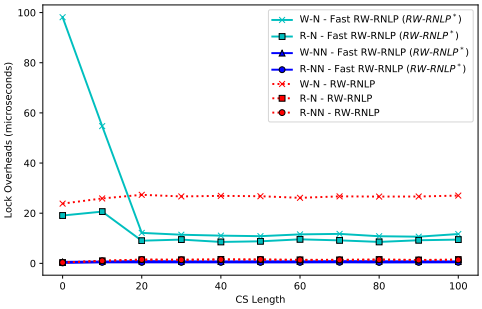

(c) Blocking.

Figure 219: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.

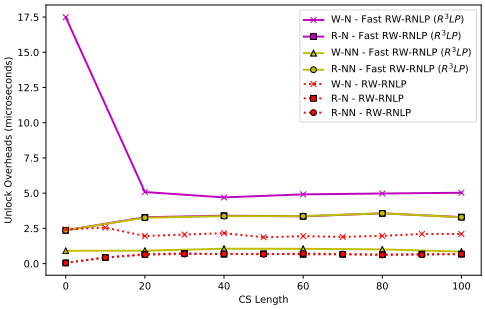
Figure 220: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.



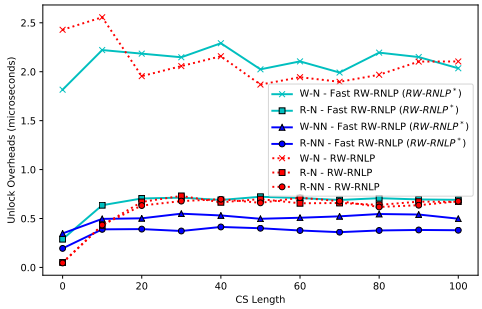

(a) Lock overhead.



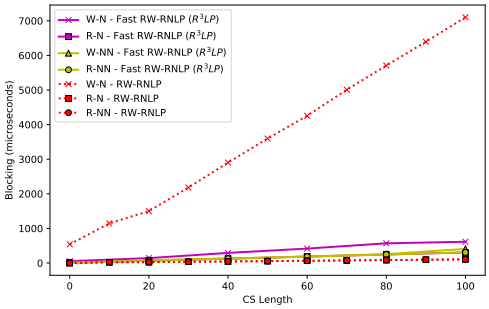

(a) Lock overhead.



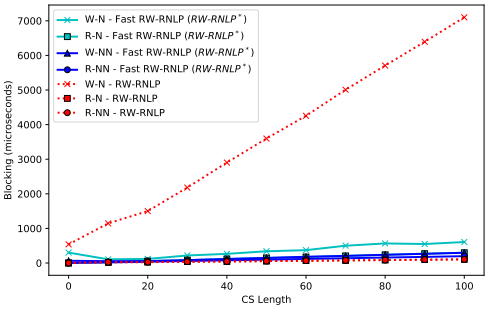

(b) Unlock overhead.




(b) Unlock overhead.



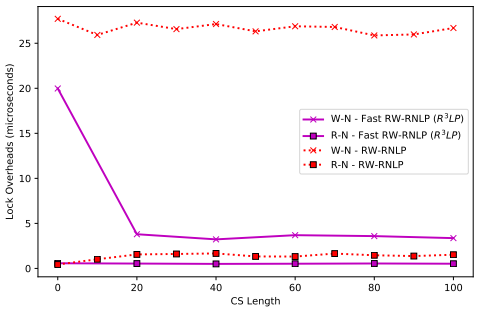

(c) Blocking.



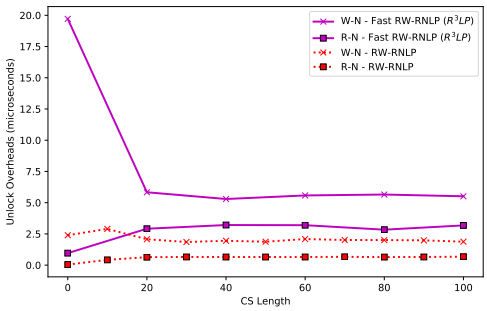

(c) Blocking.

Figure 221: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.

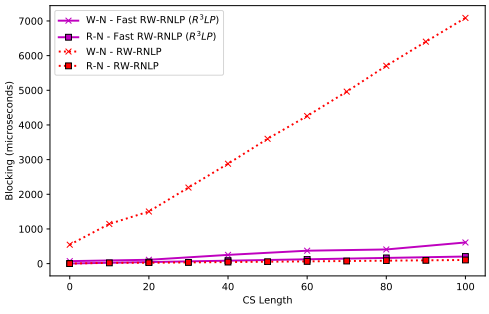
Figure 222: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

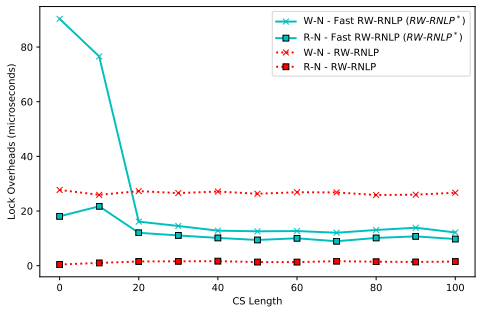



(b) Unlock overhead.

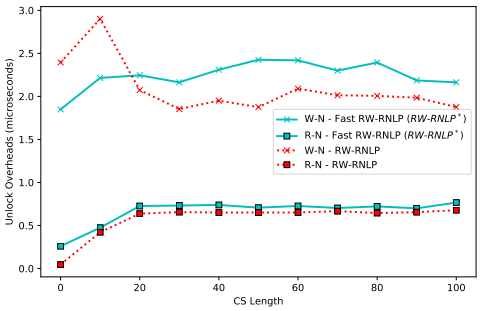



(c) Blocking.

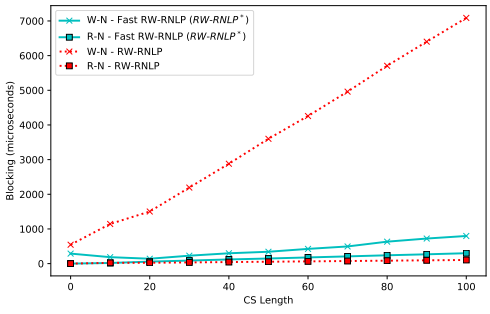
Figure 223: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

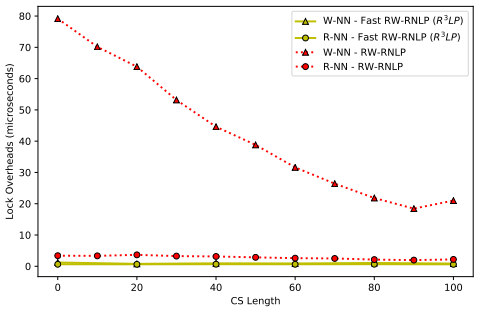



(b) Unlock overhead.

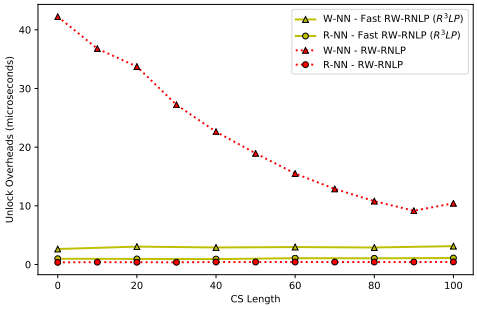



(c) Blocking.

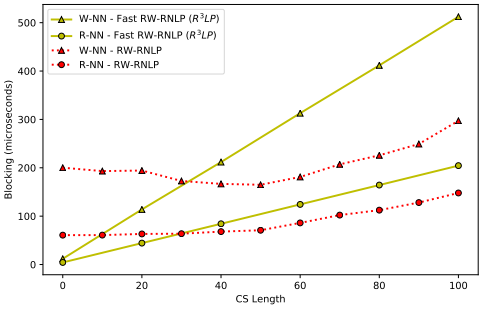
Figure 224: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

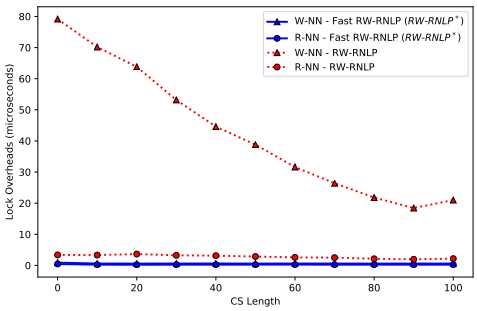



(b) Unlock overhead.

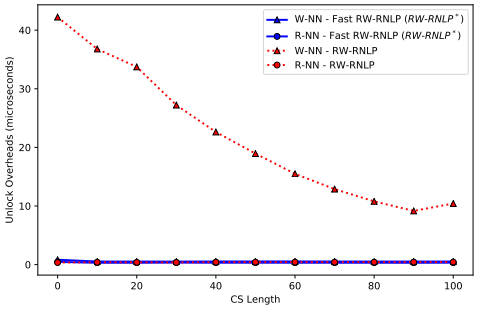



(c) Blocking.

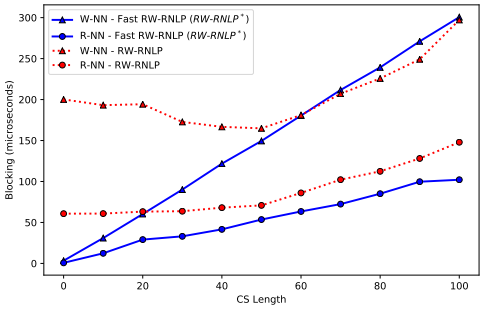
Figure 225: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

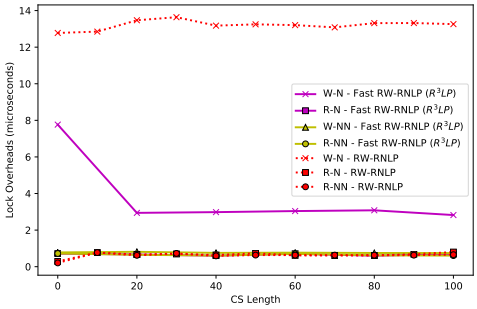



(b) Unlock overhead.

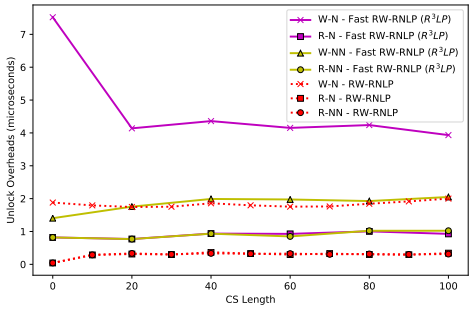



(c) Blocking.

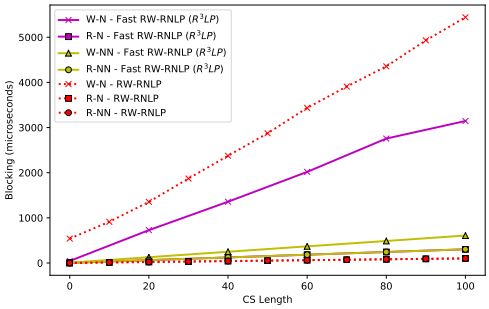
Figure 226: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

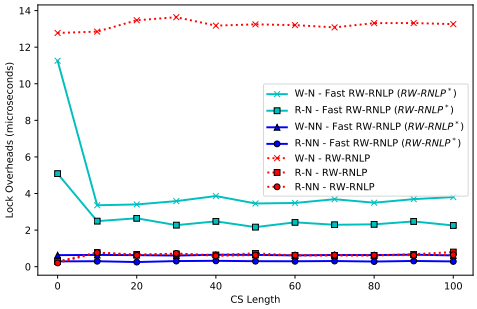



(b) Unlock overhead.

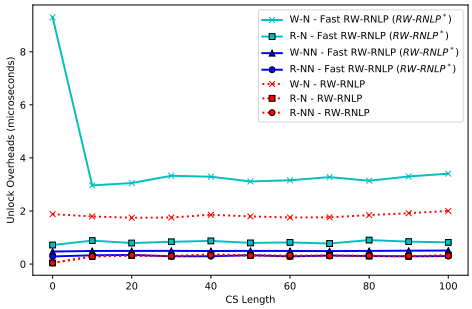



(c) Blocking.

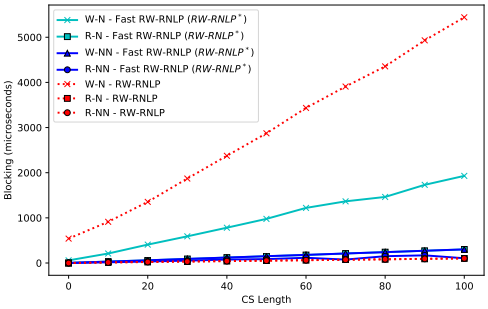
Figure 227: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

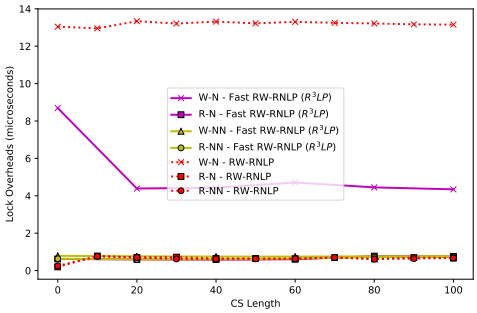



(b) Unlock overhead.

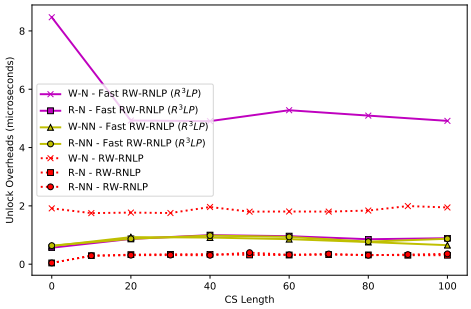



(c) Blocking.

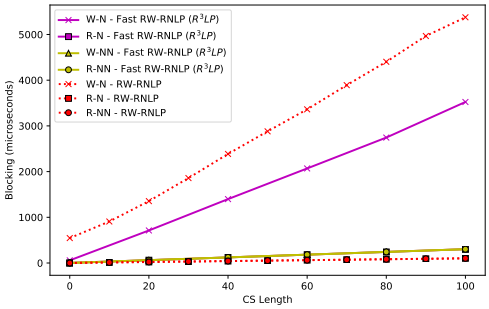
Figure 228: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

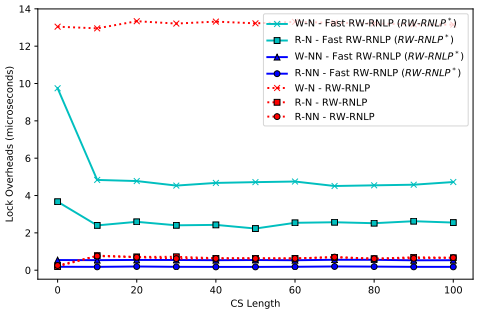



(b) Unlock overhead.

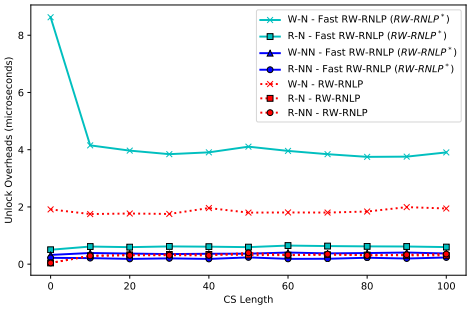



(c) Blocking.

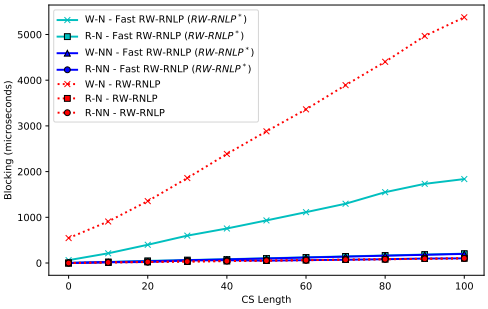
Figure 229: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

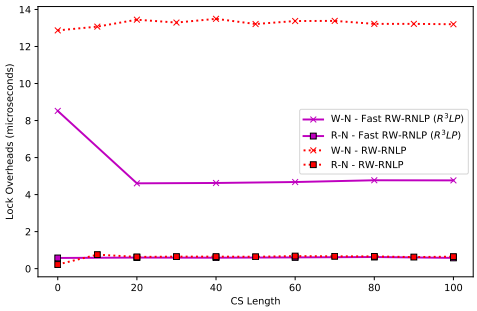



(b) Unlock overhead.

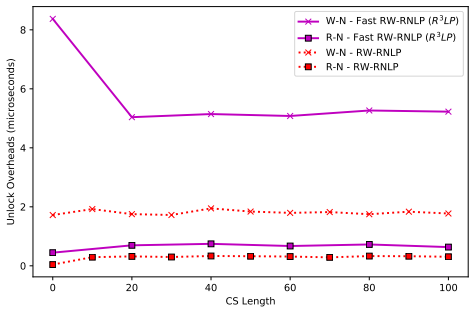



(c) Blocking.

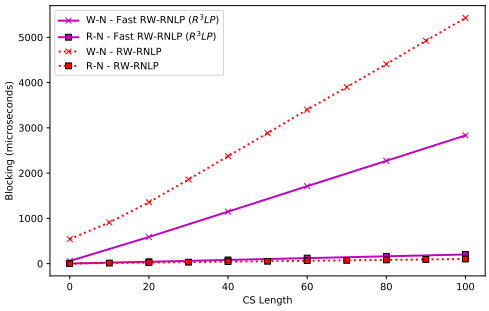
Figure 230: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

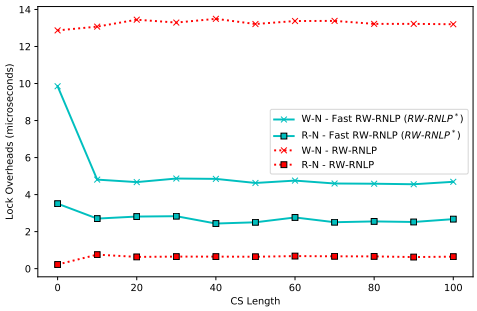



(b) Unlock overhead.

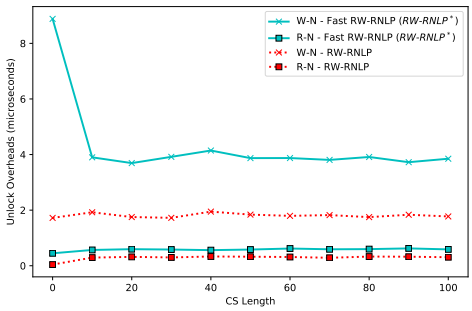



(c) Blocking.

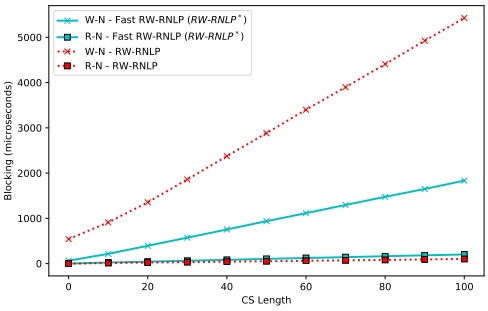
Figure 231: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

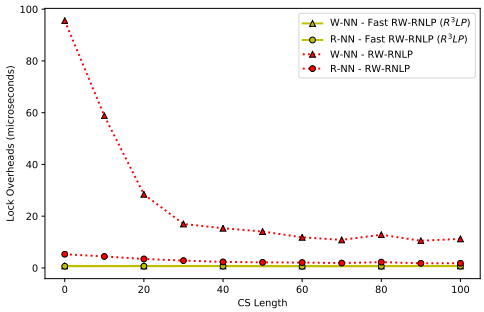



(b) Unlock overhead.

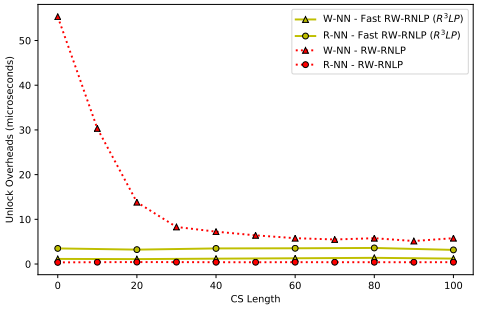



(c) Blocking.

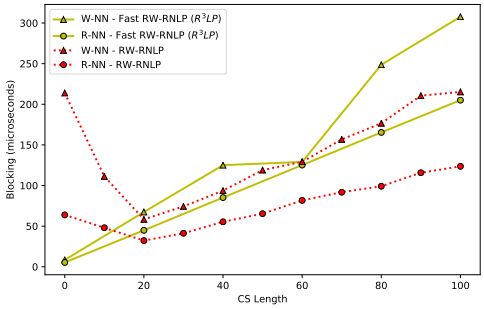
Figure 232: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

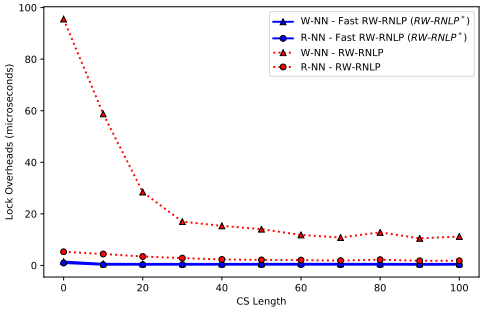



(b) Unlock overhead.

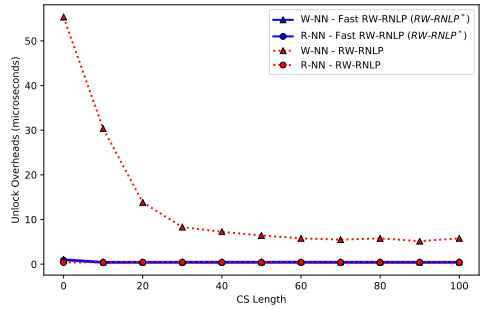



(c) Blocking.

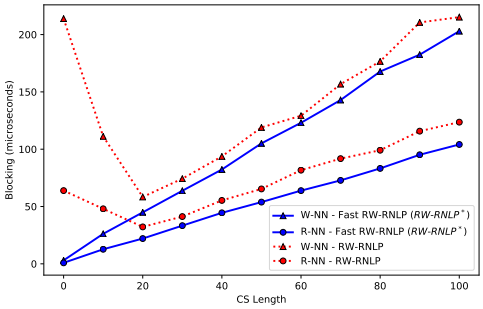
Figure 233: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

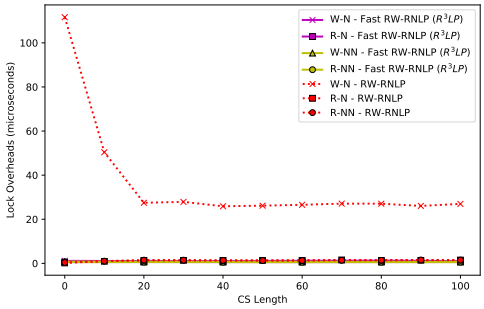



(b) Unlock overhead.

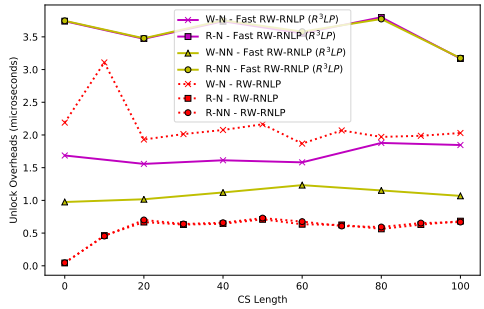



(c) Blocking.

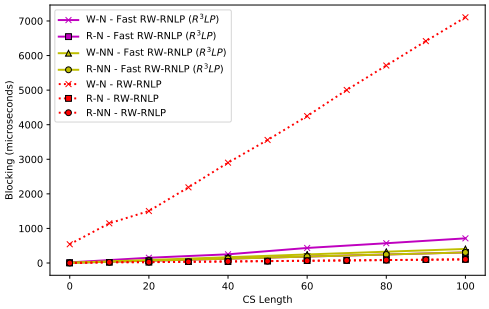
Figure 234: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

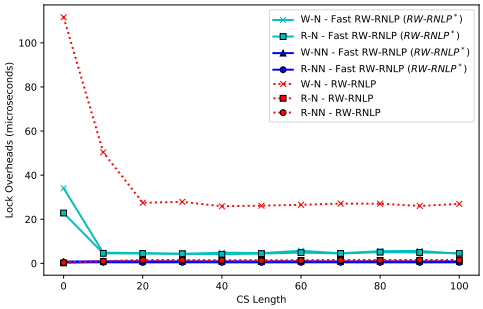



(b) Unlock overhead.

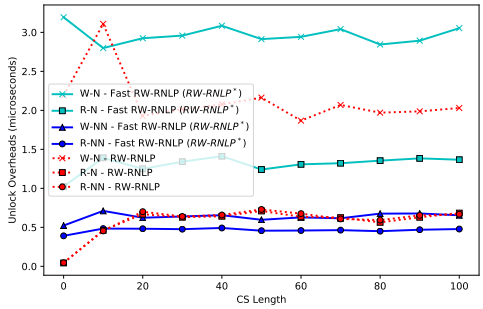



(c) Blocking.

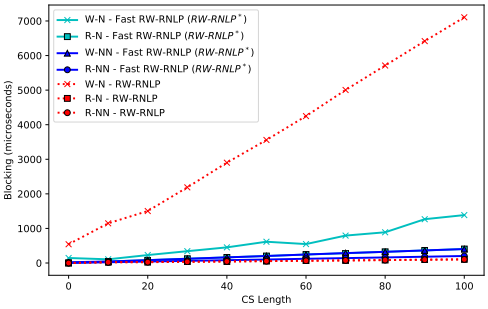
Figure 235: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

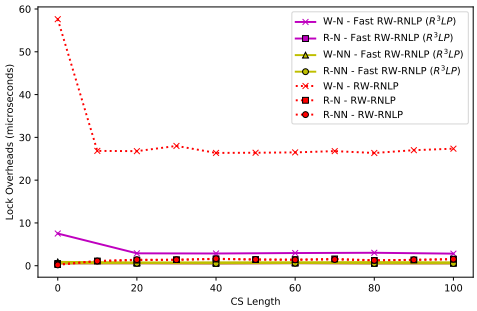



(b) Unlock overhead.

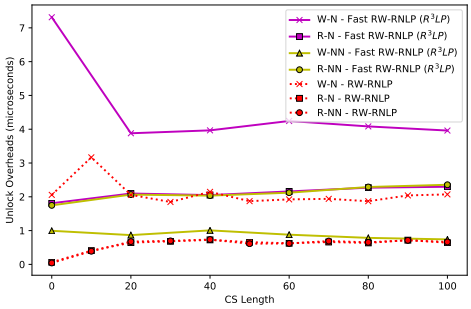



(c) Blocking.

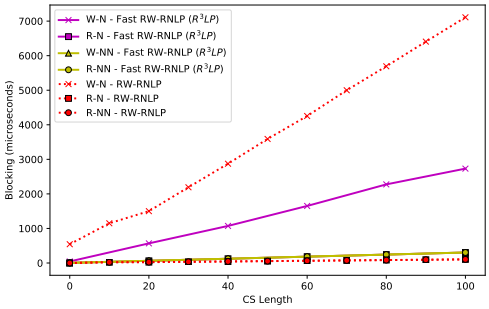
Figure 236: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

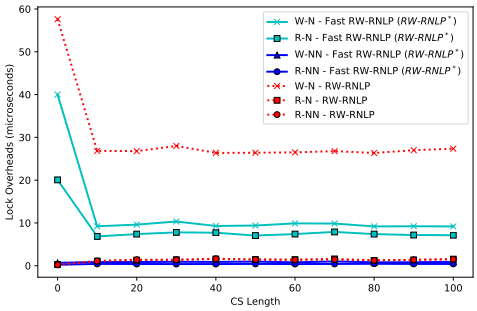



(b) Unlock overhead.

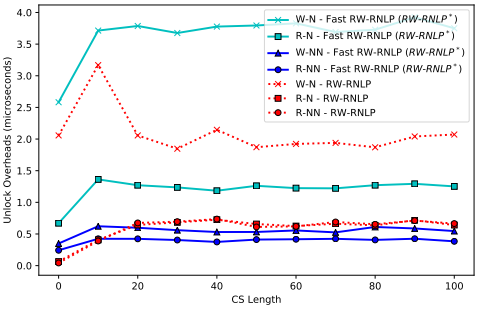



(c) Blocking.

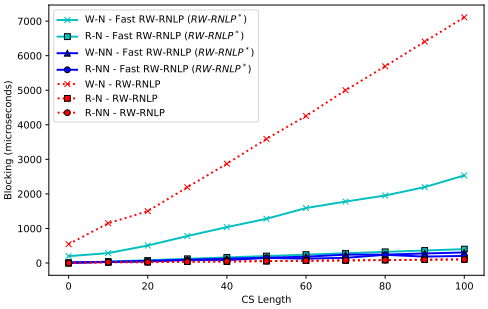
Figure 237: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

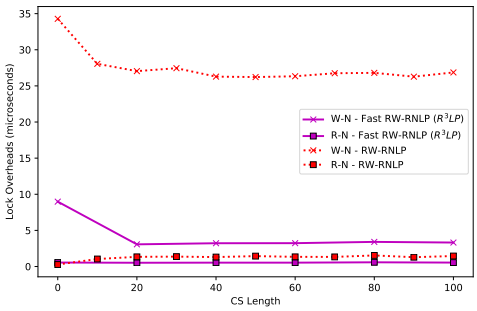



(b) Unlock overhead.

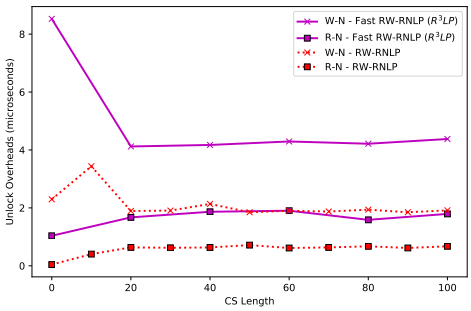



(c) Blocking.

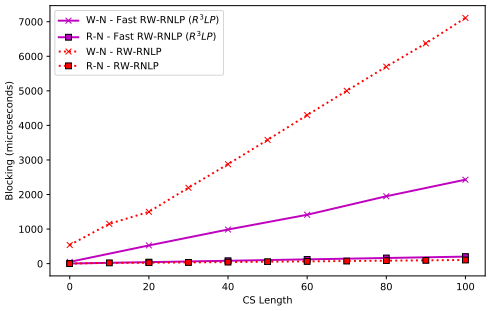
Figure 238: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

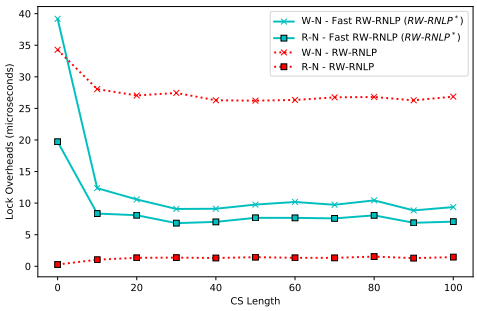



(b) Unlock overhead.

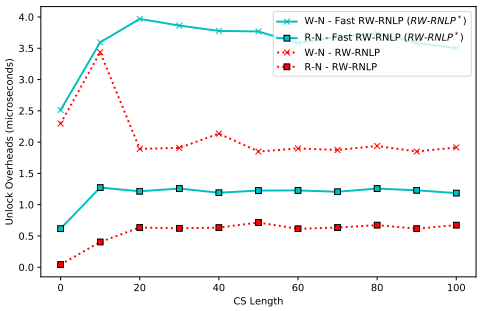



(c) Blocking.

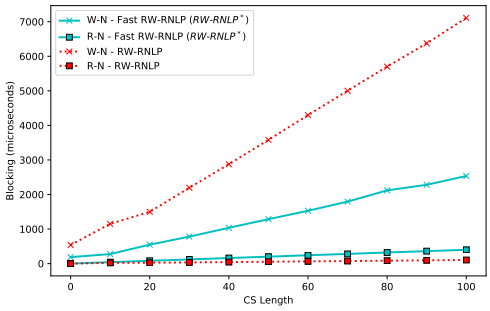
Figure 239: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

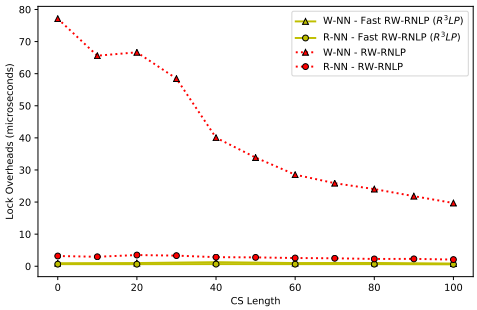



(b) Unlock overhead.

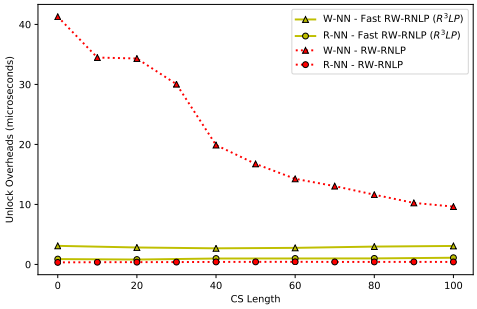



(c) Blocking.

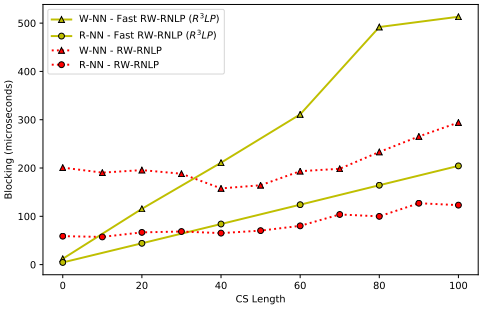
Figure 240: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

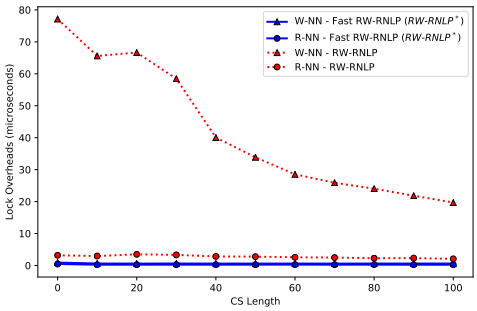



(b) Unlock overhead.

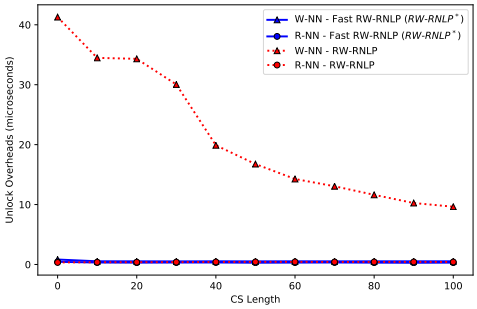



(c) Blocking.

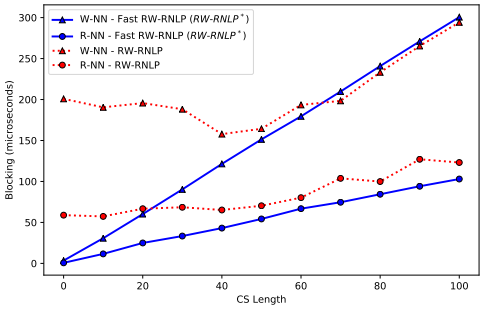
Figure 241: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

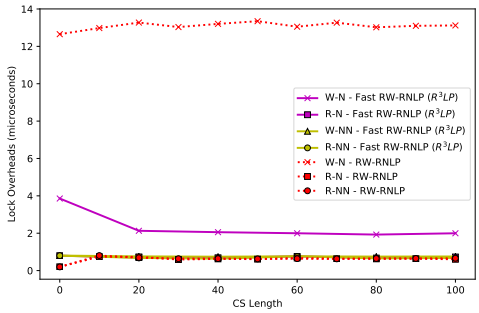



(b) Unlock overhead.

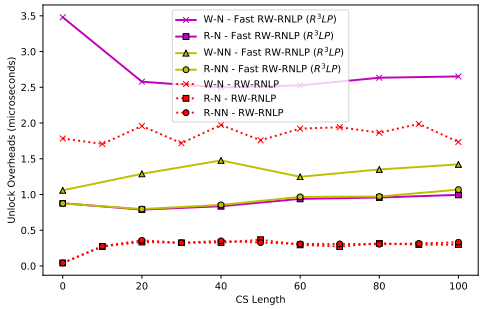



(c) Blocking.

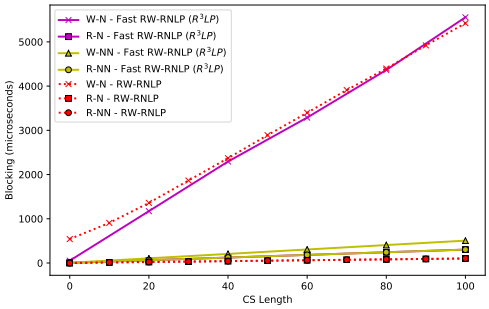
Figure 242: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 0.




(a) Lock overhead.

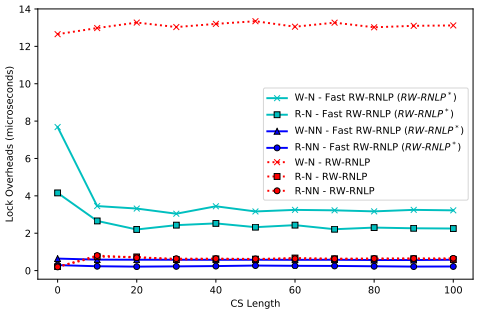



(b) Unlock overhead.

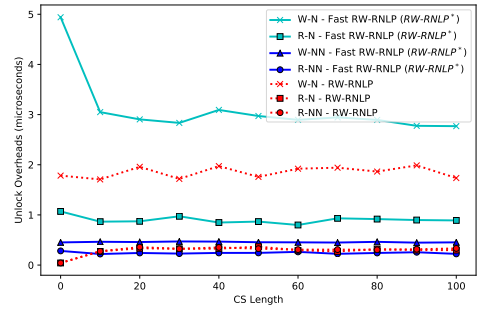



(c) Blocking.

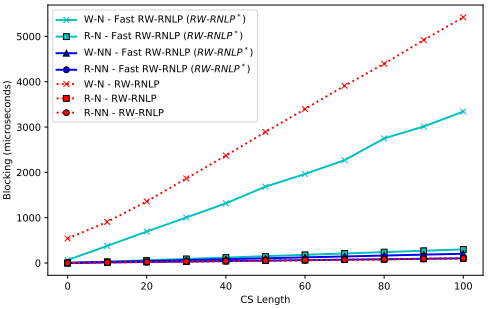
Figure 243: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

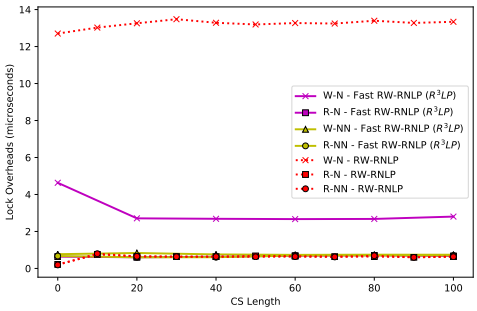



(b) Unlock overhead.

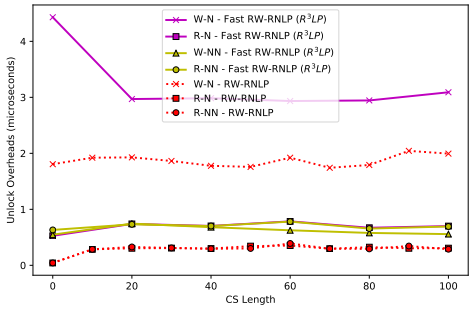



(c) Blocking.

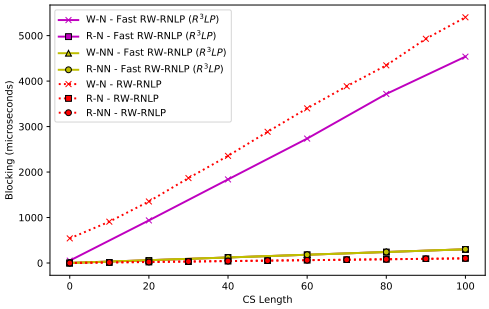
Figure 244: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

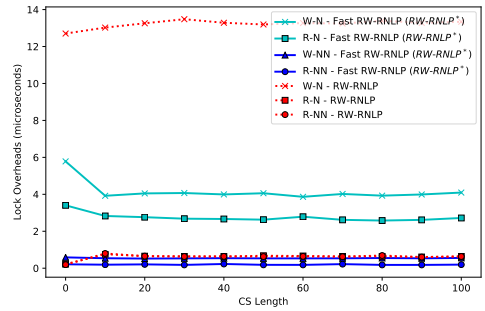



(b) Unlock overhead.

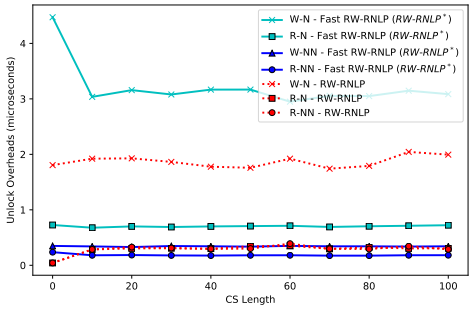



(c) Blocking.

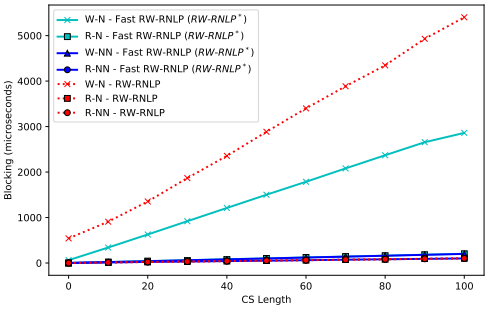
Figure 245: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

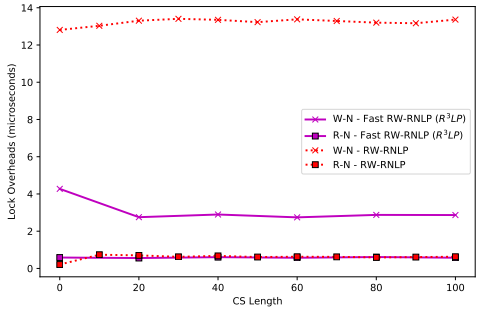



(b) Unlock overhead.

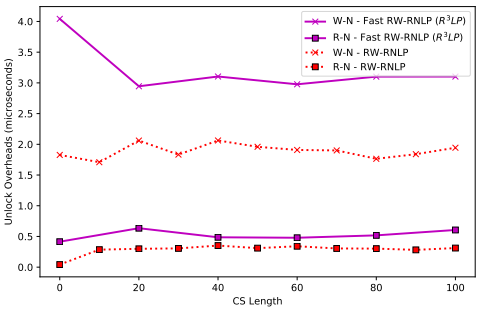



(c) Blocking.

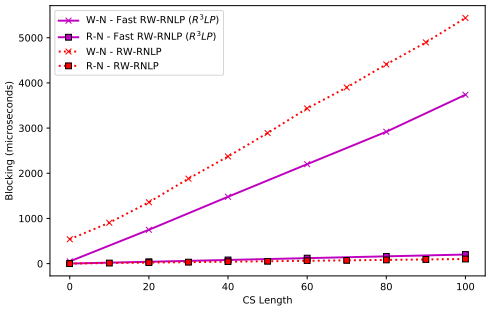
Figure 246: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

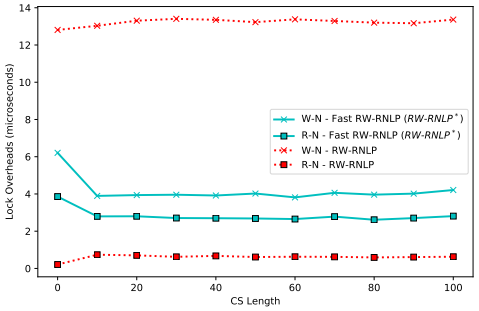



(b) Unlock overhead.

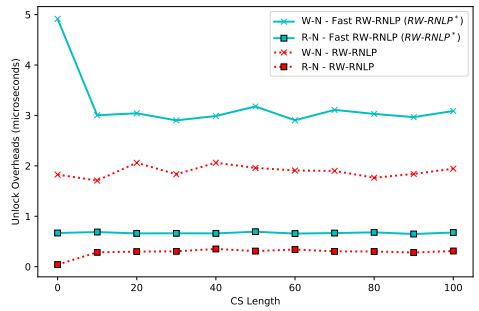



(c) Blocking.

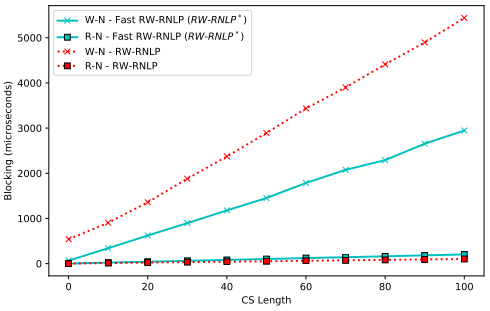
Figure 247: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

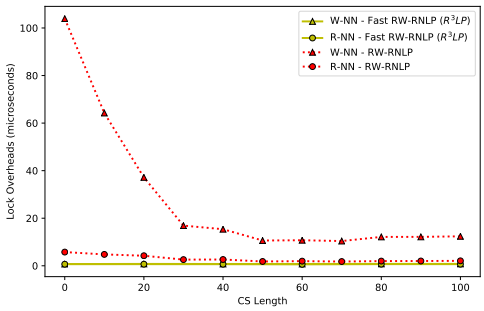



(b) Unlock overhead.

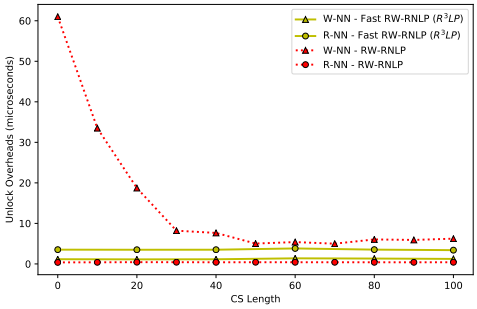



(c) Blocking.

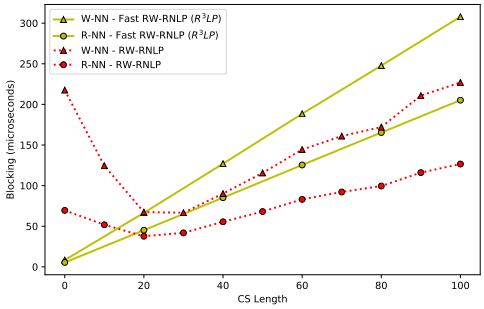
Figure 248: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

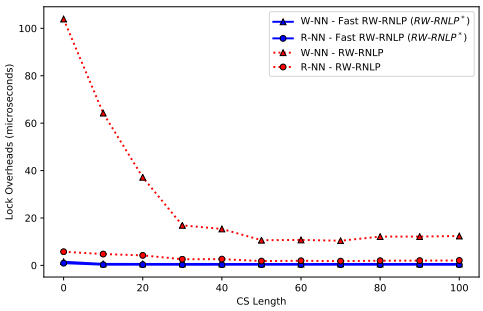



(b) Unlock overhead.

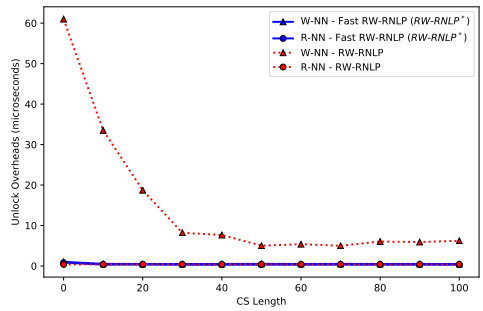



(c) Blocking.

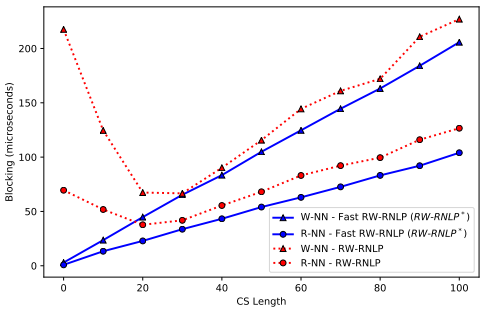
Figure 249: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 0.




(a) Lock overhead.

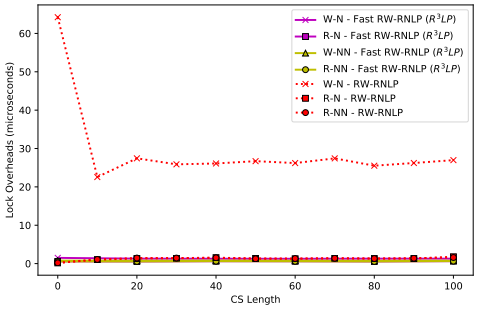



(b) Unlock overhead.

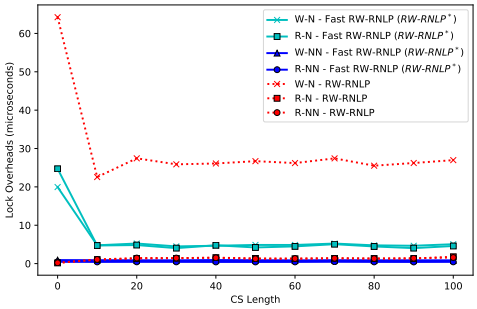



(c) Blocking.

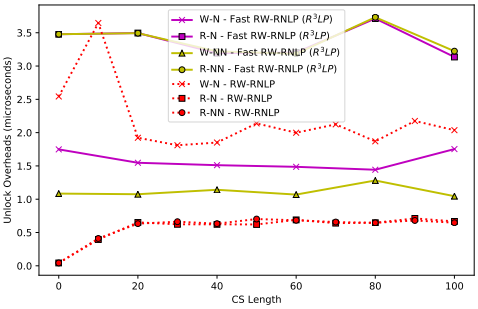
Figure 250: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 0.



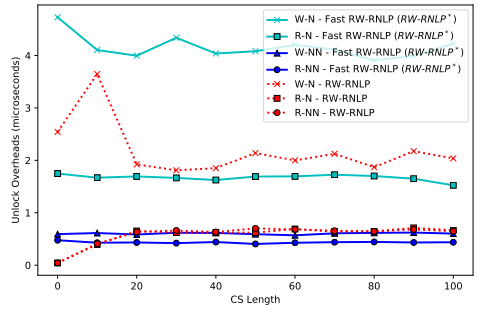

(a) Lock overhead.



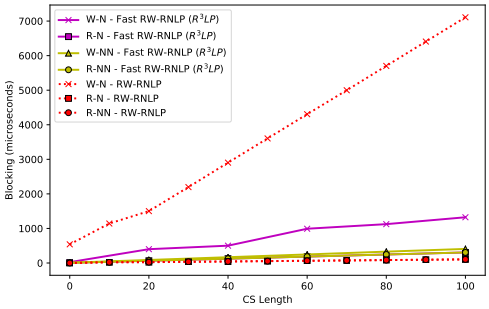

(a) Lock overhead.



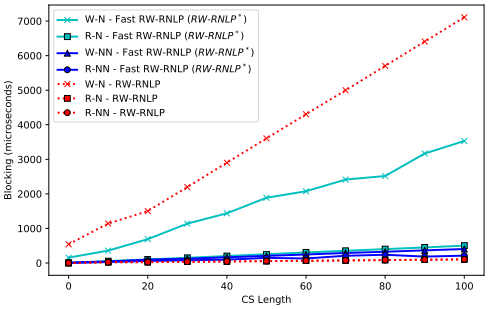

(b) Unlock overhead.




(b) Unlock overhead.



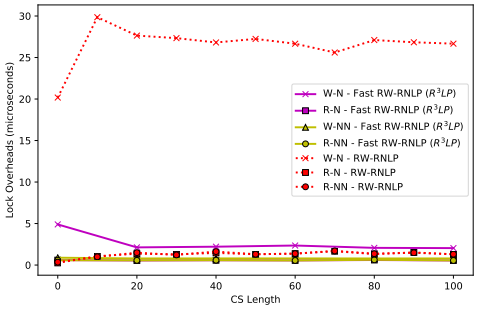

(c) Blocking.



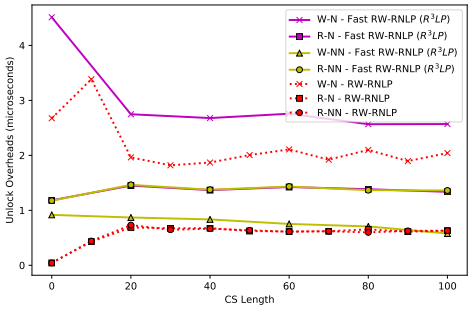

(c) Blocking.

Figure 251: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.

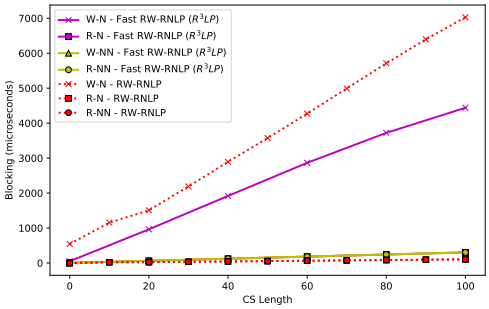
Figure 252: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

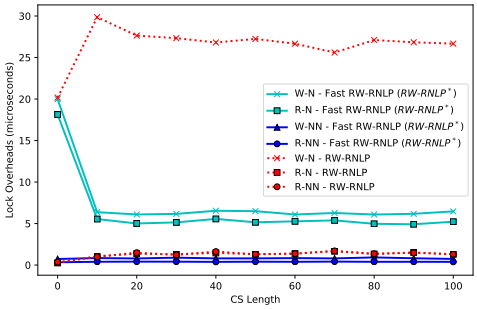



(b) Unlock overhead.

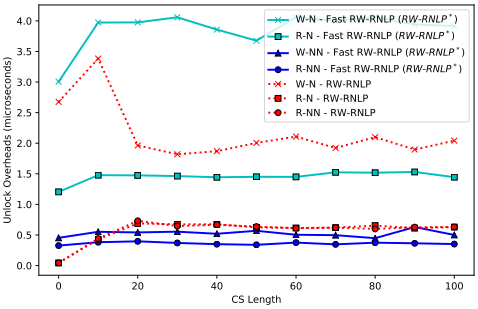



(c) Blocking.

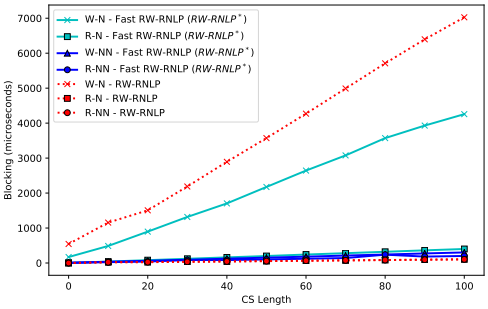
Figure 253: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

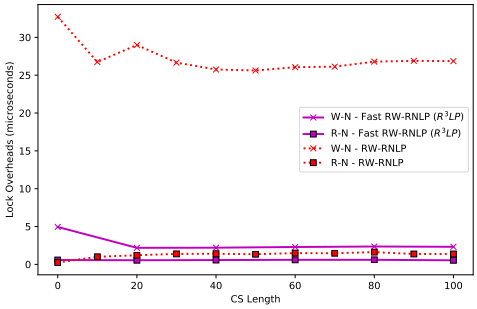



(b) Unlock overhead.

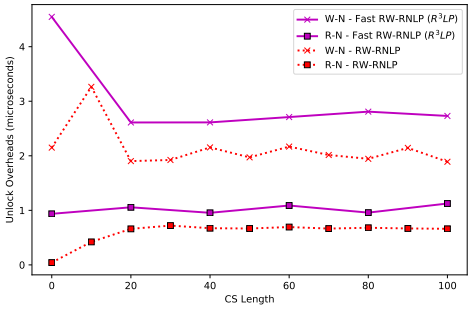



(c) Blocking.

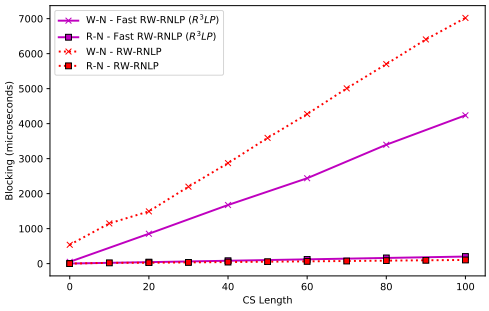
Figure 254: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

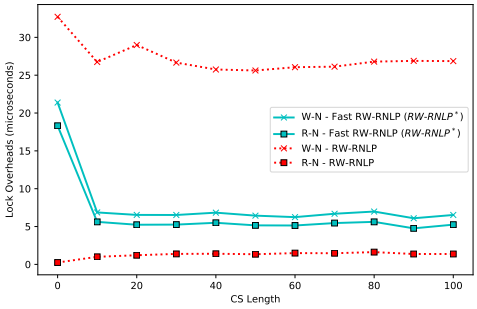



(b) Unlock overhead.

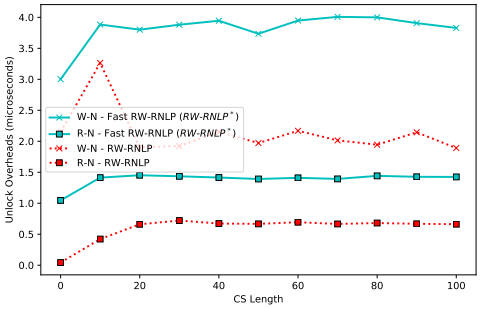



(c) Blocking.

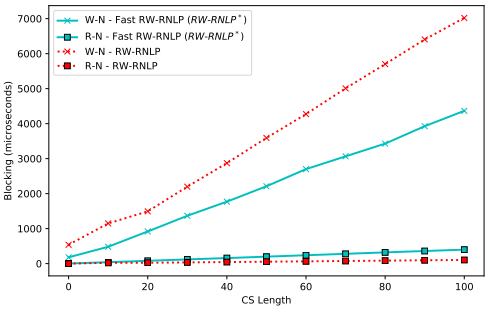
Figure 255: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

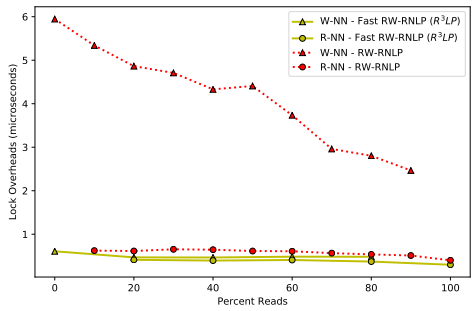



(b) Unlock overhead.

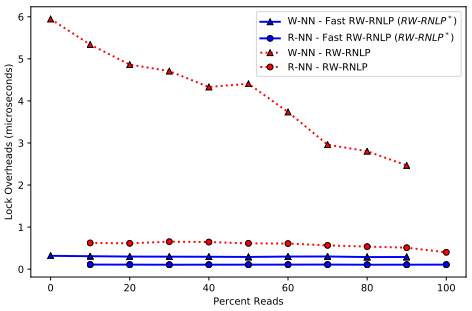



(c) Blocking.

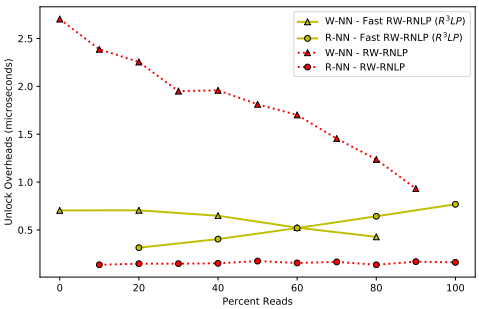
Figure 256: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposedto a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.



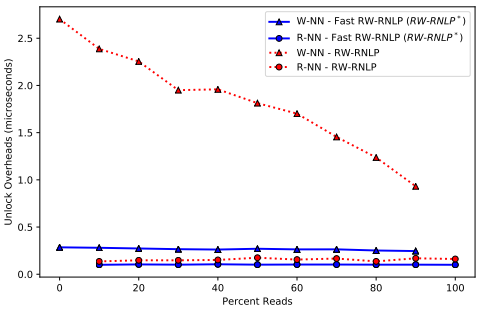

(a) Lock overhead.



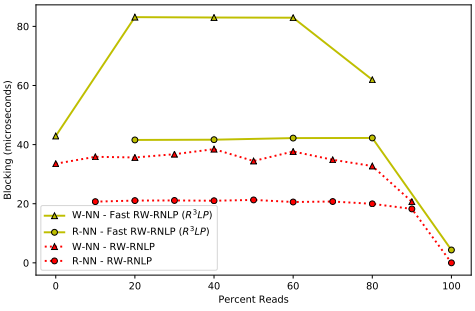

(a) Lock overhead.



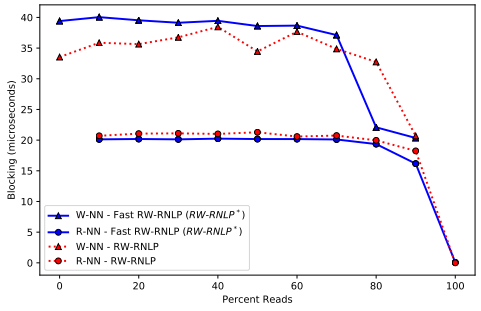

(b) Unlock overhead.




(b) Unlock overhead.



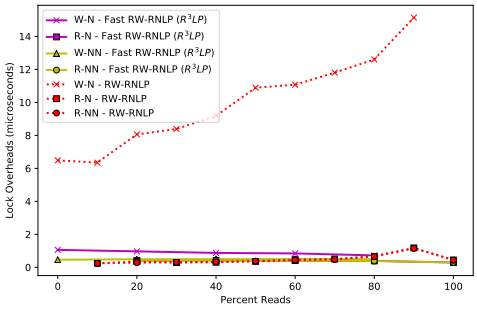

(c) Blocking.



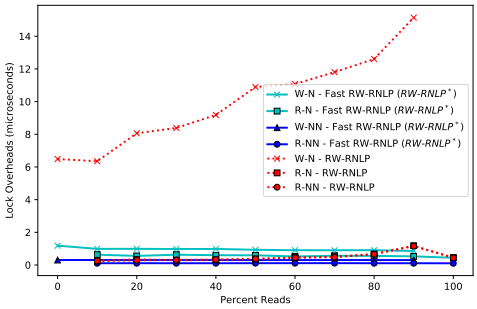

(c) Blocking.

Figure 257: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.

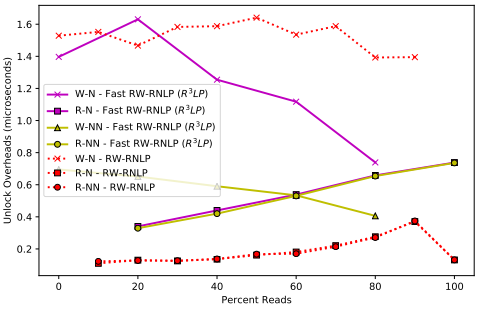
Figure 258: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.



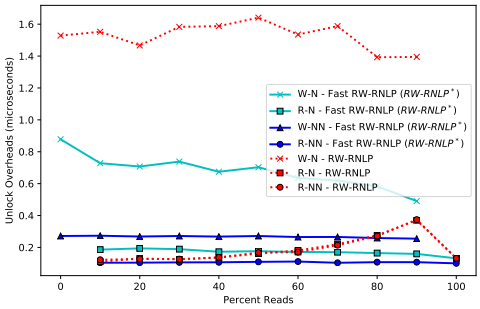

(a) Lock overhead.



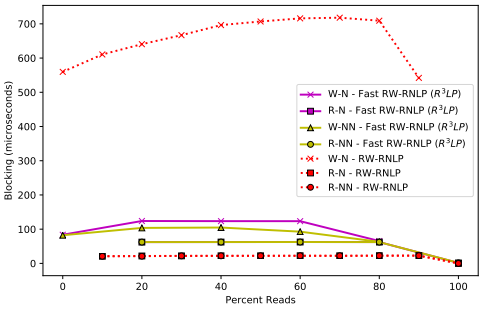

(a) Lock overhead.



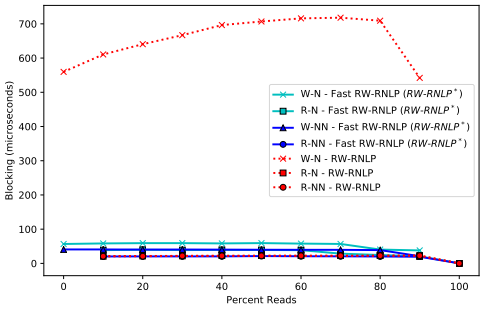

(b) Unlock overhead.




(b) Unlock overhead.



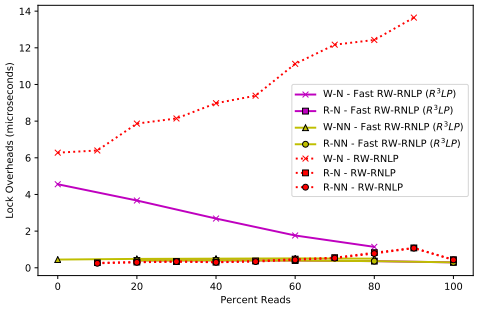

(c) Blocking.



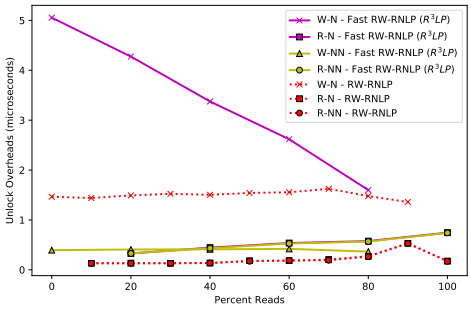

(c) Blocking.

Figure 259: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.

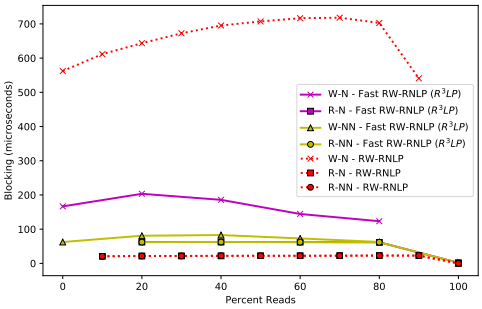
Figure 260: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

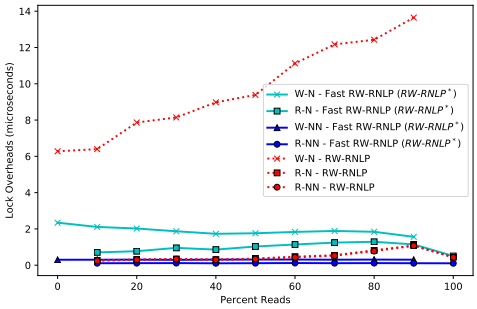



(b) Unlock overhead.

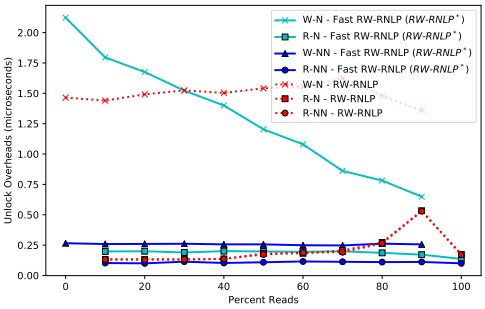



(c) Blocking.

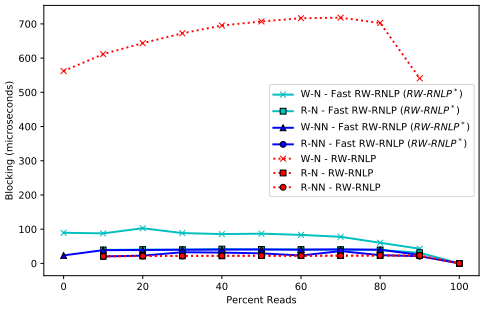
Figure 261: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

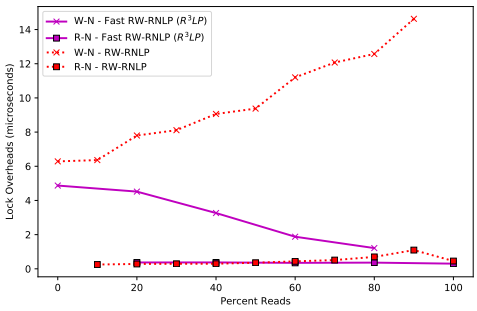



(b) Unlock overhead.

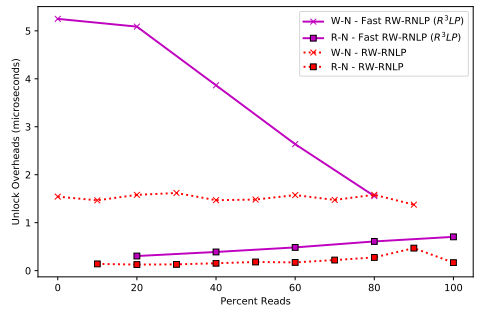



(c) Blocking.

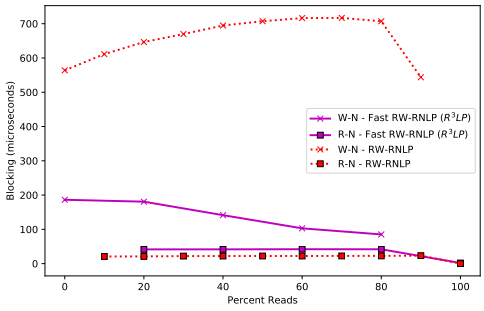
Figure 262: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

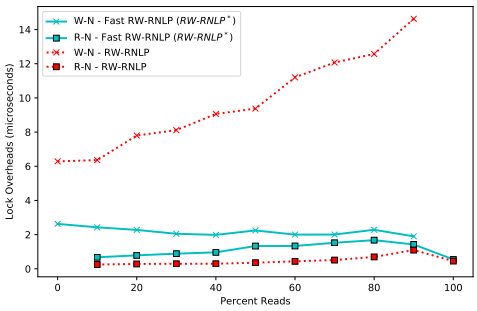



(b) Unlock overhead.

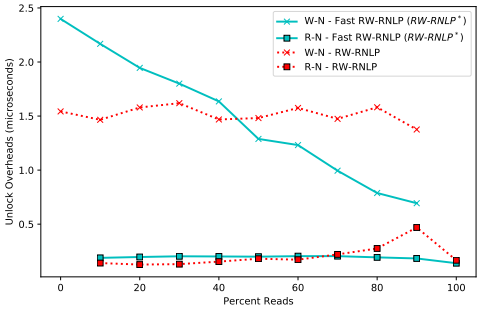



(c) Blocking.

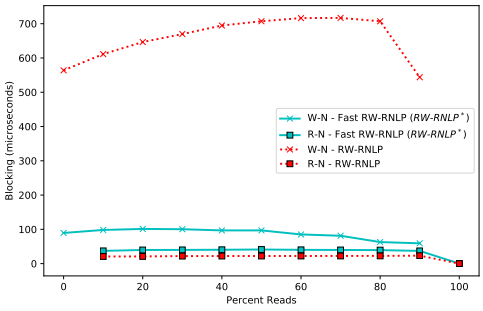
Figure 263: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

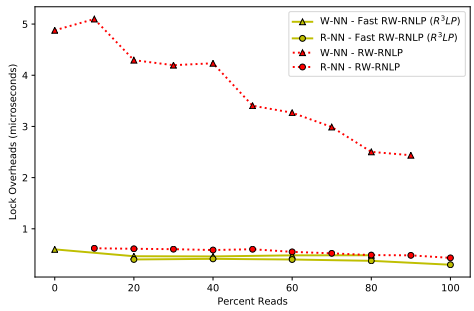



(b) Unlock overhead.

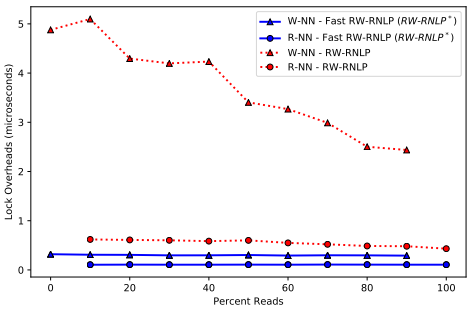



(c) Blocking.

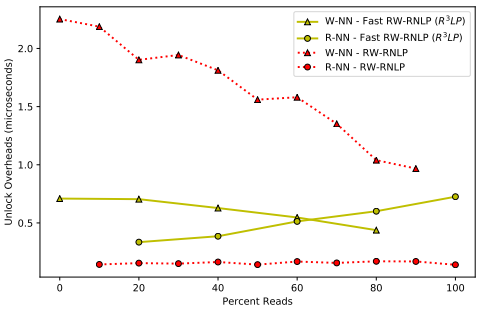
Figure 264: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.



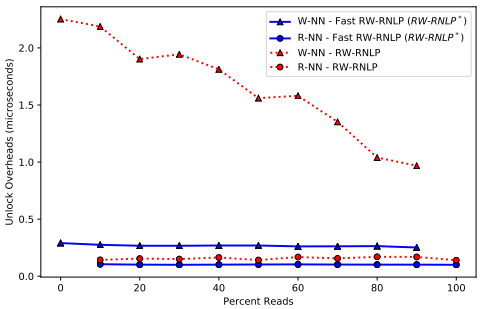

(a) Lock overhead.



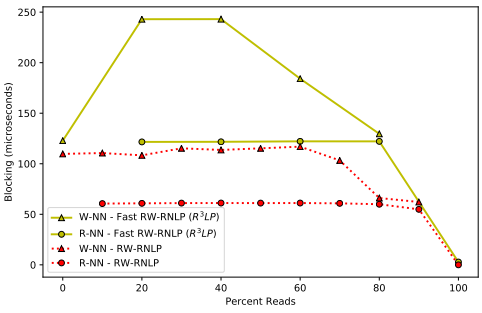

(a) Lock overhead.



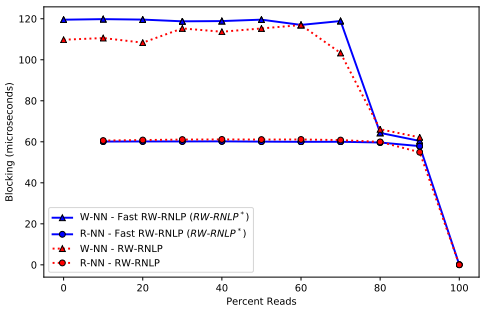

(b) Unlock overhead.




(b) Unlock overhead.



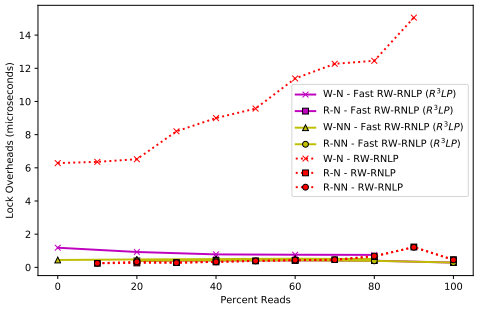

(c) Blocking.



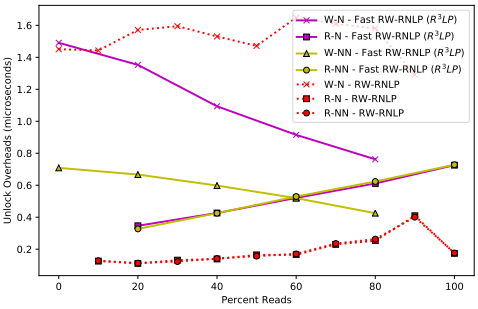

(c) Blocking.

Figure 265: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.

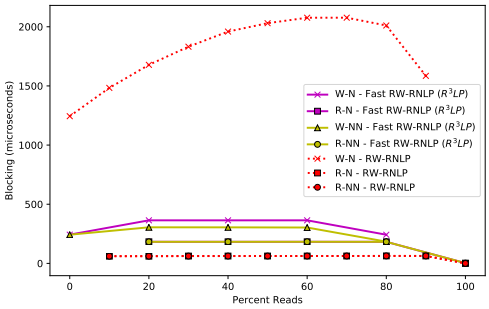
Figure 266: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

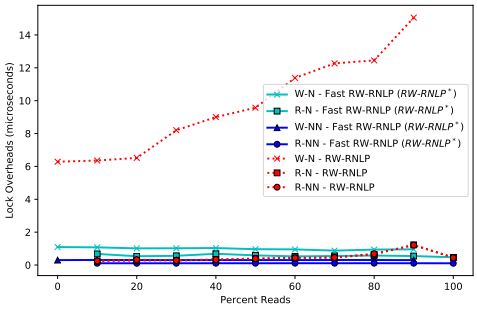



(b) Unlock overhead.

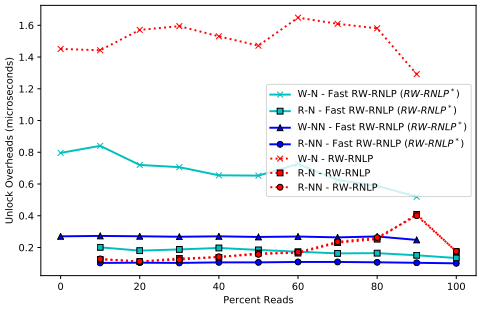



(c) Blocking.

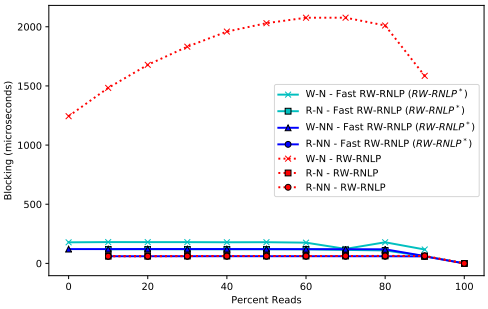
Figure 267: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

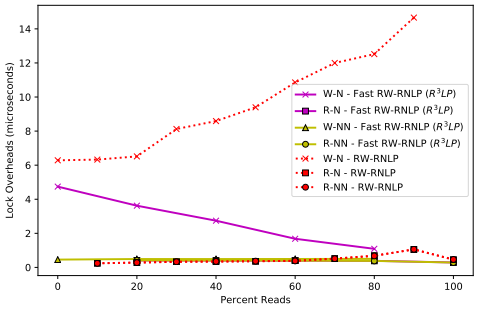



(b) Unlock overhead.

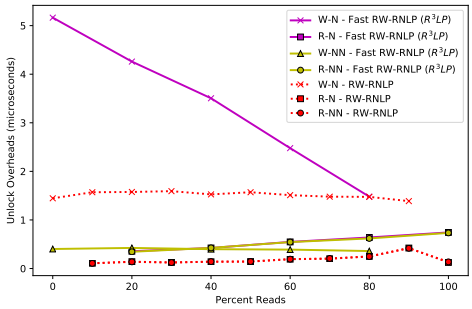



(c) Blocking.

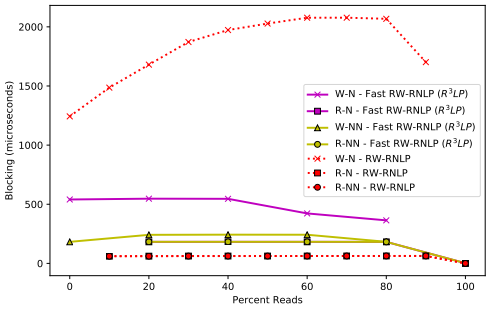
Figure 268: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

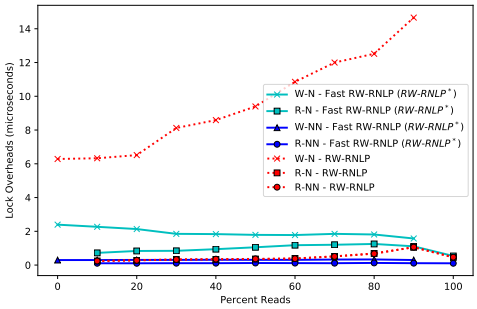



(b) Unlock overhead.

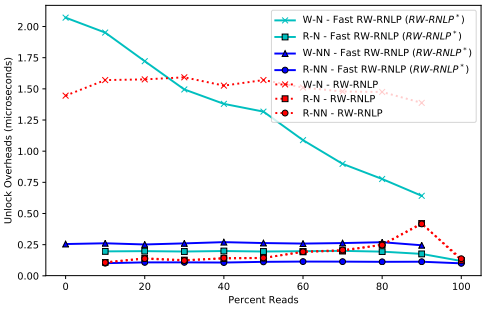



(c) Blocking.

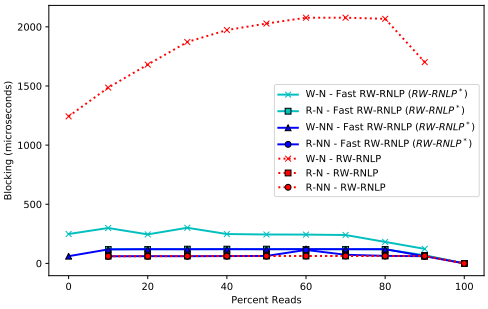
Figure 269: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

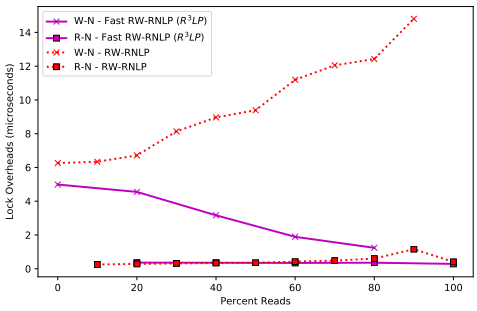



(b) Unlock overhead.

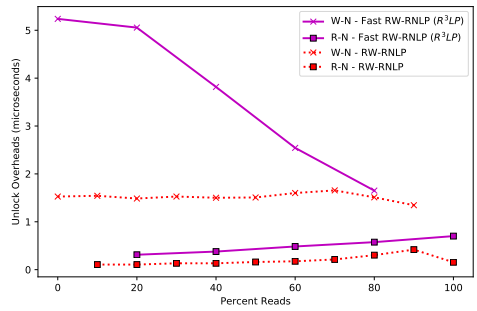



(c) Blocking.

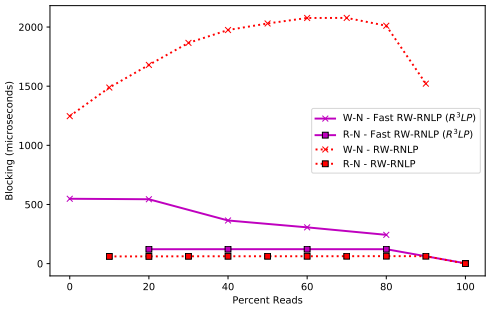
Figure 270: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

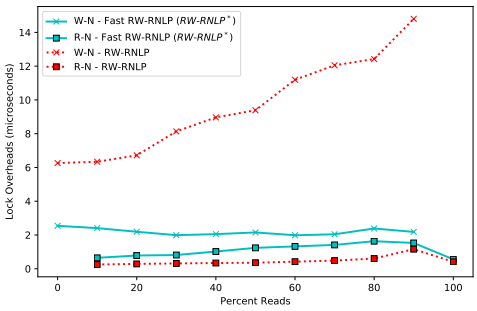



(b) Unlock overhead.

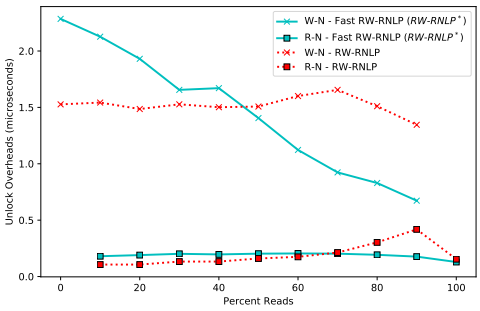



(c) Blocking.

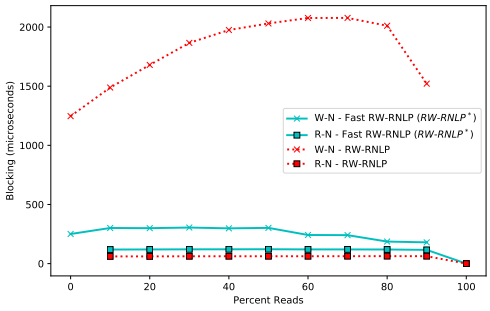
Figure 271: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

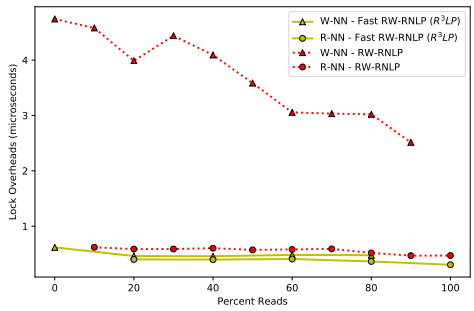



(b) Unlock overhead.

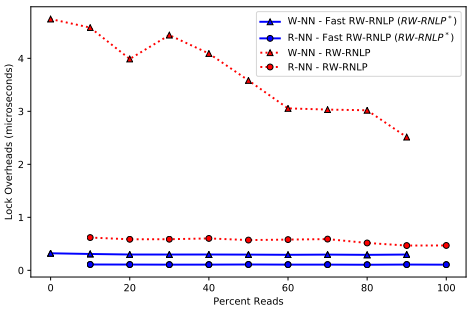



(c) Blocking.

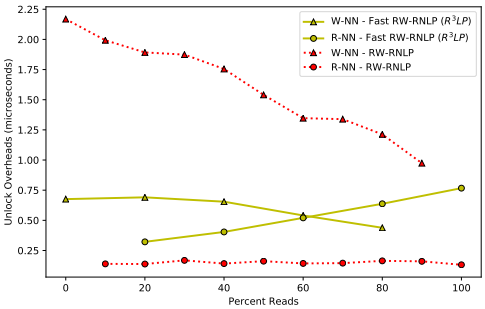
Figure 272: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.



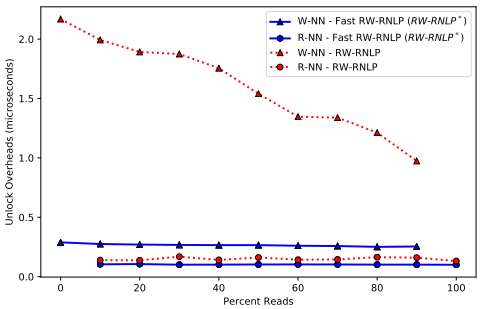

(a) Lock overhead.



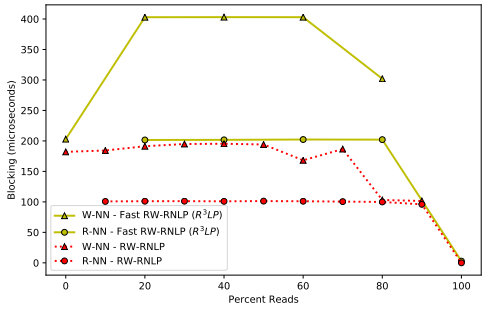

(a) Lock overhead.



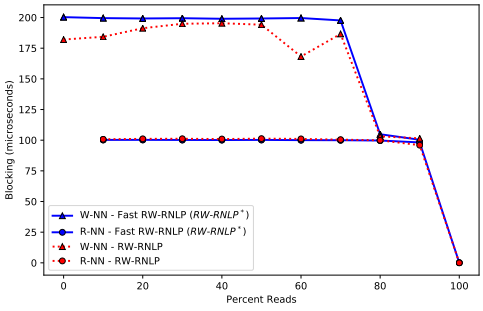

(b) Unlock overhead.




(b) Unlock overhead.



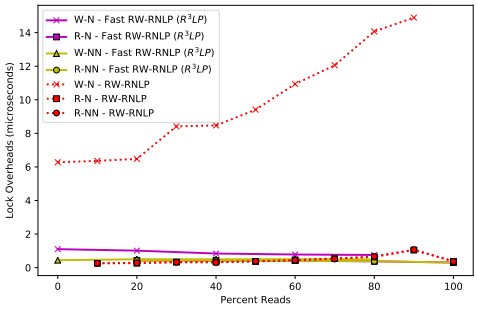

(c) Blocking.



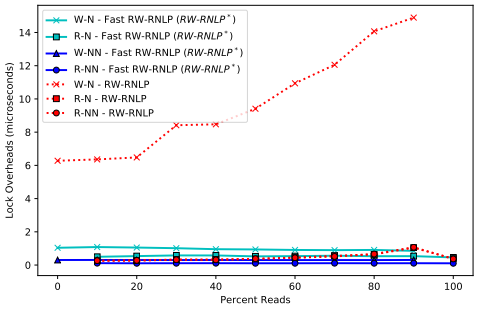

(c) Blocking.

Figure 273: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.

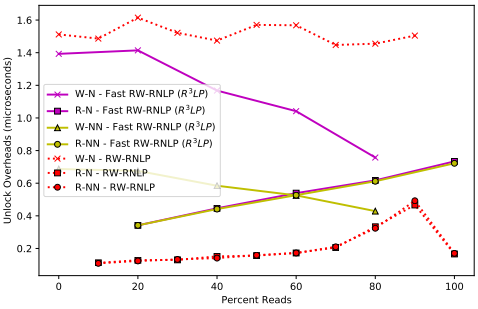
Figure 274: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.



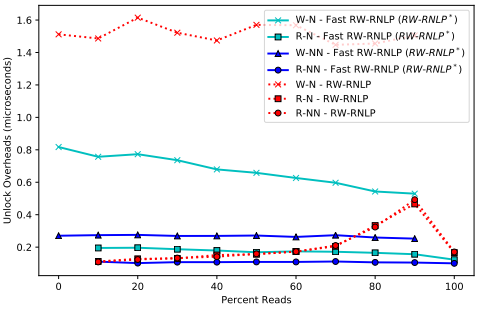

(a) Lock overhead.



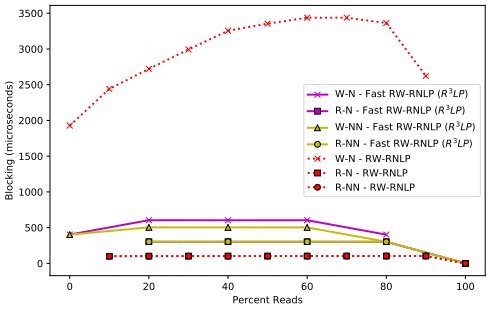

(a) Lock overhead.



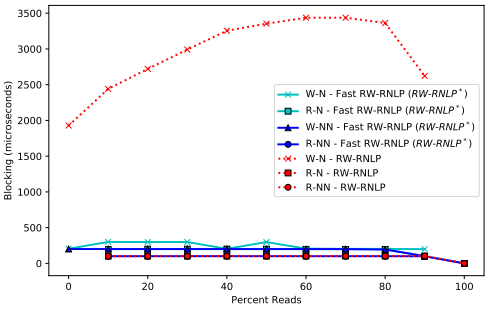

(b) Unlock overhead.




(b) Unlock overhead.



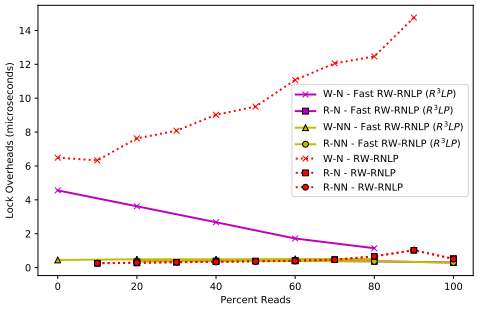

(c) Blocking.



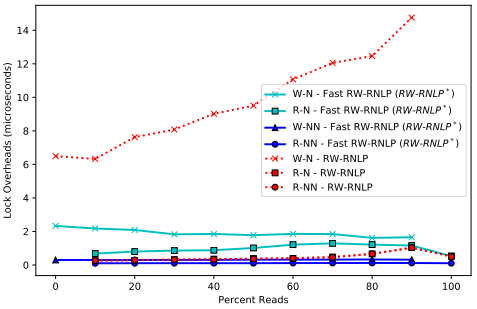

(c) Blocking.

Figure 275: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.

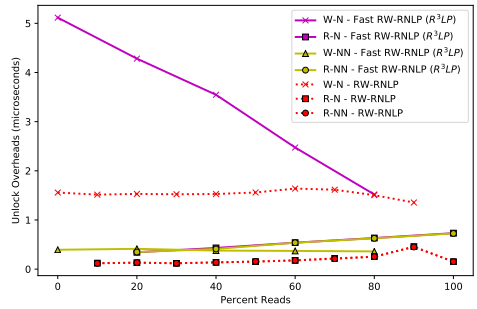
Figure 276: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.



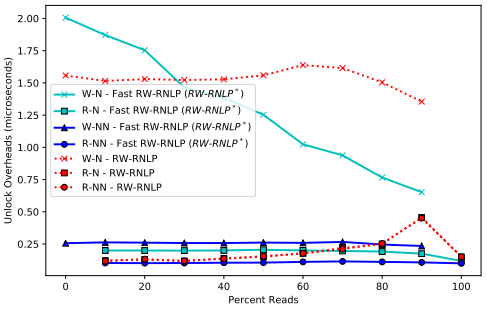

(a) Lock overhead.



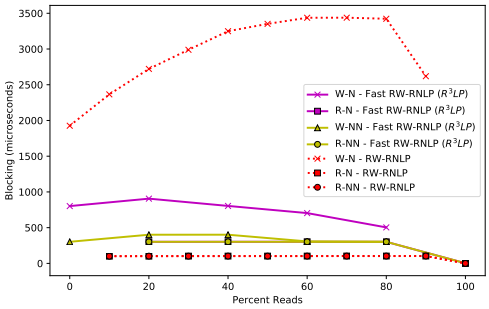

(a) Lock overhead.



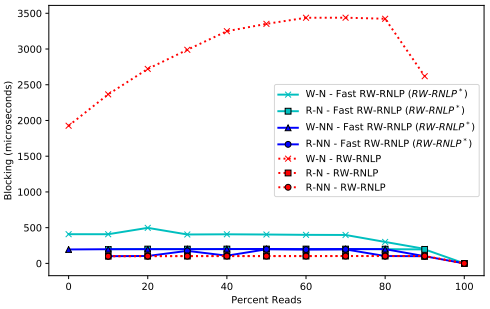

(b) Unlock overhead.




(b) Unlock overhead.



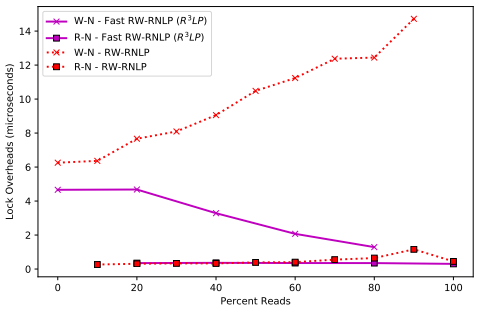

(c) Blocking.



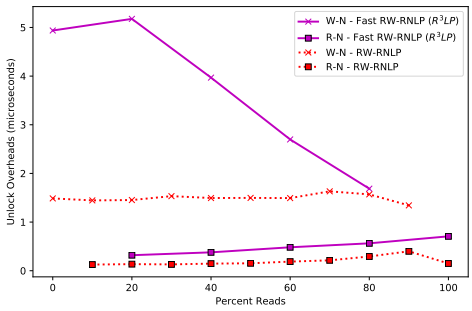

(c) Blocking.

Figure 277: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.

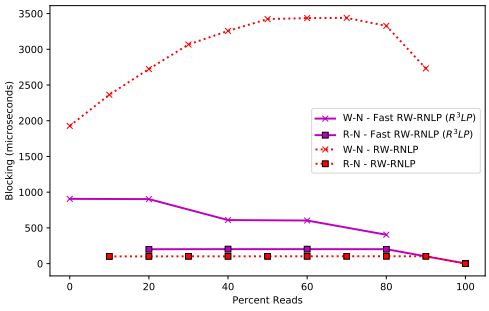
Figure 278: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

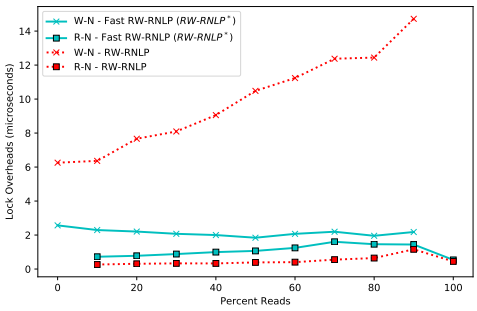



(b) Unlock overhead.

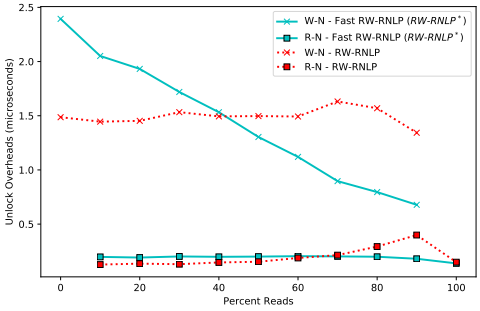



(c) Blocking.

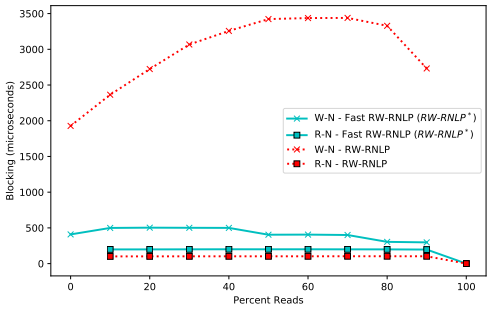
Figure 279: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

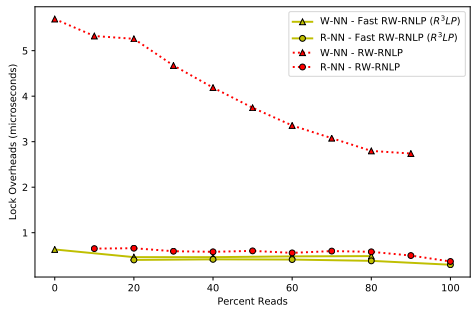



(b) Unlock overhead.

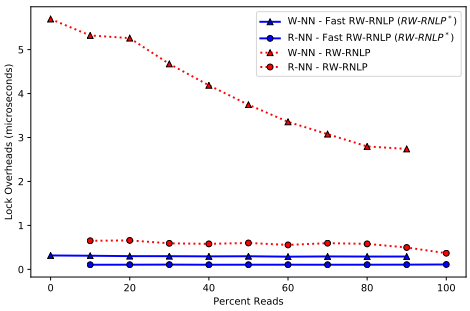



(c) Blocking.

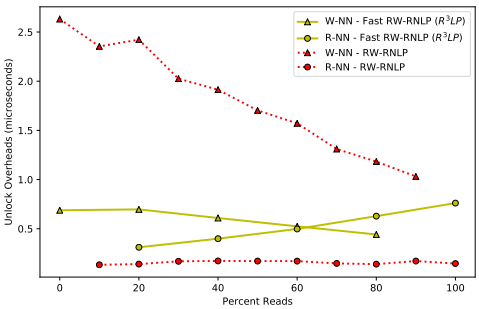
Figure 280: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.



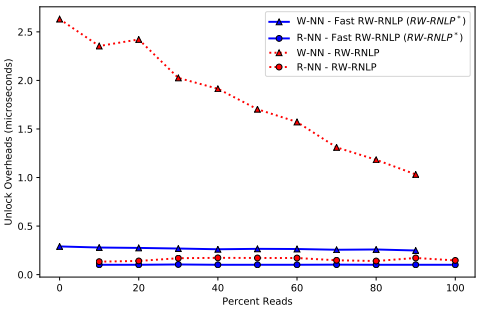

(a) Lock overhead.



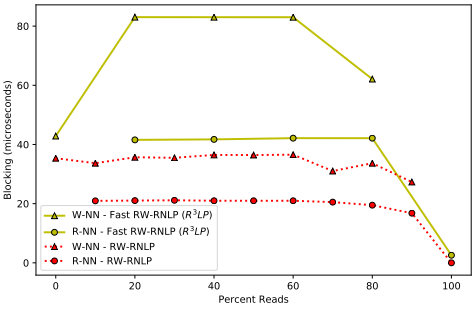

(a) Lock overhead.



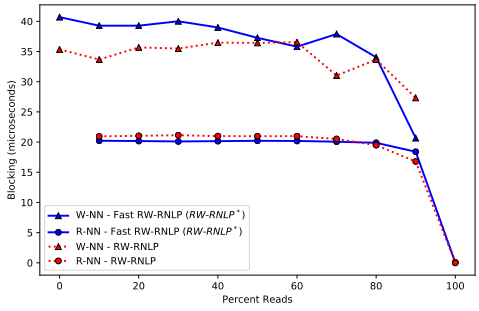

(b) Unlock overhead.




(b) Unlock overhead.



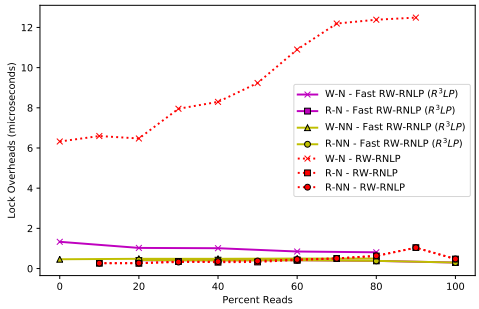

(c) Blocking.



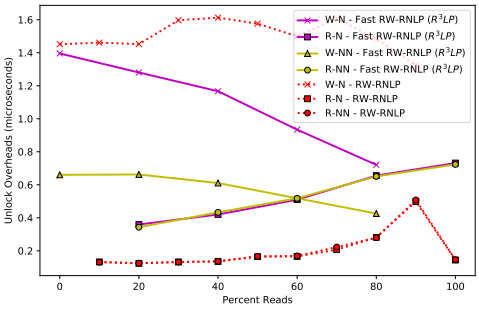

(c) Blocking.

Figure 281: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.

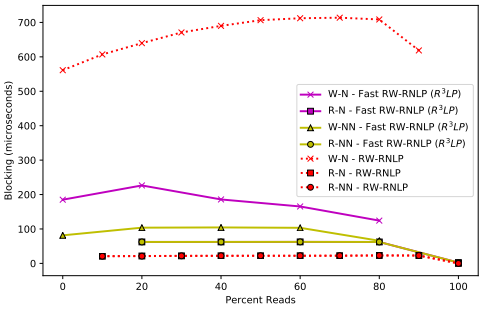
Figure 282: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

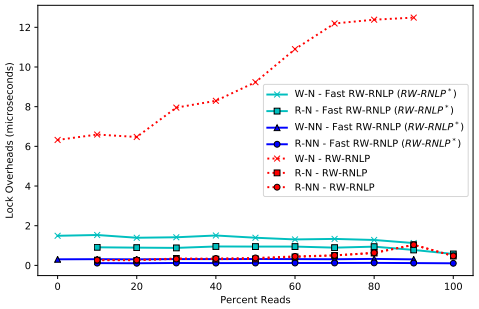



(b) Unlock overhead.

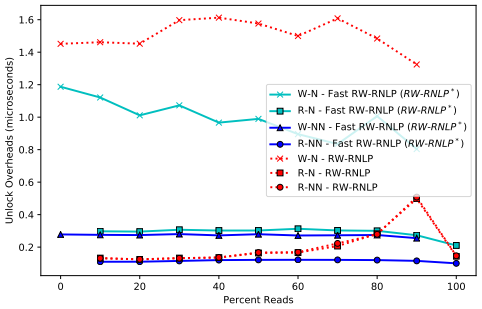



(c) Blocking.

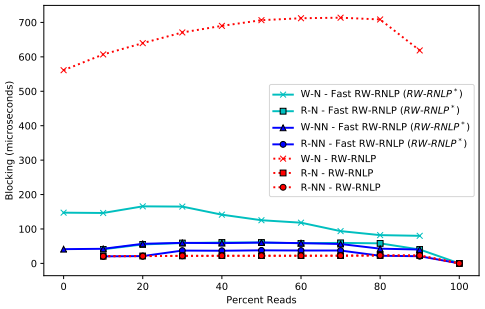
Figure 283: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

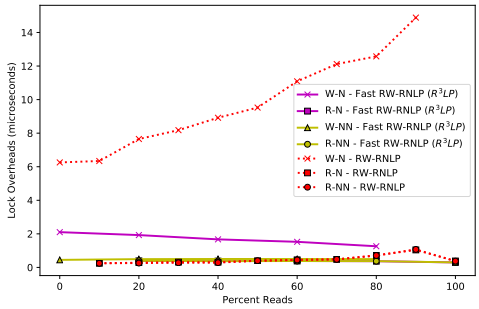



(b) Unlock overhead.

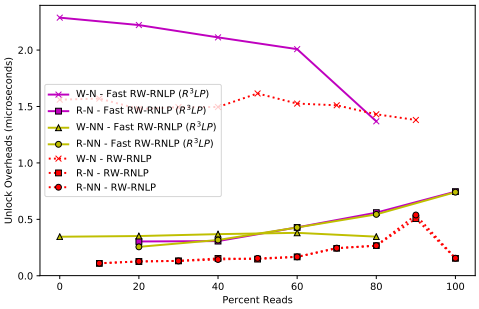



(c) Blocking.

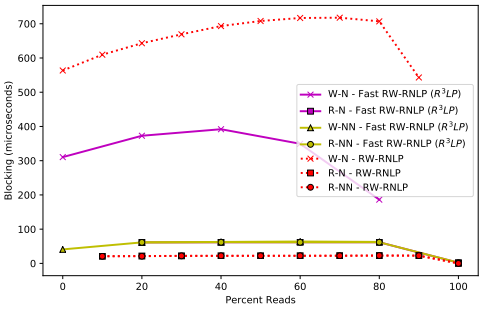
Figure 284: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

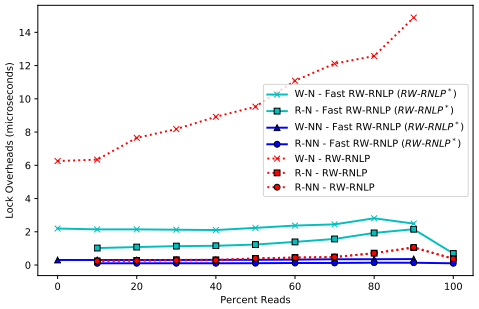



(b) Unlock overhead.

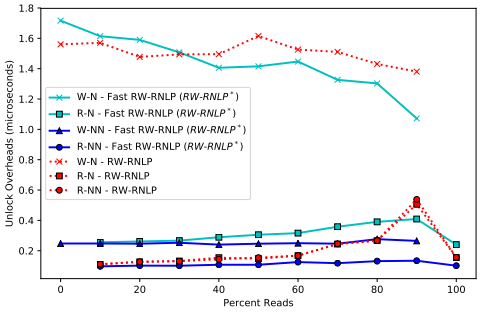



(c) Blocking.

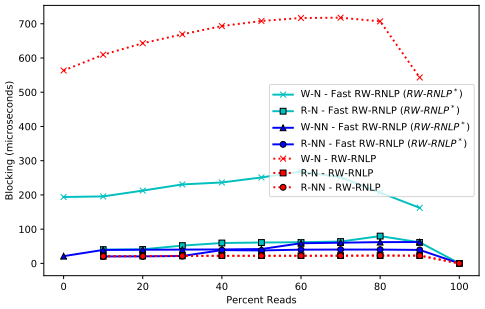
Figure 285: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

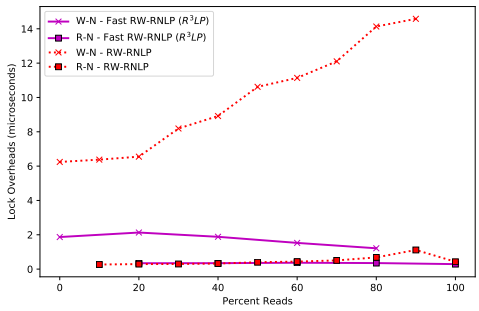



(b) Unlock overhead.

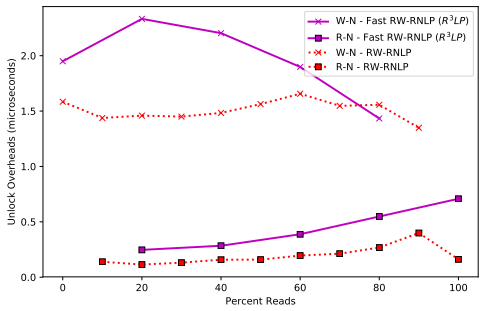



(c) Blocking.

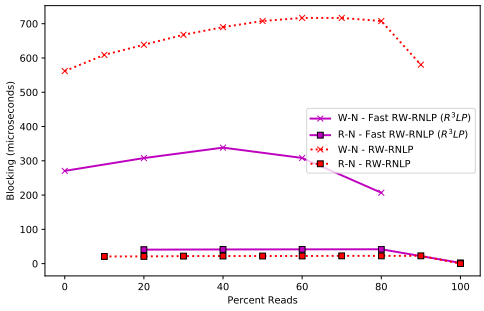
Figure 286: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

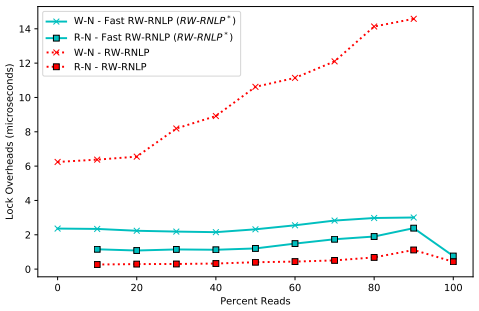



(b) Unlock overhead.

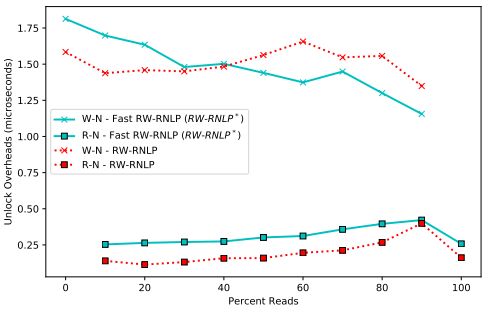



(c) Blocking.

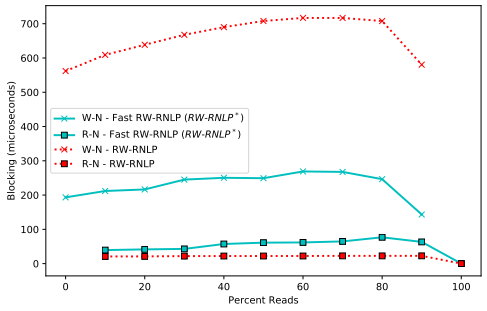
Figure 287: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

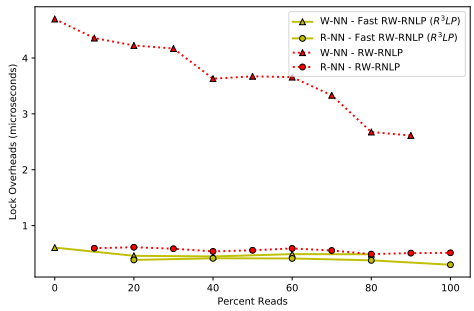



(b) Unlock overhead.

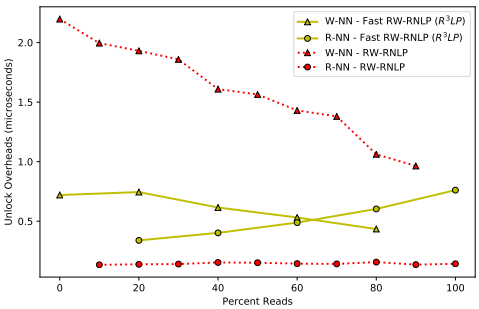



(c) Blocking.

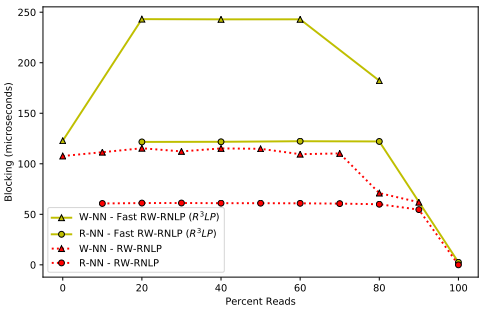
Figure 288: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

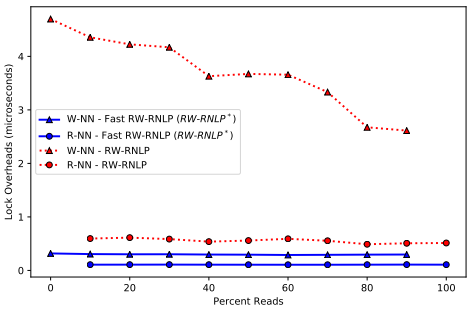



(b) Unlock overhead.

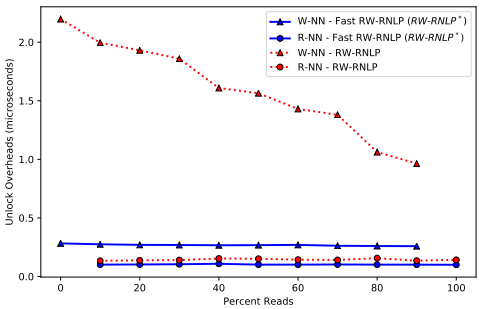



(c) Blocking.

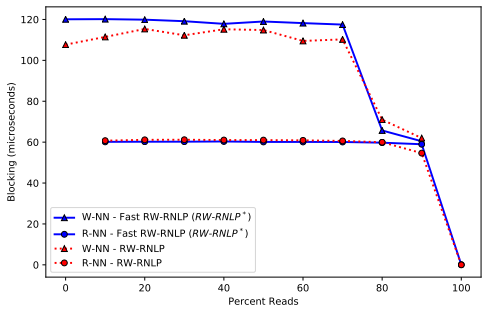
Figure 289: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

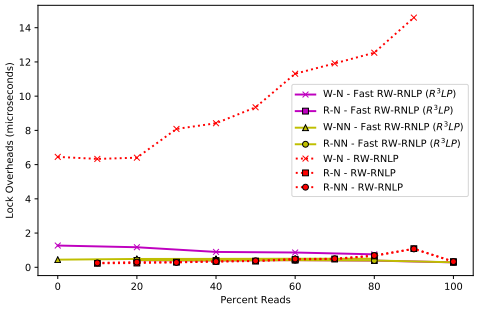



(b) Unlock overhead.

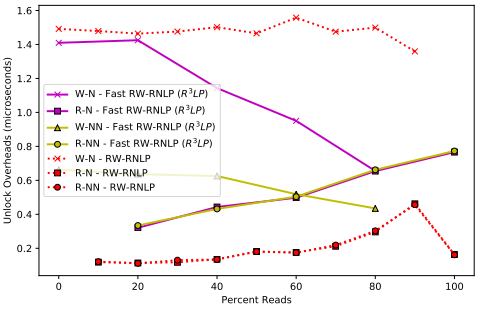



(c) Blocking.

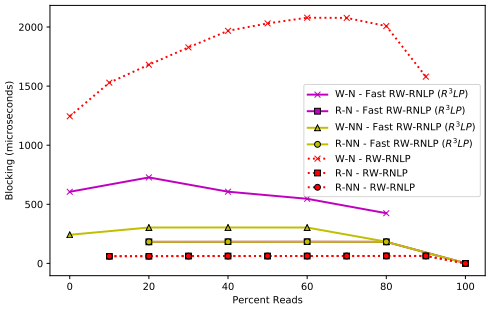
Figure 290: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

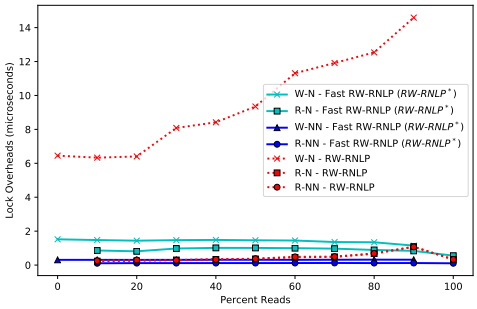



(b) Unlock overhead.

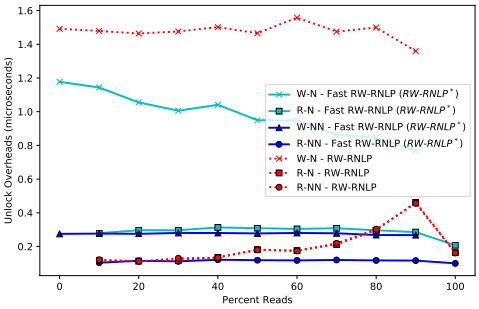



(c) Blocking.

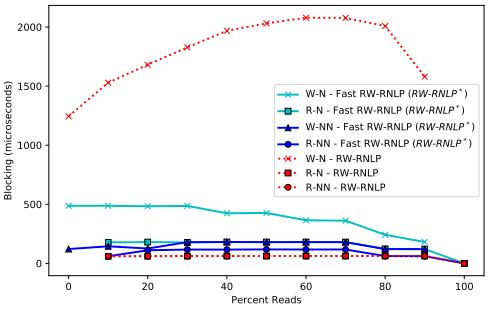
Figure 291: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

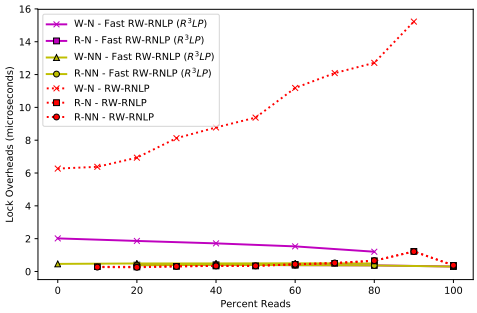



(b) Unlock overhead.

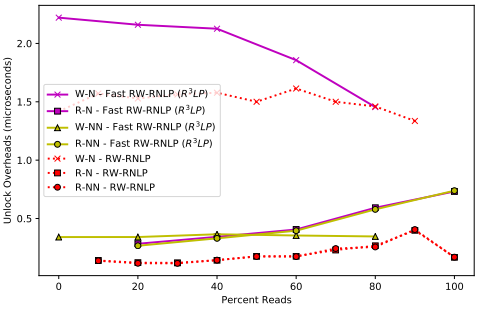



(c) Blocking.

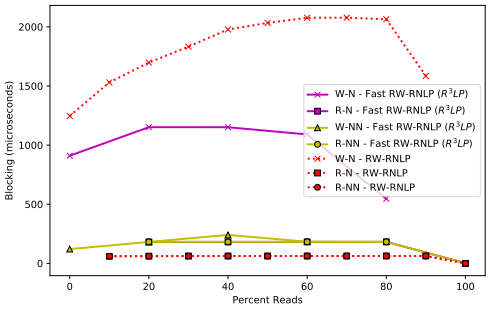
Figure 292: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

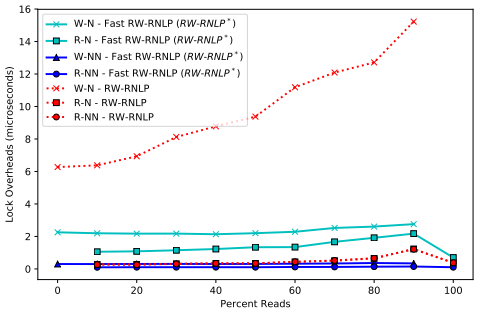



(b) Unlock overhead.

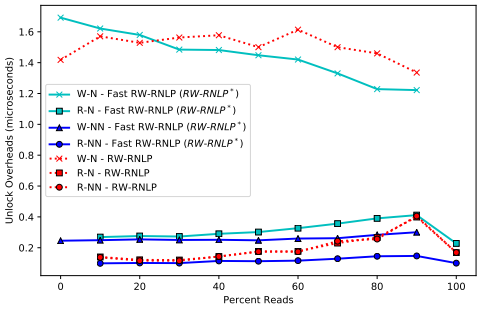



(c) Blocking.

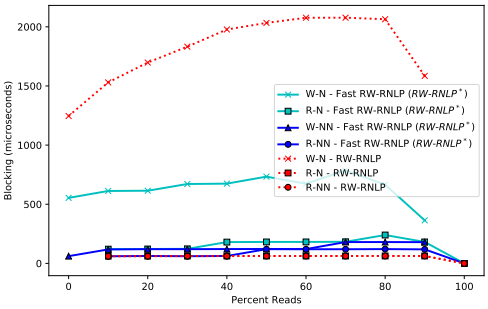
Figure 293: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

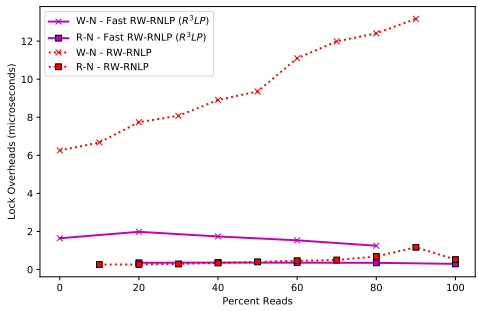



(b) Unlock overhead.

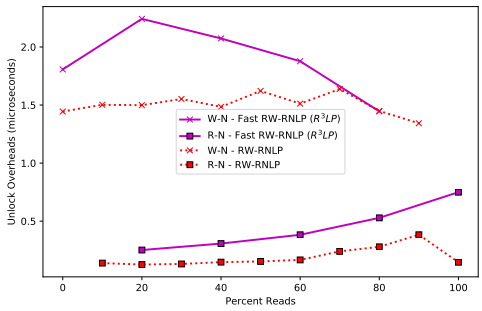



(c) Blocking.

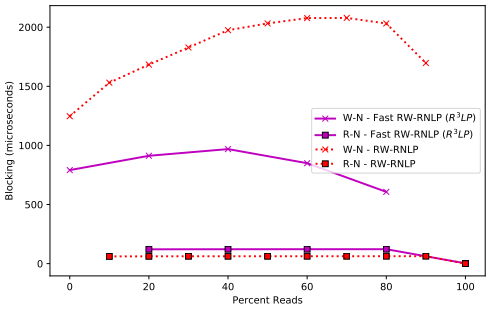
Figure 294: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

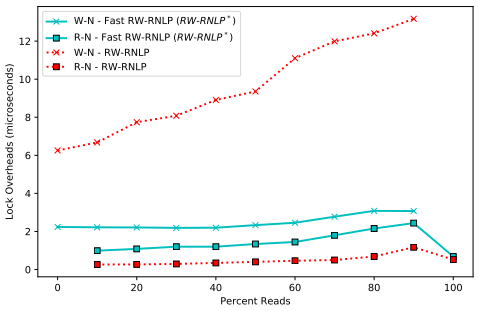



(b) Unlock overhead.

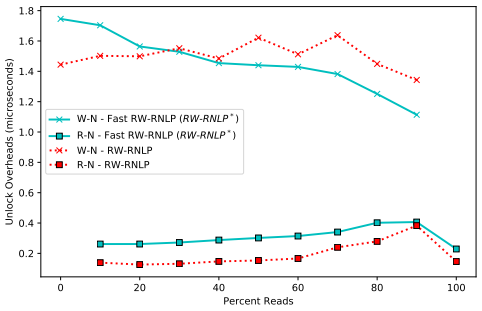



(c) Blocking.

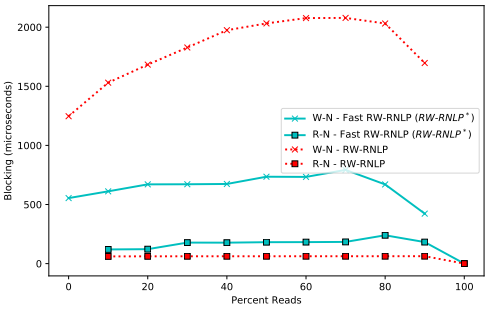
Figure 295: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

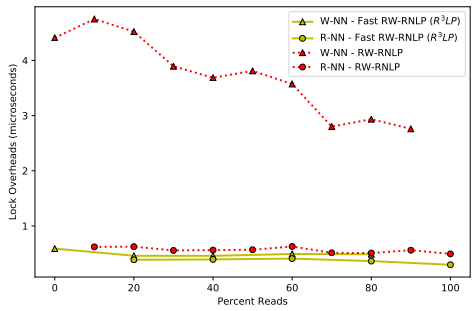



(b) Unlock overhead.

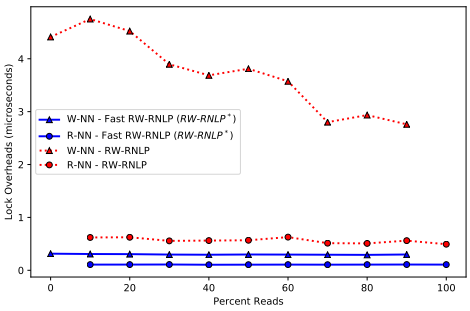



(c) Blocking.

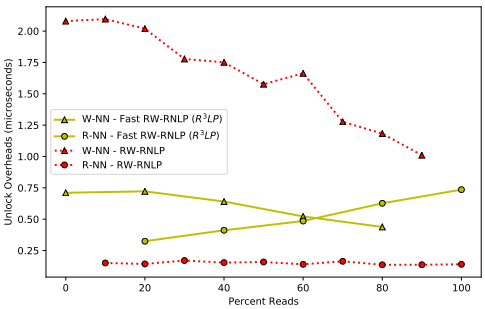
Figure 296: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.



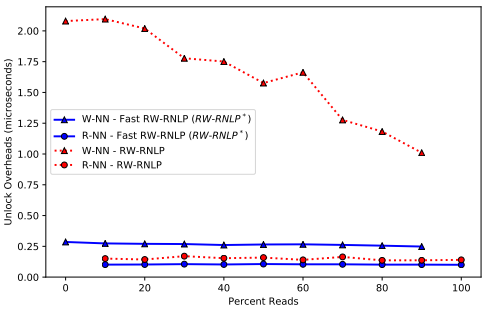

(a) Lock overhead.



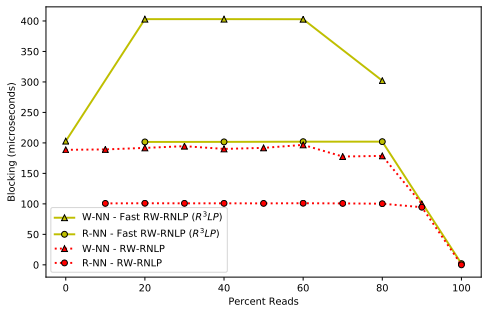

(a) Lock overhead.



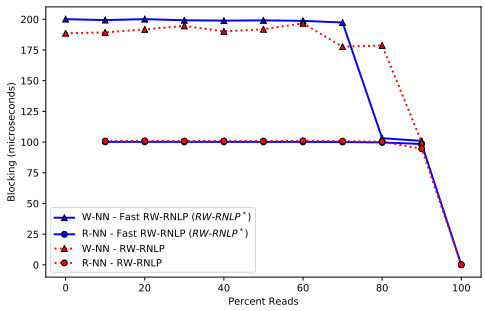

(b) Unlock overhead.




(b) Unlock overhead.



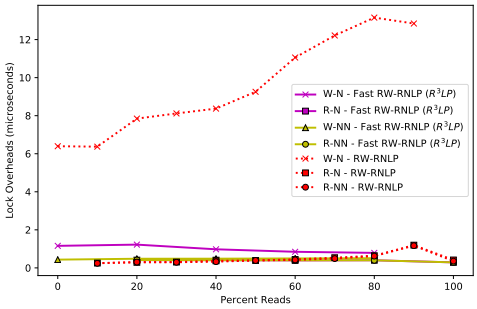

(c) Blocking.



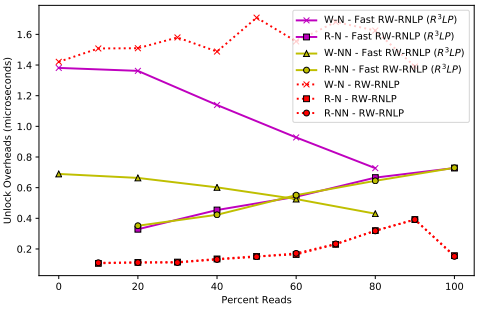

(c) Blocking.

Figure 297: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.

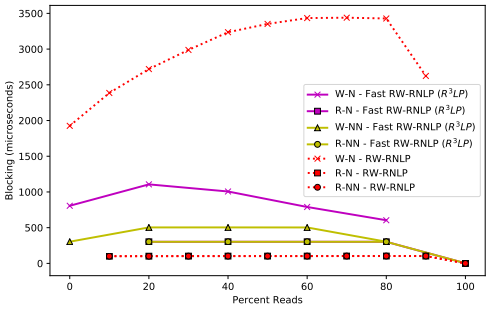
Figure 298: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

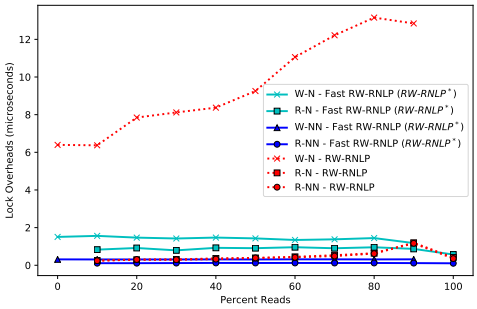



(b) Unlock overhead.

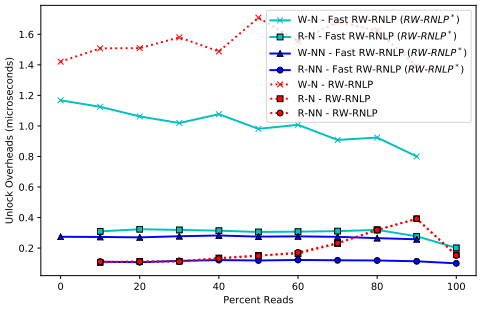



(c) Blocking.

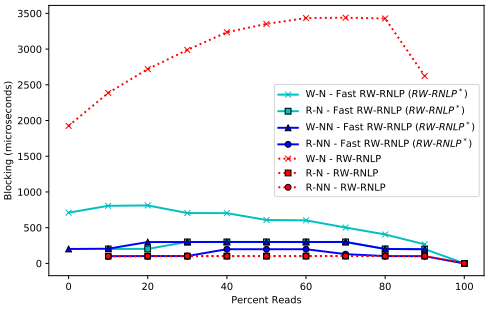
Figure 299: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

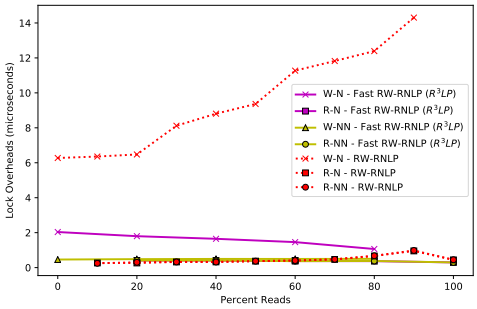



(b) Unlock overhead.

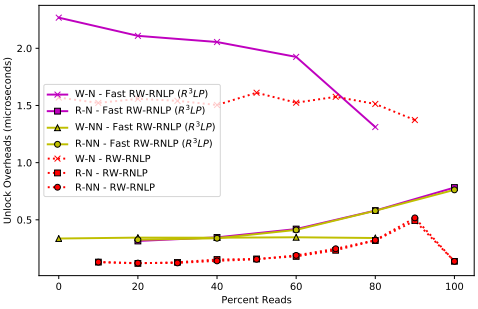



(c) Blocking.

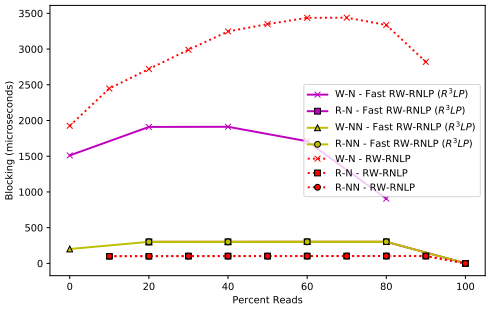
Figure 300: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

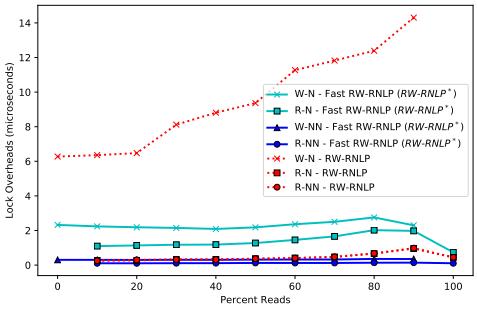



(b) Unlock overhead.

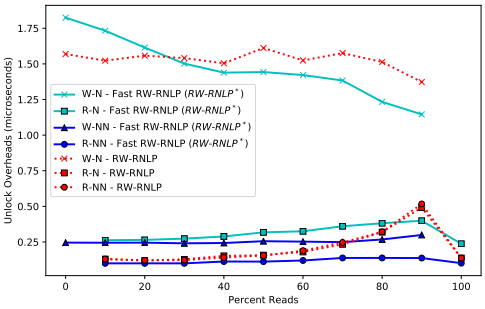



(c) Blocking.

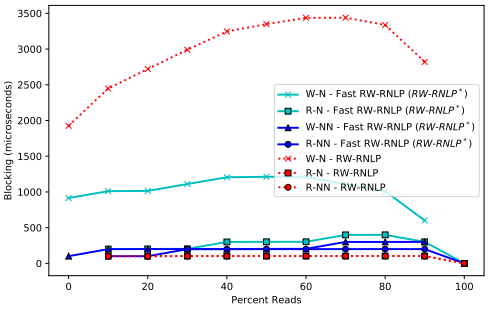
Figure 301: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

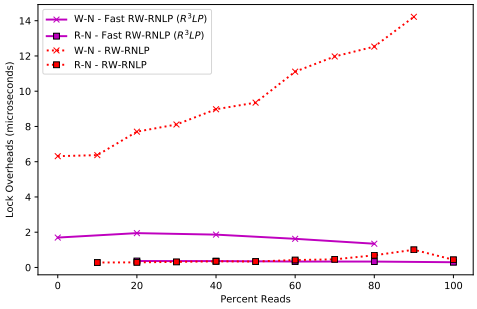



(b) Unlock overhead.

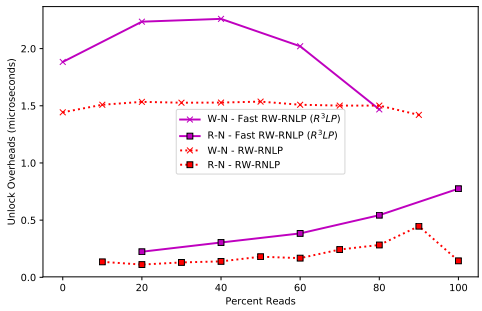



(c) Blocking.

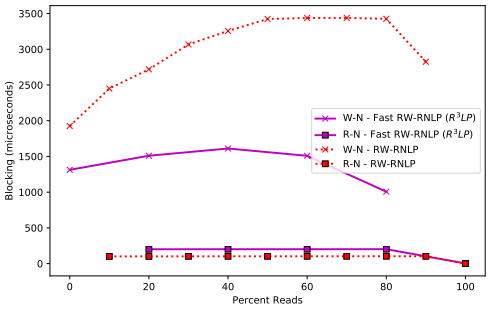
Figure 302: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

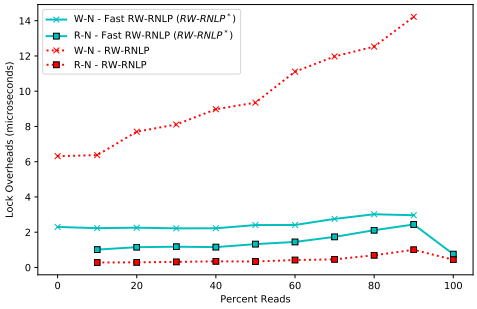



(b) Unlock overhead.

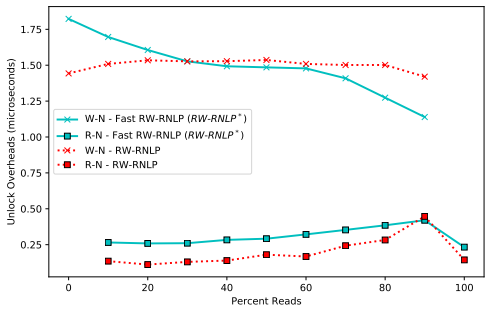



(c) Blocking.

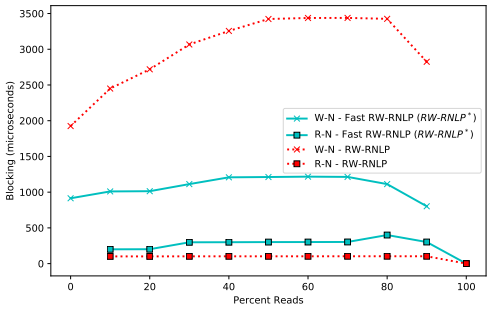
Figure 303: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

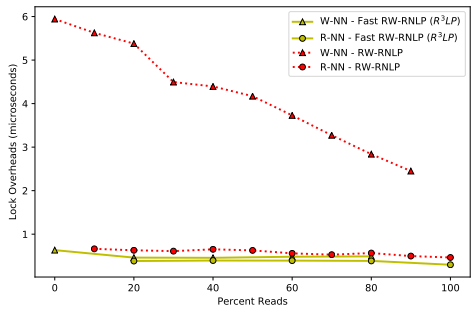



(b) Unlock overhead.

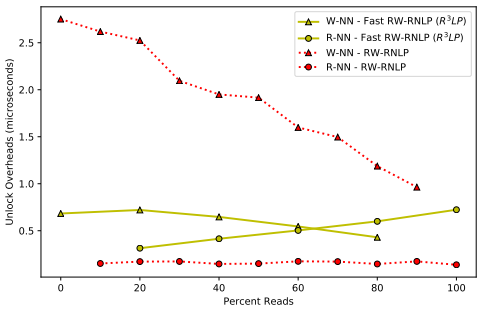



(c) Blocking.

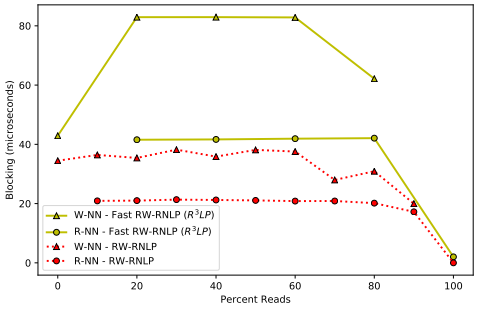
Figure 304: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

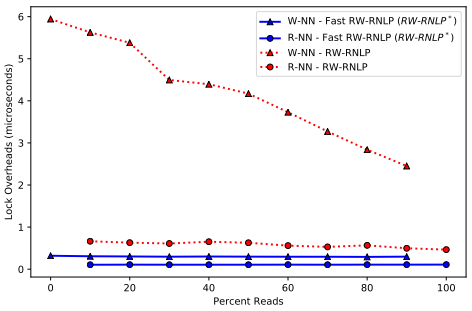



(b) Unlock overhead.

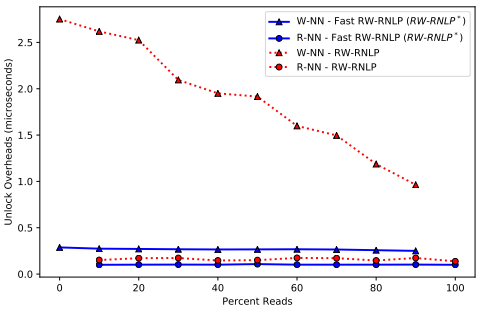



(c) Blocking.

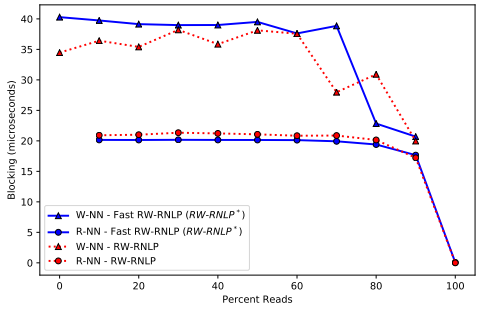
Figure 305: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

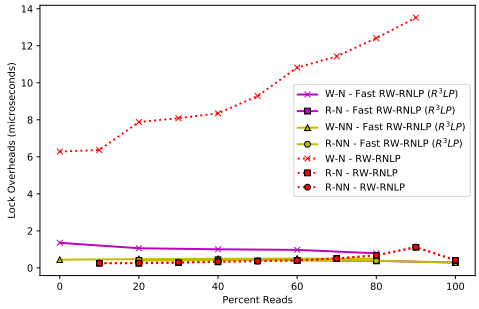



(b) Unlock overhead.

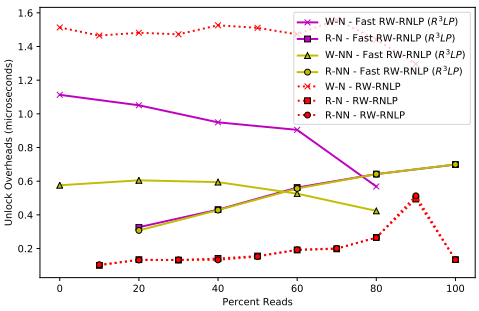



(c) Blocking.

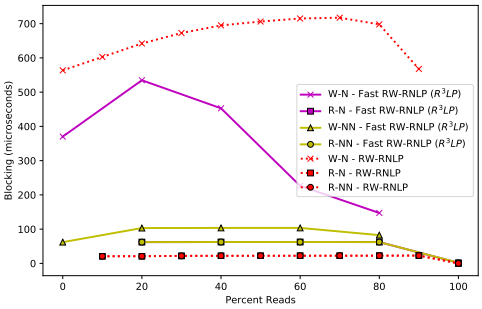
Figure 306: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

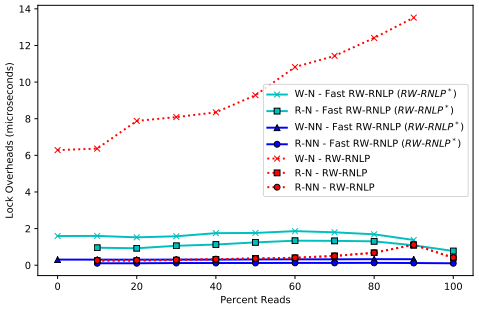



(b) Unlock overhead.

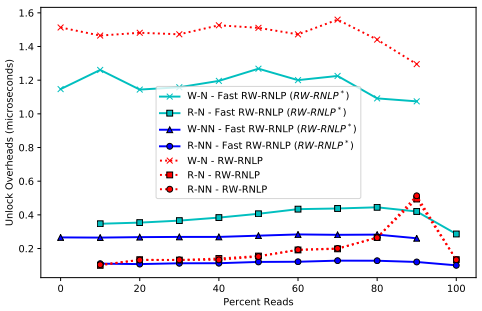



(c) Blocking.

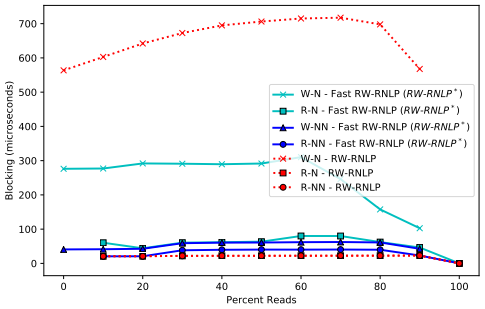
Figure 307: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

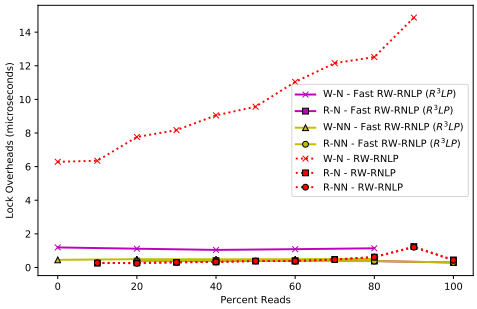



(b) Unlock overhead.

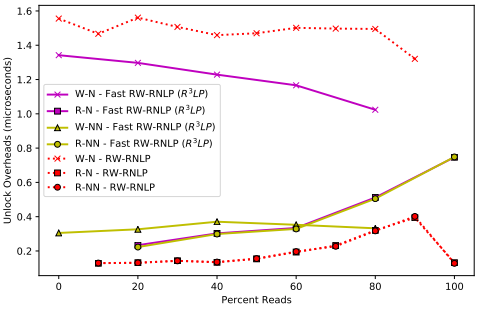



(c) Blocking.

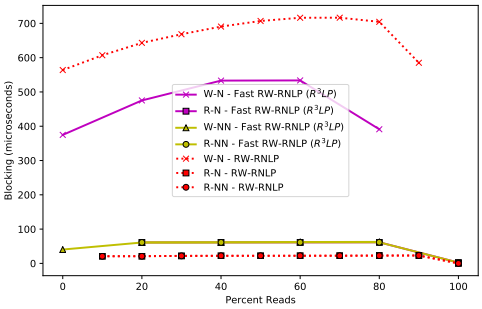
Figure 308: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

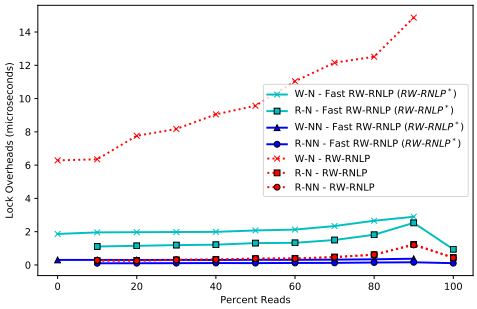



(b) Unlock overhead.

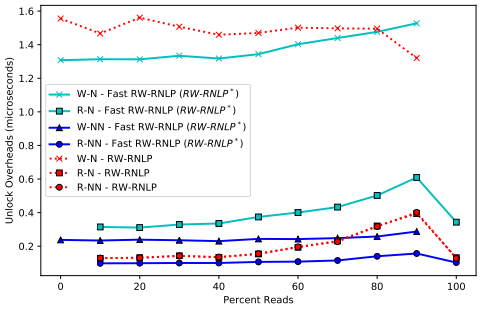



(c) Blocking.

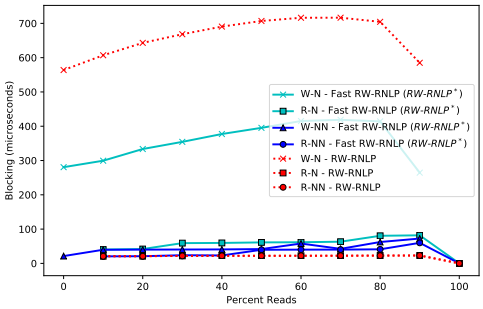
Figure 309: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

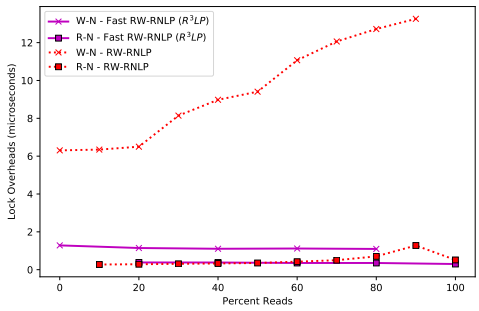



(b) Unlock overhead.

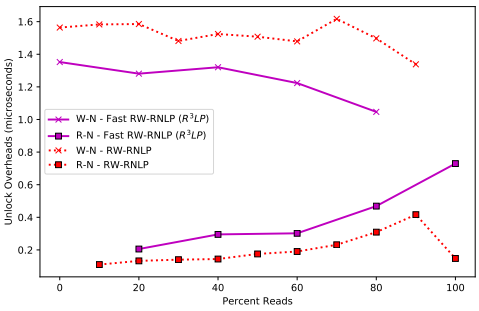



(c) Blocking.

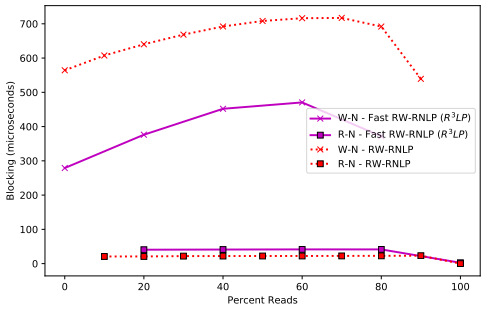
Figure 310: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

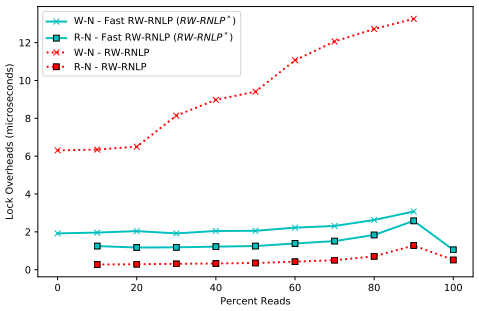



(b) Unlock overhead.

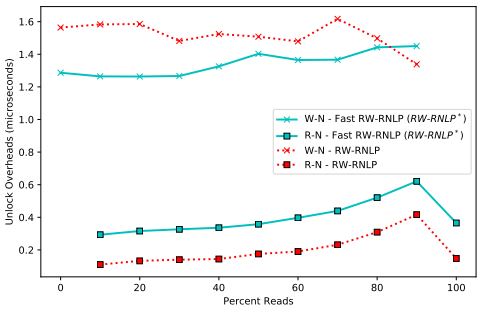



(c) Blocking.

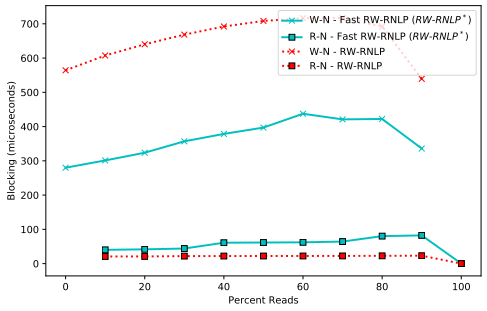
Figure 311: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

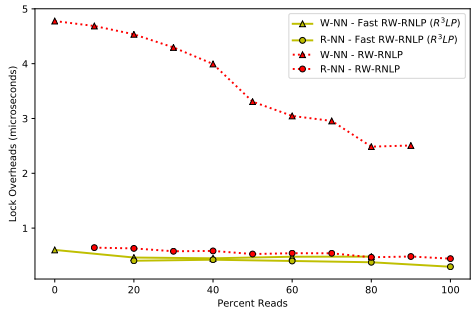



(b) Unlock overhead.

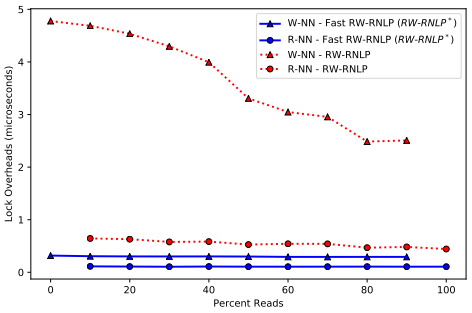



(c) Blocking.

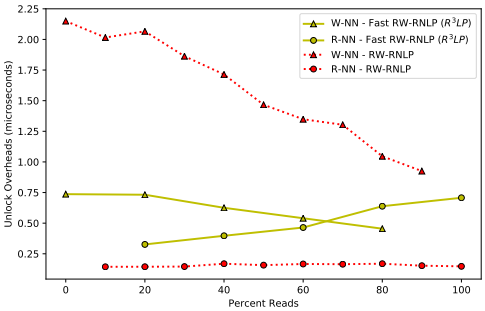
Figure 312: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.



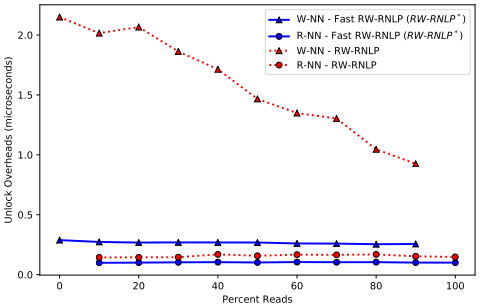

(a) Lock overhead.



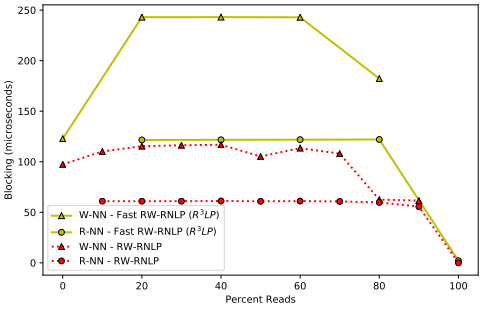

(a) Lock overhead.



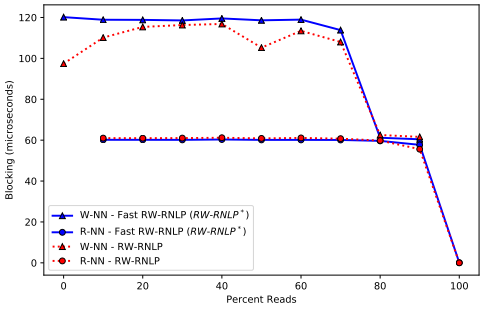

(b) Unlock overhead.




(b) Unlock overhead.



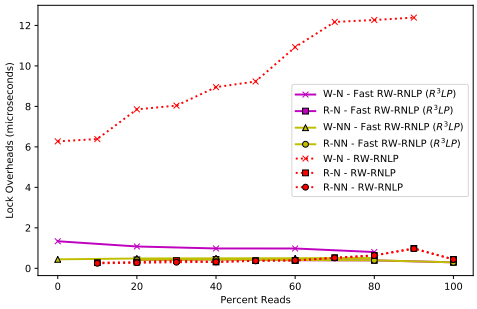

(c) Blocking.



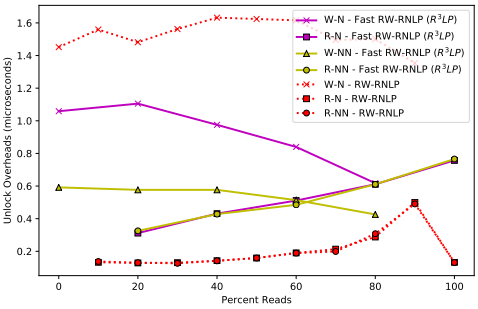

(c) Blocking.

Figure 313: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.

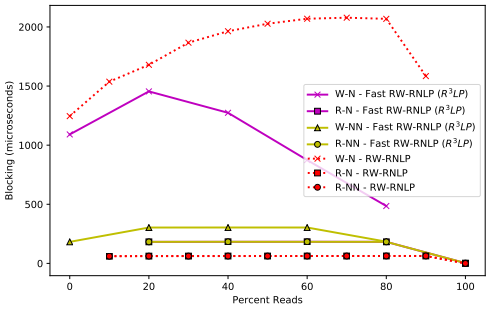
Figure 314: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

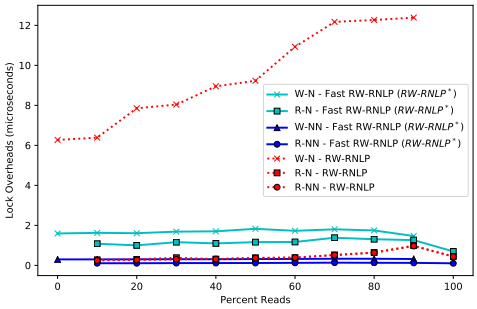



(b) Unlock overhead.

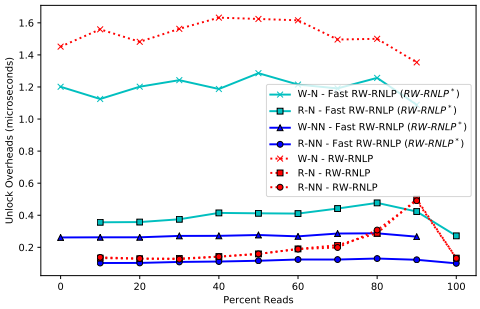



(c) Blocking.

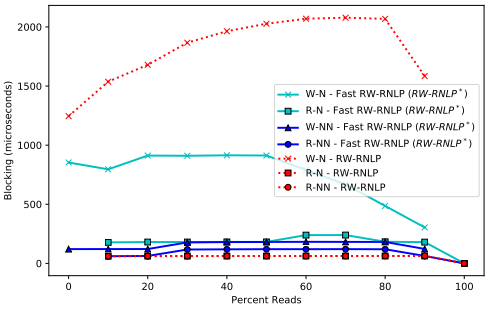
Figure 315: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

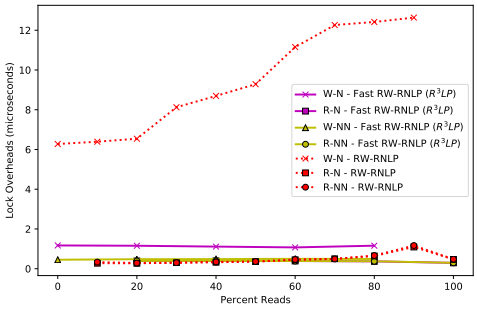



(b) Unlock overhead.

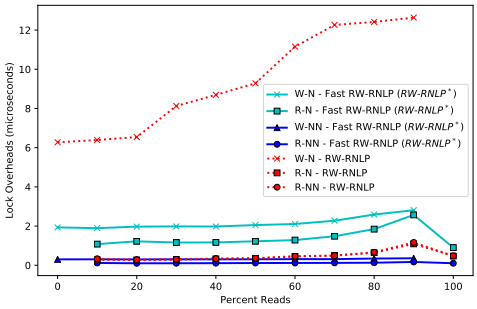



(c) Blocking.

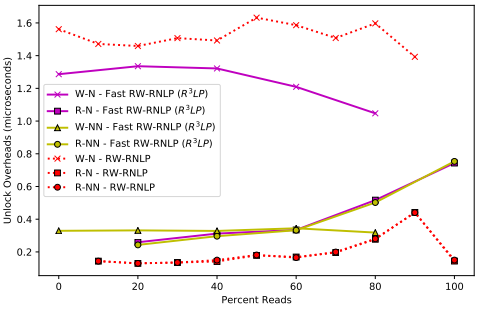
Figure 316: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.



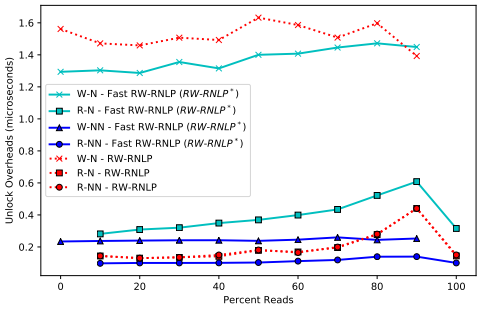

(a) Lock overhead.



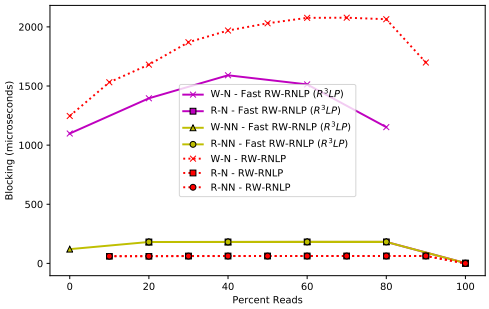

(a) Lock overhead.



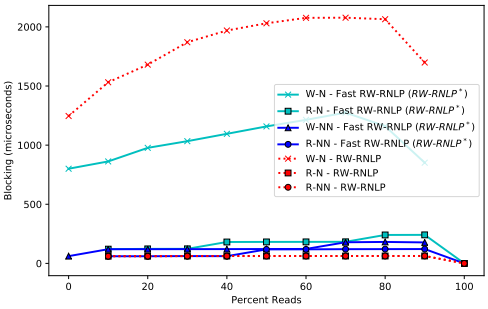

(b) Unlock overhead.




(b) Unlock overhead.



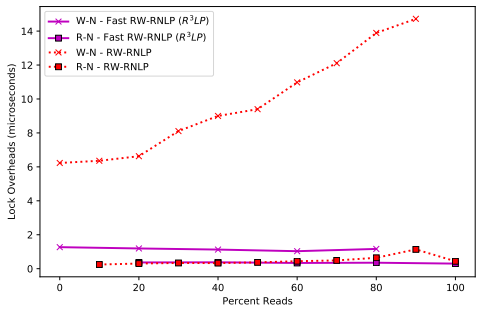

(c) Blocking.



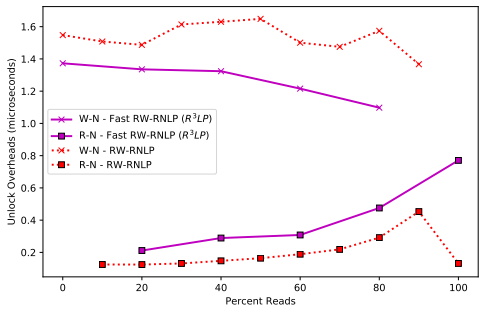

(c) Blocking.

Figure 317: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.

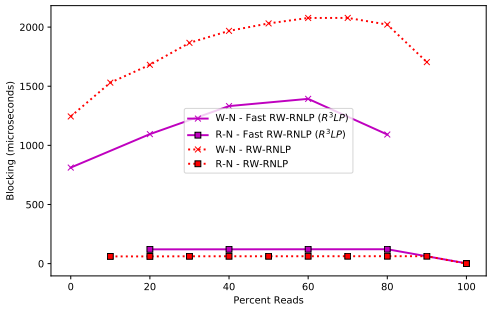
Figure 318: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

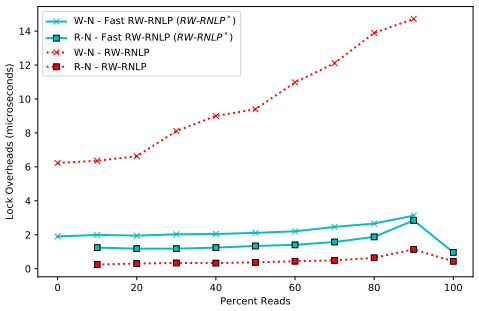



(b) Unlock overhead.

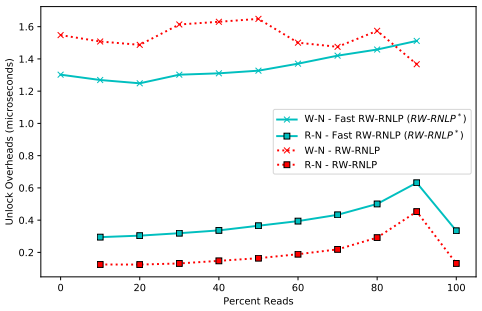



(c) Blocking.

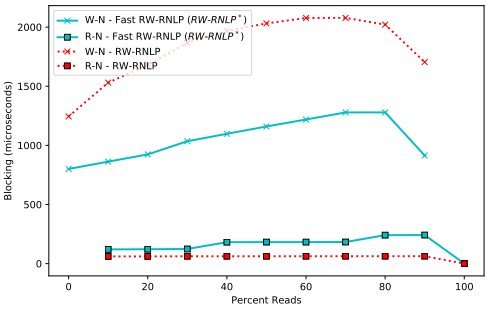
Figure 319: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

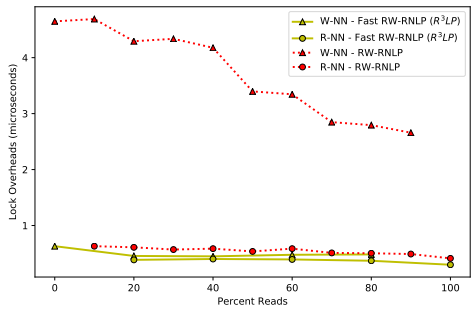



(b) Unlock overhead.

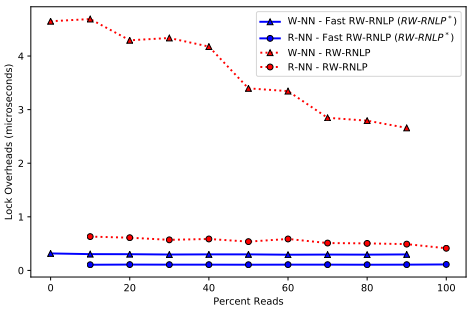



(c) Blocking.

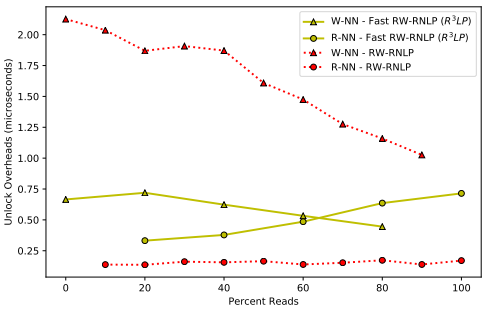
Figure 320: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.



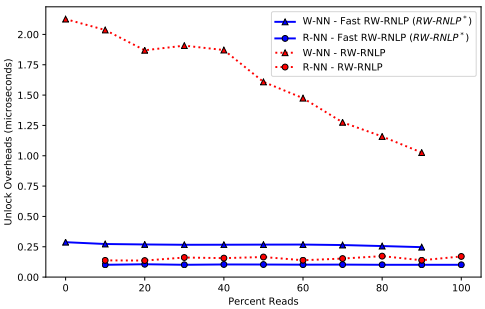

(a) Lock overhead.



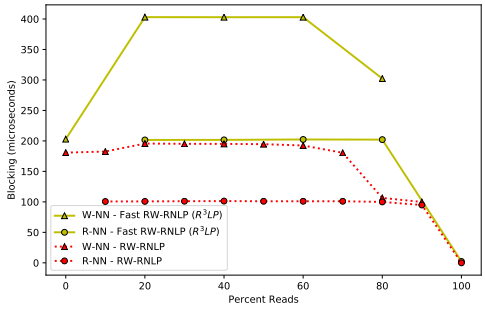

(a) Lock overhead.



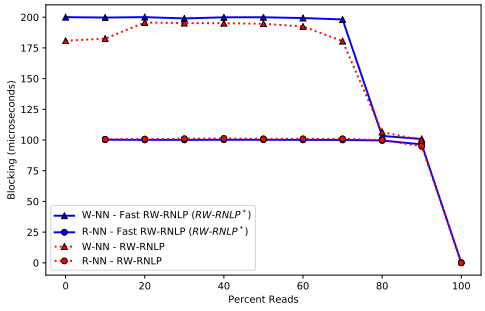

(b) Unlock overhead.




(b) Unlock overhead.



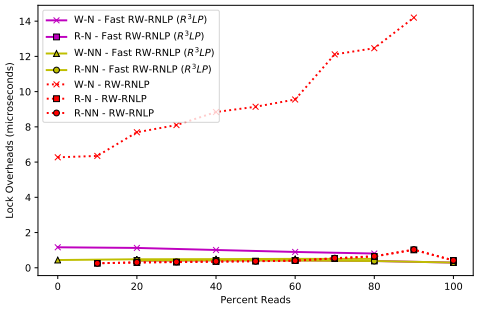

(c) Blocking.



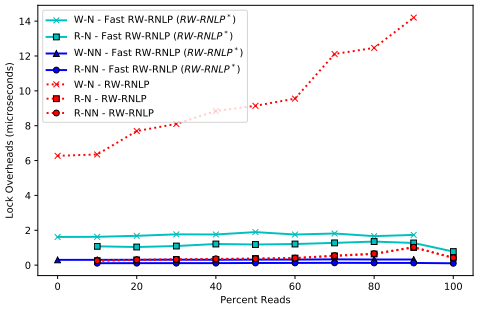

(c) Blocking.

Figure 321: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.

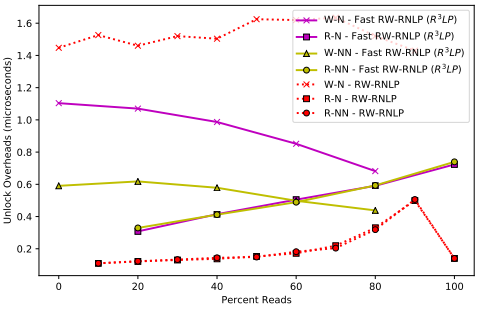
Figure 322: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.



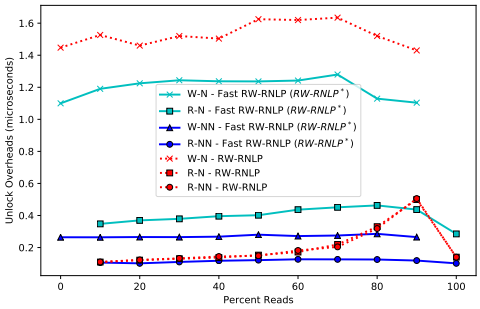

(a) Lock overhead.



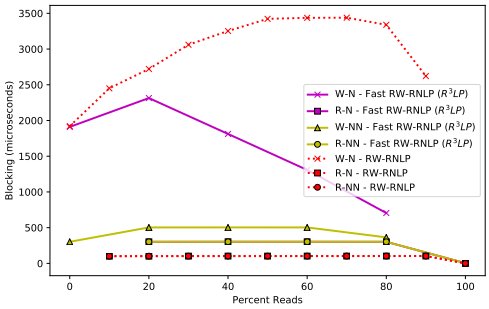

(a) Lock overhead.



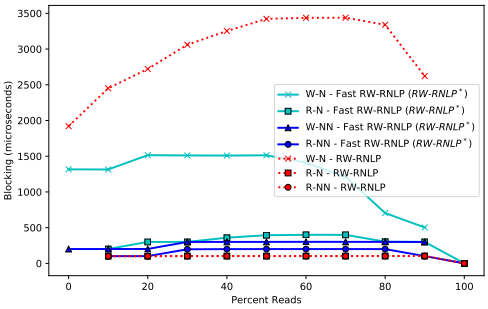

(b) Unlock overhead.




(b) Unlock overhead.



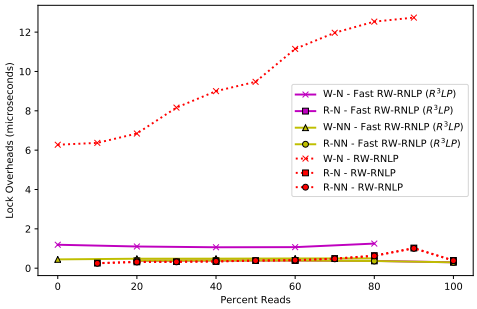

(c) Blocking.



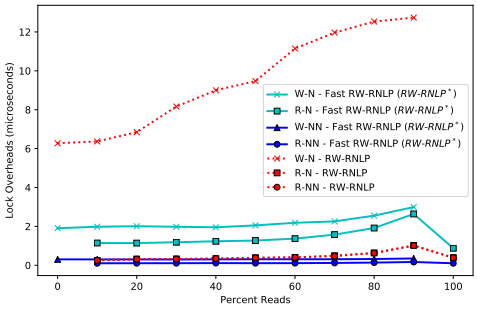

(c) Blocking.

Figure 323: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.

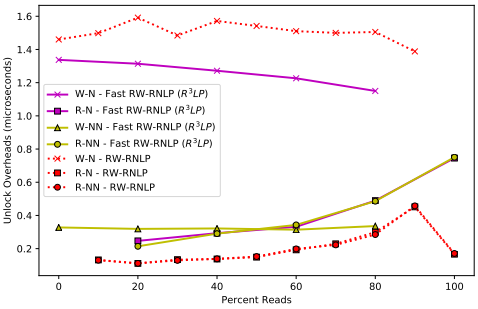
Figure 324: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.



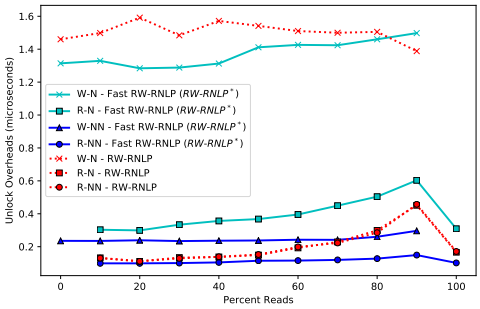

(a) Lock overhead.



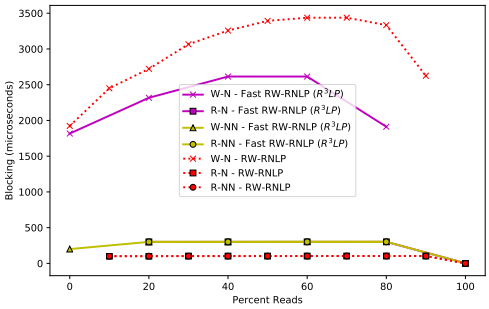

(a) Lock overhead.



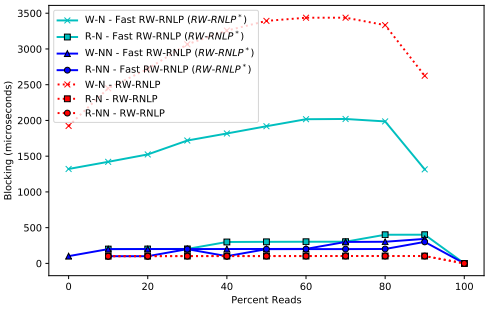

(b) Unlock overhead.




(b) Unlock overhead.



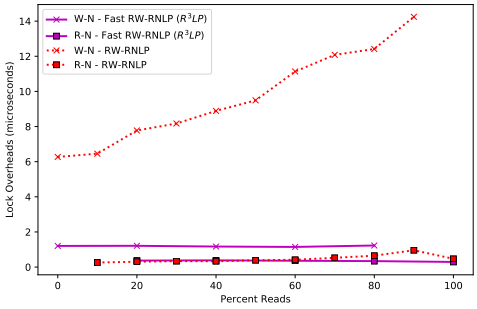

(c) Blocking.



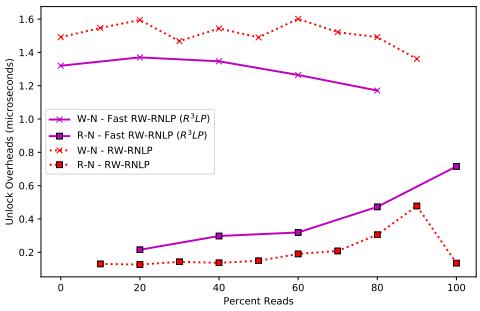

(c) Blocking.

Figure 325: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.

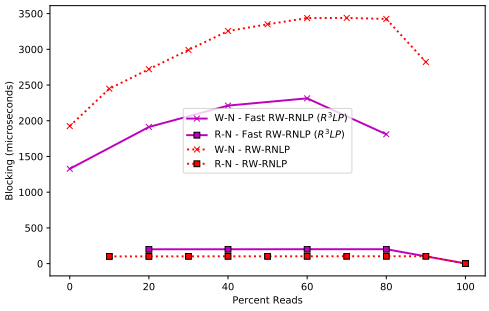
Figure 326: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

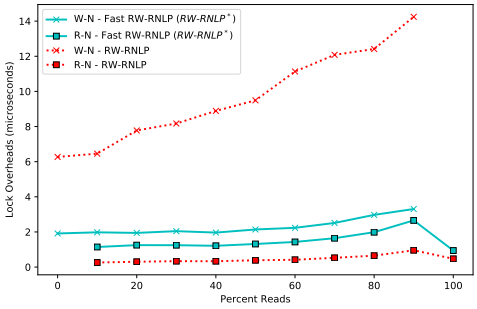



(b) Unlock overhead.

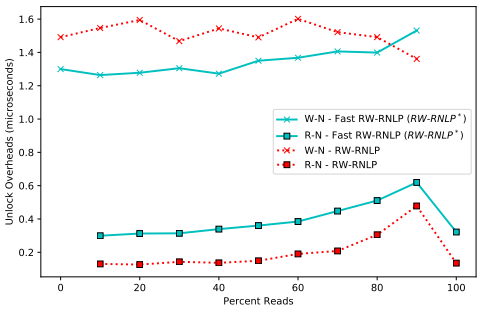



(c) Blocking.

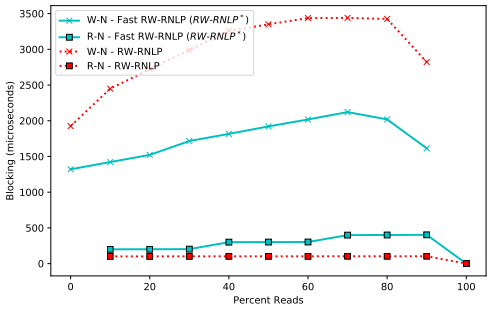
Figure 327: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

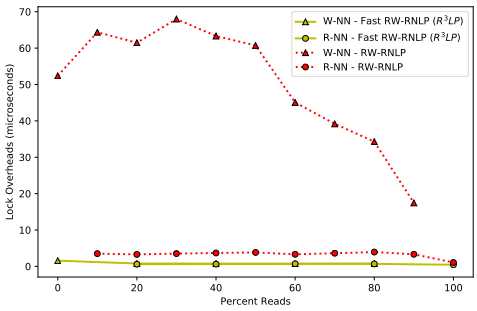



(b) Unlock overhead.

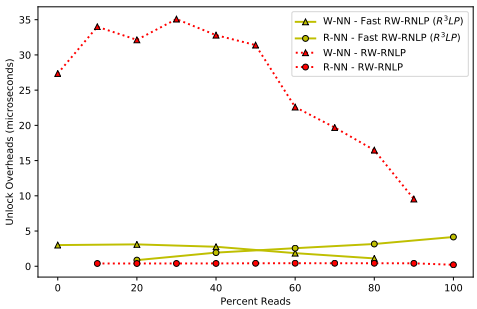



(c) Blocking.

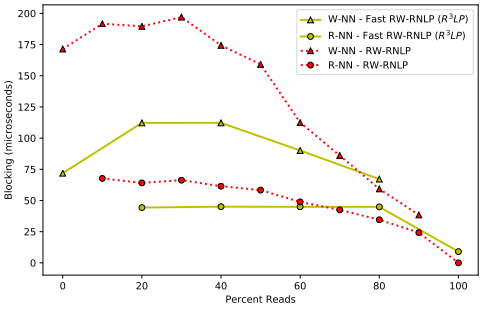
Figure 328: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

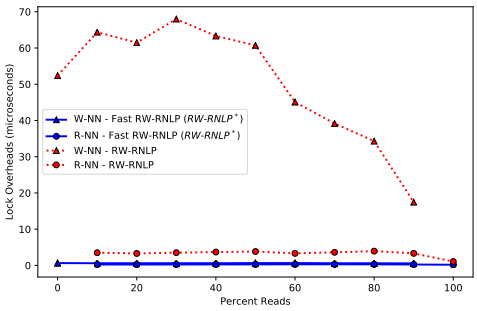



(b) Unlock overhead.

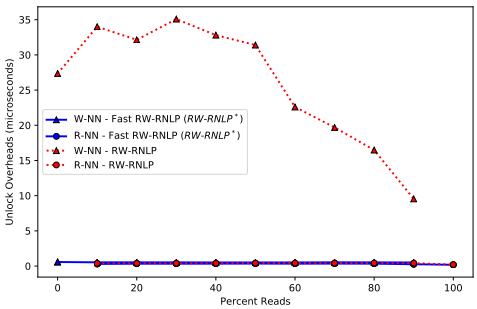



(c) Blocking.

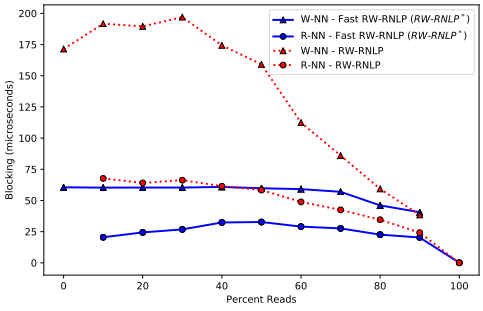
Figure 329: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

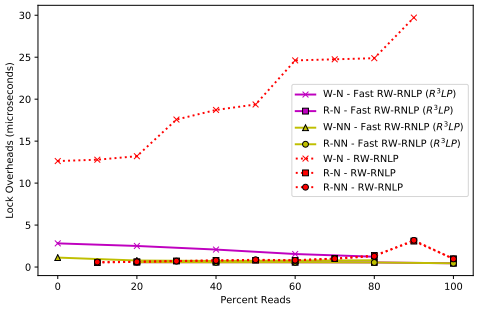



(b) Unlock overhead.

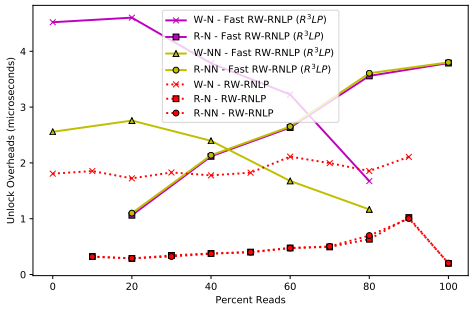



(c) Blocking.

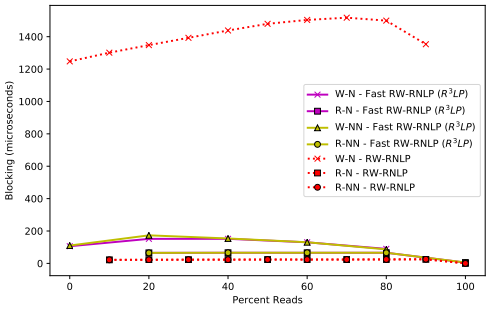
Figure 330: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

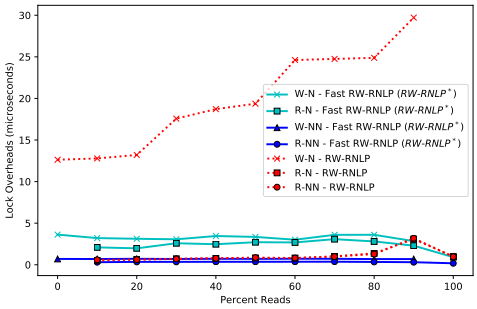



(b) Unlock overhead.

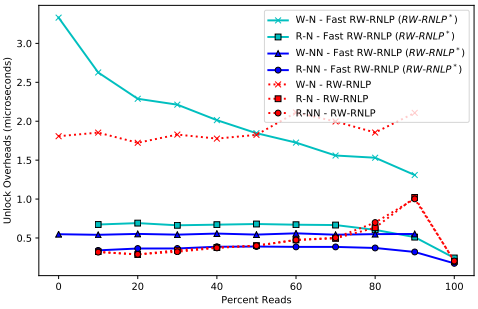



(c) Blocking.

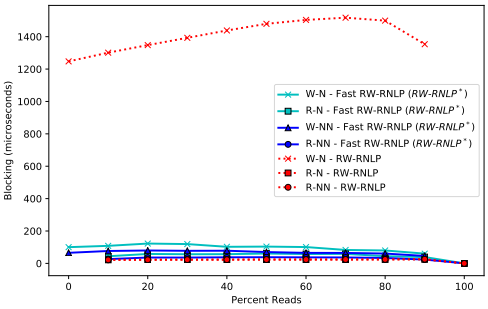
Figure 331: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

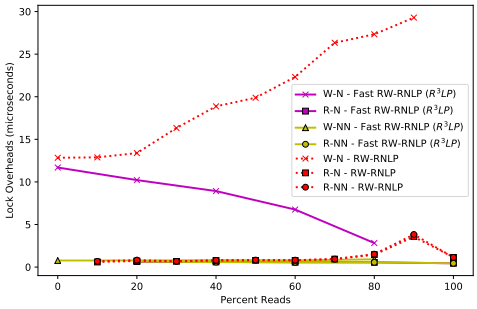



(b) Unlock overhead.

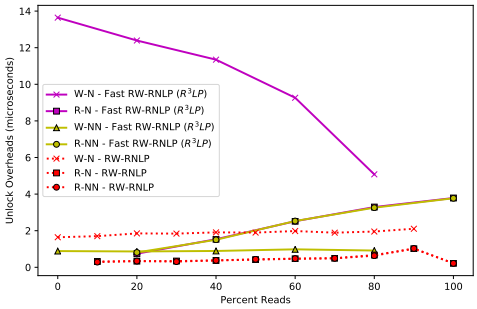



(c) Blocking.

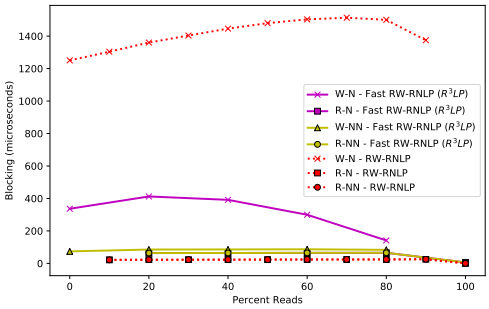
Figure 332: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

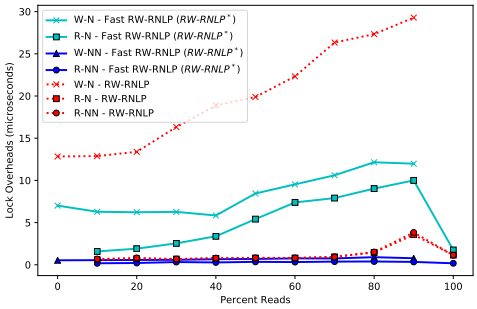



(b) Unlock overhead.

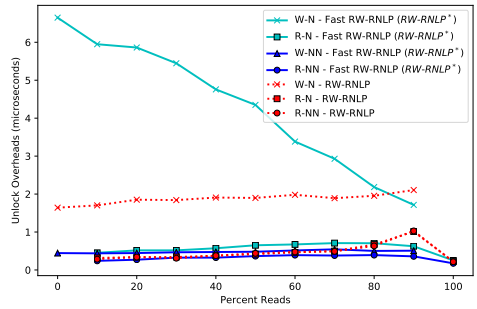



(c) Blocking.

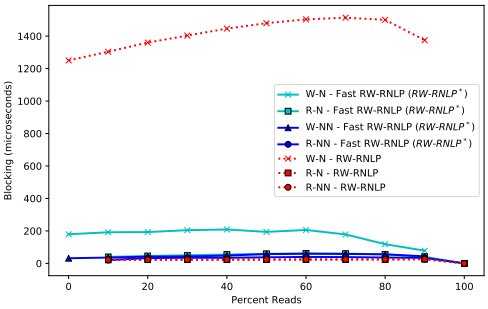
Figure 333: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

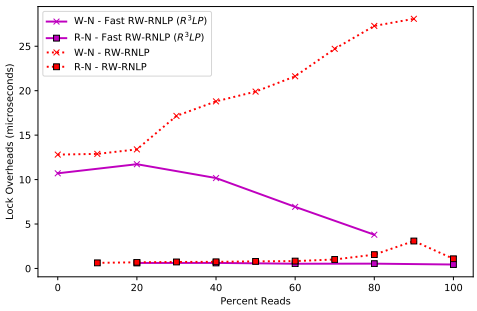



(b) Unlock overhead.

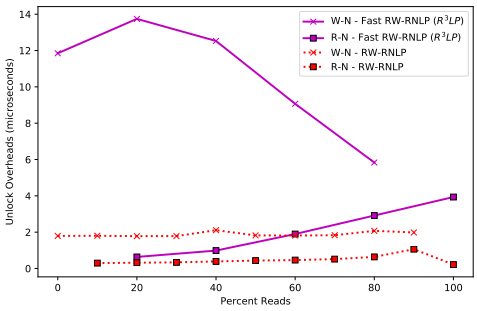



(c) Blocking.

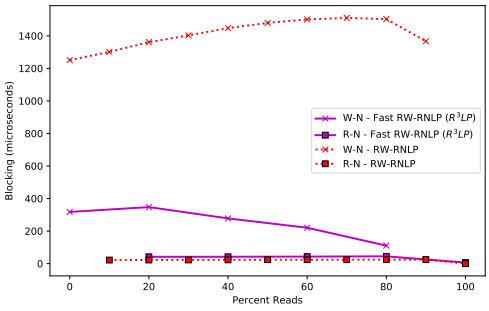
Figure 334: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

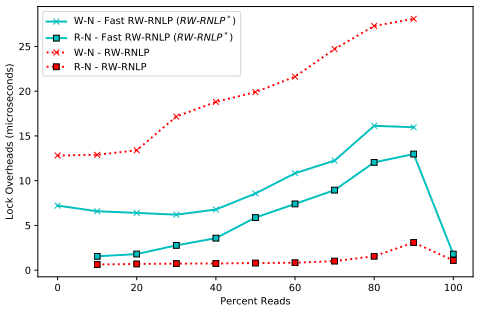



(b) Unlock overhead.

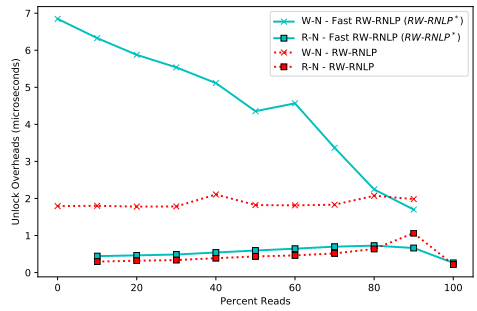



(c) Blocking.

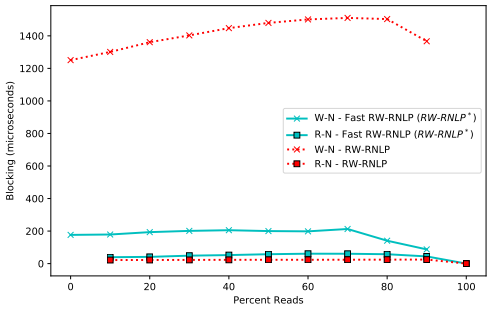
Figure 335: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

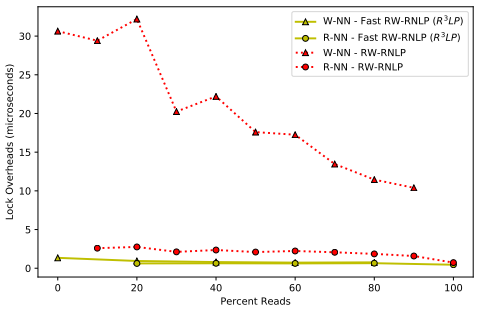



(b) Unlock overhead.

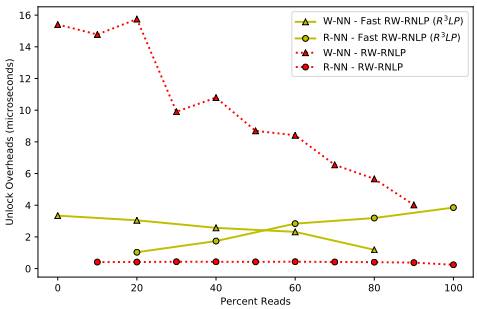



(c) Blocking.

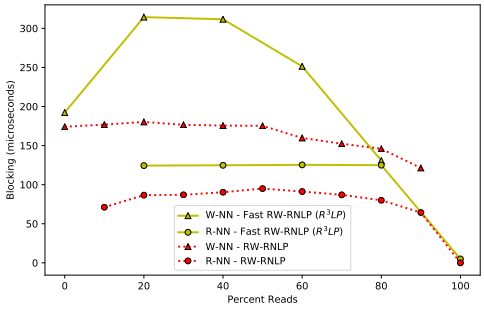
Figure 336: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

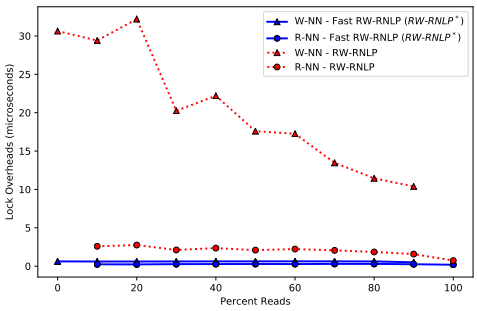



(b) Unlock overhead.

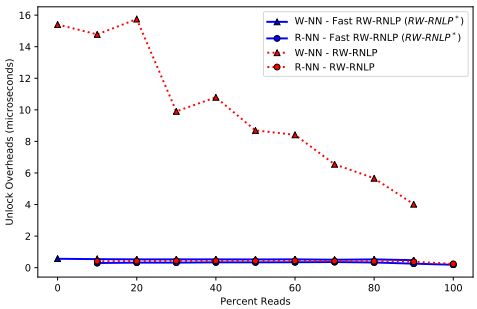



(c) Blocking.

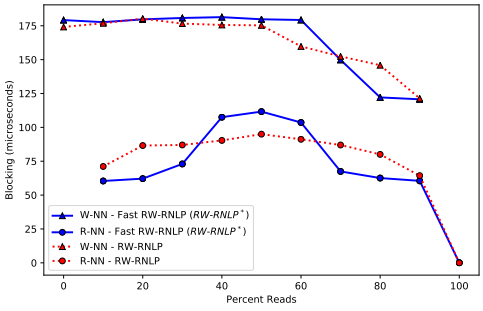
Figure 337: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

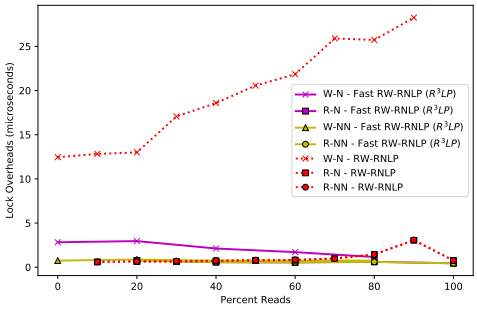



(b) Unlock overhead.

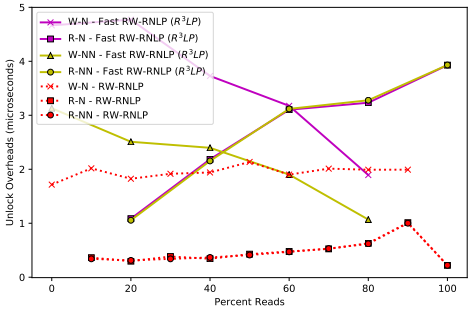



(c) Blocking.

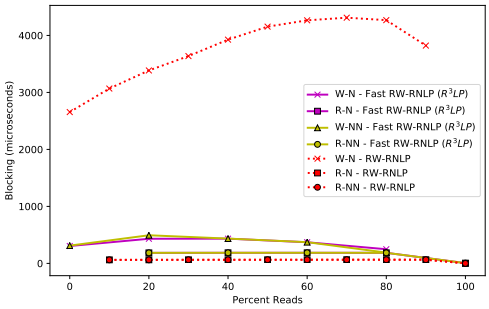
Figure 338: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

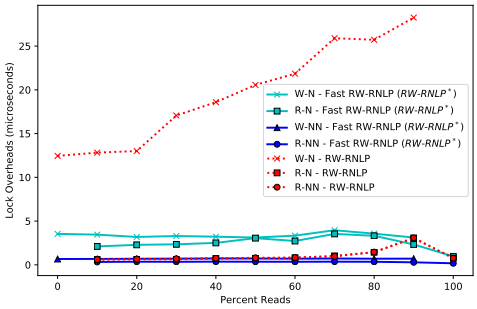



(b) Unlock overhead.

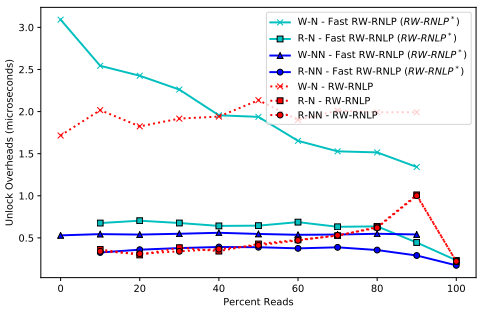



(c) Blocking.

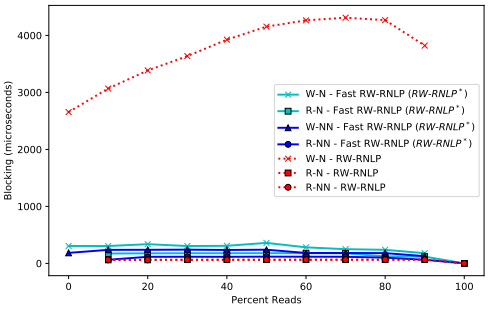
Figure 339: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

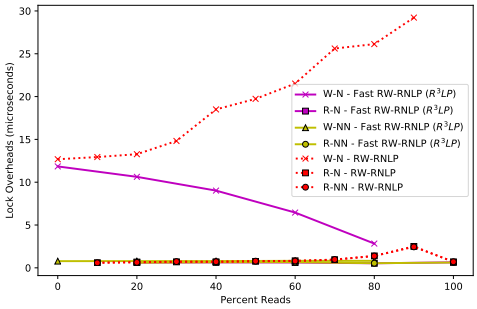



(b) Unlock overhead.

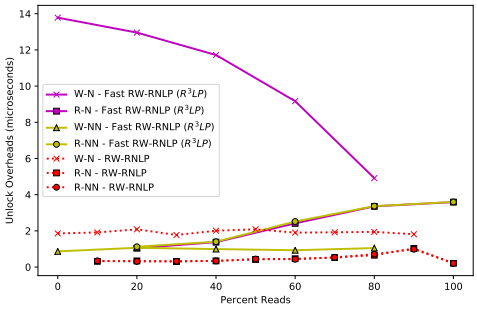



(c) Blocking.

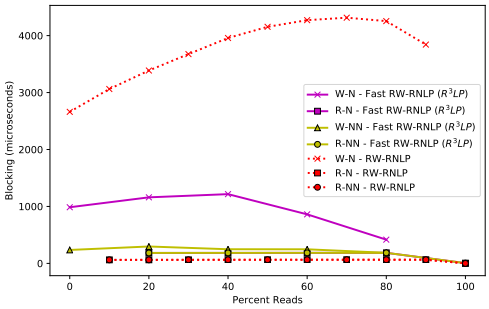
Figure 340: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

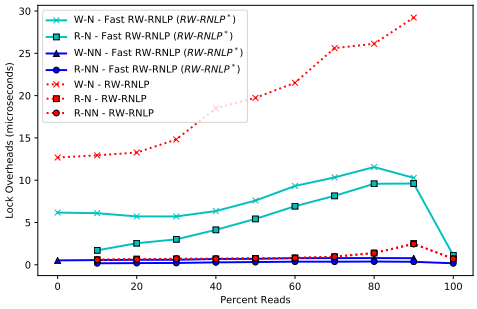



(b) Unlock overhead.

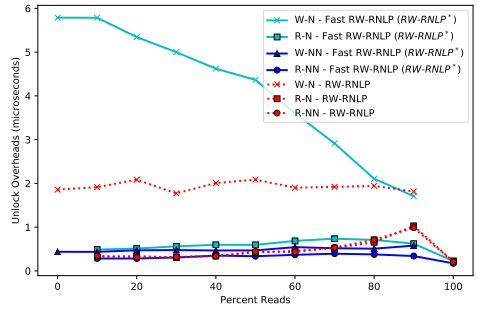



(c) Blocking.

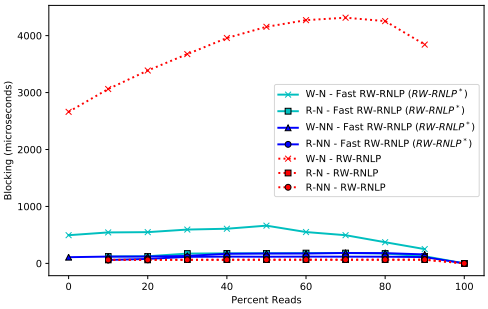
Figure 341: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

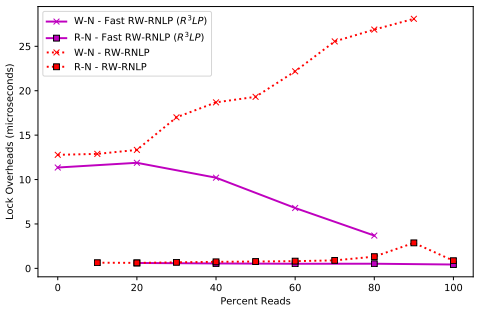



(b) Unlock overhead.

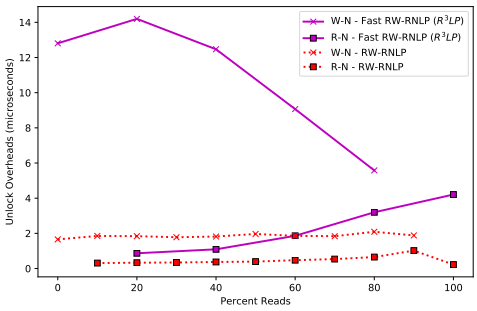



(c) Blocking.

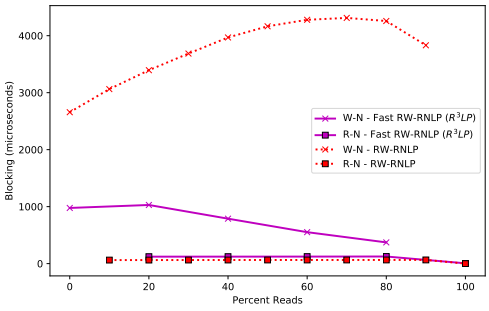
Figure 342: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

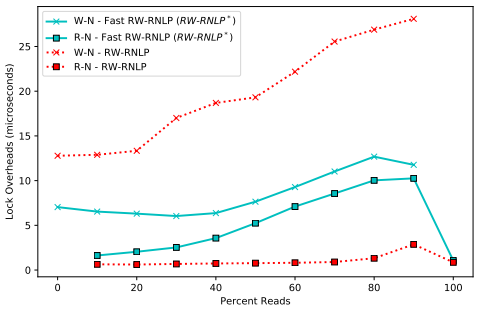



(b) Unlock overhead.

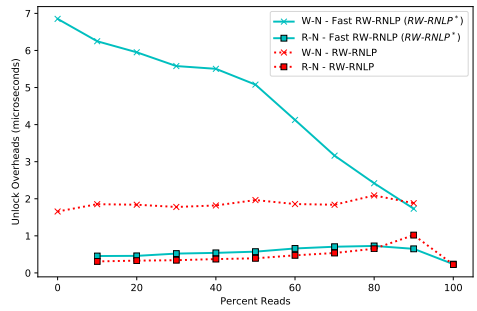



(c) Blocking.

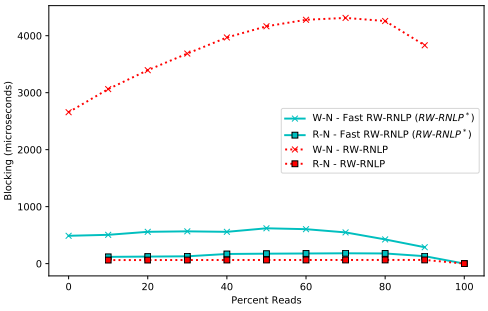
Figure 343: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

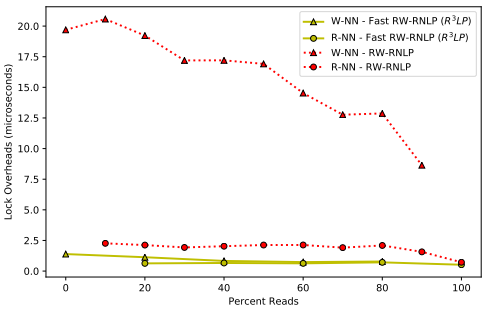



(b) Unlock overhead.

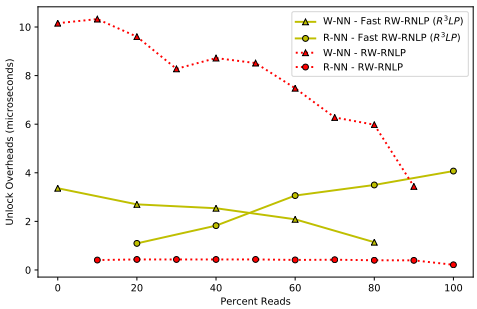



(c) Blocking.

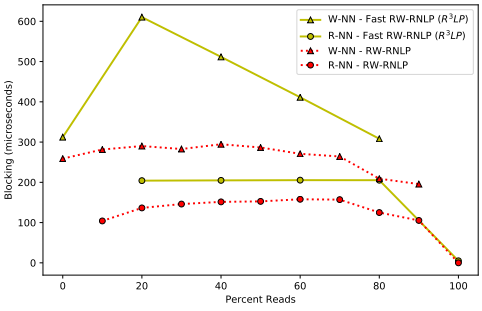
Figure 344: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

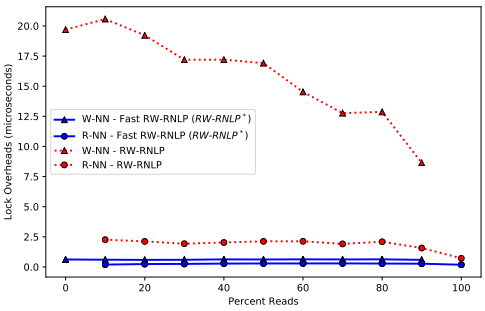



(b) Unlock overhead.

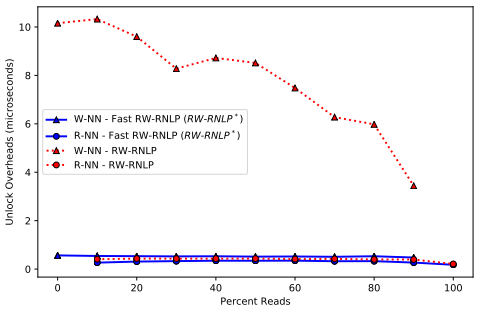



(c) Blocking.

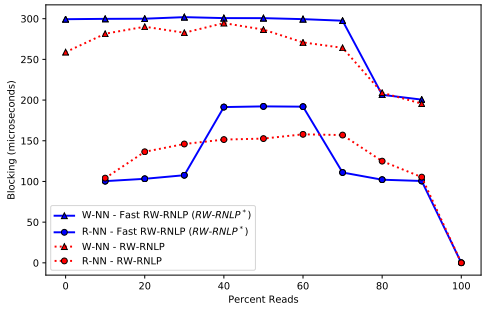
Figure 345: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

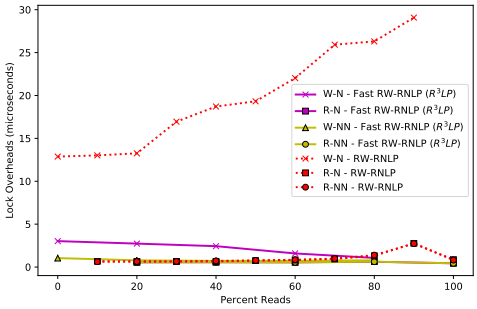



(b) Unlock overhead.

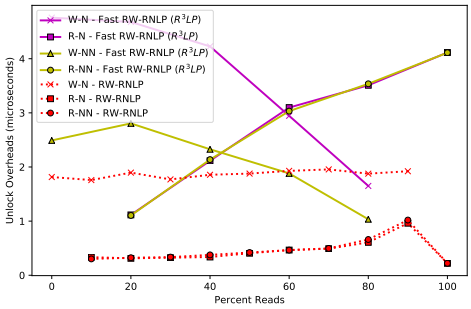



(c) Blocking.

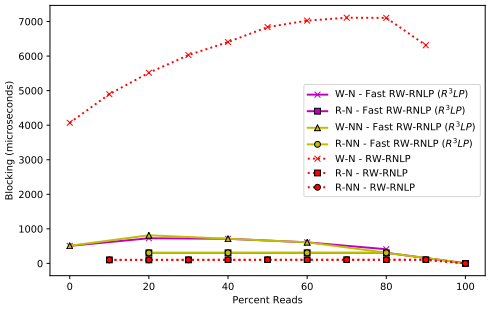
Figure 346: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

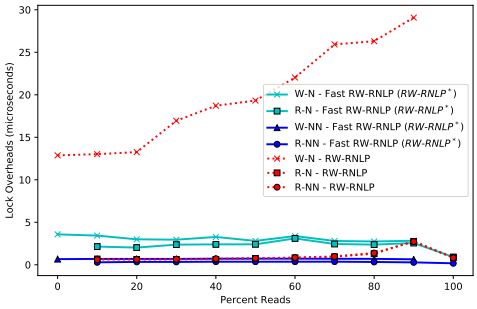



(b) Unlock overhead.

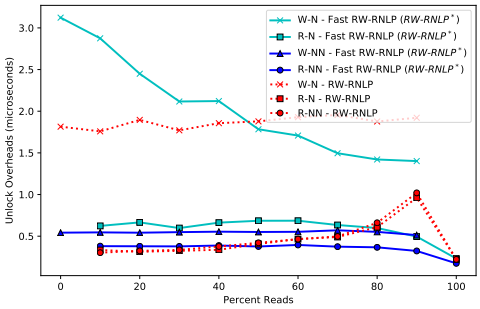



(c) Blocking.

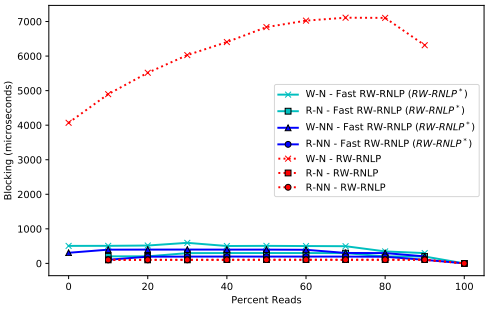
Figure 347: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

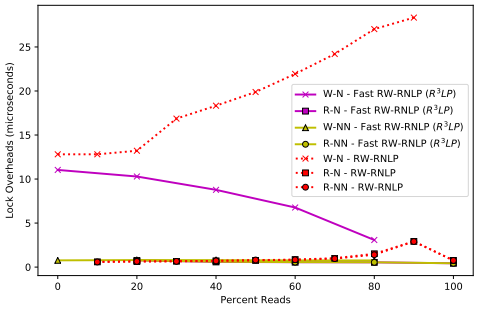



(b) Unlock overhead.

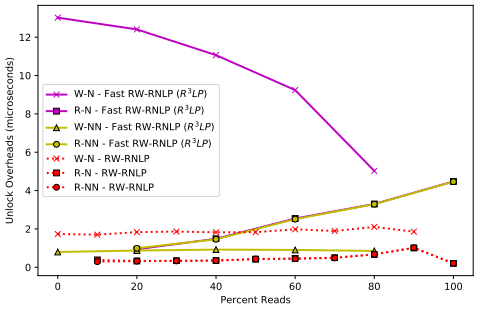



(c) Blocking.

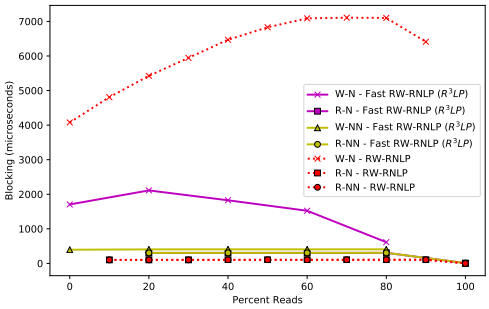
Figure 348: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

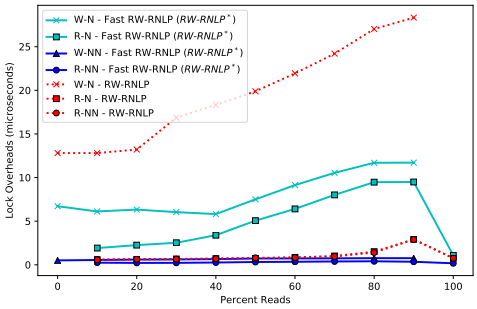



(b) Unlock overhead.

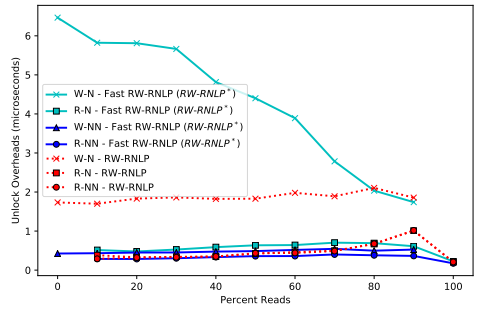



(c) Blocking.

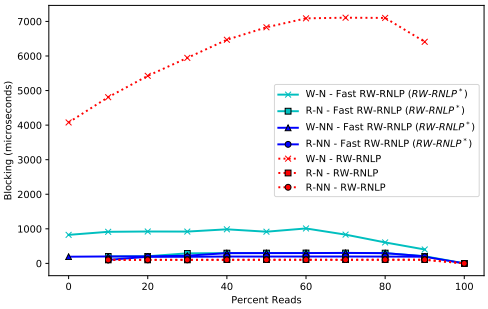
Figure 349: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

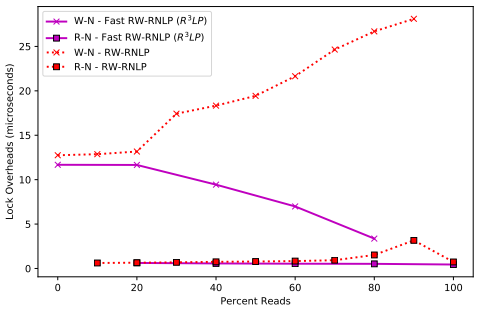



(b) Unlock overhead.

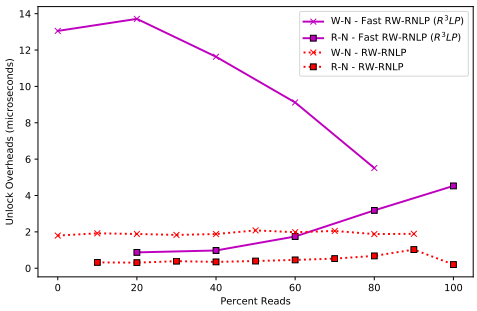



(c) Blocking.

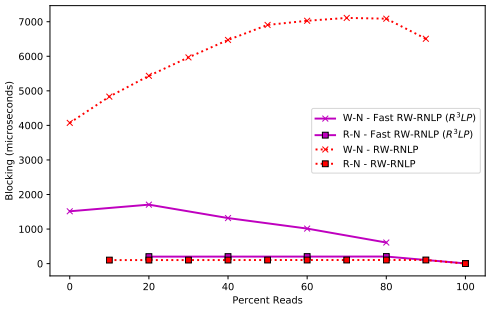
Figure 350: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

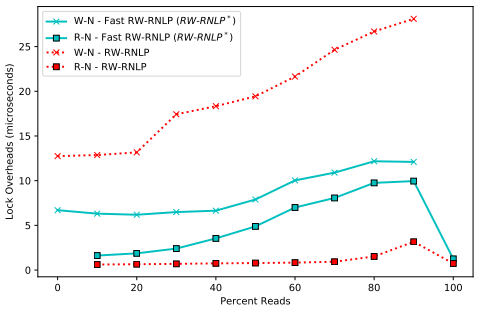



(b) Unlock overhead.

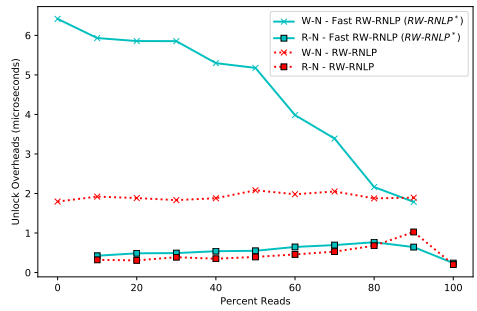



(c) Blocking.

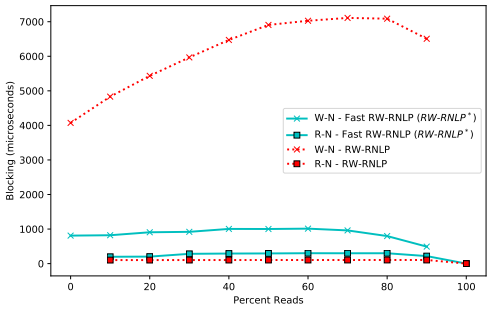
Figure 351: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

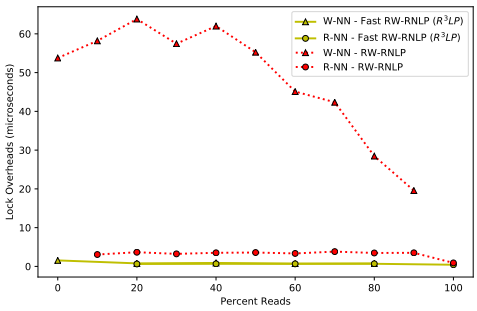



(b) Unlock overhead.

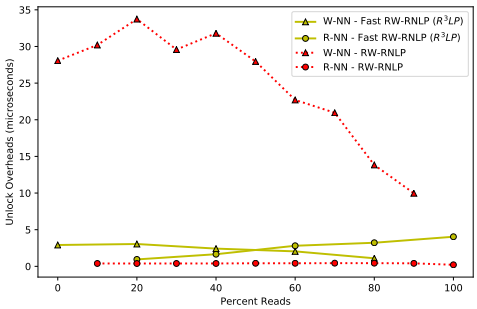



(c) Blocking.

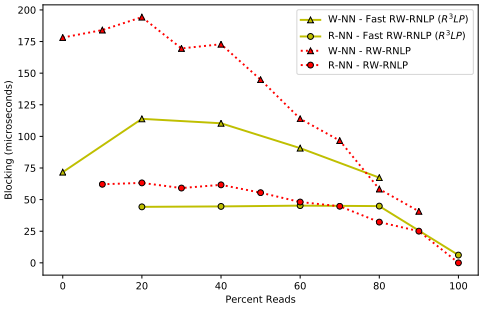
Figure 352: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 2$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

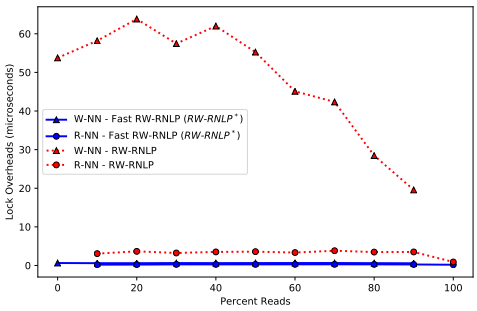



(b) Unlock overhead.

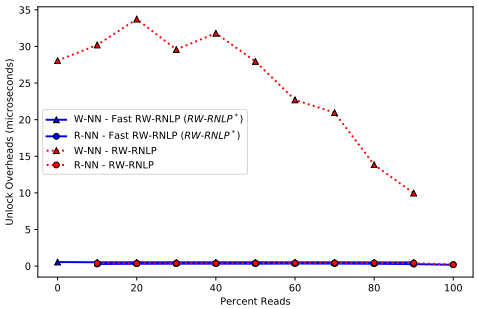



(c) Blocking.

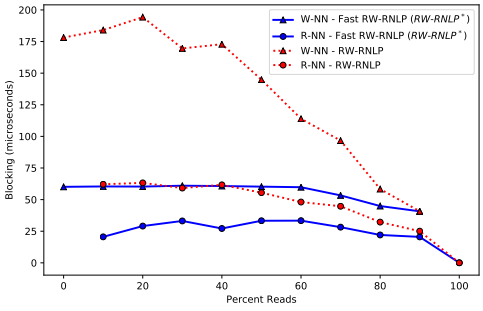
Figure 353: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

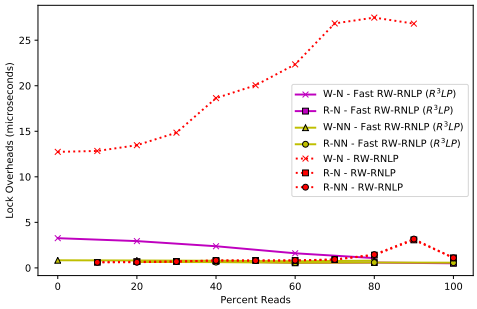



(b) Unlock overhead.

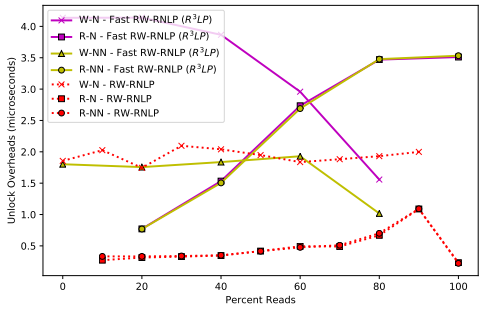



(c) Blocking.

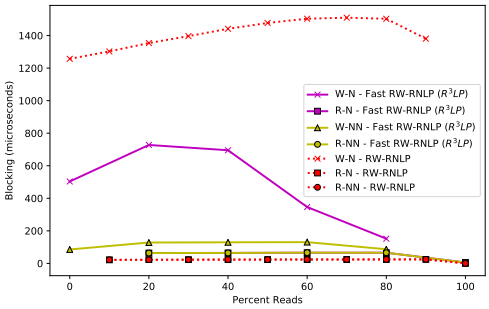
Figure 354: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

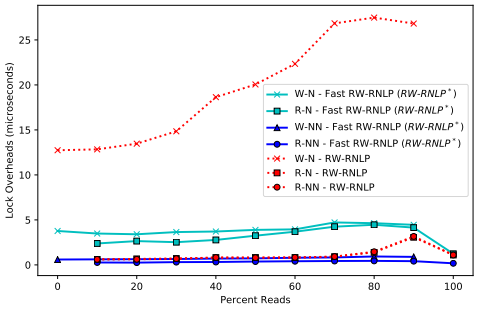



(b) Unlock overhead.

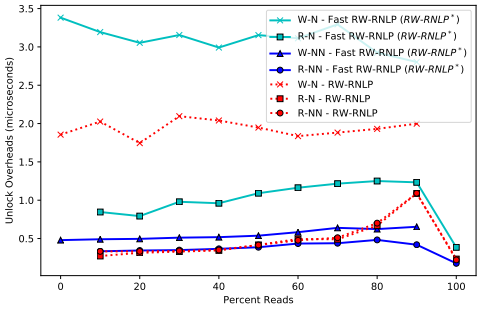



(c) Blocking.

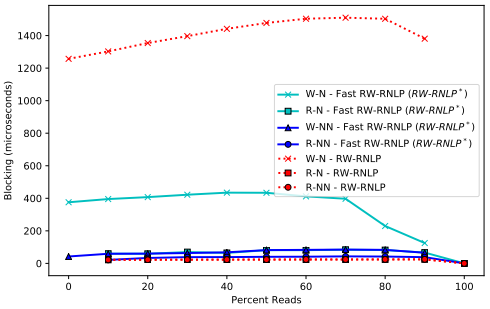
Figure 355: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

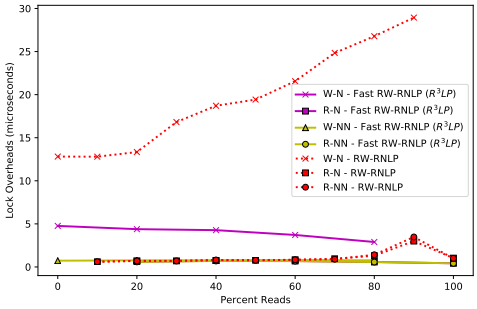



(b) Unlock overhead.

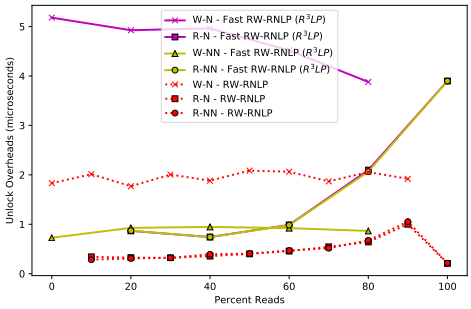



(c) Blocking.

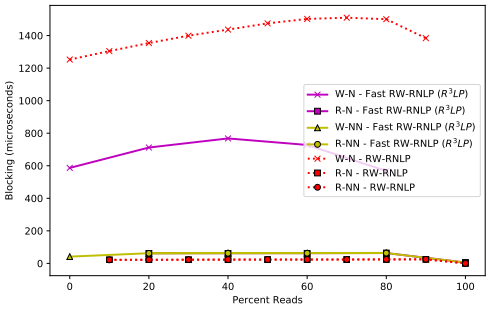
Figure 356: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

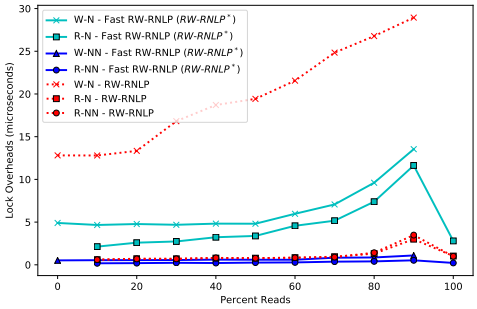



(b) Unlock overhead.

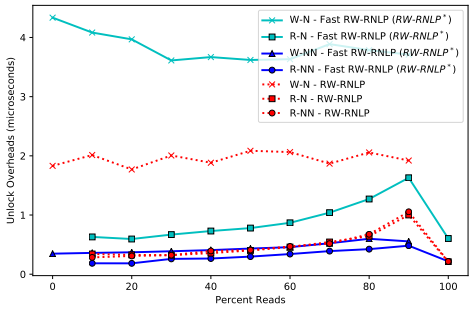



(c) Blocking.

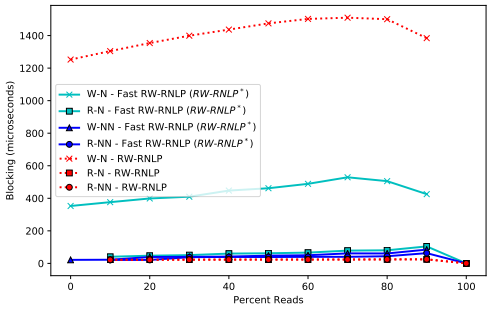
Figure 357: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

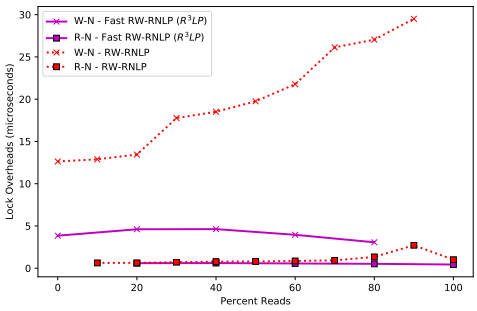



(b) Unlock overhead.

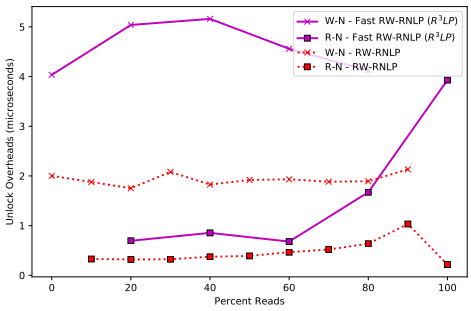



(c) Blocking.

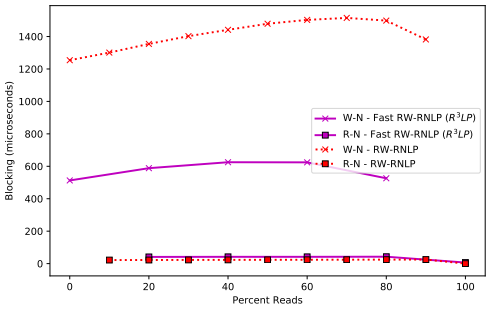
Figure 358: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

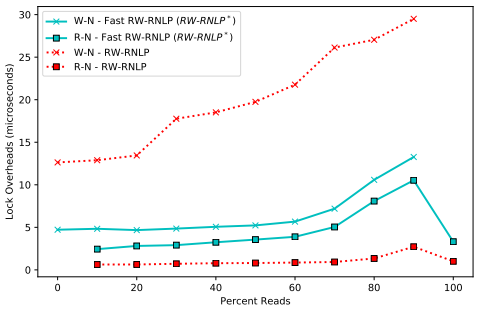



(b) Unlock overhead.

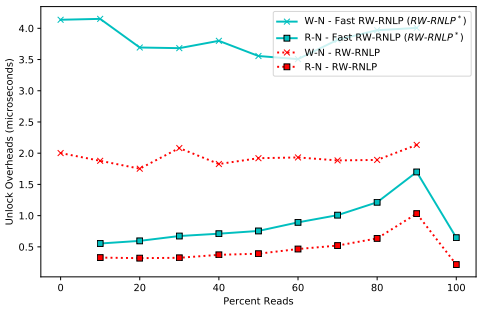



(c) Blocking.

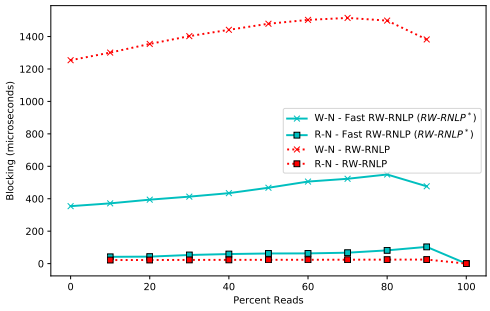
Figure 359: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

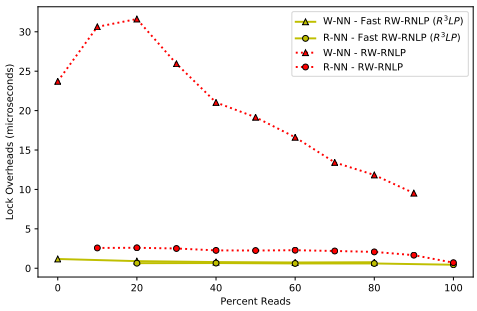



(b) Unlock overhead.

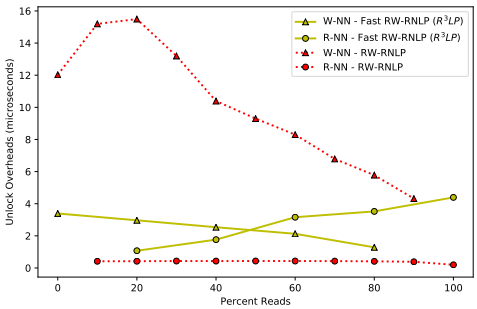



(c) Blocking.

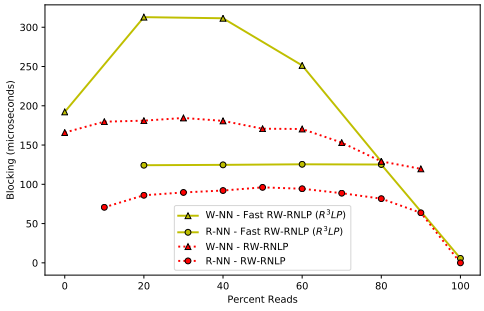
Figure 360: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

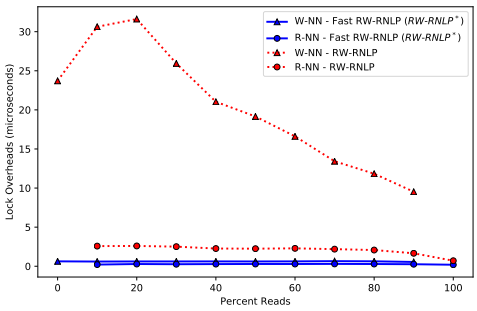



(b) Unlock overhead.

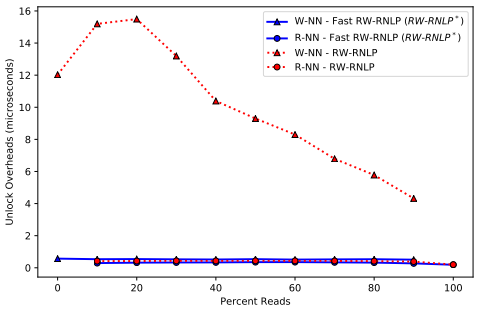



(c) Blocking.

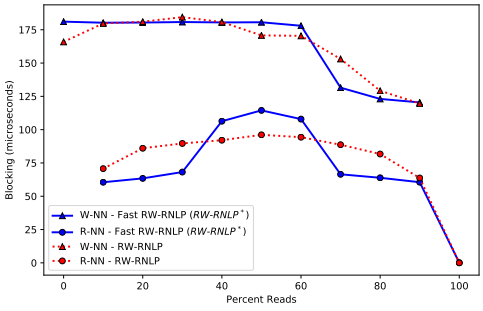
Figure 361: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

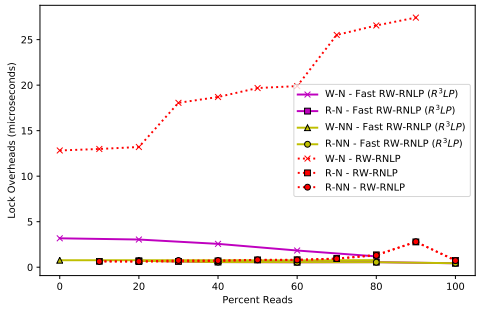



(b) Unlock overhead.

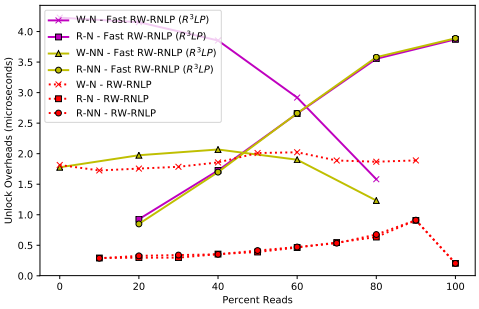



(c) Blocking.

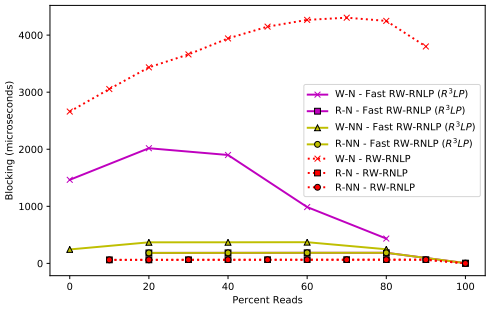
Figure 362: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

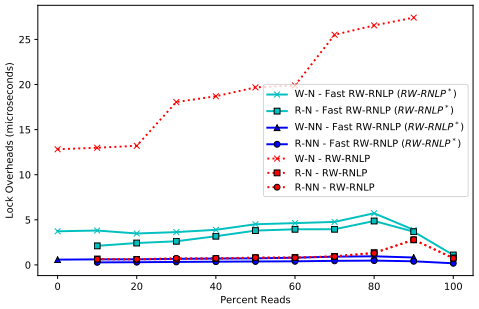



(b) Unlock overhead.

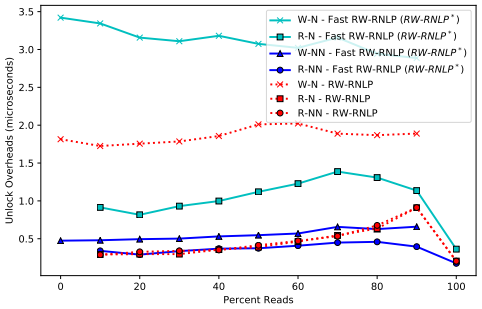



(c) Blocking.

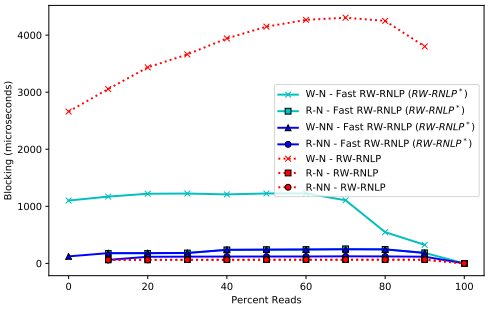
Figure 363: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

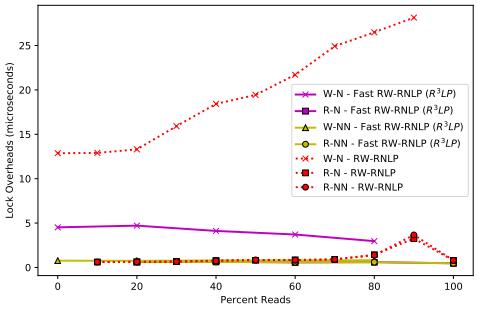



(b) Unlock overhead.

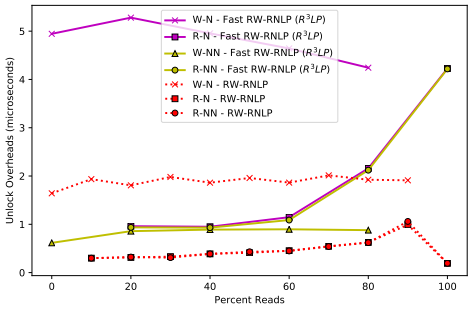



(c) Blocking.

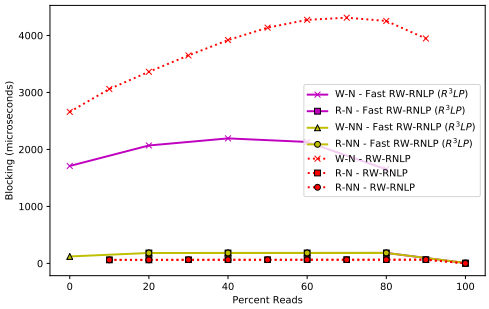
Figure 364: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

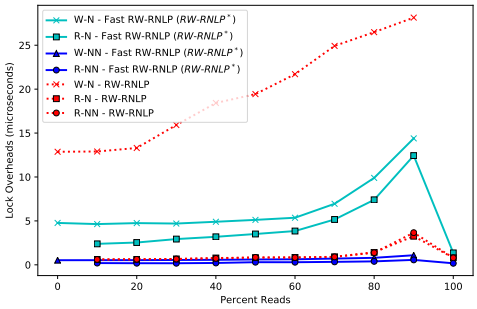



(b) Unlock overhead.

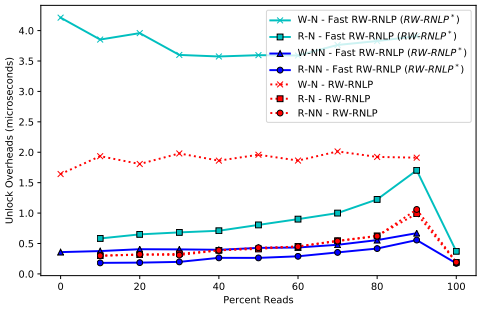



(c) Blocking.

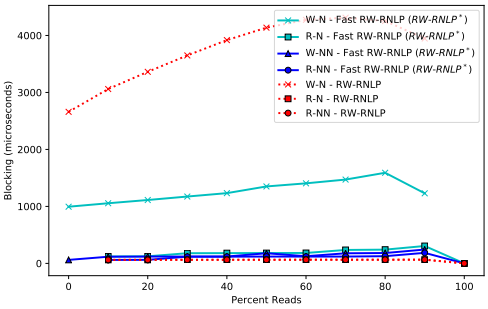
Figure 365: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

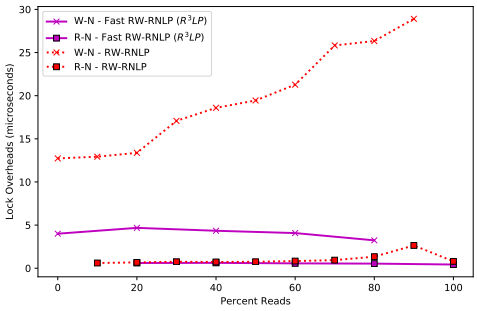



(b) Unlock overhead.

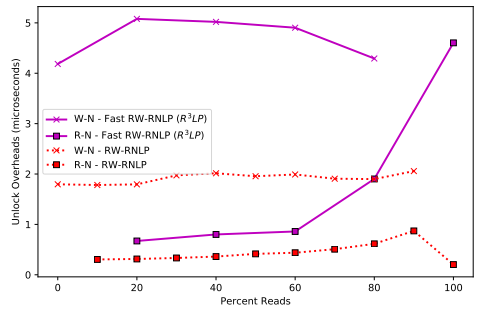



(c) Blocking.

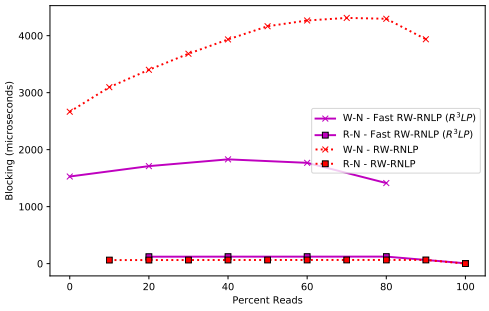
Figure 366: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

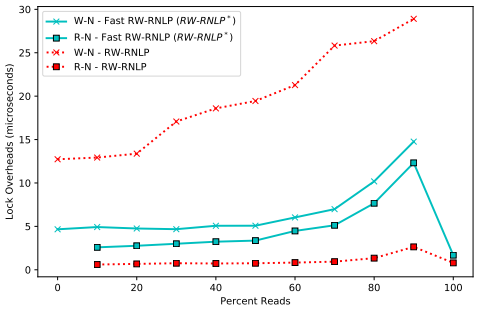



(b) Unlock overhead.

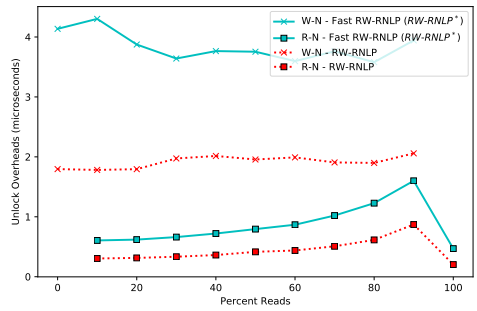



(c) Blocking.

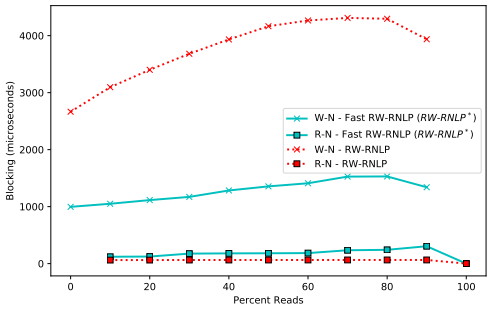
Figure 367: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

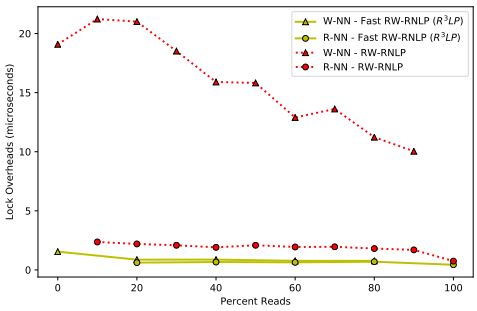



(b) Unlock overhead.

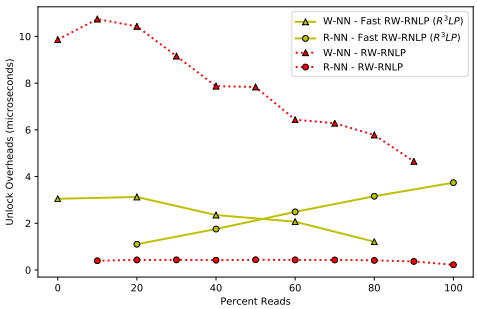



(c) Blocking.

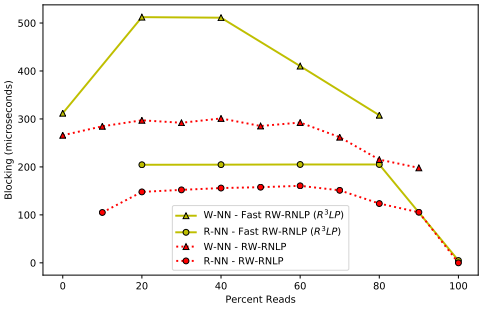
Figure 368: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

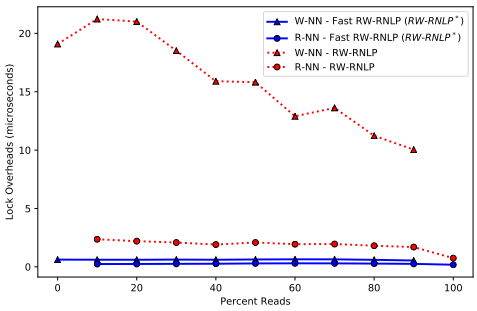



(b) Unlock overhead.

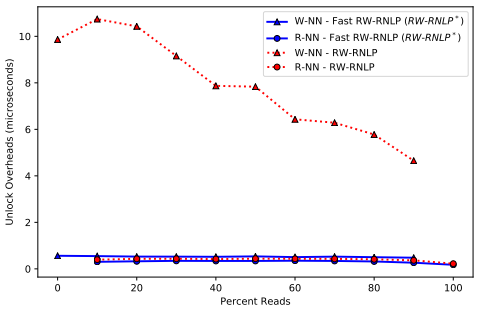



(c) Blocking.

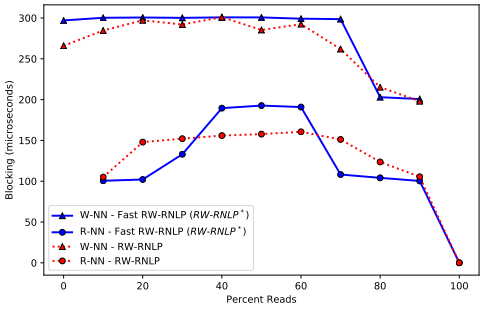
Figure 369: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

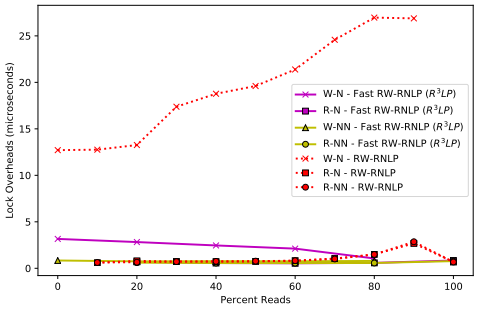



(b) Unlock overhead.

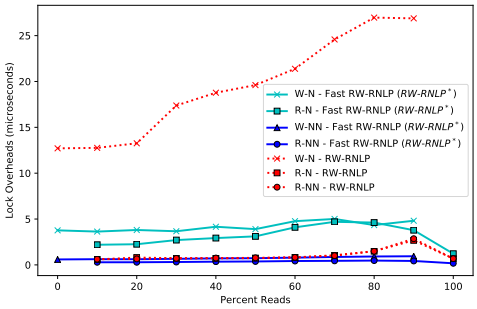



(c) Blocking.

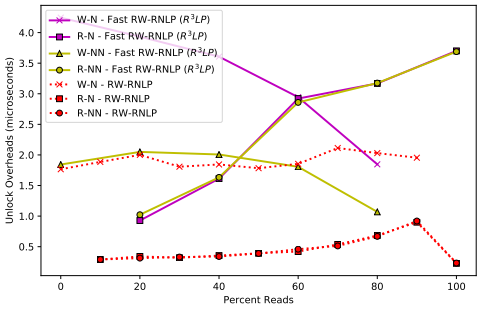
Figure 370: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.



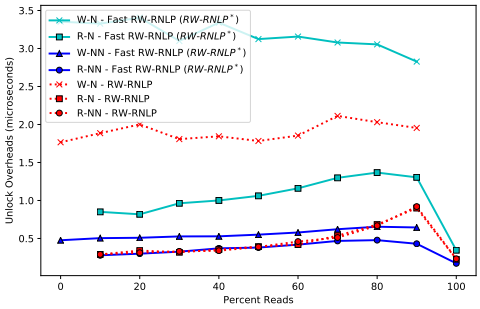

(a) Lock overhead.



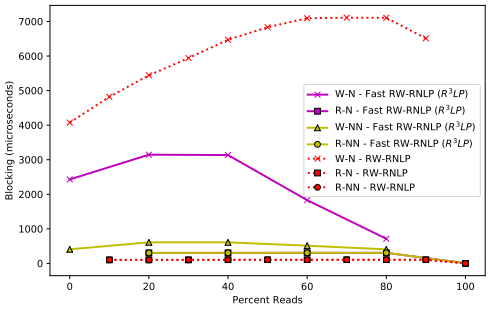

(a) Lock overhead.



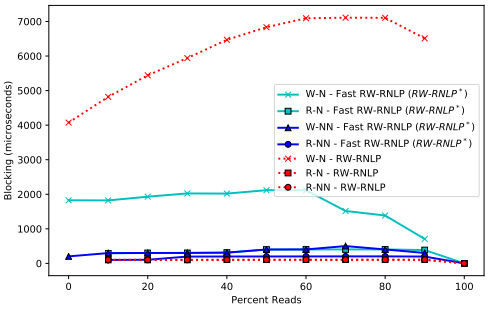

(b) Unlock overhead.




(b) Unlock overhead.



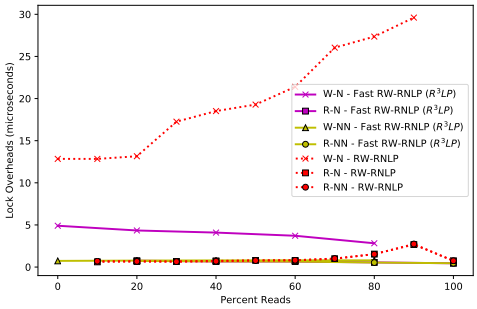

(c) Blocking.



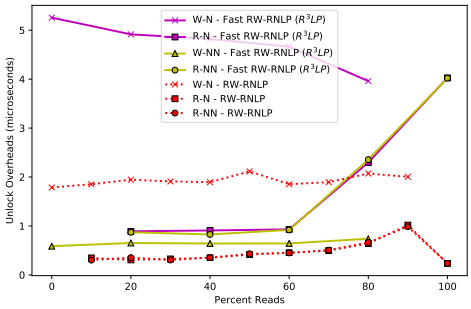

(c) Blocking.

Figure 371: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.

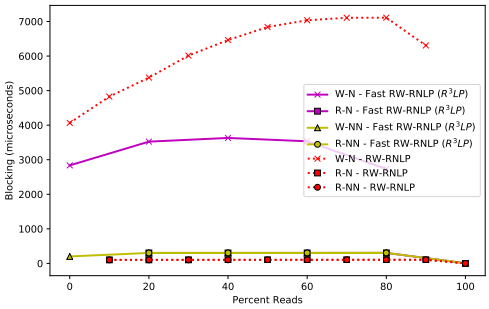
Figure 372: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

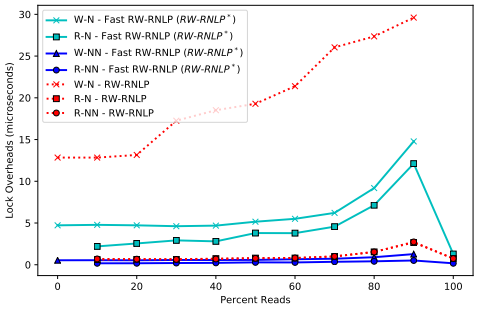



(b) Unlock overhead.

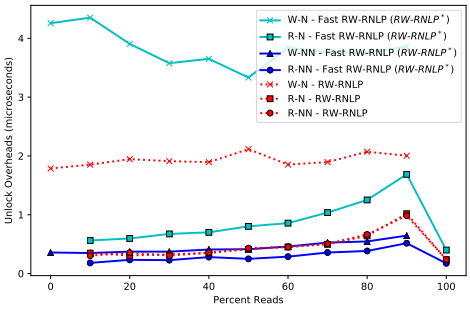



(c) Blocking.

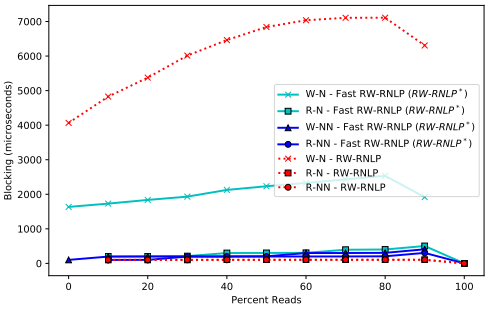
Figure 373: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

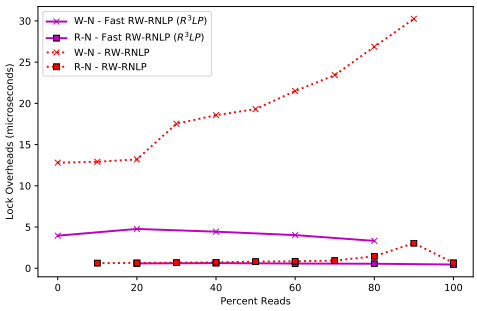



(b) Unlock overhead.

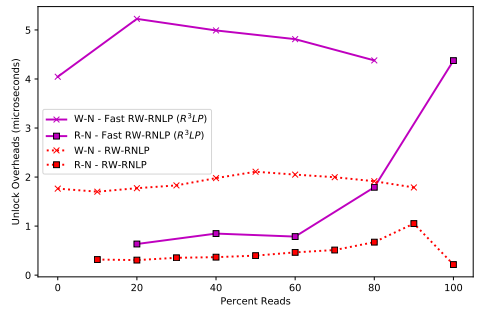



(c) Blocking.

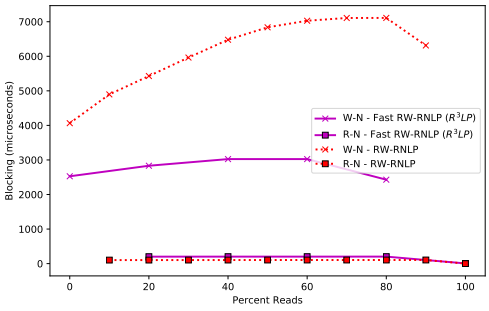
Figure 374: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

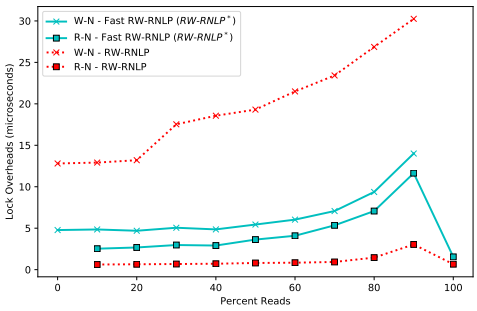



(b) Unlock overhead.

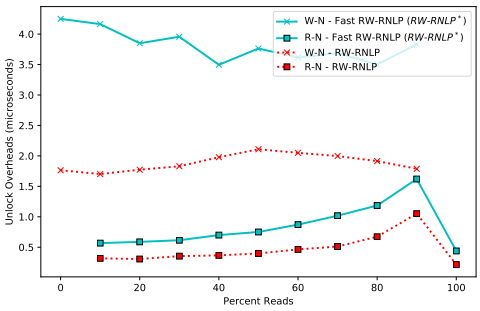



(c) Blocking.

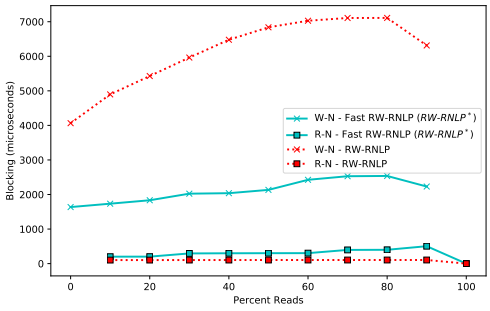
Figure 375: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

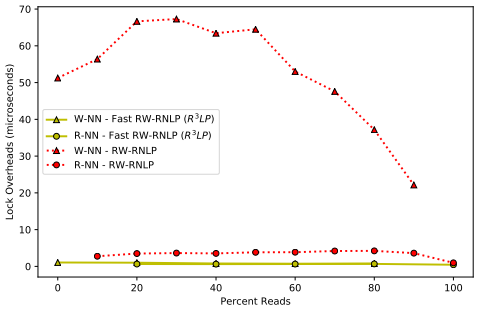



(b) Unlock overhead.

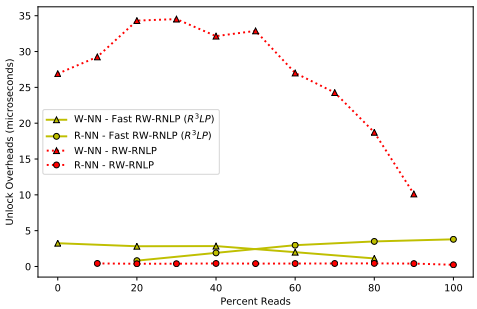



(c) Blocking.

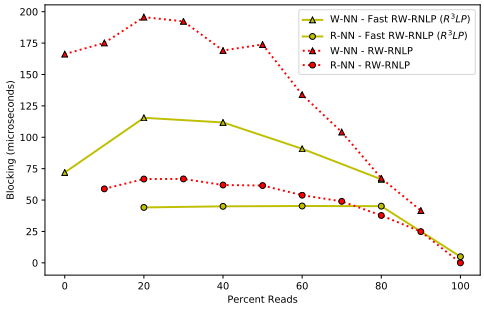
Figure 376: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 4$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

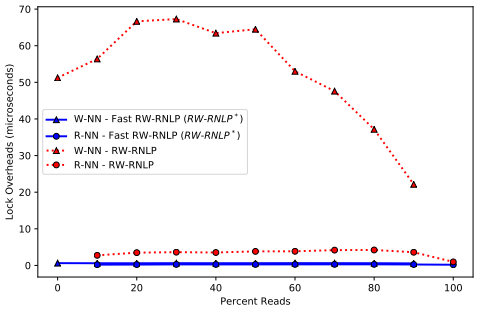



(b) Unlock overhead.

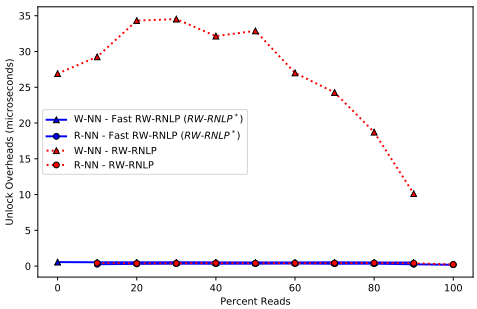



(c) Blocking.

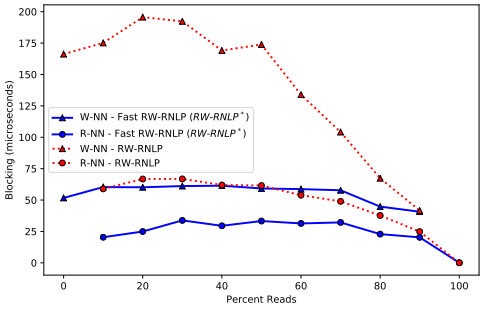
Figure 377: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

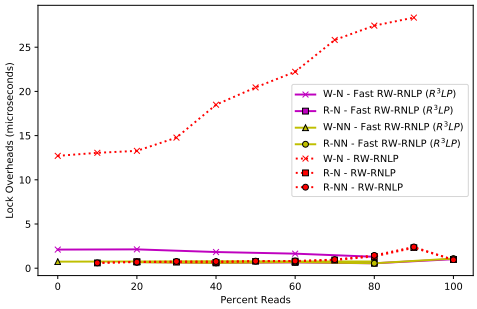



(b) Unlock overhead.

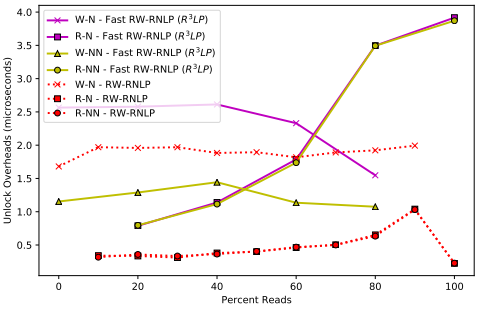



(c) Blocking.

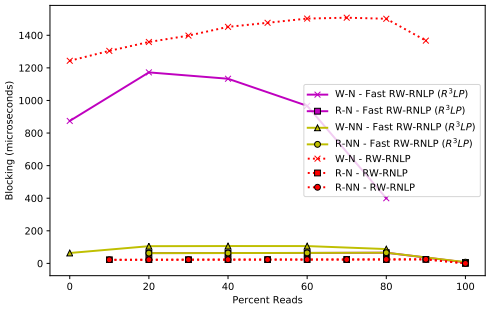
Figure 378: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

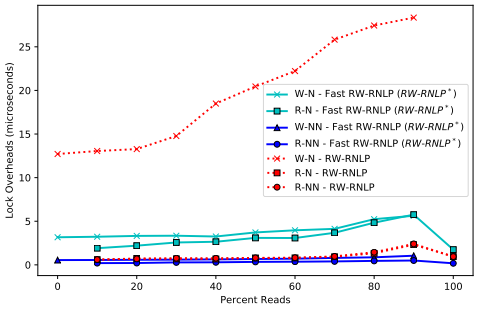



(b) Unlock overhead.

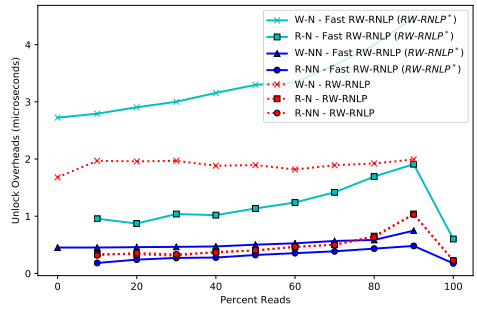



(c) Blocking.

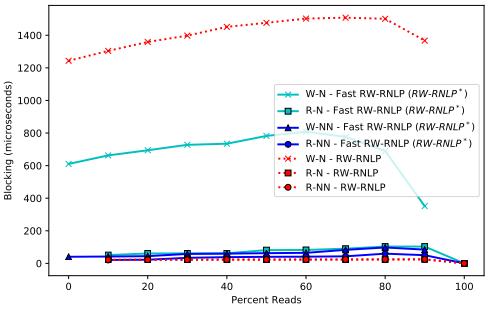
Figure 379: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

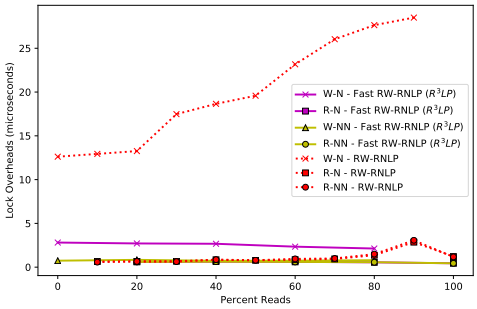



(b) Unlock overhead.

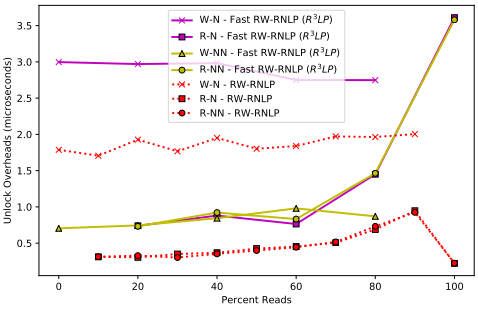



(c) Blocking.

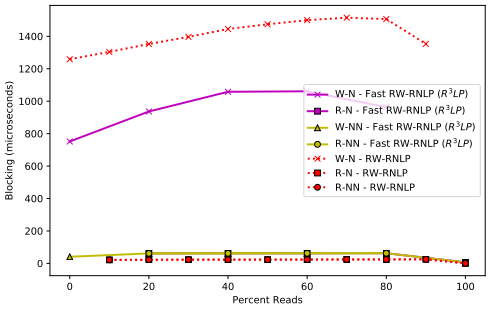
Figure 380: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

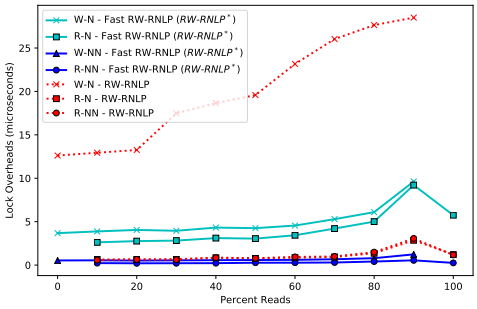



(b) Unlock overhead.

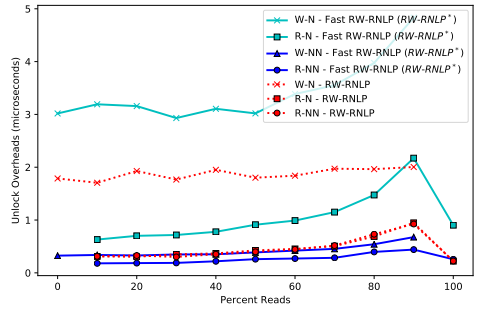



(c) Blocking.

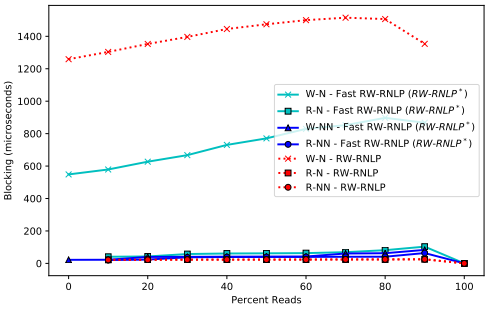
Figure 381: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

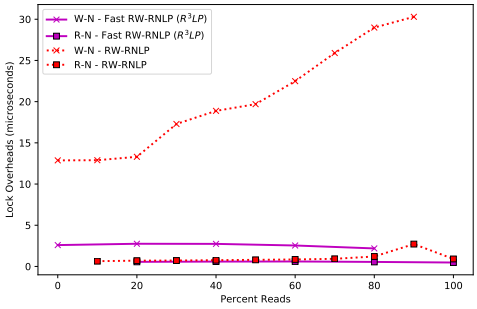



(b) Unlock overhead.

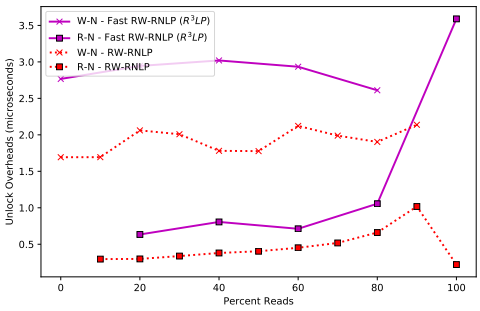



(c) Blocking.

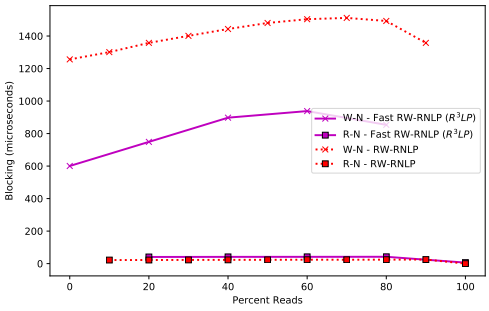
Figure 382: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

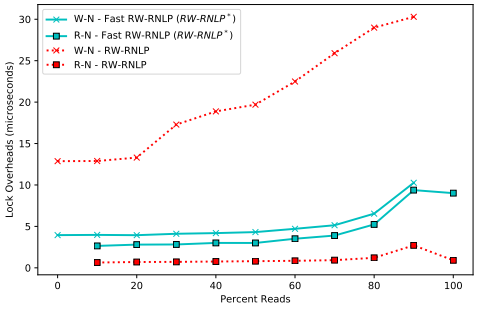



(b) Unlock overhead.

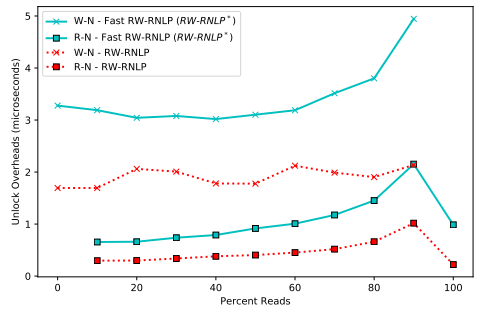



(c) Blocking.

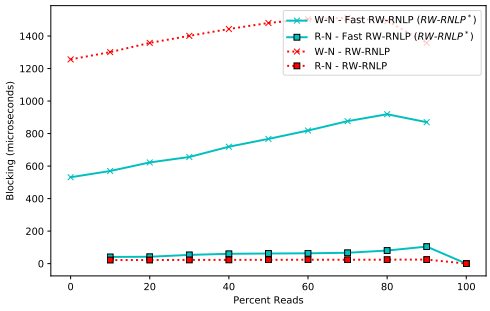
Figure 383: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

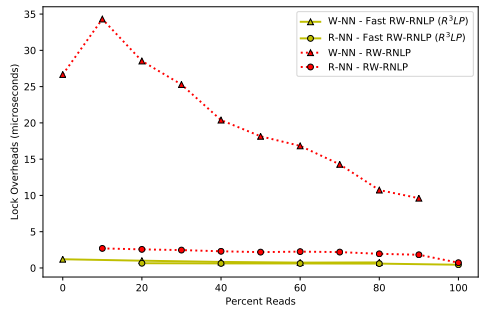



(b) Unlock overhead.

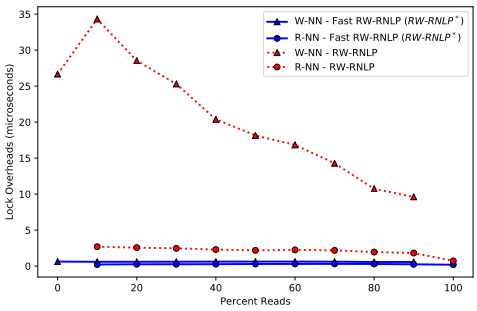



(c) Blocking.

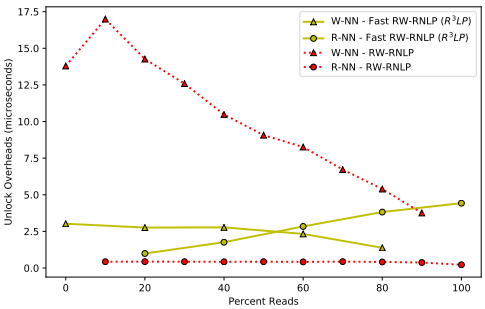
Figure 384: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.



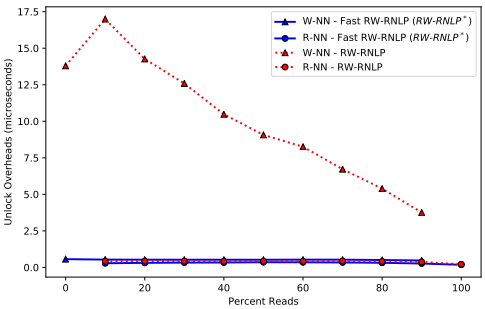

(a) Lock overhead.



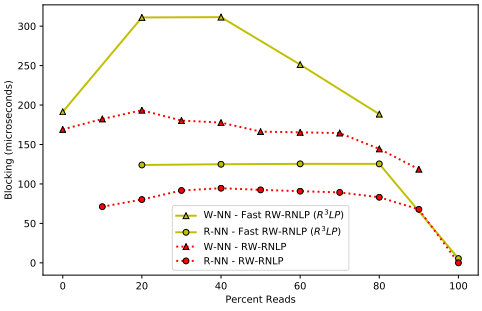

(a) Lock overhead.



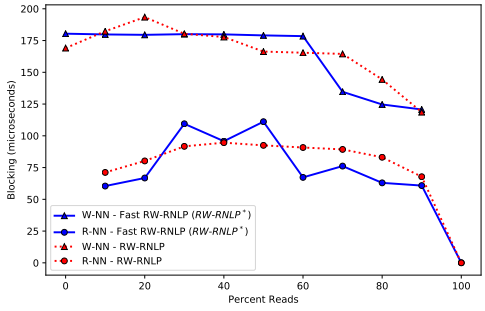

(b) Unlock overhead.




(b) Unlock overhead.



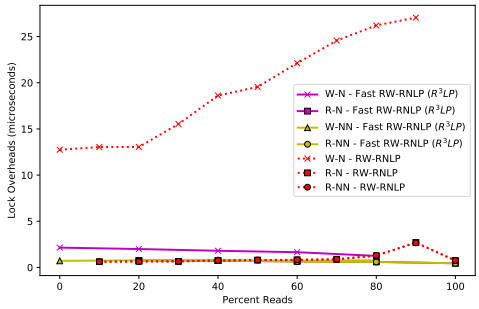

(c) Blocking.



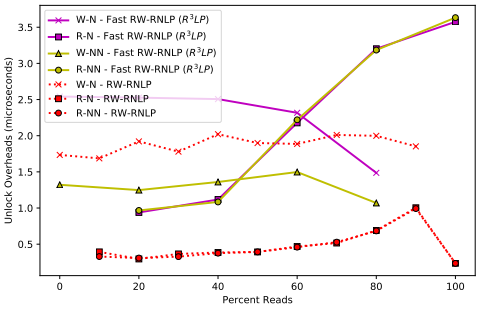

(c) Blocking.

Figure 385: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.

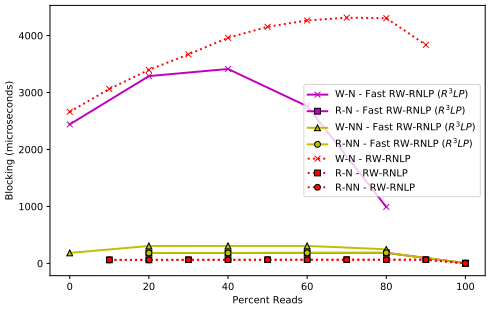
Figure 386: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

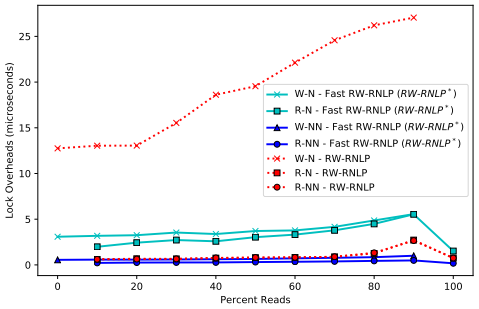



(b) Unlock overhead.

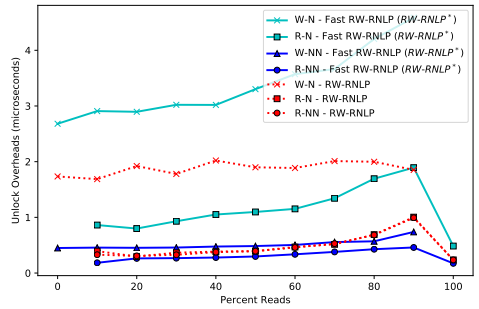



(c) Blocking.

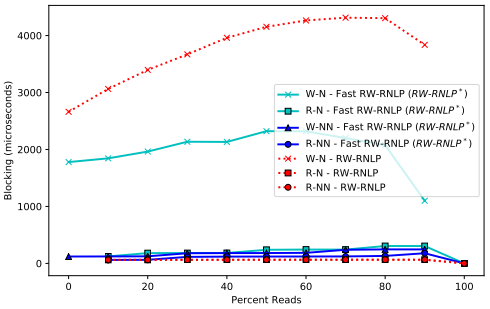
Figure 387: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

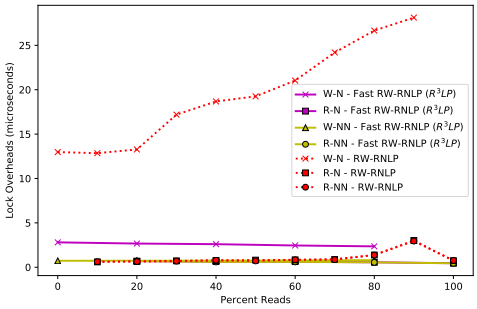



(b) Unlock overhead.

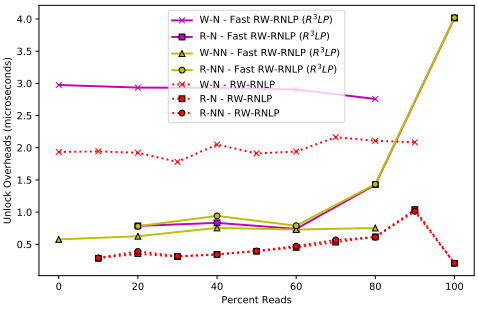



(c) Blocking.

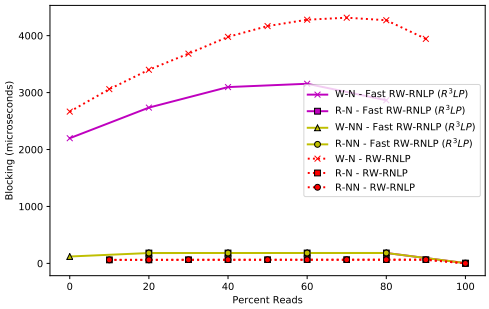
Figure 388: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

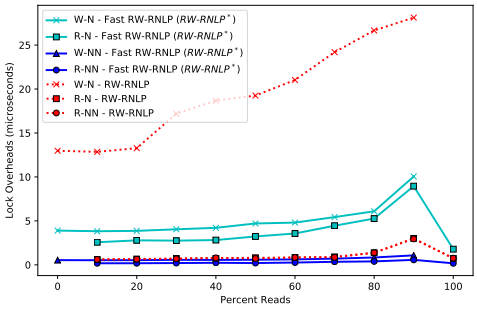



(b) Unlock overhead.

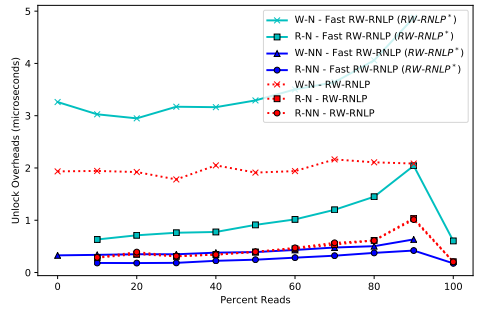



(c) Blocking.

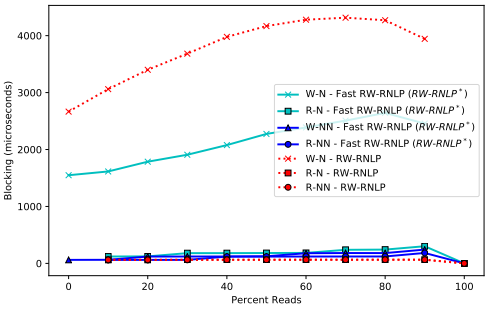
Figure 389: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

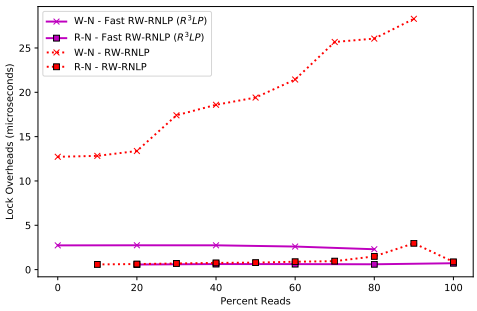



(b) Unlock overhead.

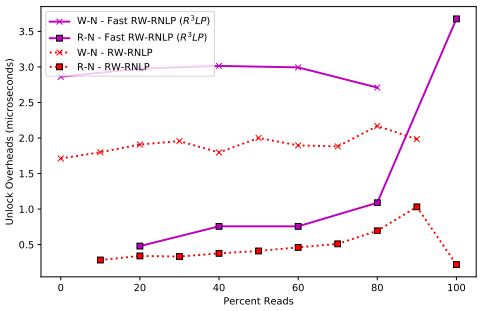



(c) Blocking.

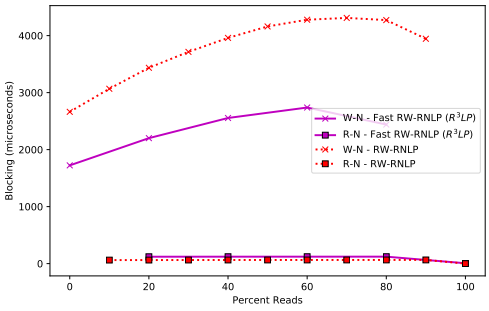
Figure 390: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

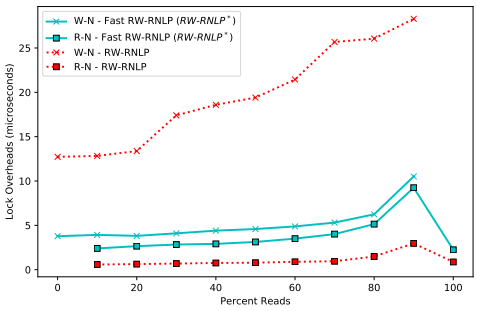



(b) Unlock overhead.

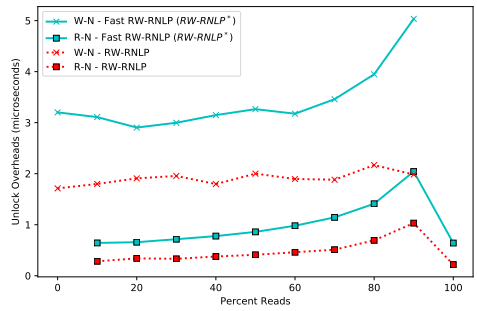



(c) Blocking.

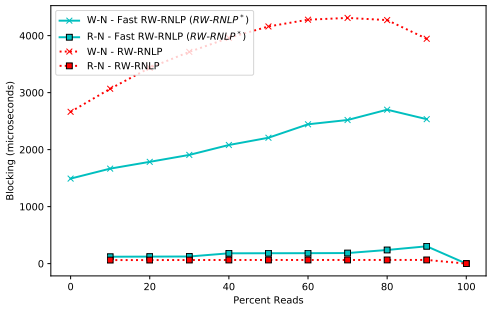
Figure 391: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

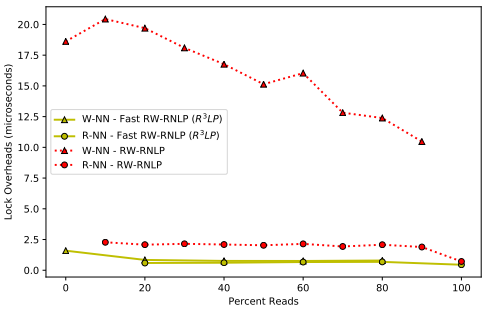



(b) Unlock overhead.

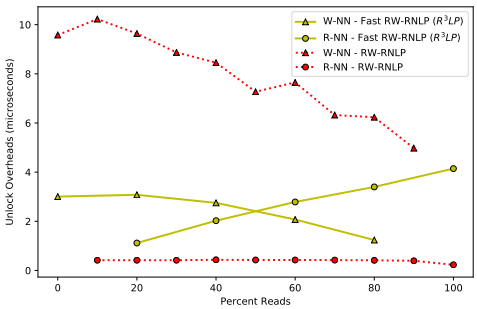



(c) Blocking.

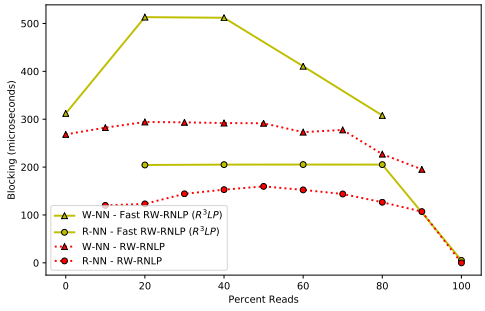
Figure 392: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

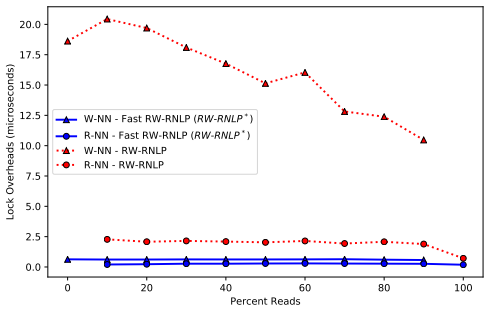



(b) Unlock overhead.

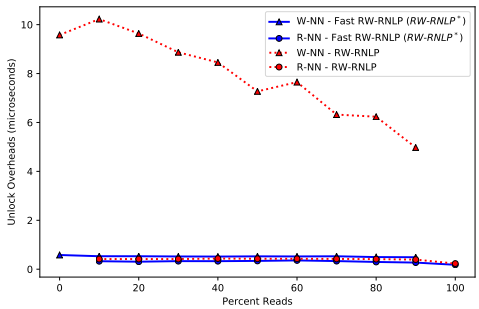



(c) Blocking.

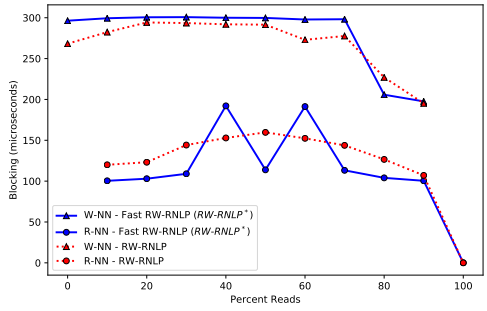
Figure 393: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

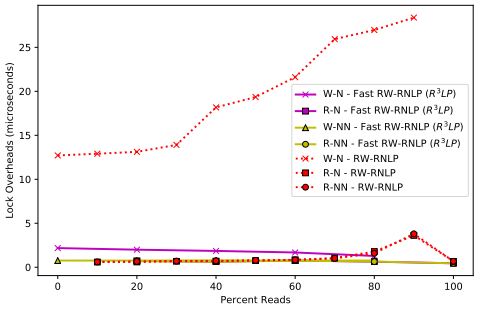



(b) Unlock overhead.

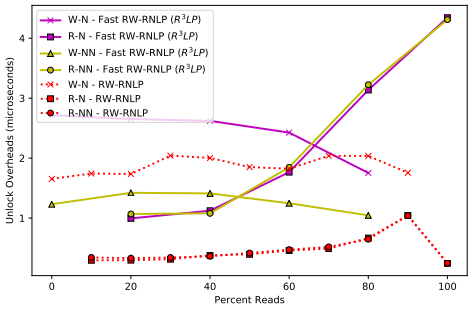



(c) Blocking.

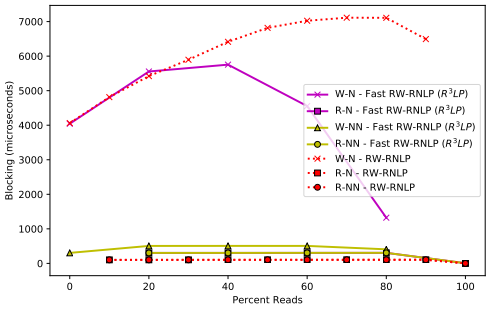
Figure 394: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 1$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.




(a) Lock overhead.

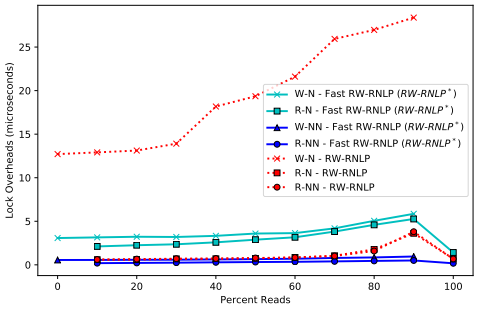



(b) Unlock overhead.

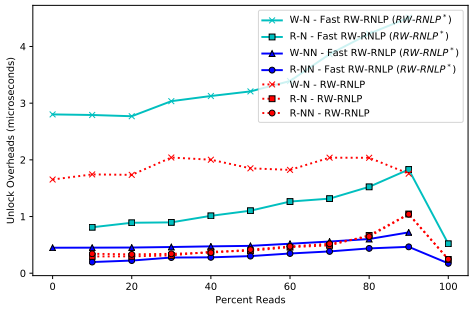



(c) Blocking.

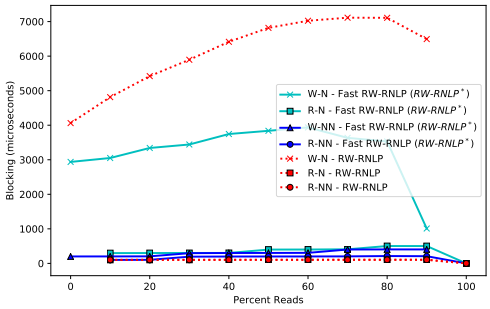
Figure 395: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

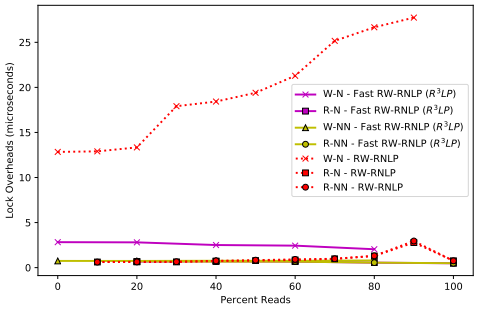



(b) Unlock overhead.

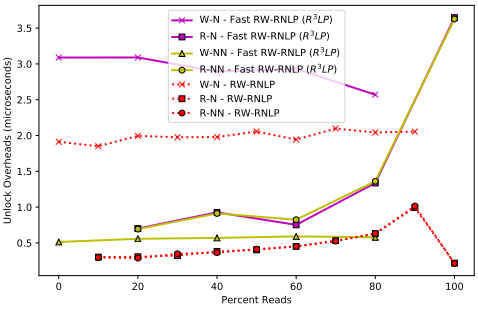



(c) Blocking.

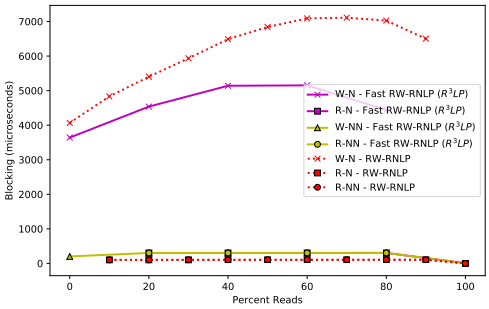
Figure 396: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

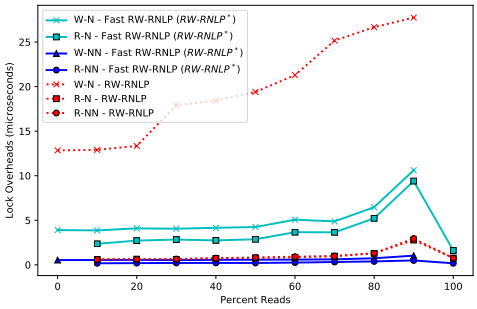



(b) Unlock overhead.

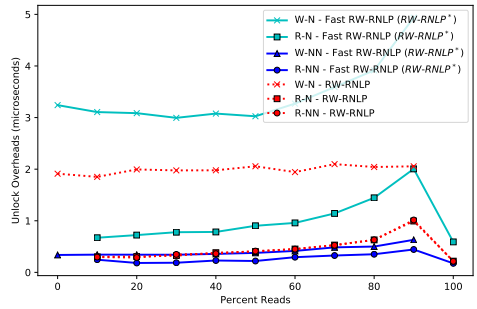



(c) Blocking.

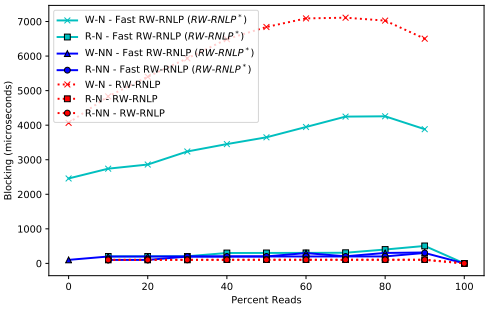
Figure 397: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

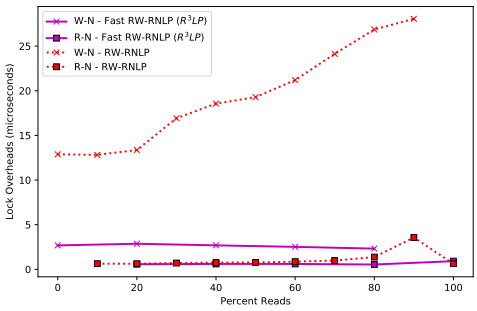



(b) Unlock overhead.

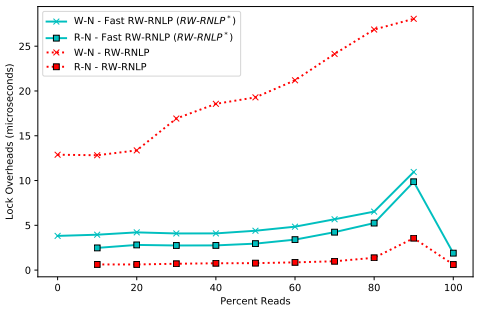



(c) Blocking.

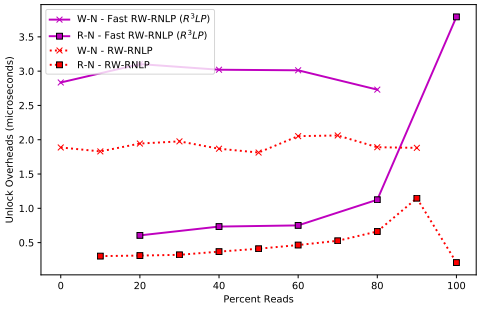
Figure 398: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.



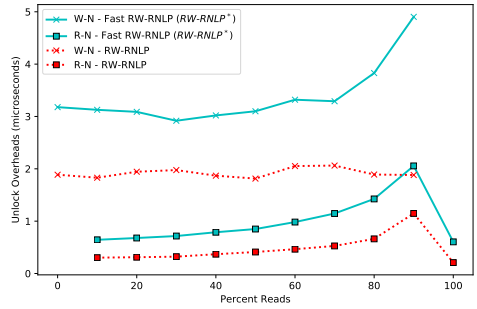

(a) Lock overhead.



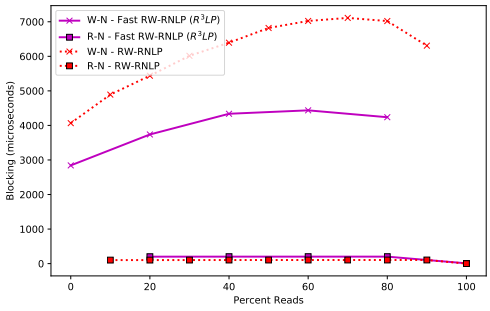

(a) Lock overhead.



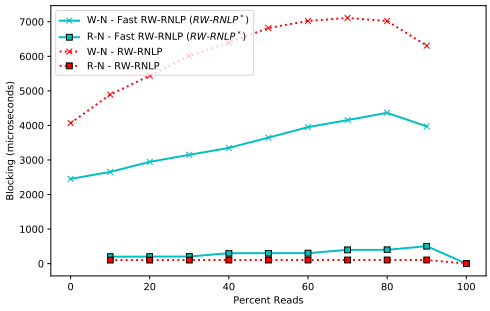

(b) Unlock overhead.




(b) Unlock overhead.



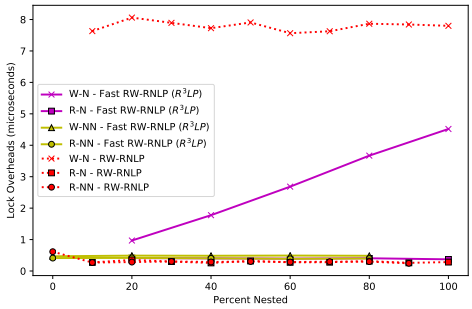

(c) Blocking.



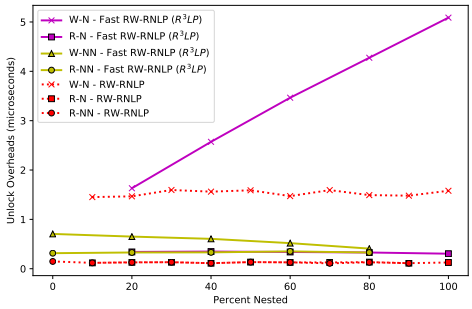

(c) Blocking.

Figure 399: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.

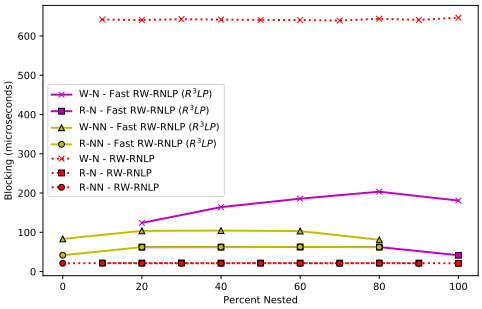
Figure 400: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i| = 6$. Each request was randomly chosen to be a read (as opposed to a write) with probability as shown and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

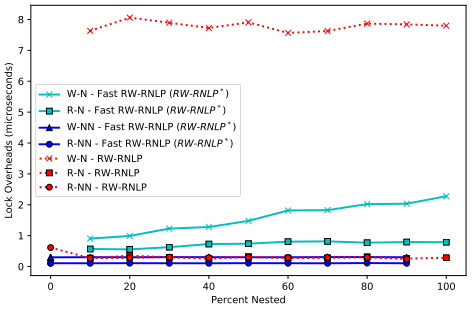



(b) Unlock overhead.

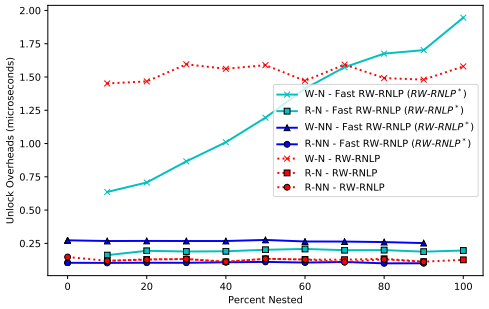



(c) Blocking.

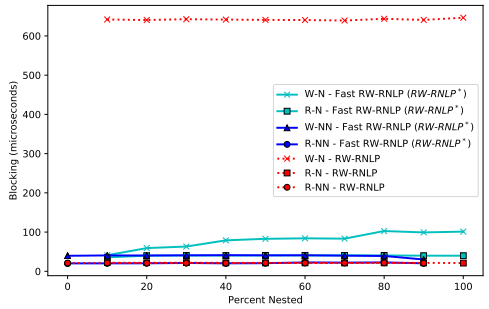
Figure 401: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

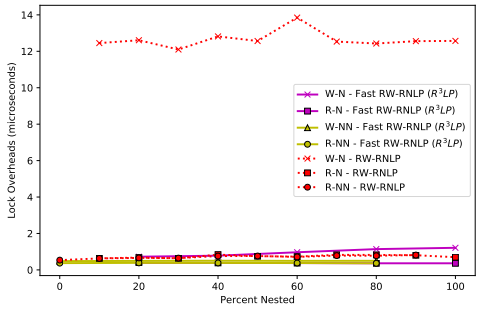



(b) Unlock overhead.

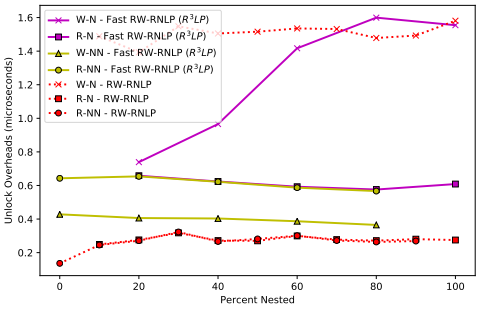



(c) Blocking.

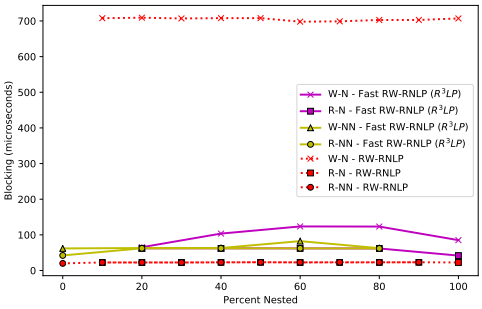
Figure 402: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

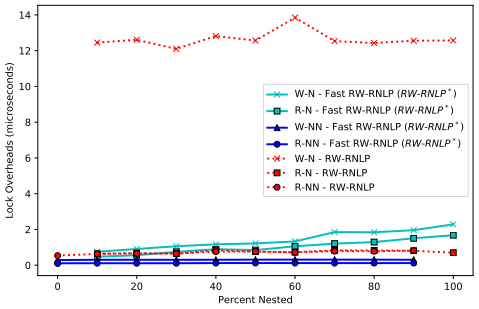



(b) Unlock overhead.

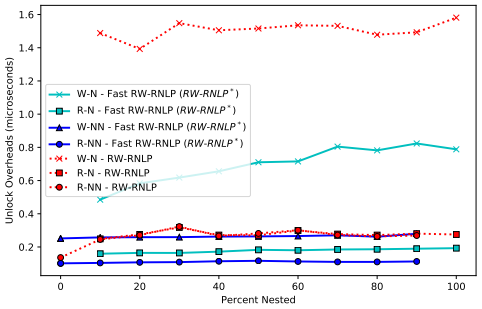



(c) Blocking.

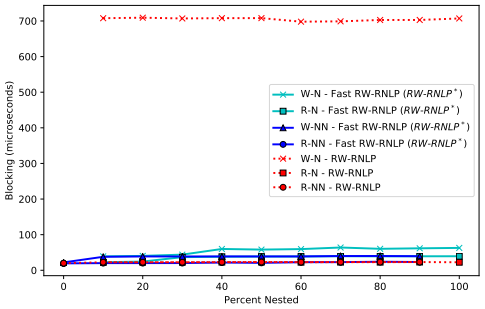
Figure 403: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

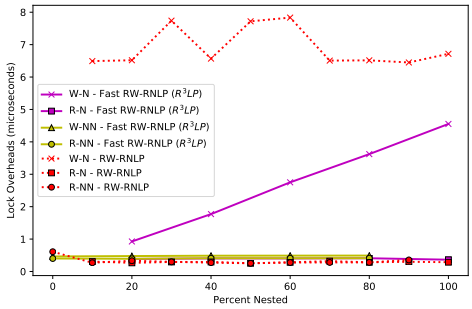



(b) Unlock overhead.

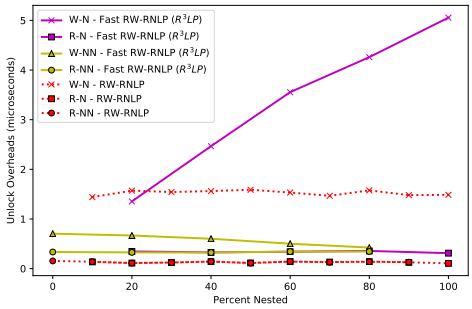



(c) Blocking.

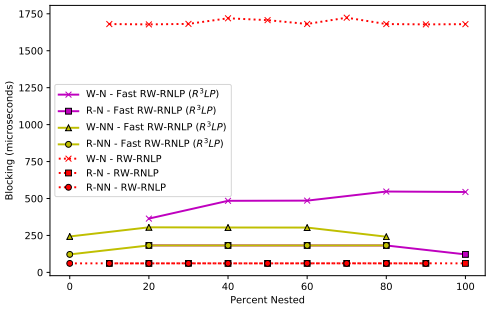
Figure 404: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

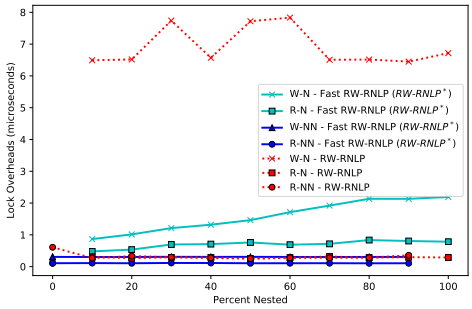



(b) Unlock overhead.

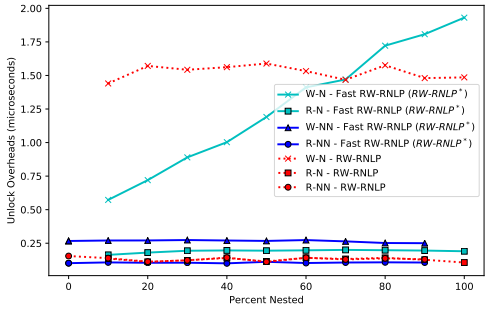



(c) Blocking.

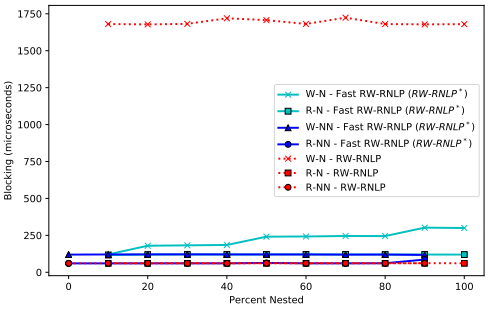
Figure 405: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

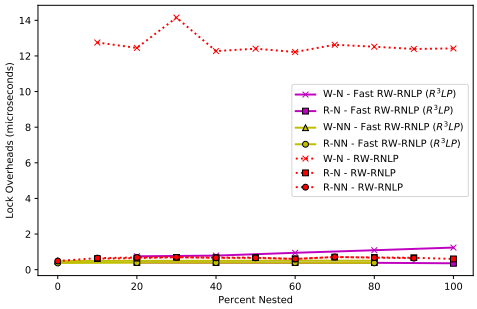



(b) Unlock overhead.

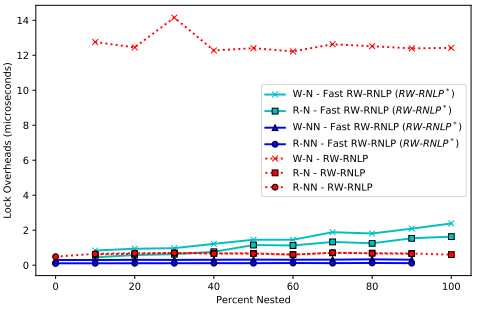



(c) Blocking.

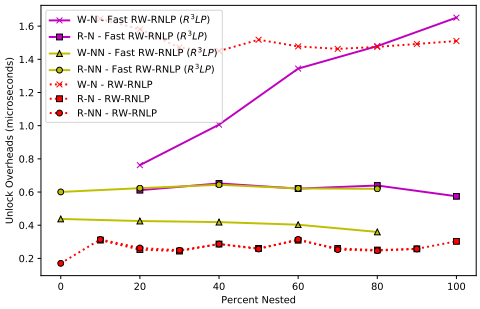
Figure 406: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.



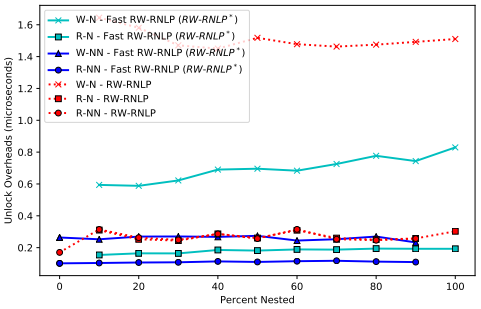

(a) Lock overhead.



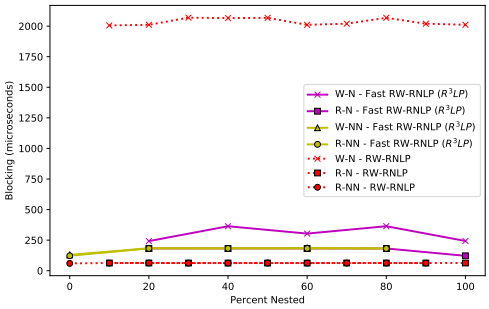

(a) Lock overhead.



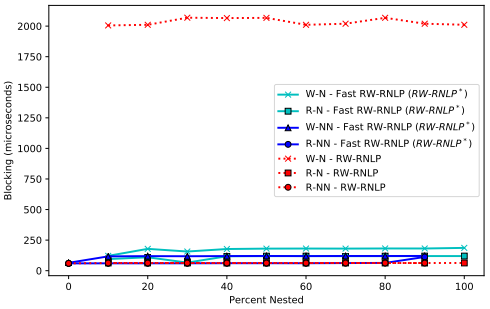

(b) Unlock overhead.




(b) Unlock overhead.



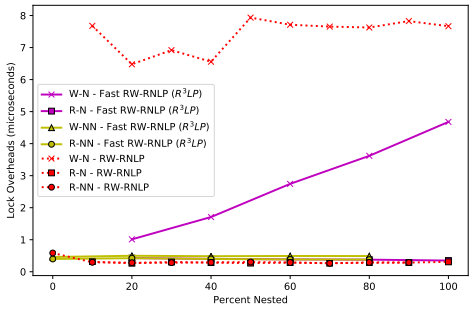

(c) Blocking.



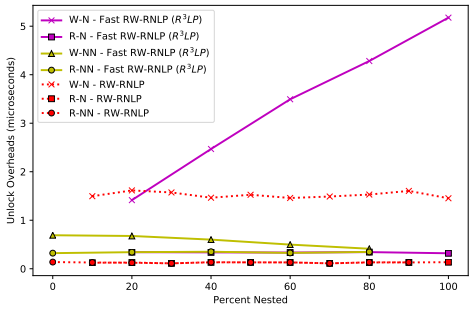

(c) Blocking.

Figure 407: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.

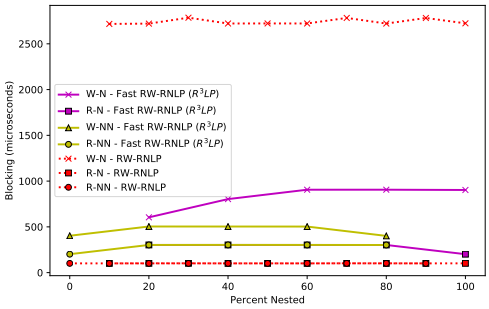
Figure 408: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

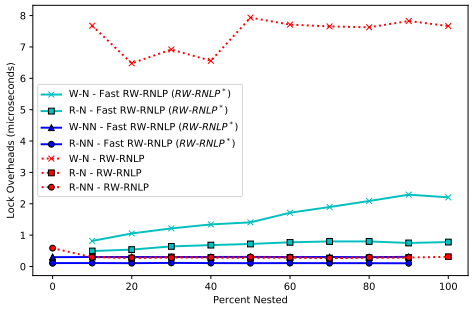



(b) Unlock overhead.

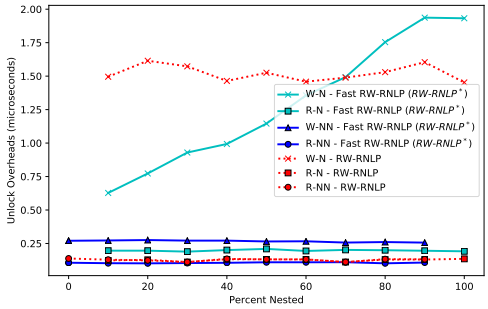



(c) Blocking.

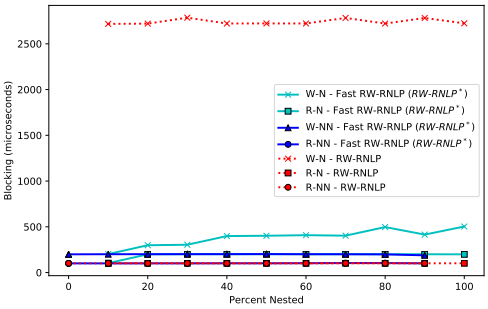
Figure 409: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

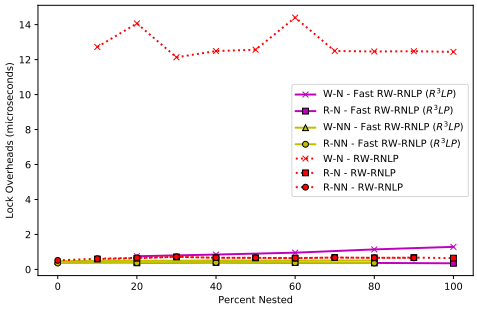



(b) Unlock overhead.

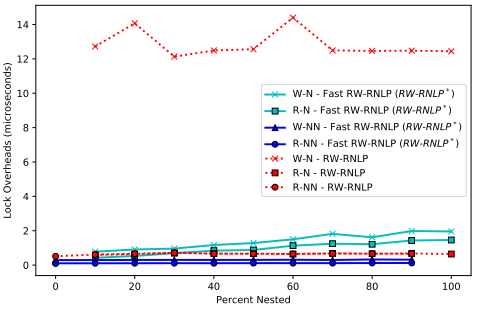



(c) Blocking.

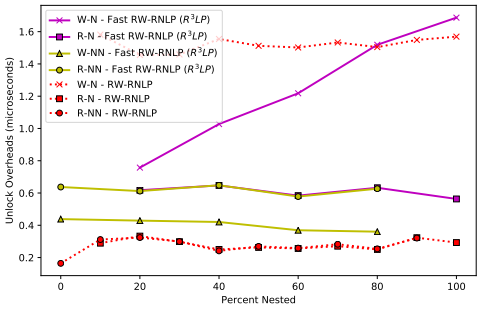
Figure 410: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.



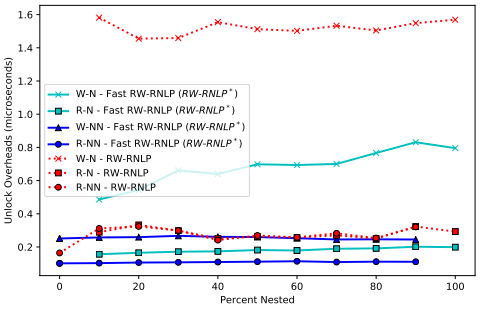

(a) Lock overhead.



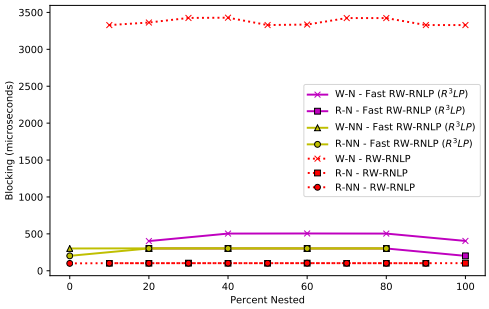

(a) Lock overhead.



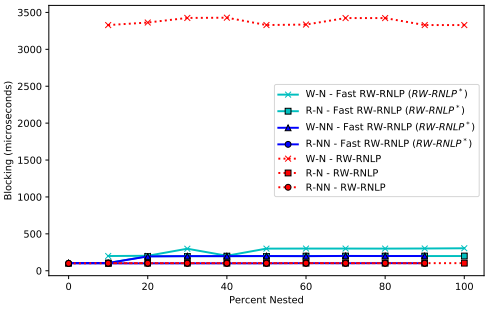

(b) Unlock overhead.




(b) Unlock overhead.



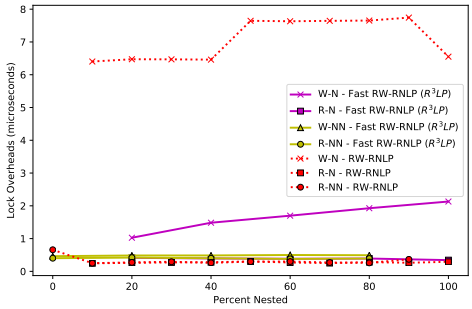

(c) Blocking.



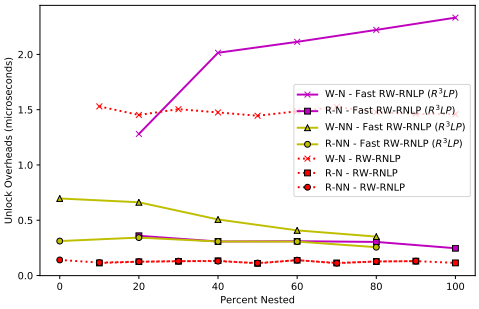

(c) Blocking.

Figure 411: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.

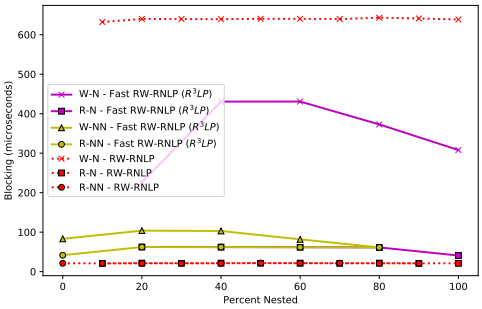
Figure 412: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

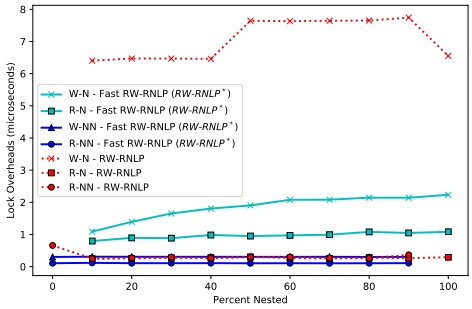



(b) Unlock overhead.

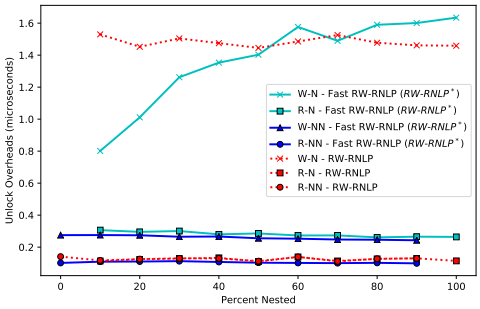



(c) Blocking.

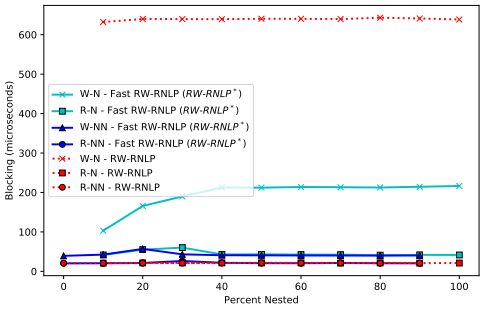
Figure 413: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

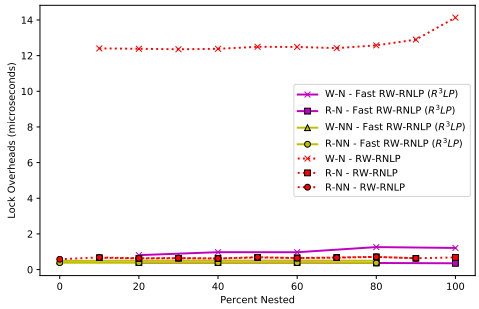



(b) Unlock overhead.

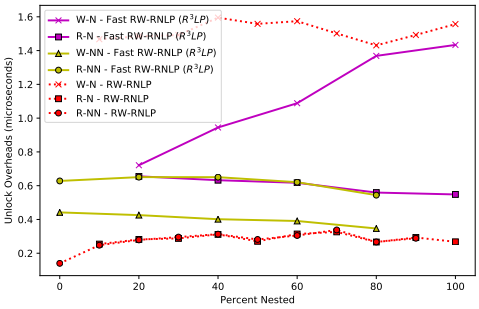



(c) Blocking.

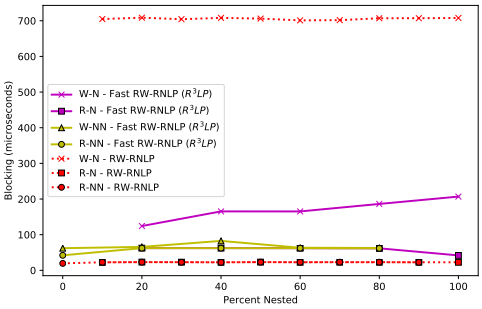
Figure 414: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

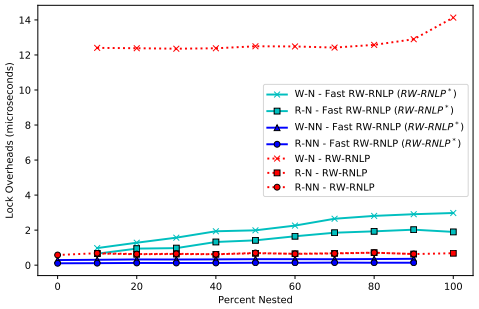



(b) Unlock overhead.

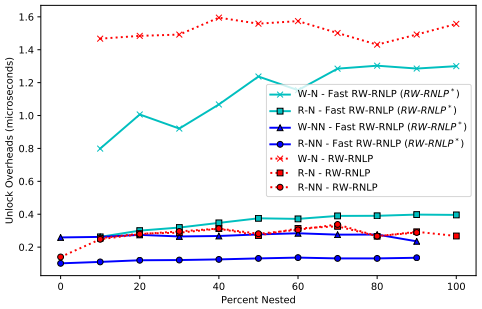



(c) Blocking.

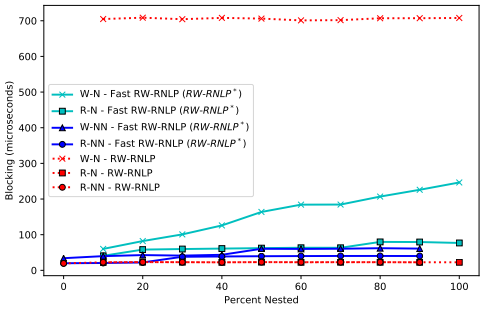
Figure 415: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

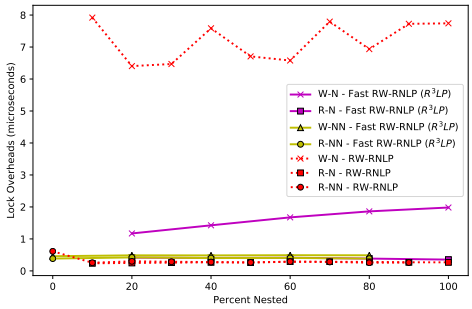



(b) Unlock overhead.

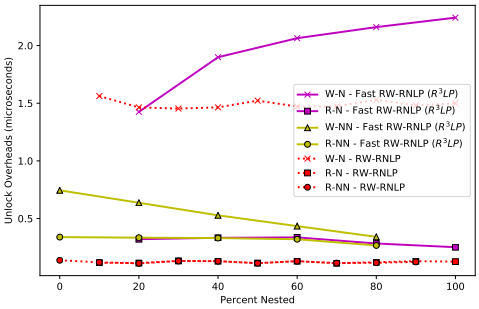



(c) Blocking.

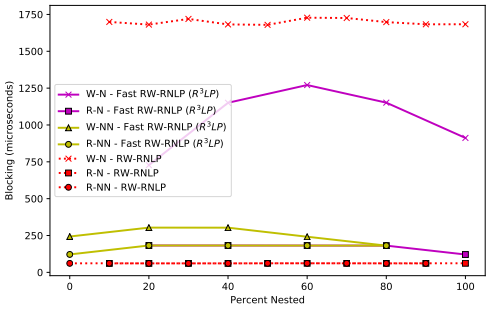
Figure 416: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

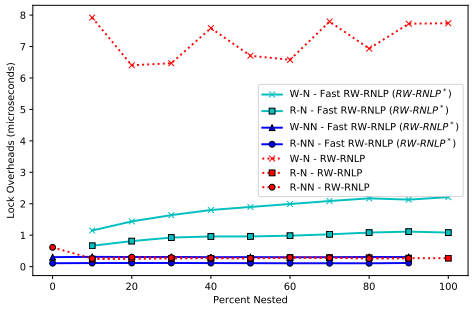



(b) Unlock overhead.

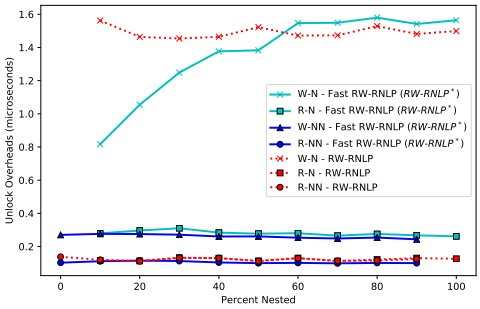



(c) Blocking.

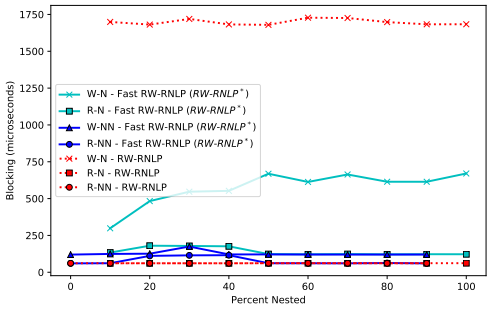
Figure 417: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

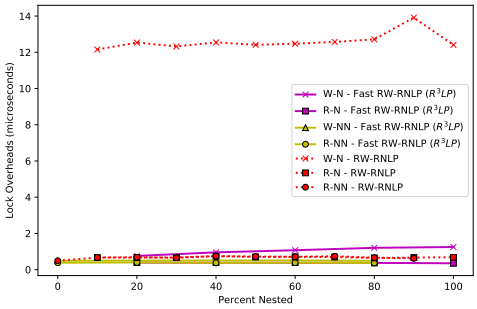



(b) Unlock overhead.

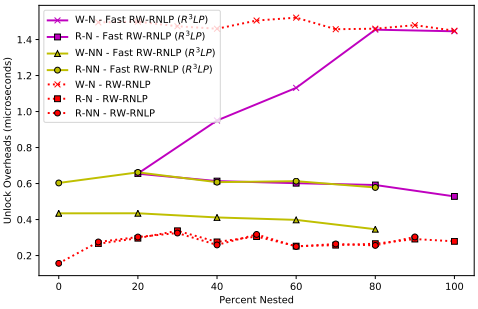



(c) Blocking.

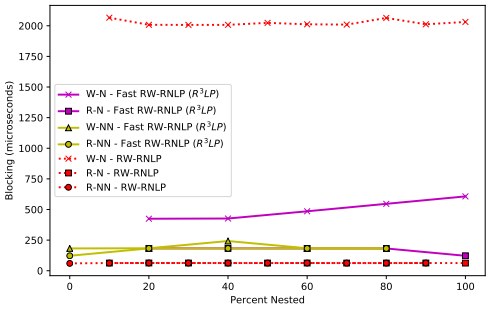
Figure 418: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

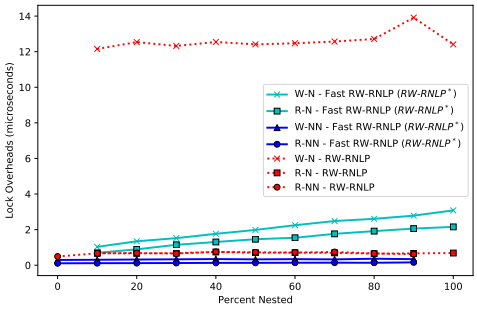



(b) Unlock overhead.

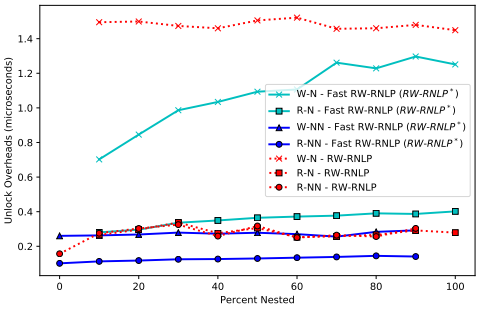



(c) Blocking.

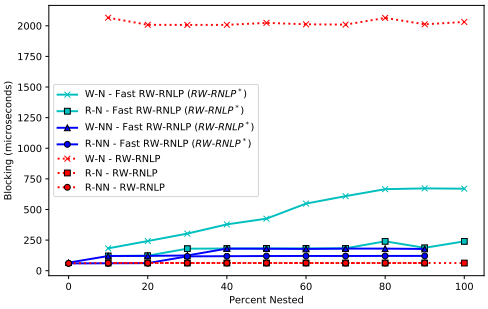
Figure 419: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

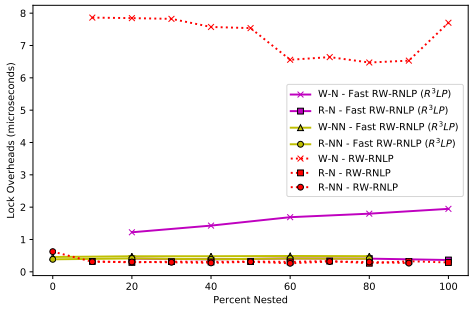



(b) Unlock overhead.

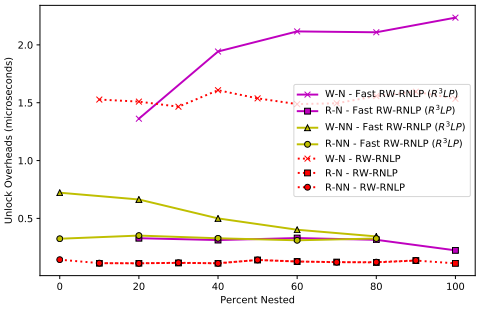



(c) Blocking.

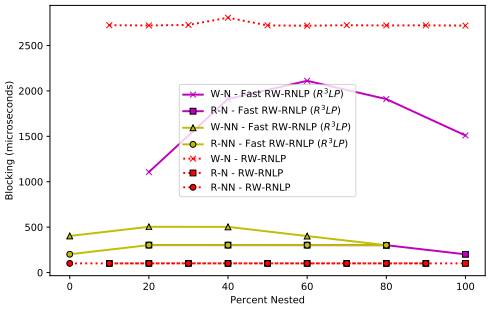
Figure 420: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

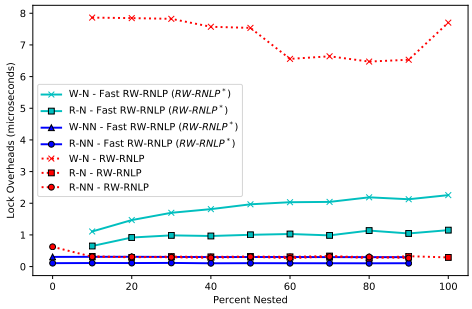



(b) Unlock overhead.

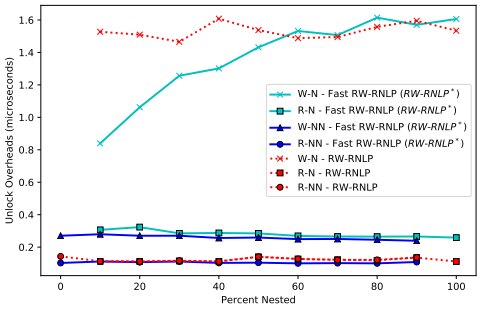



(c) Blocking.

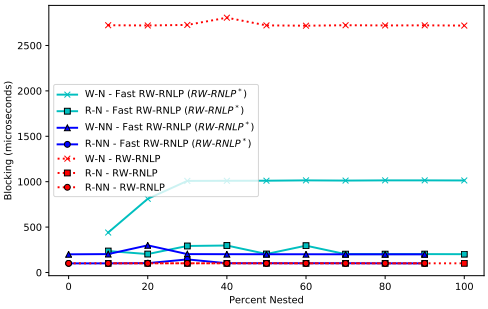
Figure 421: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

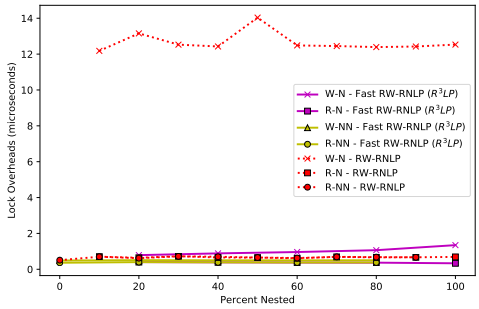



(b) Unlock overhead.

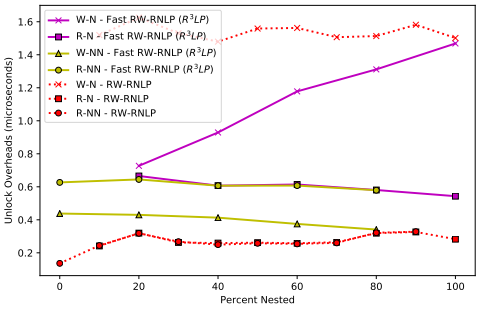



(c) Blocking.

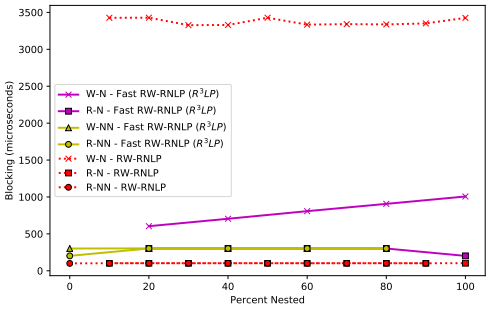
Figure 422: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

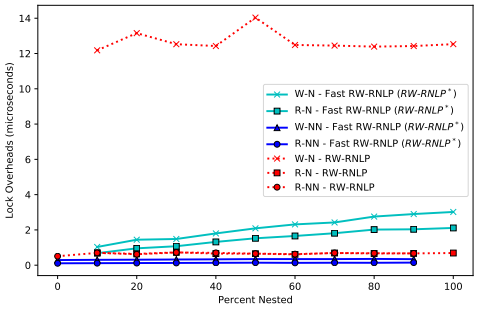



(b) Unlock overhead.

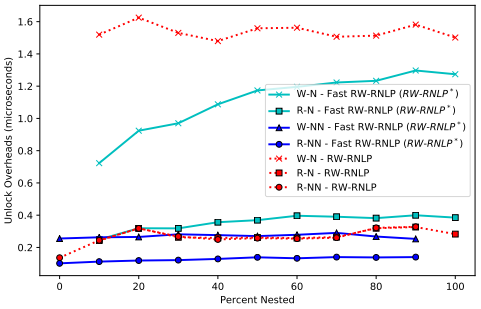



(c) Blocking.

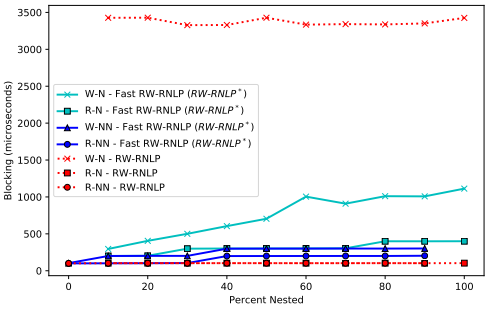
Figure 423: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

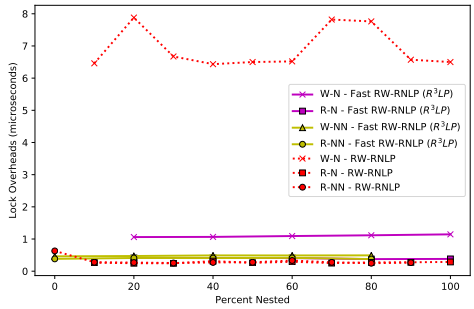



(b) Unlock overhead.

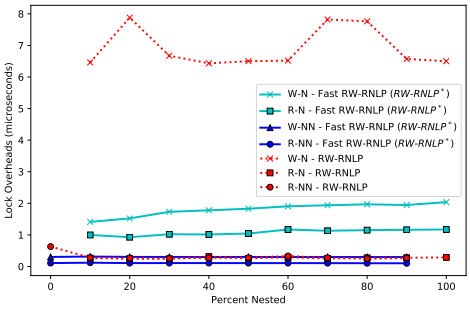



(c) Blocking.

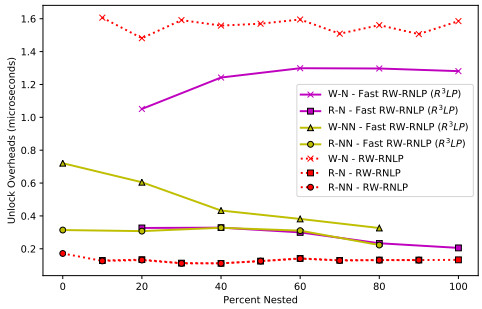
Figure 424: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.



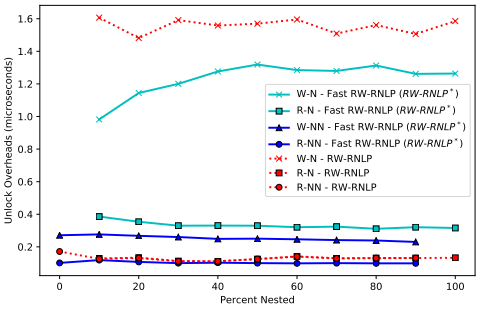

(a) Lock overhead.



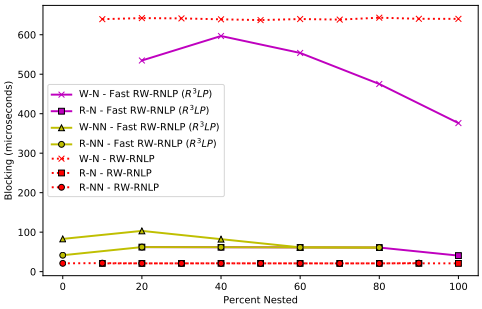

(a) Lock overhead.



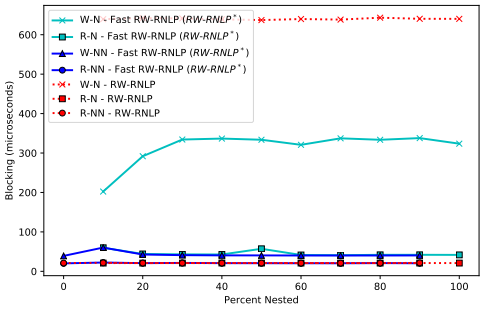

(b) Unlock overhead.




(b) Unlock overhead.



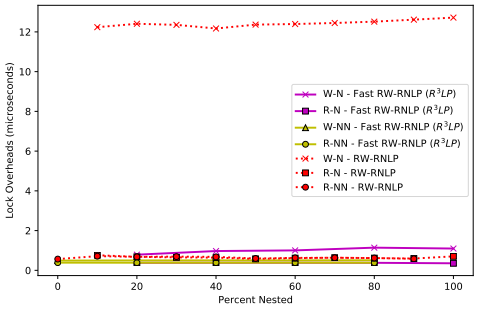

(c) Blocking.



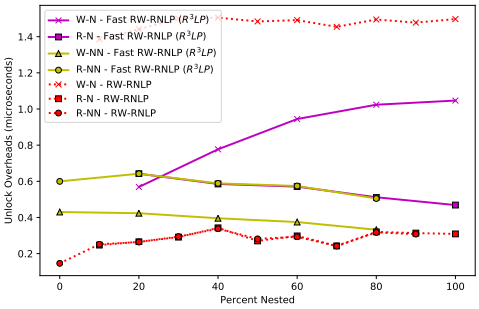

(c) Blocking.

Figure 425: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.

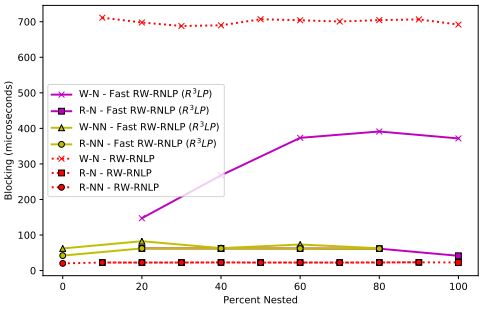
Figure 426: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

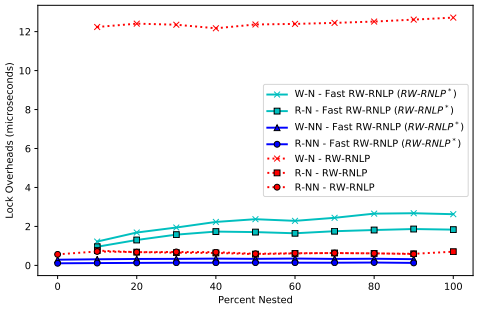



(b) Unlock overhead.

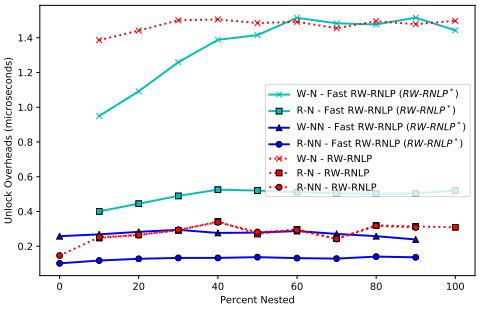



(c) Blocking.

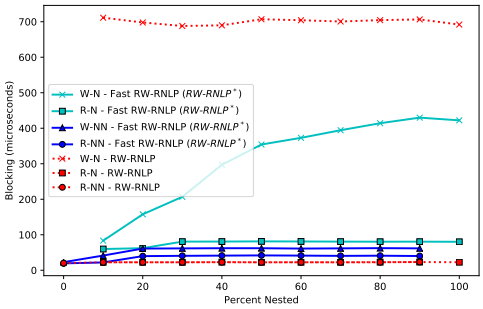
Figure 427: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

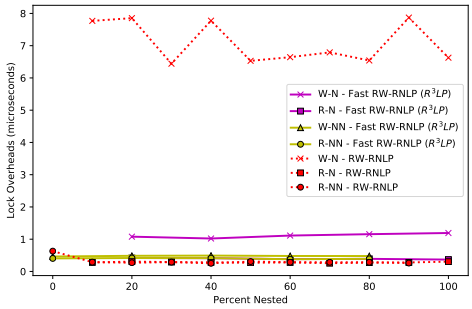



(b) Unlock overhead.

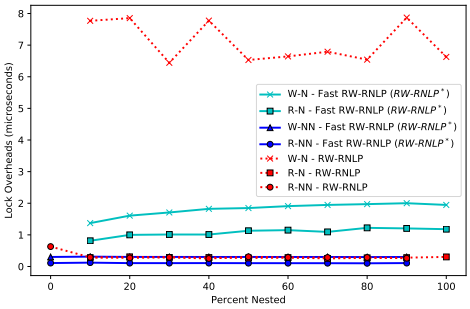



(c) Blocking.

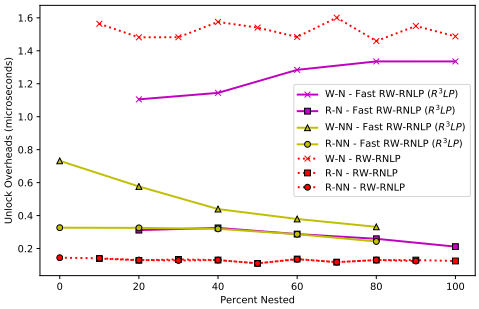
Figure 428: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.



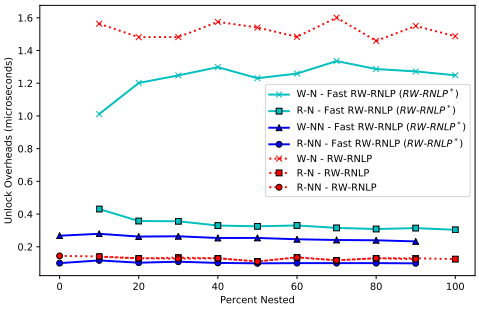

(a) Lock overhead.



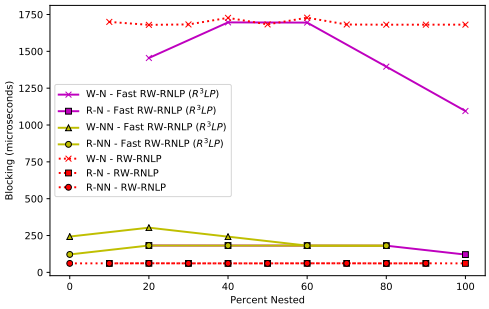

(a) Lock overhead.



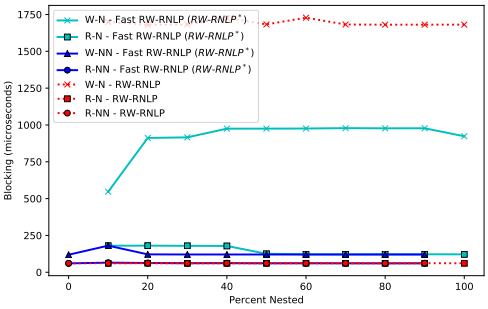

(b) Unlock overhead.




(b) Unlock overhead.



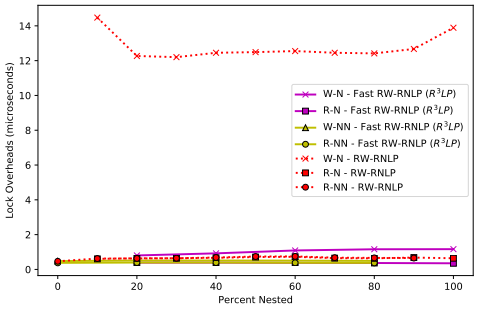

(c) Blocking.



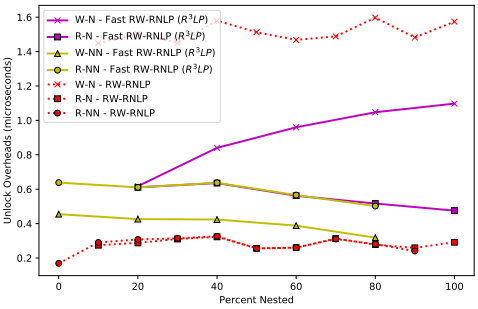

(c) Blocking.

Figure 429: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.

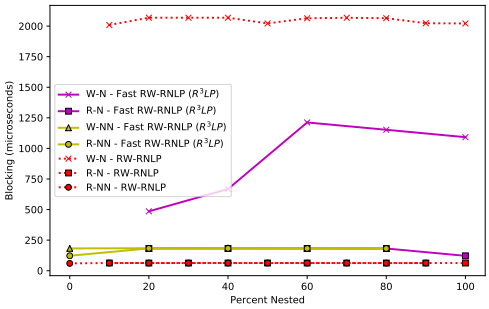
Figure 430: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

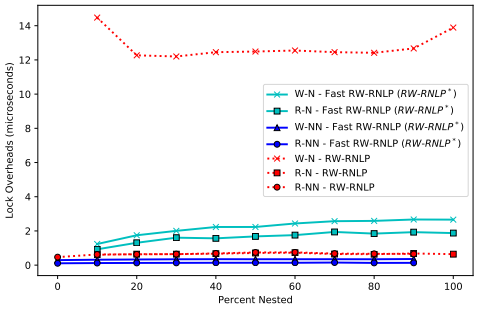



(b) Unlock overhead.

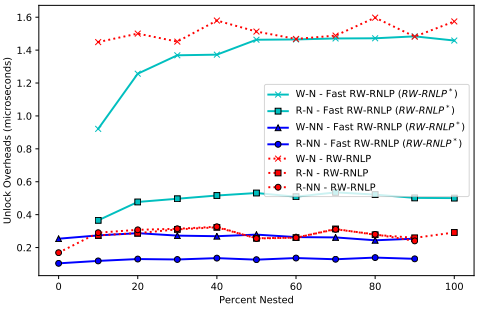



(c) Blocking.

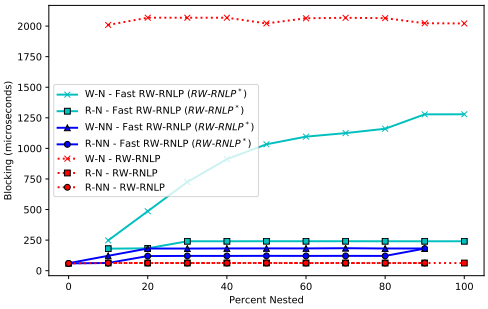
Figure 431: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

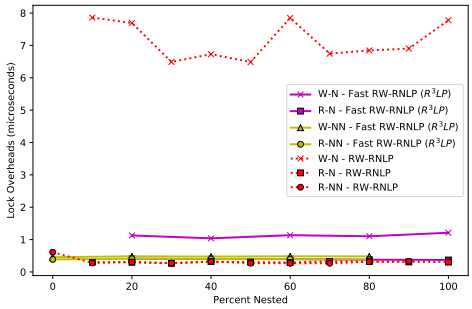



(b) Unlock overhead.

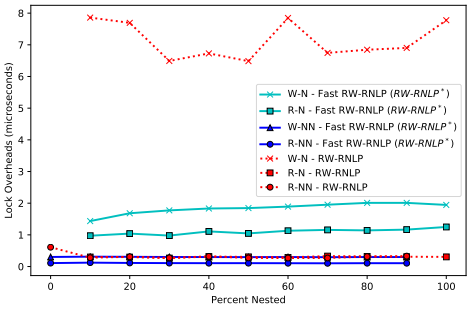



(c) Blocking.

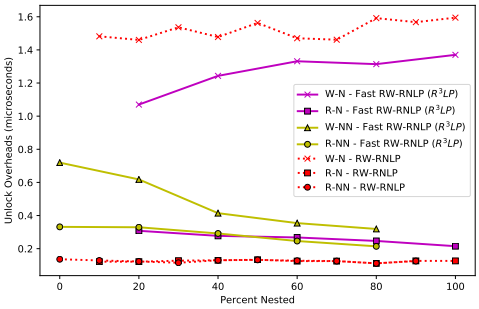
Figure 432: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.



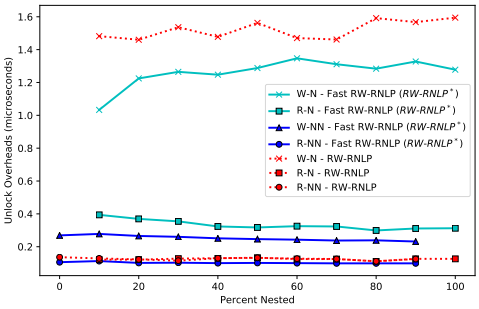

(a) Lock overhead.



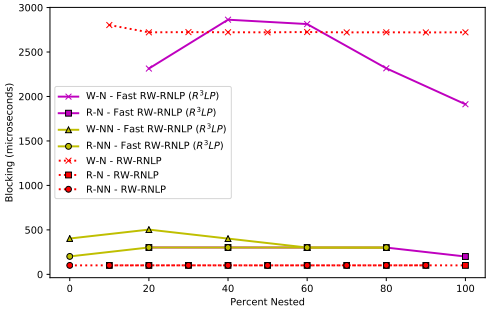

(a) Lock overhead.



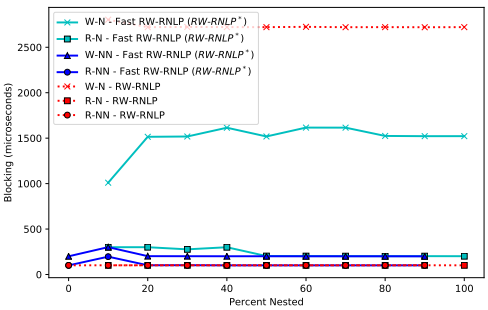

(b) Unlock overhead.




(b) Unlock overhead.



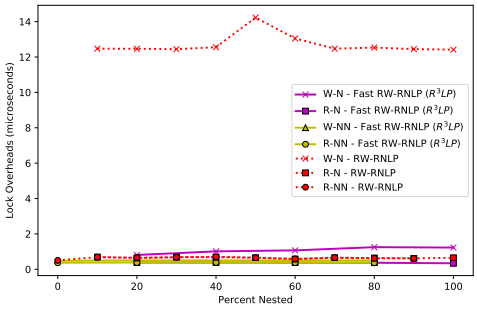

(c) Blocking.



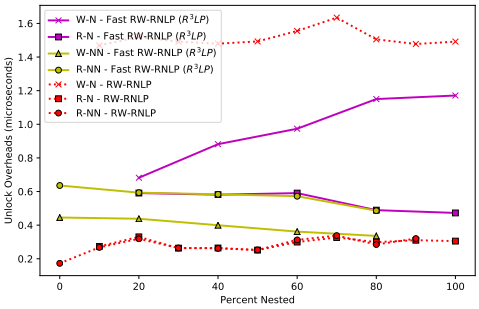

(c) Blocking.

Figure 433: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.

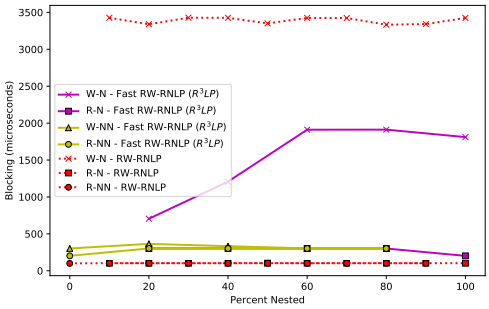
Figure 434: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

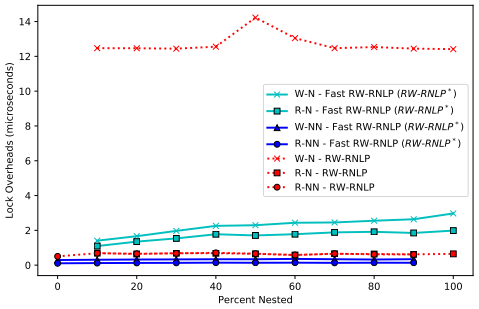



(b) Unlock overhead.

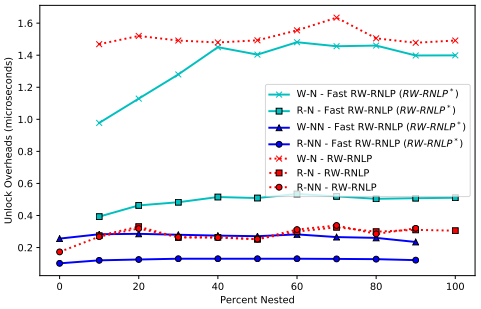



(c) Blocking.

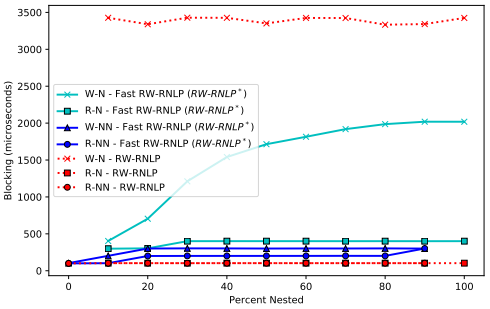
Figure 435: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

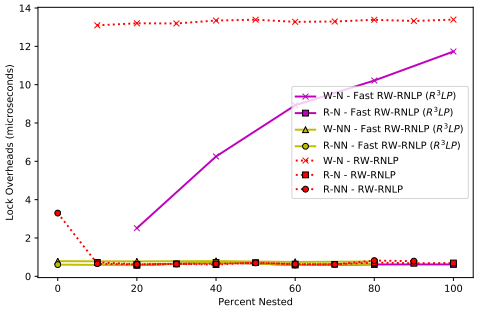



(b) Unlock overhead.

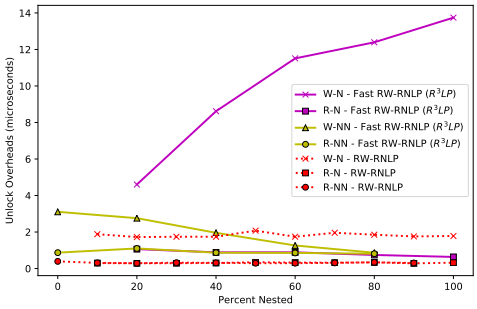



(c) Blocking.

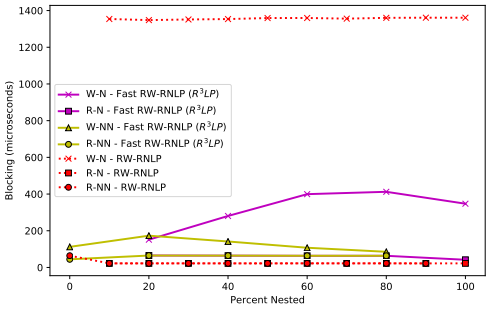
Figure 436: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

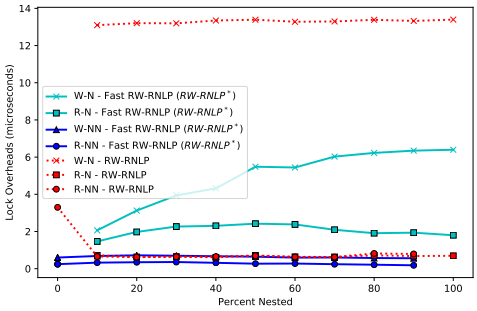



(b) Unlock overhead.

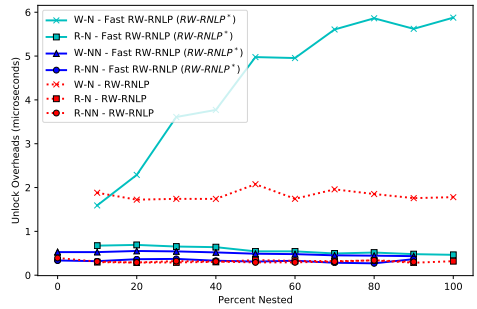



(c) Blocking.

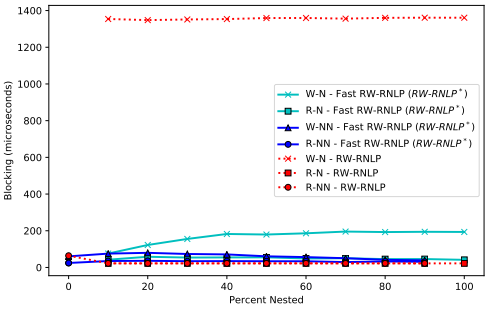
Figure 437: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

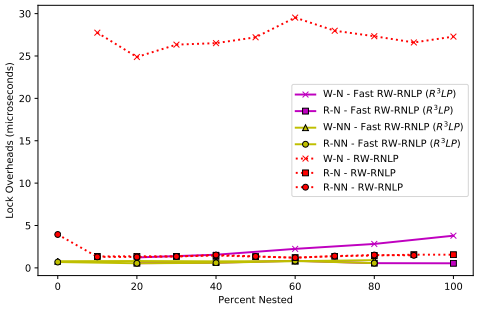



(b) Unlock overhead.

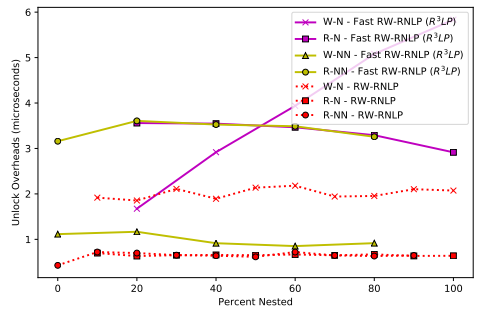



(c) Blocking.

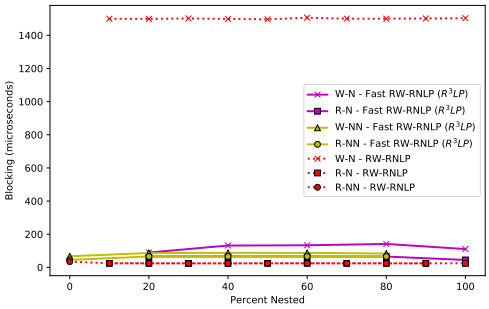
Figure 438: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

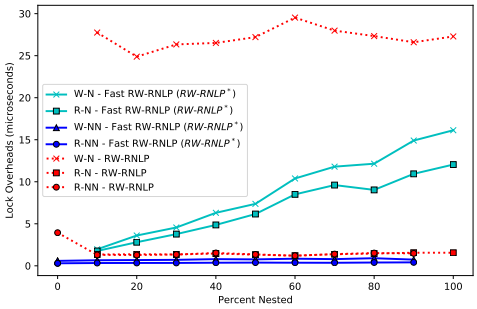



(b) Unlock overhead.

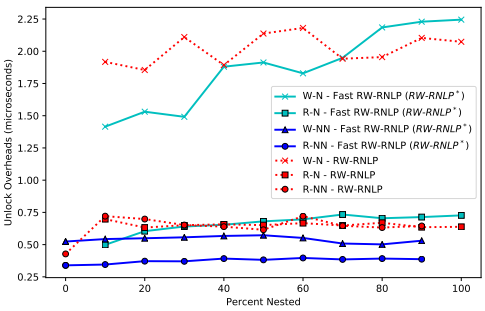



(c) Blocking.

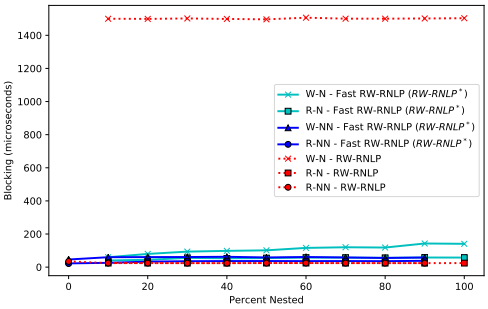
Figure 439: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

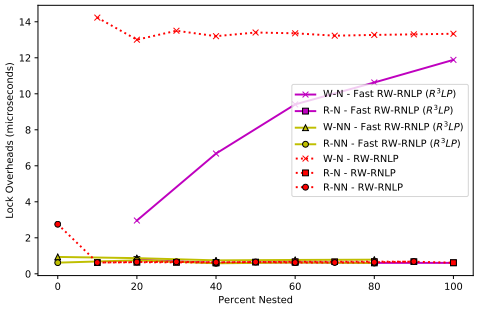



(b) Unlock overhead.

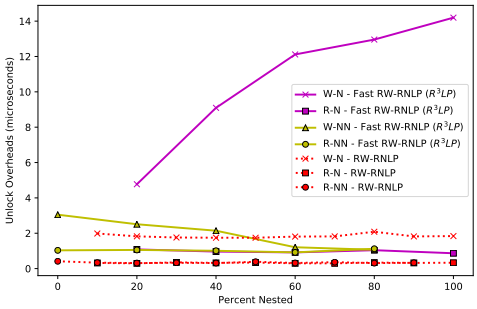



(c) Blocking.

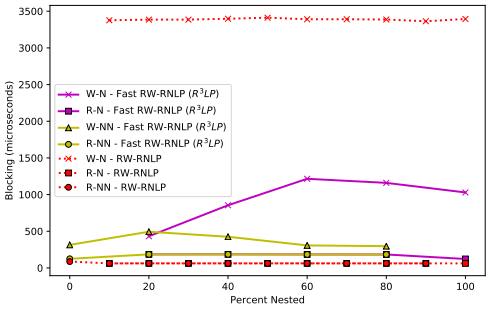
Figure 440: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

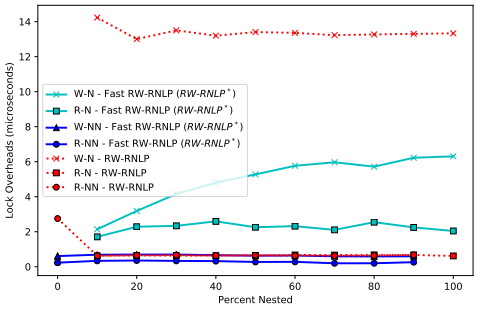



(b) Unlock overhead.

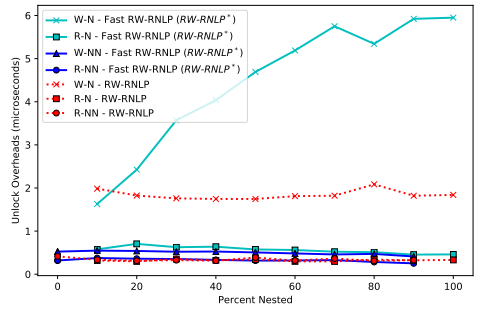



(c) Blocking.

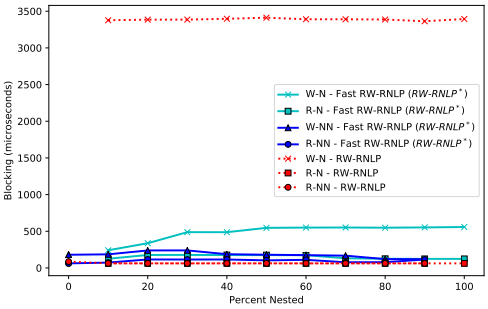
Figure 441: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

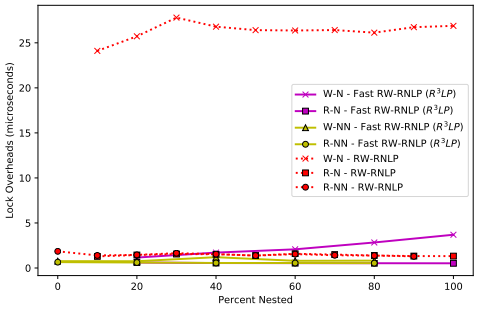



(b) Unlock overhead.

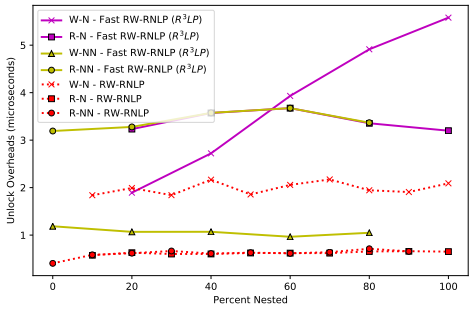



(c) Blocking.

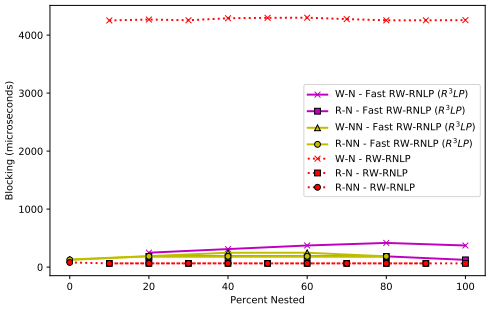
Figure 442: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

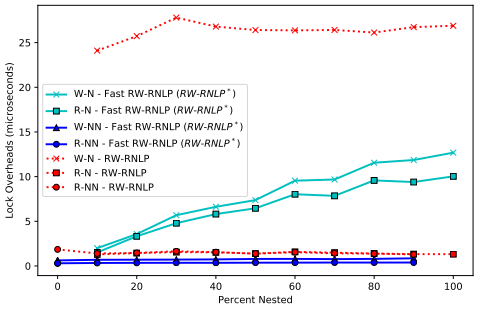



(b) Unlock overhead.

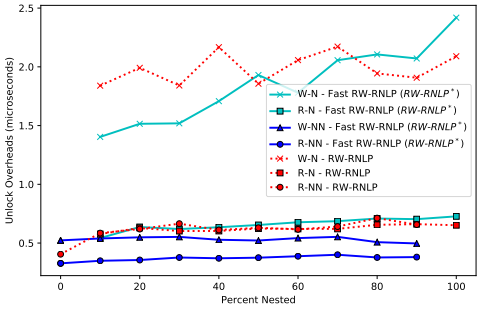



(c) Blocking.

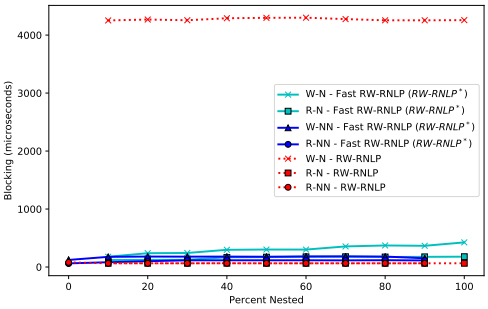
Figure 443: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

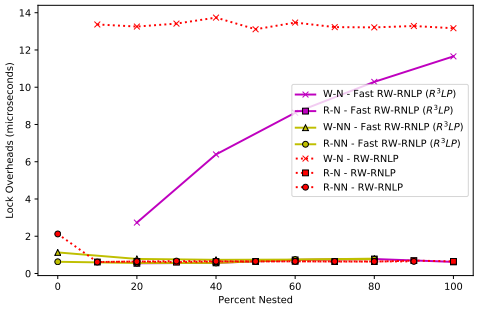



(b) Unlock overhead.

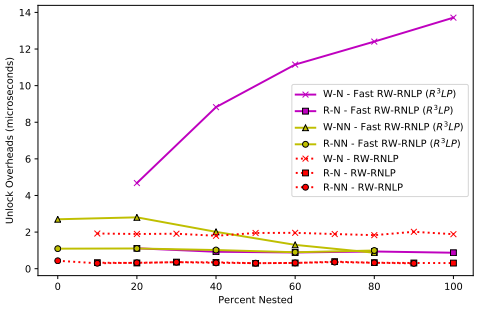



(c) Blocking.

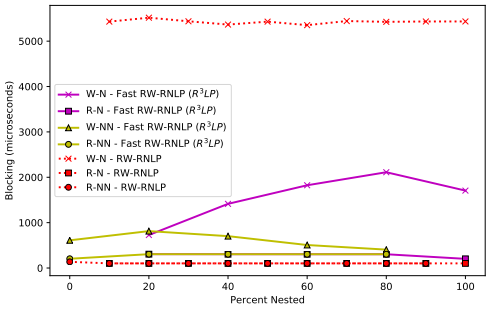
Figure 444: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

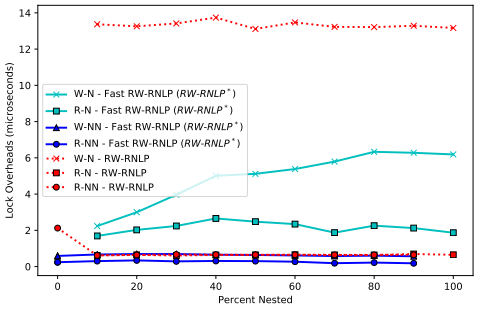



(b) Unlock overhead.

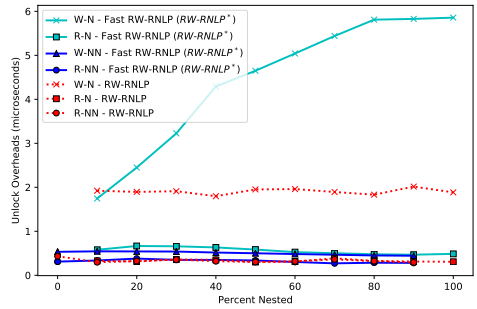



(c) Blocking.

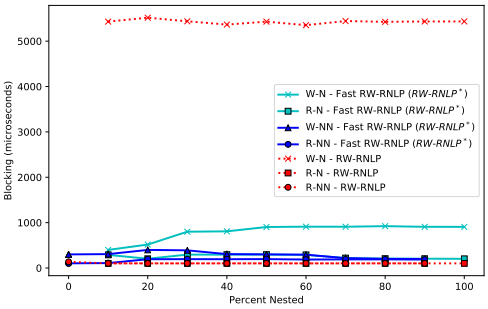
Figure 445: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

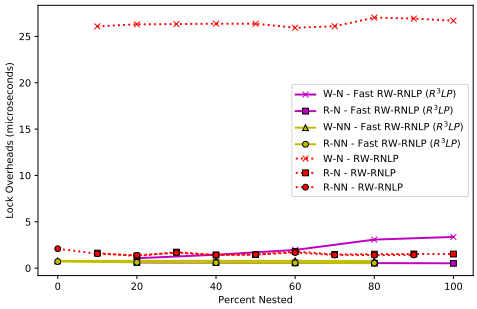



(b) Unlock overhead.

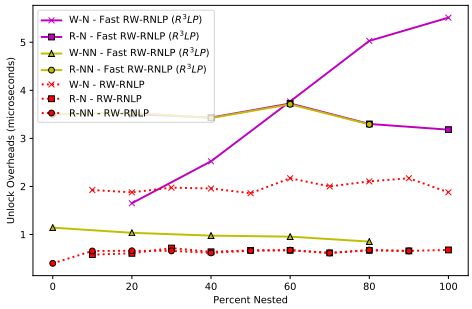



(c) Blocking.

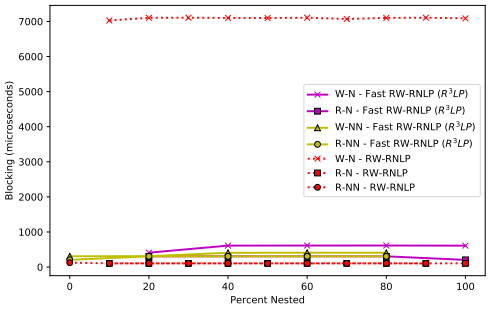
Figure 446: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

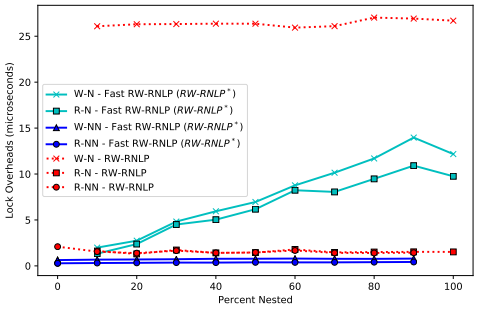



(b) Unlock overhead.

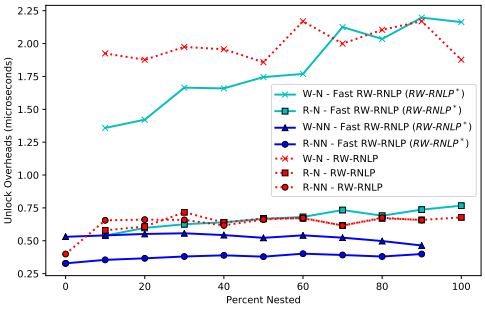



(c) Blocking.

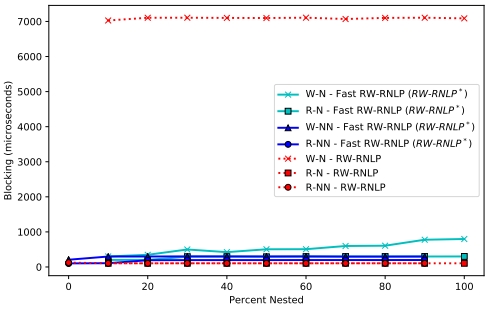
Figure 447: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

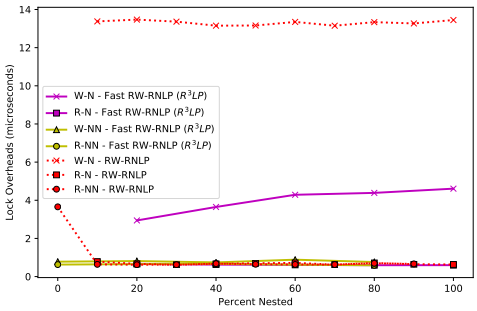



(b) Unlock overhead.

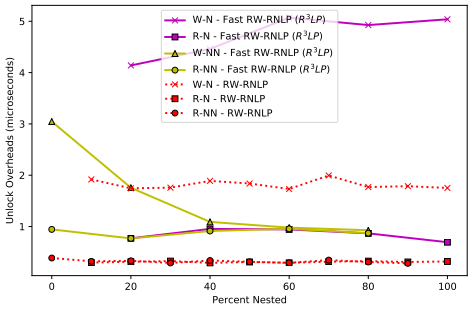



(c) Blocking.

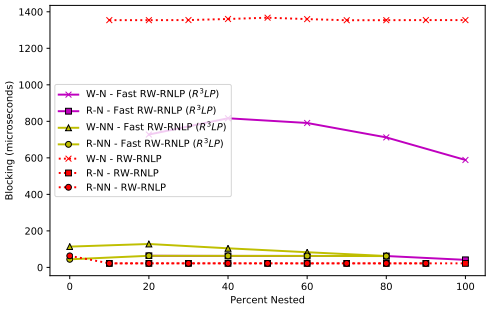
Figure 448: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 2$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

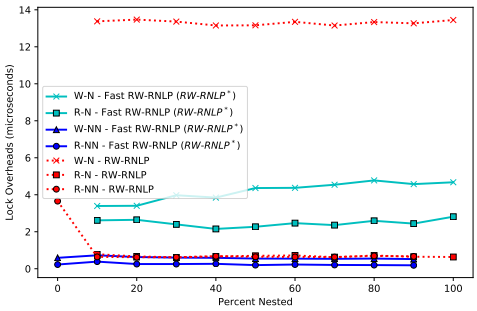



(b) Unlock overhead.

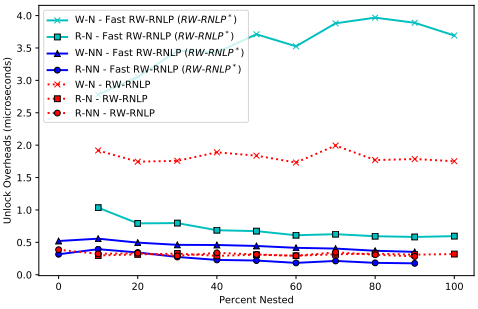



(c) Blocking.

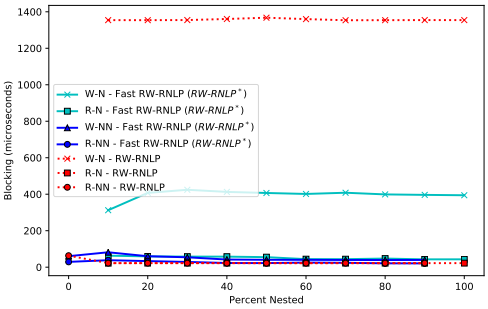
Figure 449: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

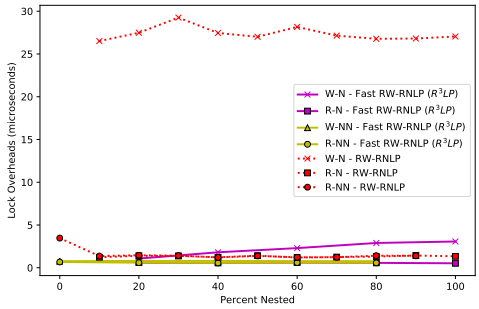



(b) Unlock overhead.

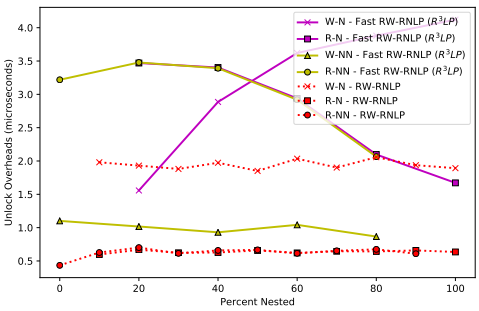



(c) Blocking.

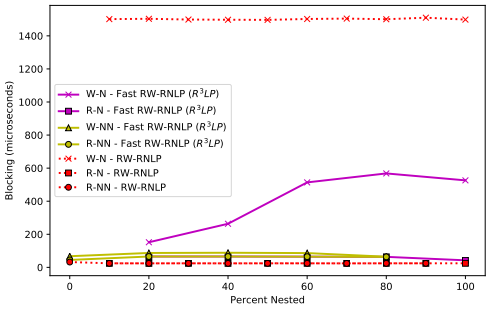
Figure 450: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

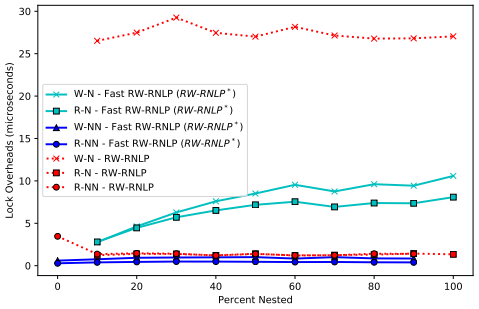



(b) Unlock overhead.

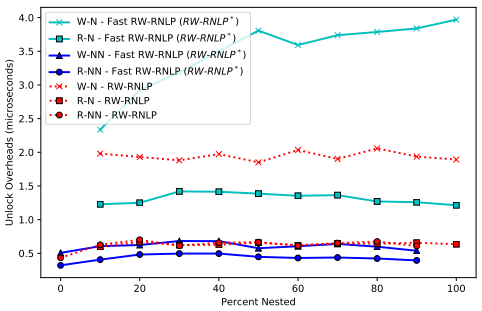



(c) Blocking.

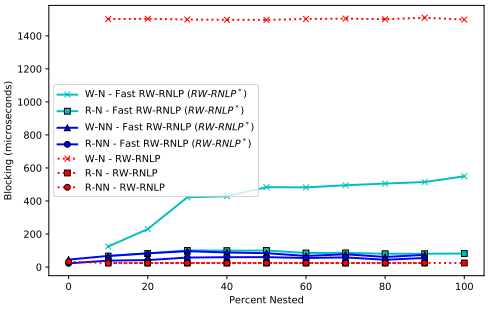
Figure 451: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

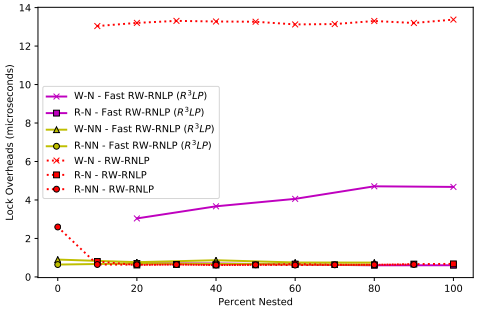



(b) Unlock overhead.

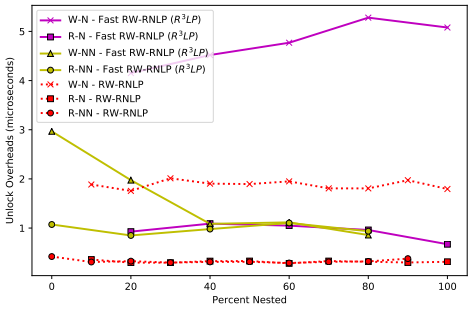



(c) Blocking.

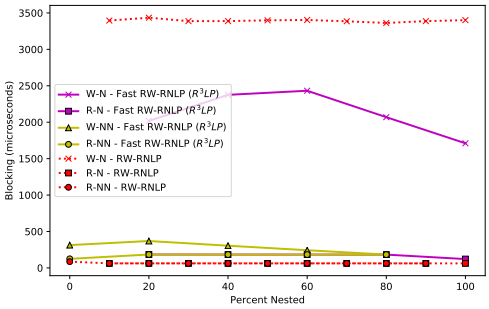
Figure 452: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

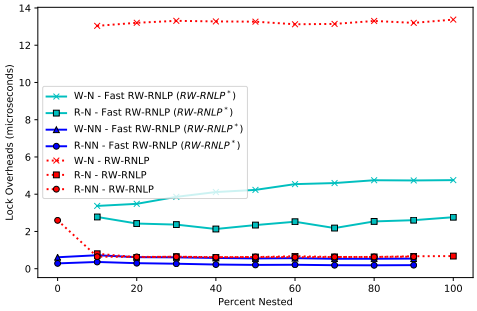



(b) Unlock overhead.

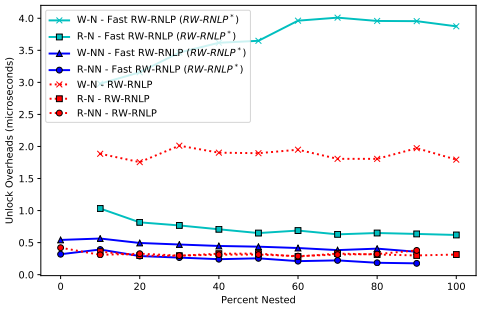



(c) Blocking.

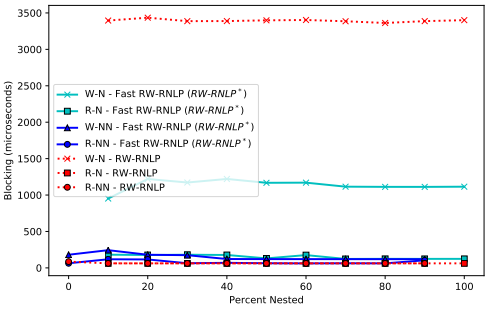
Figure 453: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

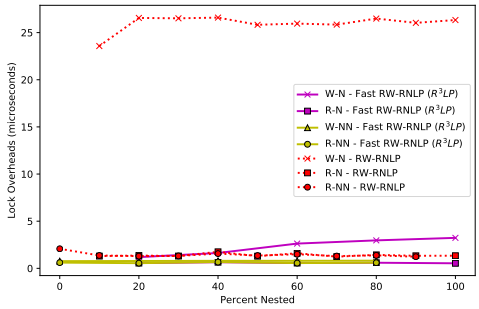



(b) Unlock overhead.

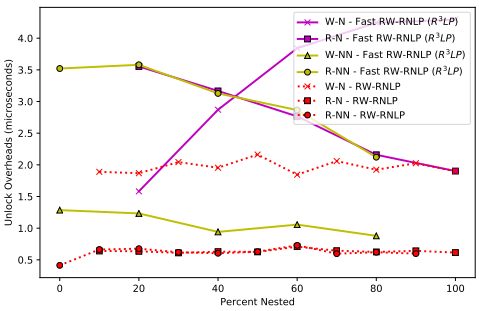



(c) Blocking.

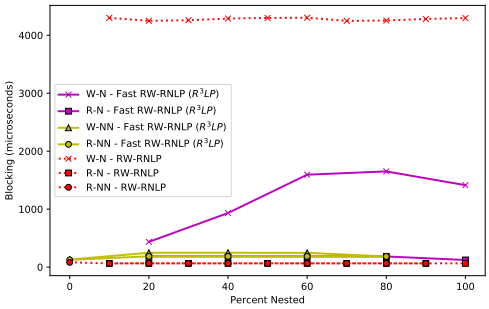
Figure 454: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

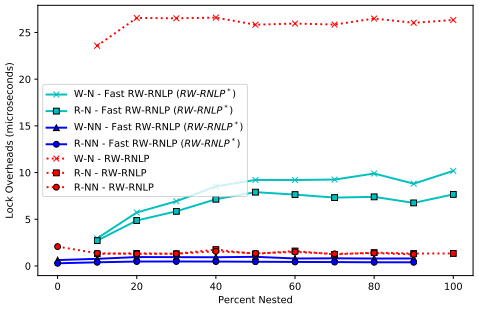



(b) Unlock overhead.

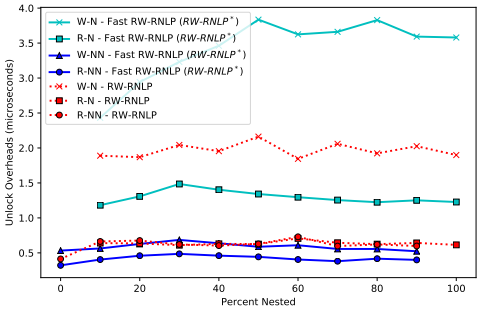



(c) Blocking.

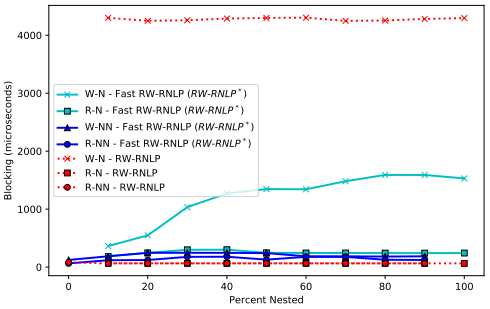
Figure 455: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

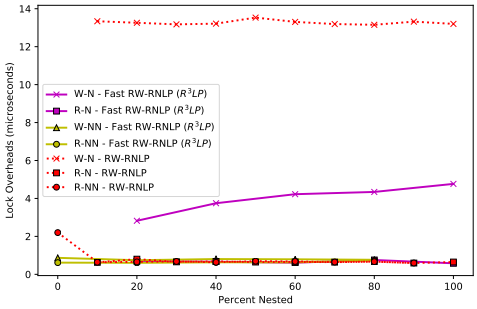



(b) Unlock overhead.

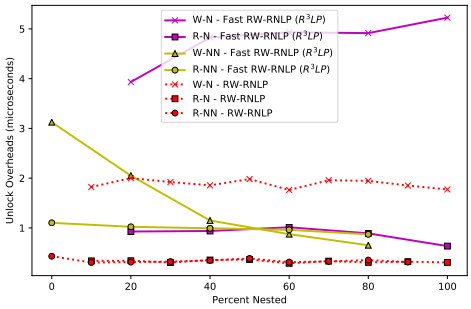



(c) Blocking.

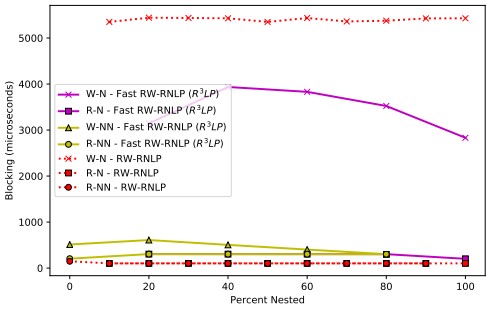
Figure 456: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

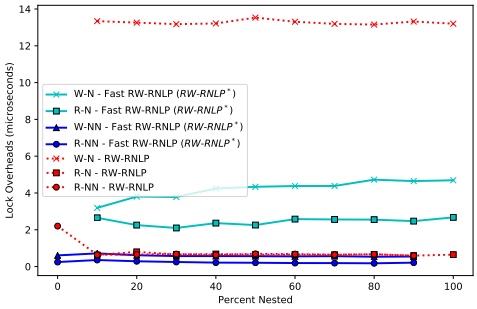



(b) Unlock overhead.

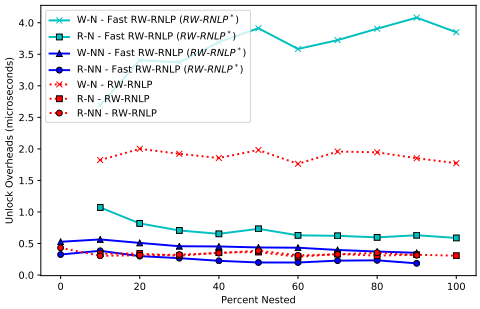



(c) Blocking.

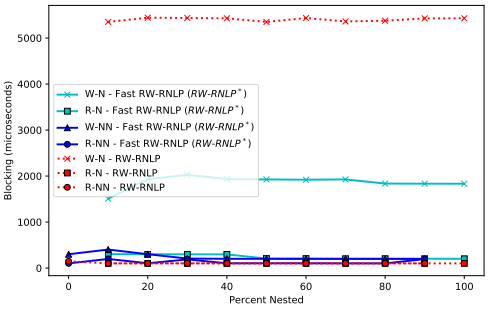
Figure 457: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

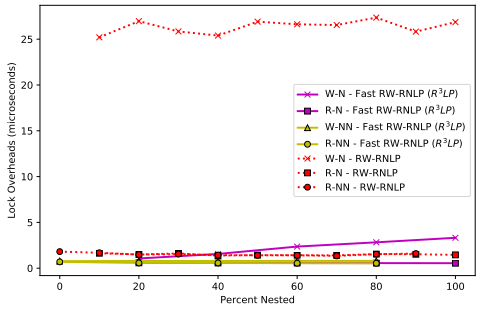



(b) Unlock overhead.

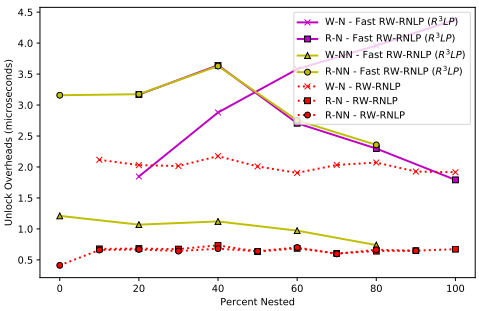



(c) Blocking.

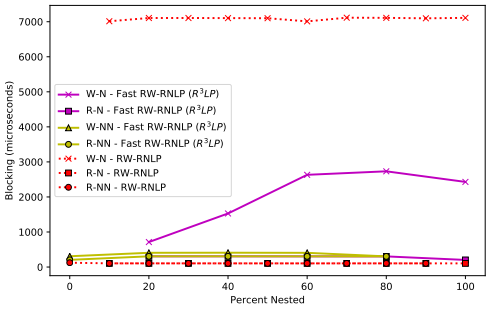
Figure 458: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

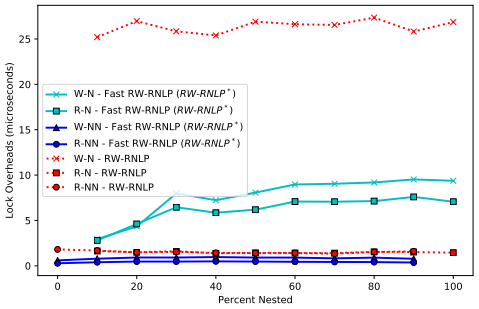



(b) Unlock overhead.

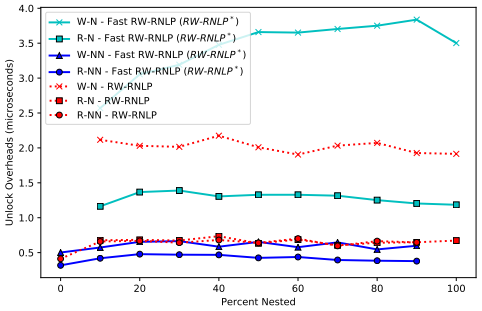



(c) Blocking.

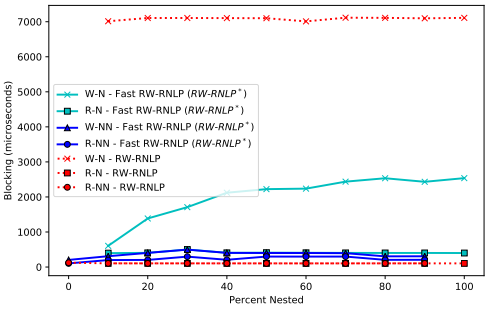
Figure 459: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

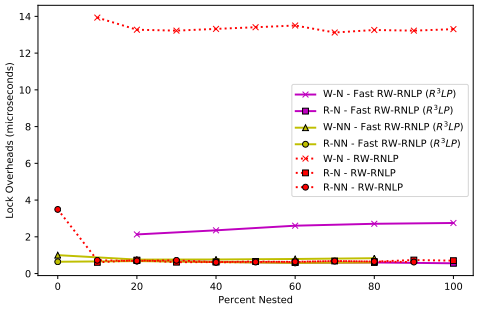



(b) Unlock overhead.

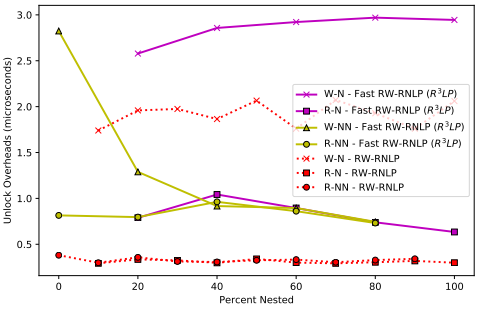



(c) Blocking.

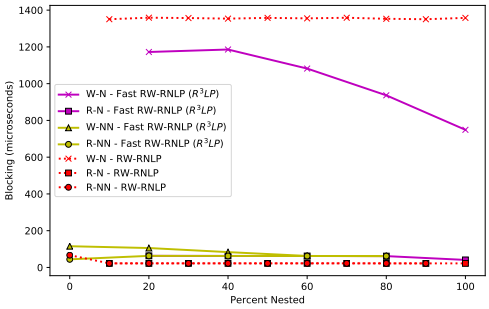
Figure 460: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 4$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

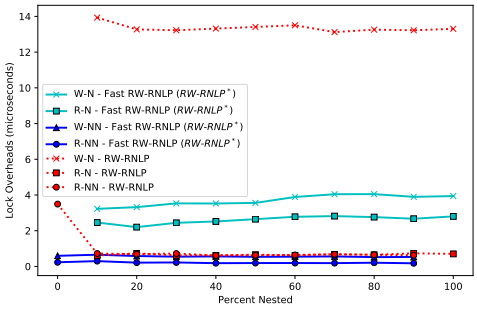



(b) Unlock overhead.

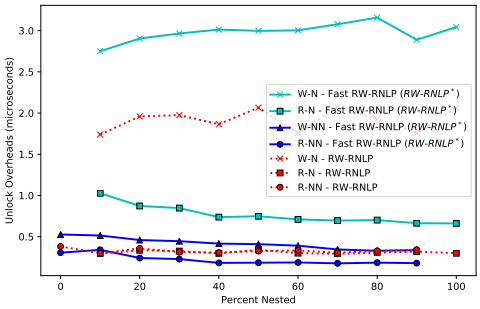



(c) Blocking.

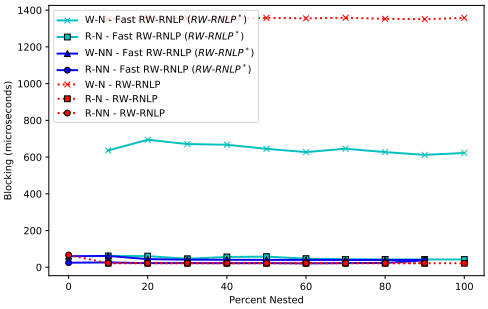
Figure 461: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

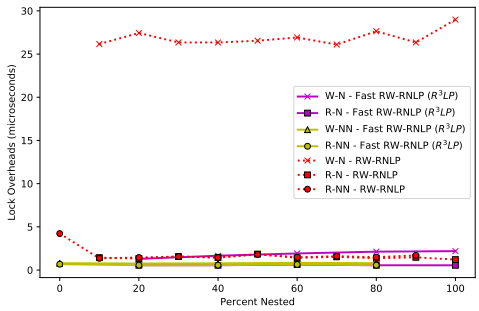



(b) Unlock overhead.

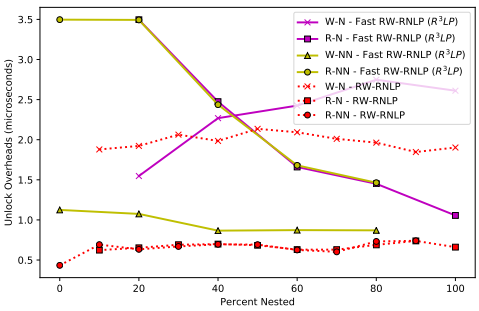



(c) Blocking.

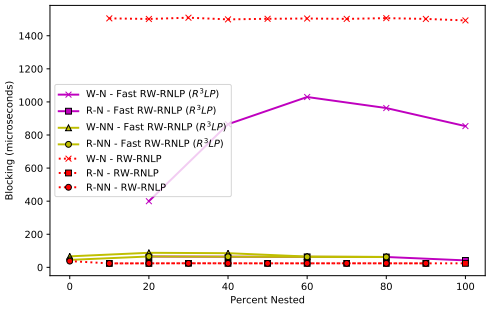
Figure 462: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

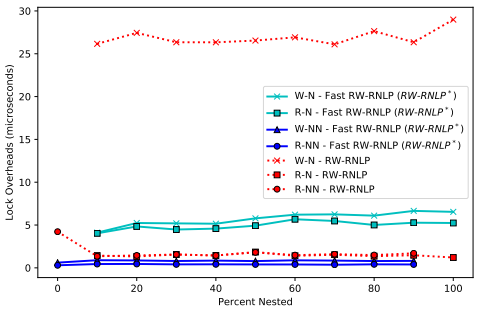



(b) Unlock overhead.

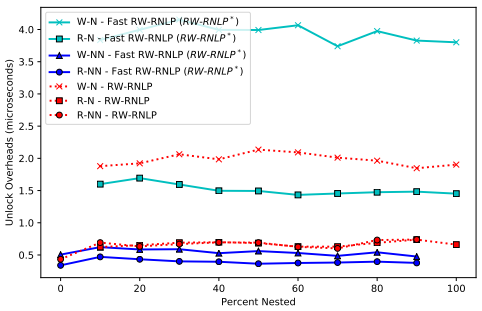



(c) Blocking.

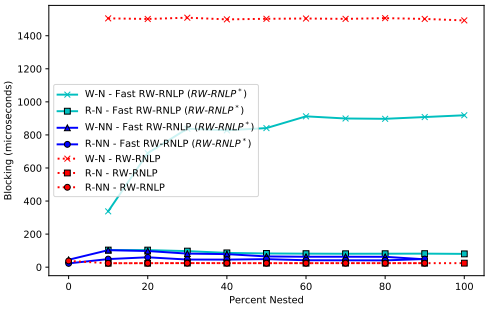
Figure 463: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

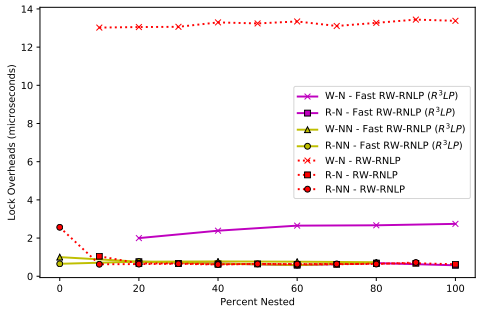



(b) Unlock overhead.

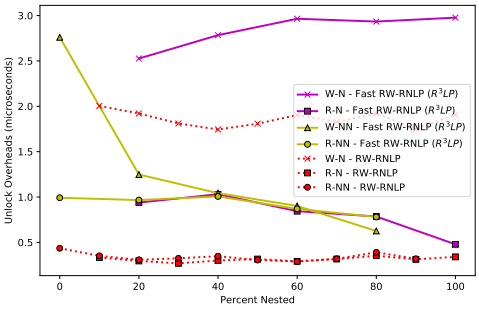



(c) Blocking.

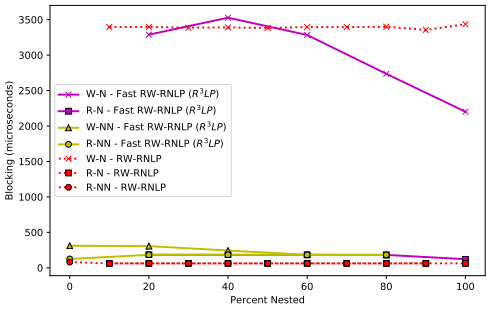
Figure 464: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

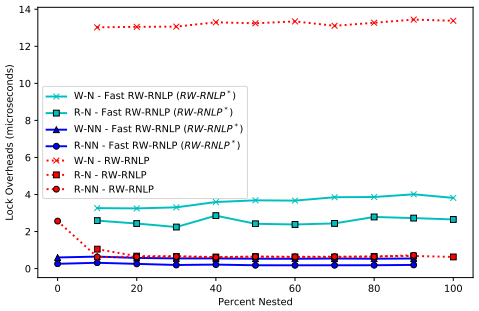



(b) Unlock overhead.

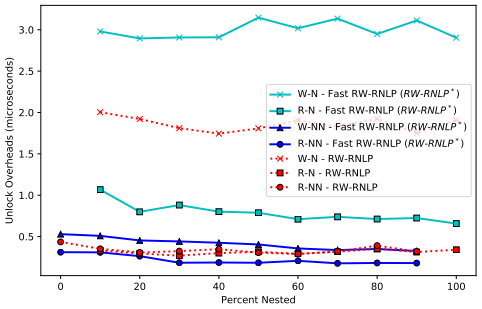



(c) Blocking.

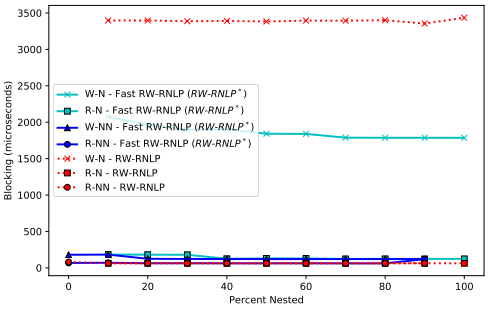
Figure 465: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

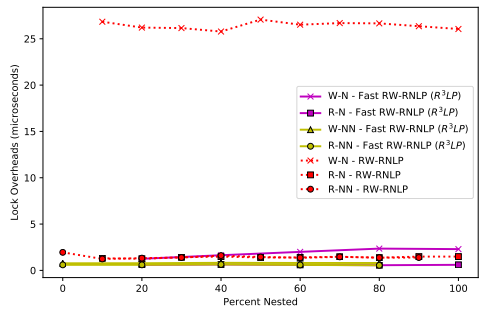



(b) Unlock overhead.

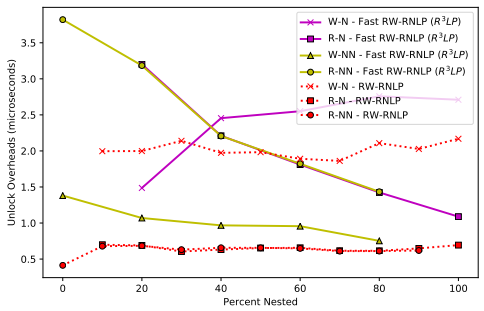



(c) Blocking.

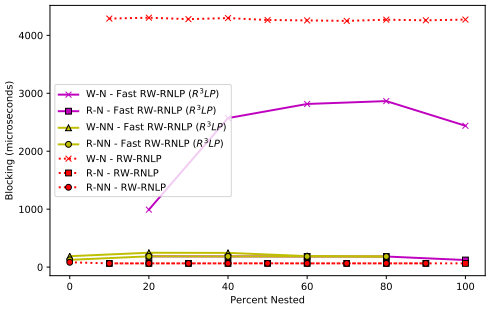
Figure 466: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

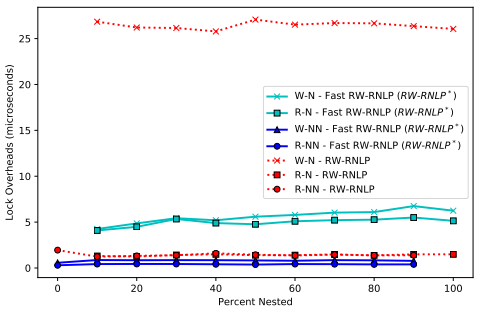



(b) Unlock overhead.

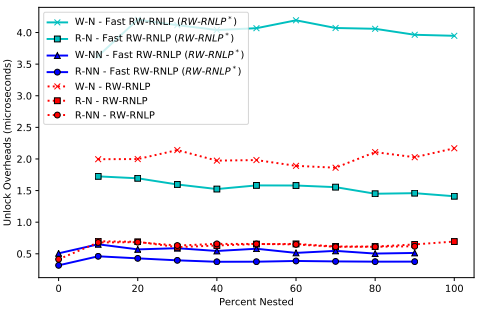



(c) Blocking.

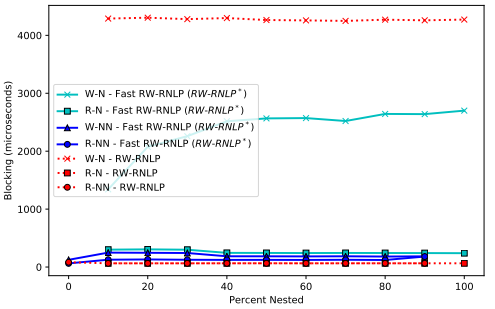
Figure 467: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

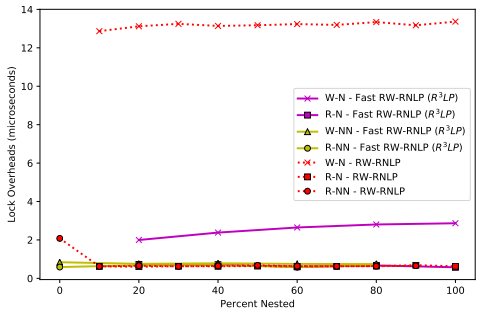



(b) Unlock overhead.

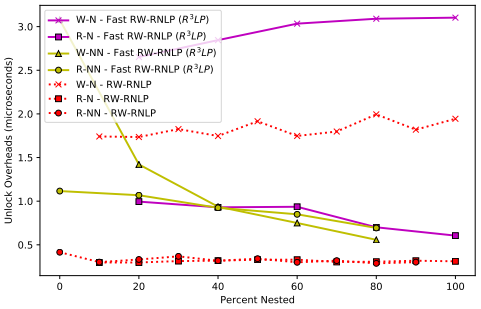



(c) Blocking.

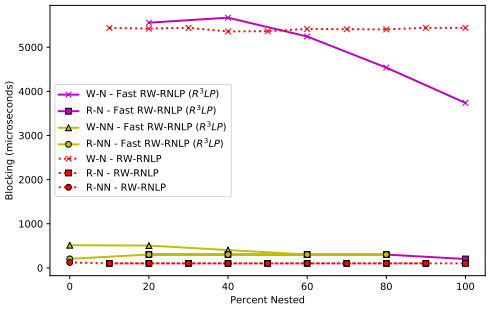
Figure 468: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

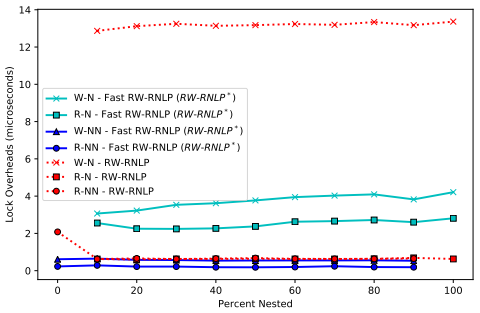



(b) Unlock overhead.

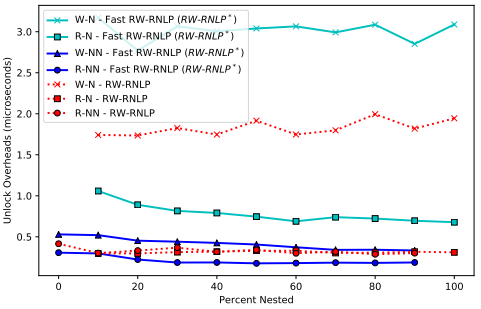



(c) Blocking.

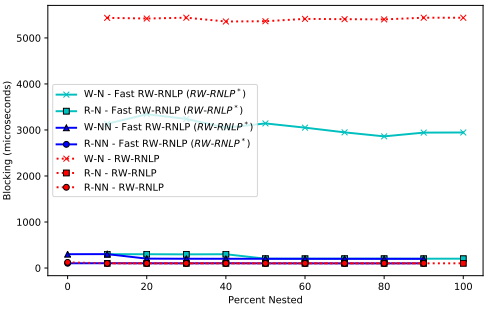
Figure 469: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

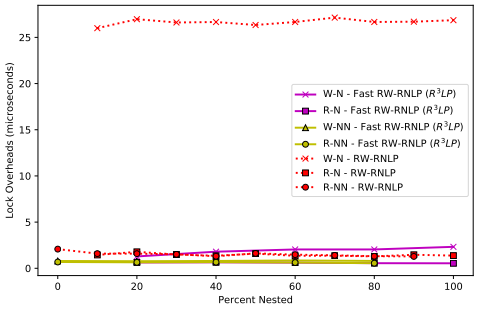



(b) Unlock overhead.

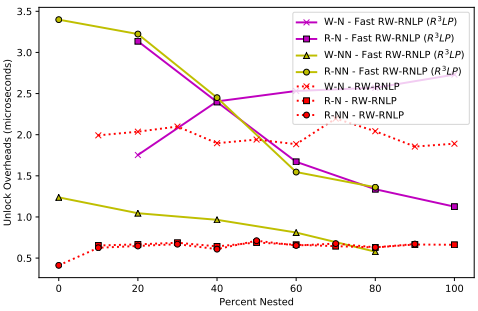



(c) Blocking.

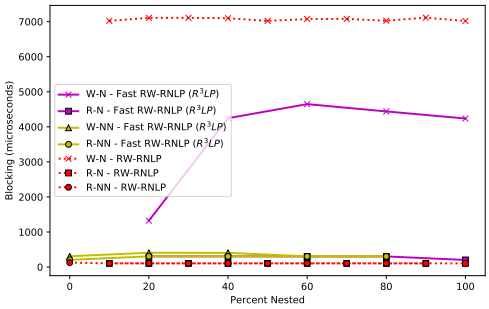
Figure 470: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

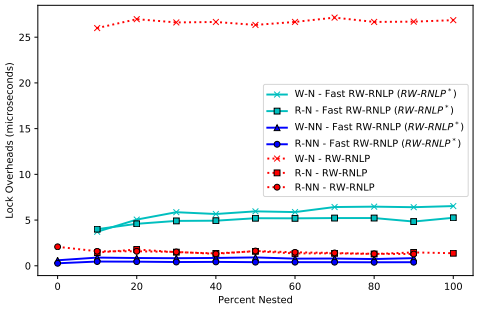



(b) Unlock overhead.

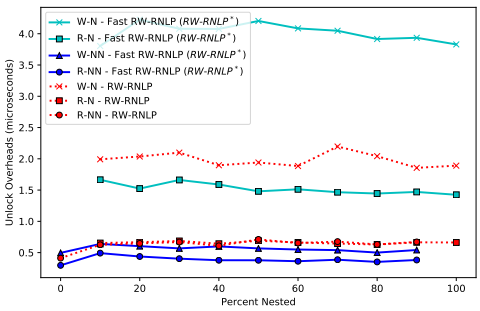



(c) Blocking.

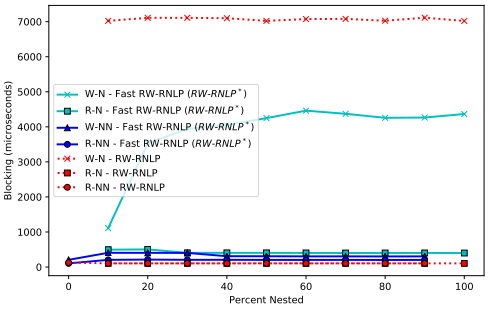
Figure 471: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

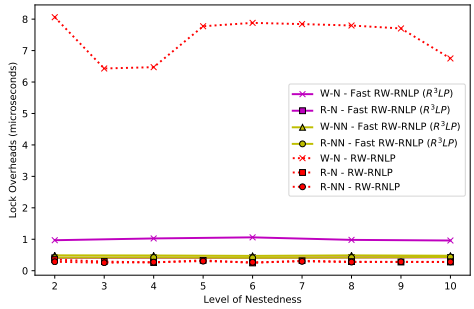



(b) Unlock overhead.

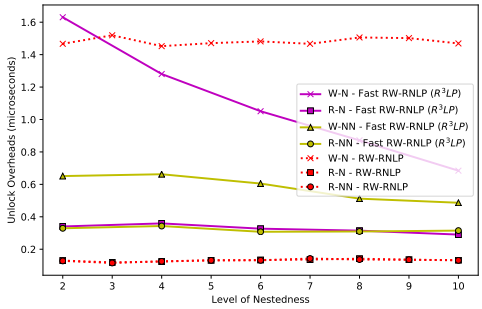



(c) Blocking.

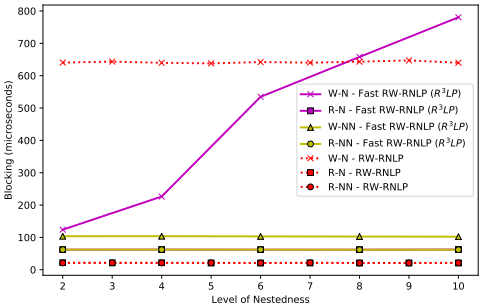
Figure 472: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, $|D_i| = 1$ for non-nested requests, and $|D_i| = 6$ for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability as shown. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP if some requests were nested, as read requests can access any resource.




(a) Lock overhead.

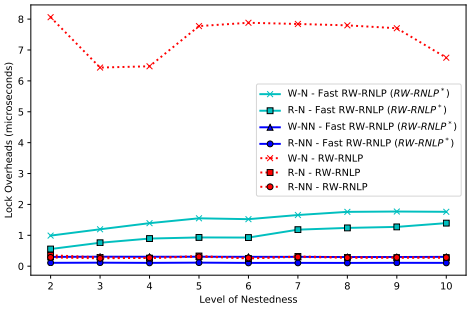



(b) Unlock overhead.

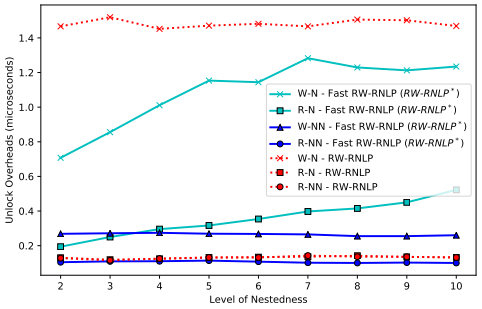



(c) Blocking.

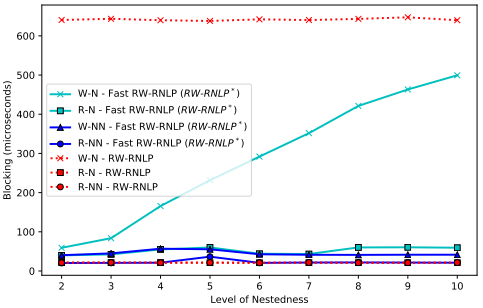
Figure 473: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

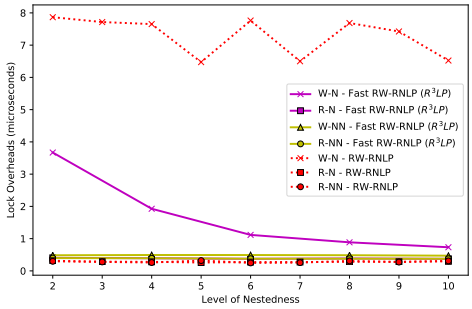



(b) Unlock overhead.

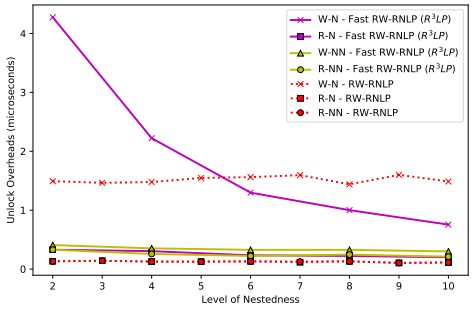



(c) Blocking.

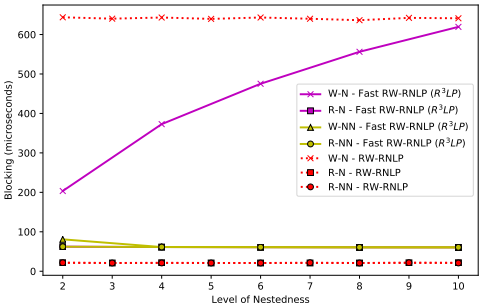
Figure 474: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

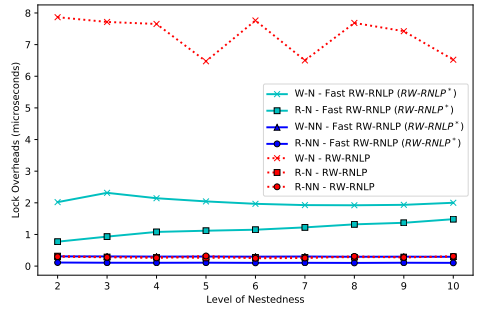



(b) Unlock overhead.

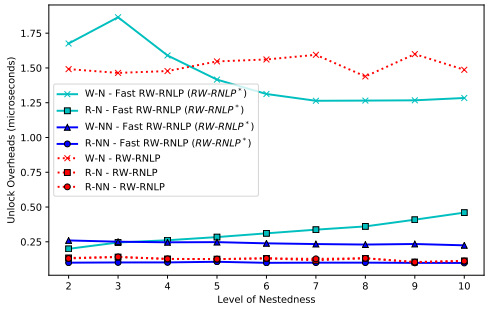



(c) Blocking.

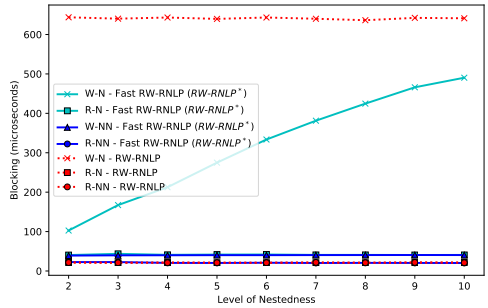
Figure 475: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

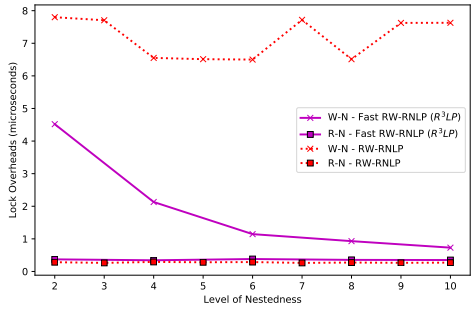



(b) Unlock overhead.

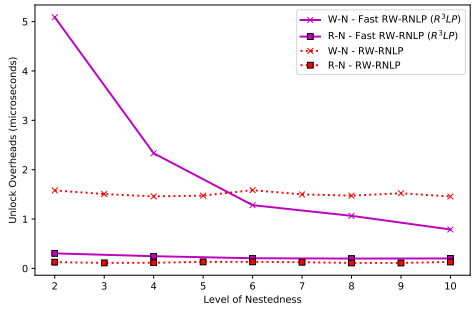



(c) Blocking.

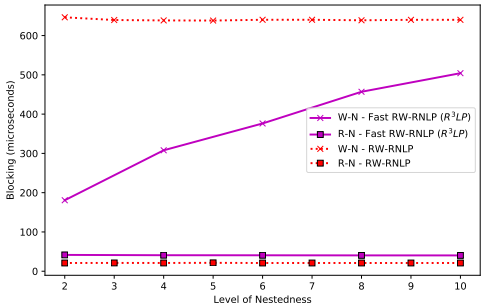
Figure 476: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

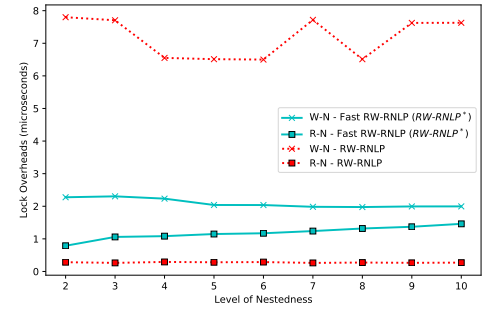



(b) Unlock overhead.

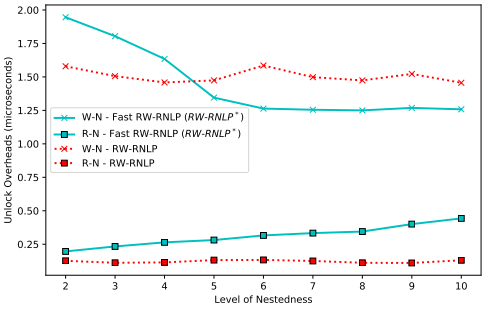



(c) Blocking.

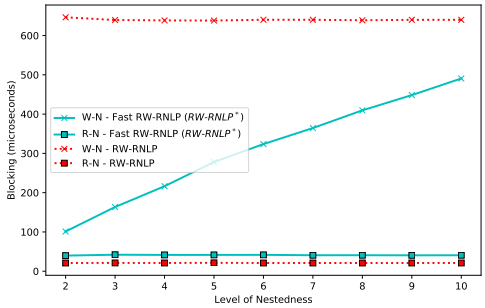
Figure 477: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

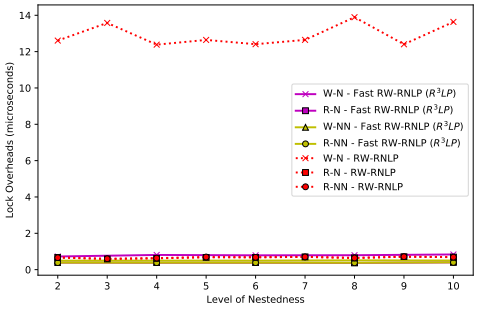



(b) Unlock overhead.

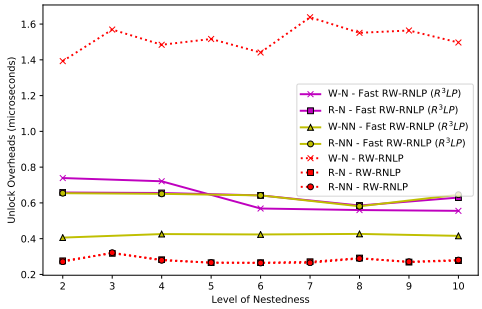



(c) Blocking.

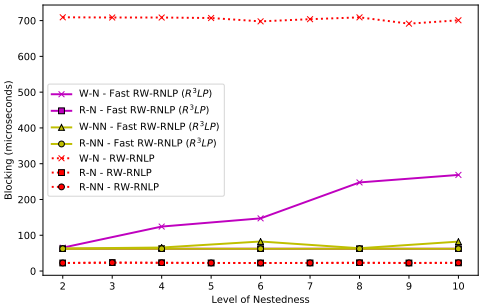
Figure 478: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

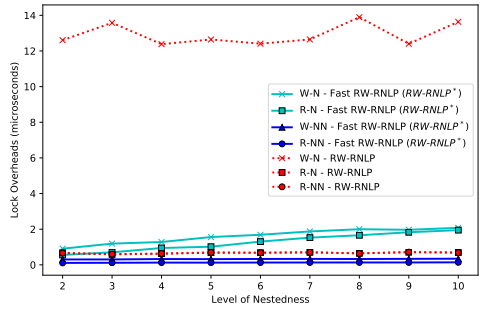



(b) Unlock overhead.

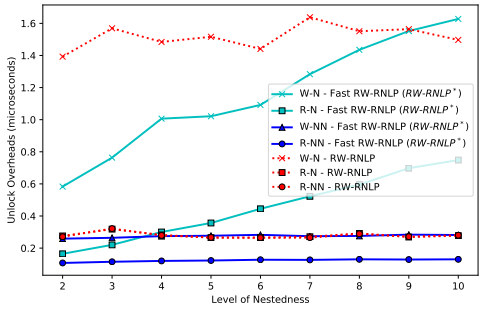



(c) Blocking.

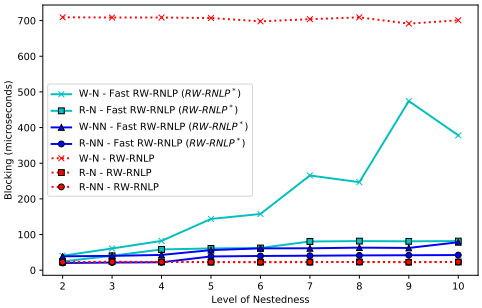
Figure 479: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

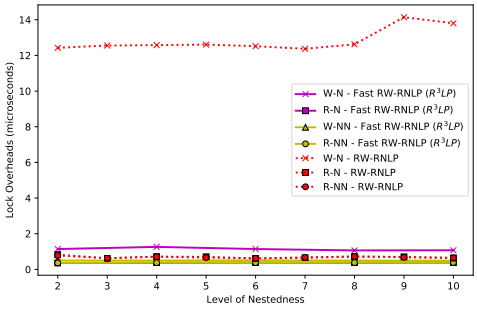



(b) Unlock overhead.

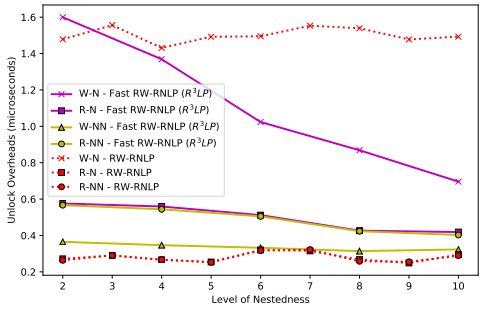



(c) Blocking.

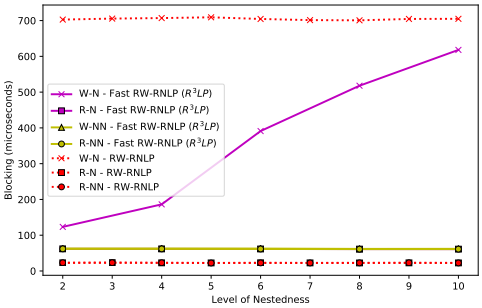
Figure 480: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

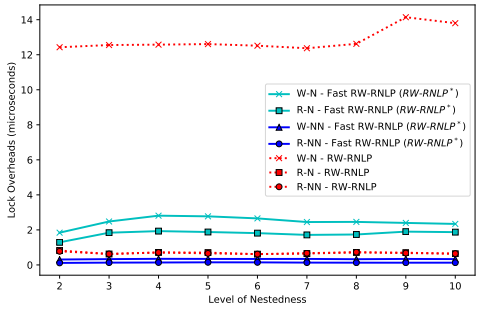



(b) Unlock overhead.

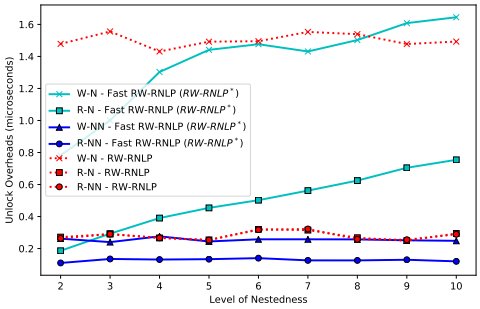



(c) Blocking.

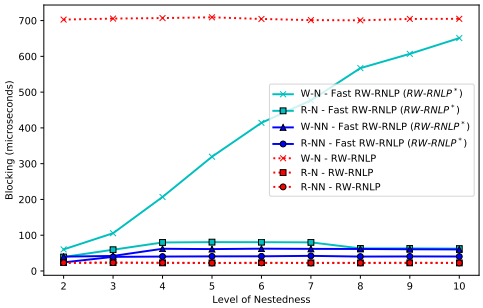
Figure 481: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

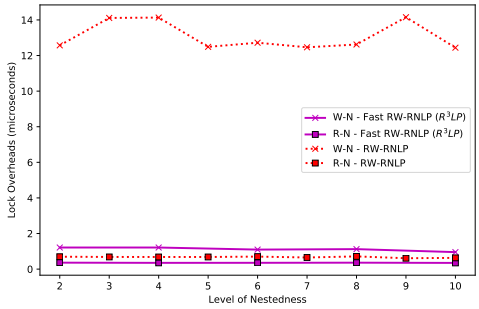



(b) Unlock overhead.

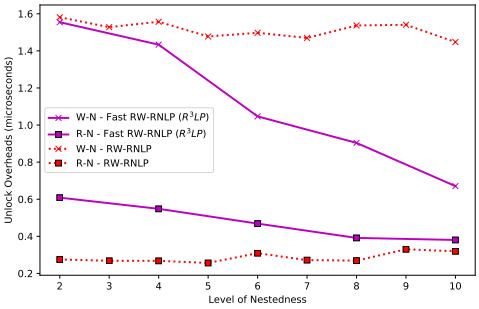



(c) Blocking.

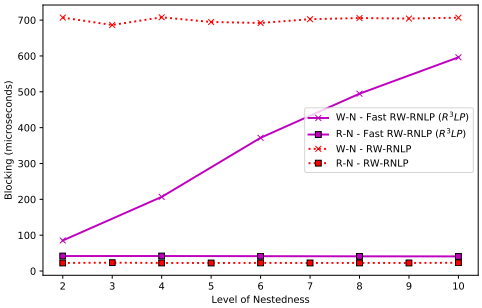
Figure 482: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

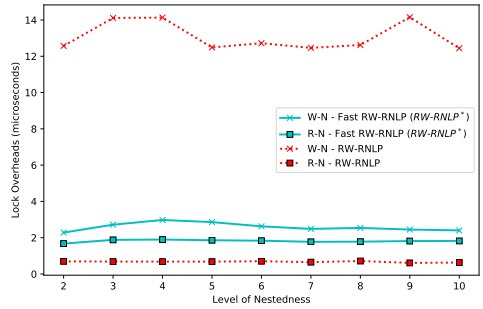



(b) Unlock overhead.

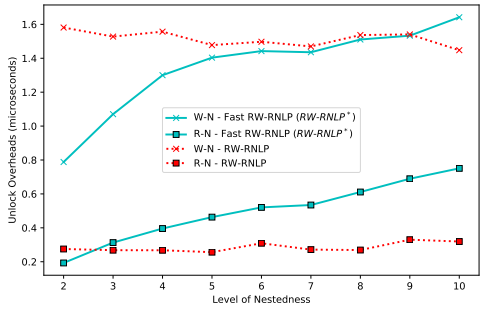



(c) Blocking.

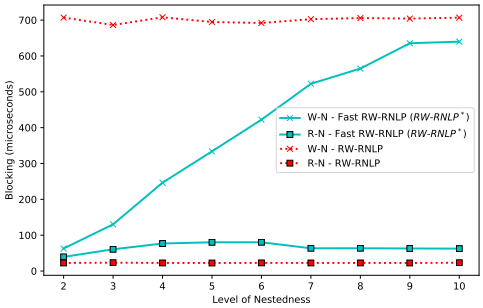
Figure 483: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

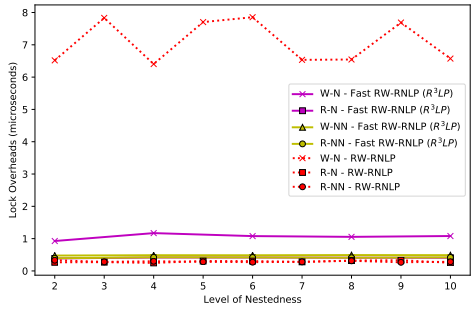



(b) Unlock overhead.

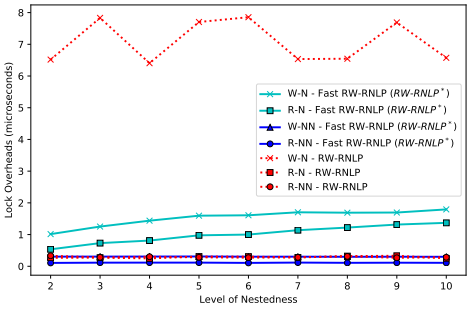



(c) Blocking.

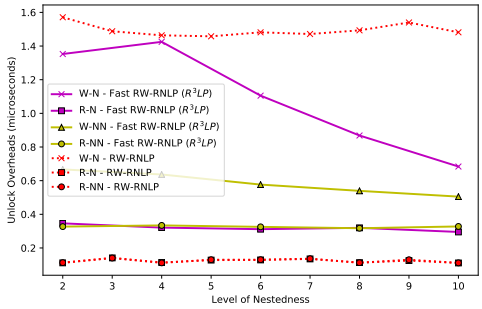
Figure 484: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.



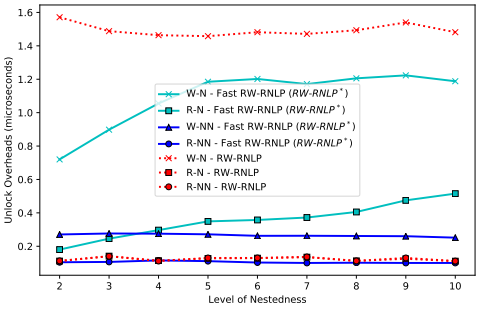

(a) Lock overhead.



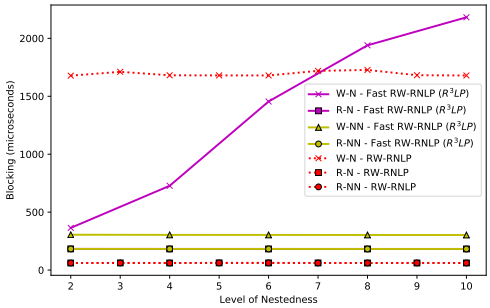

(a) Lock overhead.



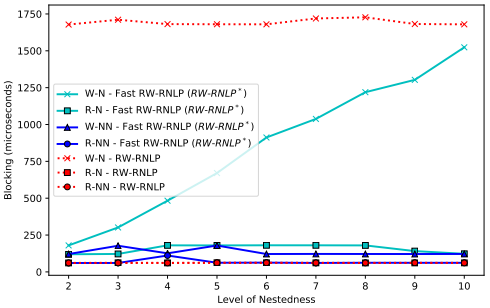

(b) Unlock overhead.




(b) Unlock overhead.



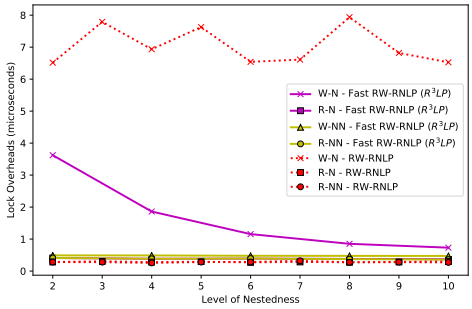

(c) Blocking.



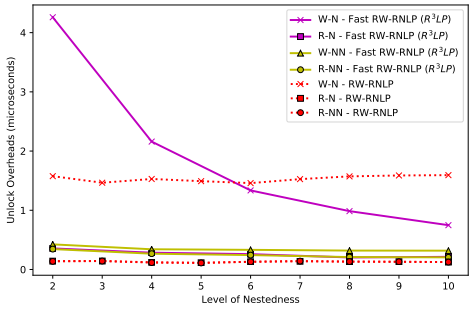

(c) Blocking.

Figure 485: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.

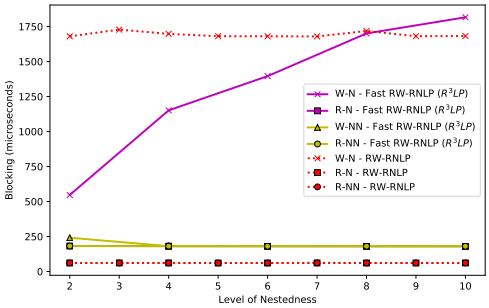
Figure 486: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

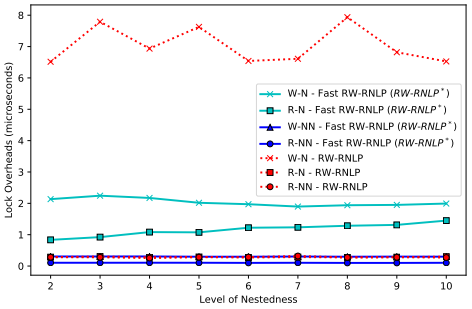



(b) Unlock overhead.

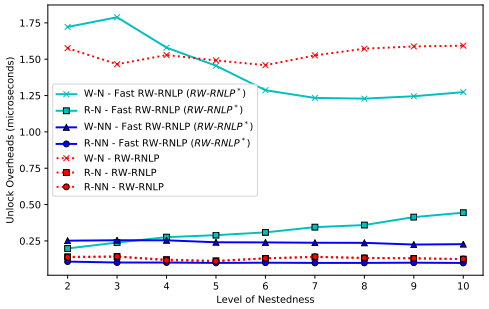



(c) Blocking.

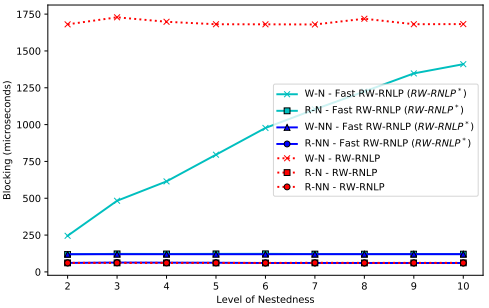
Figure 487: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

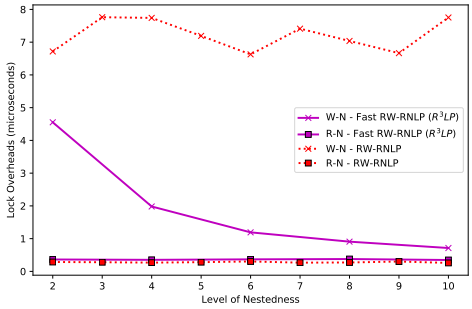



(b) Unlock overhead.

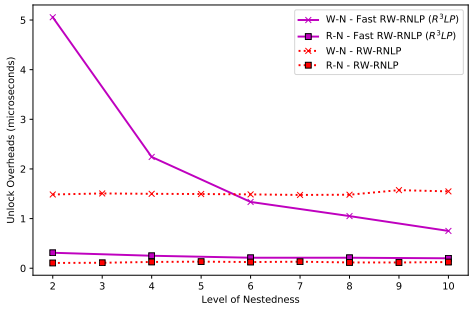



(c) Blocking.

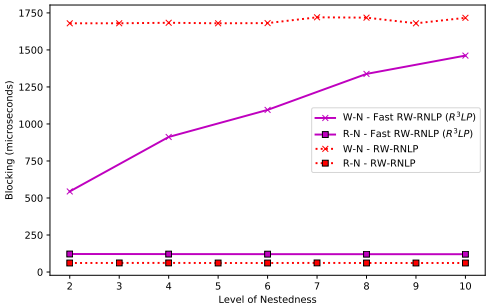
Figure 488: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

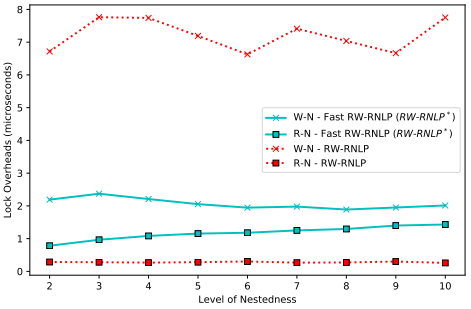



(b) Unlock overhead.

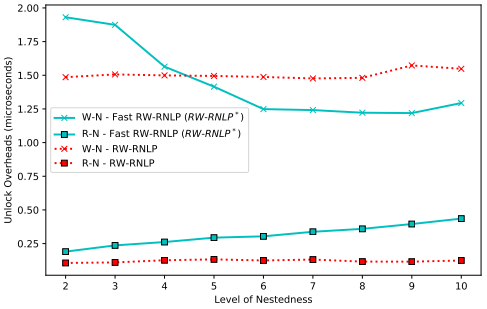



(c) Blocking.

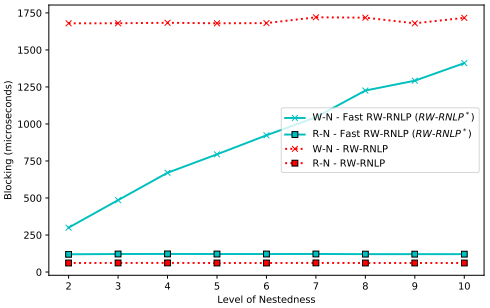
Figure 489: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

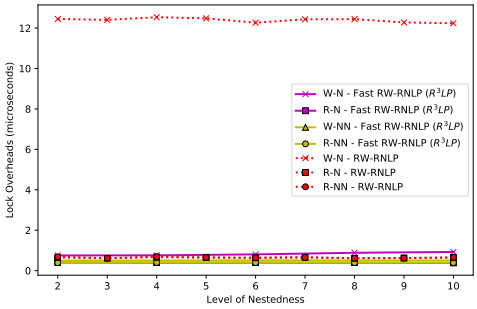



(b) Unlock overhead.

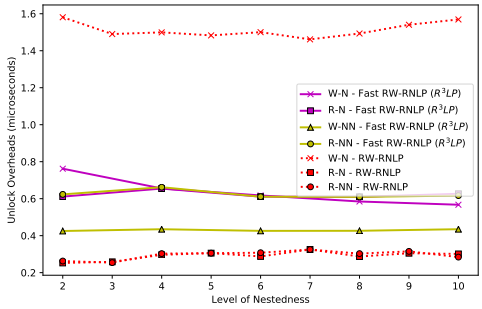



(c) Blocking.

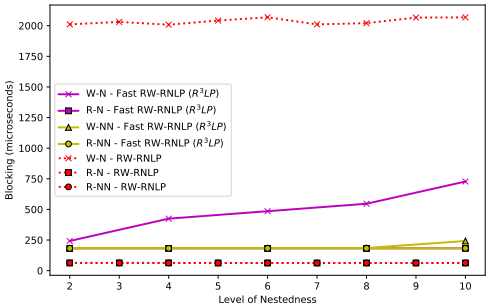
Figure 490: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

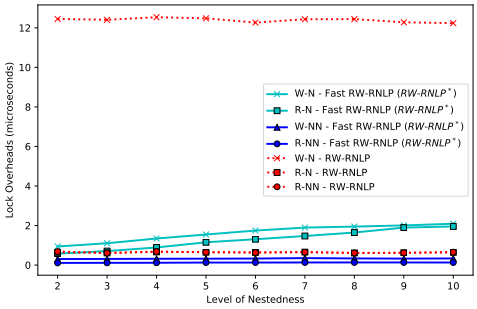



(b) Unlock overhead.

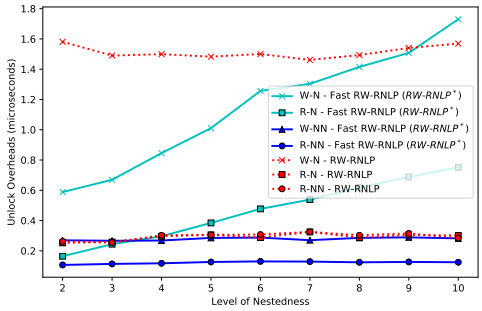



(c) Blocking.

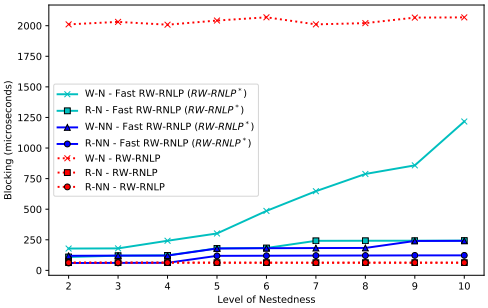
Figure 491: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

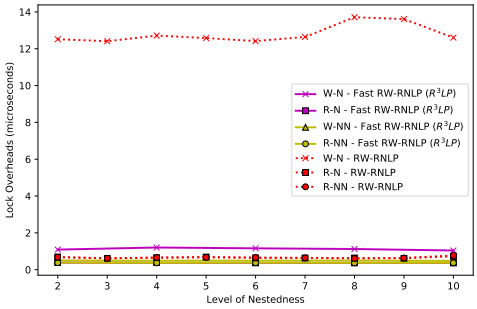



(b) Unlock overhead.

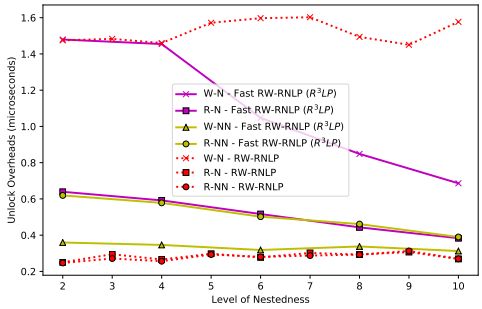



(c) Blocking.

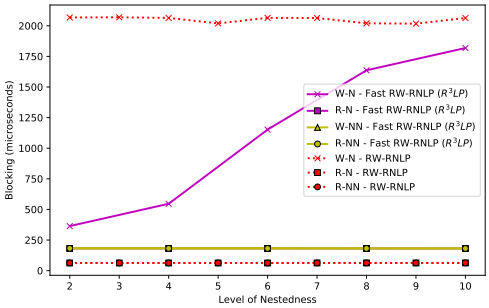
Figure 492: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

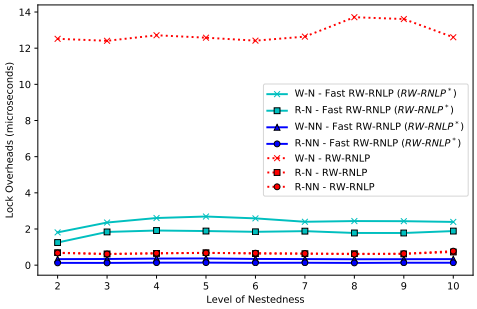



(b) Unlock overhead.

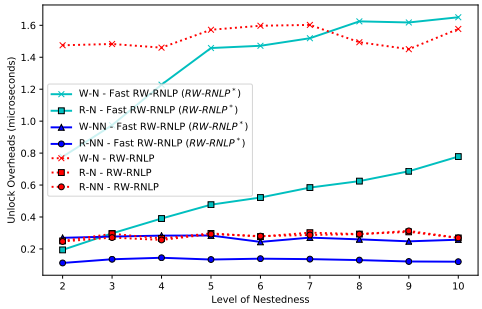



(c) Blocking.

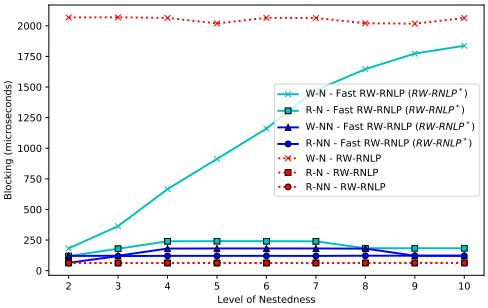
Figure 493: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

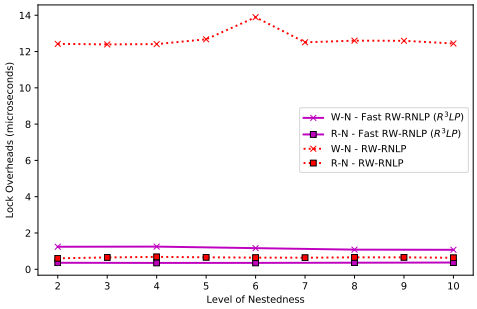



(b) Unlock overhead.

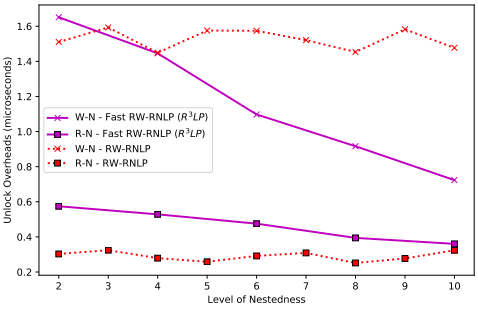



(c) Blocking.

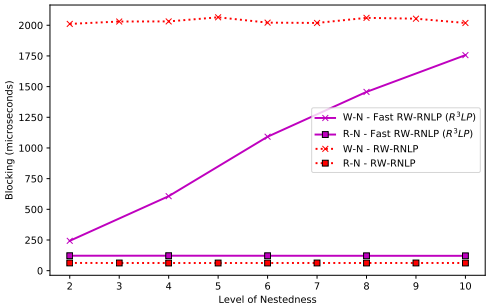
Figure 494: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

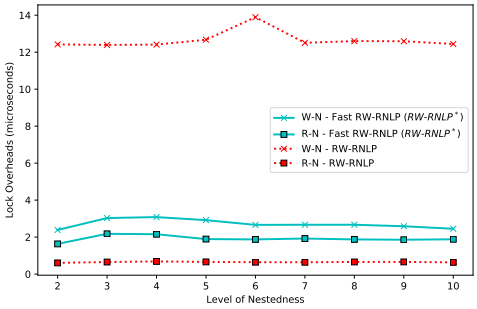



(b) Unlock overhead.

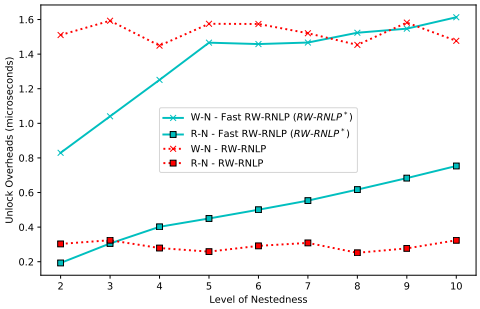



(c) Blocking.

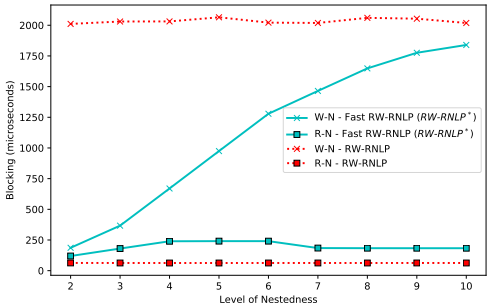
Figure 495: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

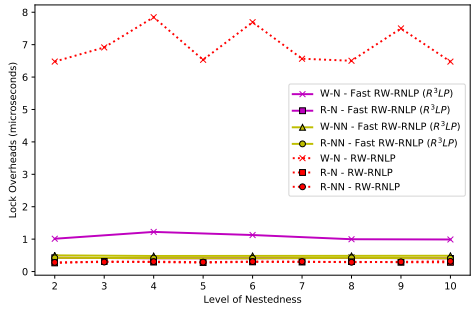



(b) Unlock overhead.

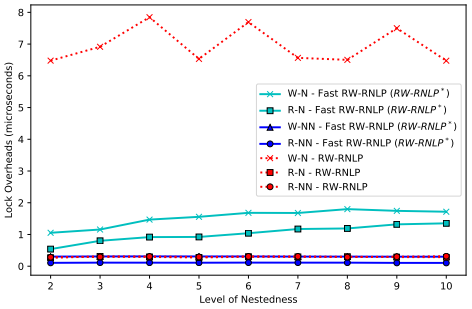



(c) Blocking.

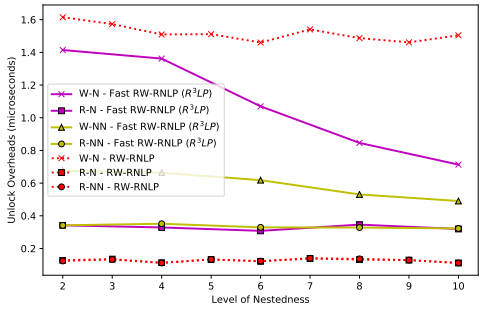
Figure 496: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.



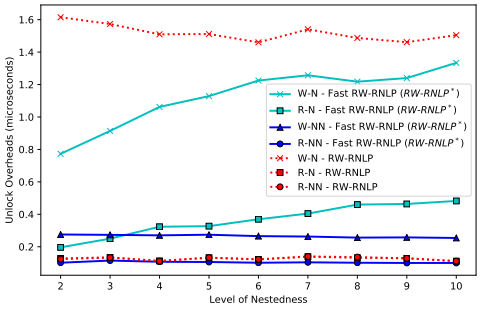

(a) Lock overhead.



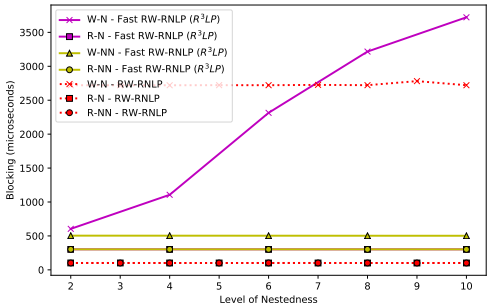

(a) Lock overhead.



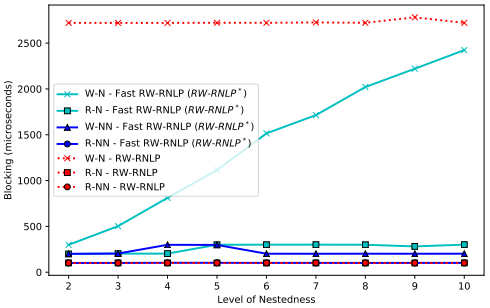

(b) Unlock overhead.




(b) Unlock overhead.



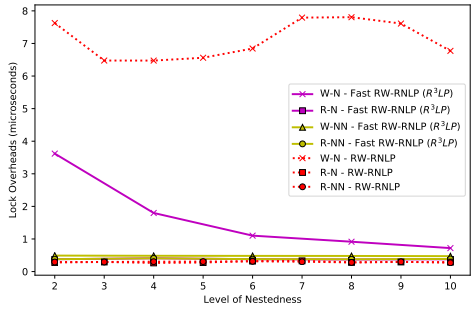

(c) Blocking.



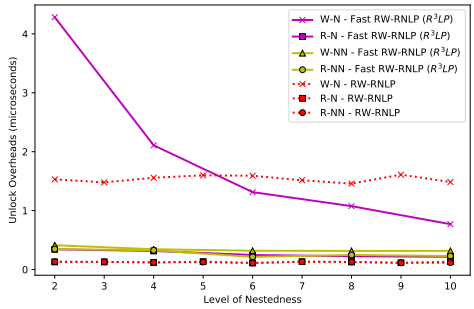

(c) Blocking.

Figure 497: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.

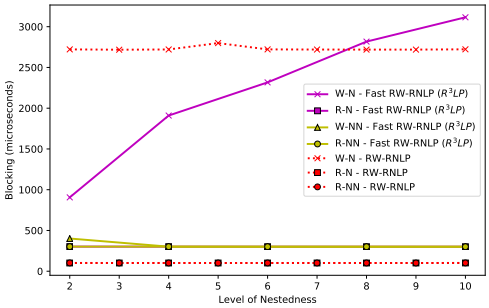
Figure 498: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

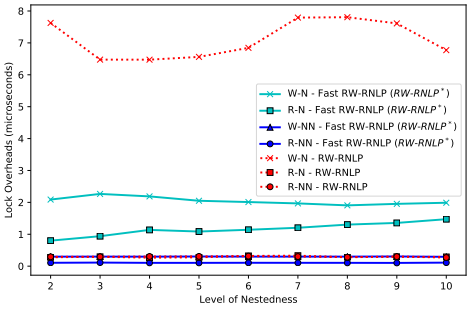



(b) Unlock overhead.

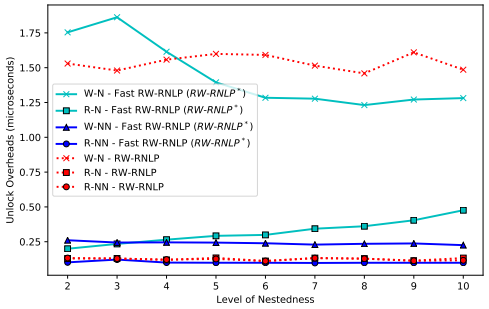



(c) Blocking.

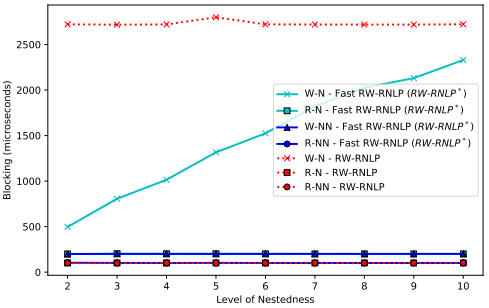
Figure 499: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

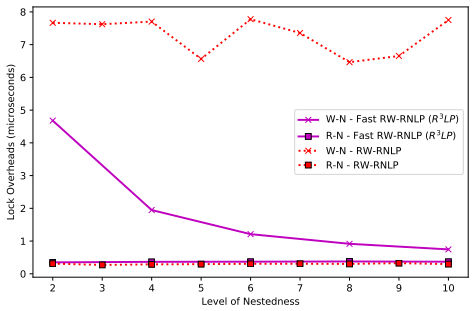



(b) Unlock overhead.

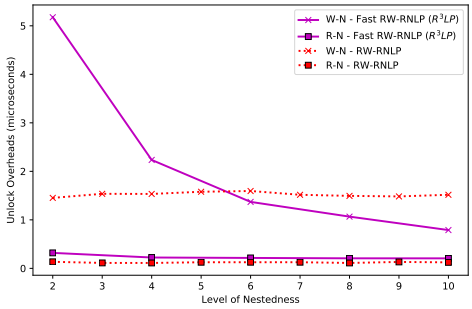



(c) Blocking.

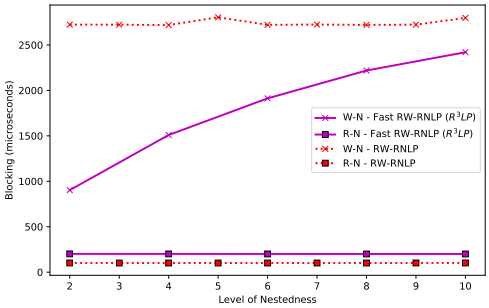
Figure 500: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

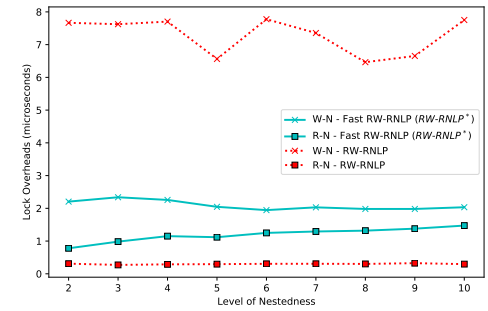



(b) Unlock overhead.

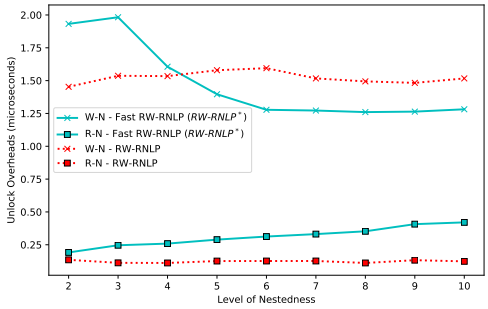



(c) Blocking.

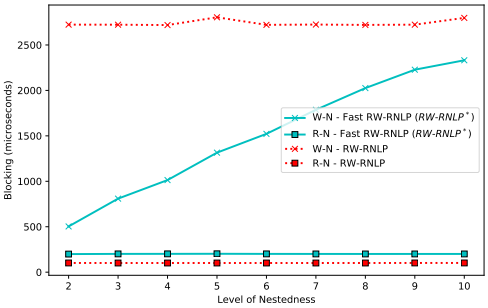
Figure 501: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

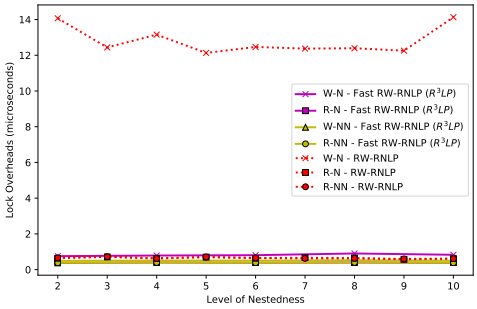



(b) Unlock overhead.

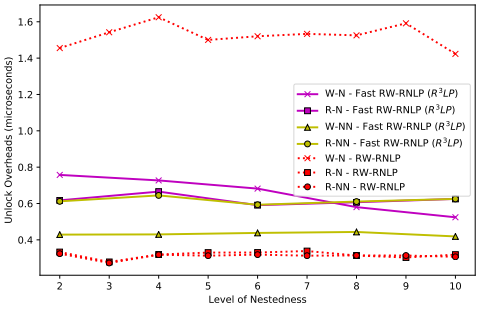



(c) Blocking.

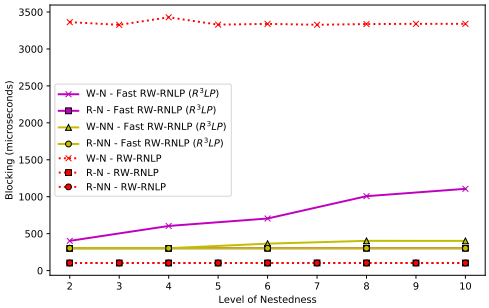
Figure 502: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

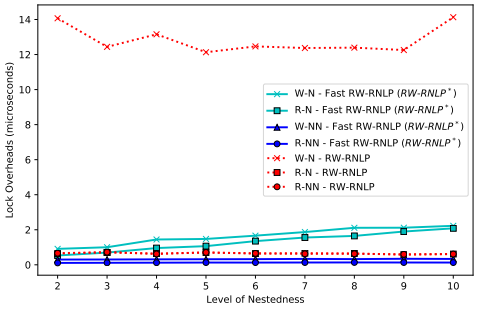



(b) Unlock overhead.

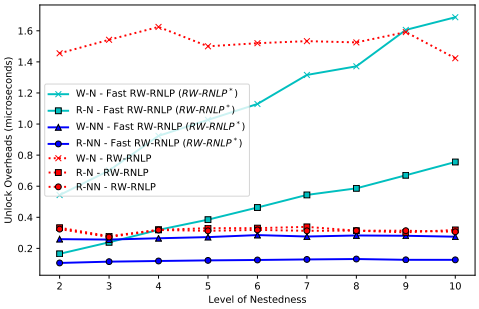



(c) Blocking.

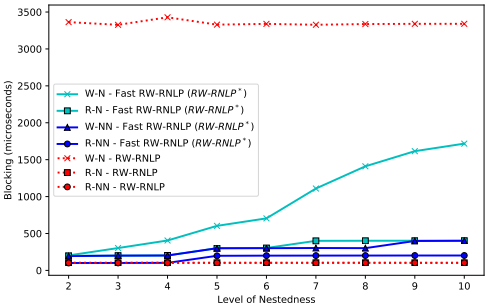
Figure 503: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

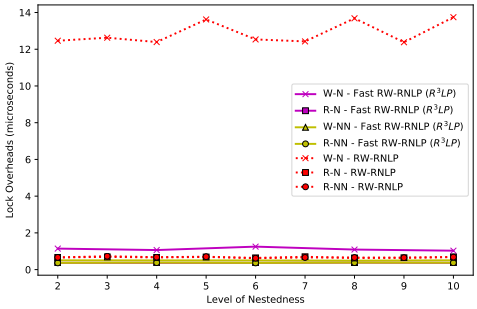



(b) Unlock overhead.

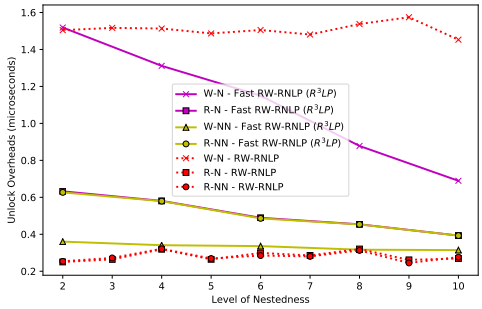



(c) Blocking.

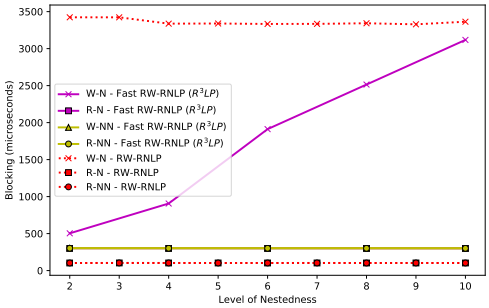
Figure 504: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

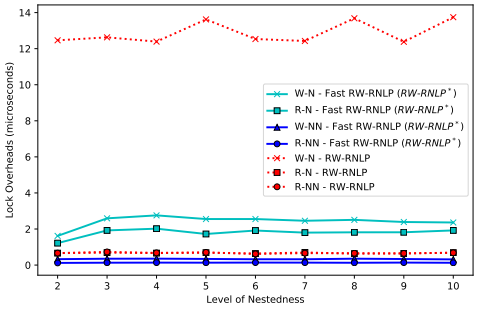



(b) Unlock overhead.

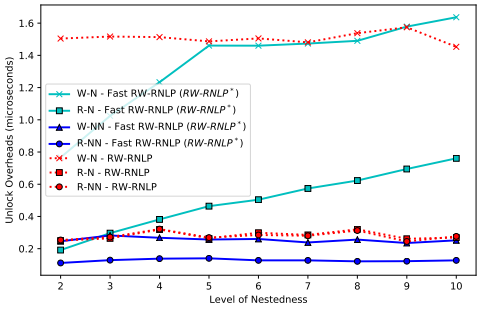



(c) Blocking.

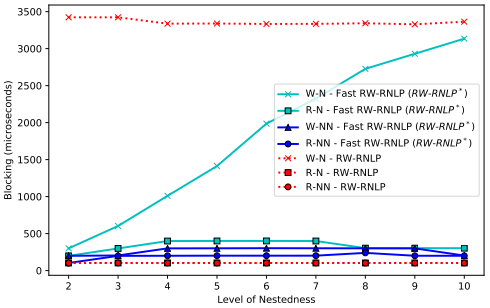
Figure 505: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

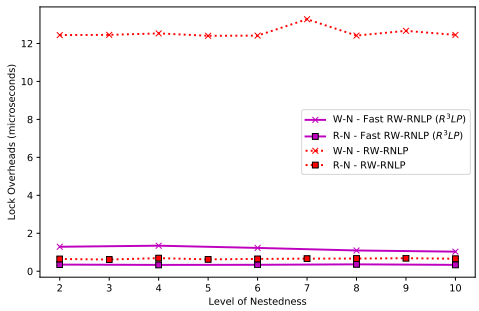



(b) Unlock overhead.

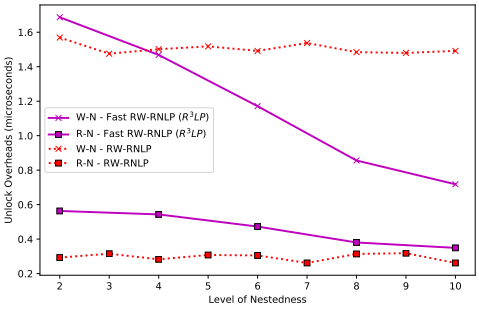



(c) Blocking.

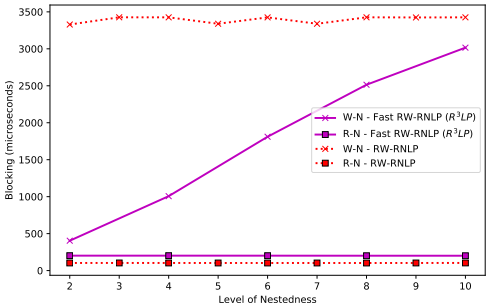
Figure 506: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

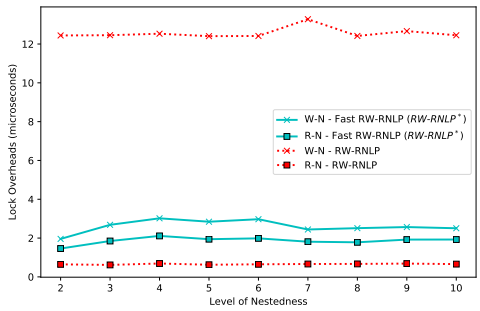



(b) Unlock overhead.

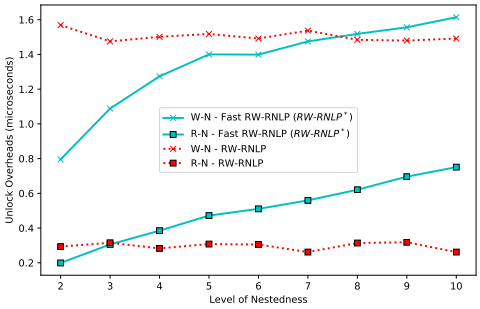



(c) Blocking.

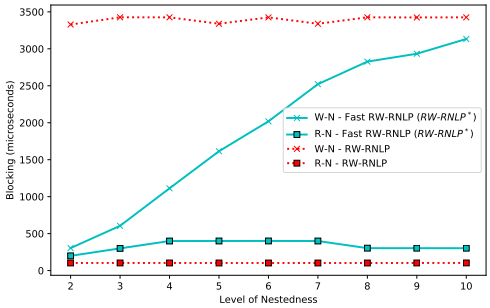
Figure 507: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

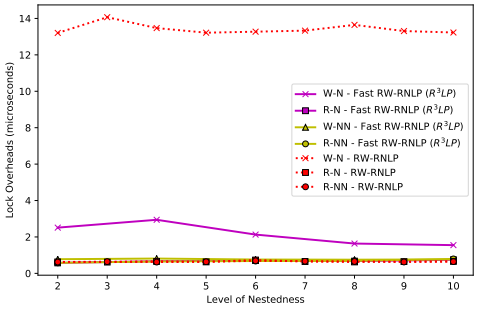



(b) Unlock overhead.

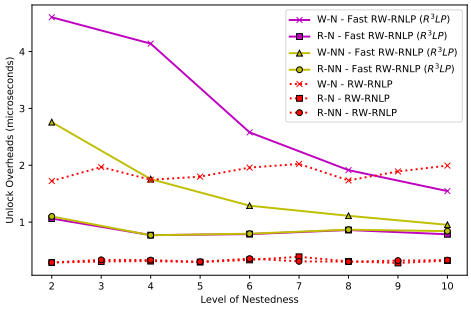



(c) Blocking.

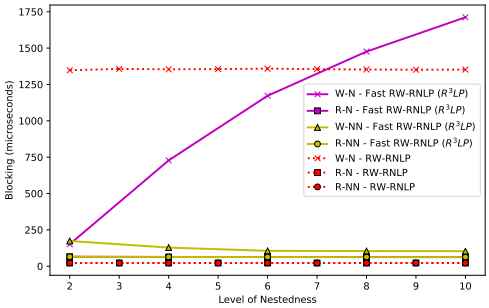
Figure 508: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 18$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

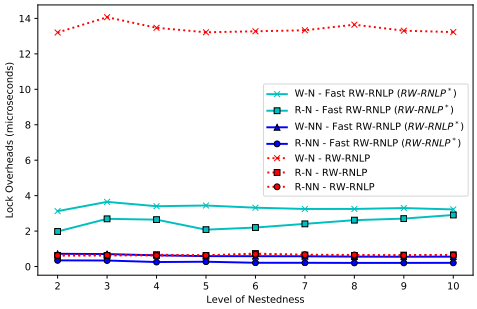



(b) Unlock overhead.

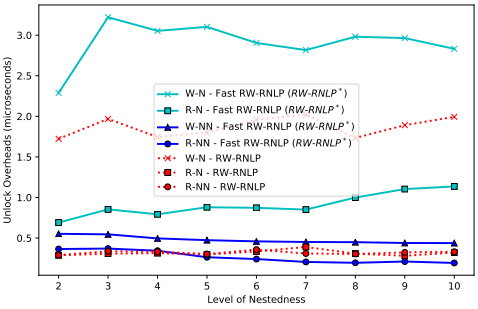



(c) Blocking.

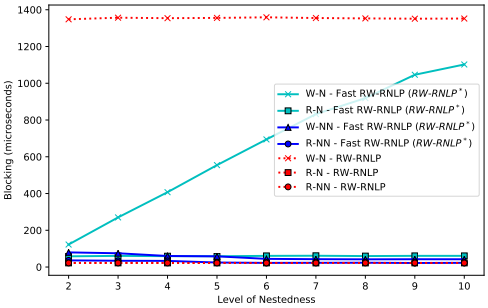
Figure 509: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

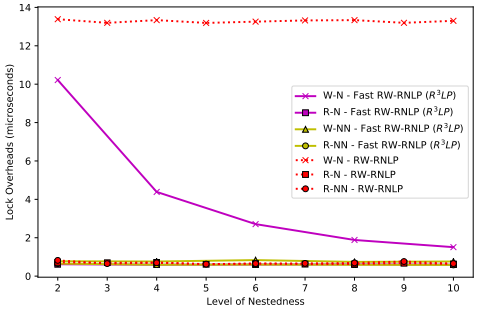



(b) Unlock overhead.

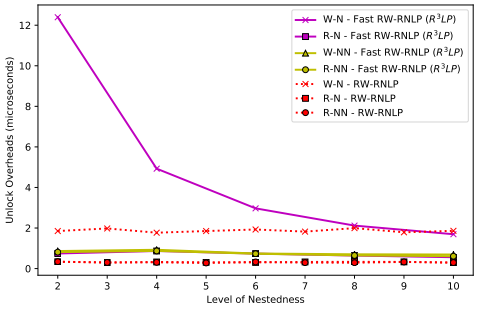



(c) Blocking.

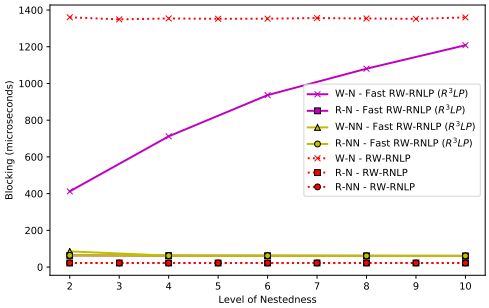
Figure 510: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

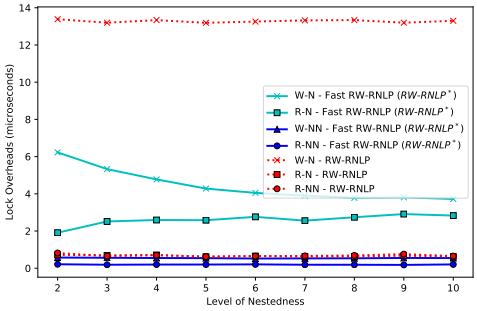



(b) Unlock overhead.

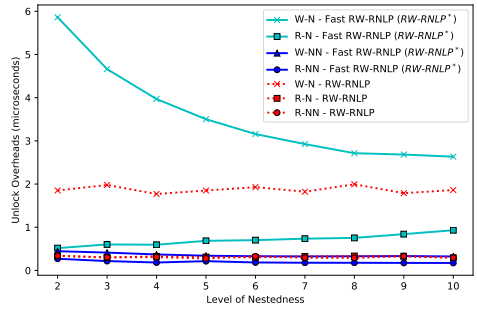



(c) Blocking.

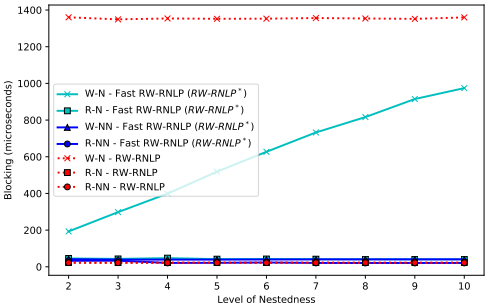
Figure 511: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

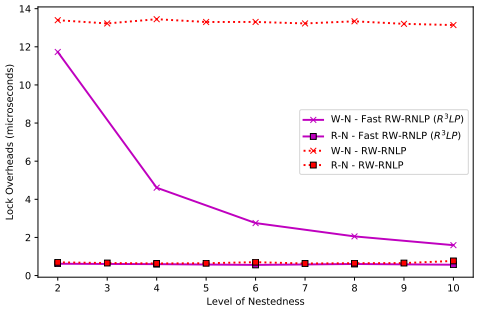



(b) Unlock overhead.

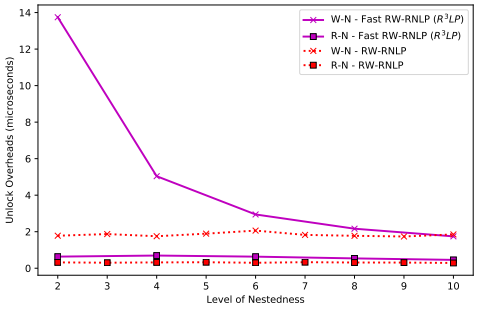



(c) Blocking.

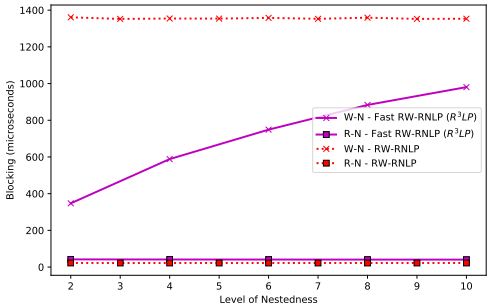
Figure 512: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

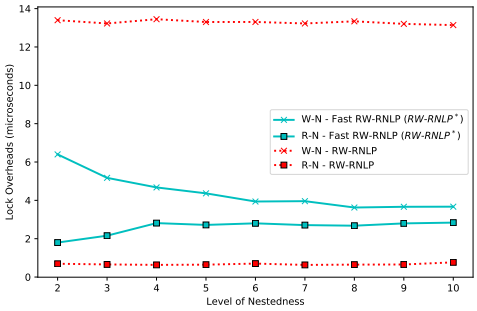



(b) Unlock overhead.

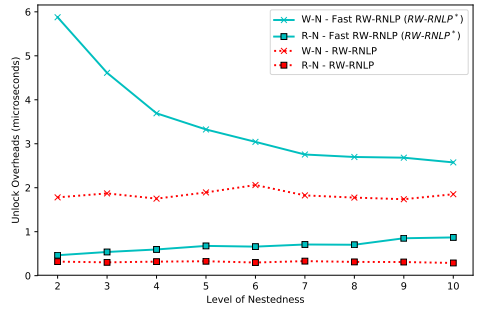



(c) Blocking.

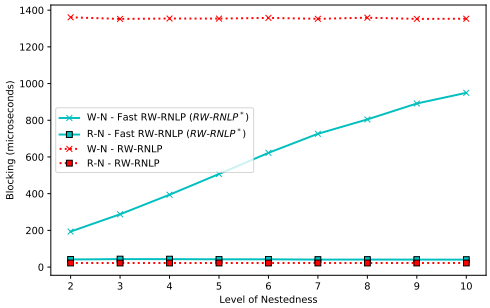
Figure 513: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

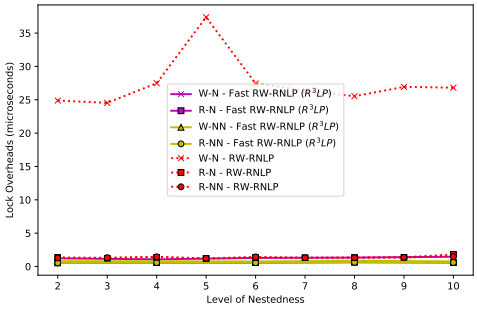



(b) Unlock overhead.

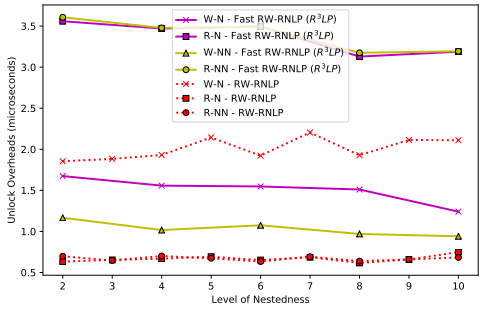



(c) Blocking.

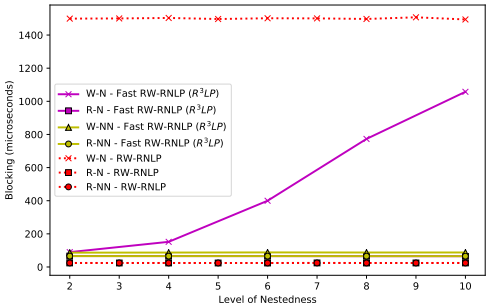
Figure 514: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

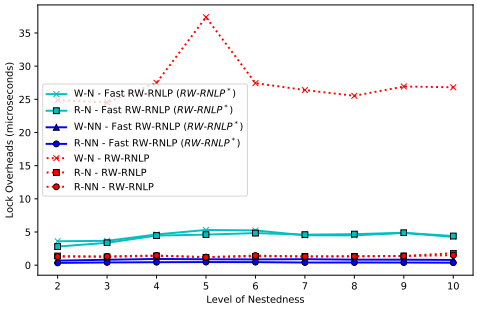



(b) Unlock overhead.

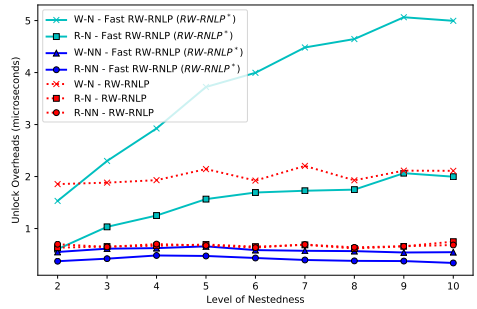



(c) Blocking.

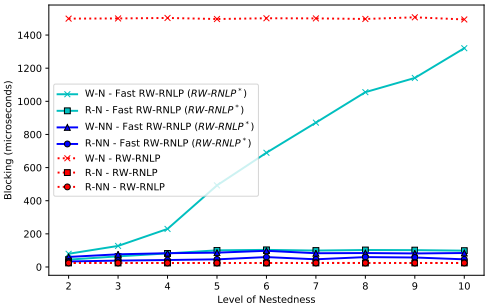
Figure 515: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

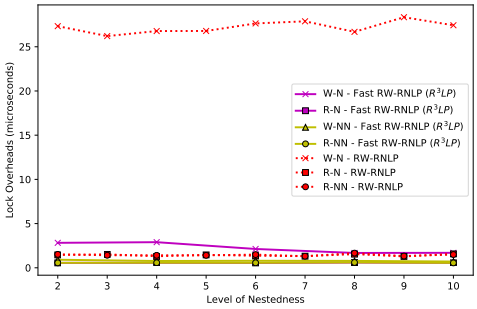



(b) Unlock overhead.

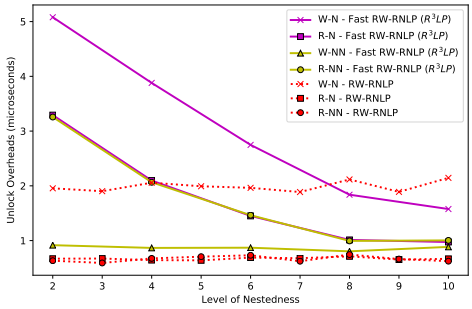



(c) Blocking.

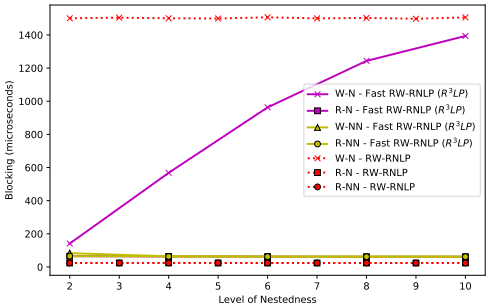
Figure 516: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

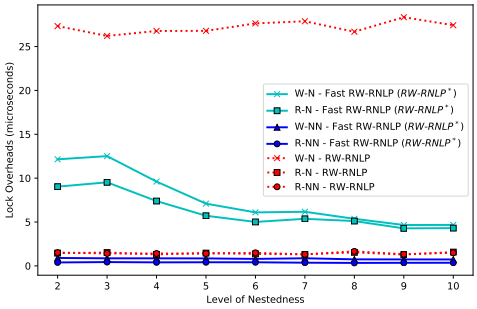



(b) Unlock overhead.

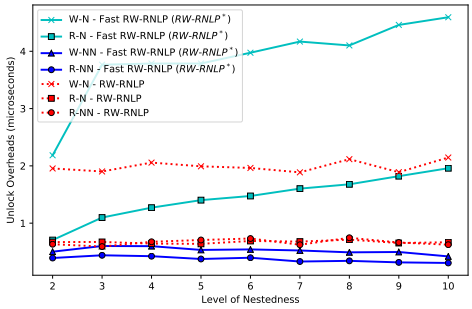



(c) Blocking.

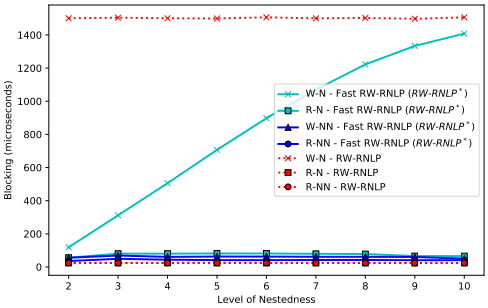
Figure 517: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

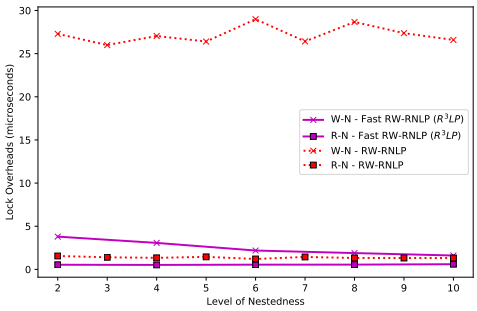



(b) Unlock overhead.

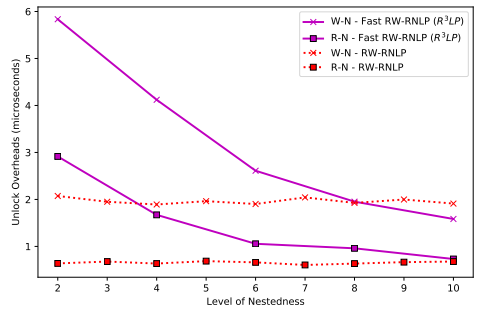



(c) Blocking.

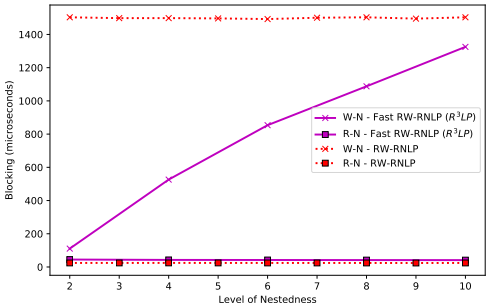
Figure 518: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

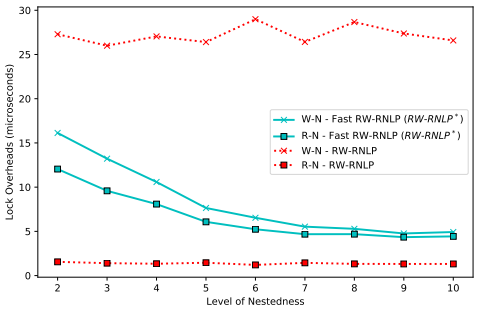



(b) Unlock overhead.

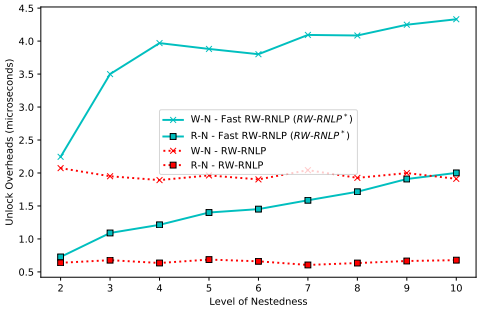



(c) Blocking.

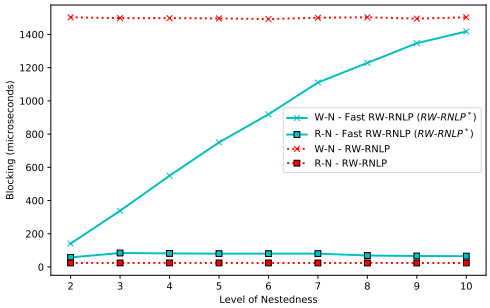
Figure 519: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

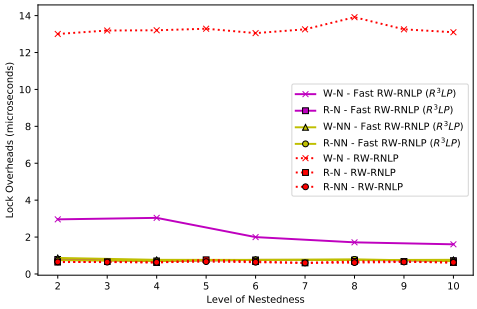



(b) Unlock overhead.

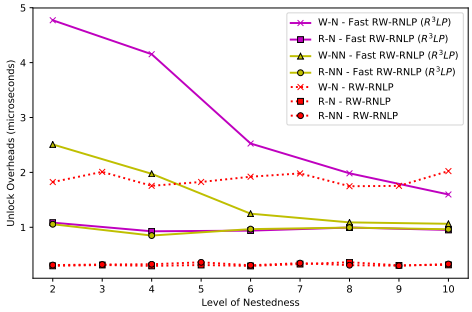



(c) Blocking.

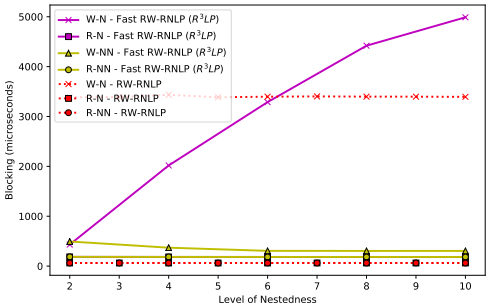
Figure 520: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 20\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

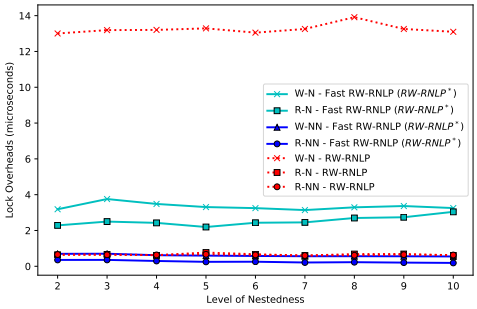



(b) Unlock overhead.

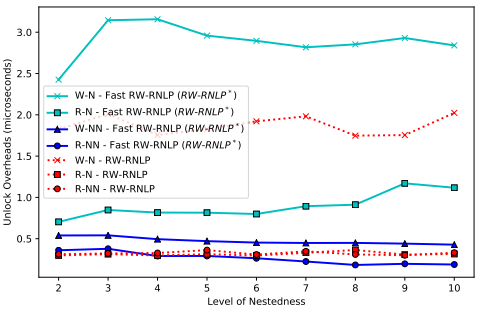



(c) Blocking.

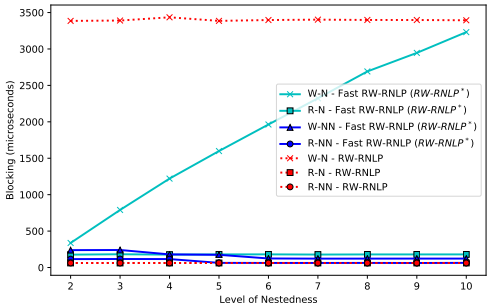
Figure 521: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

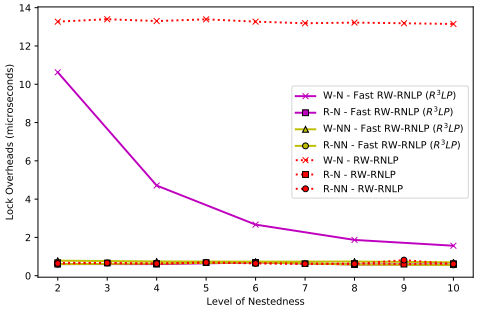



(b) Unlock overhead.

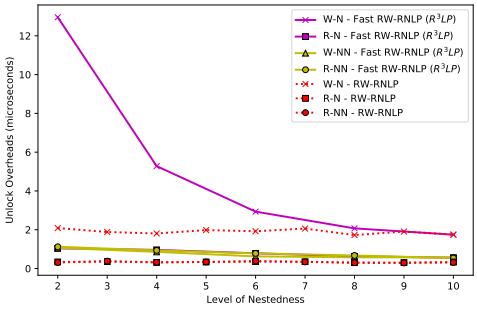



(c) Blocking.

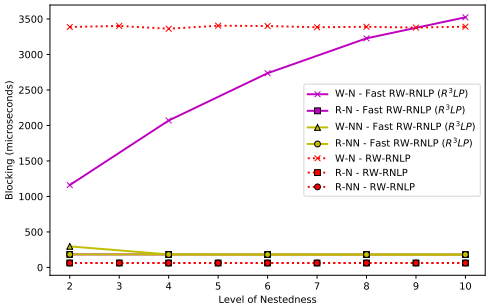
Figure 522: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

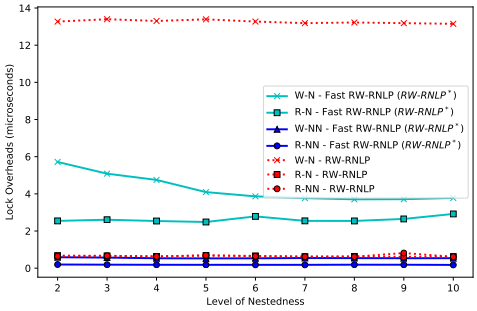



(b) Unlock overhead.

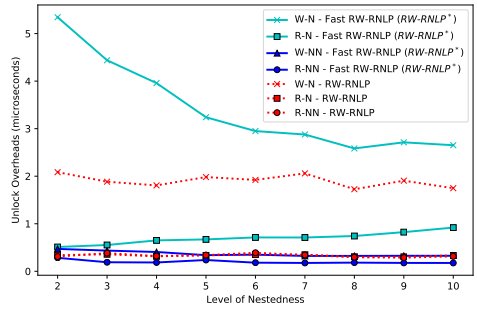



(c) Blocking.

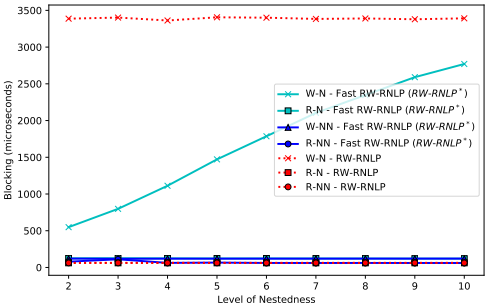
Figure 523: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

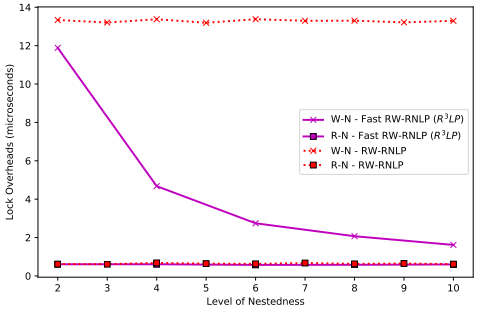



(b) Unlock overhead.

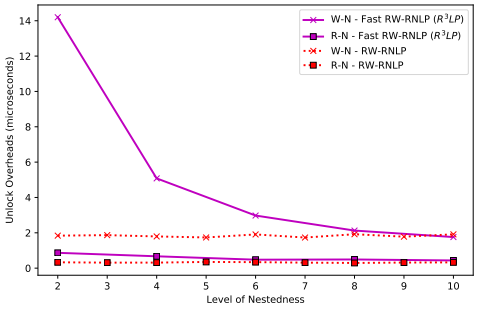



(c) Blocking.

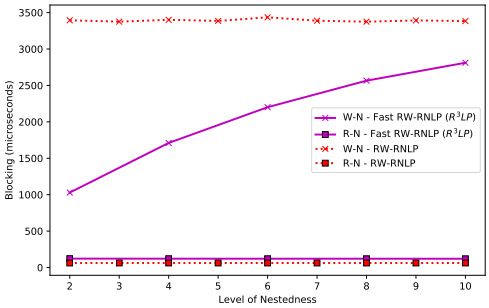
Figure 524: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

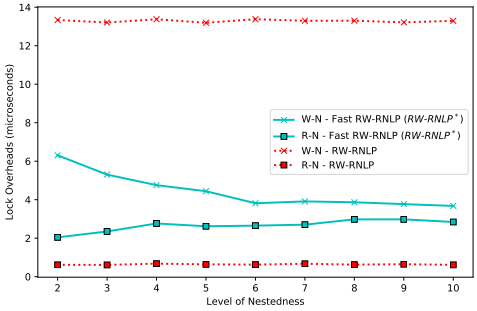



(b) Unlock overhead.

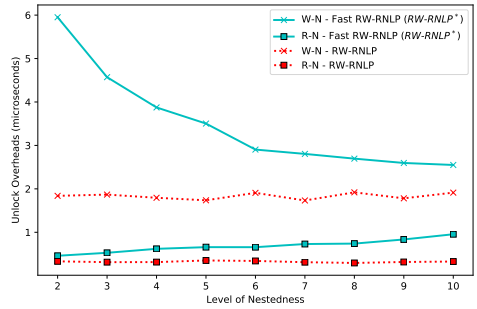



(c) Blocking.

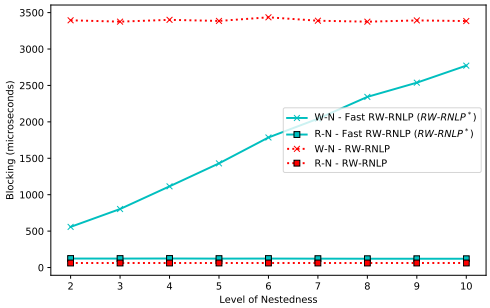
Figure 525: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

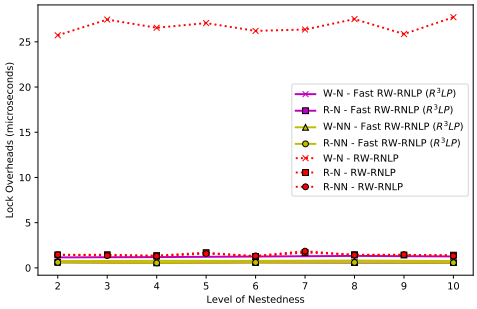



(b) Unlock overhead.

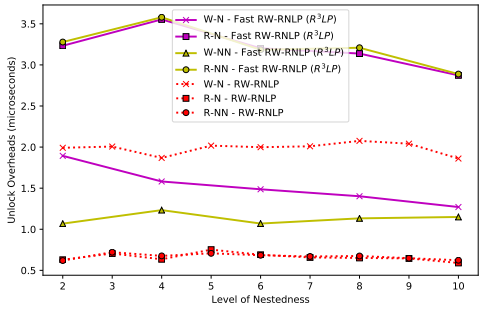



(c) Blocking.

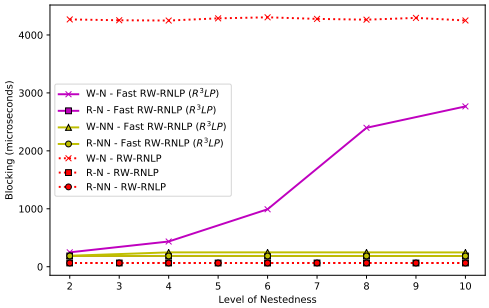
Figure 526: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

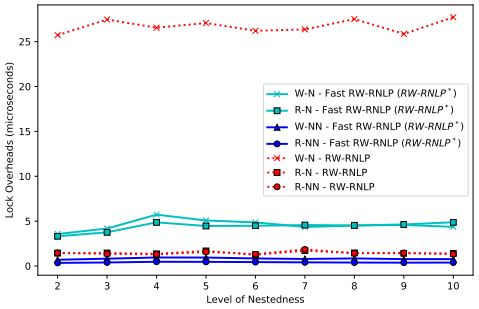



(b) Unlock overhead.

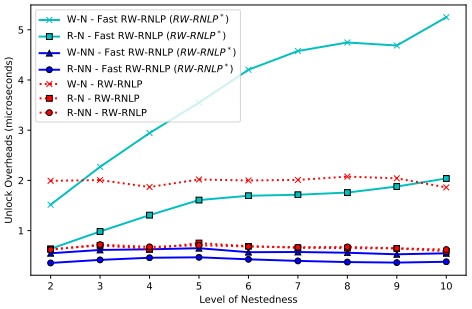



(c) Blocking.

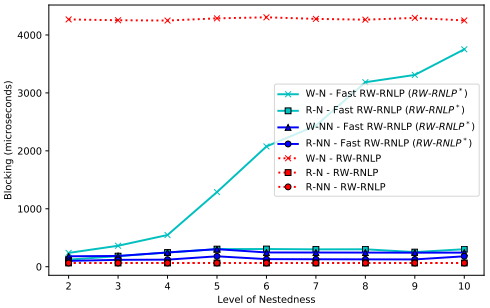
Figure 527: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

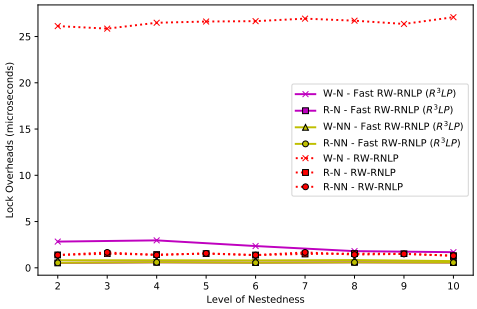



(b) Unlock overhead.

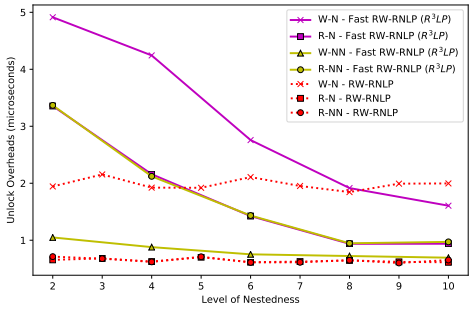



(c) Blocking.

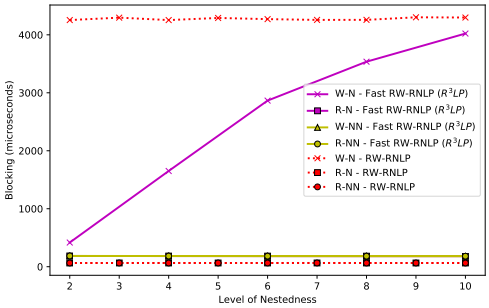
Figure 528: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

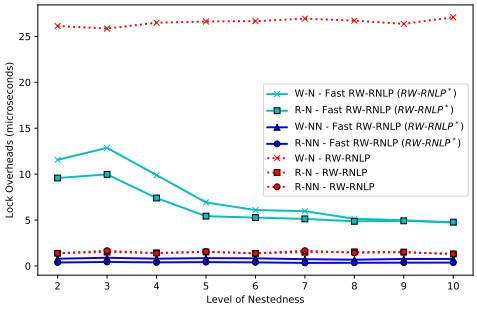



(b) Unlock overhead.

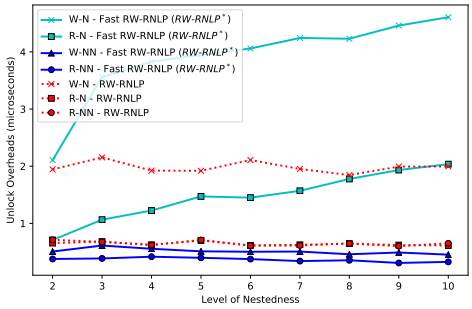



(c) Blocking.

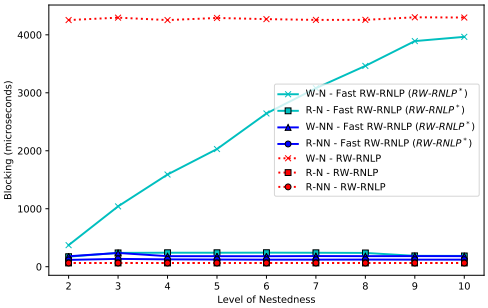
Figure 529: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

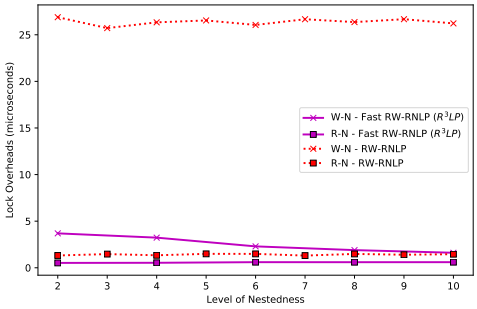



(b) Unlock overhead.

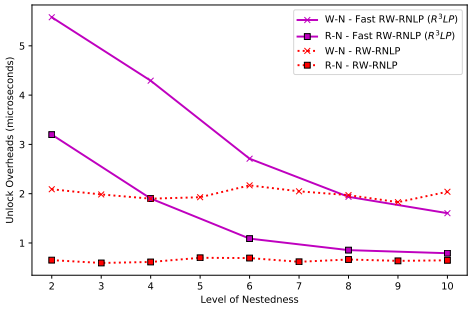



(c) Blocking.

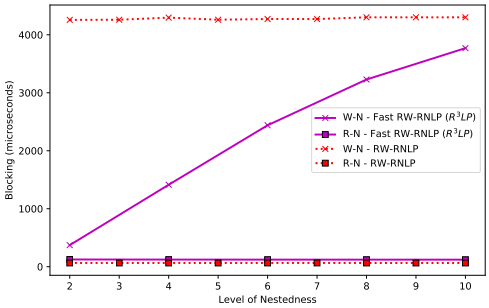
Figure 530: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

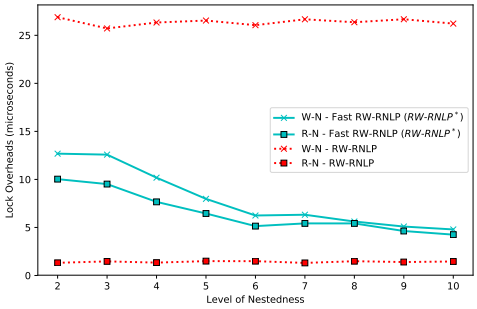



(b) Unlock overhead.

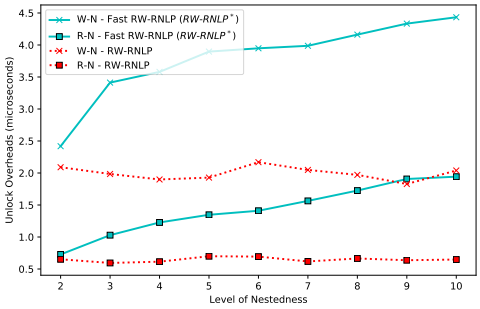



(c) Blocking.

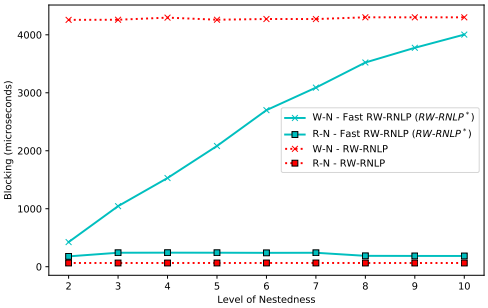
Figure 531: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

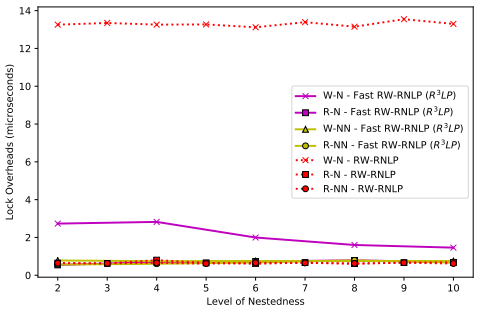



(b) Unlock overhead.

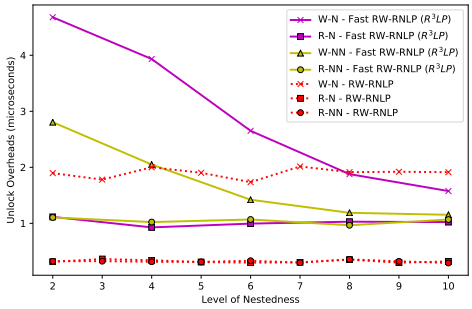



(c) Blocking.

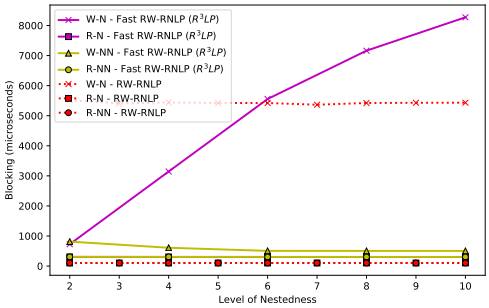
Figure 532: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 60\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

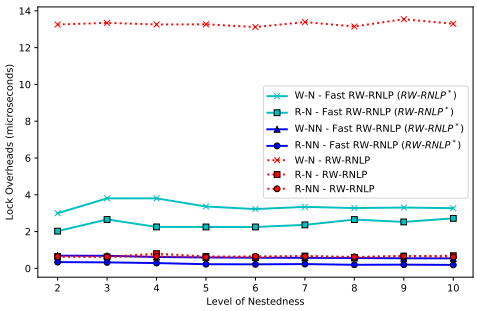



(b) Unlock overhead.

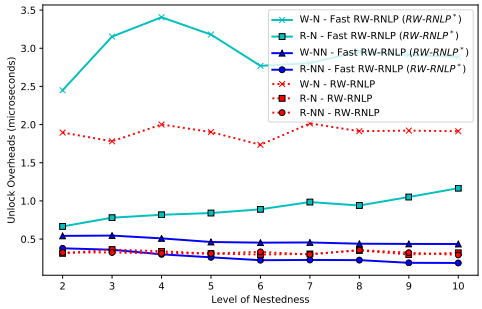



(c) Blocking.

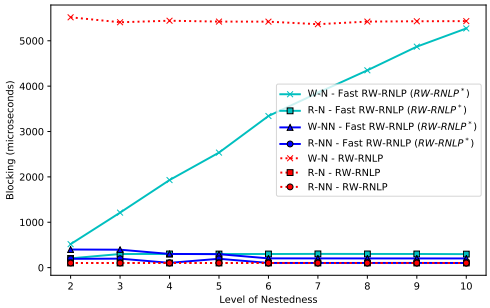
Figure 533: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

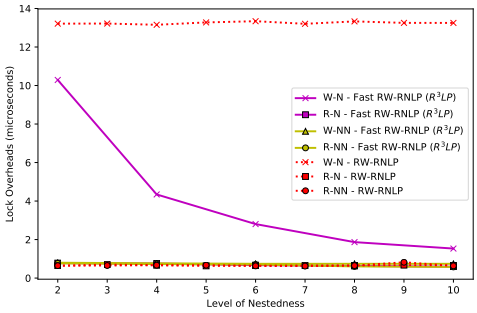



(b) Unlock overhead.

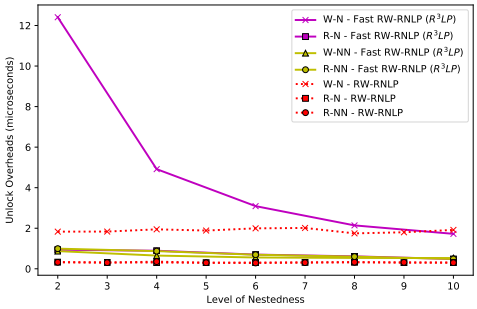



(c) Blocking.

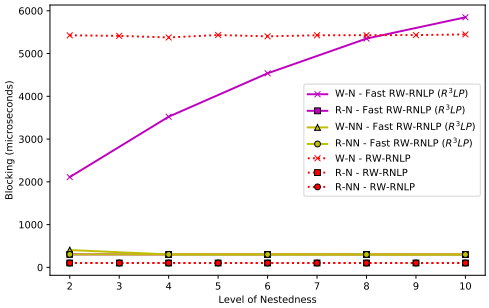
Figure 534: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

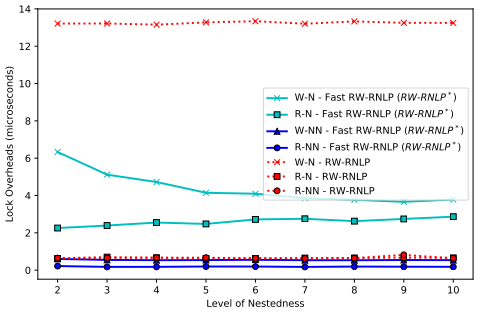



(b) Unlock overhead.

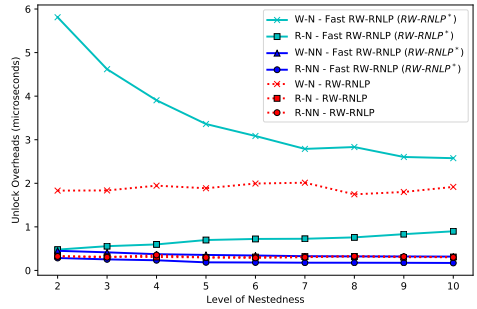



(c) Blocking.

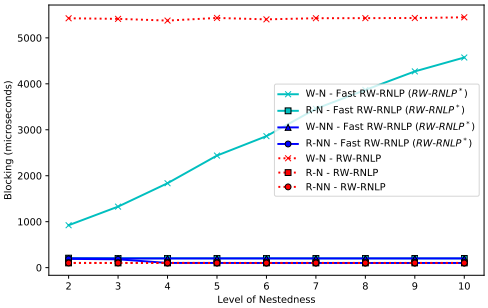
Figure 535: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

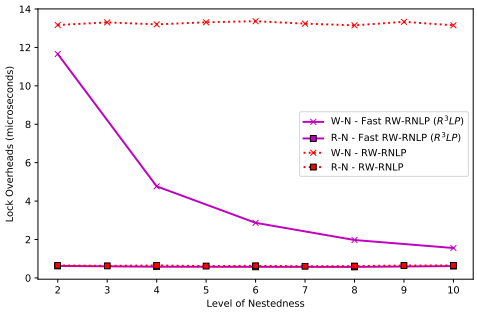



(b) Unlock overhead.

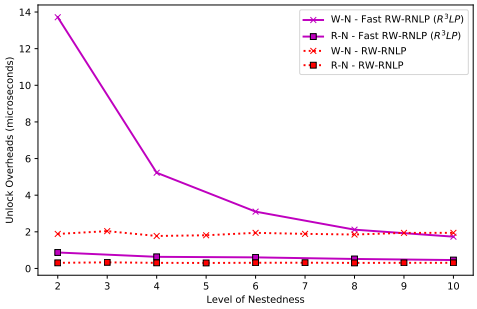



(c) Blocking.

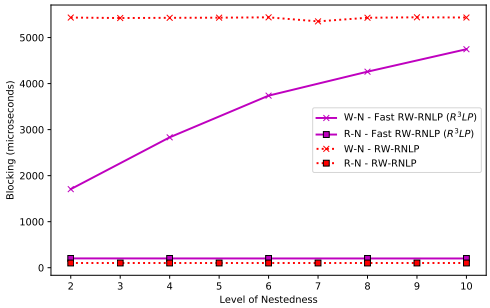
Figure 536: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

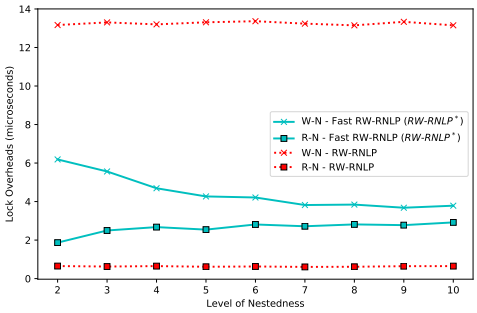



(b) Unlock overhead.

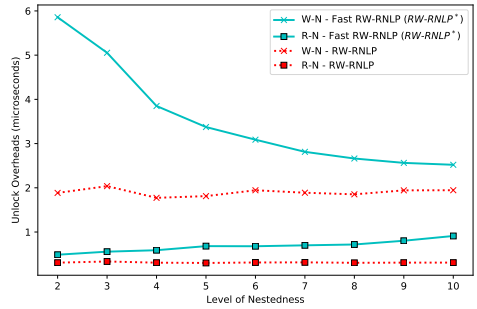



(c) Blocking.

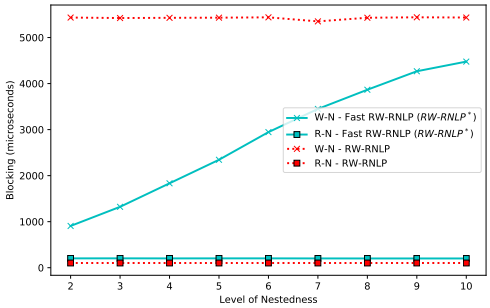
Figure 537: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

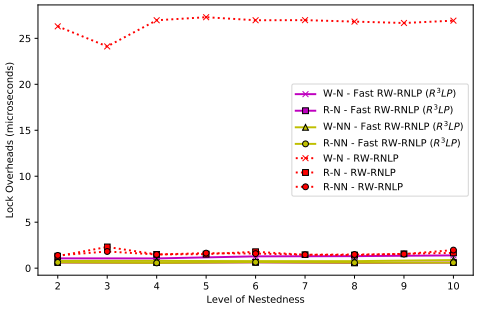



(b) Unlock overhead.

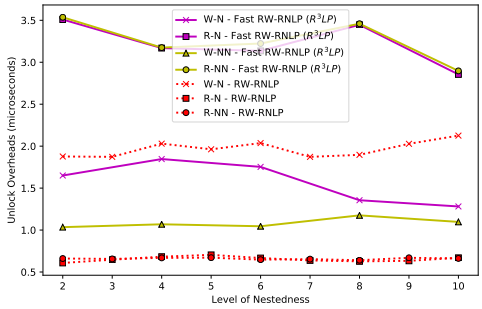



(c) Blocking.

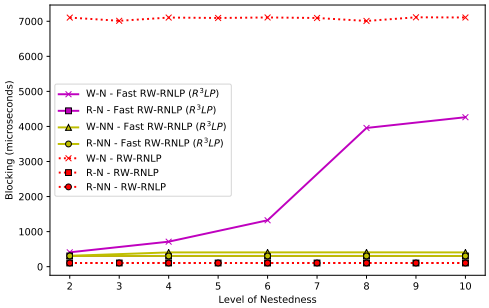
Figure 538: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.2 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

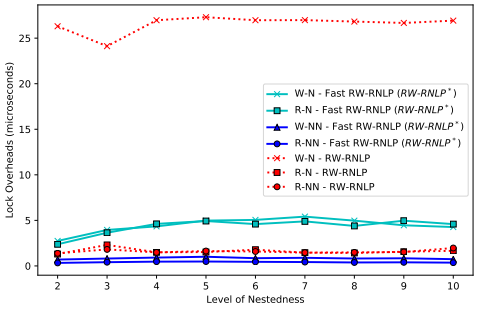



(b) Unlock overhead.

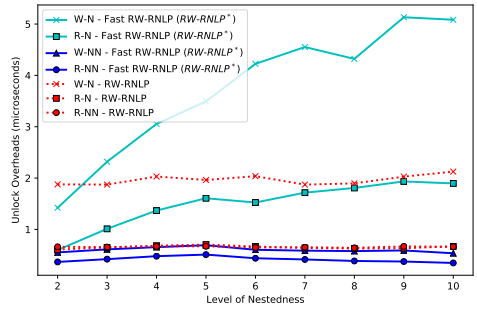



(c) Blocking.

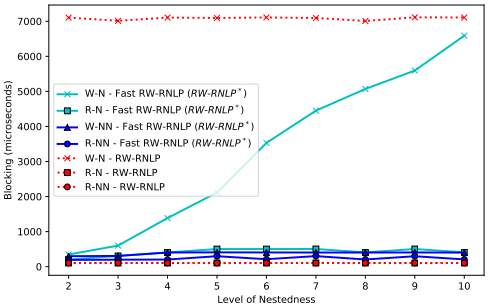
Figure 539: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

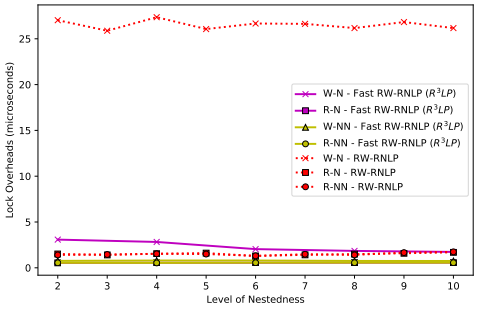



(b) Unlock overhead.

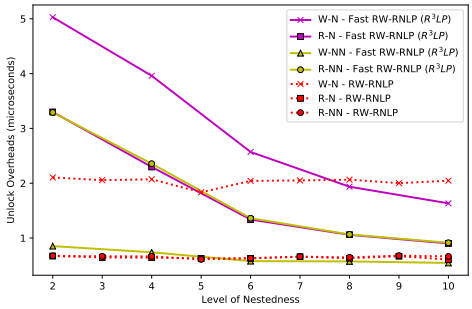



(c) Blocking.

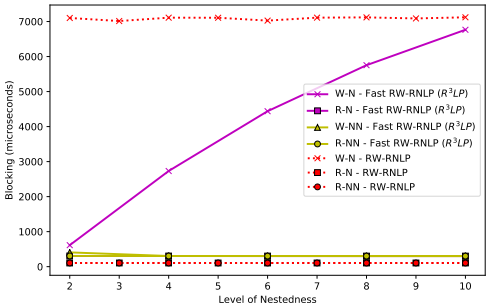
Figure 540: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.2. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

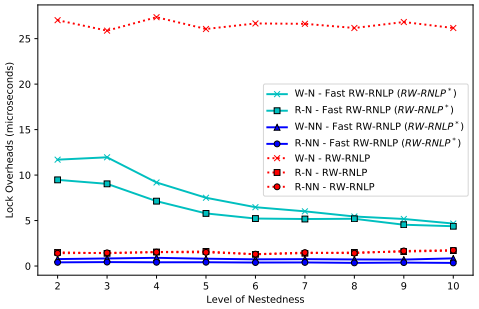



(b) Unlock overhead.

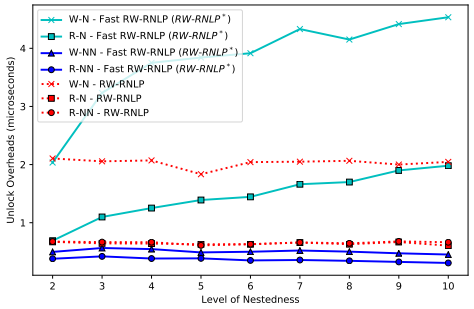



(c) Blocking.

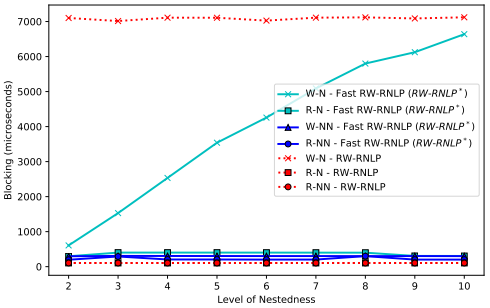
Figure 541: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

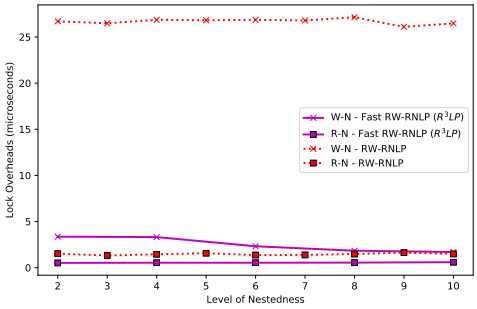



(b) Unlock overhead.

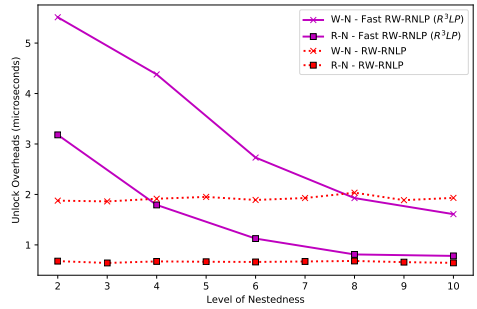



(c) Blocking.

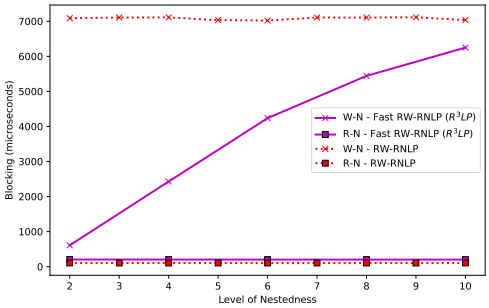
Figure 542: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested and non-nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 0.8. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.

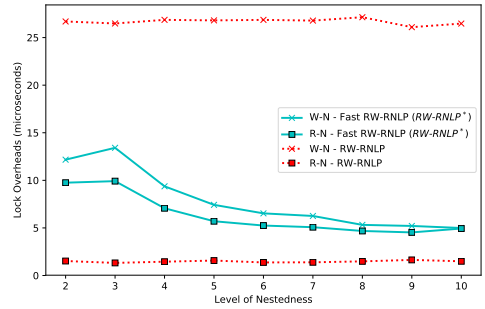



(b) Unlock overhead.

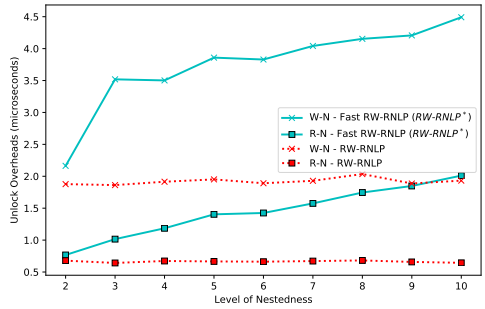



(c) Blocking.

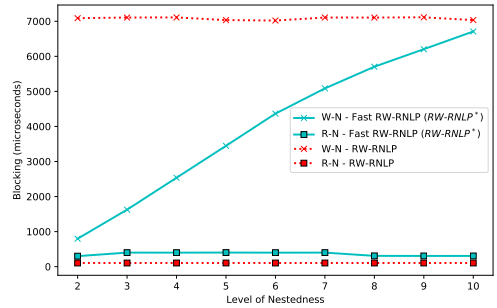
Figure 543: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the R$^3$LP. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.




(a) Lock overhead.




(b) Unlock overhead.




(c) Blocking.

Figure 544: **(a)** Lock and **(b)** unlock overheads and **(c)** blocking for nested read and write requests under the RW-RNLP and the fast RW-RNLP with the RW-RNLP*. Here, for each request $\mathcal{R}_i$, $m = 36$, $L_i = 100\mu$s, $n_r = 64$, and $|D_i|$ is as shown for nested requests. Each request was randomly chosen to be a read (as opposed to a write) with probability 0.8 and to be a nested request with probability 1. Due to write expansion, $|D_i|$ was inflated to 64 for all write requests under the RW-RNLP, as read requests can access any resource.

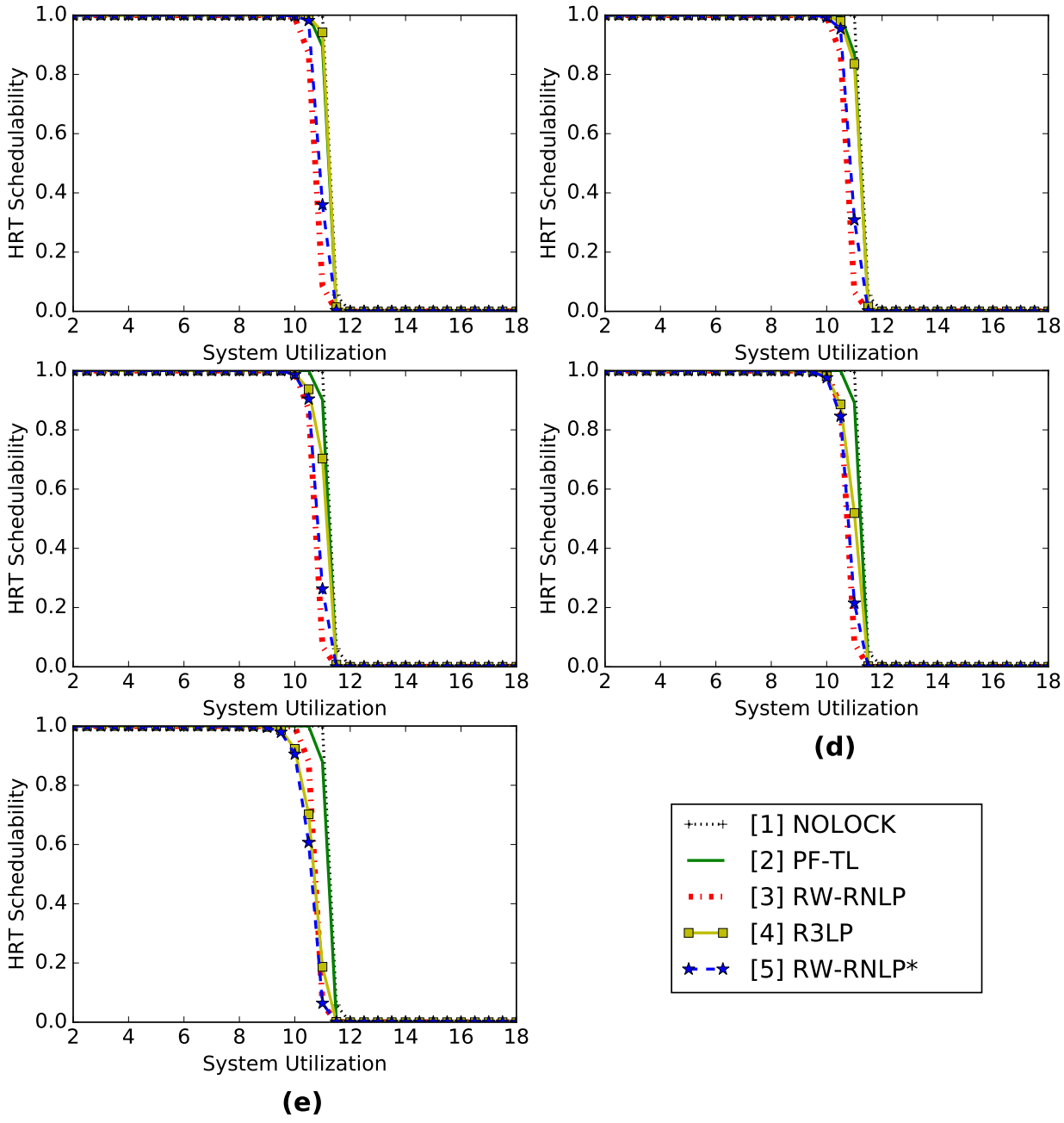



Figure 545: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, short critical-section lengths, and read probability 0 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

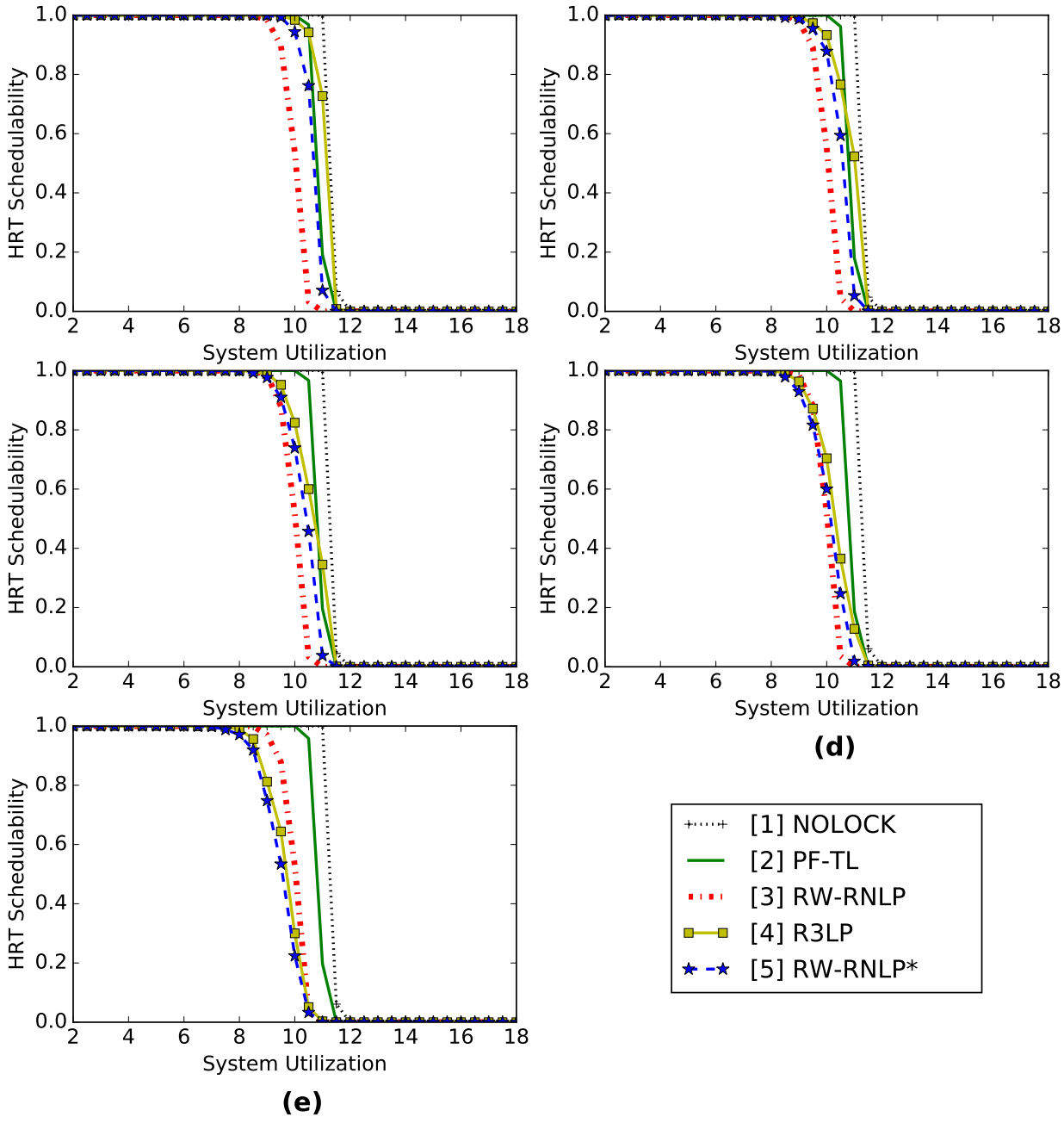



Figure 546: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, short critical-section lengths, and read probability 0 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

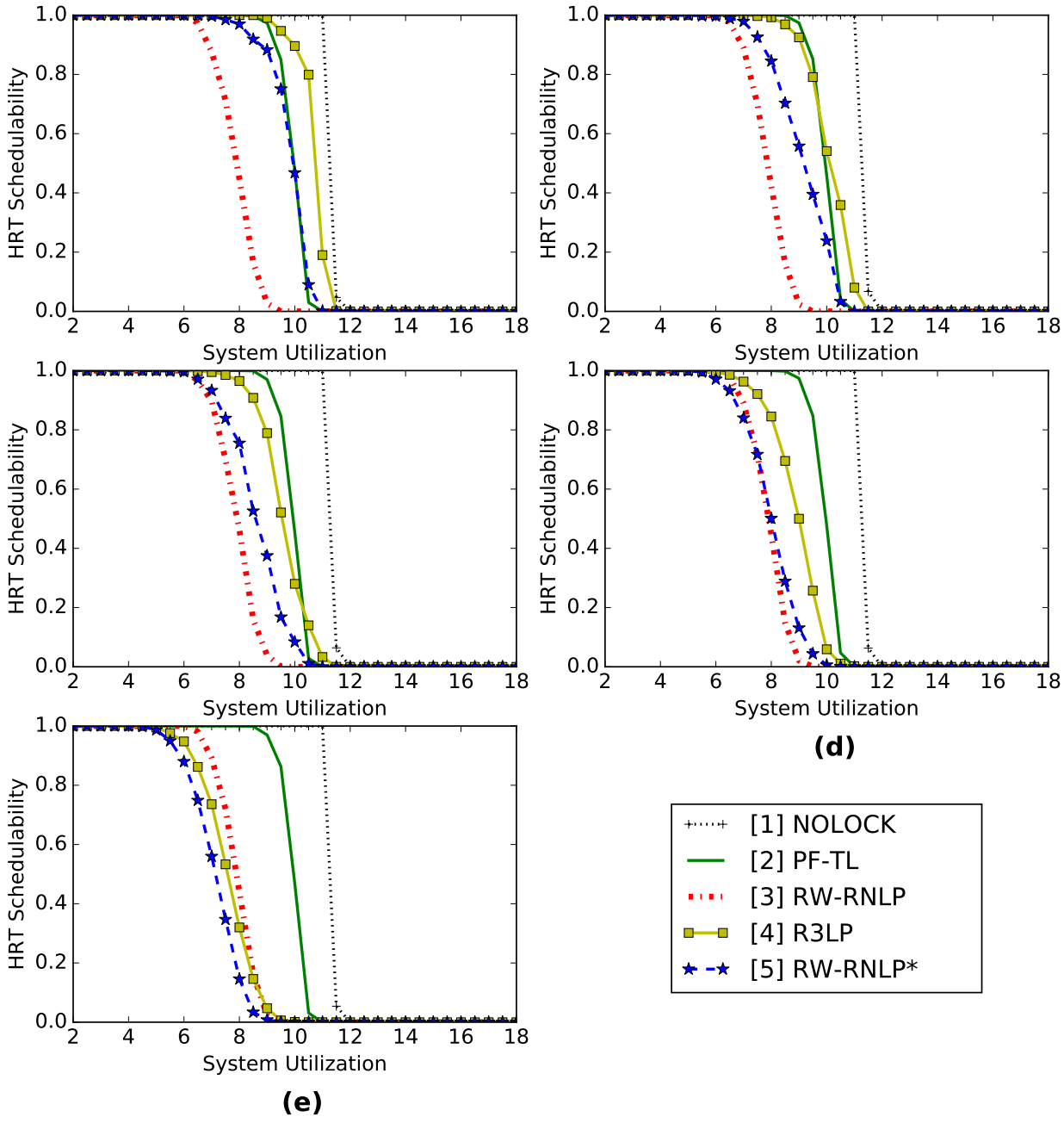



Figure 547: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, short critical-section lengths, and read probability 0 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

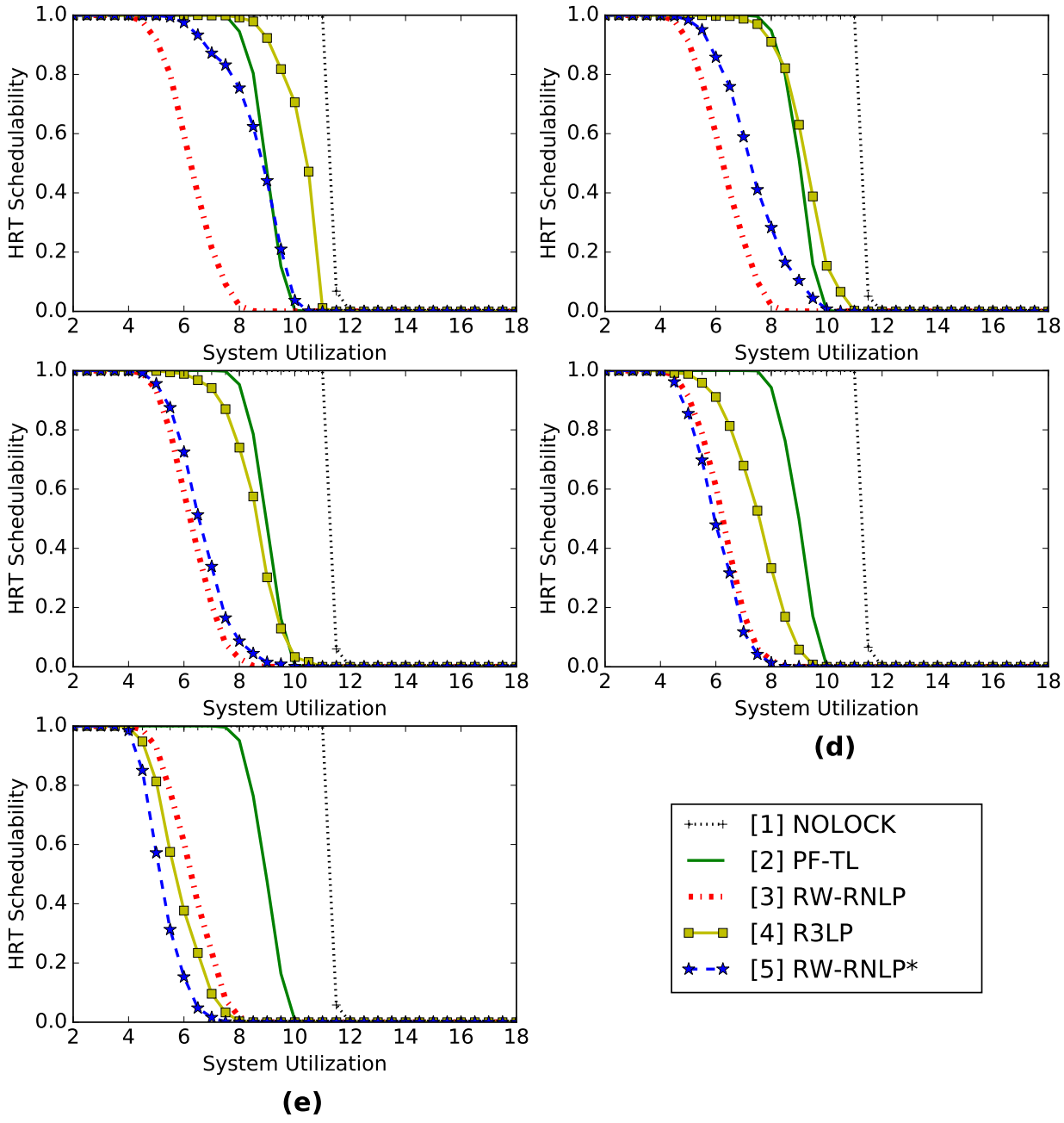



Figure 548: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, short critical-section lengths, and read probability 0 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

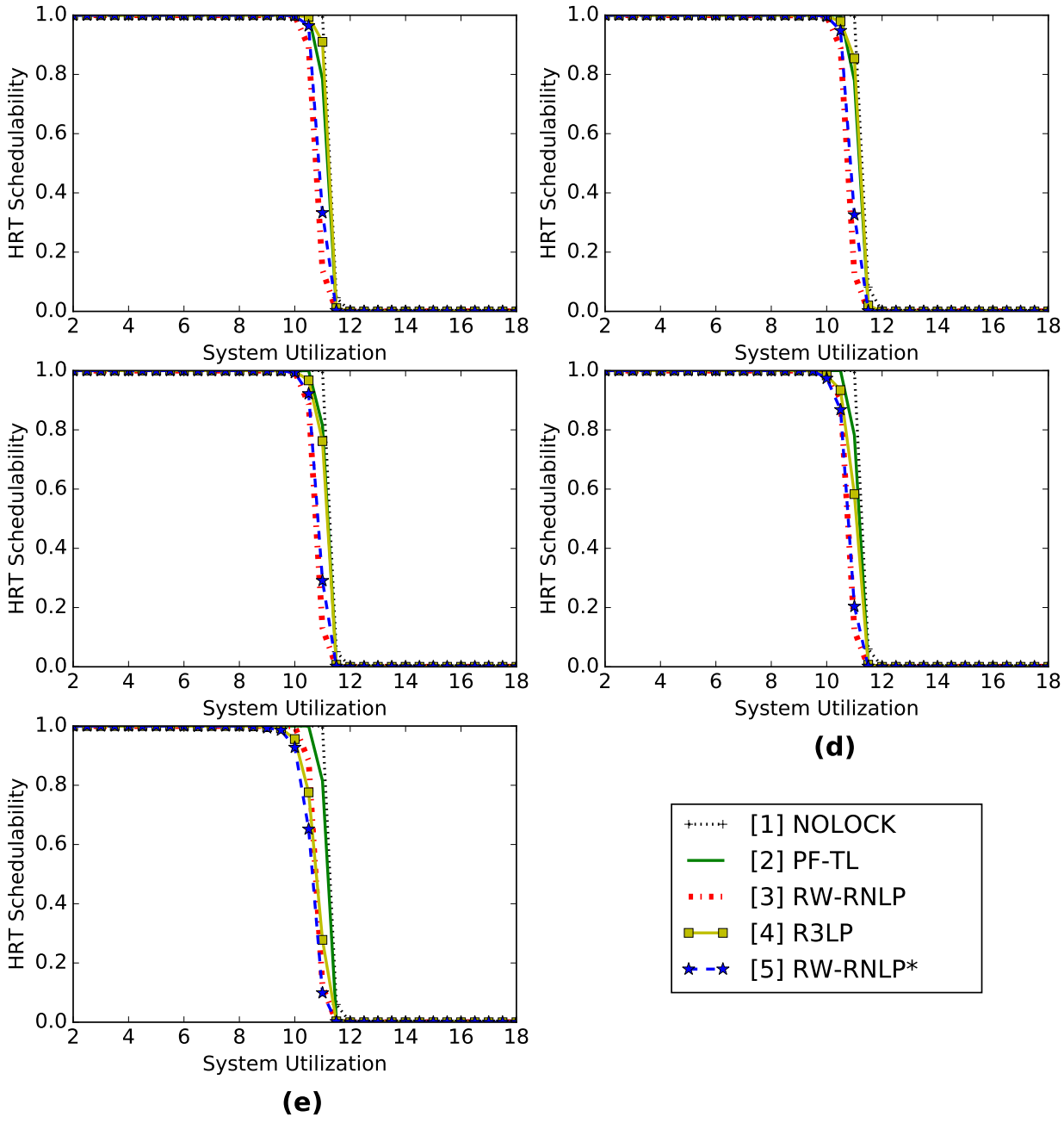



Figure 549: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, short critical-section lengths, and read probability 0.2 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

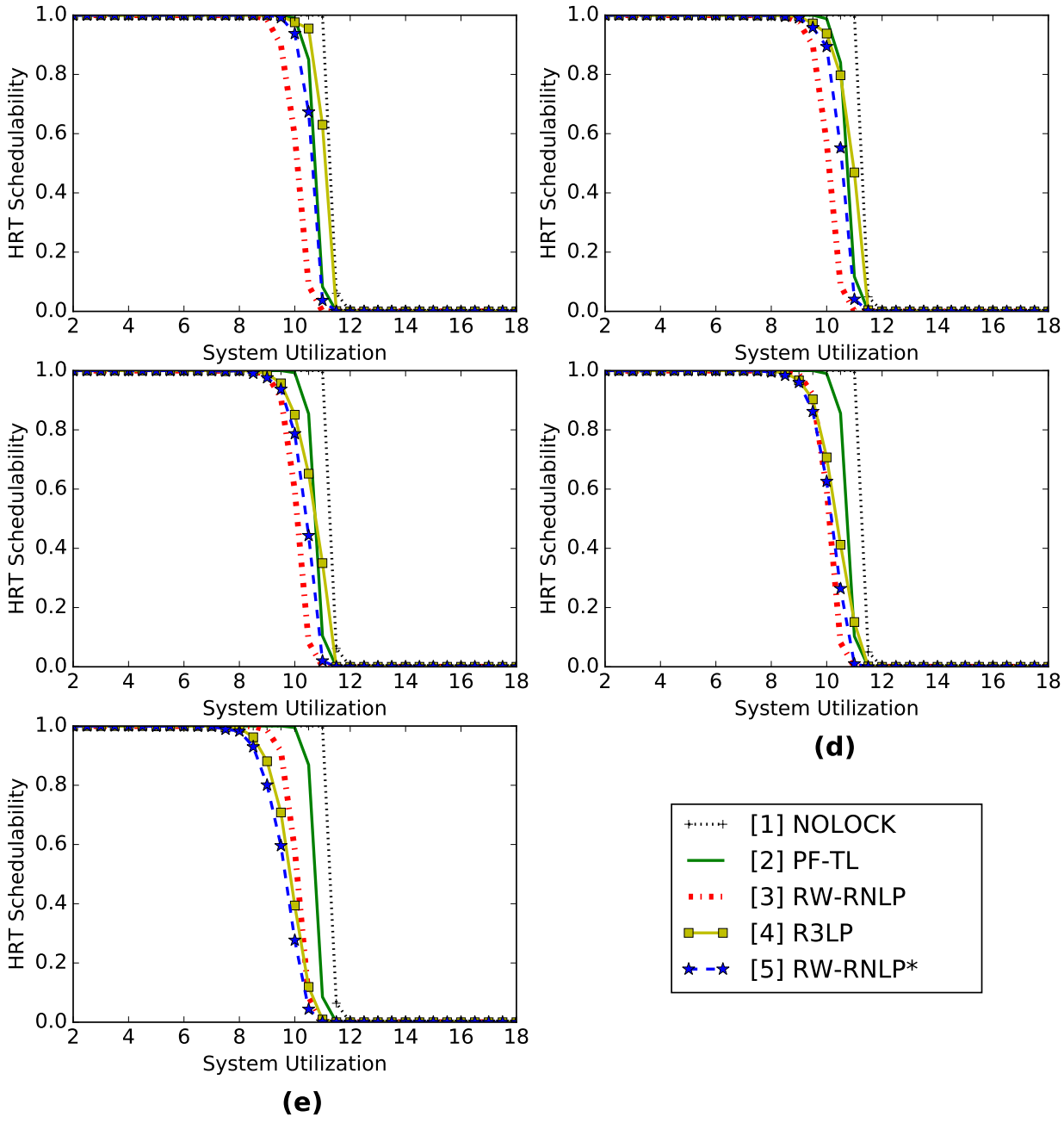



Figure 550: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, short critical-section lengths, and read probability 0.2 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

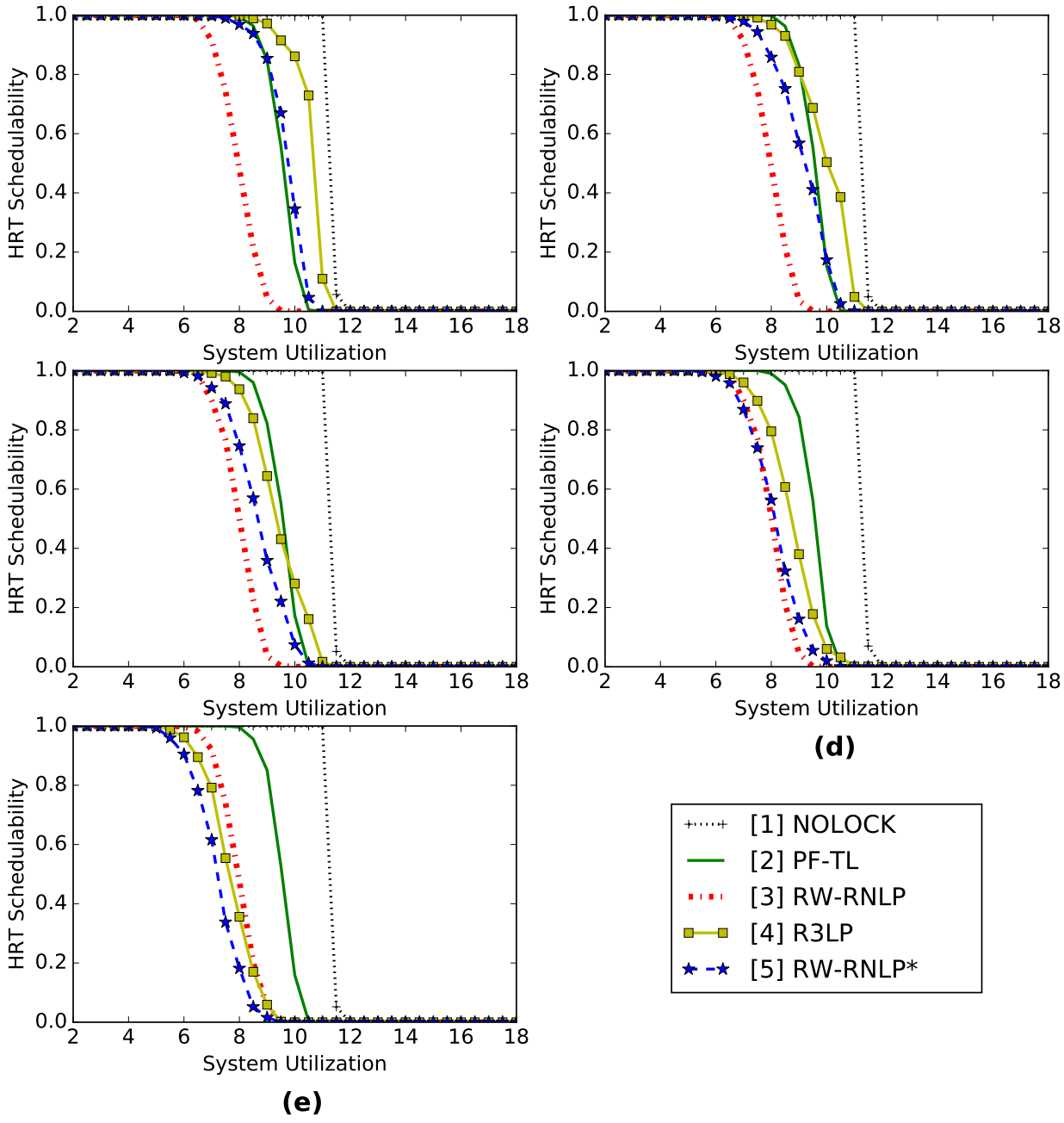



Figure 551: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, short critical-section lengths, and read probability 0.2 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

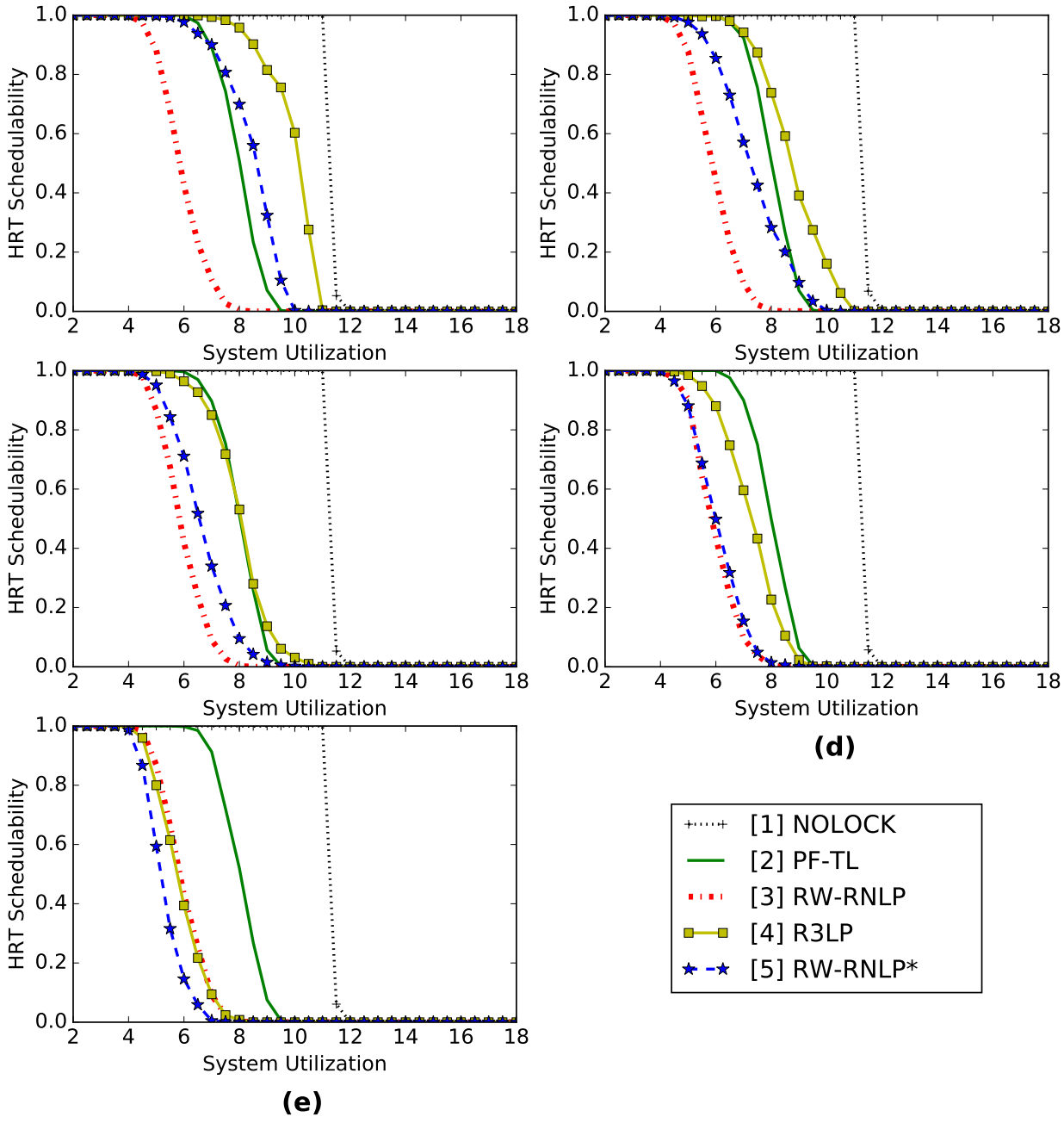



Figure 552: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, short critical-section lengths, and read probability 0.2 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

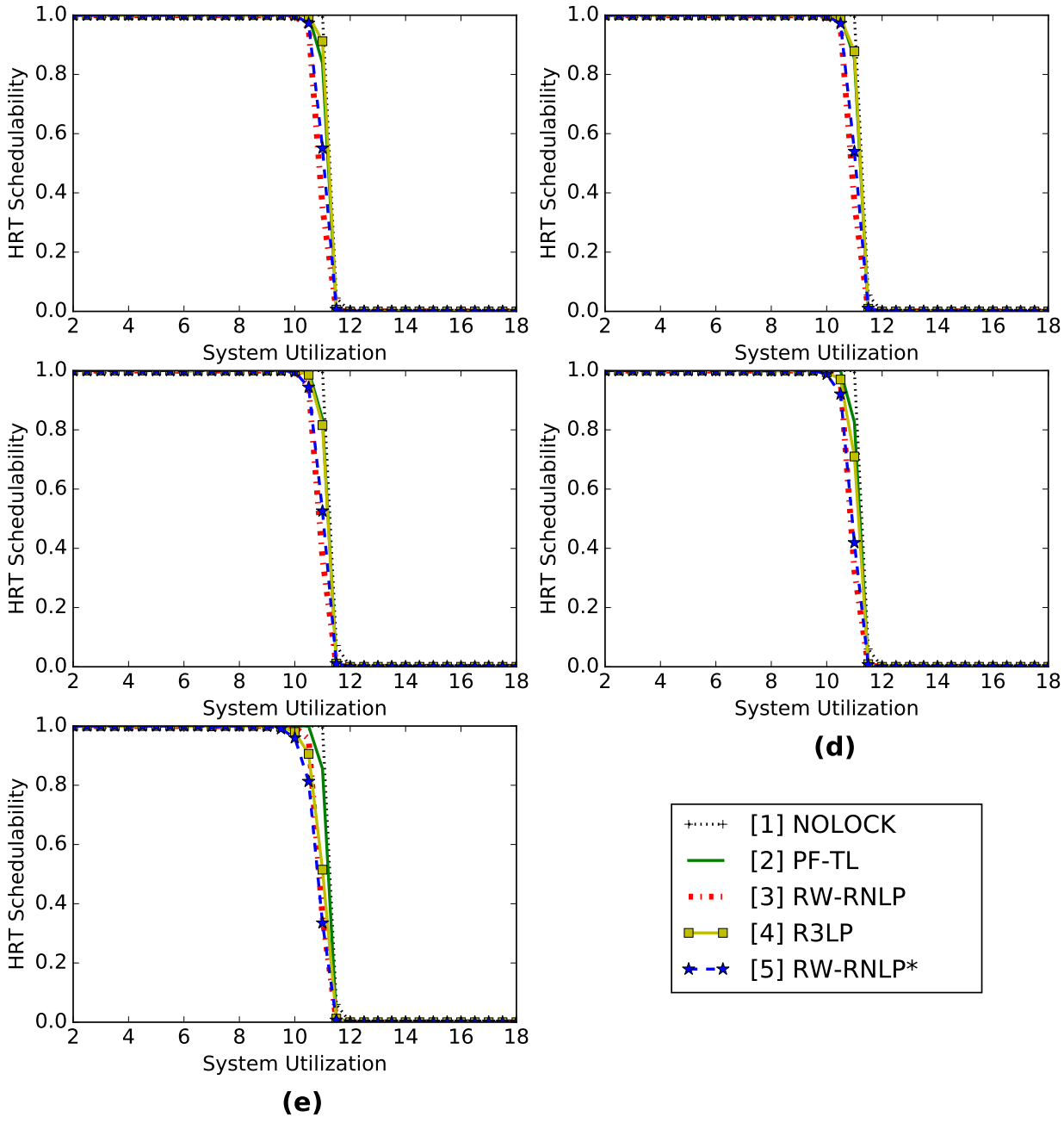



Figure 553: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, short critical-section lengths, and read probability 0.5 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

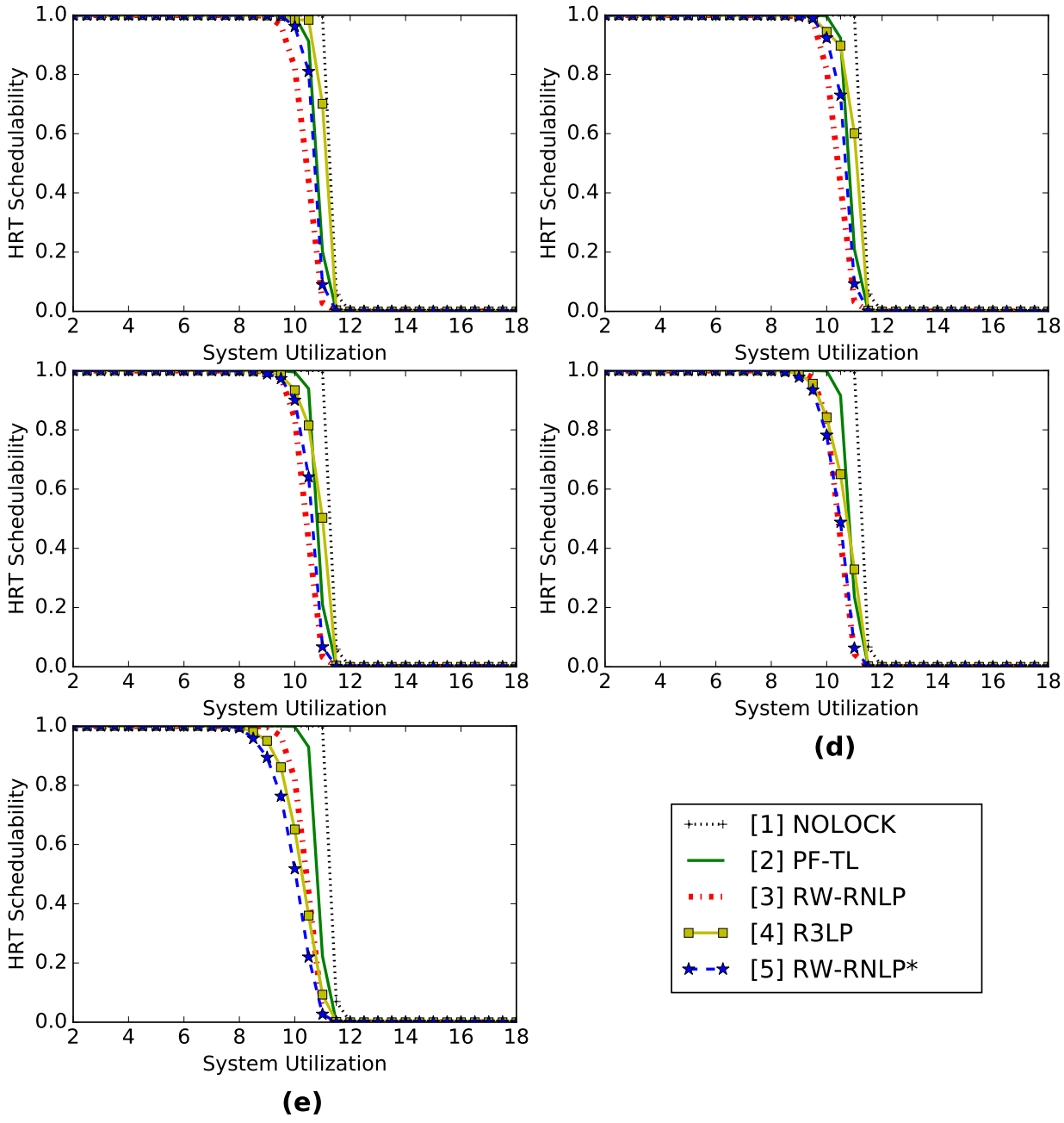



Figure 554: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, short critical-section lengths, and read probability 0.5 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

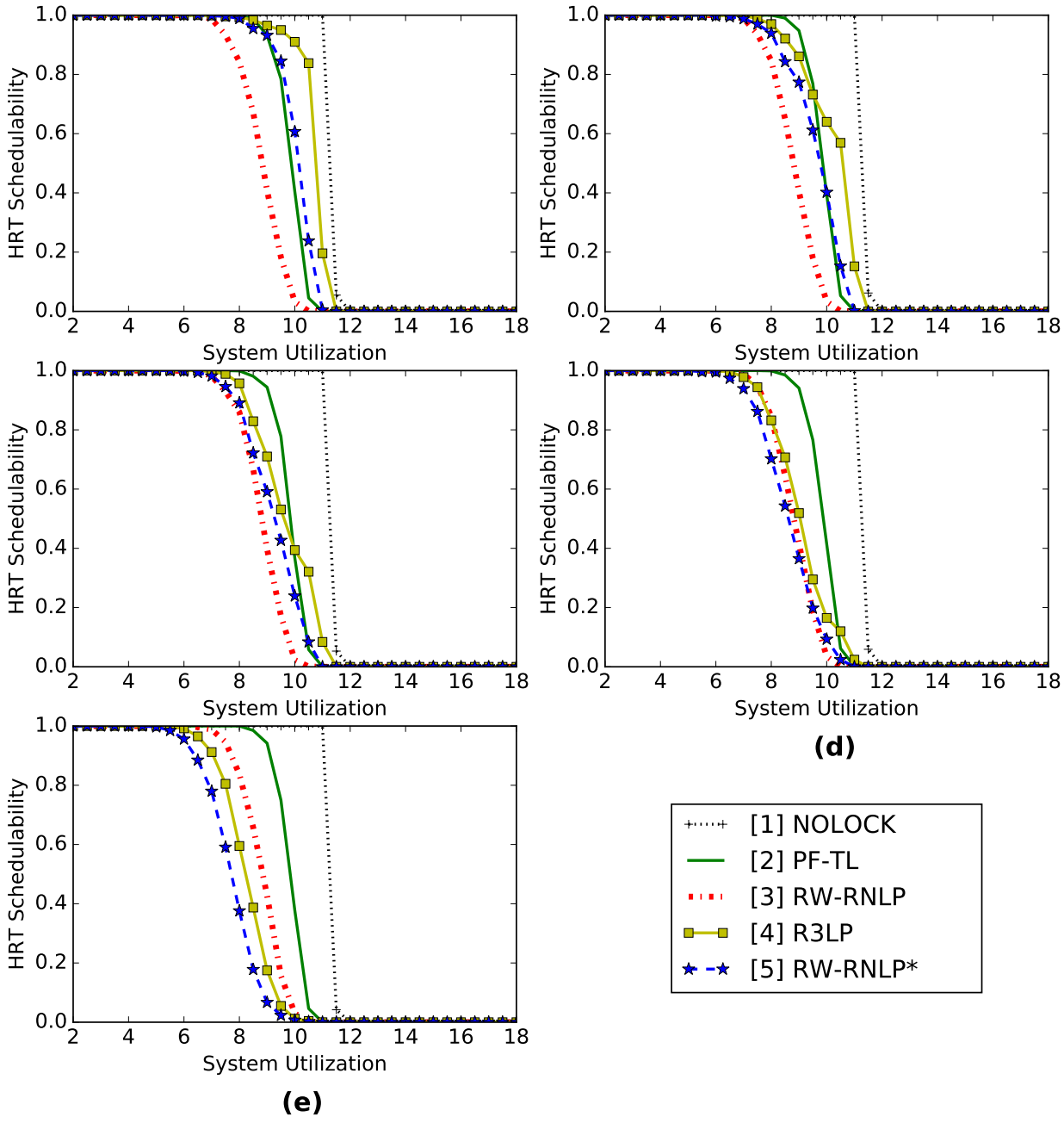



Figure 555: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, short critical-section lengths, and read probability 0.5 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

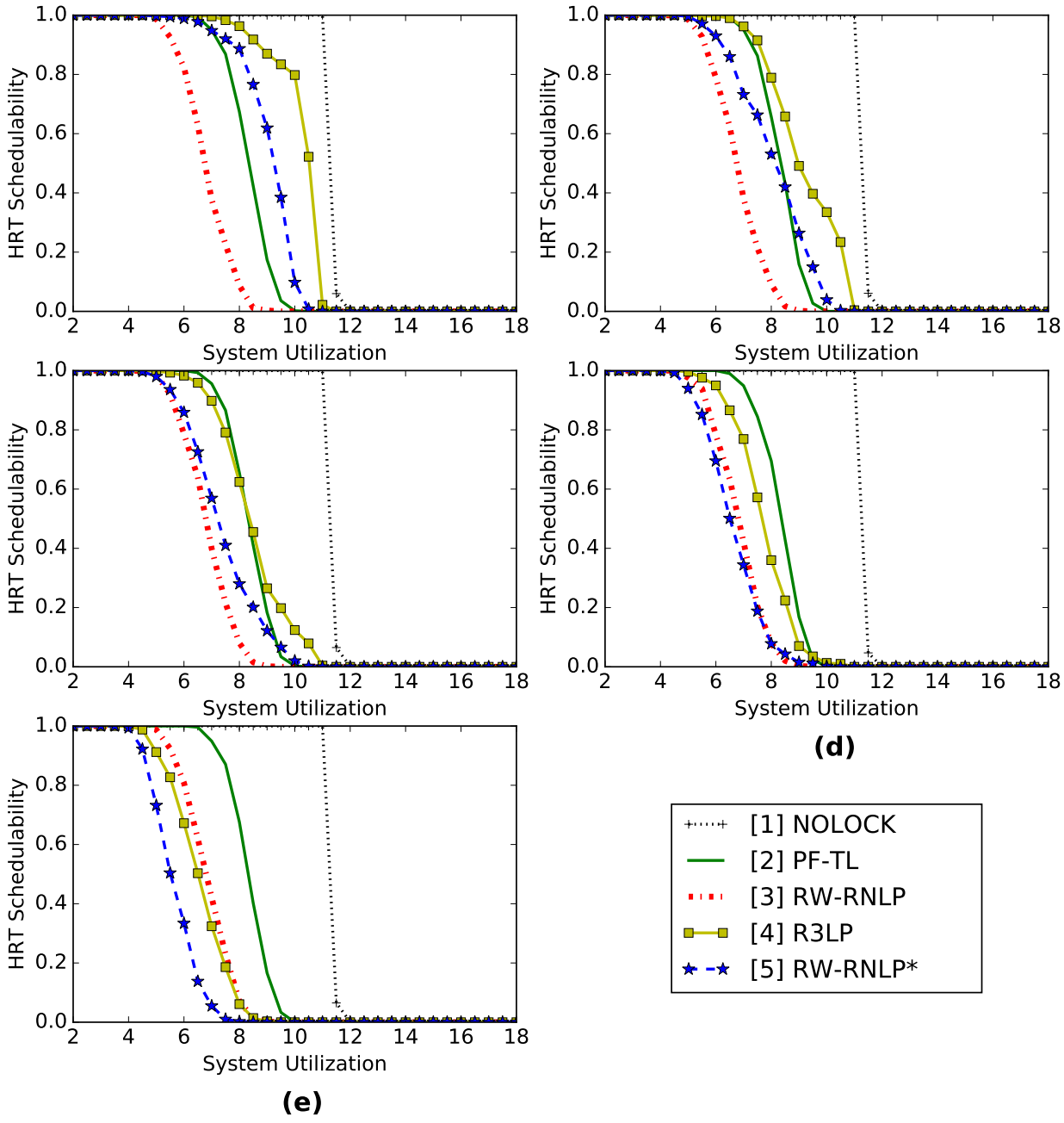



Figure 556: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, short critical-section lengths, and read probability 0.5 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

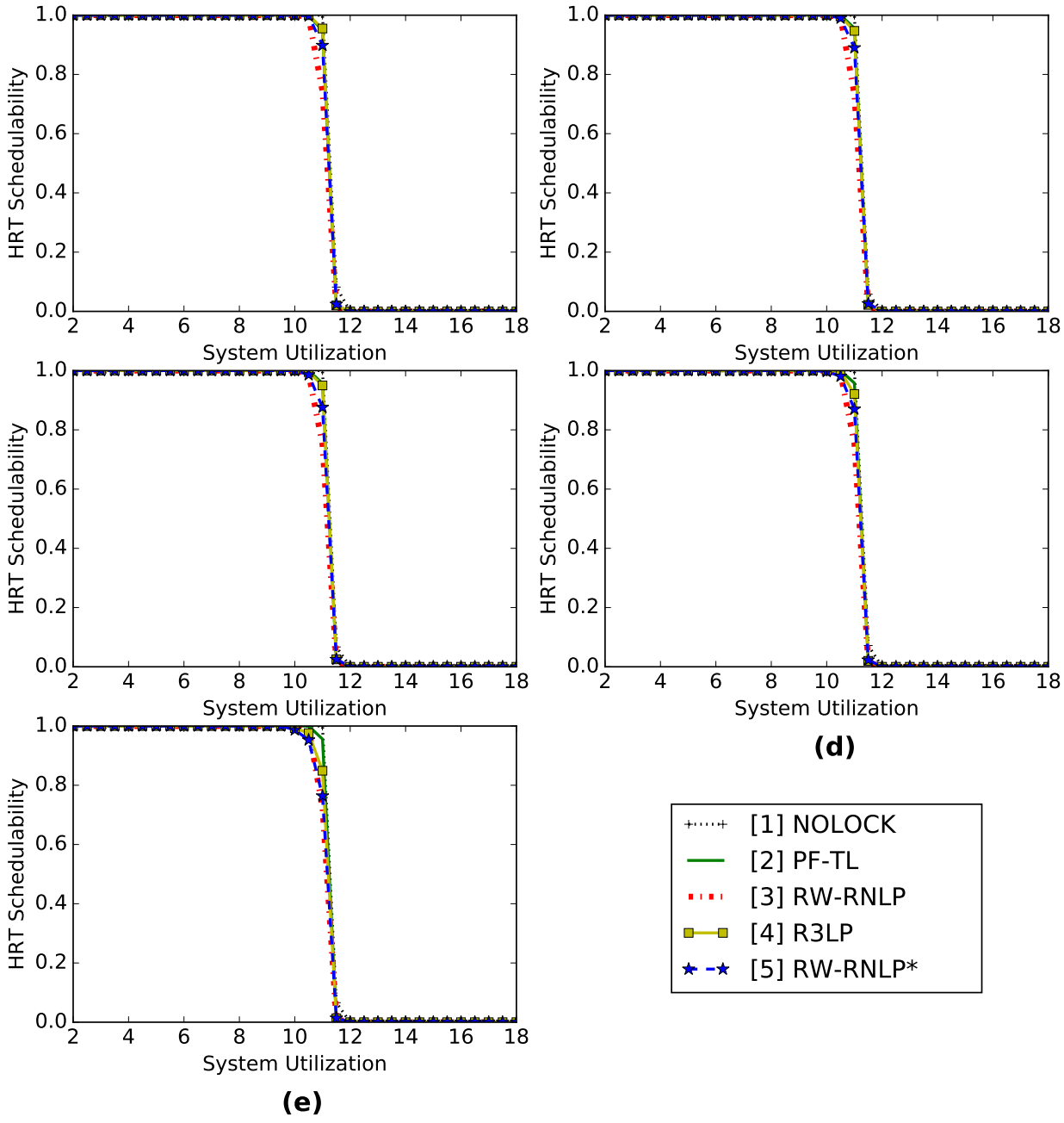



Figure 557: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, short critical-section lengths, and read probability 0.8 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

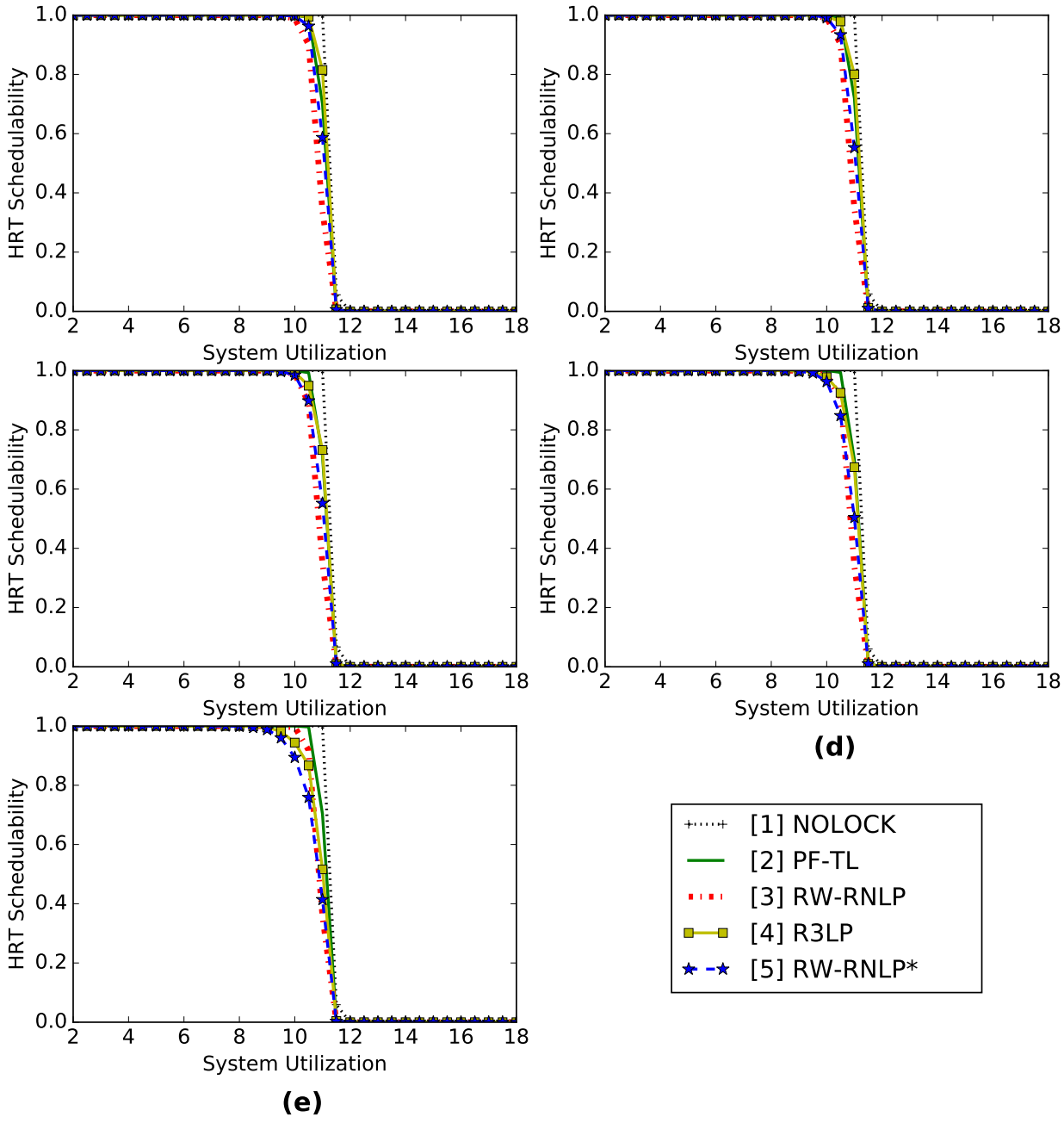



Figure 558: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, short critical-section lengths, and read probability 0.8 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

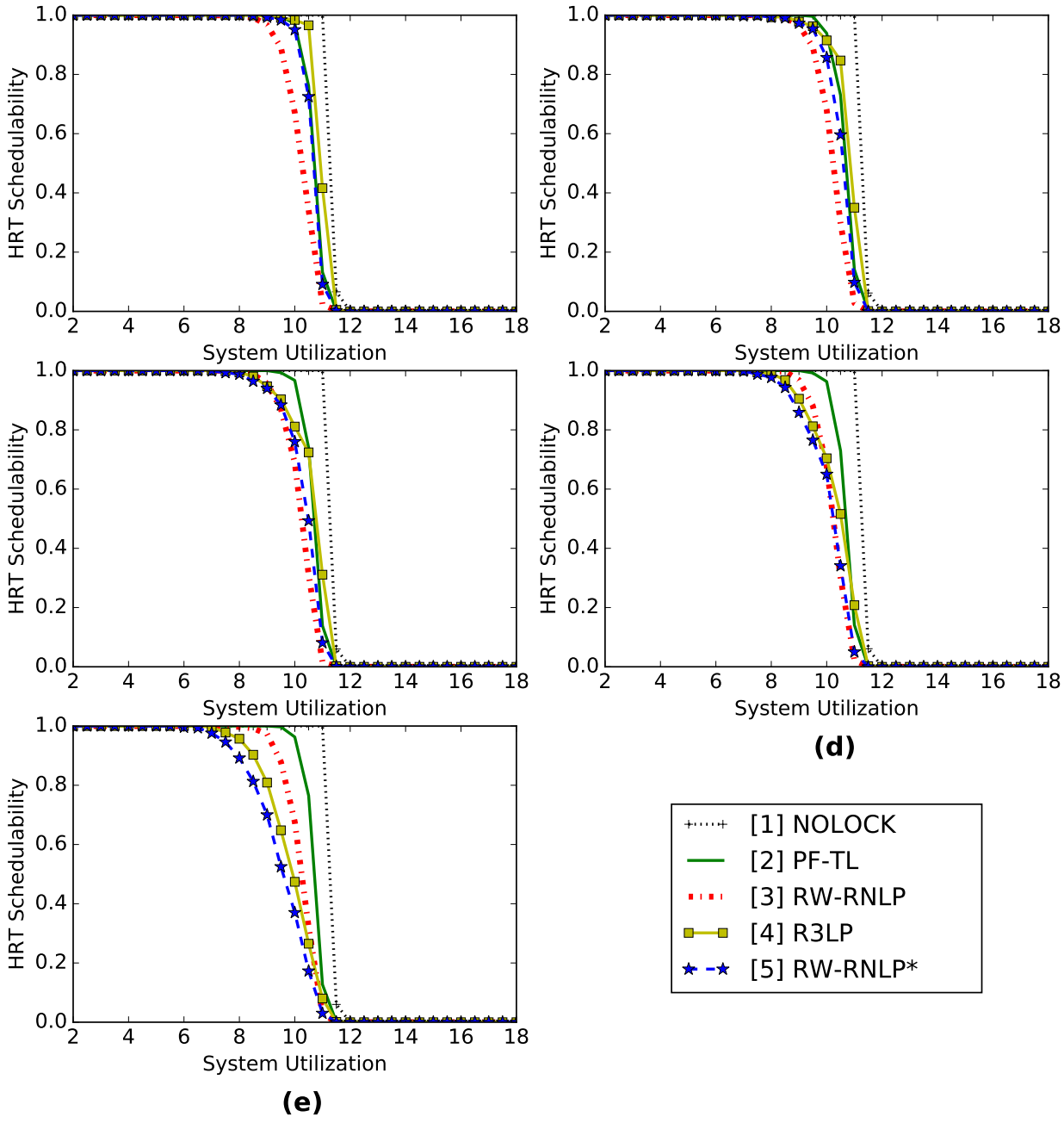



Figure 559: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, short critical-section lengths, and read probability 0.8 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

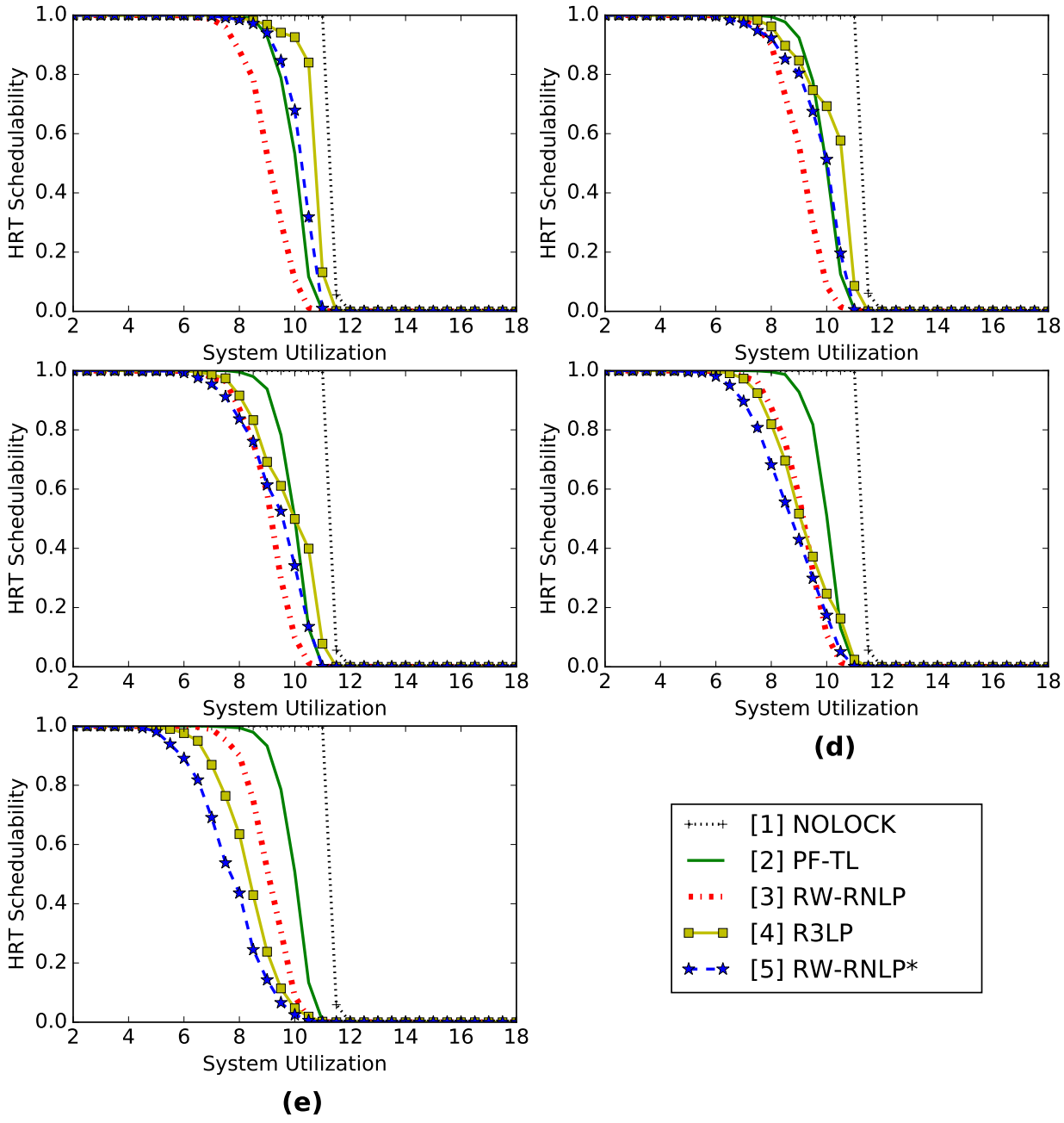



Figure 560: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, short critical-section lengths, and read probability 0.8 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

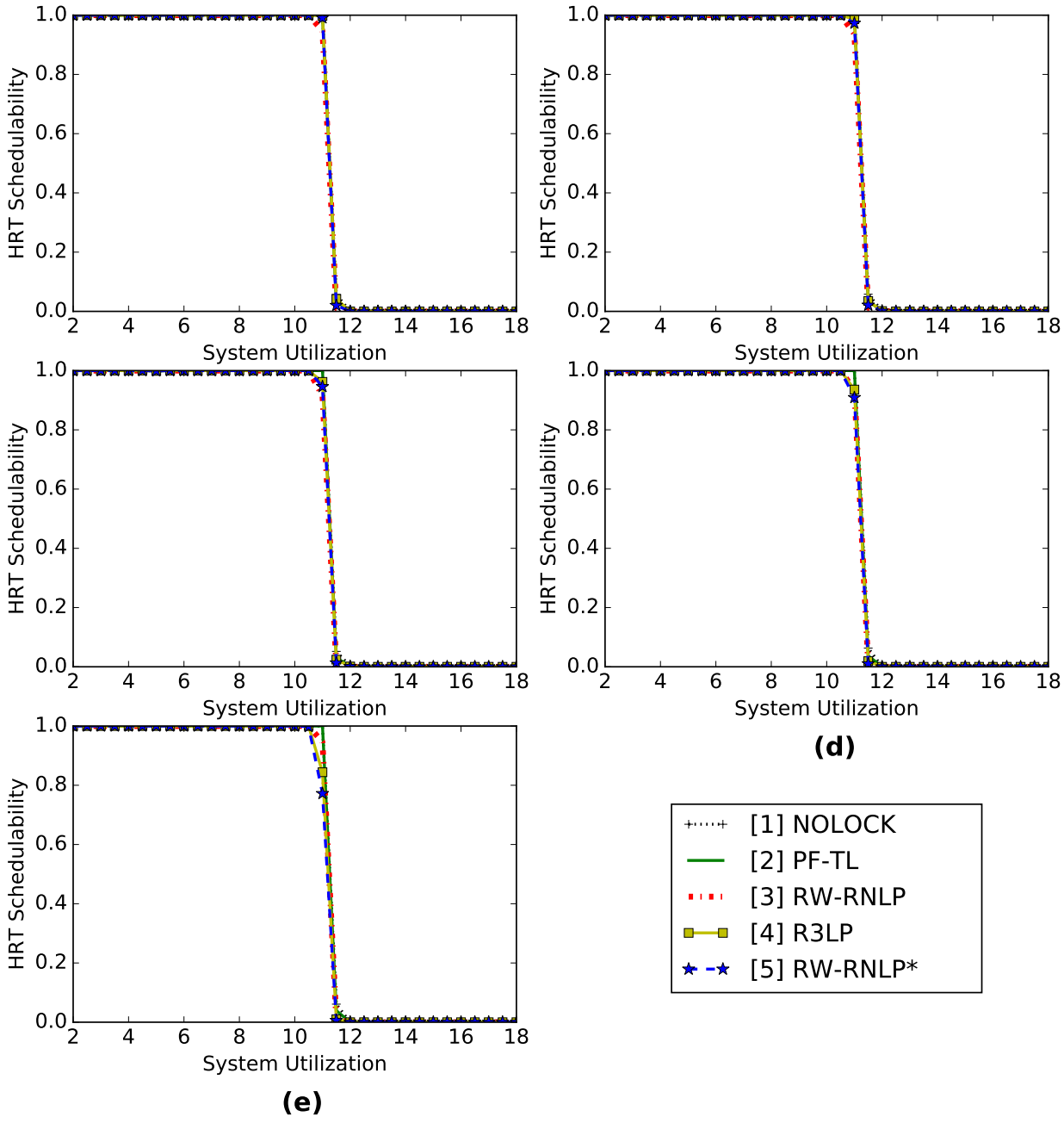



Figure 561: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, short critical-section lengths, and read probability 0 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

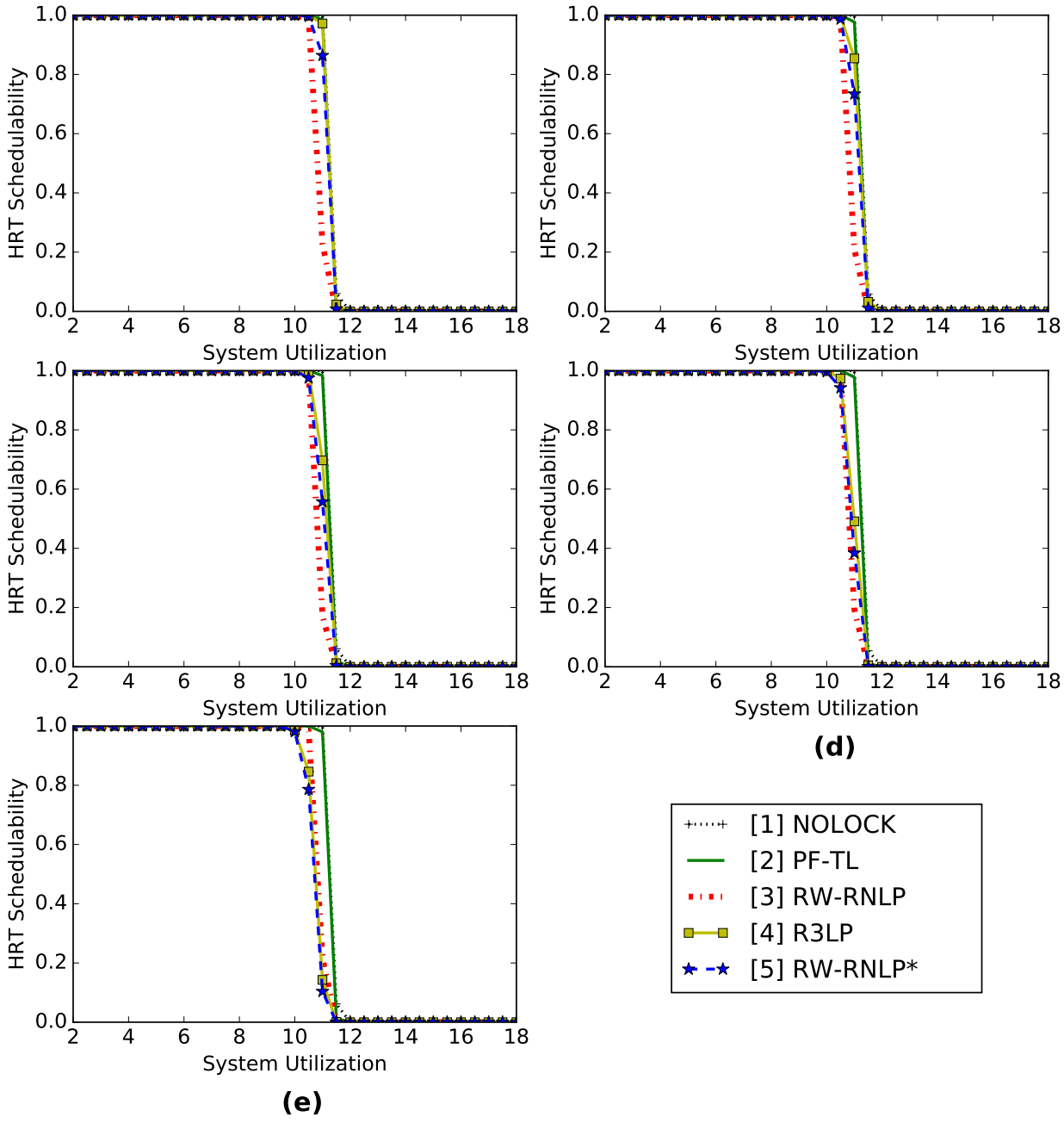



Figure 562: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, short critical-section lengths, and read probability 0 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

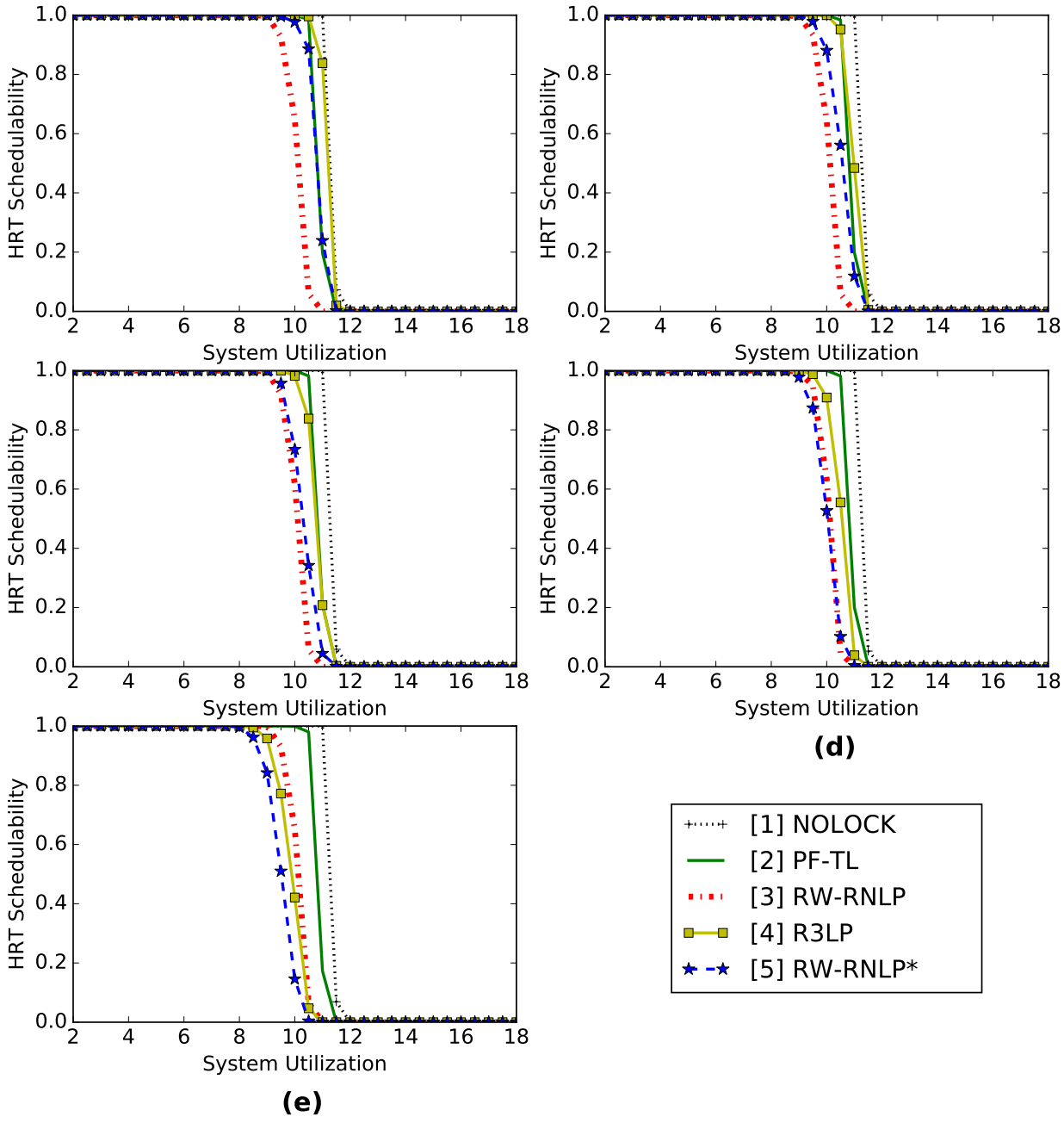



Figure 563: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, short critical-section lengths, and read probability 0 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

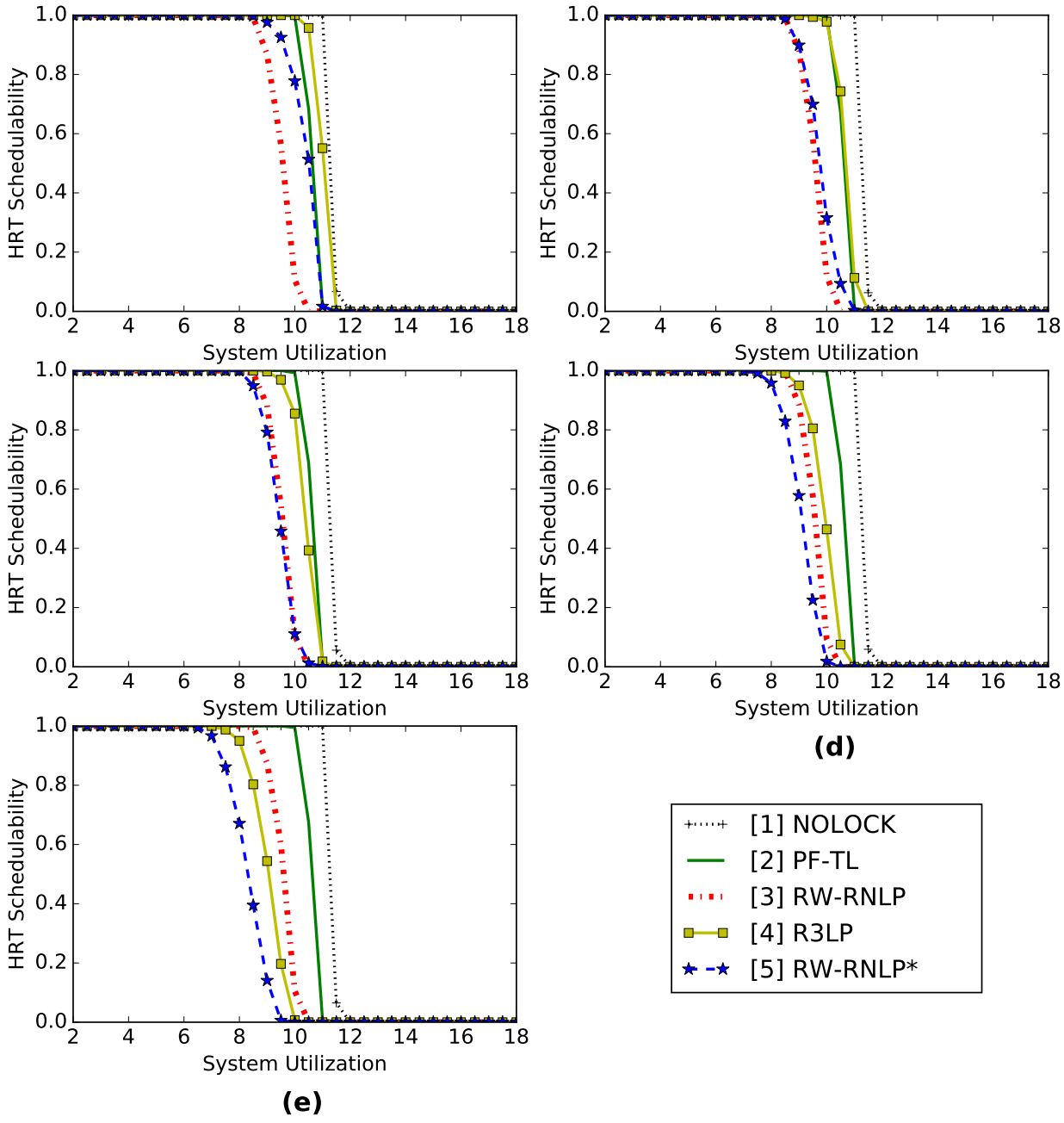



Figure 564: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, short critical-section lengths, and read probability 0 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

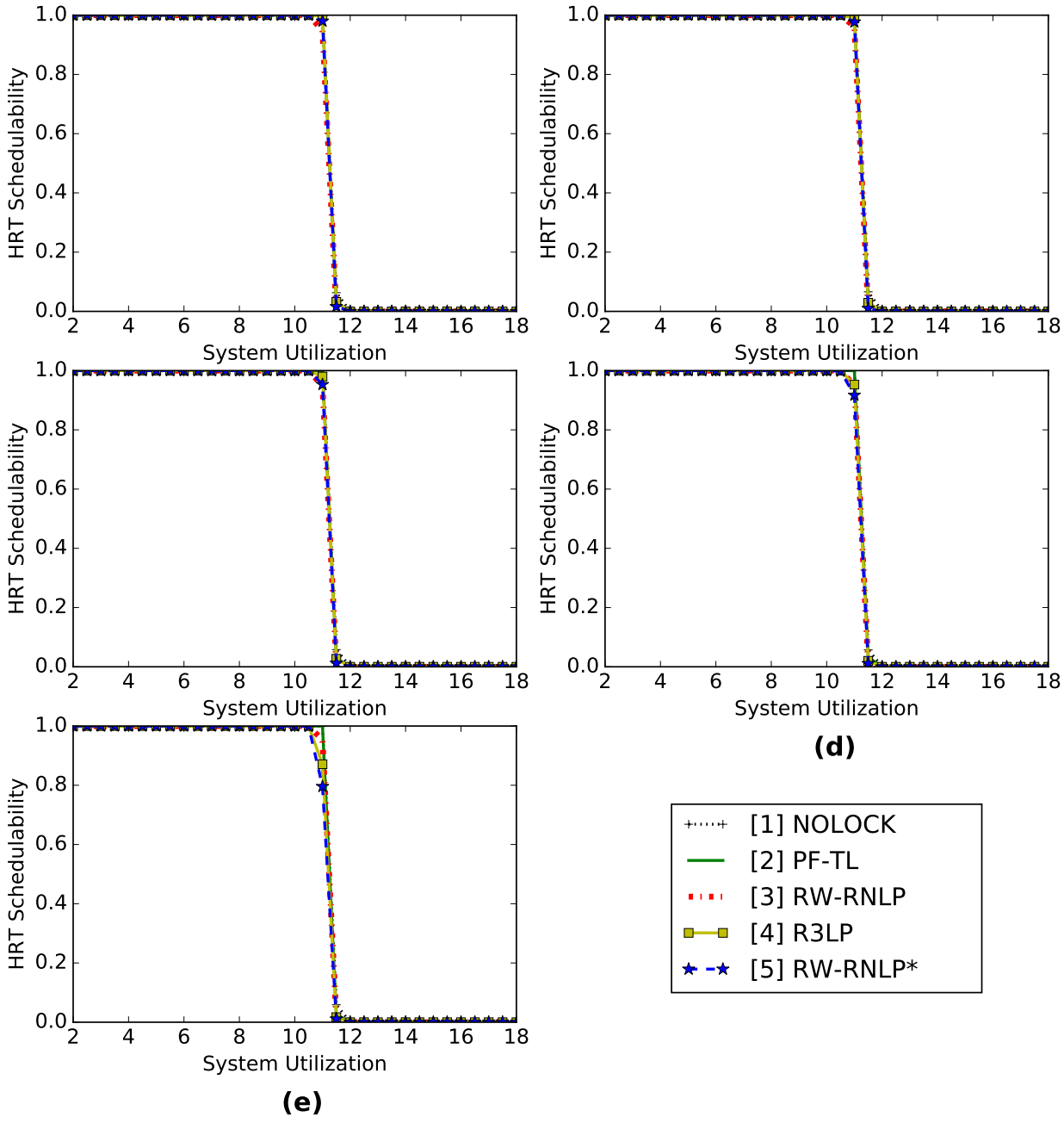



Figure 565: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, short critical-section lengths, and read probability 0.2 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

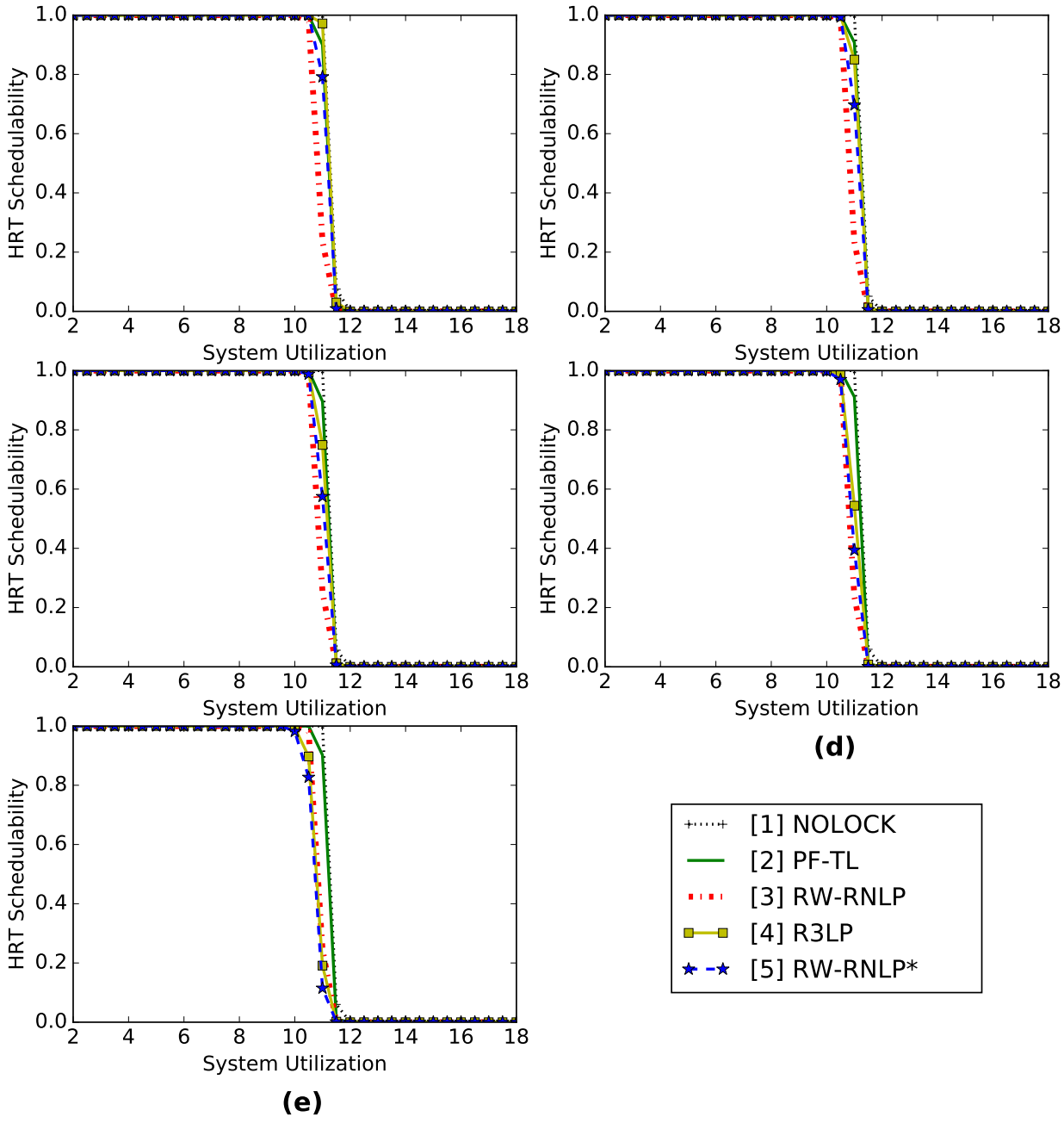



Figure 566: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, short critical-section lengths, and read probability 0.2 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

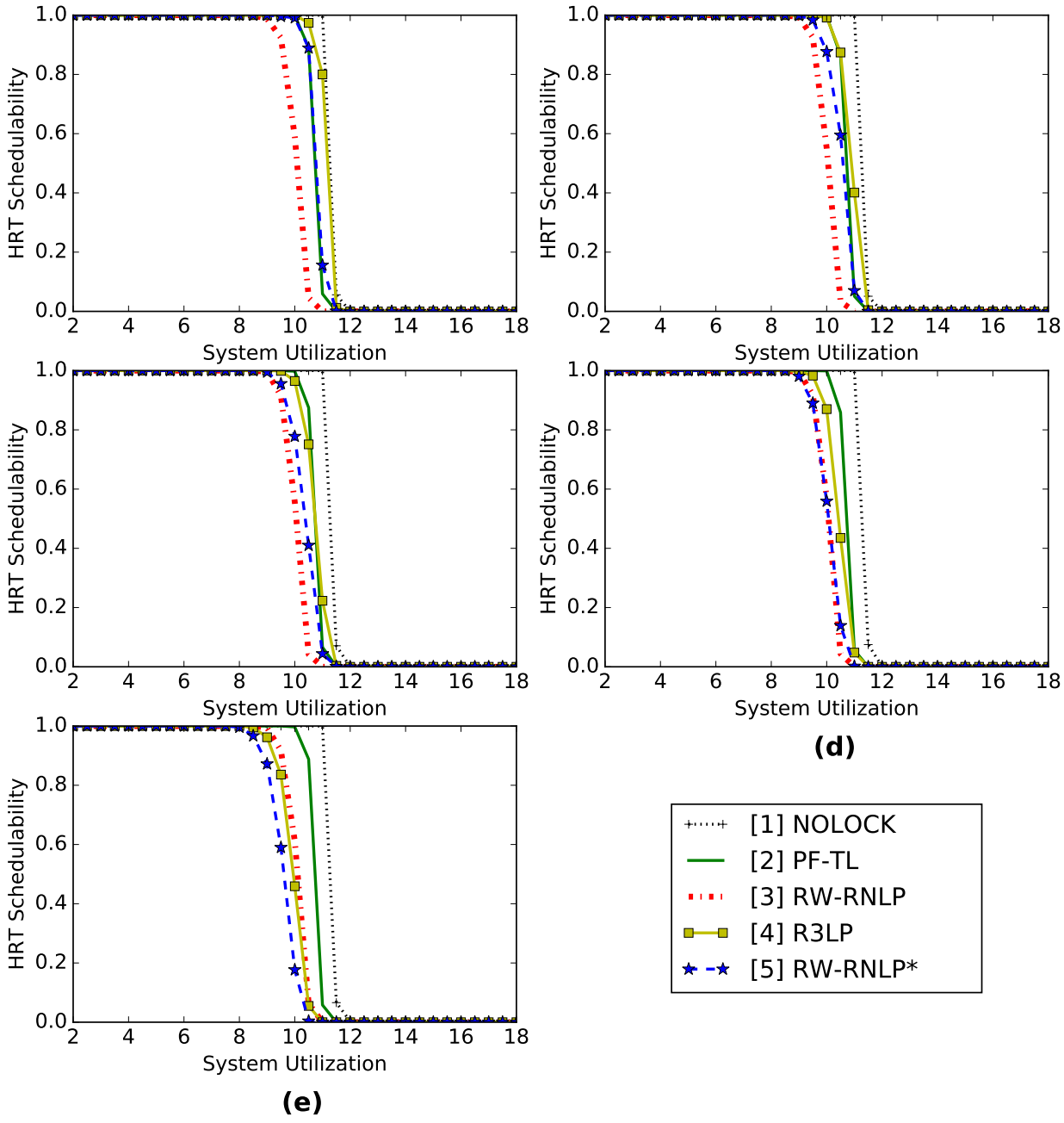



Figure 567: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, short critical-section lengths, and read probability 0.2 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

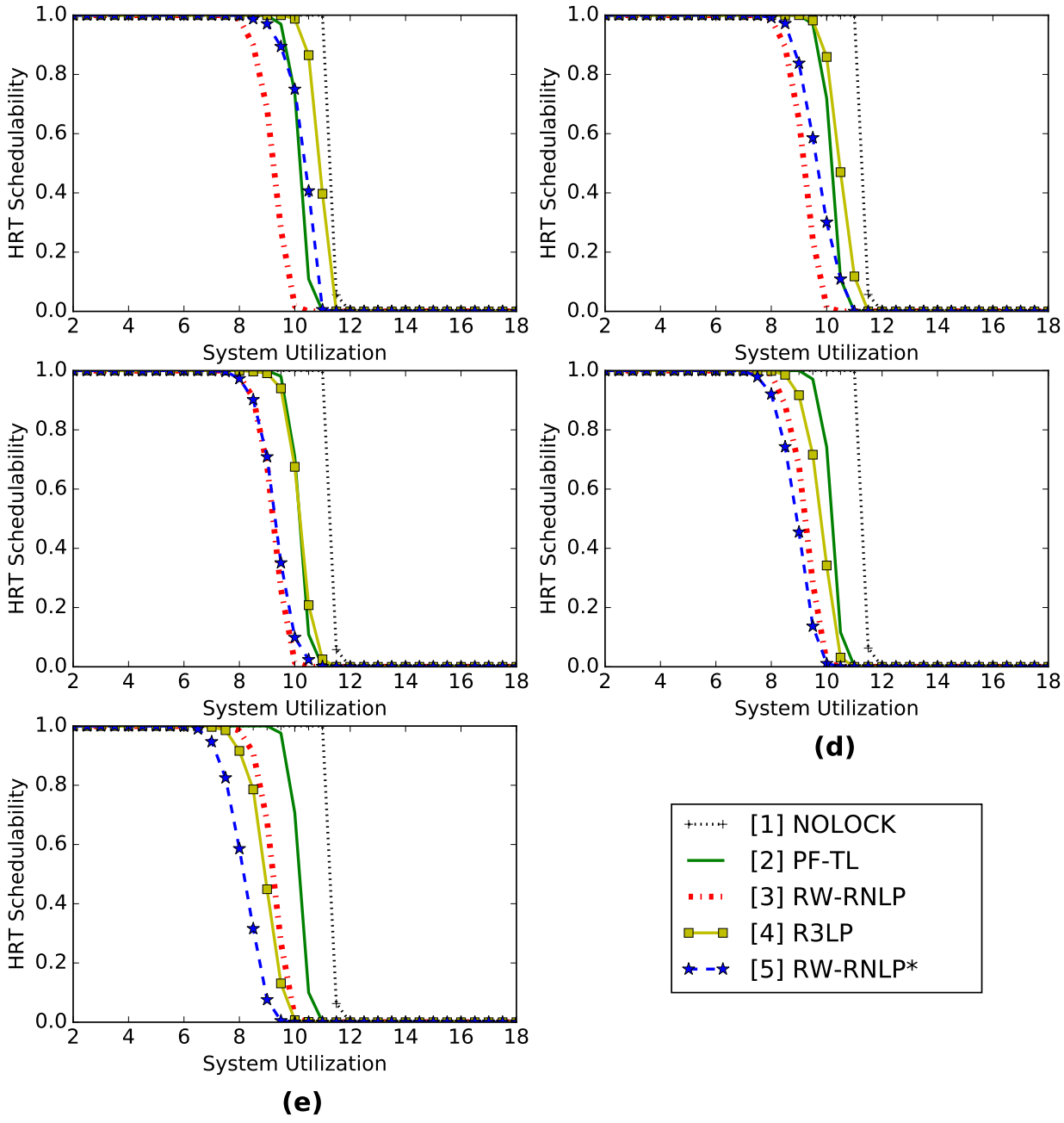



Figure 568: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, short critical-section lengths, and read probability 0.2 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

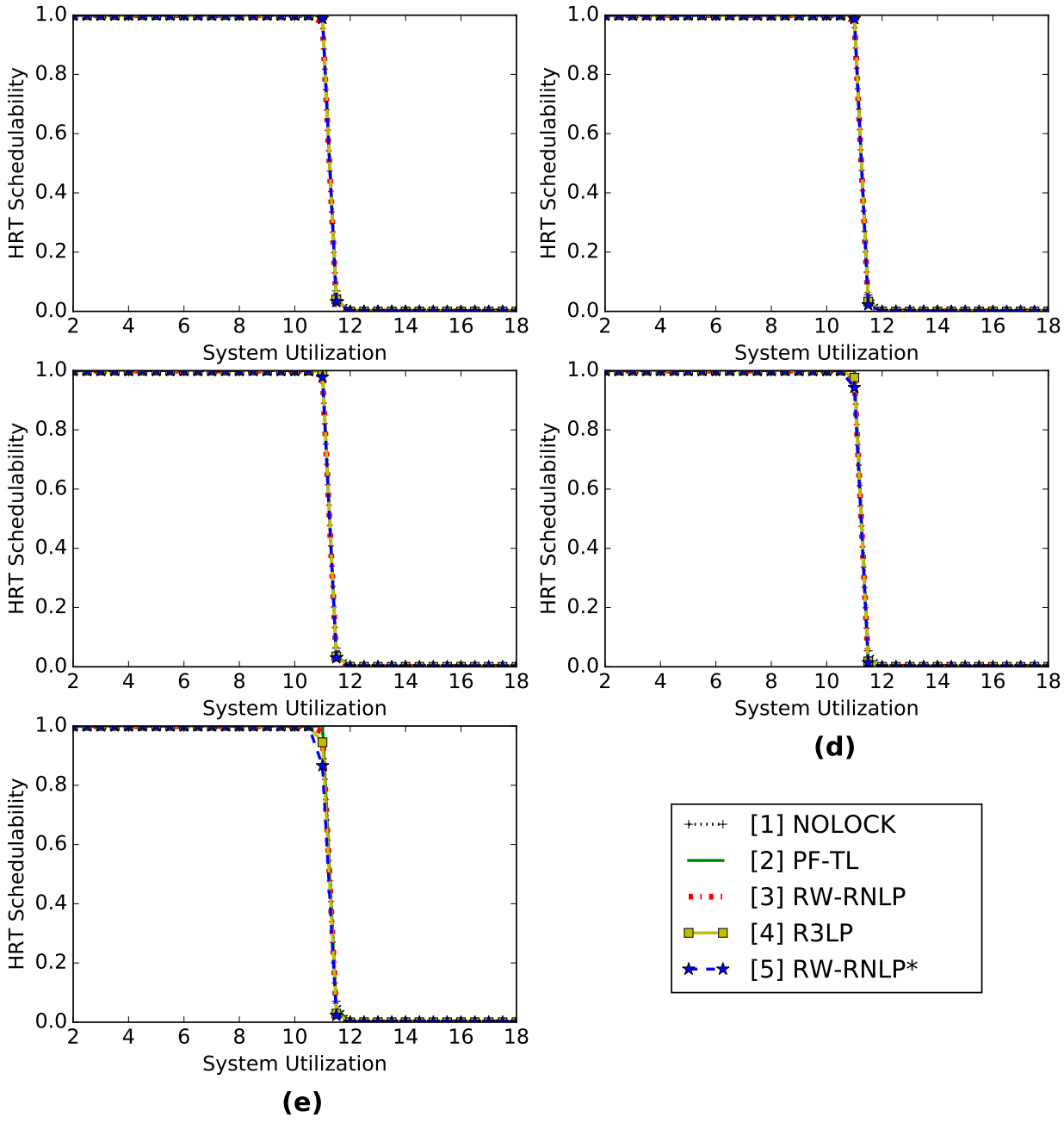



Figure 569: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, short critical-section lengths, and read probability 0.5 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

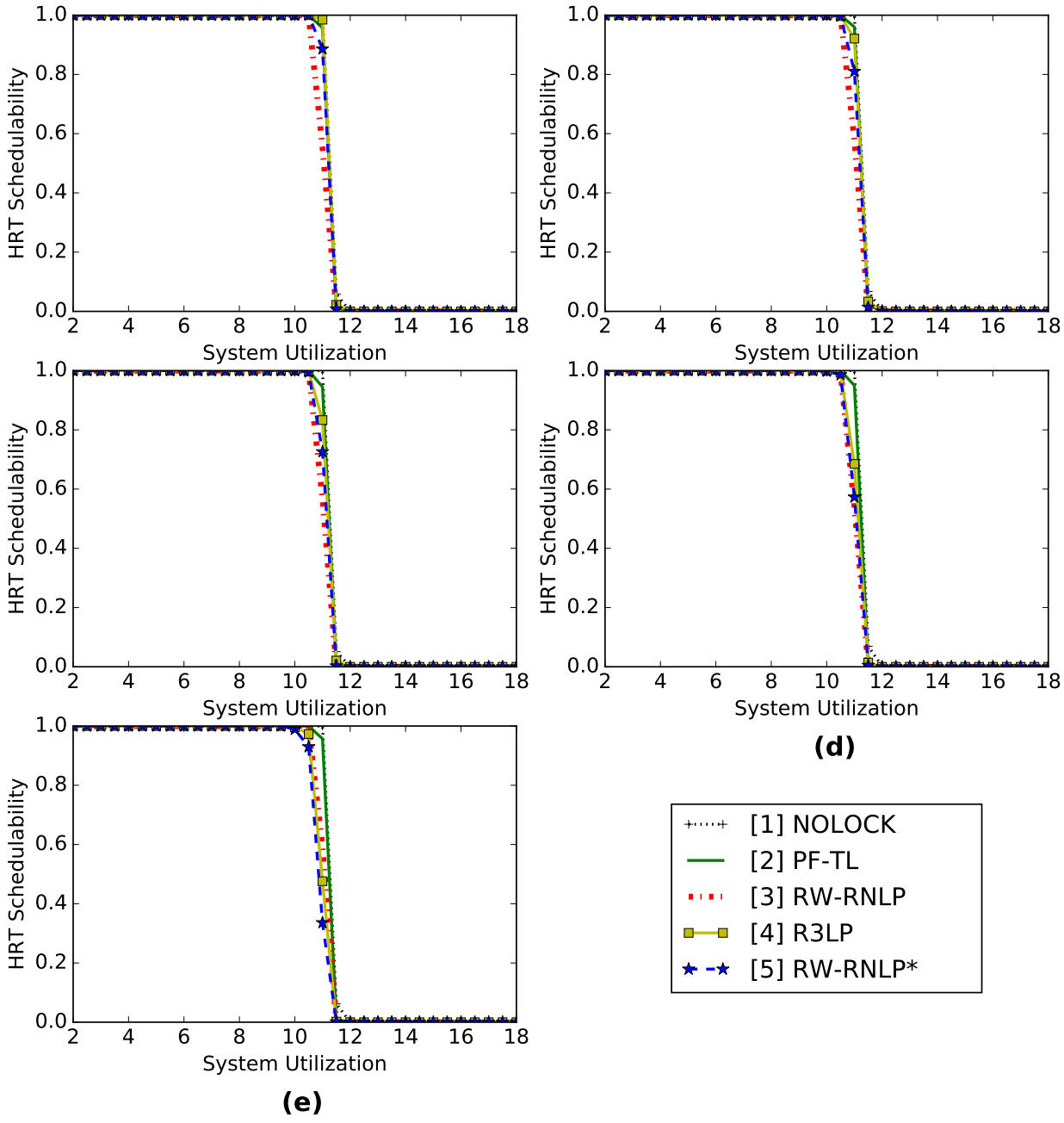



Figure 570: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, short critical-section lengths, and read probability 0.5 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

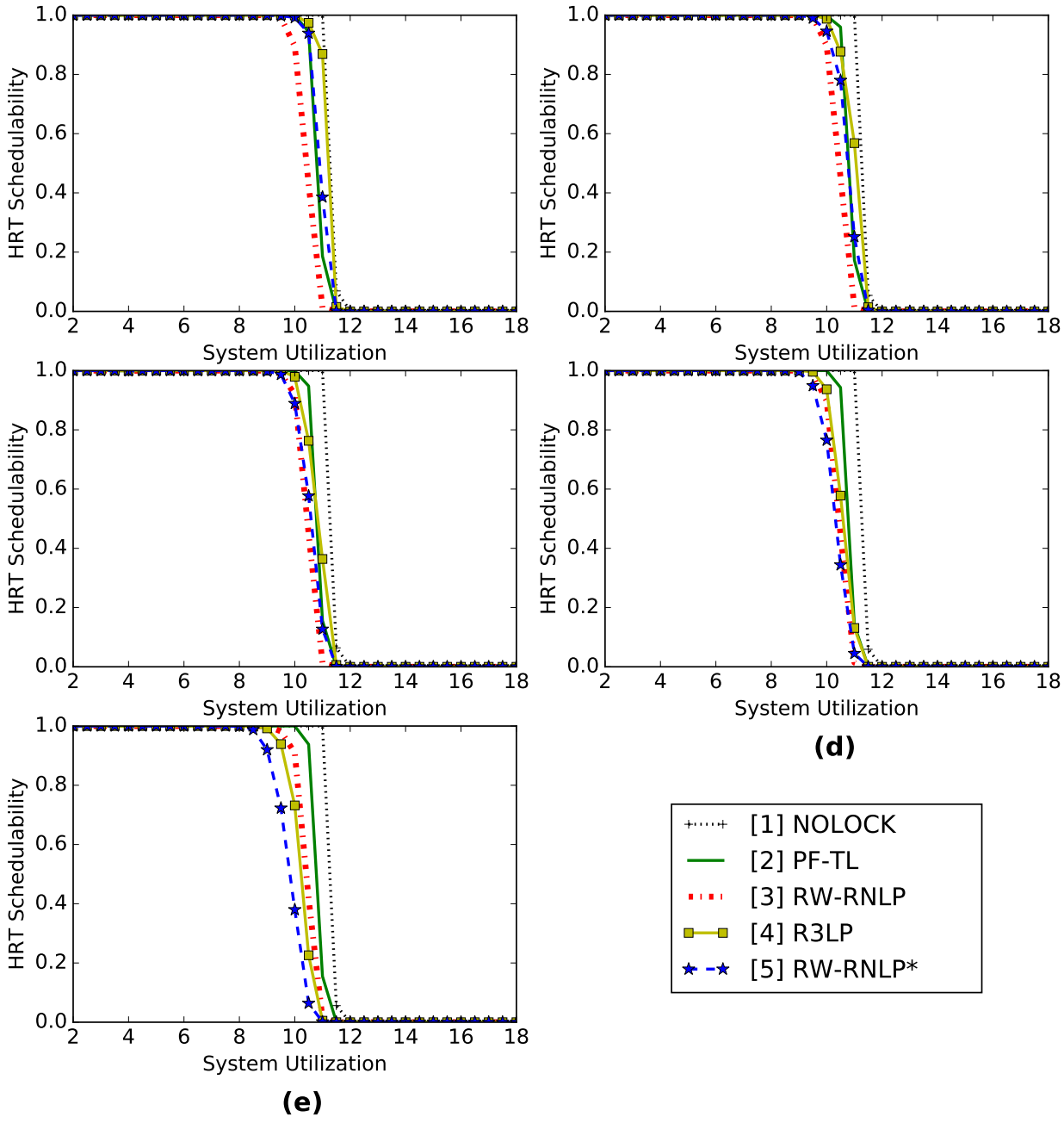



Figure 571: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, short critical-section lengths, and read probability 0.5 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

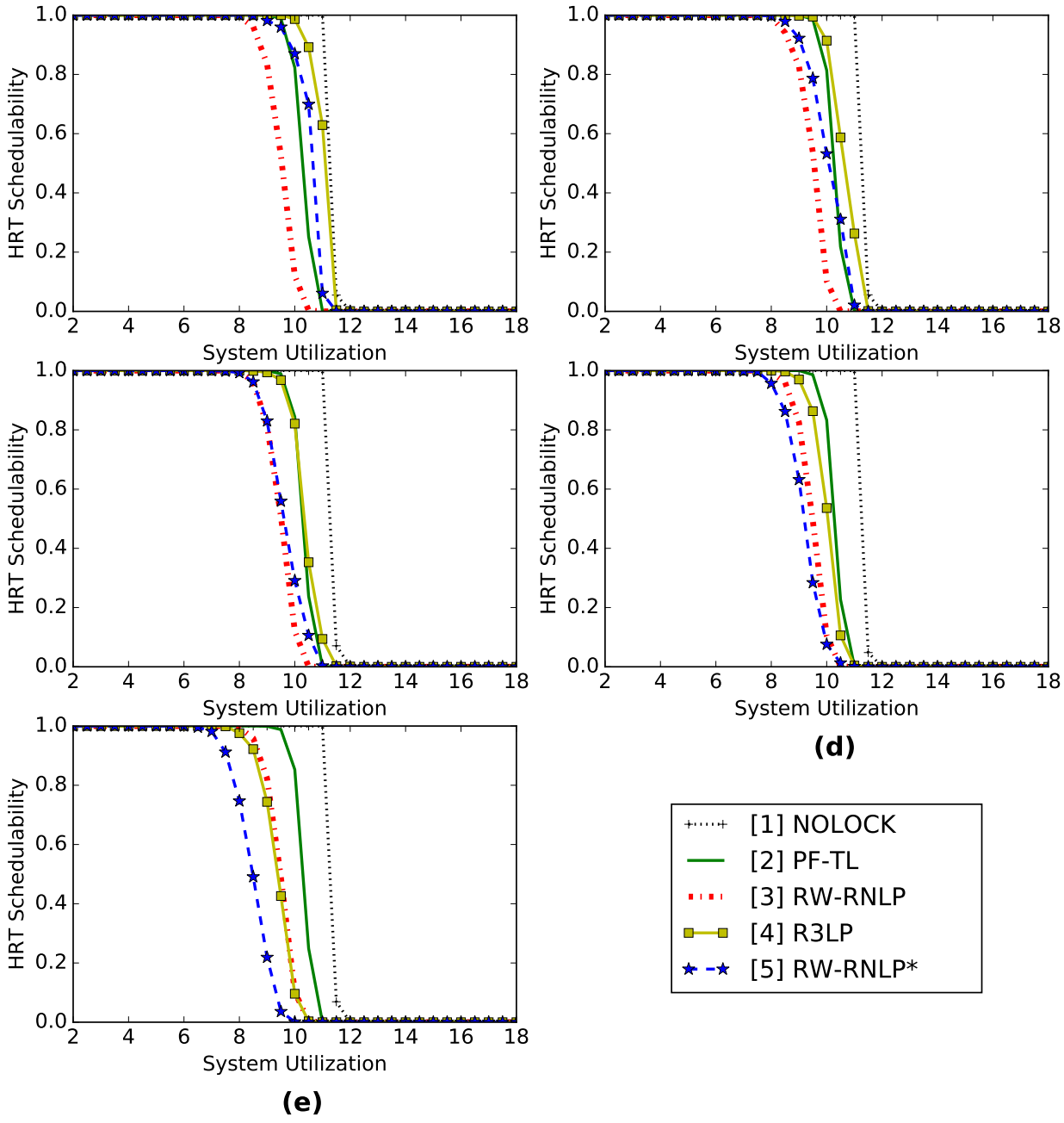



Figure 572: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, short critical-section lengths, and read probability 0.5 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

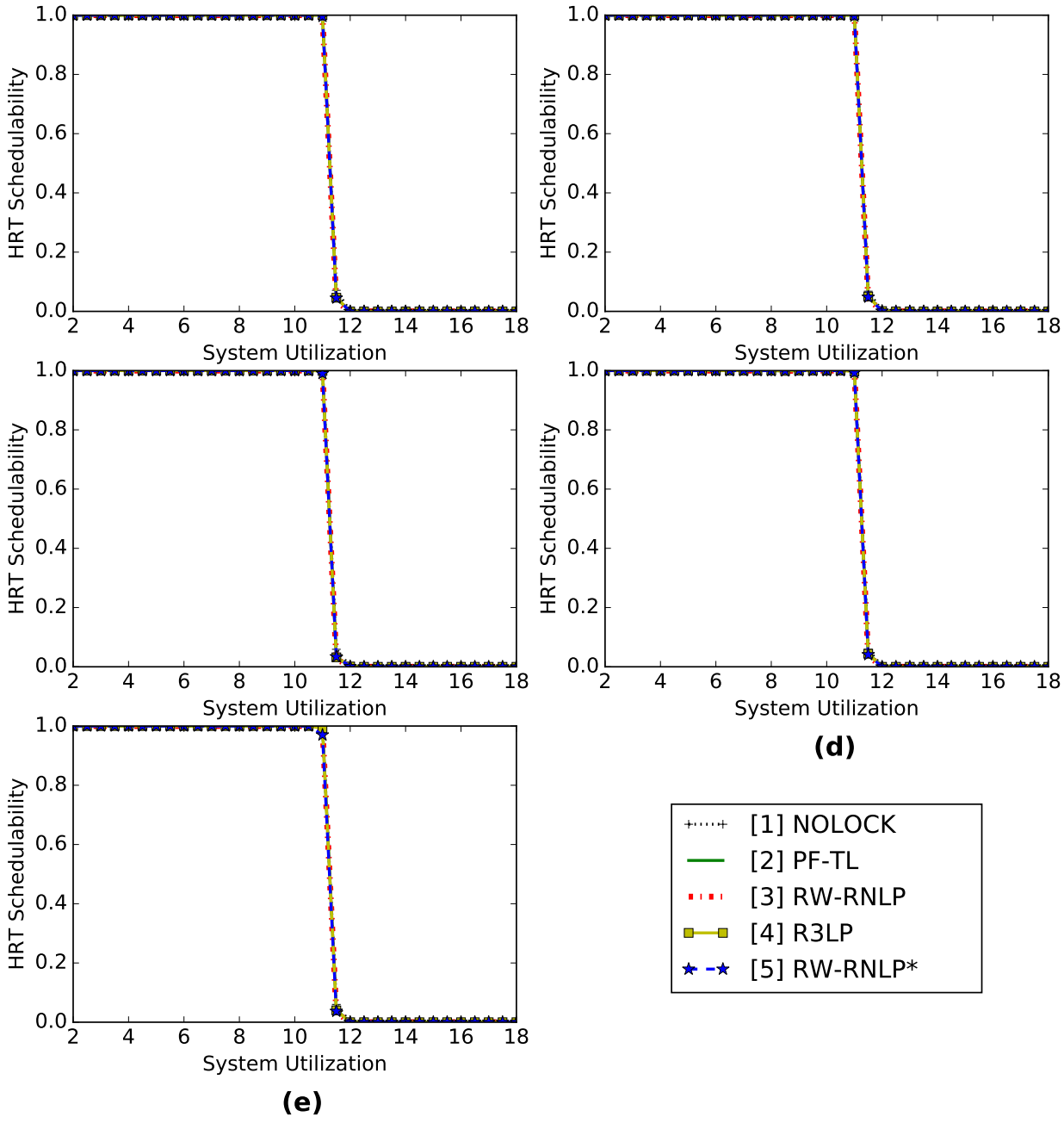



Figure 573: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, short critical-section lengths, and read probability 0.8 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

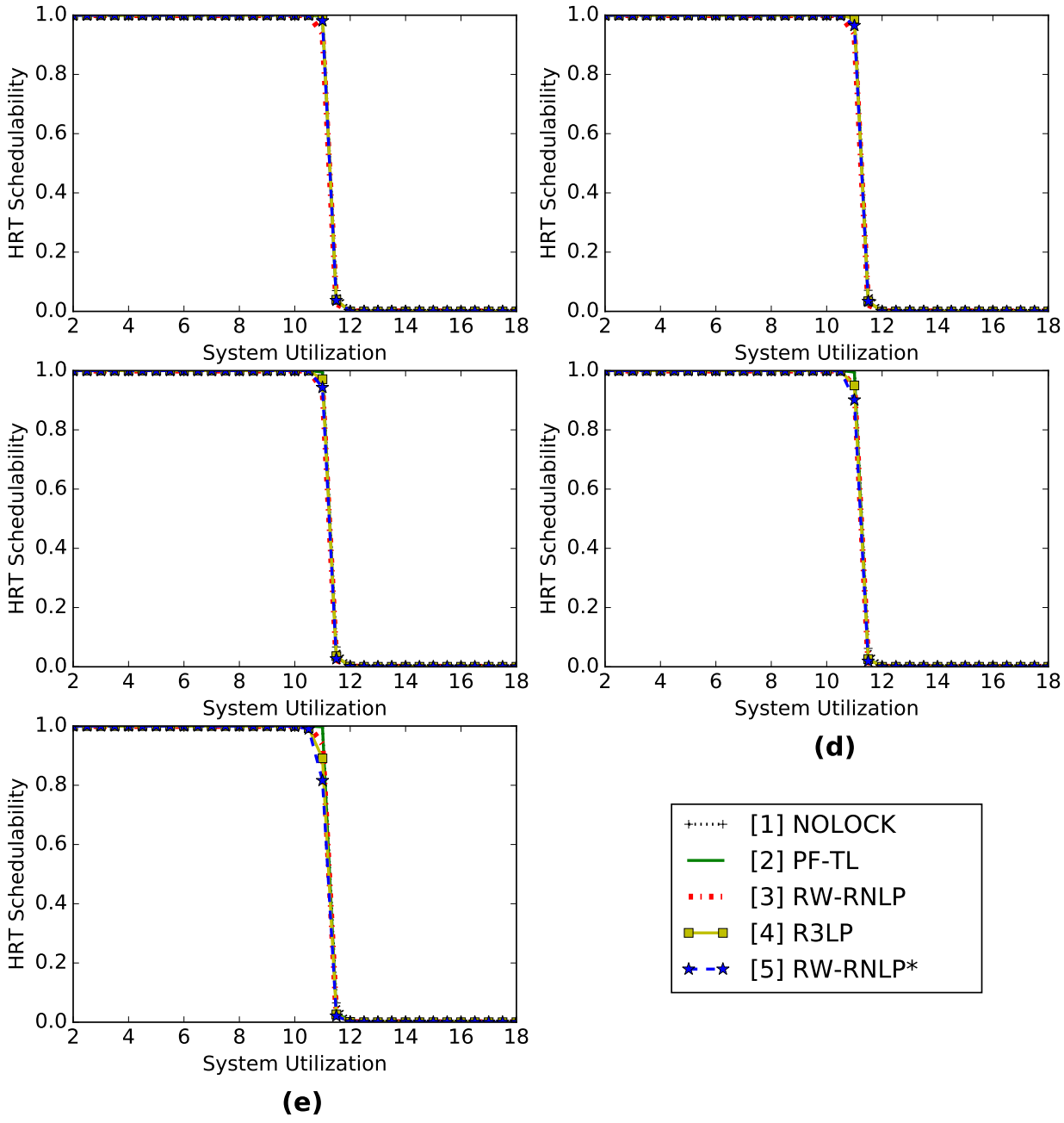



Figure 574: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, short critical-section lengths, and read probability 0.8 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

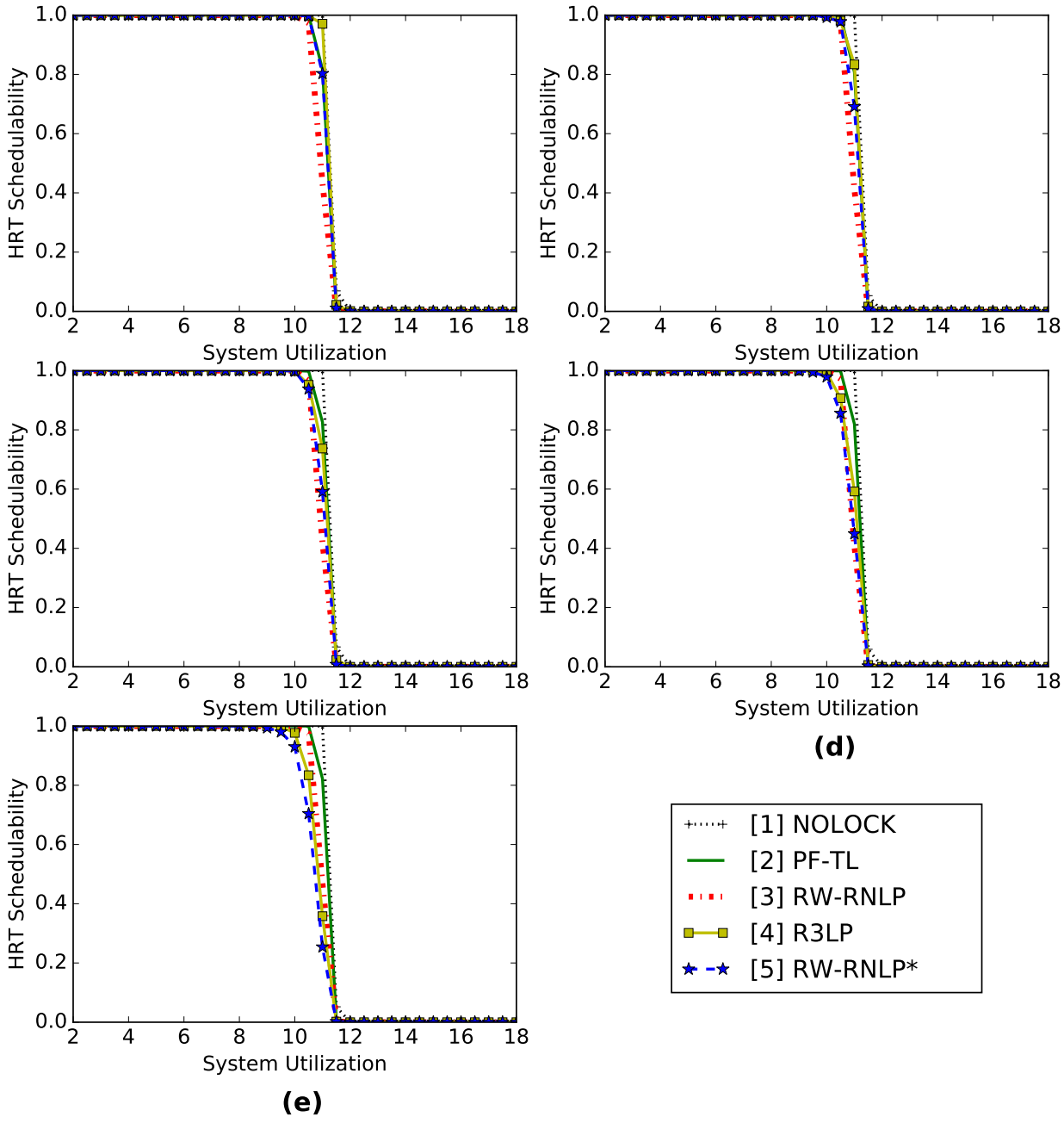



Figure 575: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, short critical-section lengths, and read probability 0.8 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

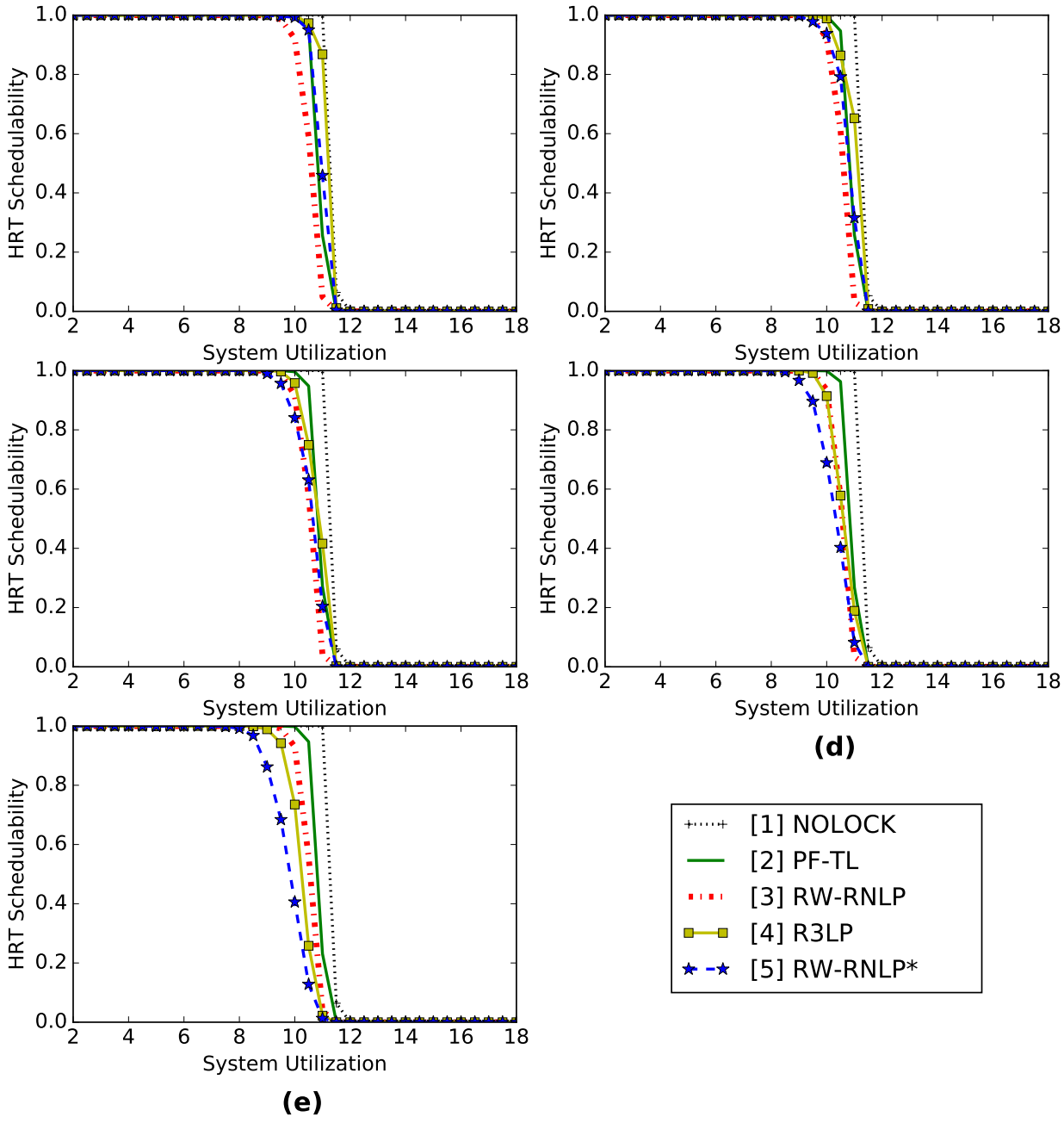



Figure 576: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, short critical-section lengths, and read probability 0.8 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

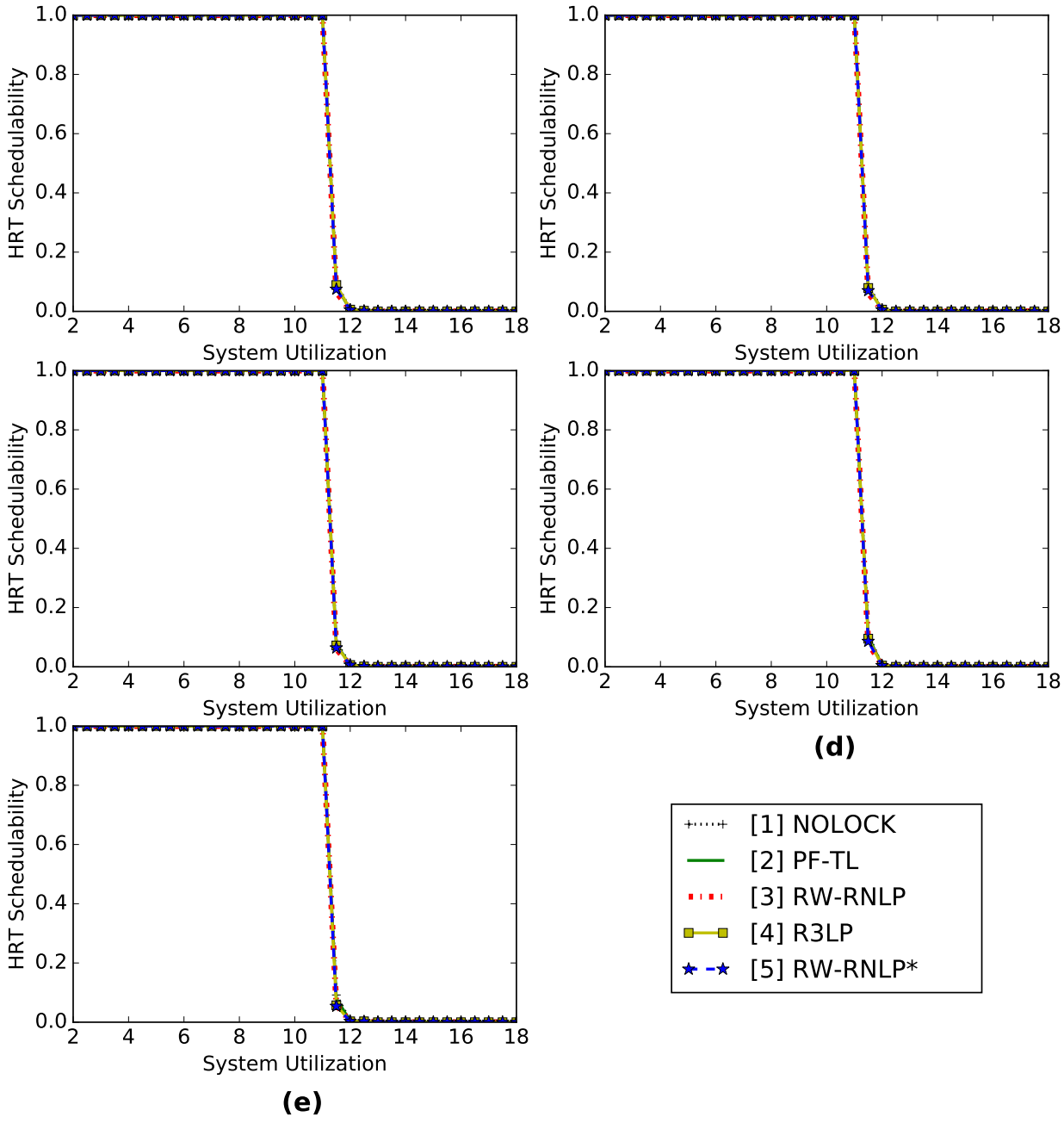



Figure 577: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, short critical-section lengths, and read probability 0 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

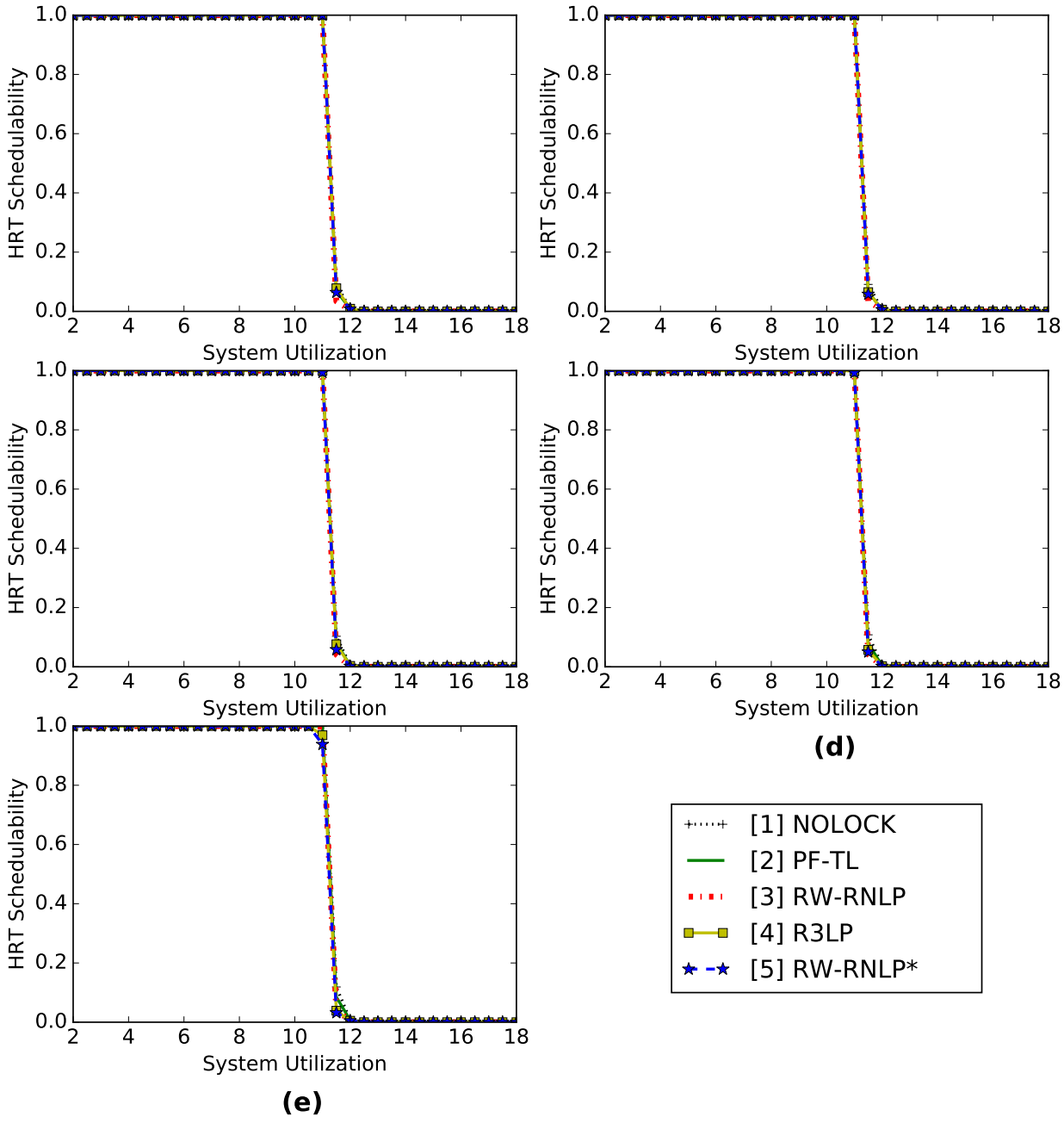



Figure 578: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, short critical-section lengths, and read probability 0 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

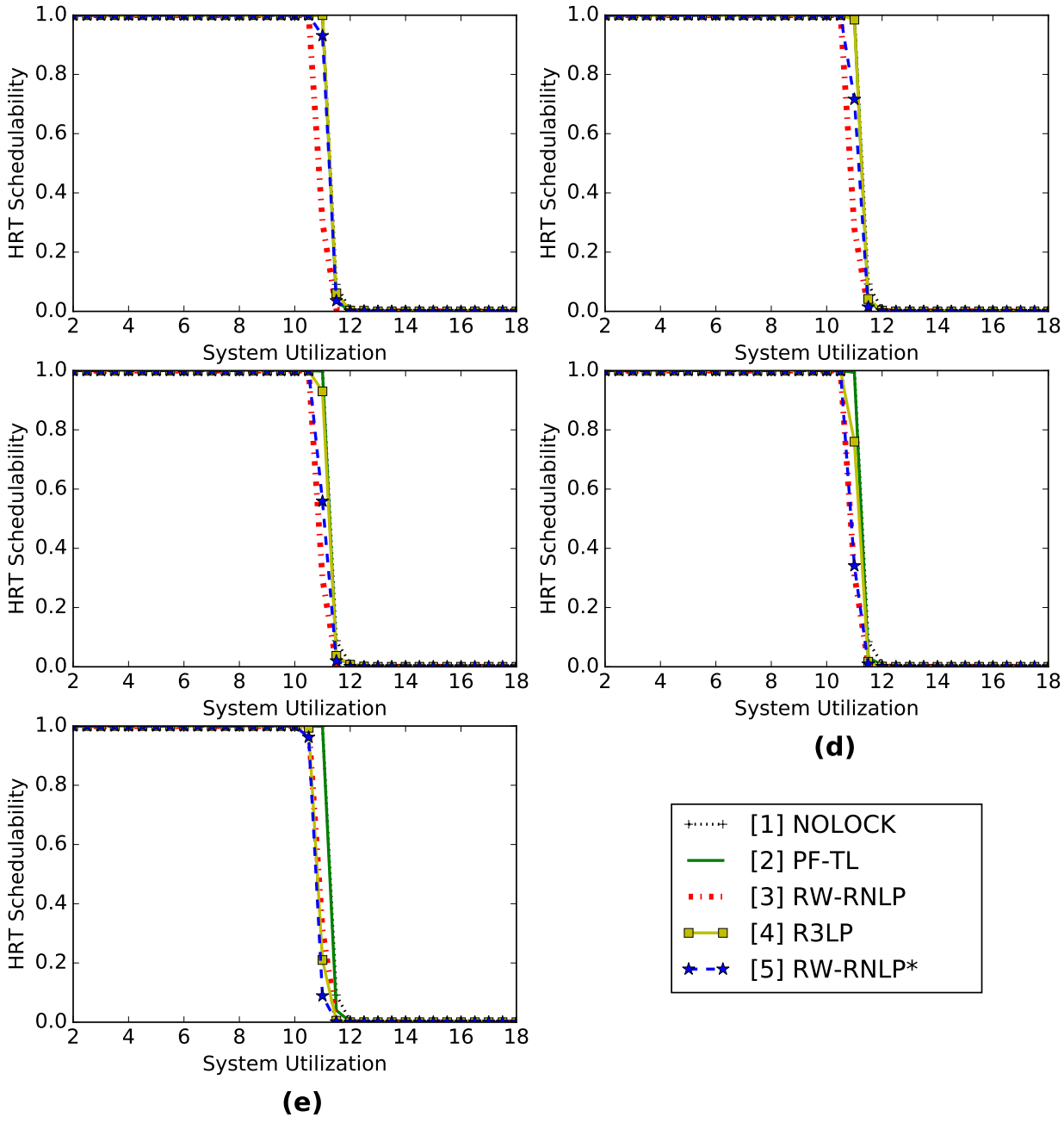



Figure 579: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, short critical-section lengths, and read probability 0 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

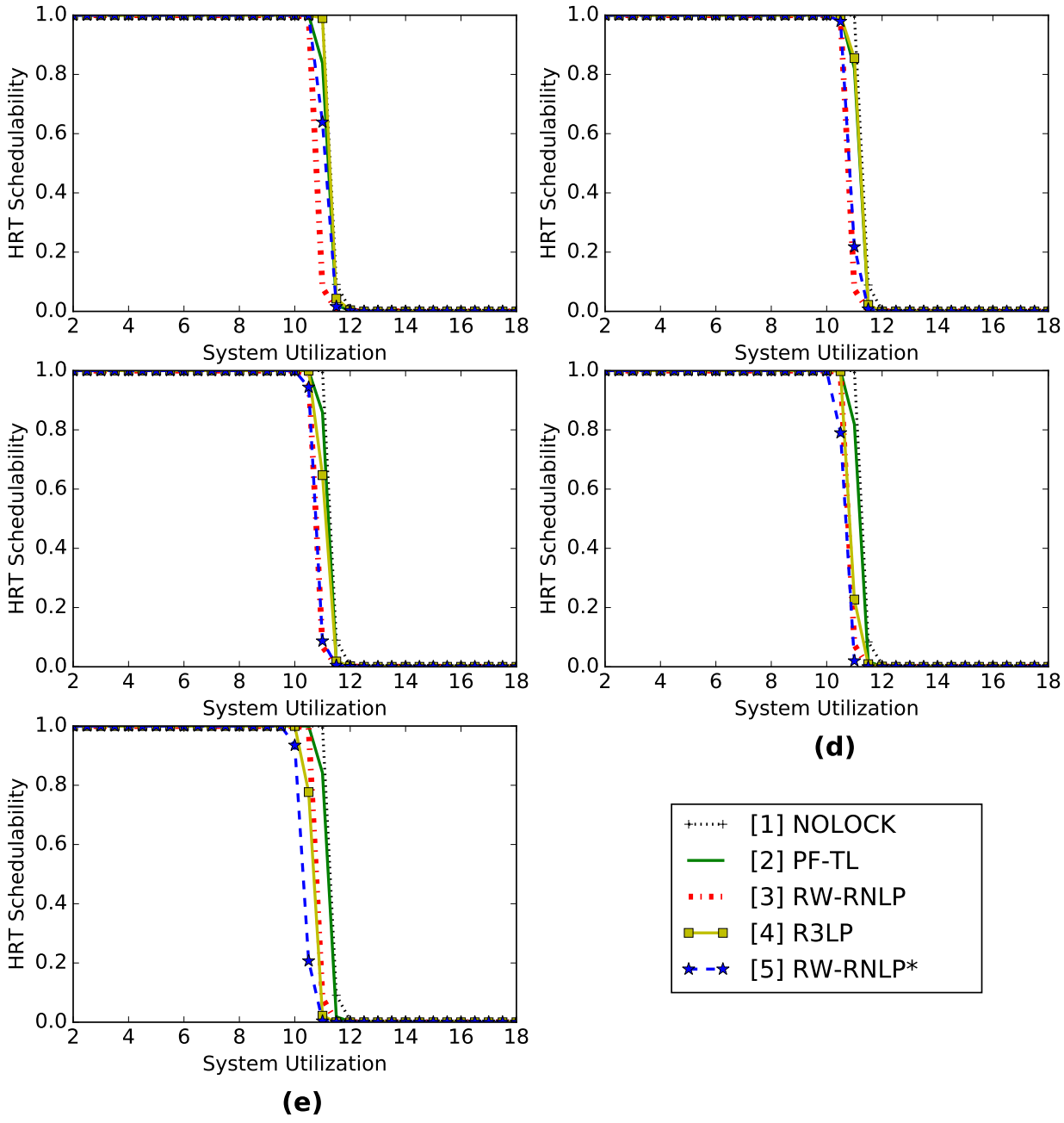



Figure 580: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, short critical-section lengths, and read probability 0 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

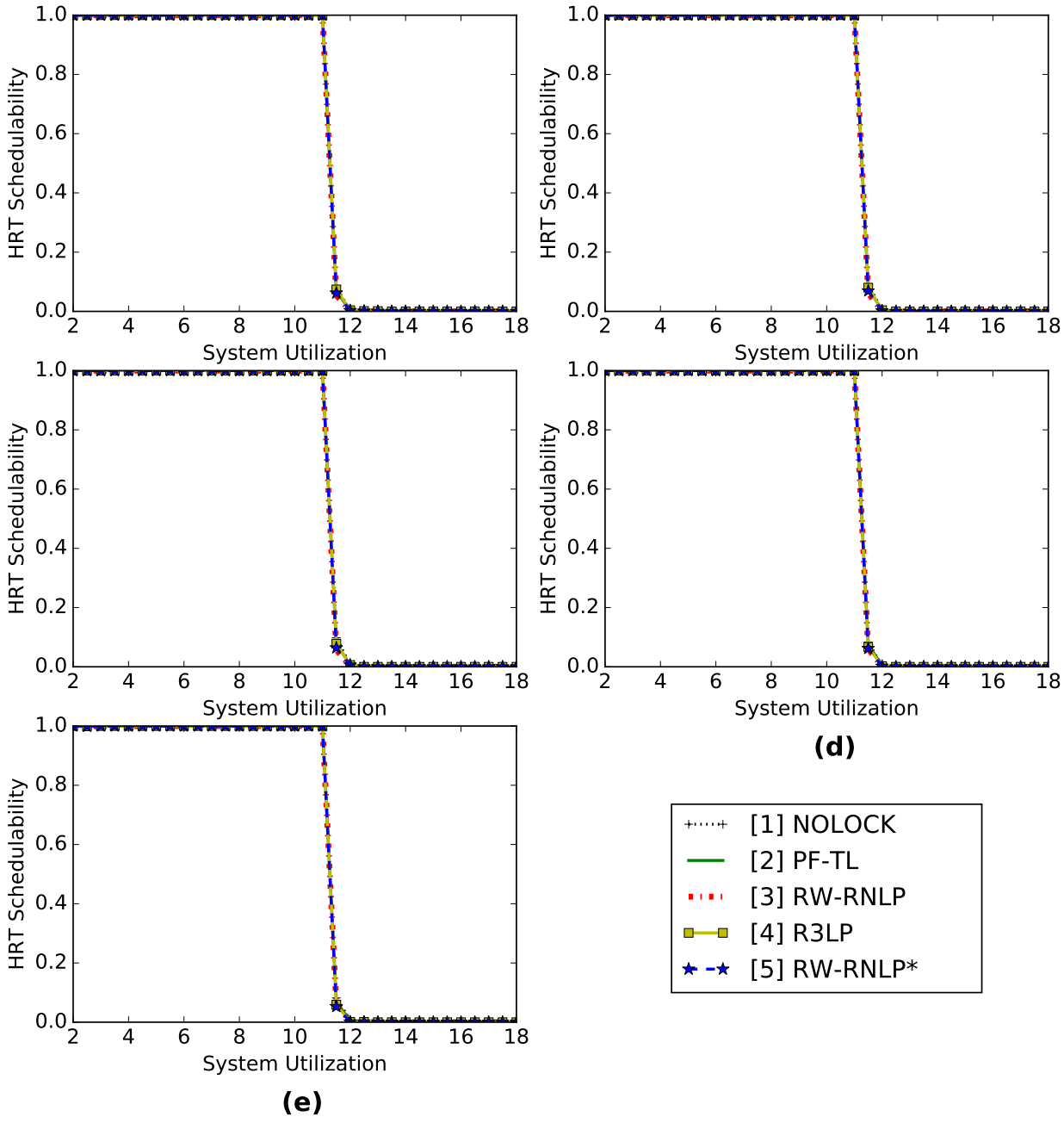



Figure 581: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, short critical-section lengths, and read probability 0.2 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

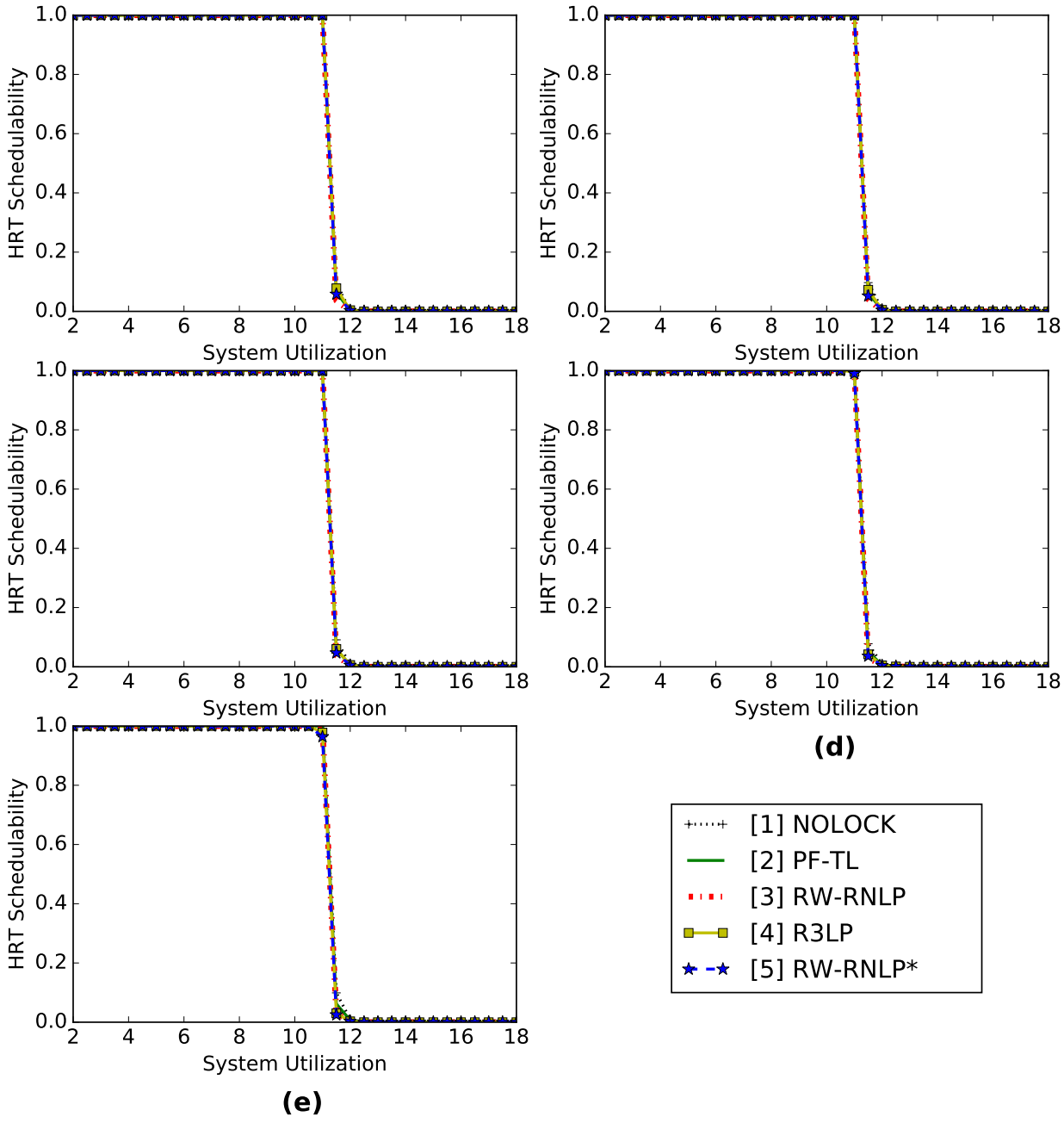



Figure 582: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, short critical-section lengths, and read probability 0.2 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

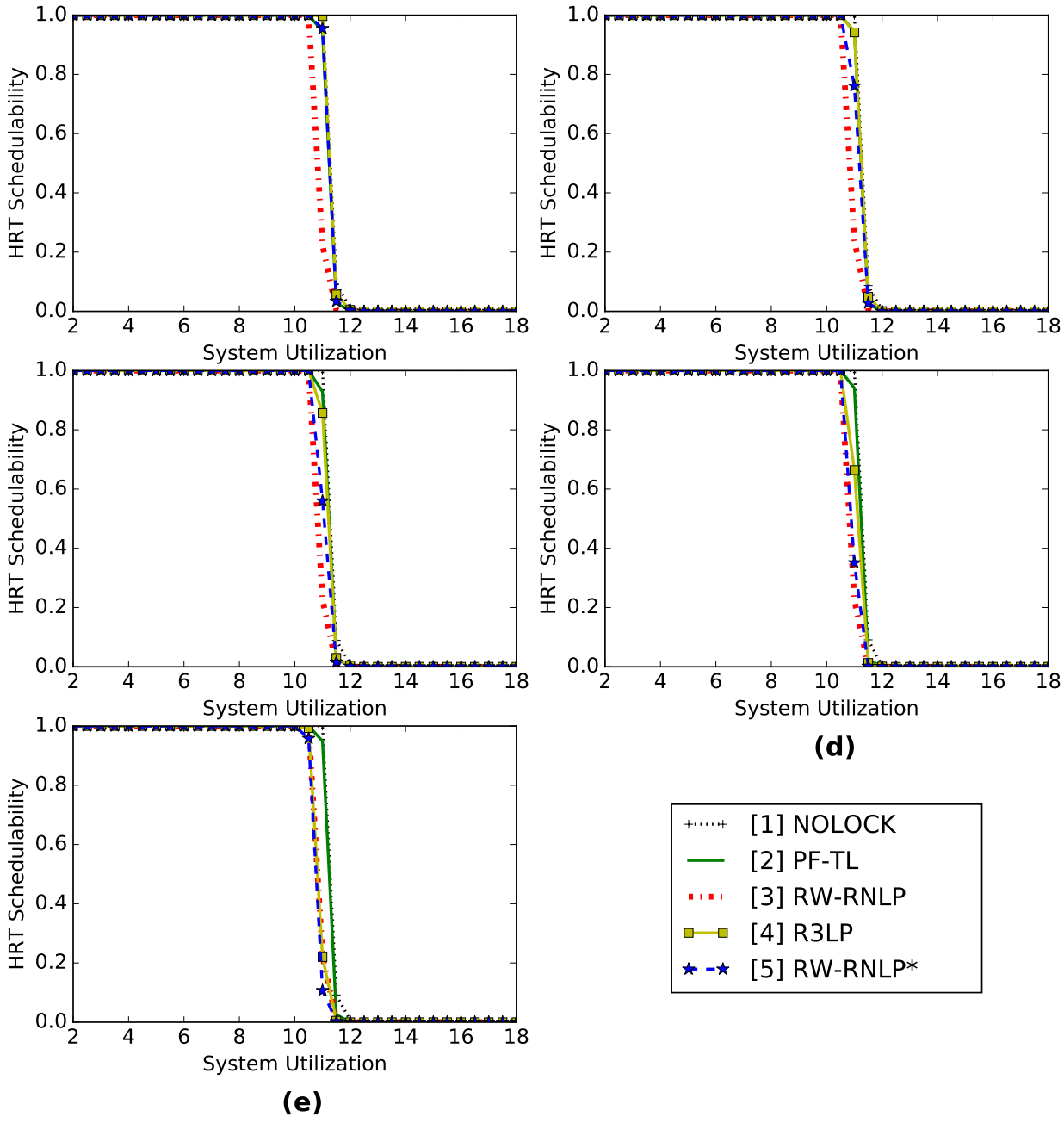



Figure 583: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, short critical-section lengths, and read probability 0.2 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

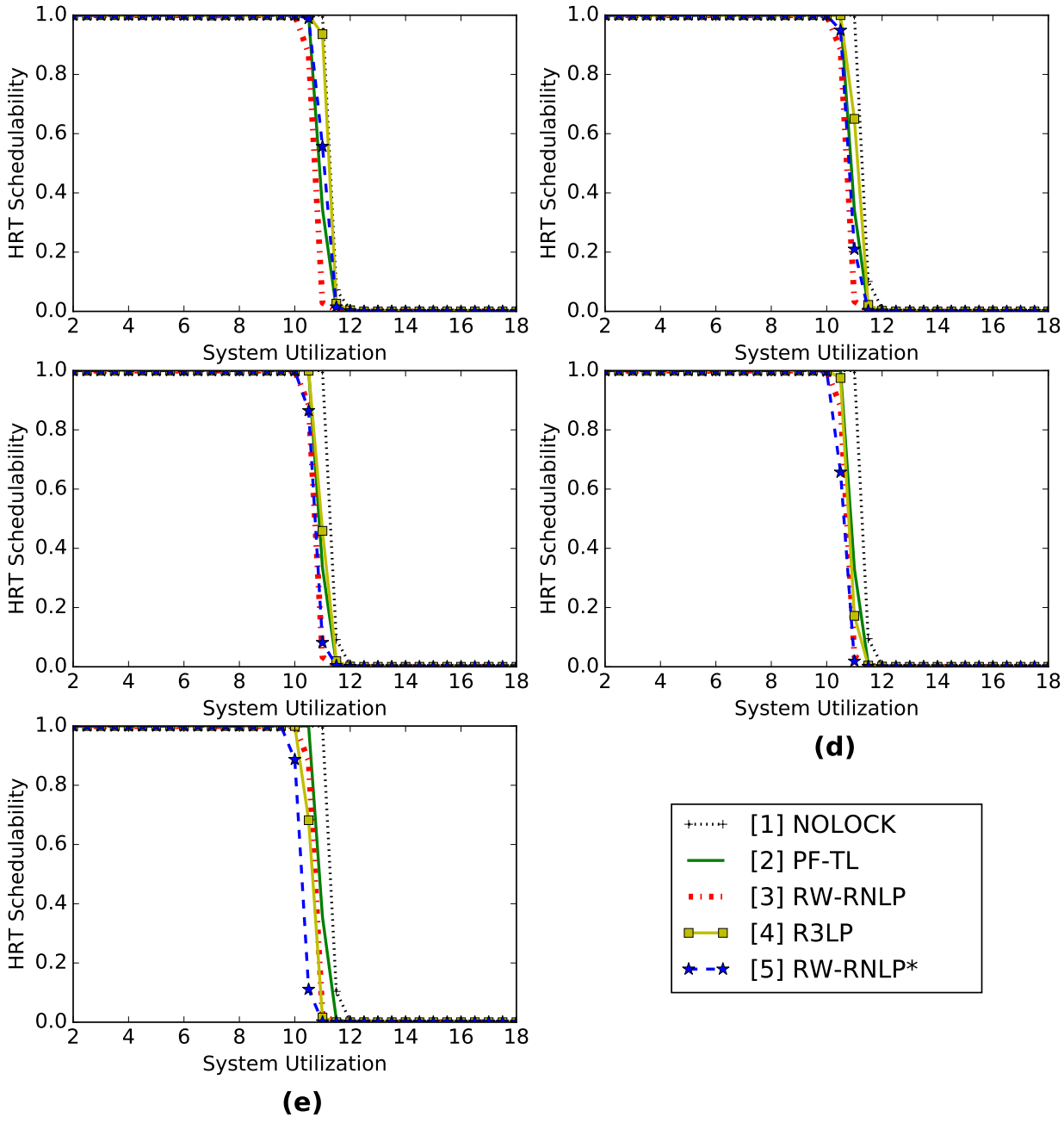



Figure 584: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, short critical-section lengths, and read probability 0.2 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

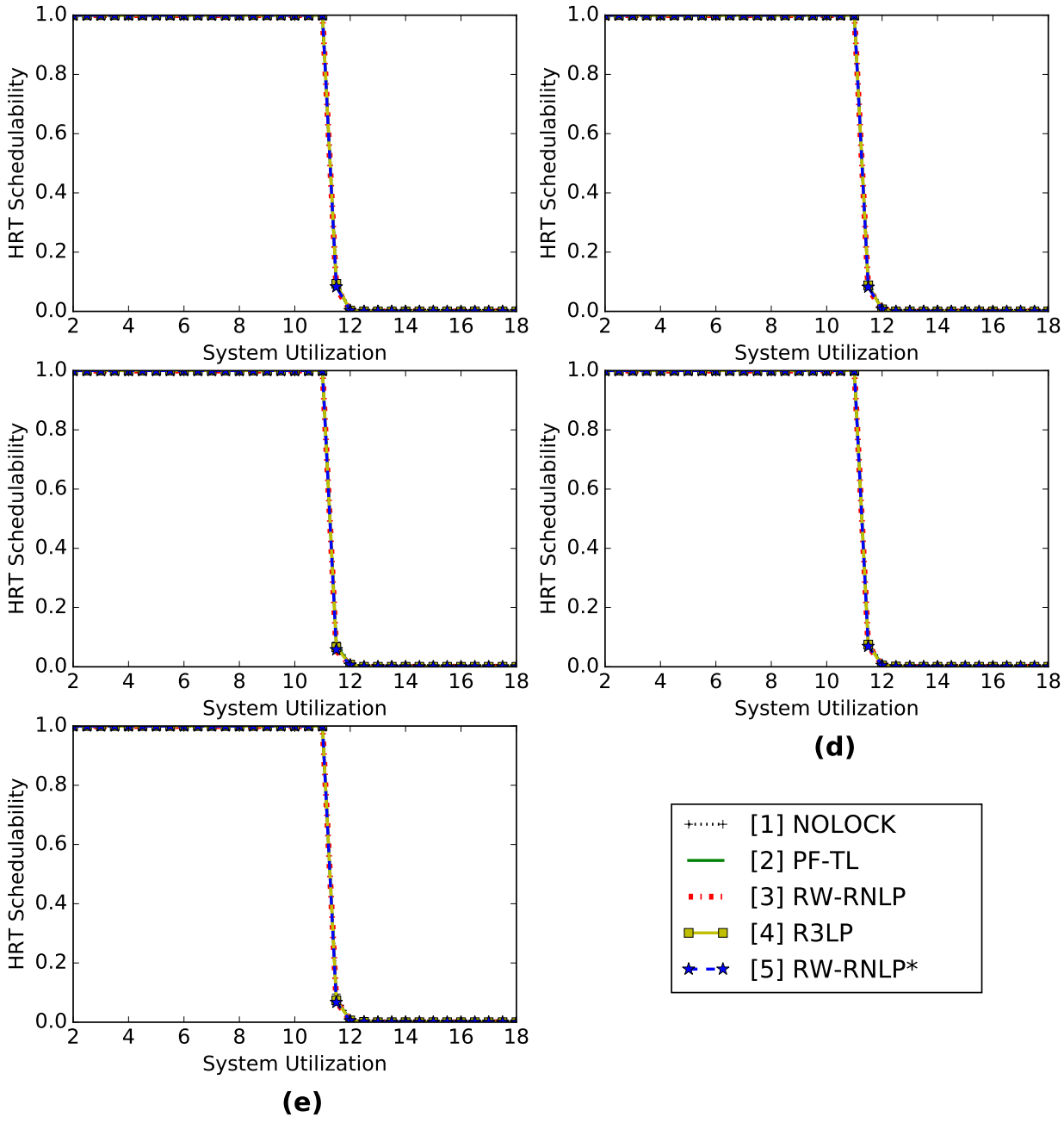



Figure 585: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, short critical-section lengths, and read probability 0.5 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

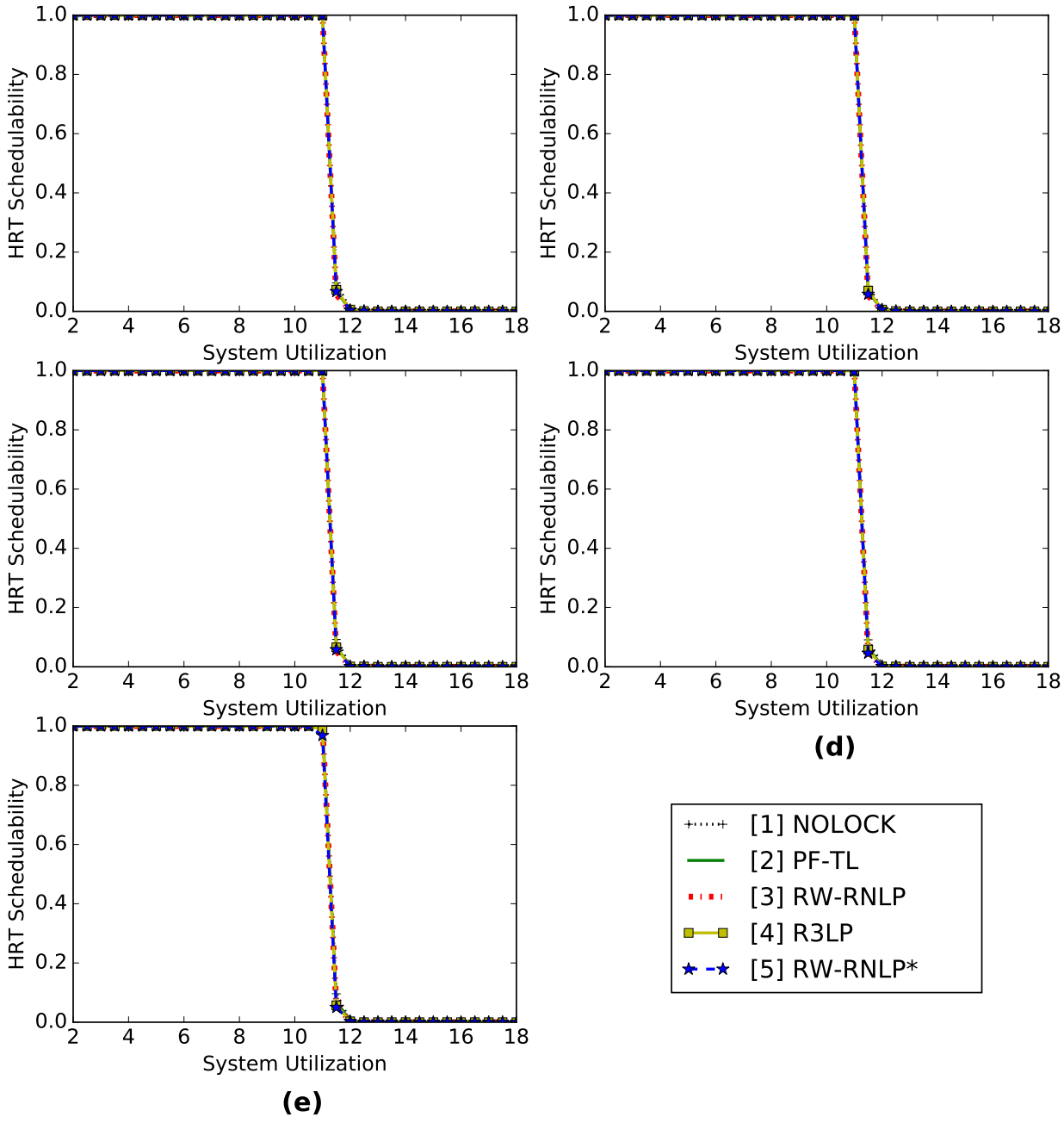



Figure 586: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, short critical-section lengths, and read probability 0.5 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

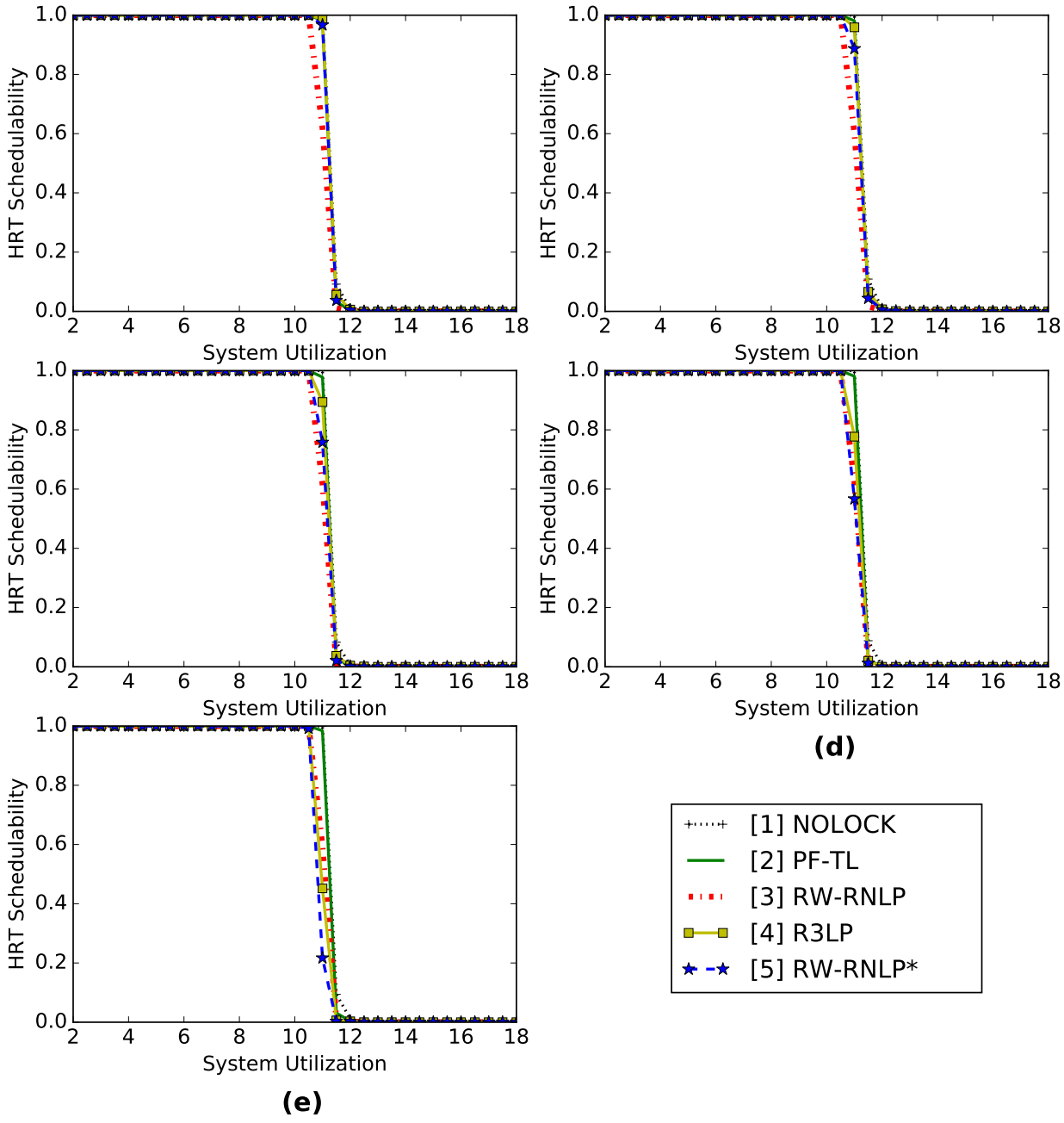



Figure 587: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, short critical-section lengths, and read probability 0.5 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

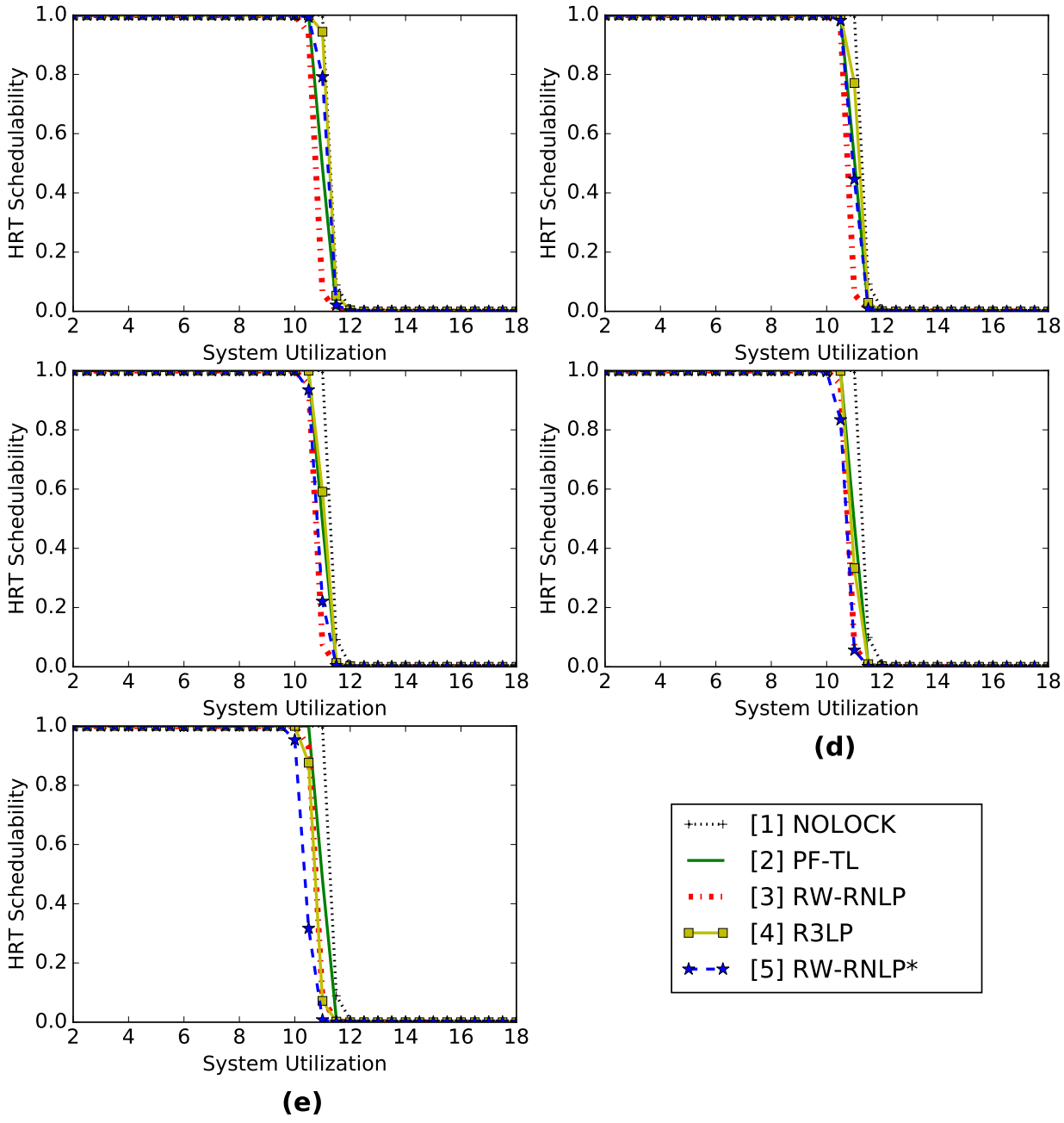



Figure 588: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, short critical-section lengths, and read probability 0.5 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

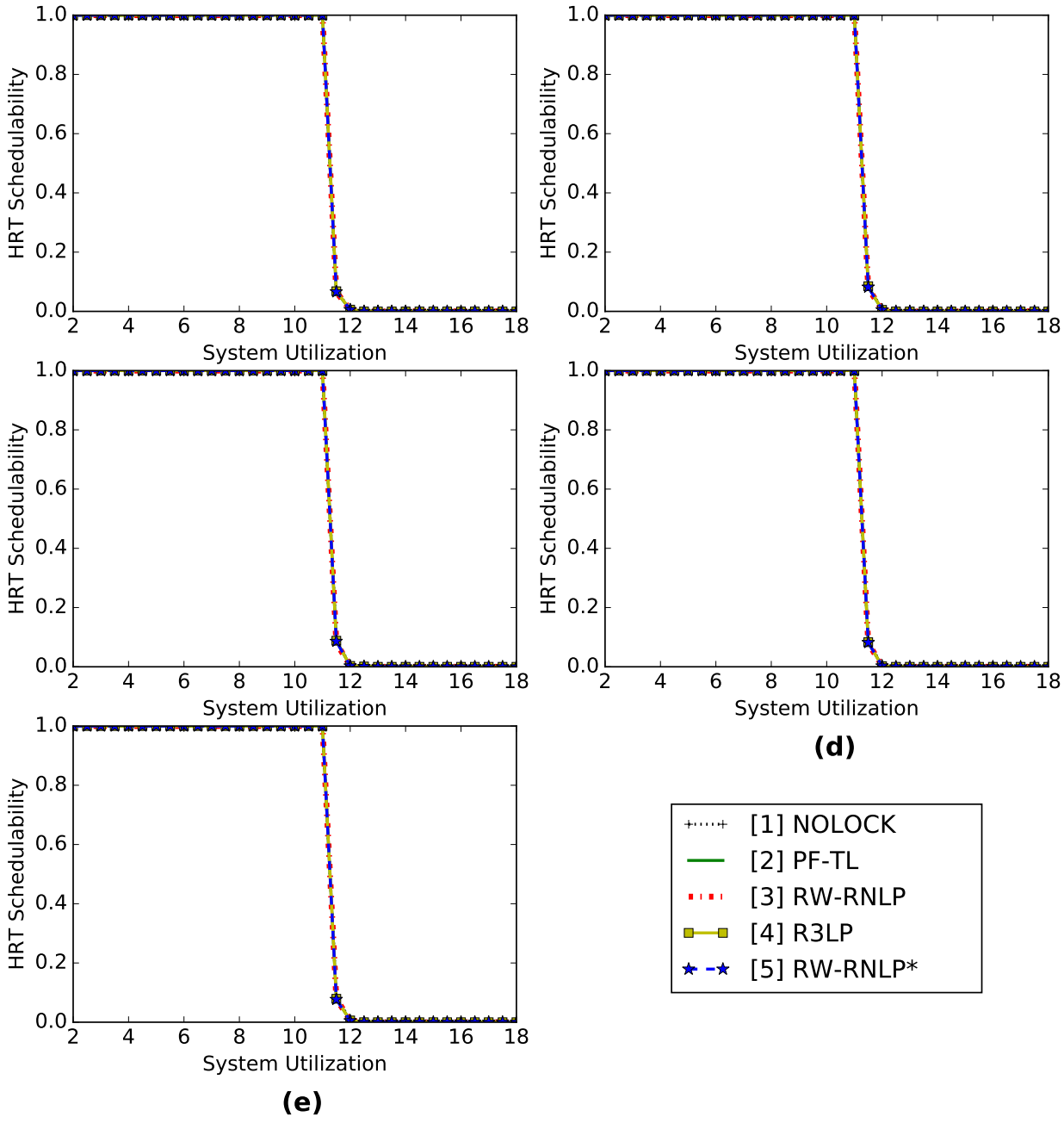



Figure 589: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, short critical-section lengths, and read probability 0.8 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

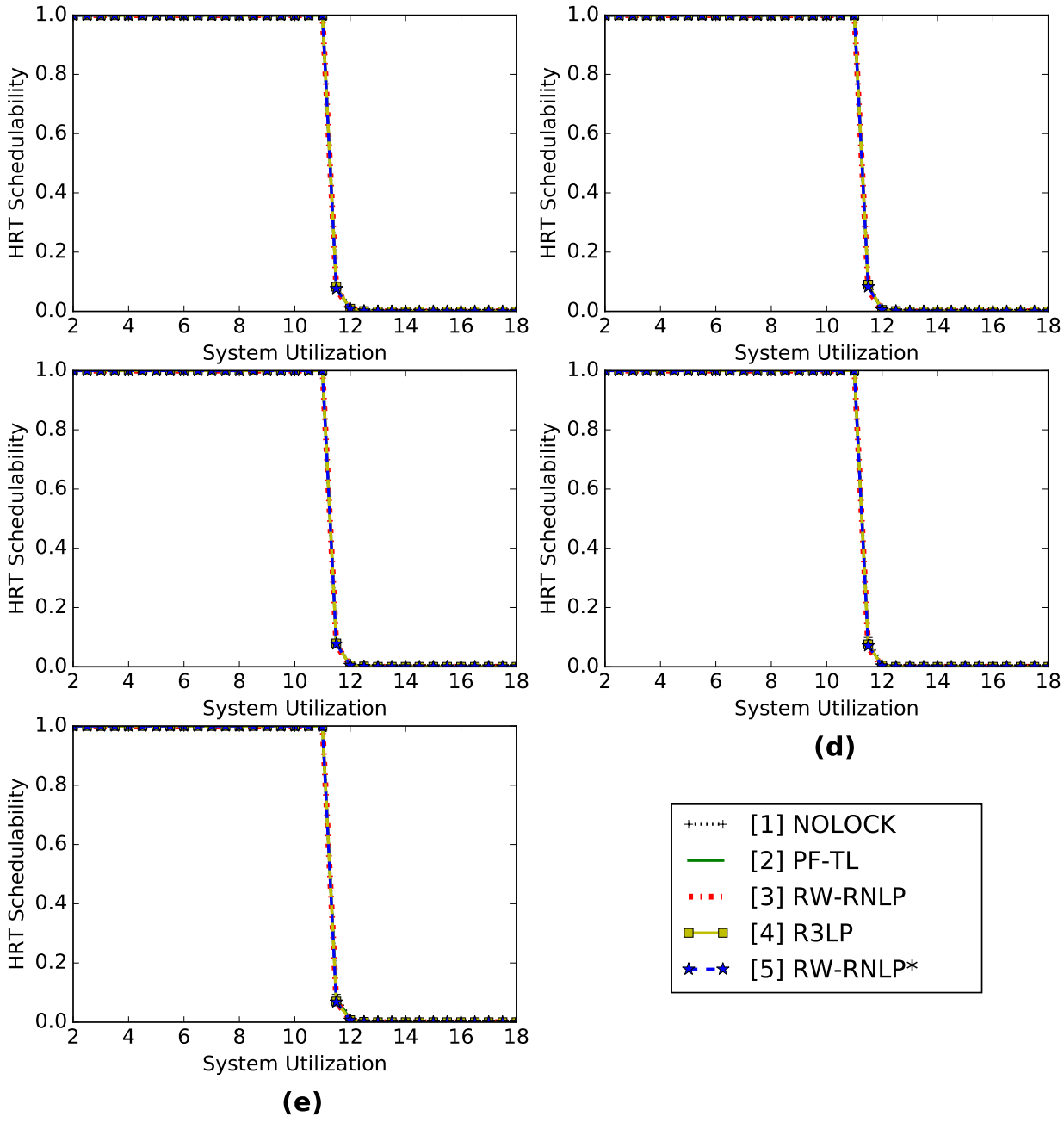



Figure 590: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, short critical-section lengths, and read probability 0.8 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

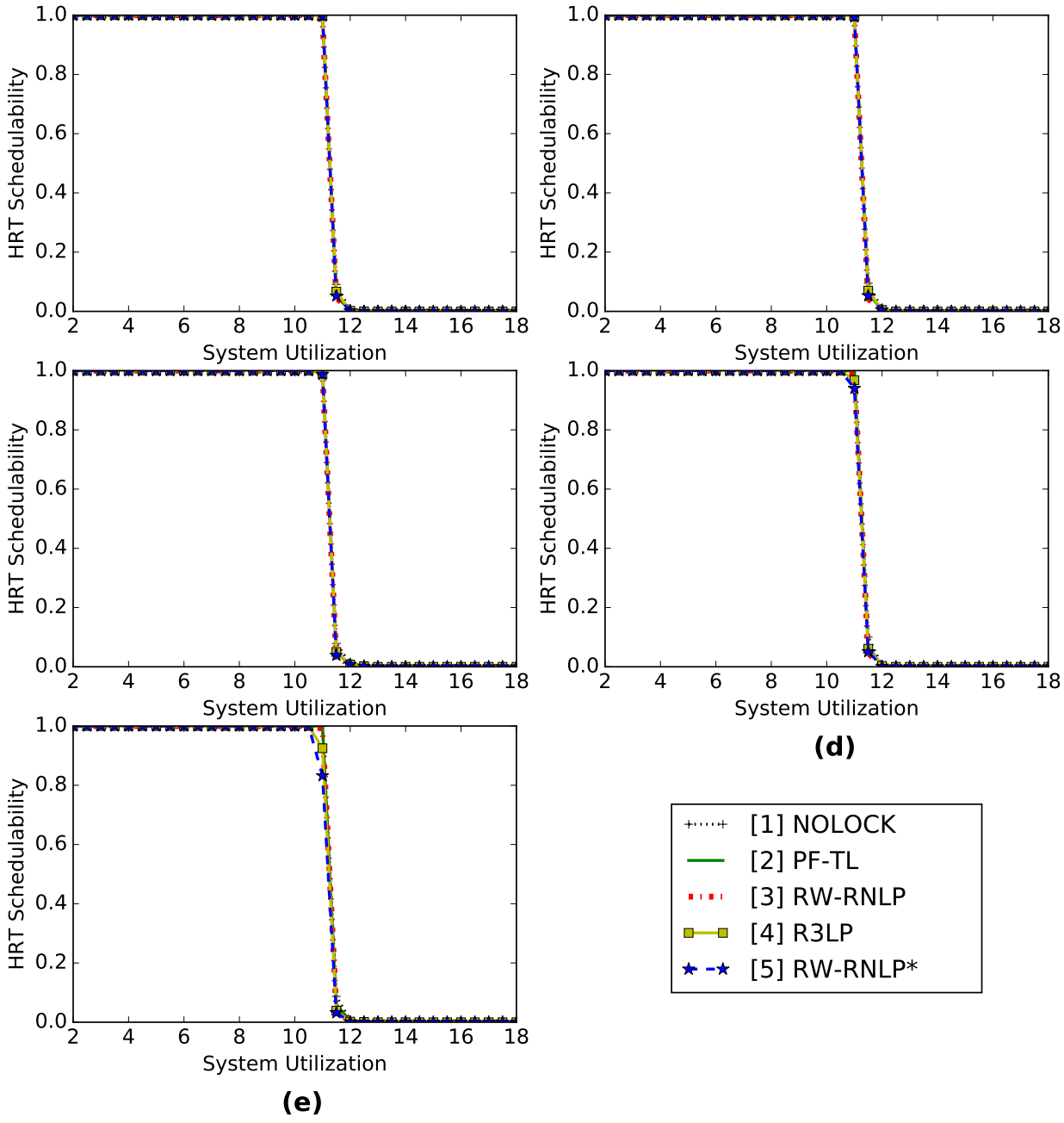



Figure 591: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, short critical-section lengths, and read probability 0.8 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

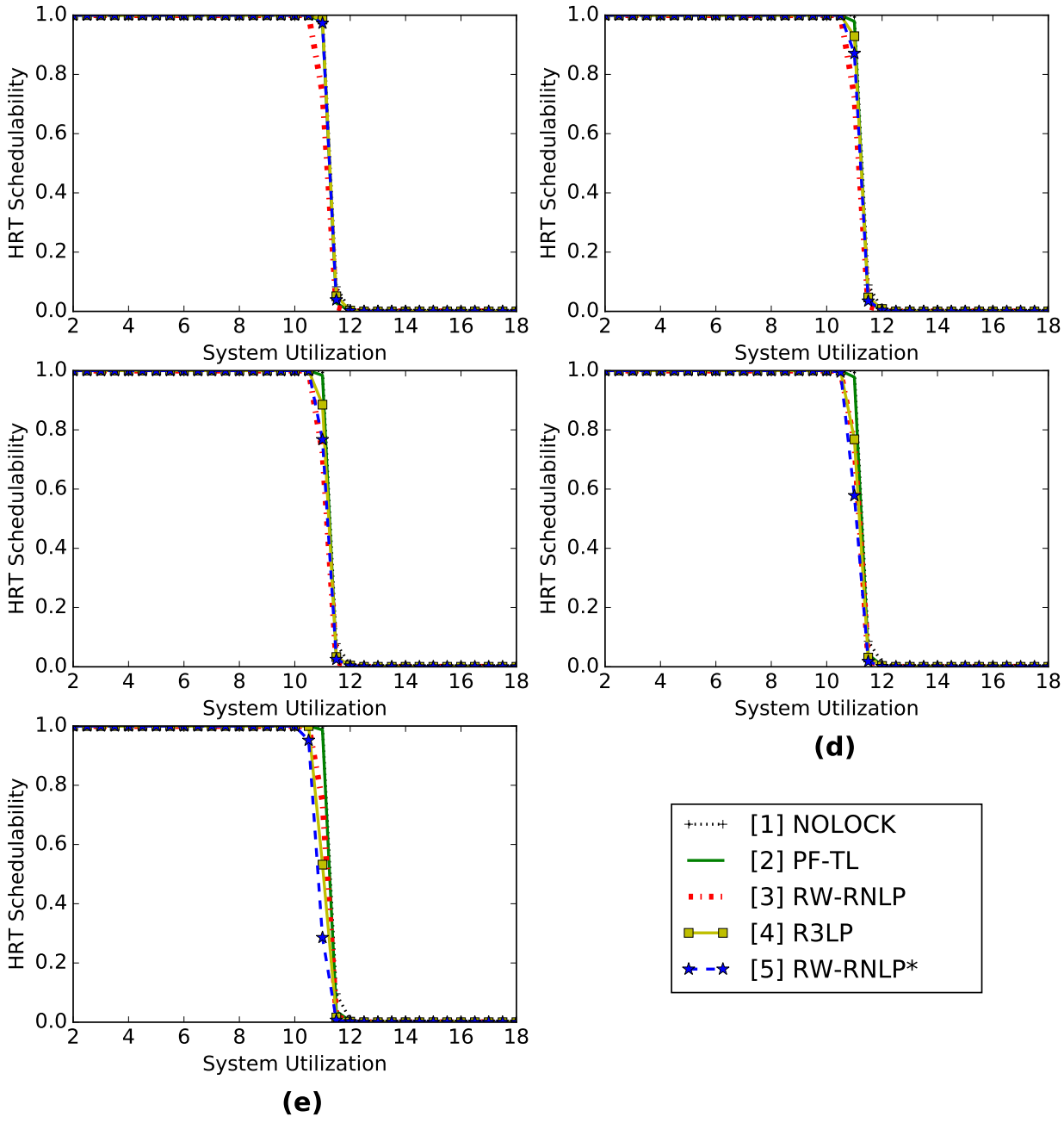



Figure 592: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, short critical-section lengths, and read probability 0.8 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

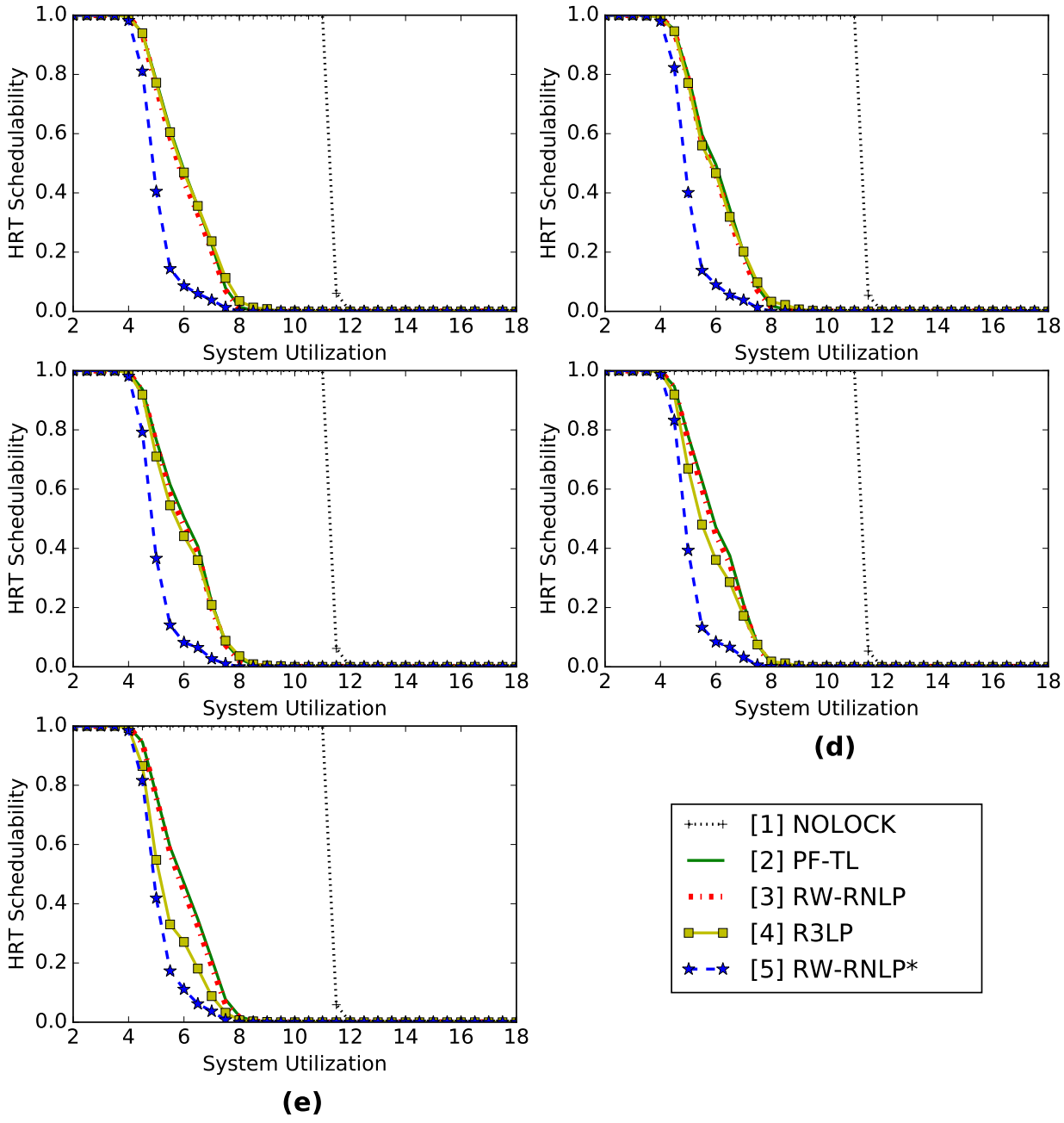



Figure 593: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, long critical-section lengths, and read probability 0 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

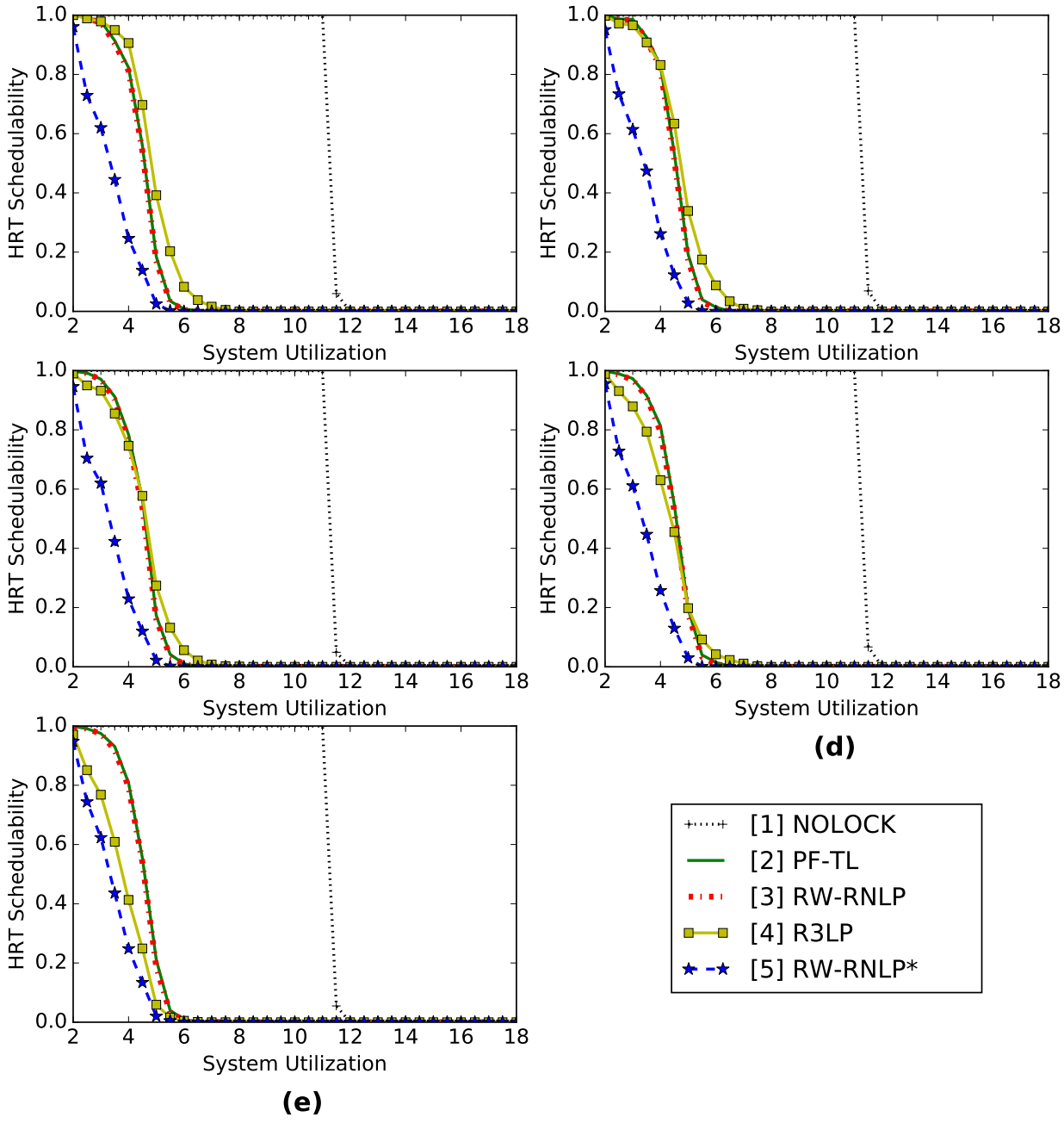



Figure 594: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, long critical-section lengths, and read probability 0 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

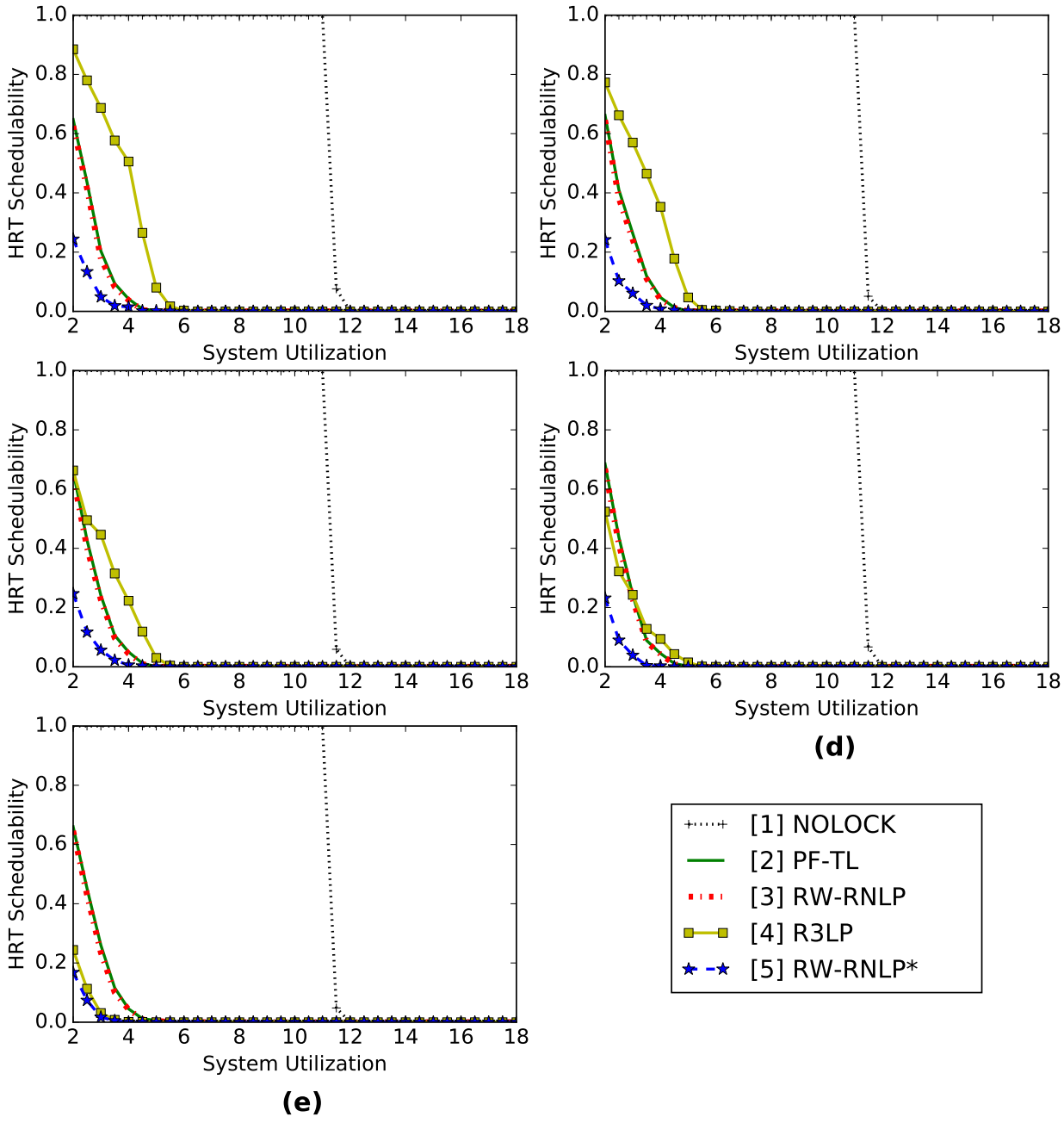



Figure 595: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, long critical-section lengths, and read probability 0 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

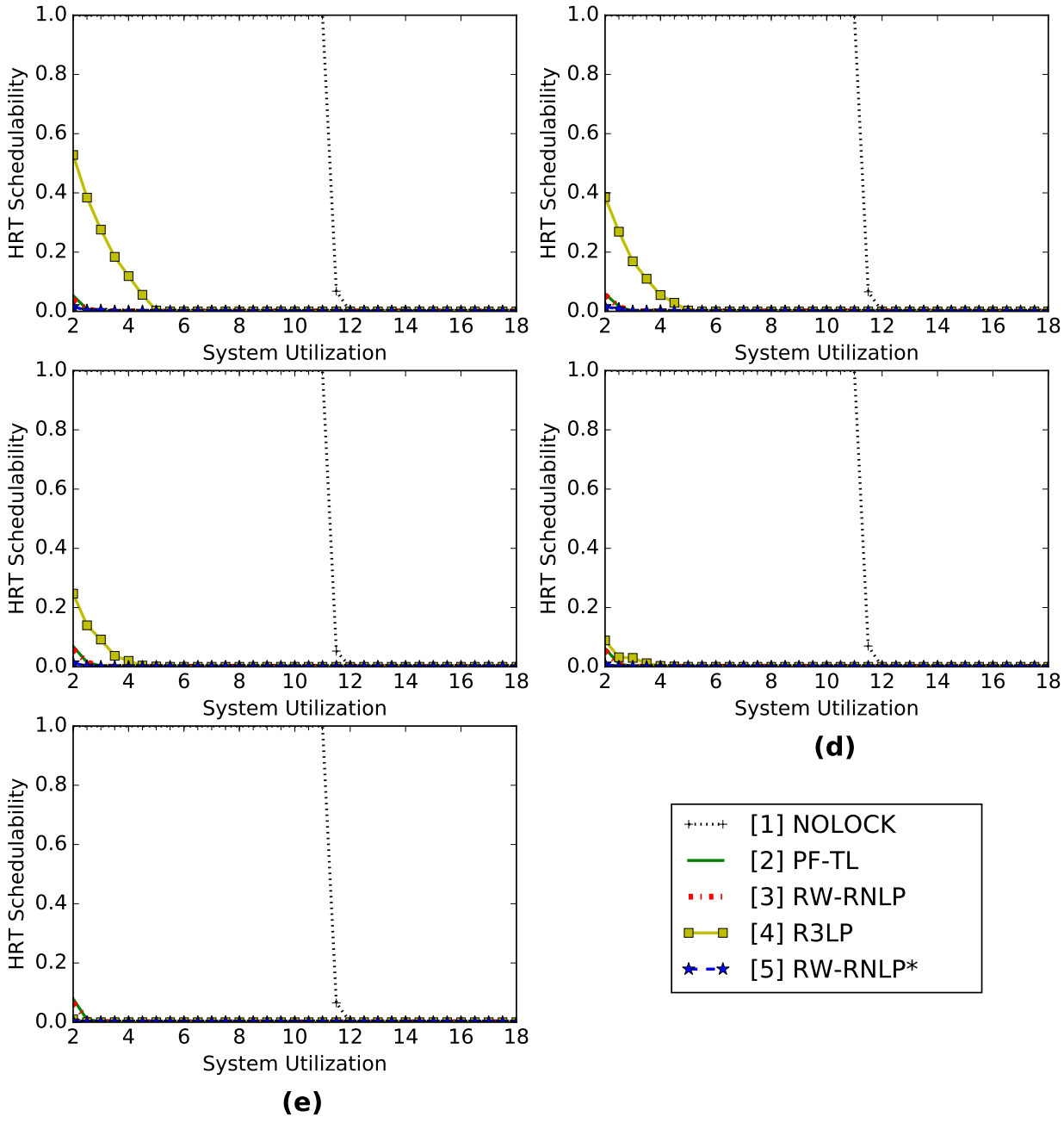



Figure 596: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, long critical-section lengths, and read probability 0 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

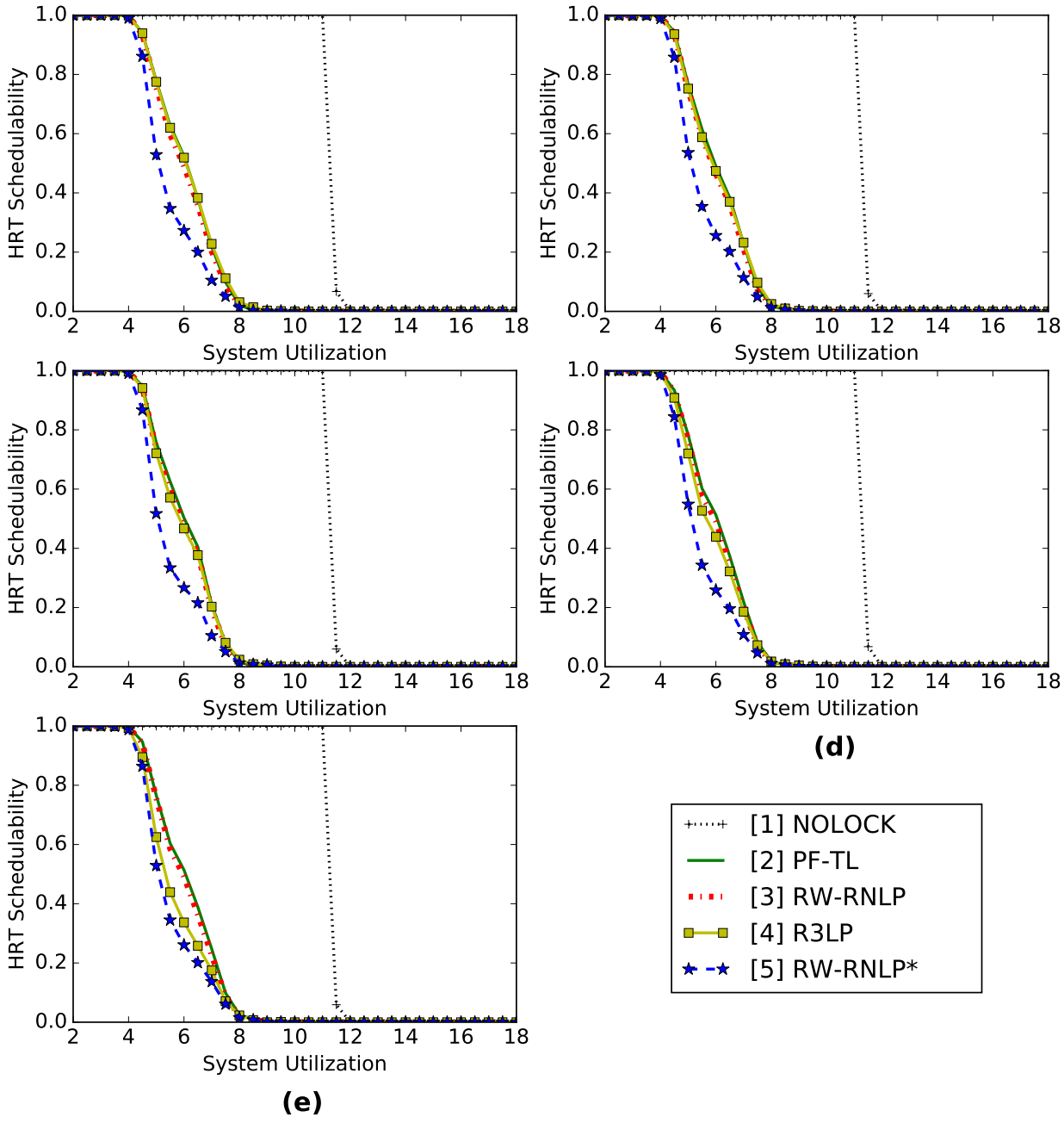



Figure 597: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, long critical-section lengths, and read probability 0.2 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

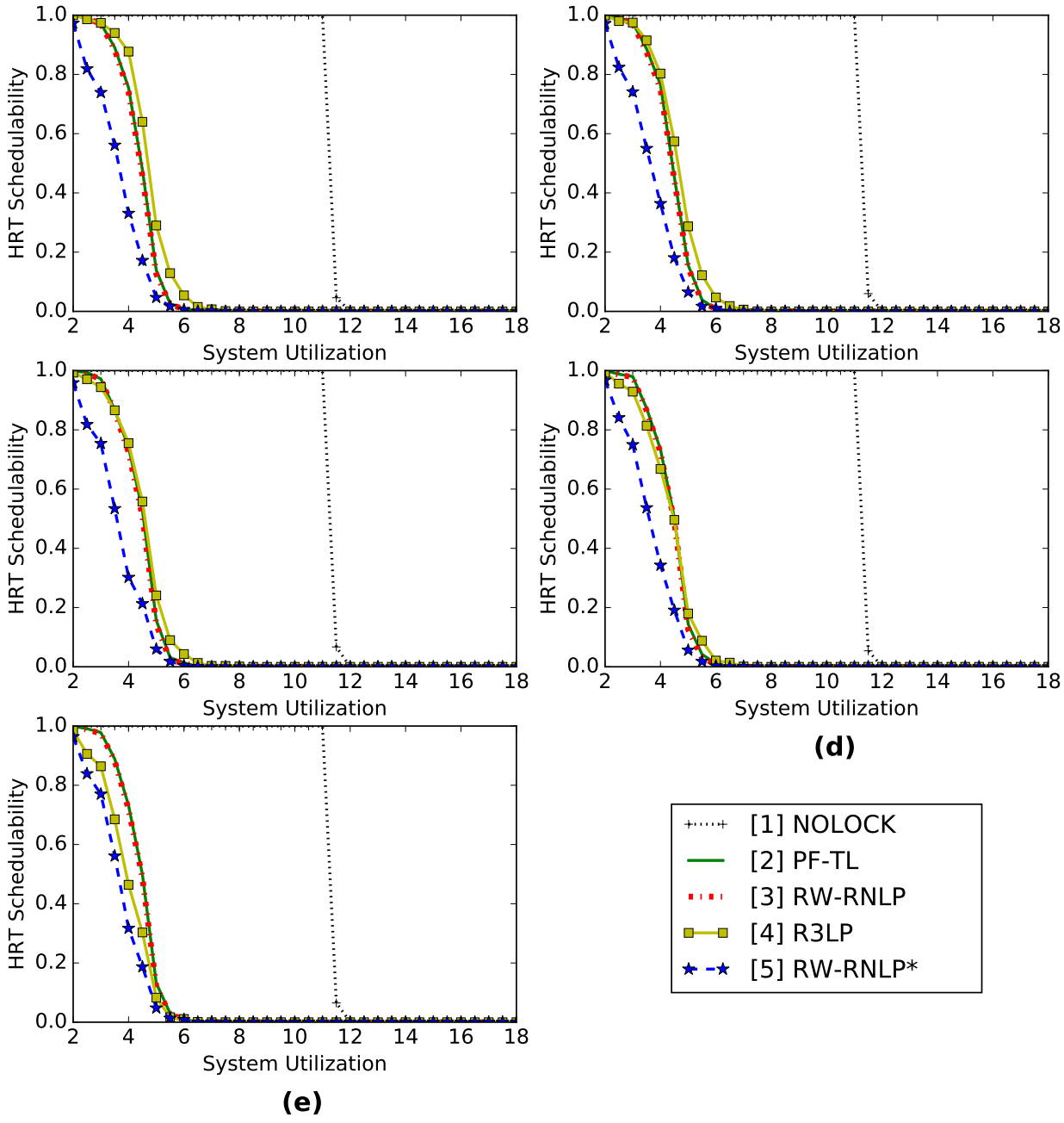



Figure 598: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, long critical-section lengths, and read probability 0.2 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

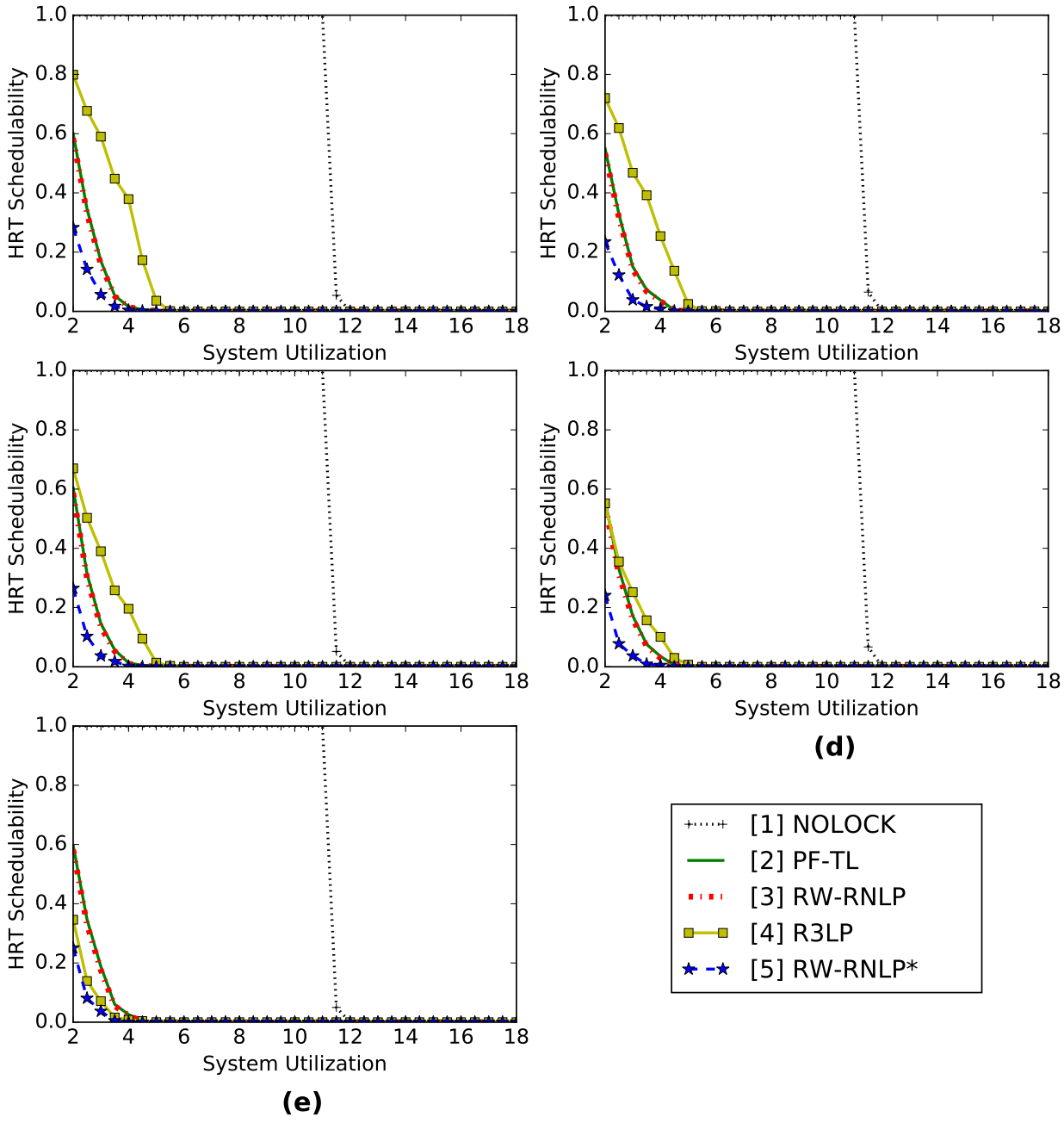



Figure 599: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, long critical-section lengths, and read probability 0.2 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

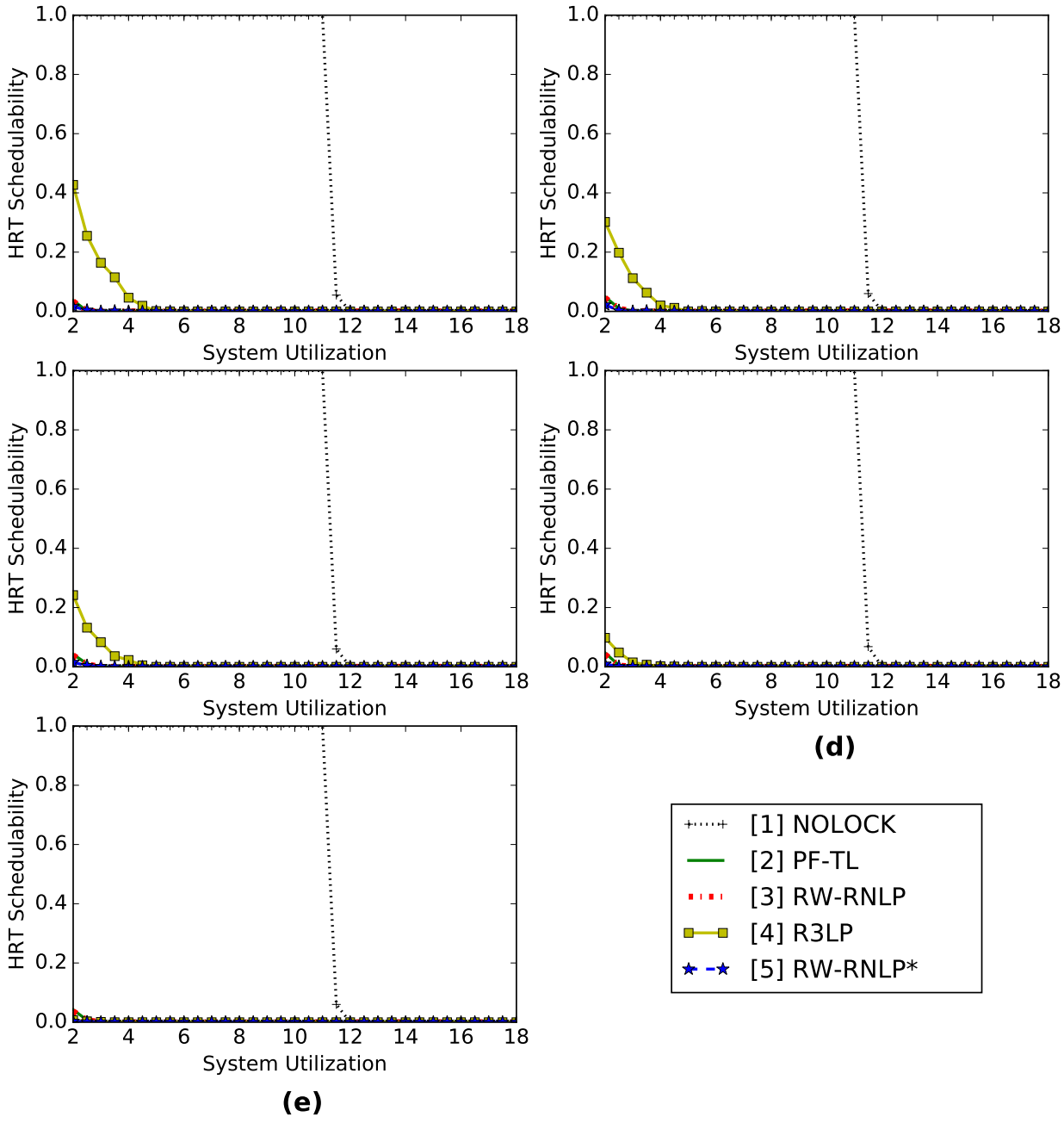



Figure 600: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, long critical-section lengths, and read probability 0.2 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

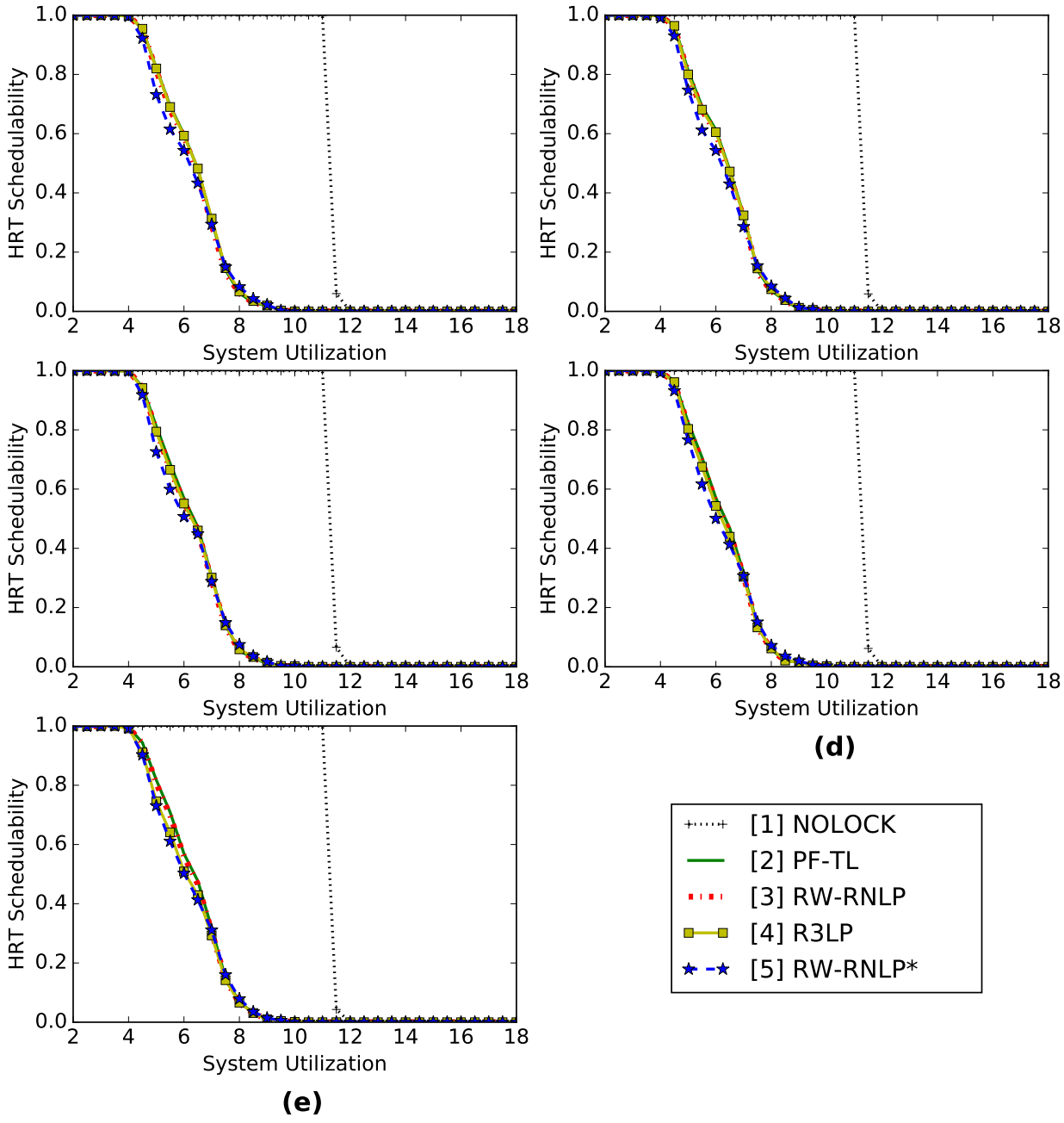



Figure 601: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, long critical-section lengths, and read probability 0.5 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

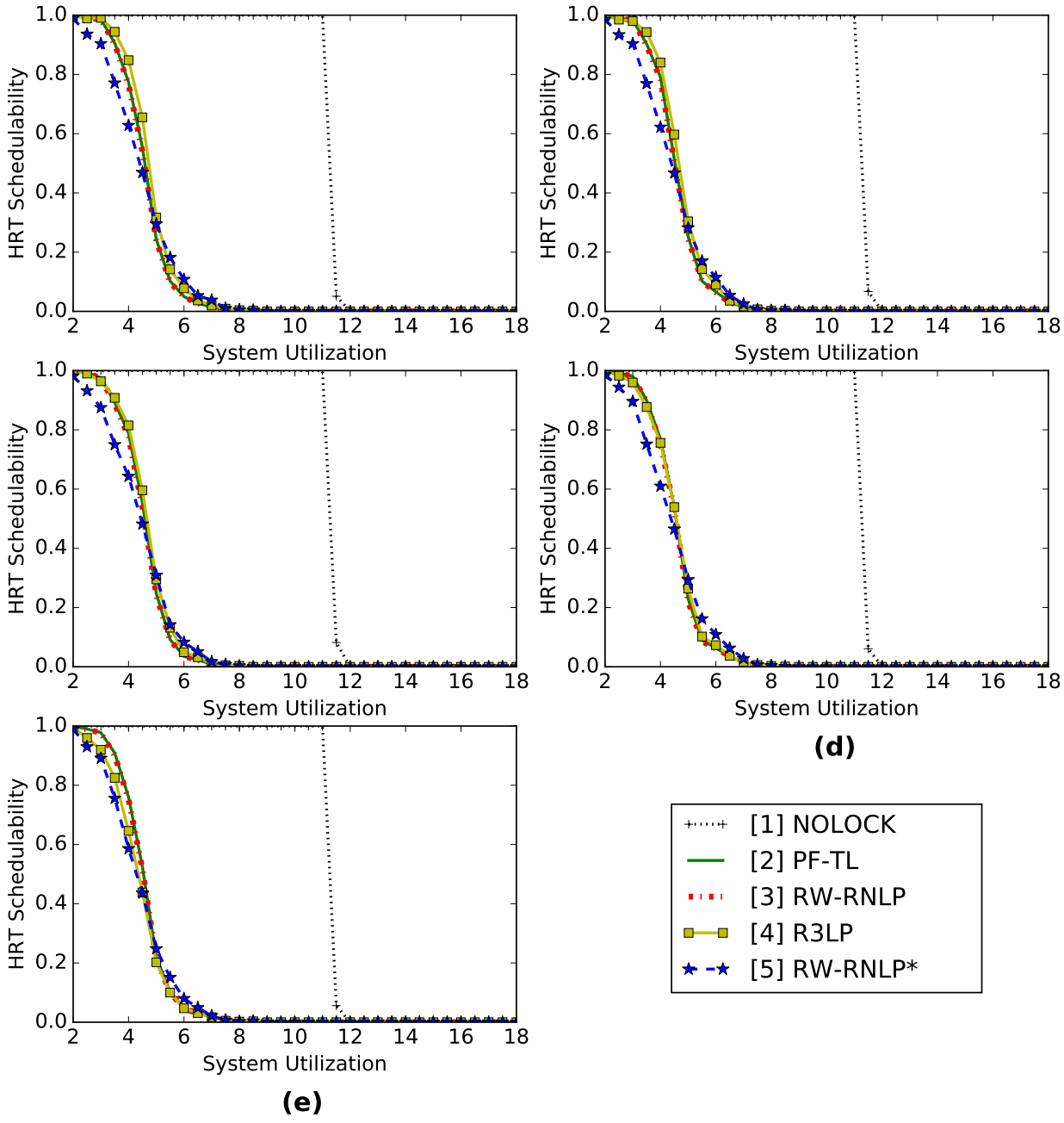



Figure 602: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, long critical-section lengths, and read probability 0.5 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

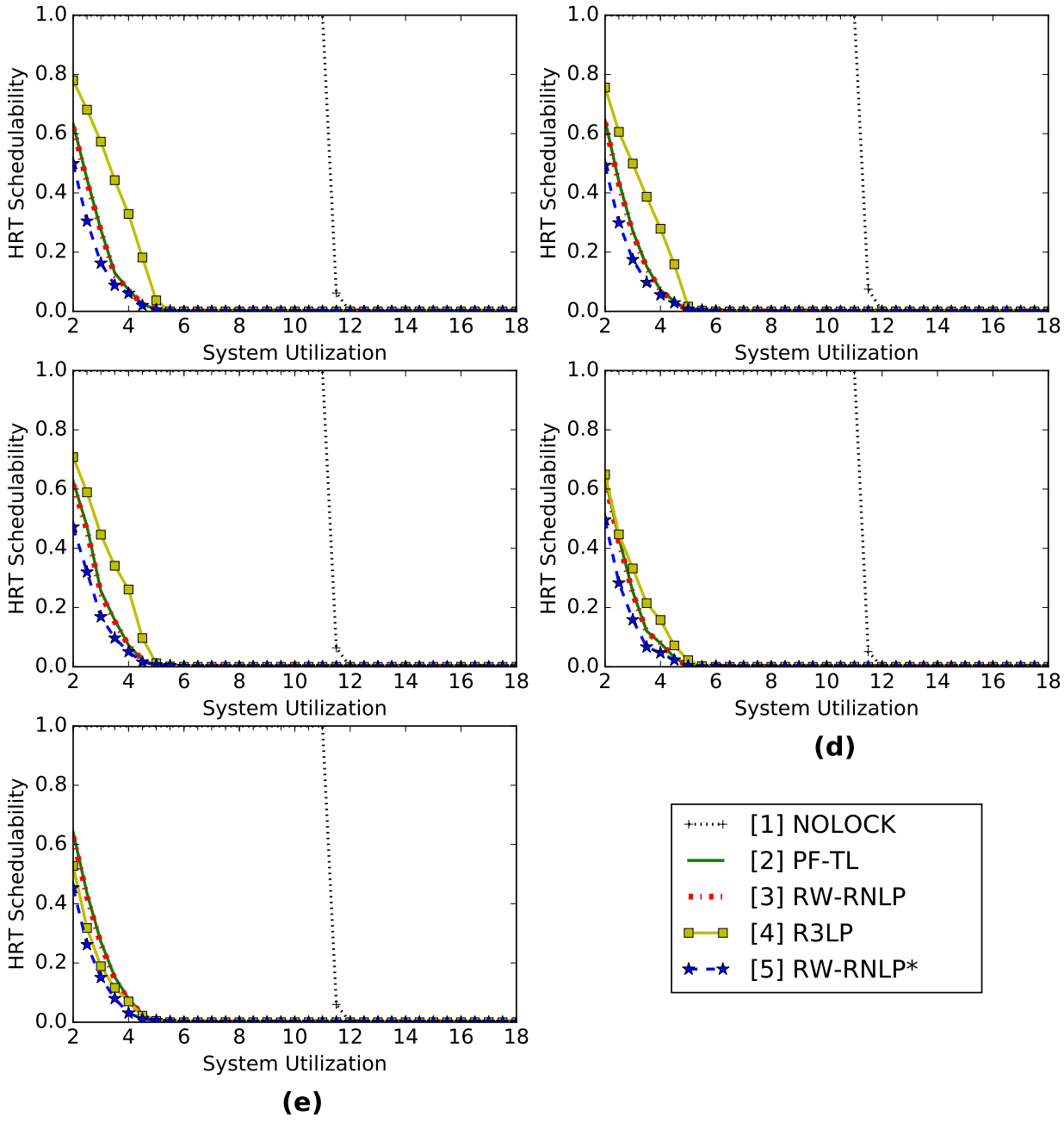



Figure 603: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, long critical-section lengths, and read probability 0.5 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

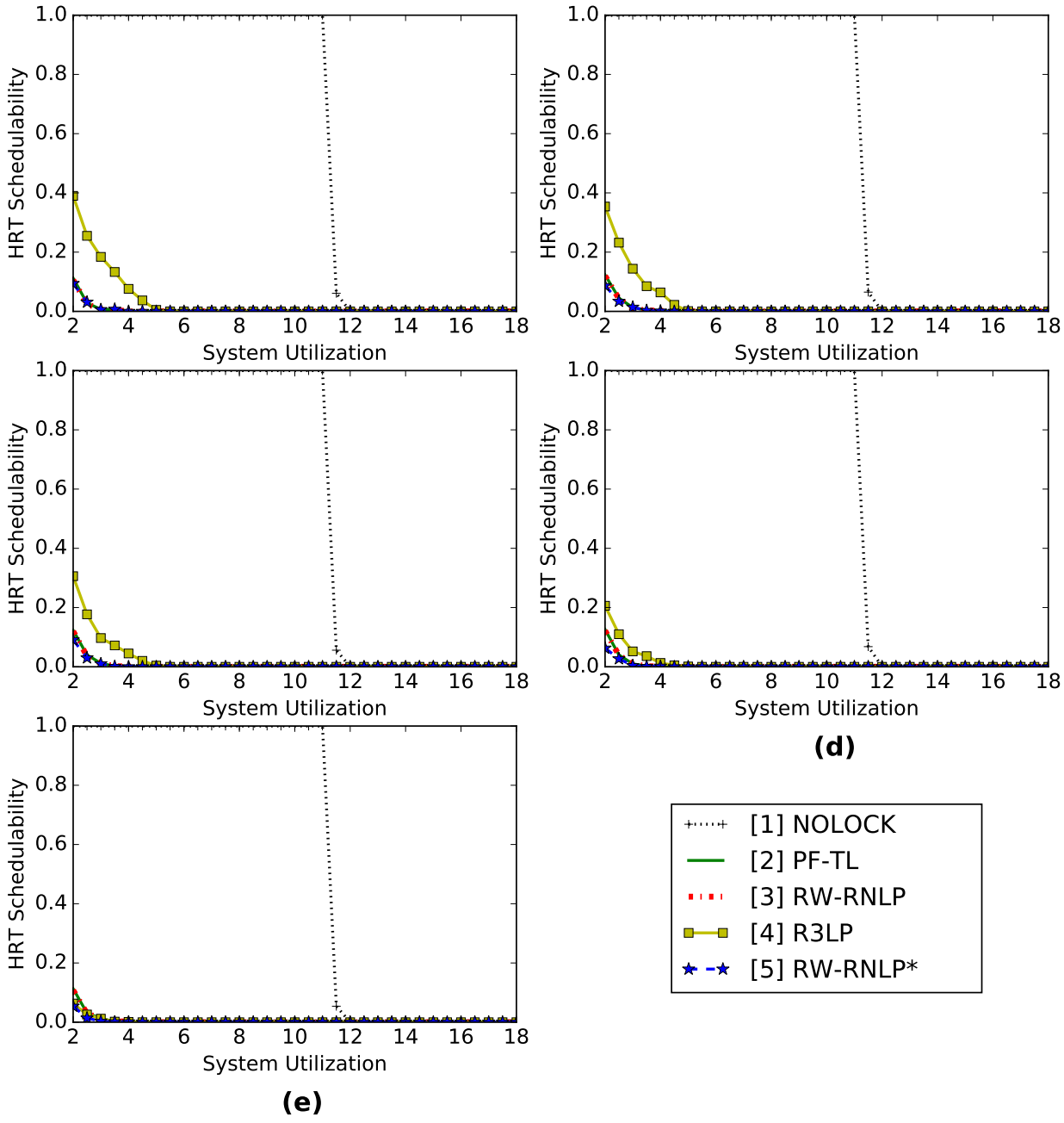



Figure 604: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, long critical-section lengths, and read probability 0.5 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

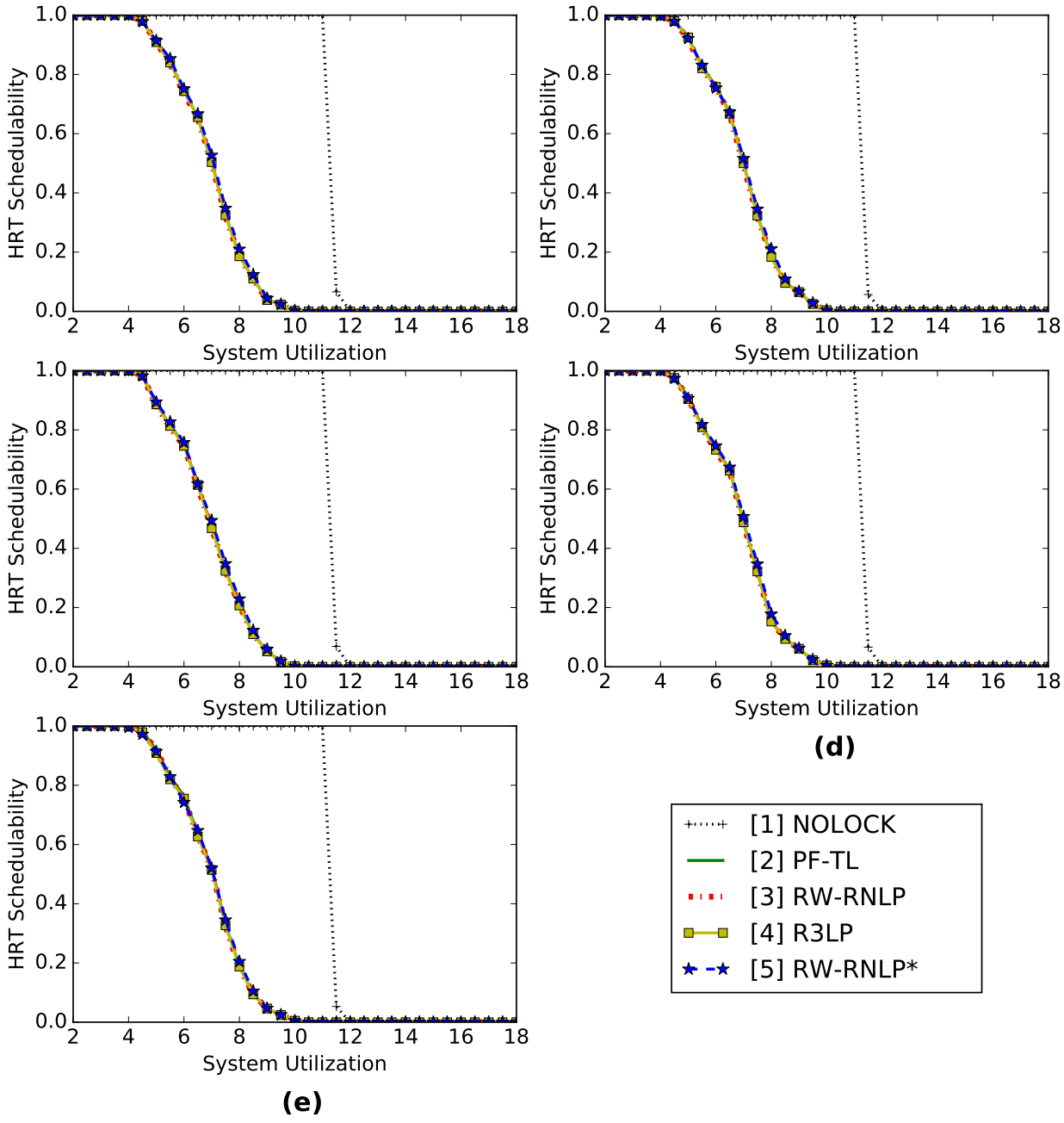



Figure 605: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, long critical-section lengths, and read probability 0.8 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

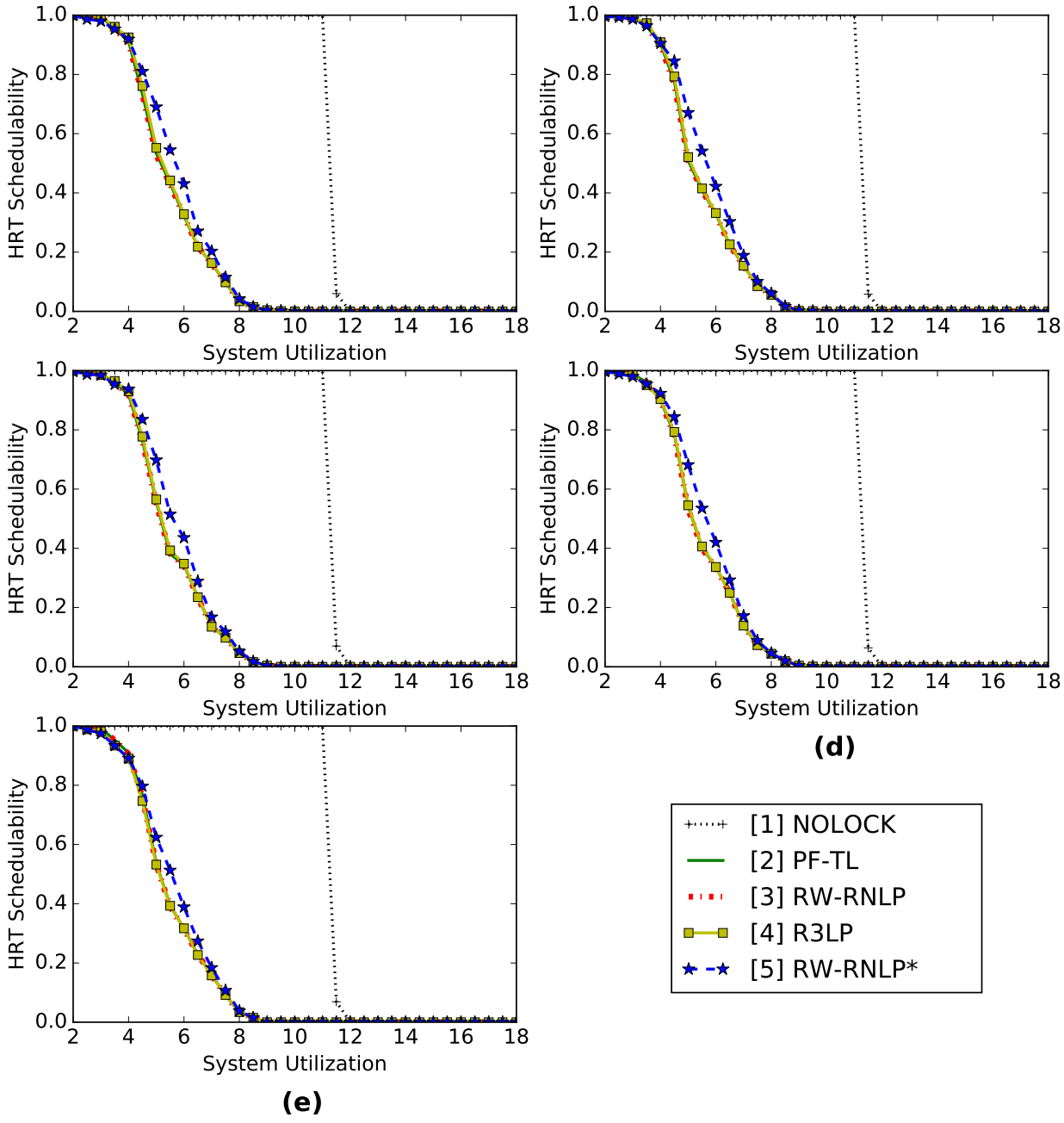



Figure 606: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, long critical-section lengths, and read probability 0.8 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

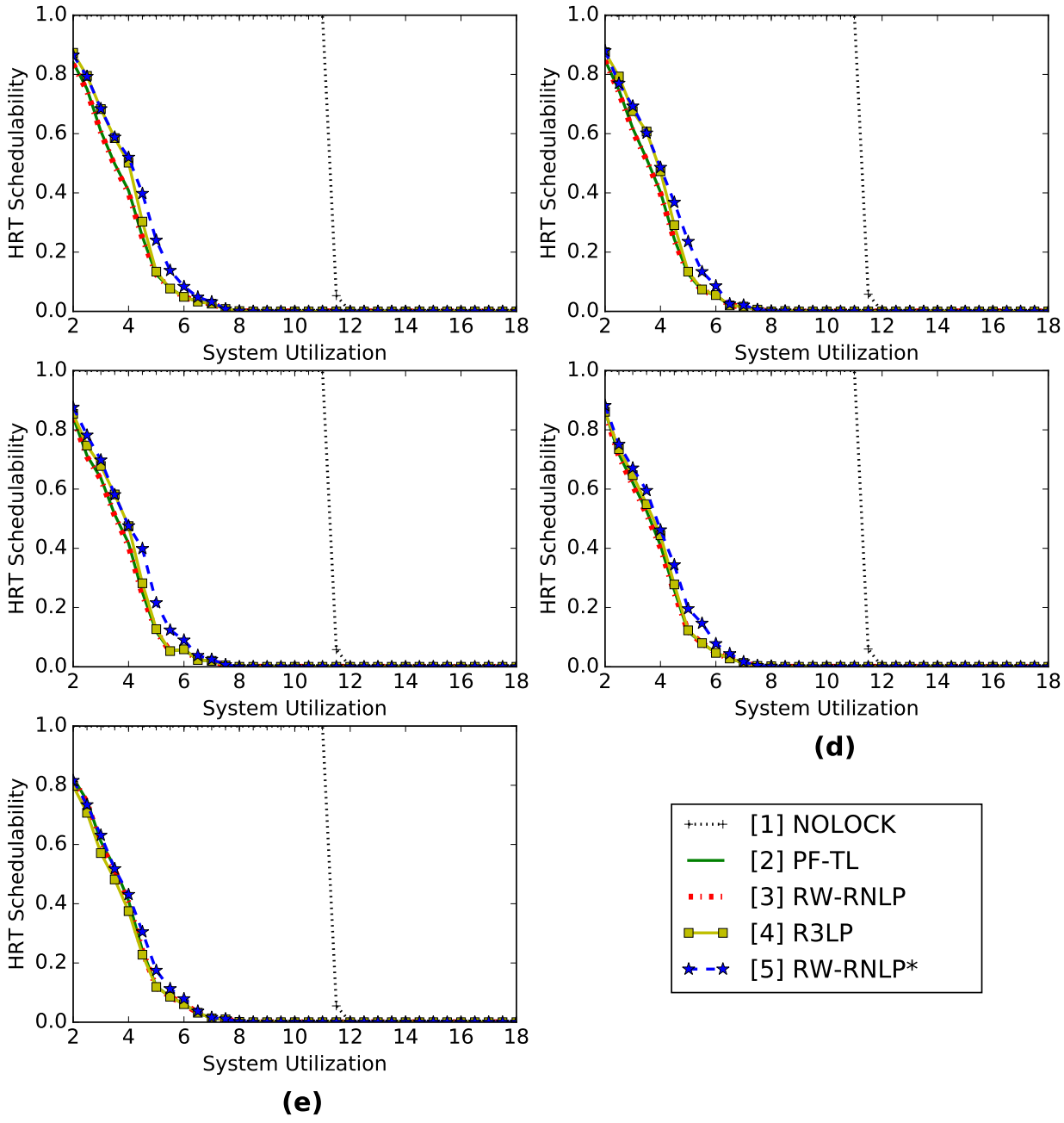



Figure 607: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, long critical-section lengths, and read probability 0.8 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

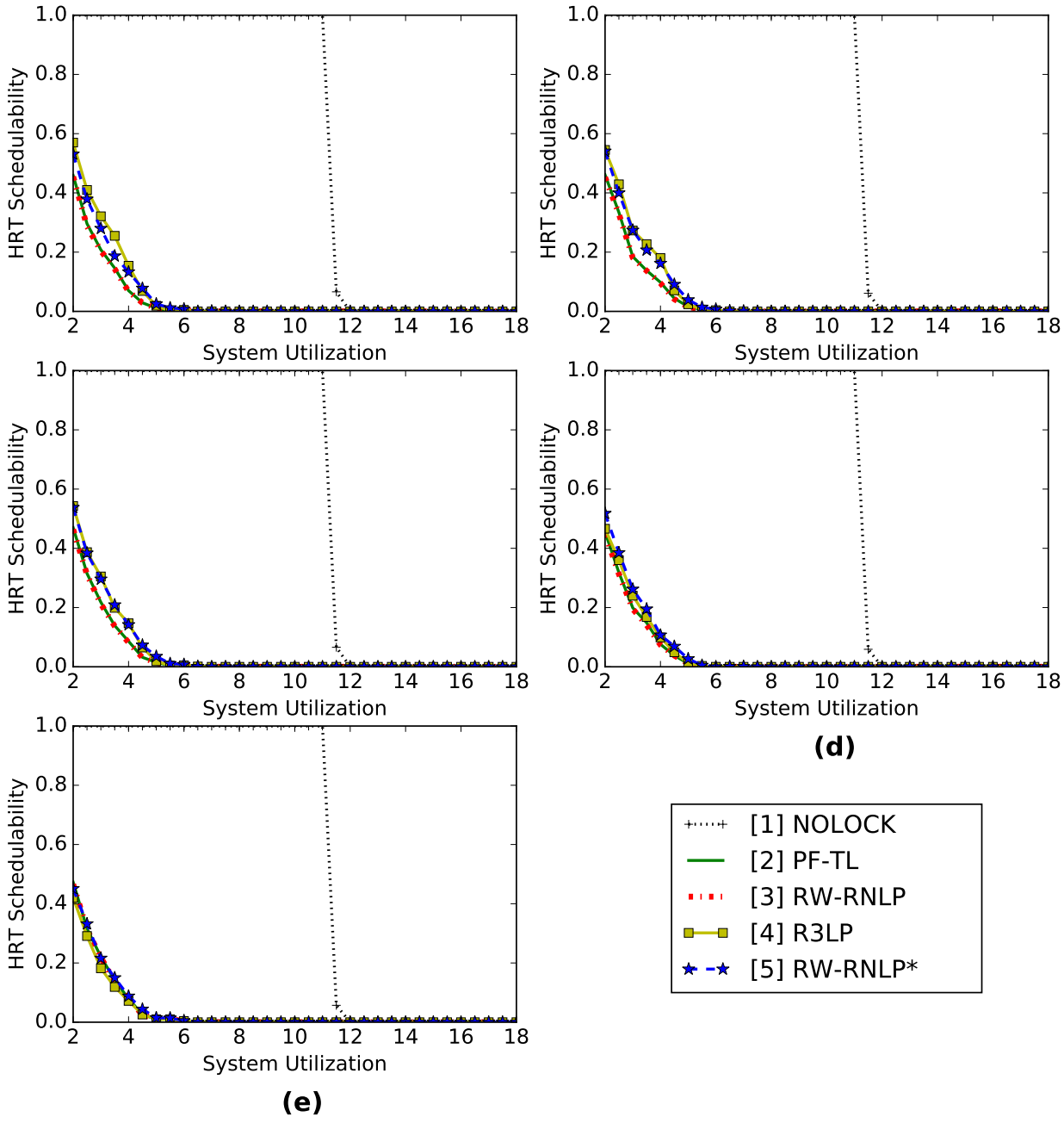



Figure 608: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, short periods, long critical-section lengths, and read probability 0.8 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

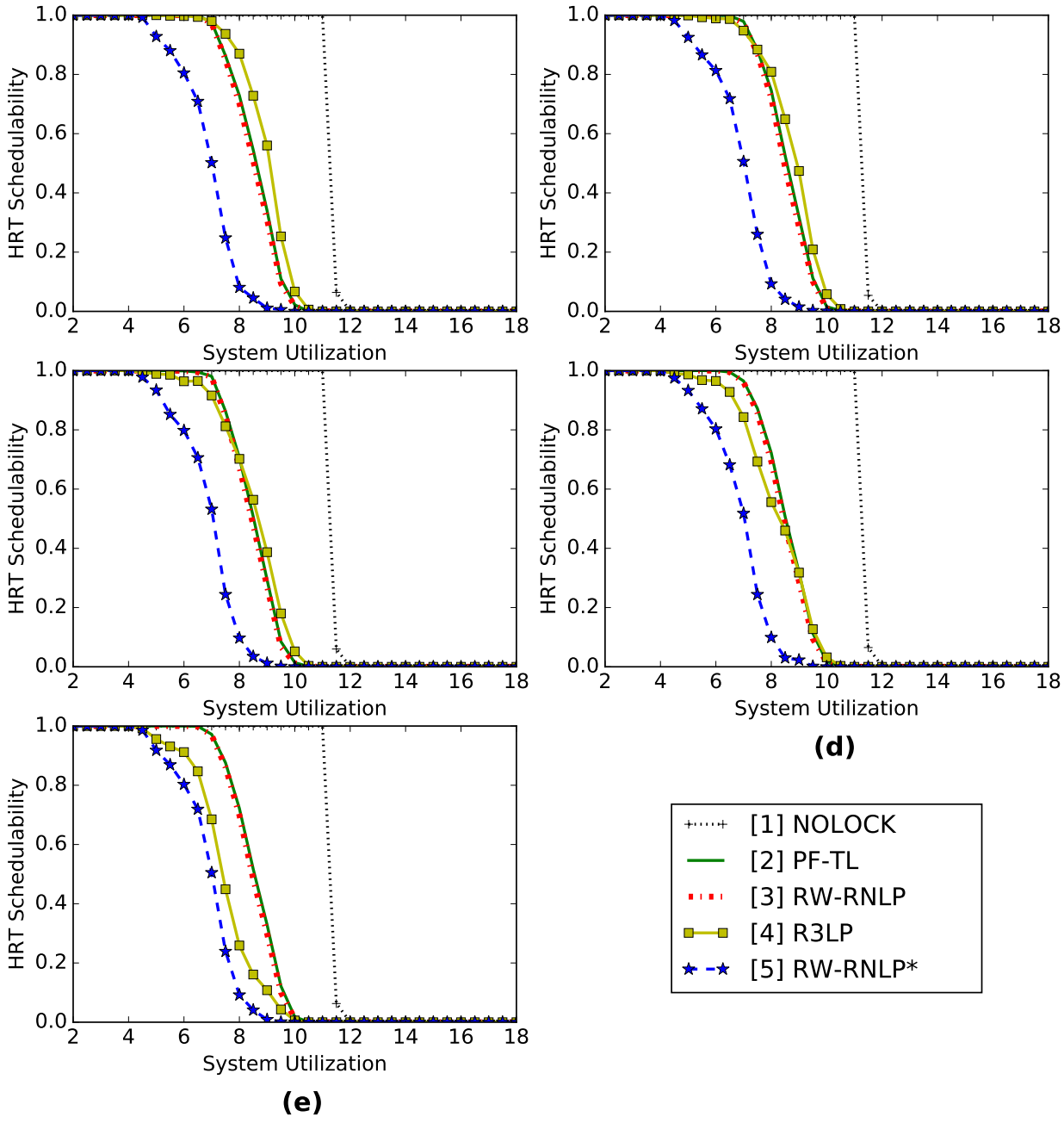



Figure 609: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, long critical-section lengths, and read probability 0 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

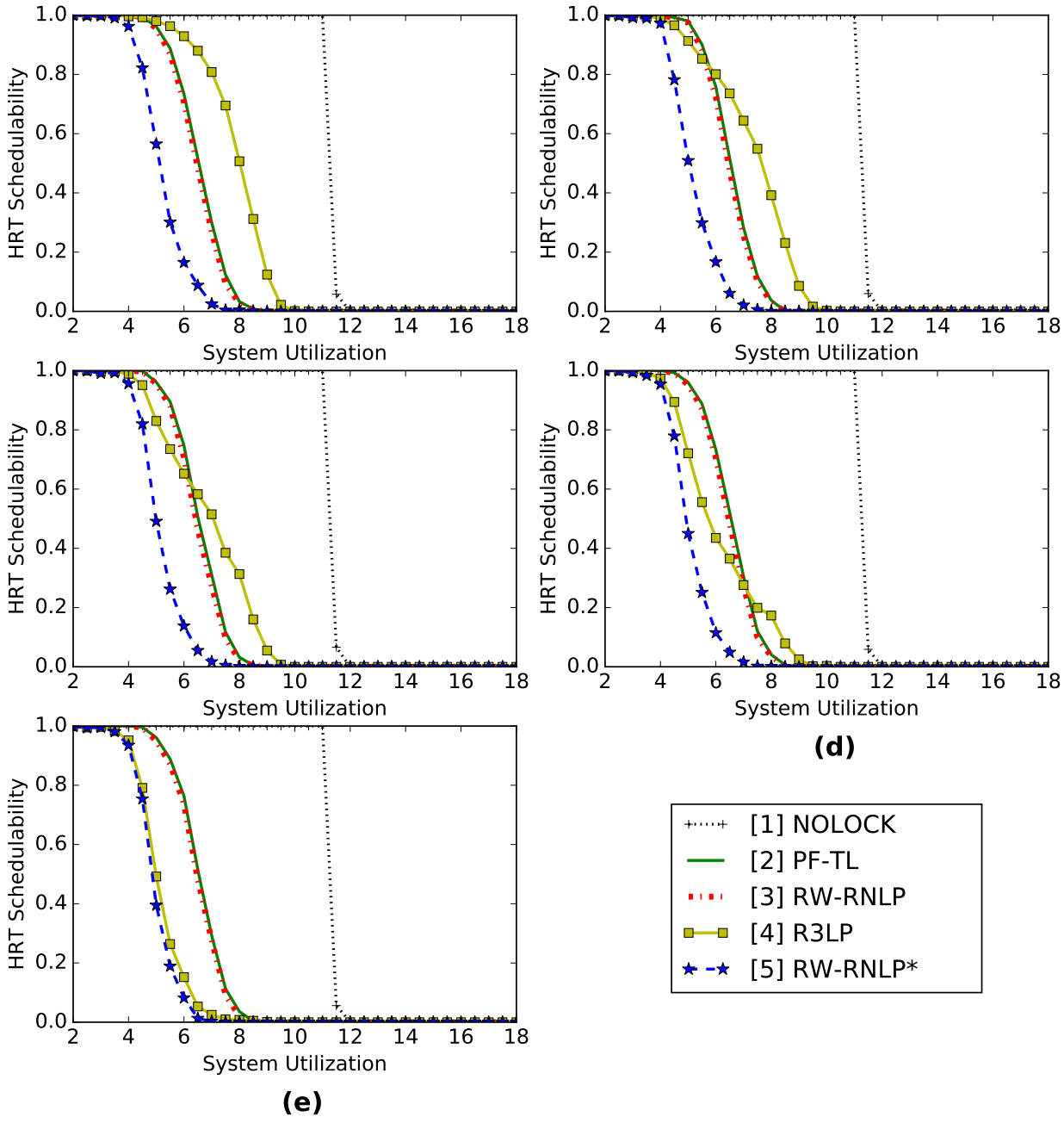



Figure 610: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, long critical-section lengths, and read probability 0 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

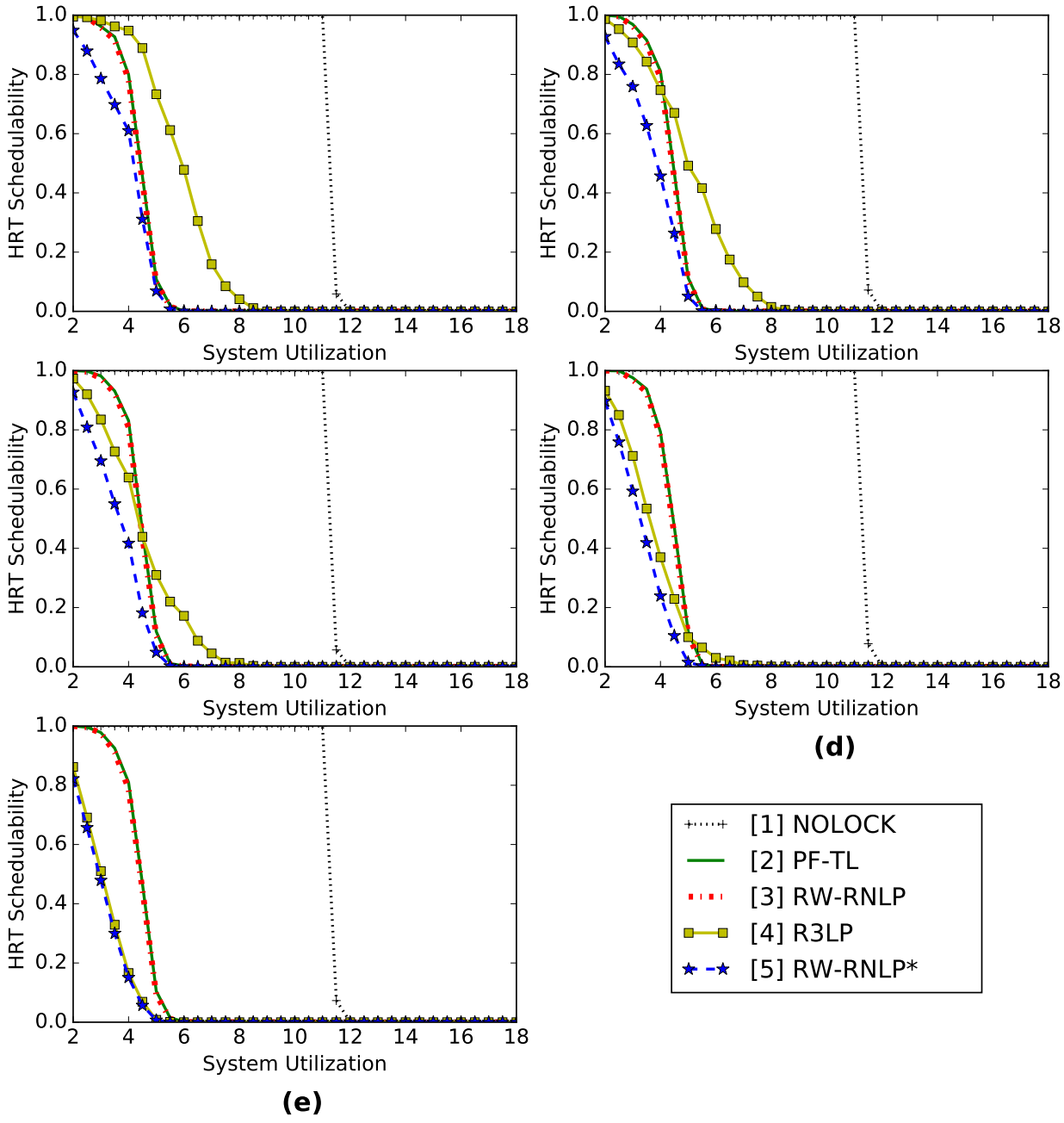



Figure 611: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, long critical-section lengths, and read probability 0 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

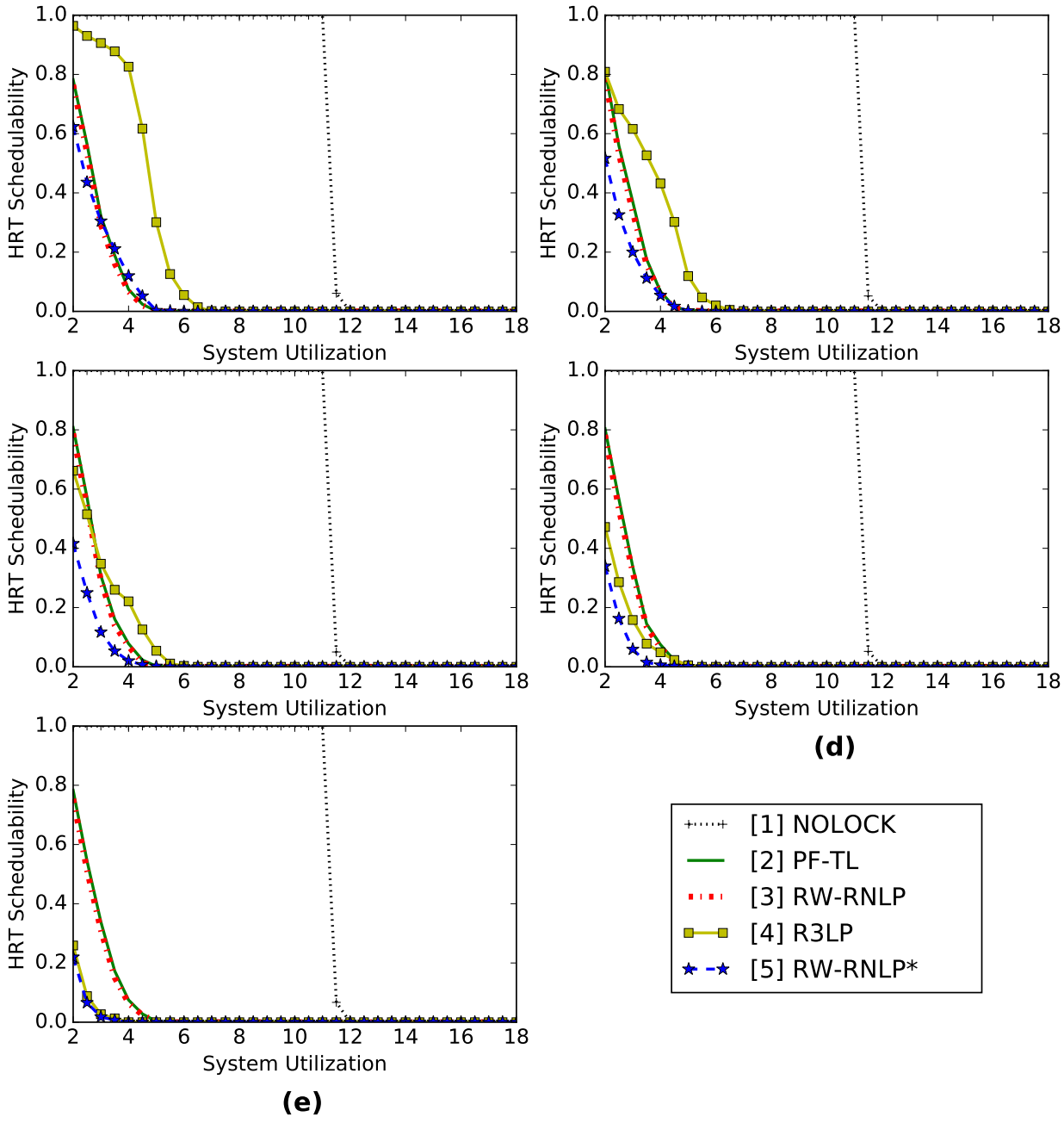



Figure 612: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, long critical-section lengths, and read probability 0 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

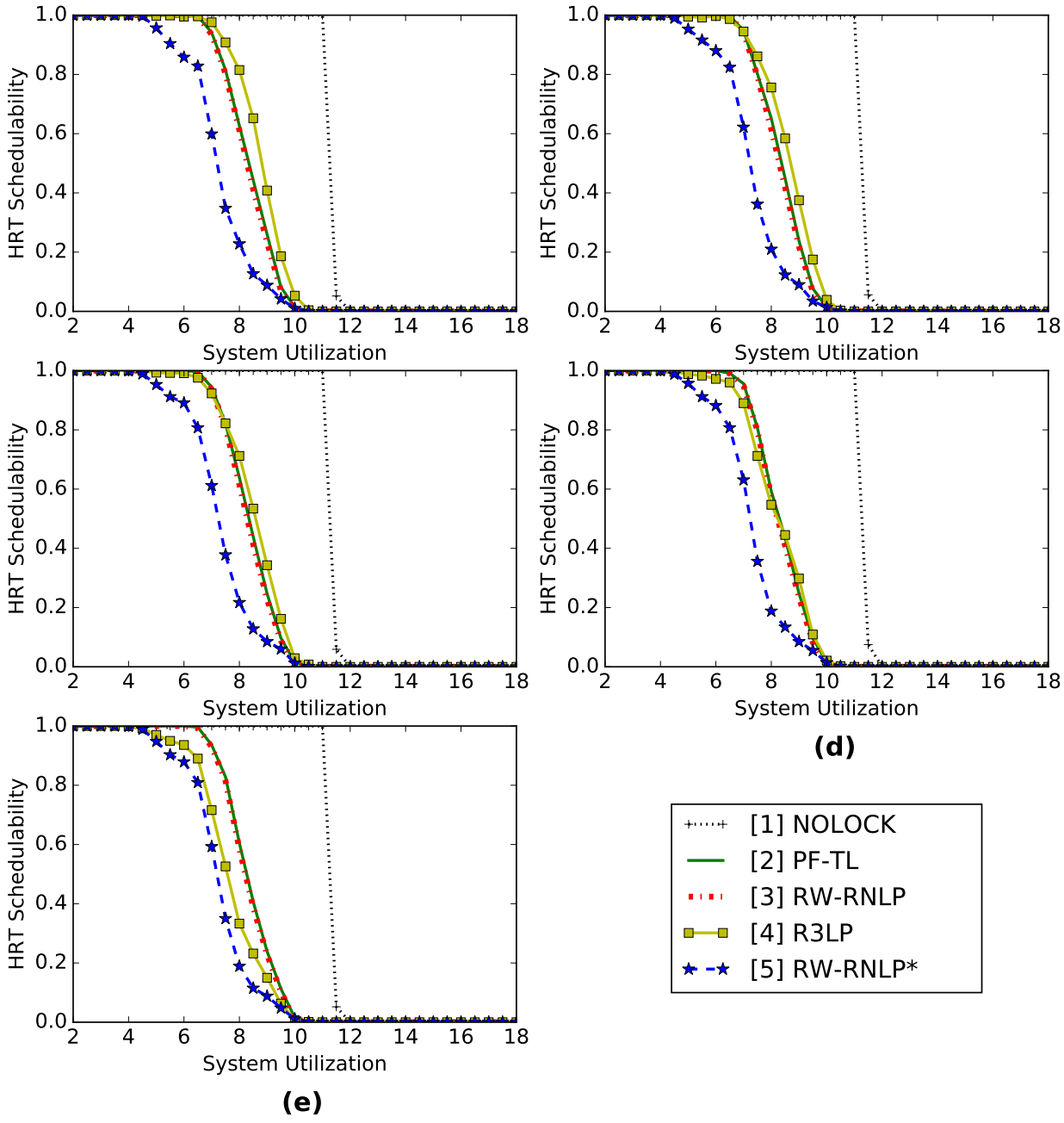



Figure 613: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, long critical-section lengths, and read probability 0.2 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

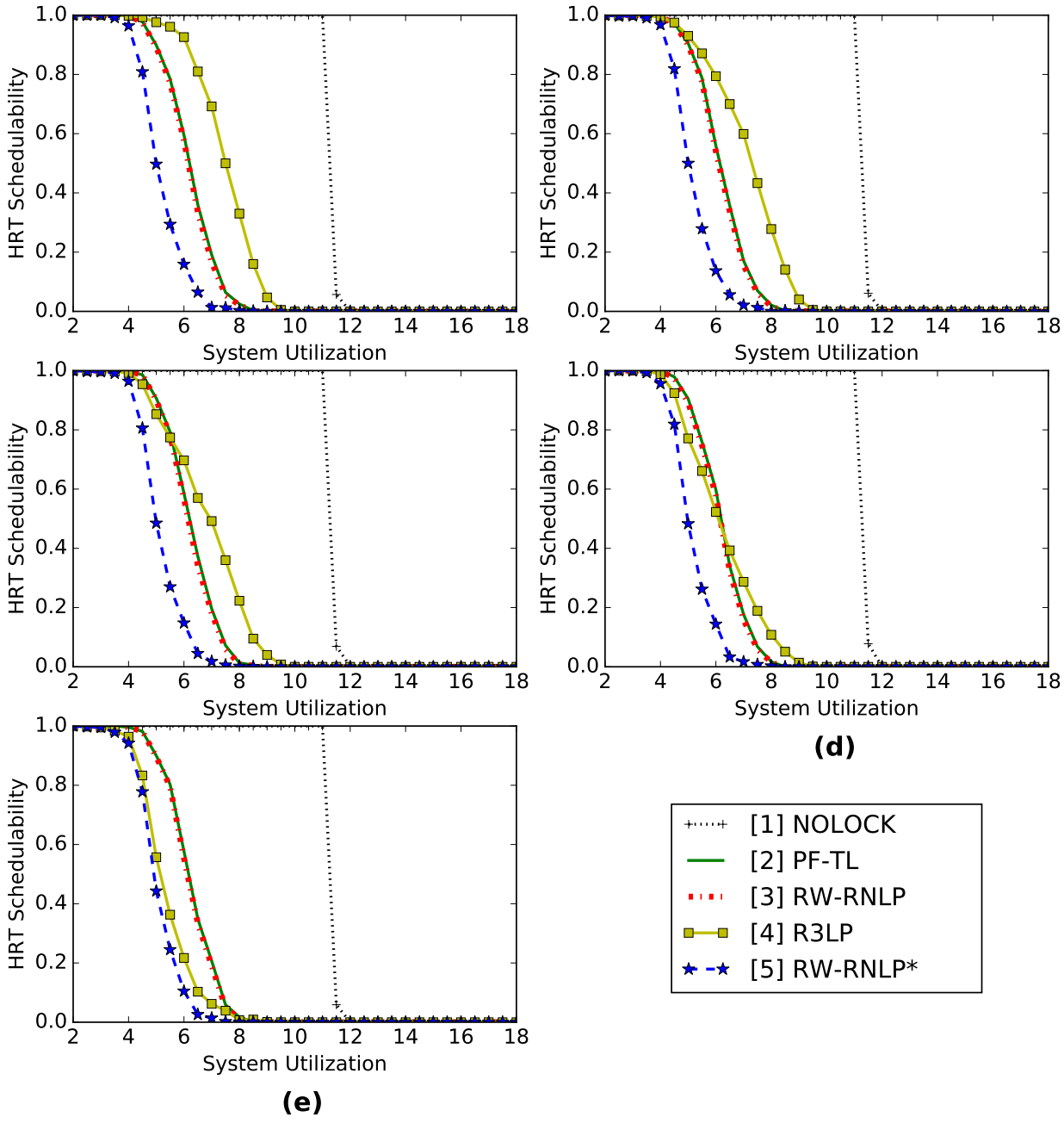



Figure 614: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, long critical-section lengths, and read probability 0.2 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

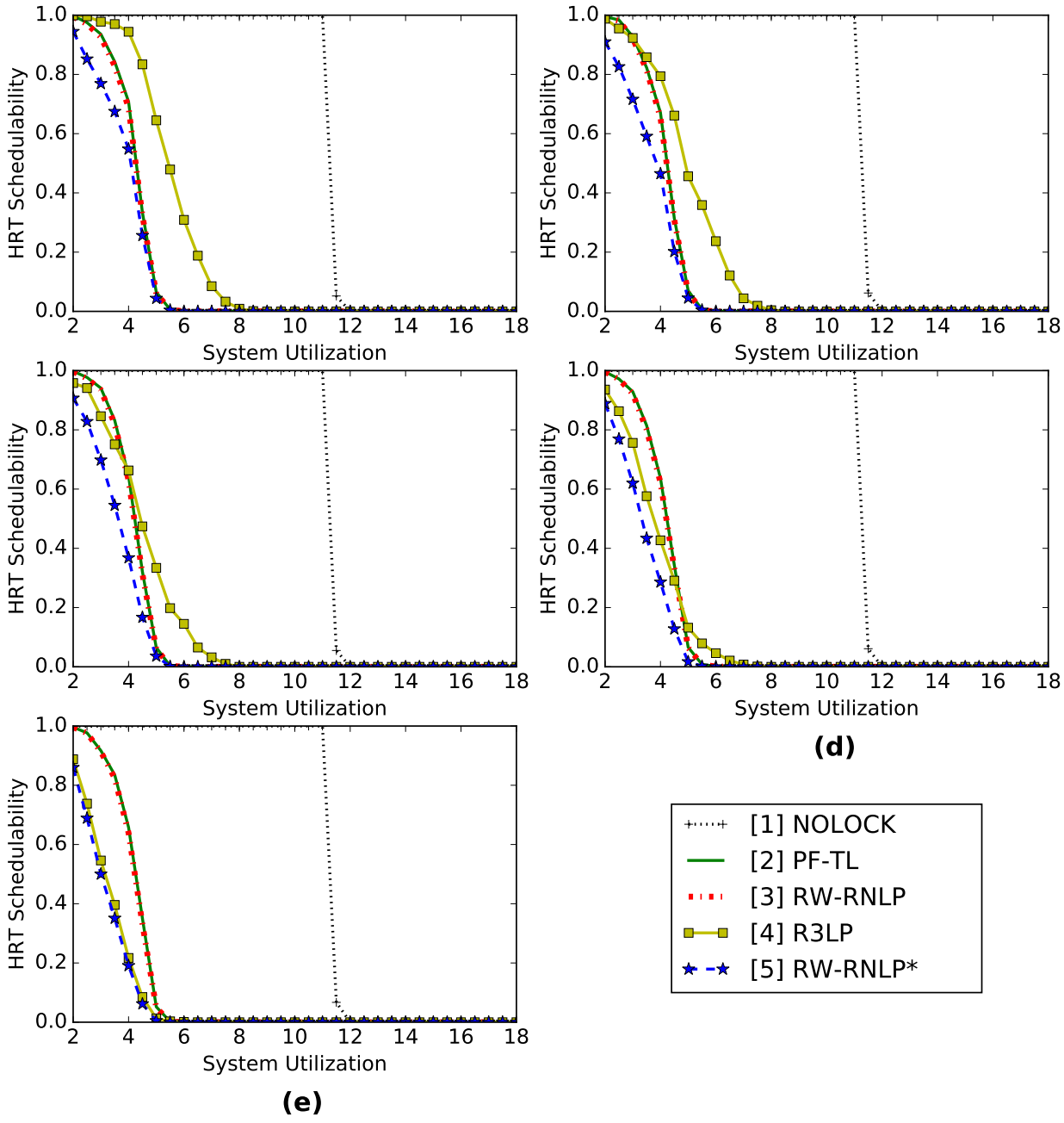



Figure 615: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, long critical-section lengths, and read probability 0.2 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

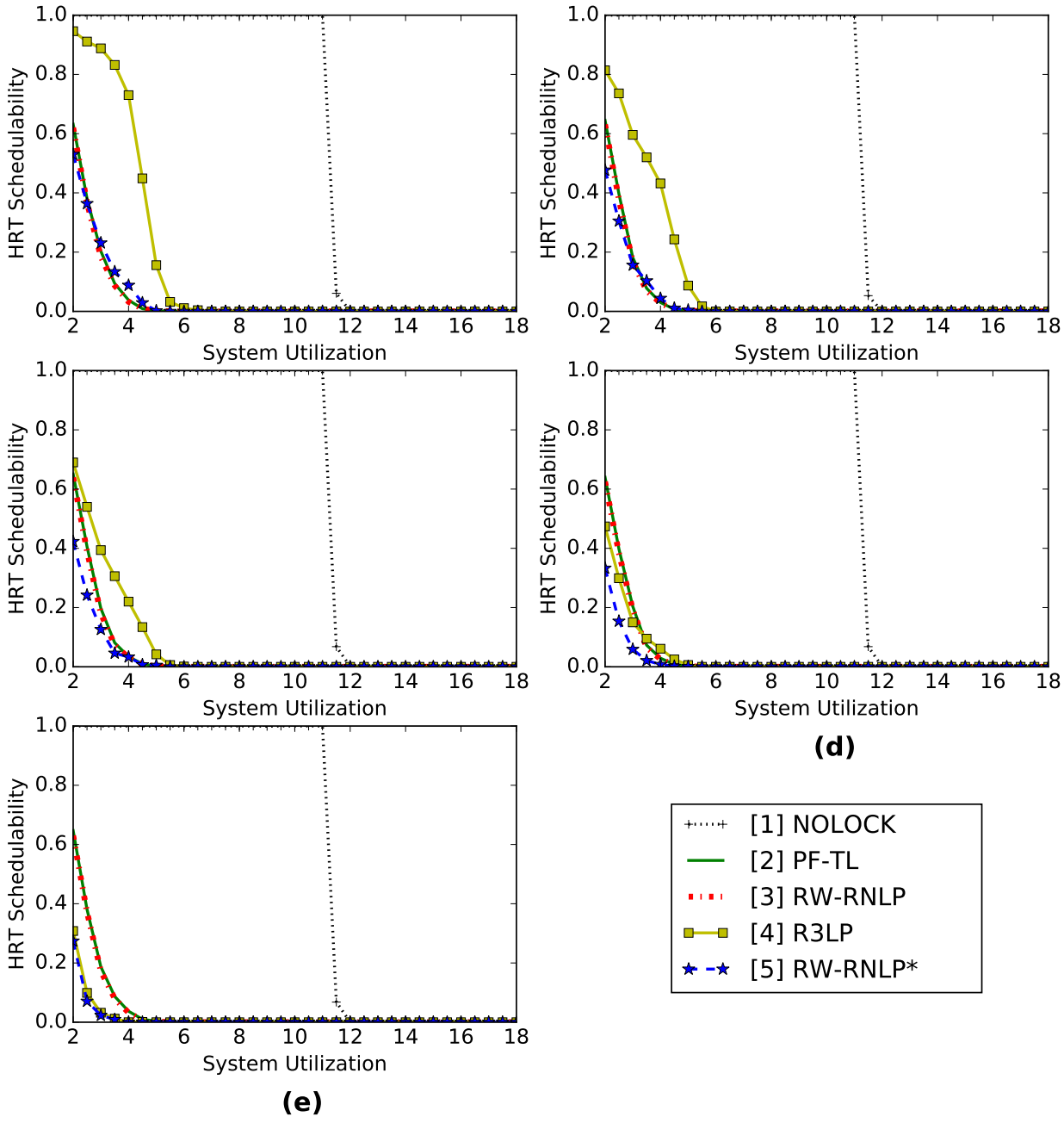



Figure 616: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, long critical-section lengths, and read probability 0.2 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

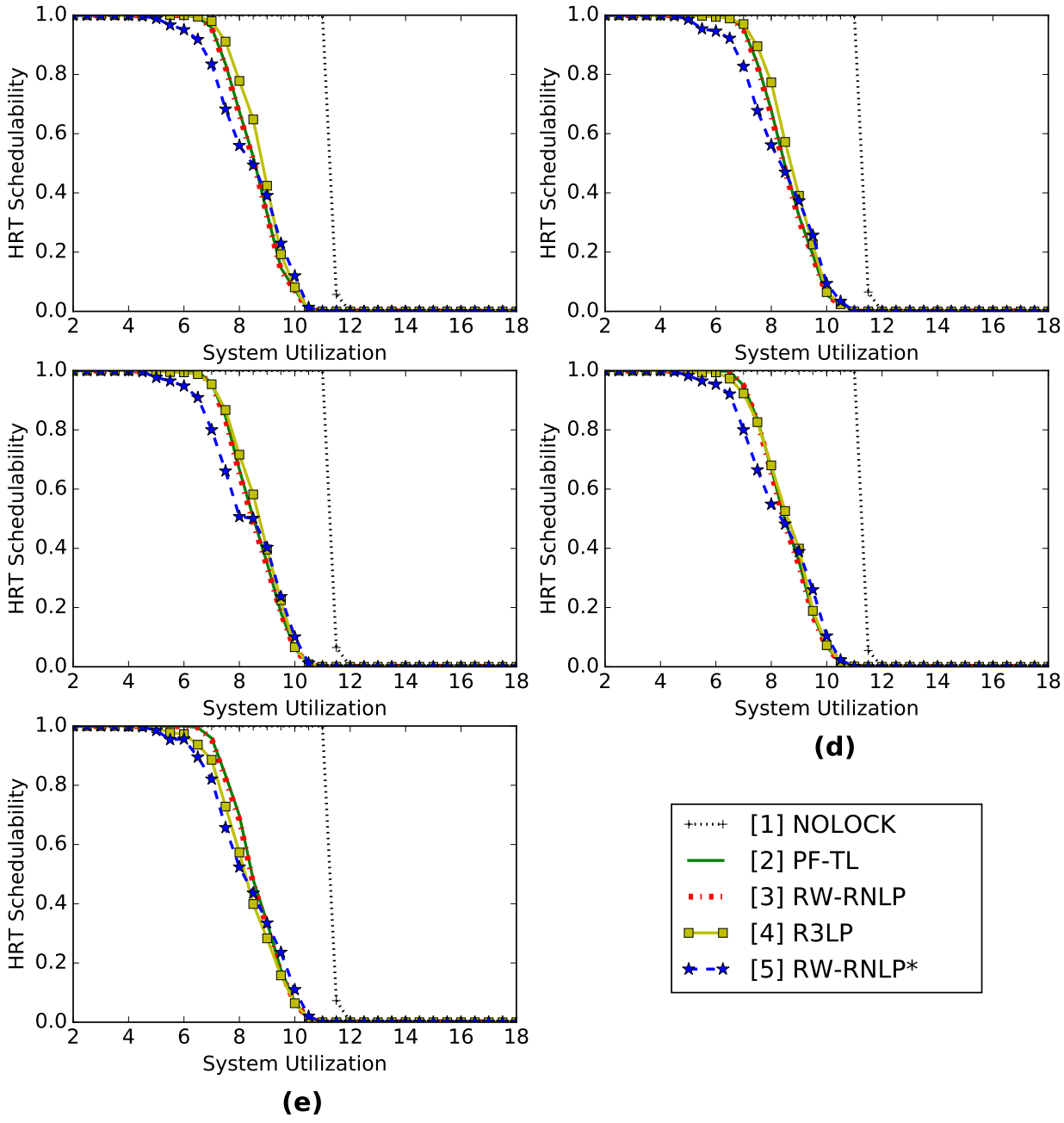



Figure 617: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, long critical-section lengths, and read probability 0.5 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

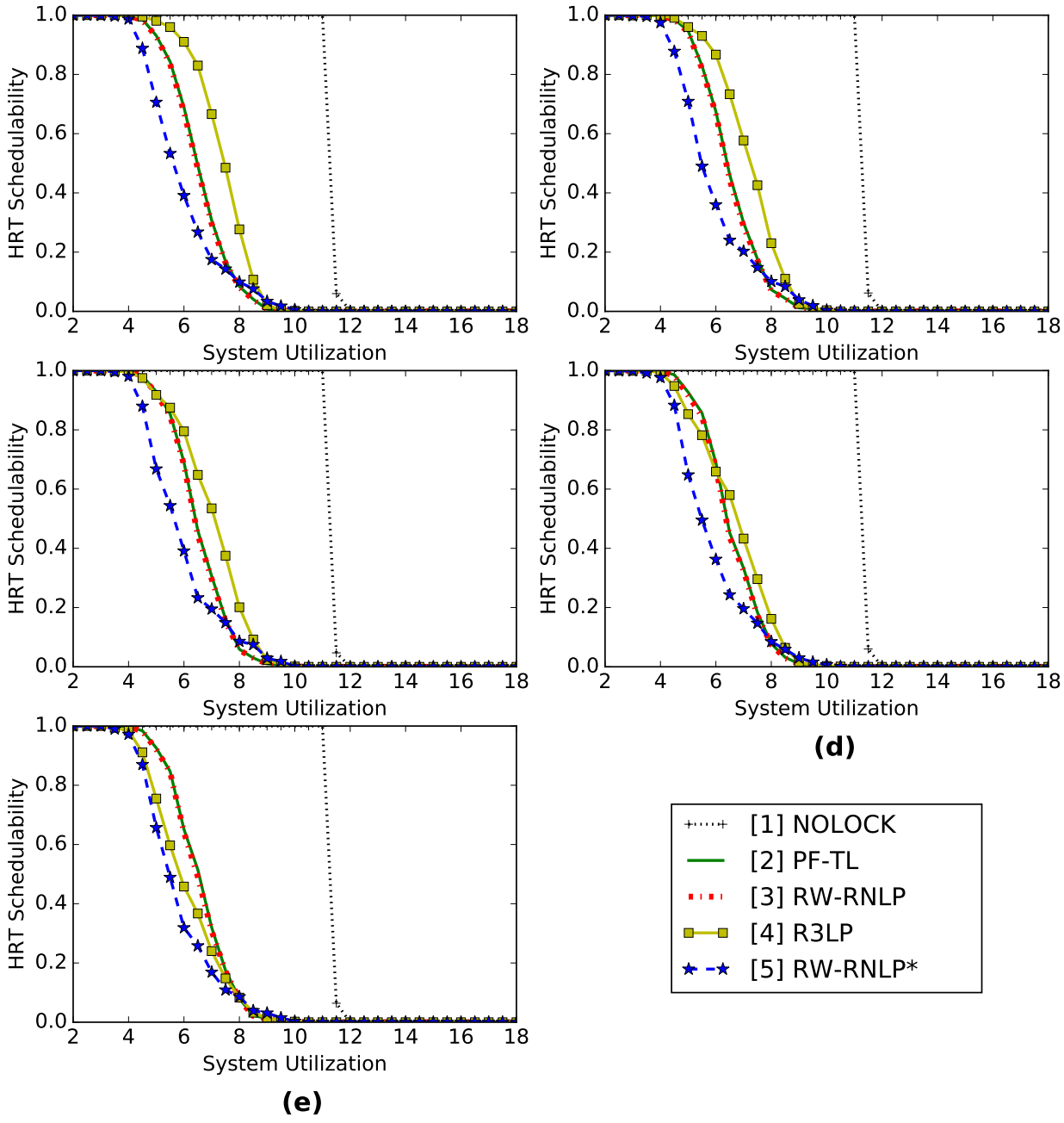



Figure 618: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, long critical-section lengths, and read probability 0.5 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

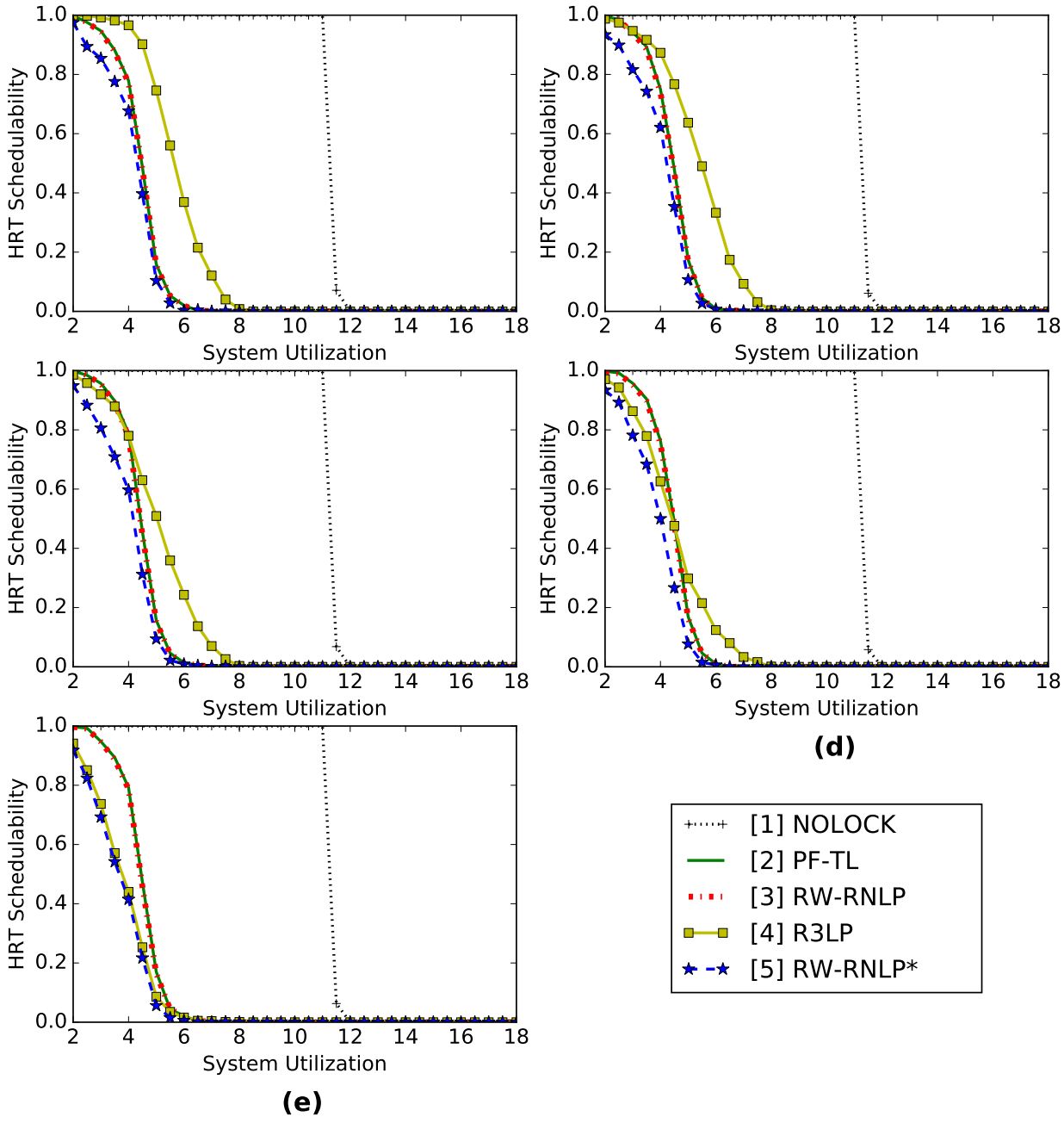



Figure 619: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, long critical-section lengths, and read probability 0.5 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

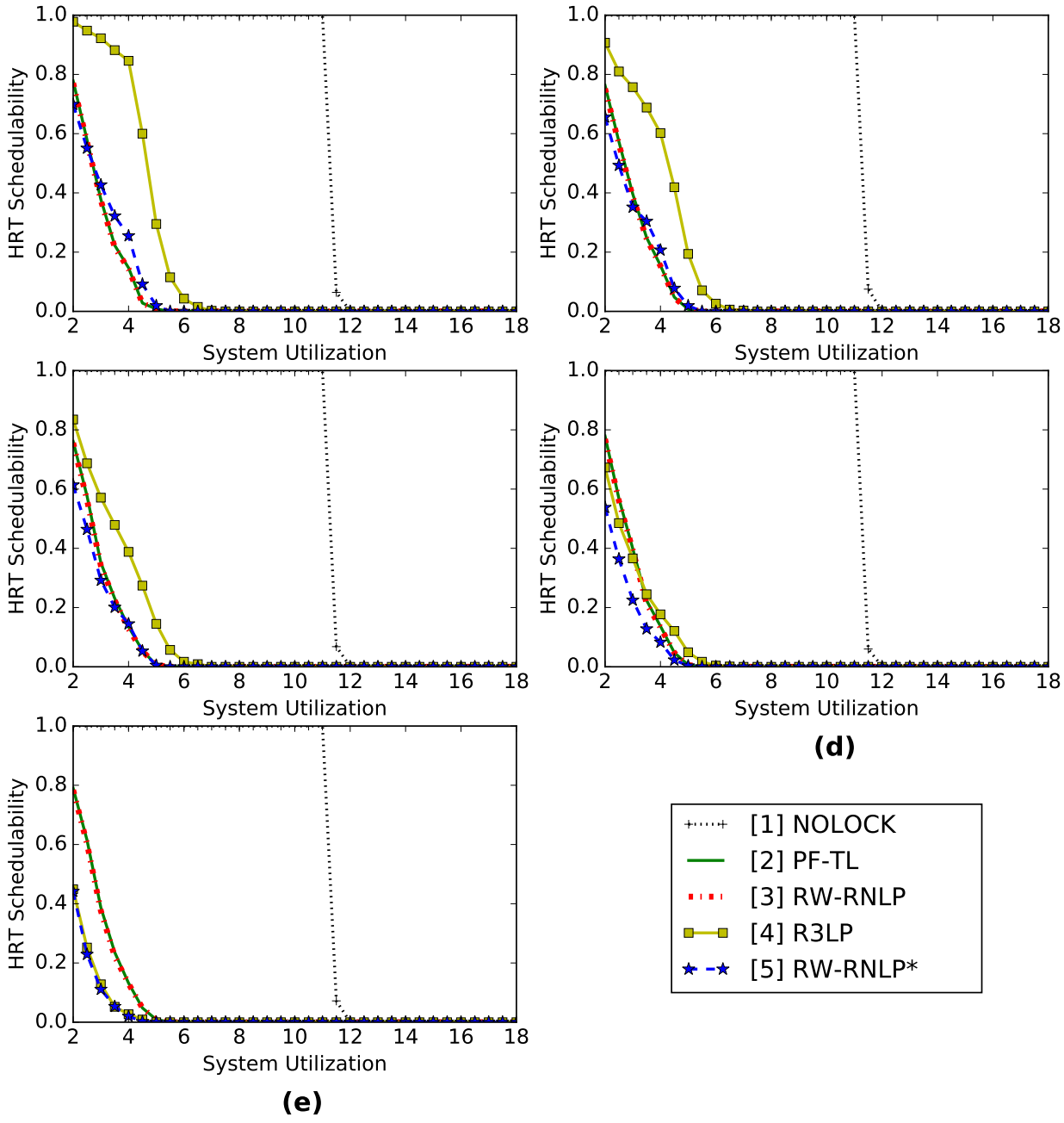



Figure 620: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, long critical-section lengths, and read probability 0.5 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

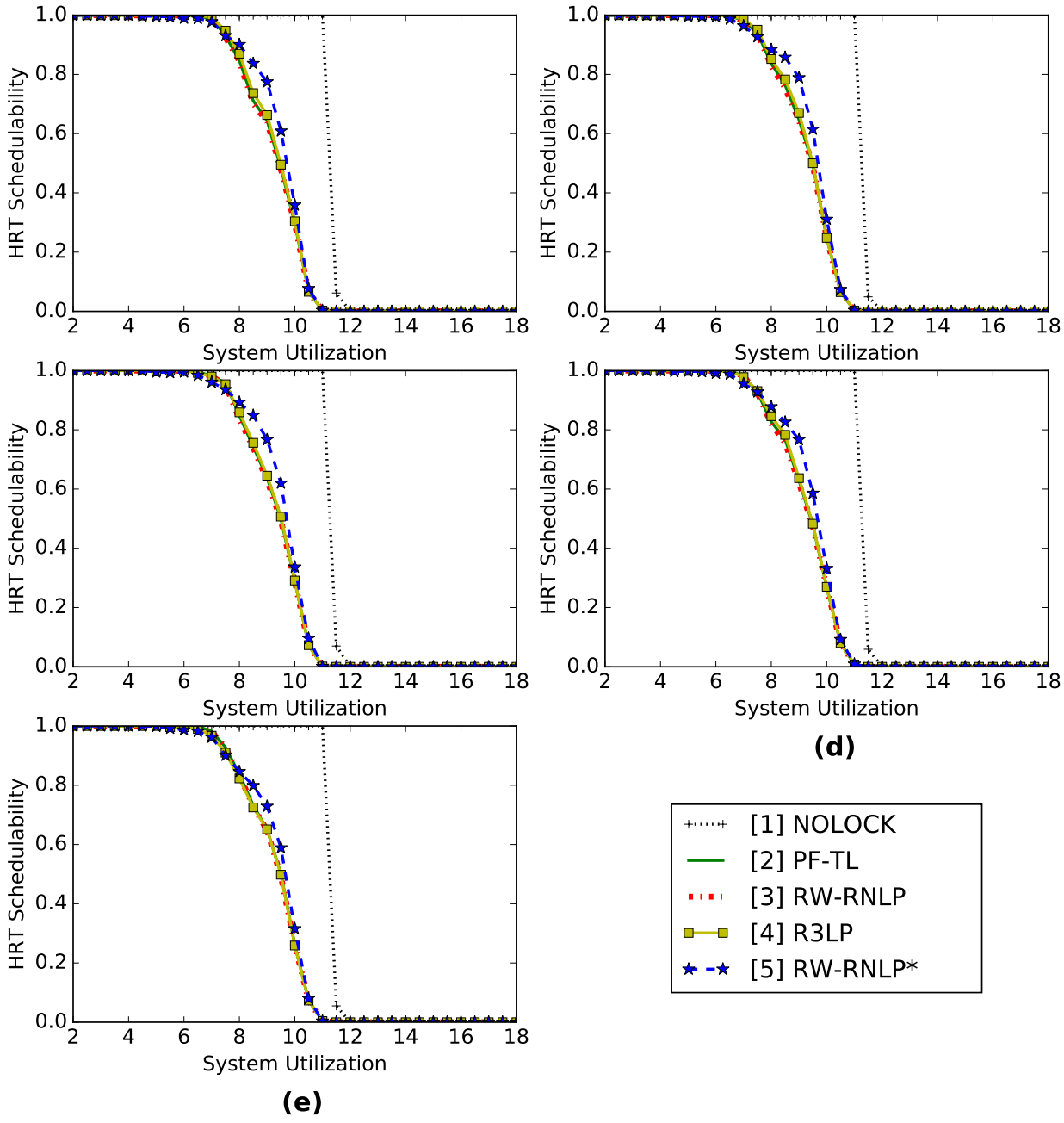



Figure 621: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, long critical-section lengths, and read probability 0.8 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

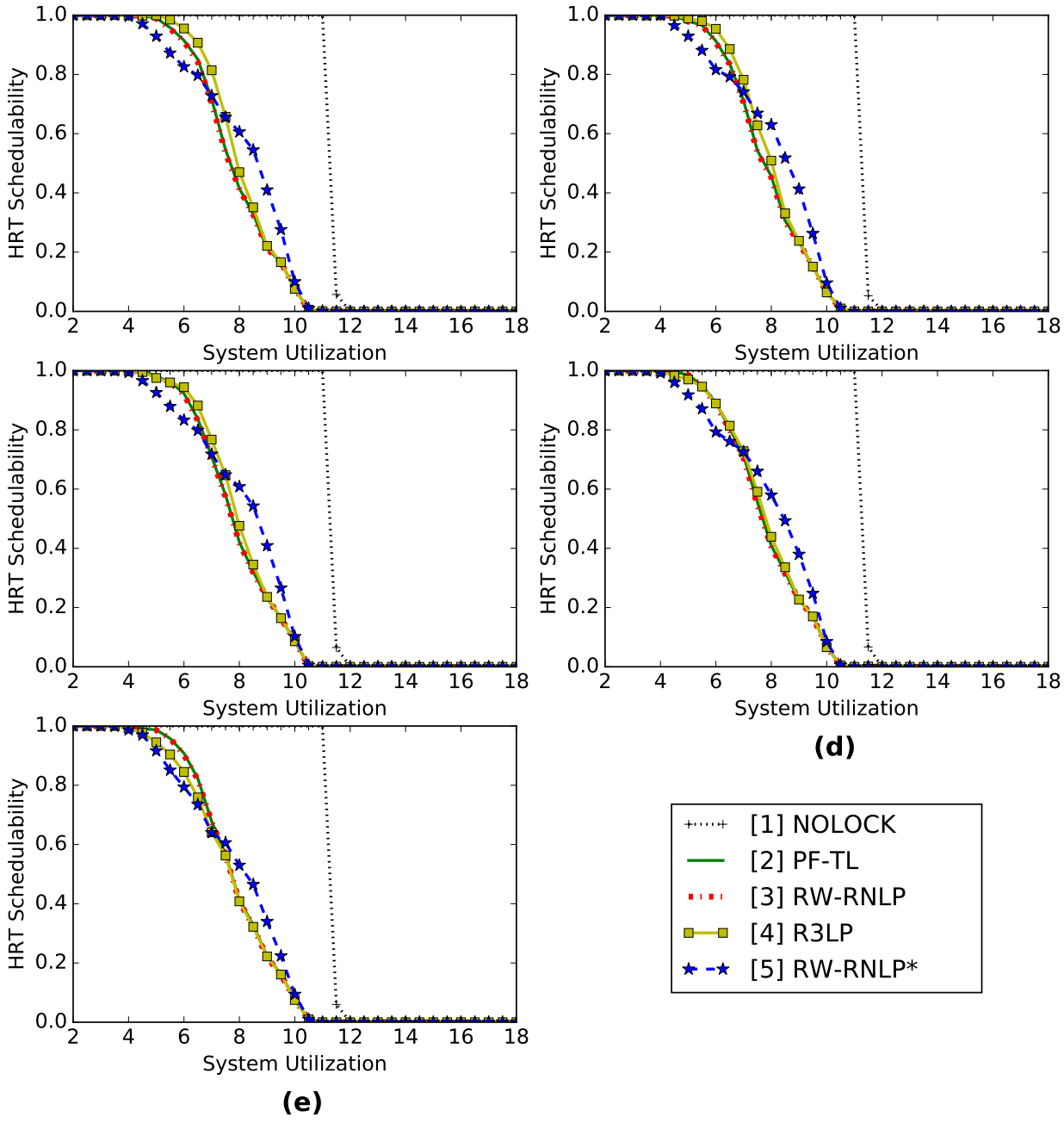



Figure 622: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, long critical-section lengths, and read probability 0.8 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

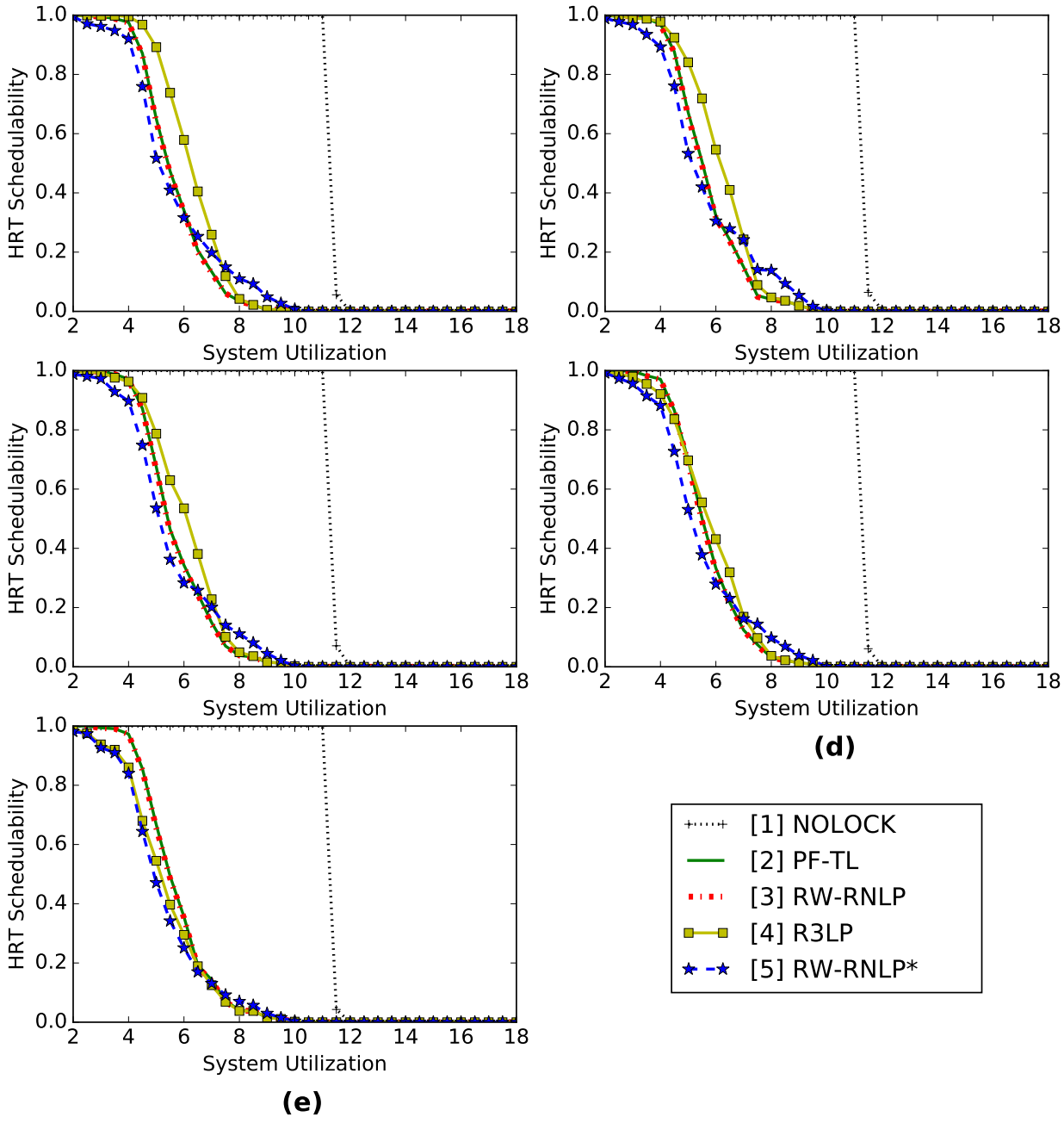



Figure 623: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, long critical-section lengths, and read probability 0.8 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

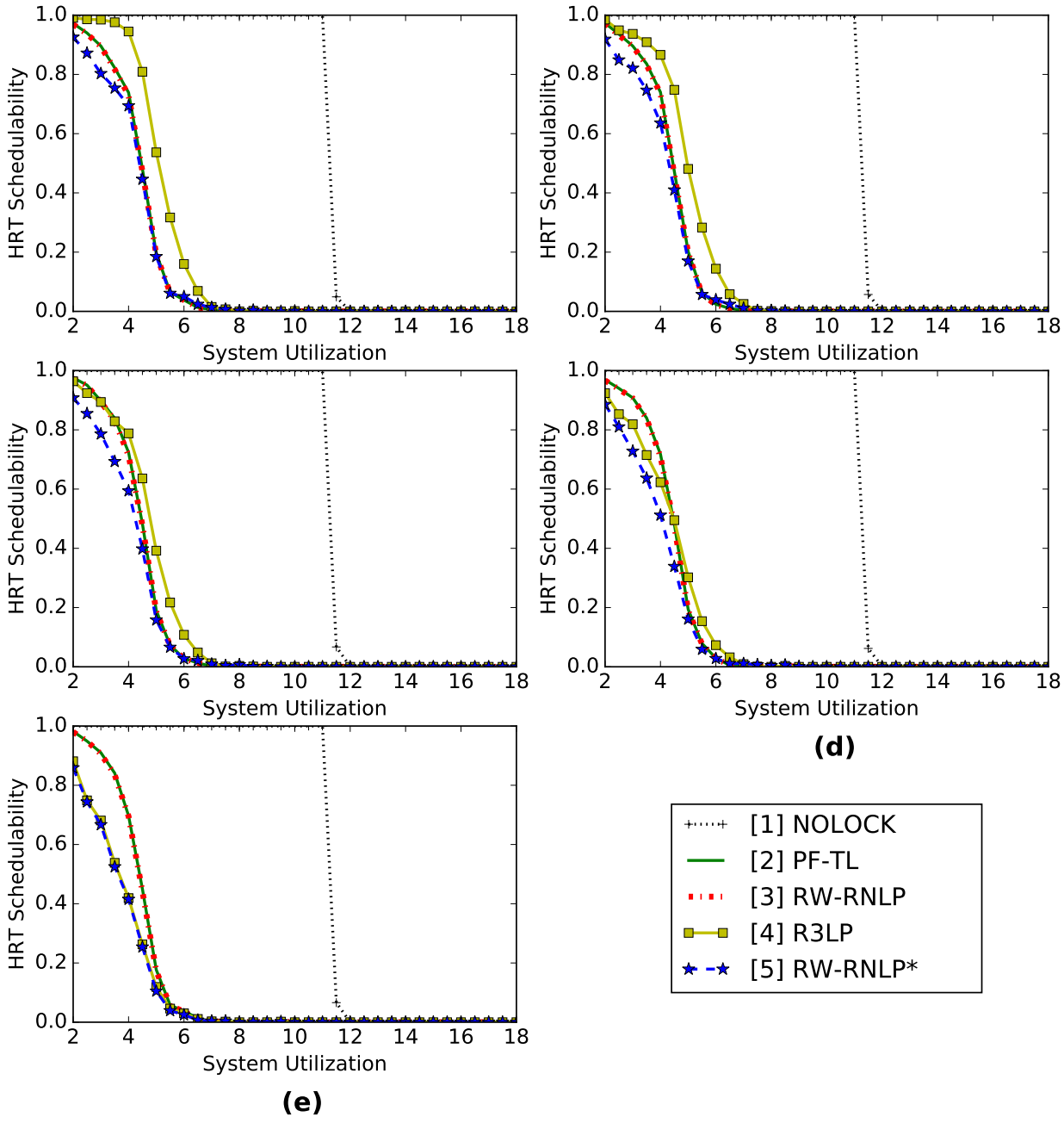



Figure 624: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, moderate periods, long critical-section lengths, and read probability 0.8 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

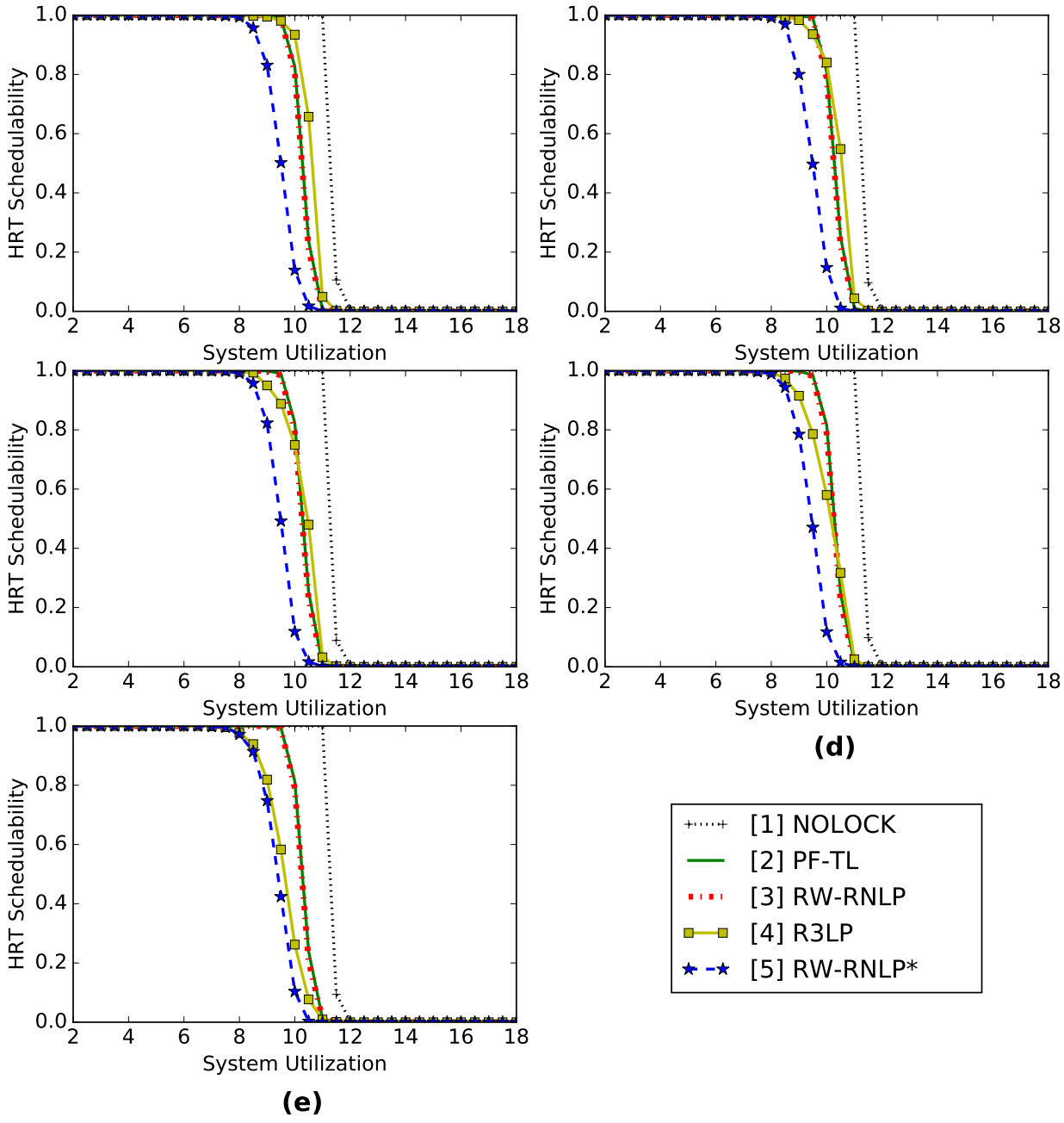



Figure 625: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, long critical-section lengths, and read probability 0 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

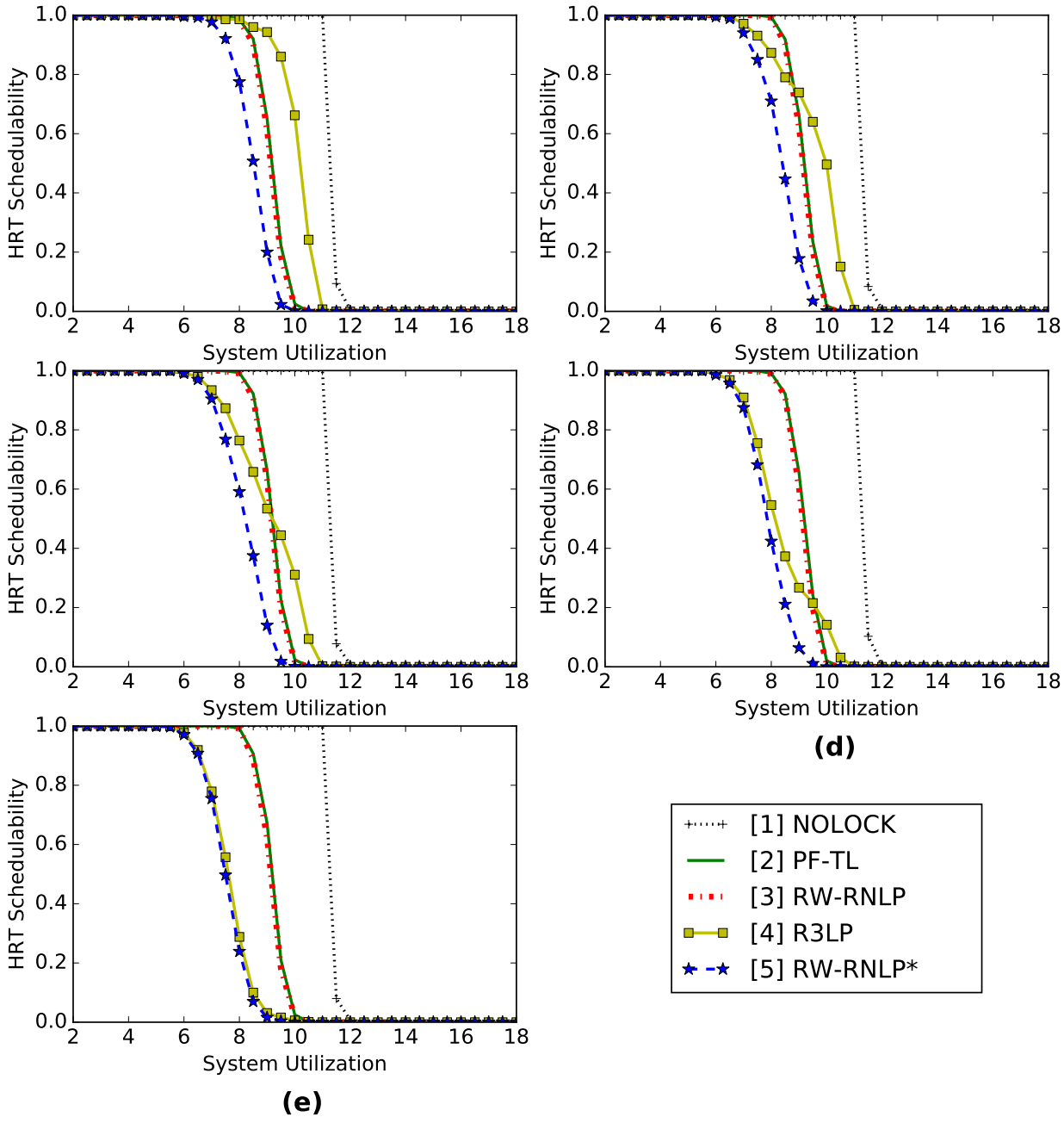



Figure 626: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, long critical-section lengths, and read probability 0 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

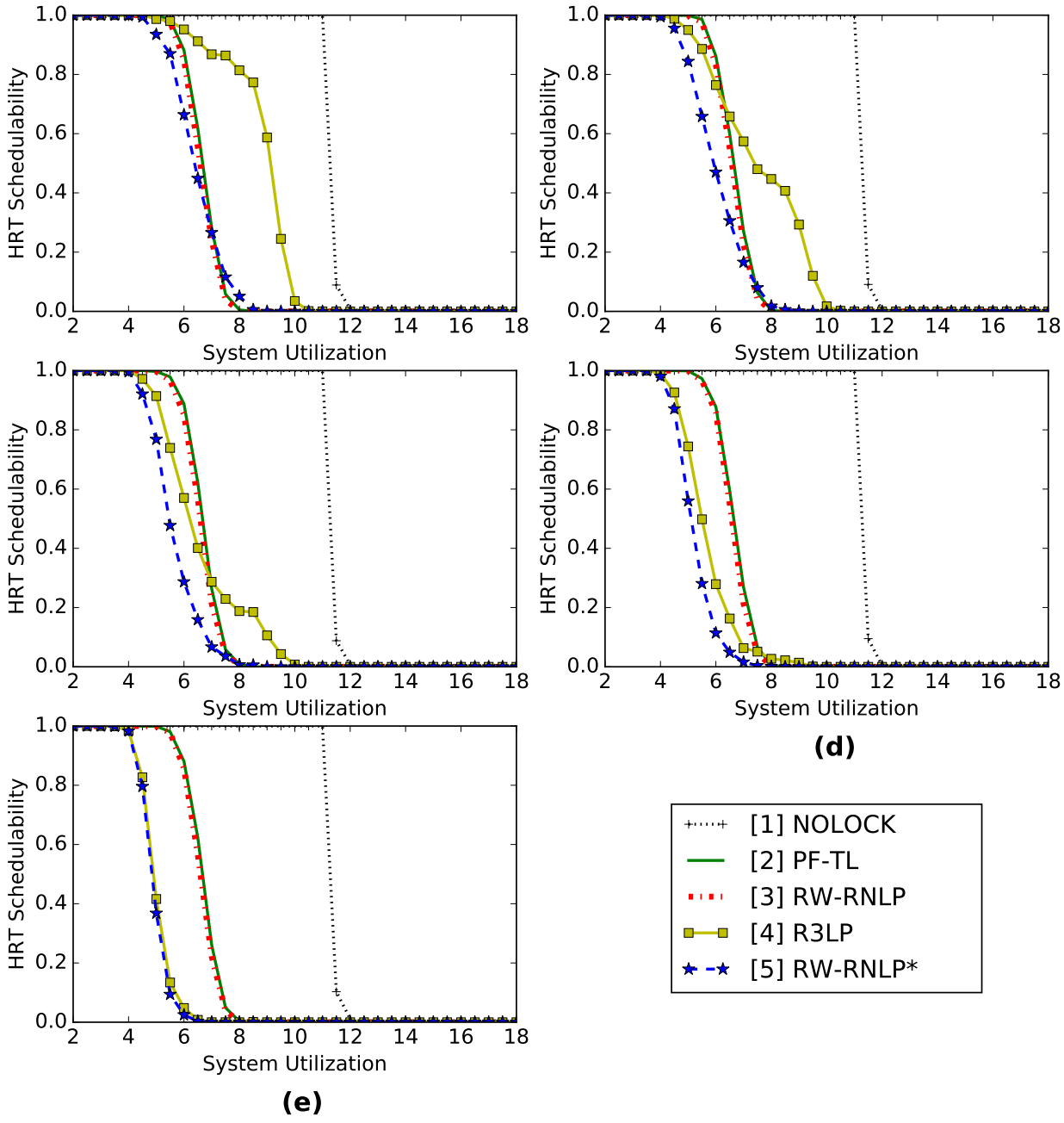



Figure 627: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, long critical-section lengths, and read probability 0 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

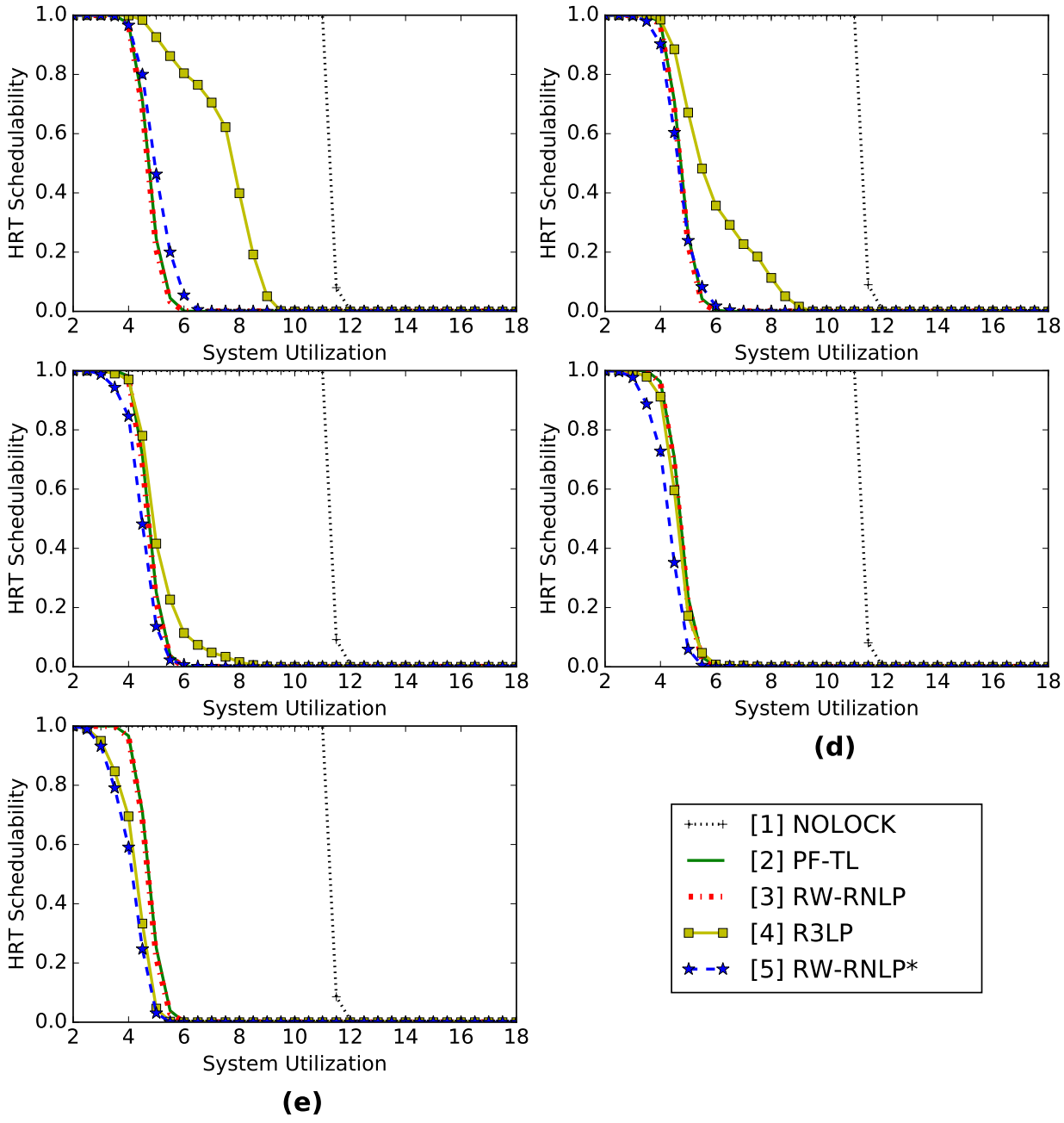



Figure 628: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, long critical-section lengths, and read probability 0 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

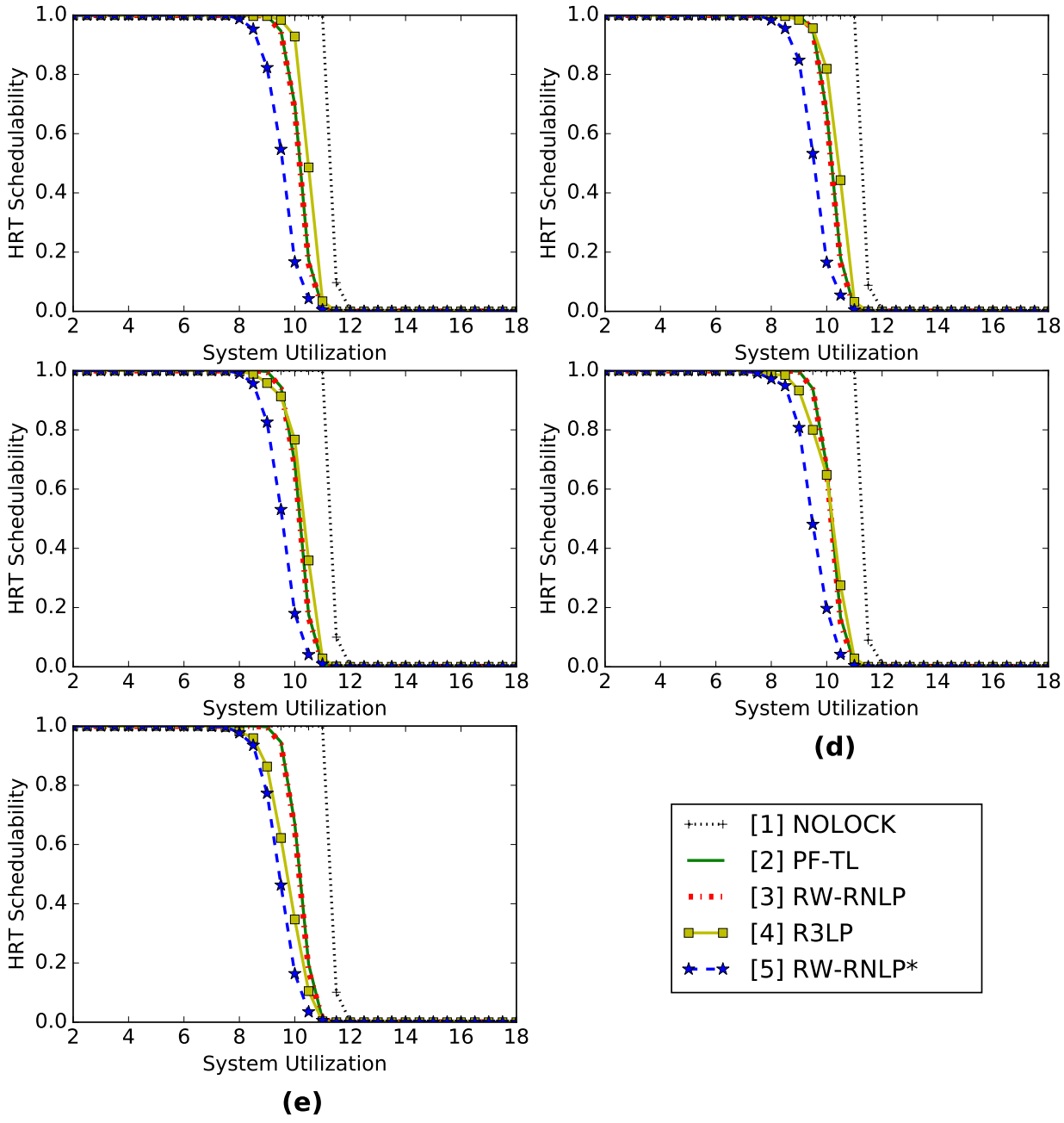



Figure 629: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, long critical-section lengths, and read probability 0.2 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

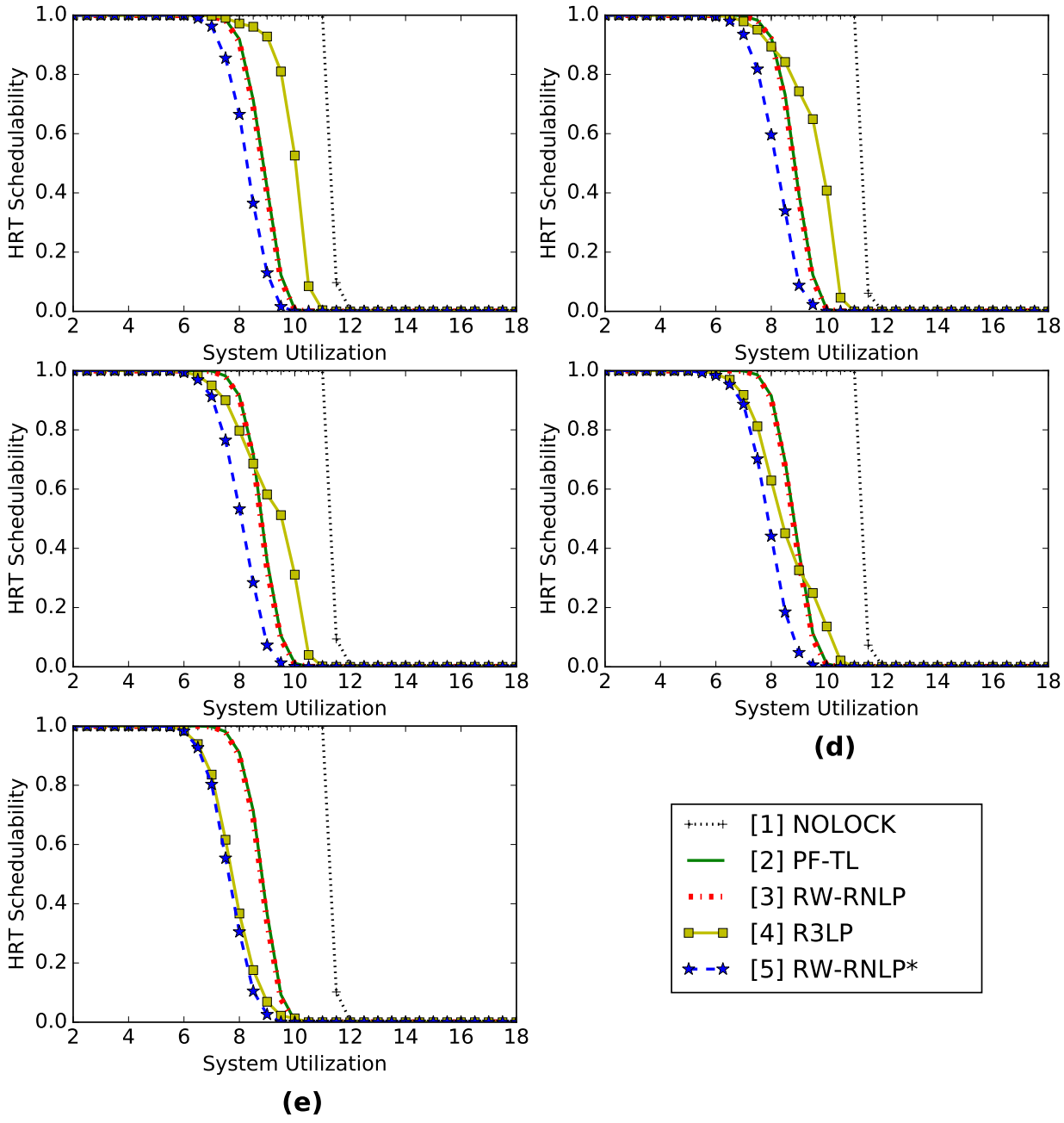



Figure 630: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, long critical-section lengths, and read probability 0.2 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

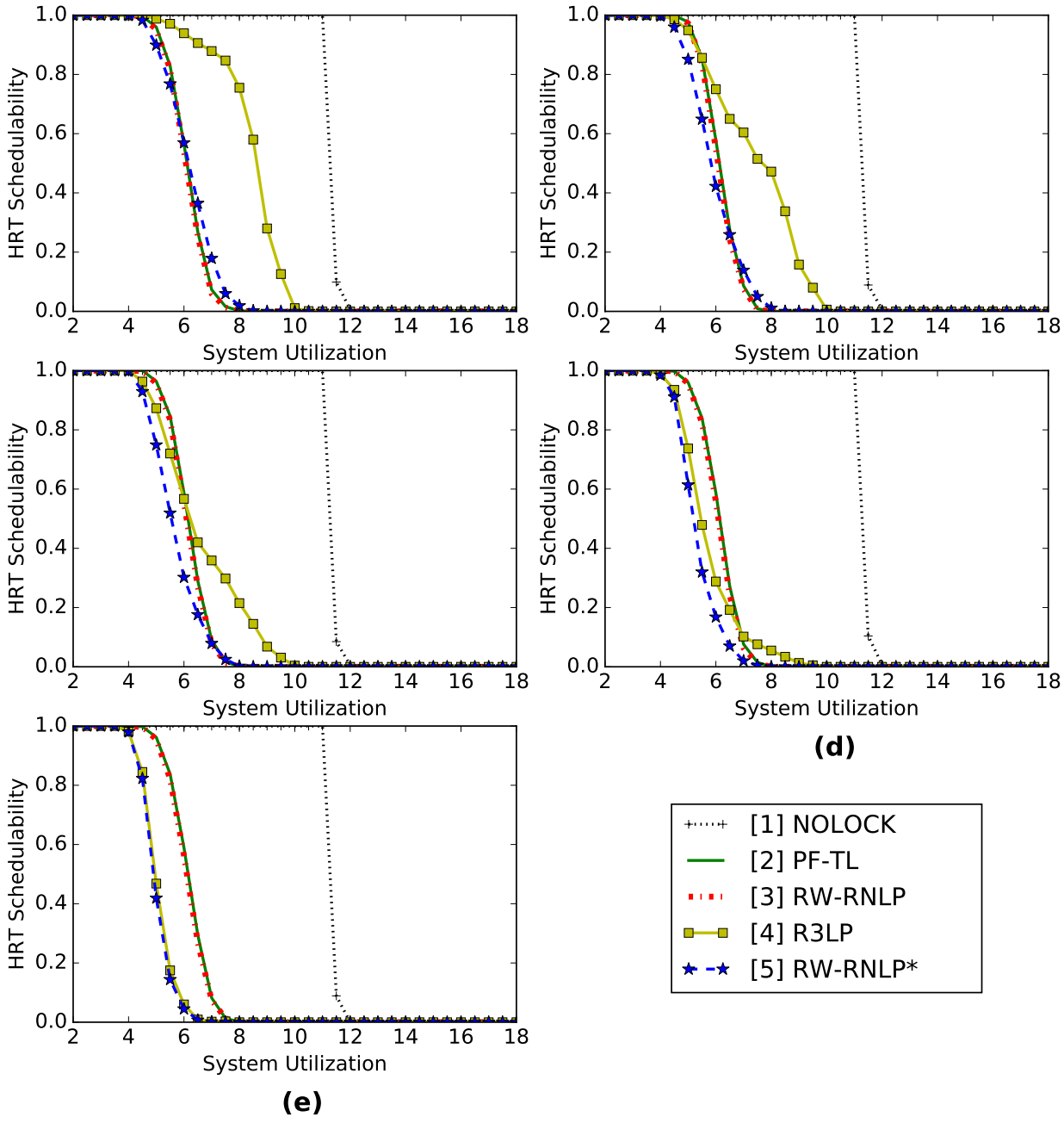



Figure 631: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, long critical-section lengths, and read probability 0.2 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

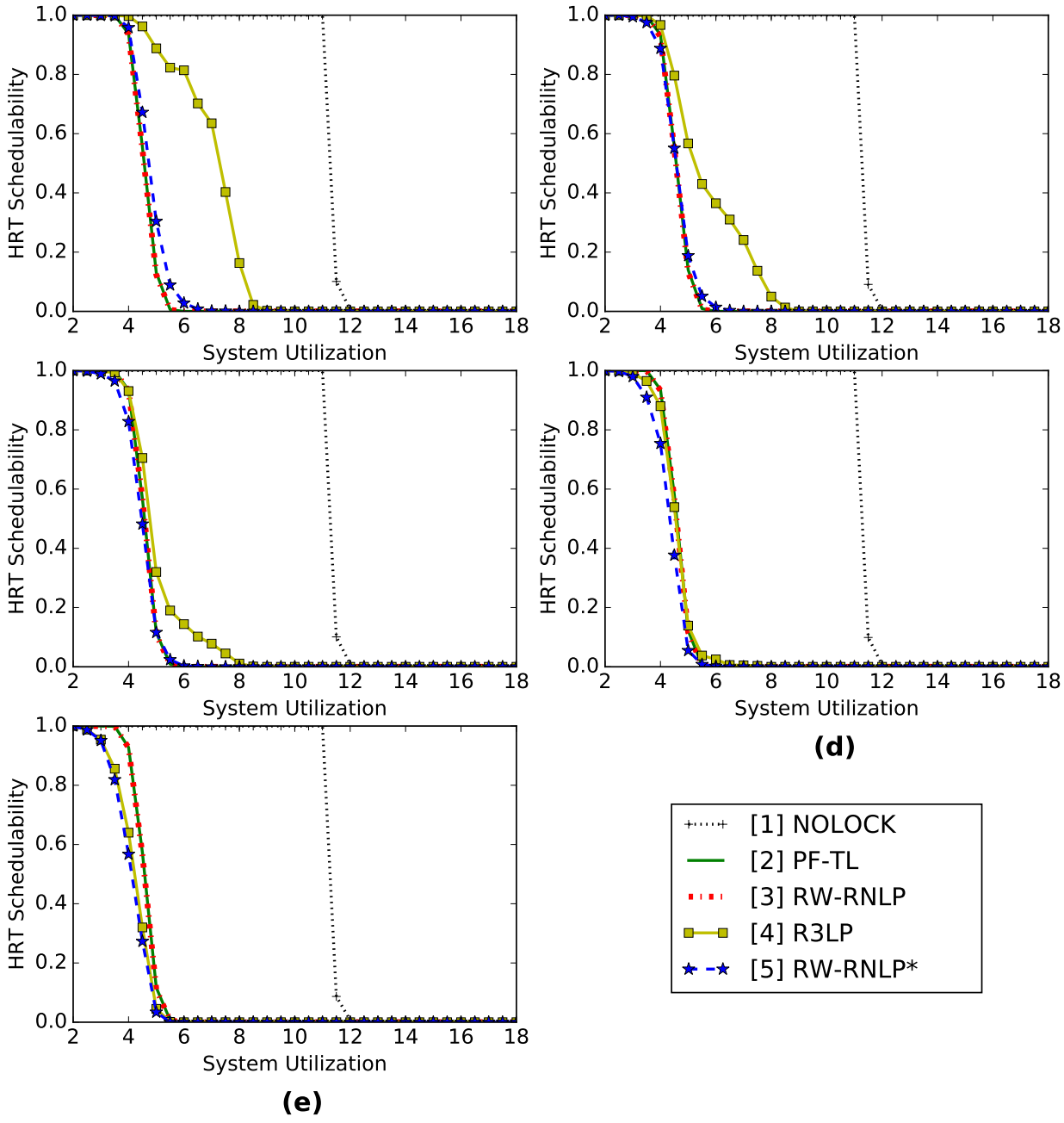



Figure 632: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, long critical-section lengths, and read probability 0.2 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

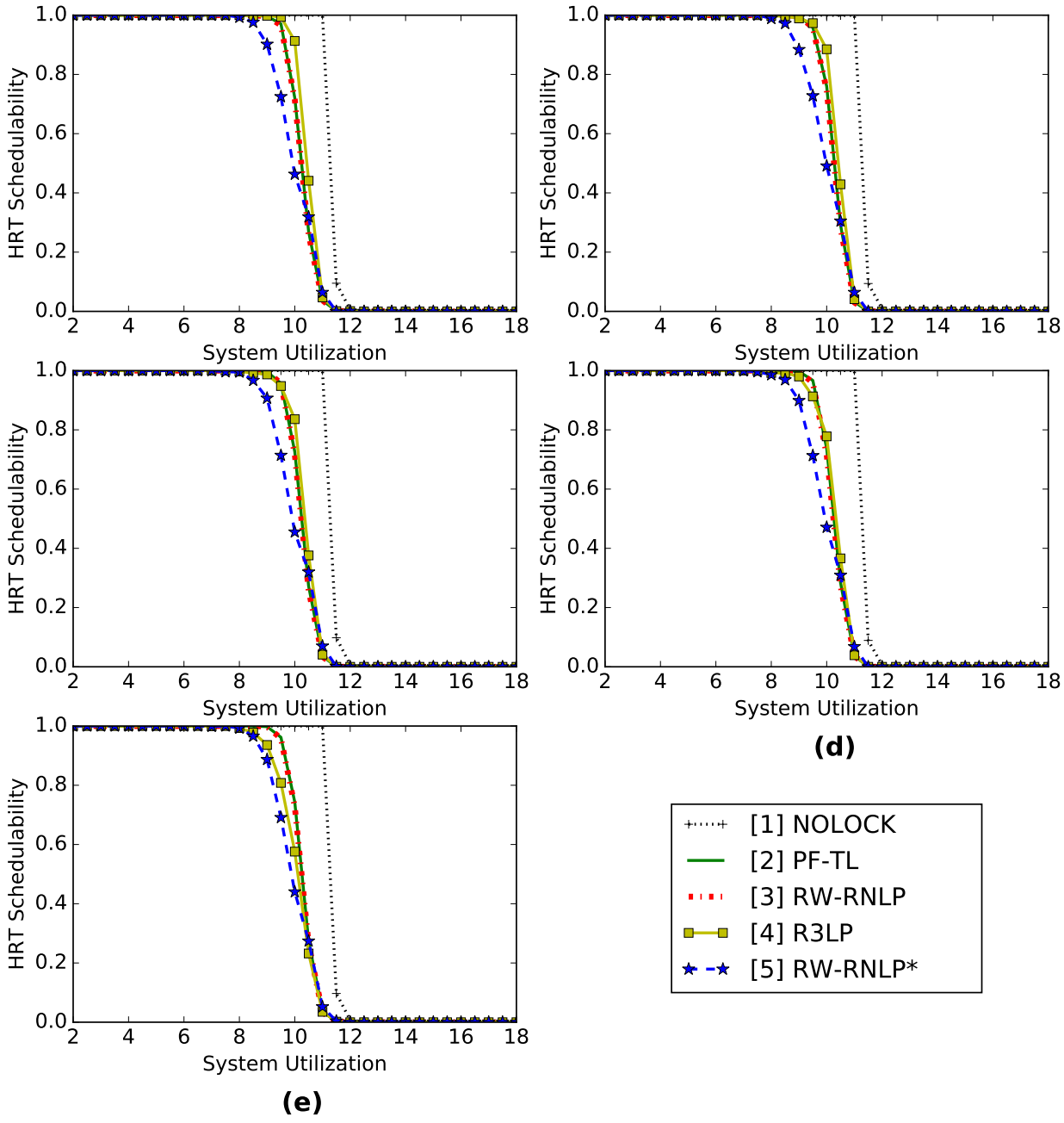



Figure 633: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, long critical-section lengths, and read probability 0.5 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

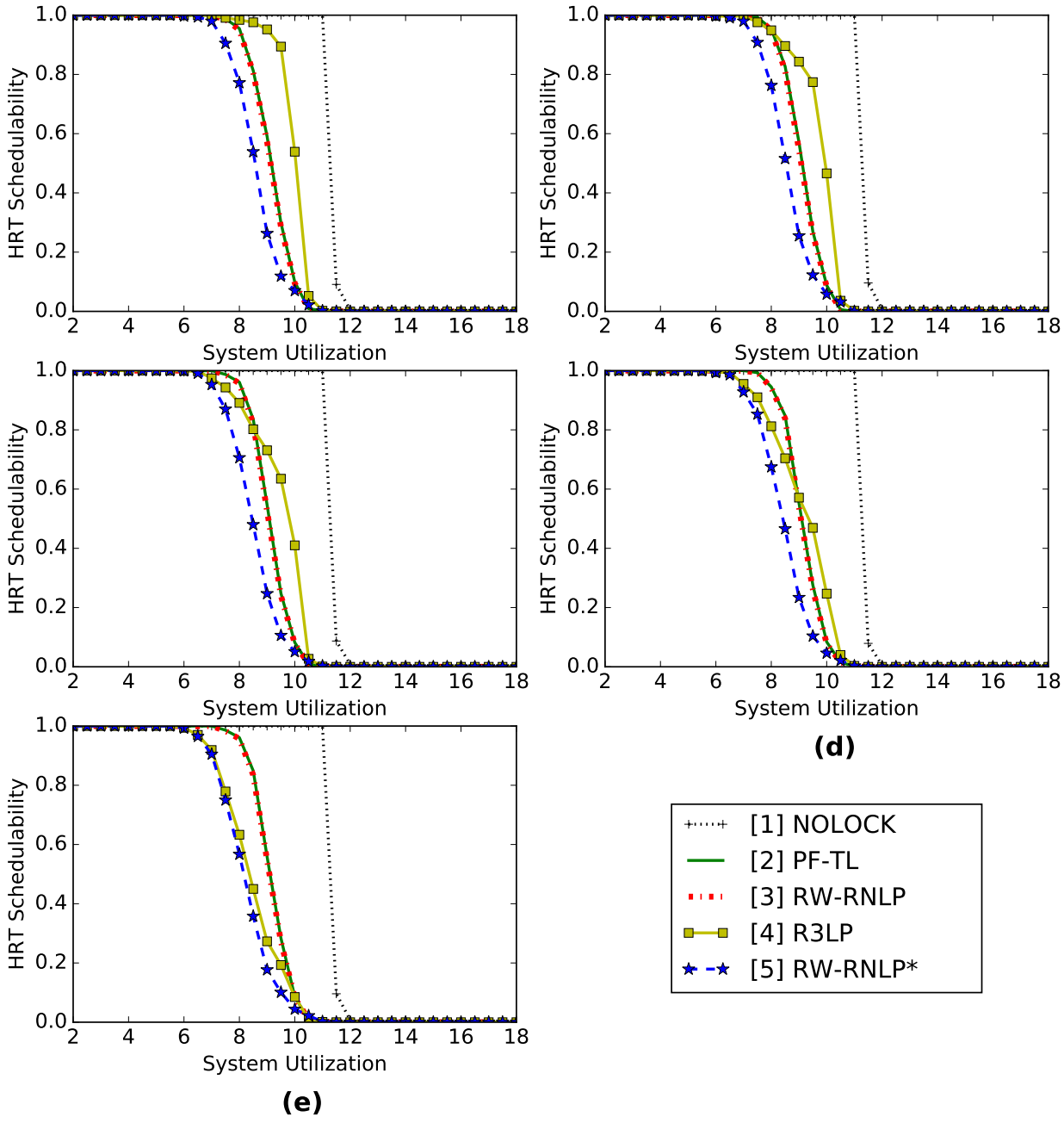



Figure 634: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, long critical-section lengths, and read probability 0.5 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

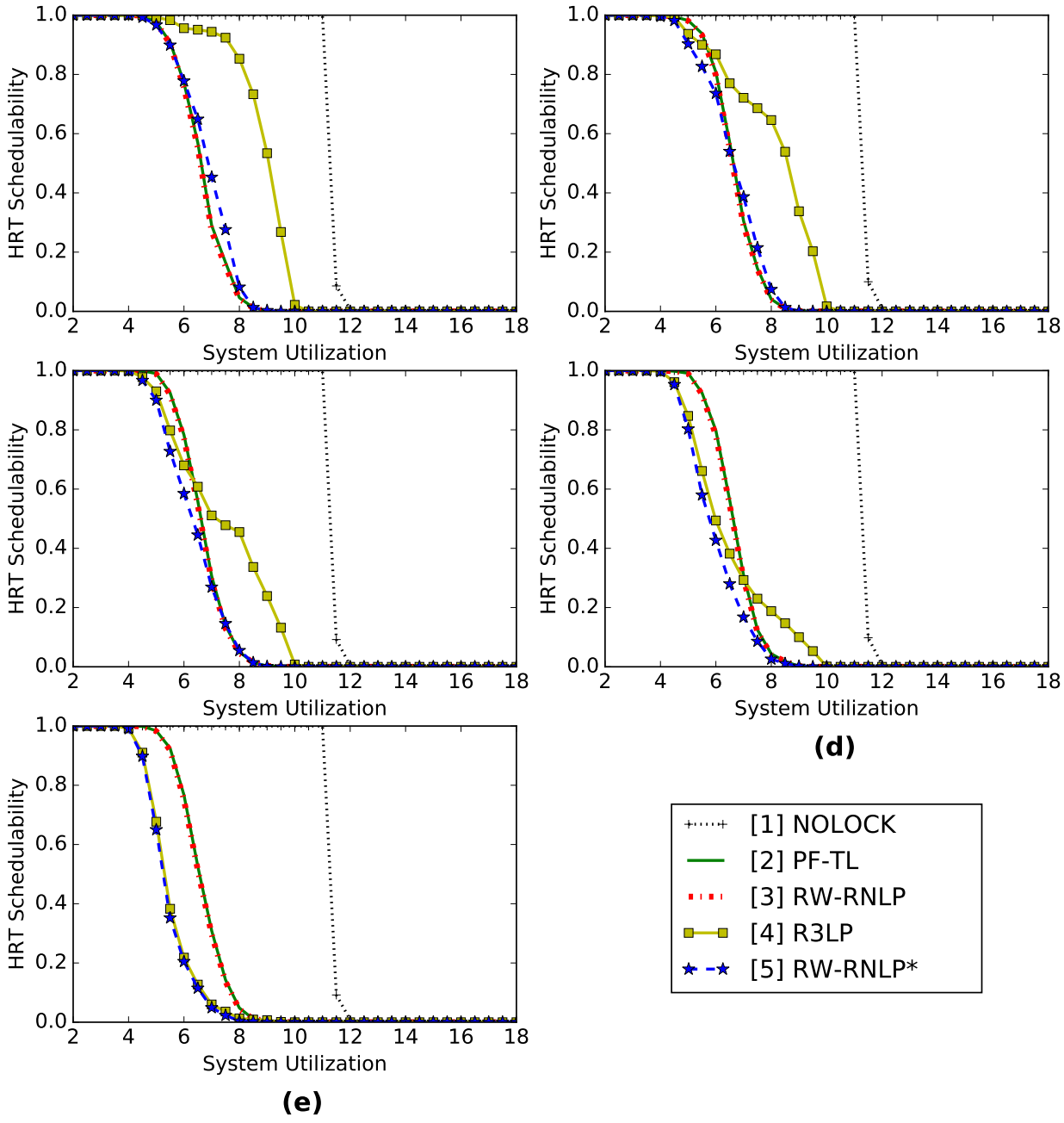



Figure 635: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, long critical-section lengths, and read probability 0.5 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

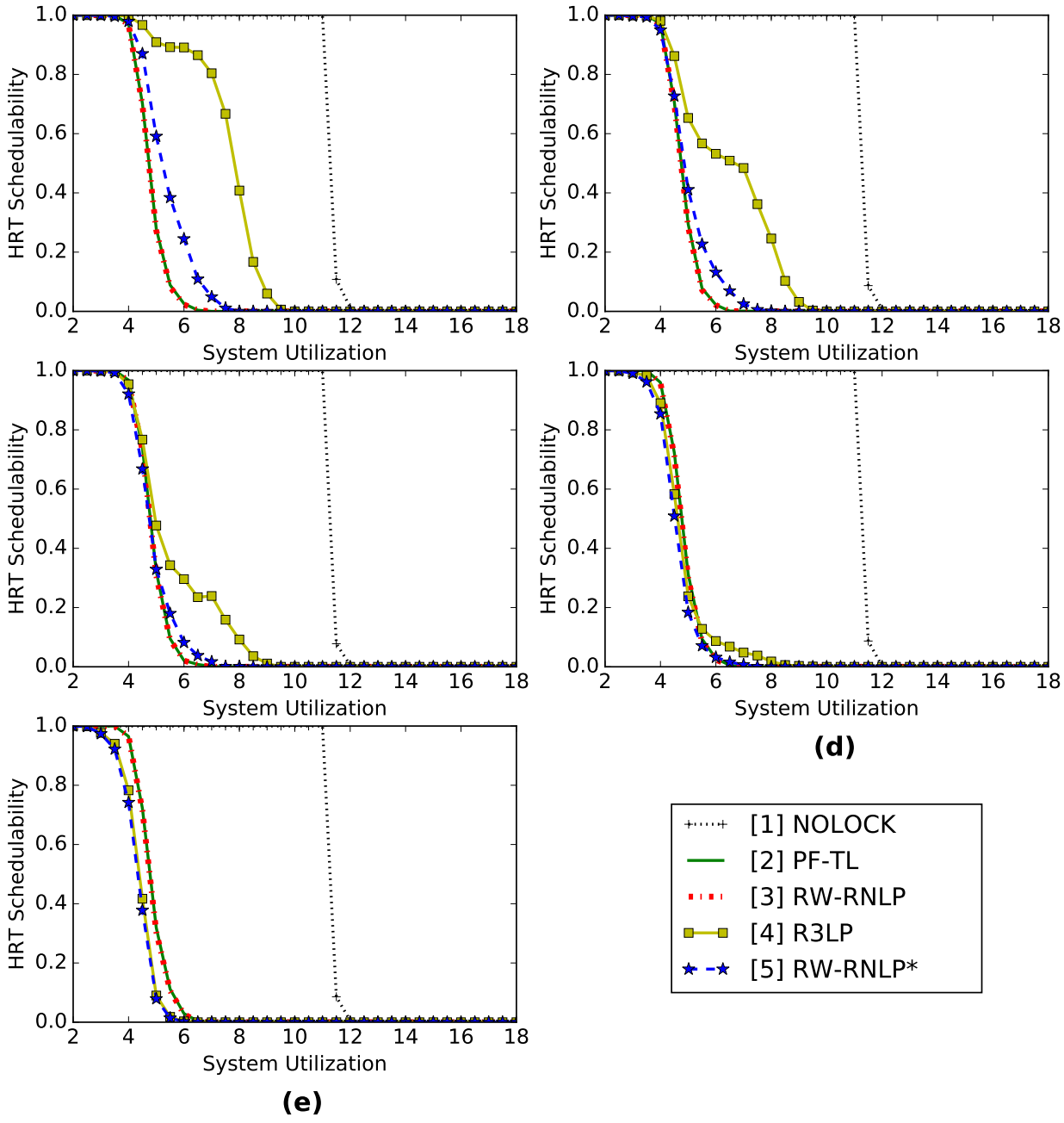



Figure 636: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, long critical-section lengths, and read probability 0.5 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

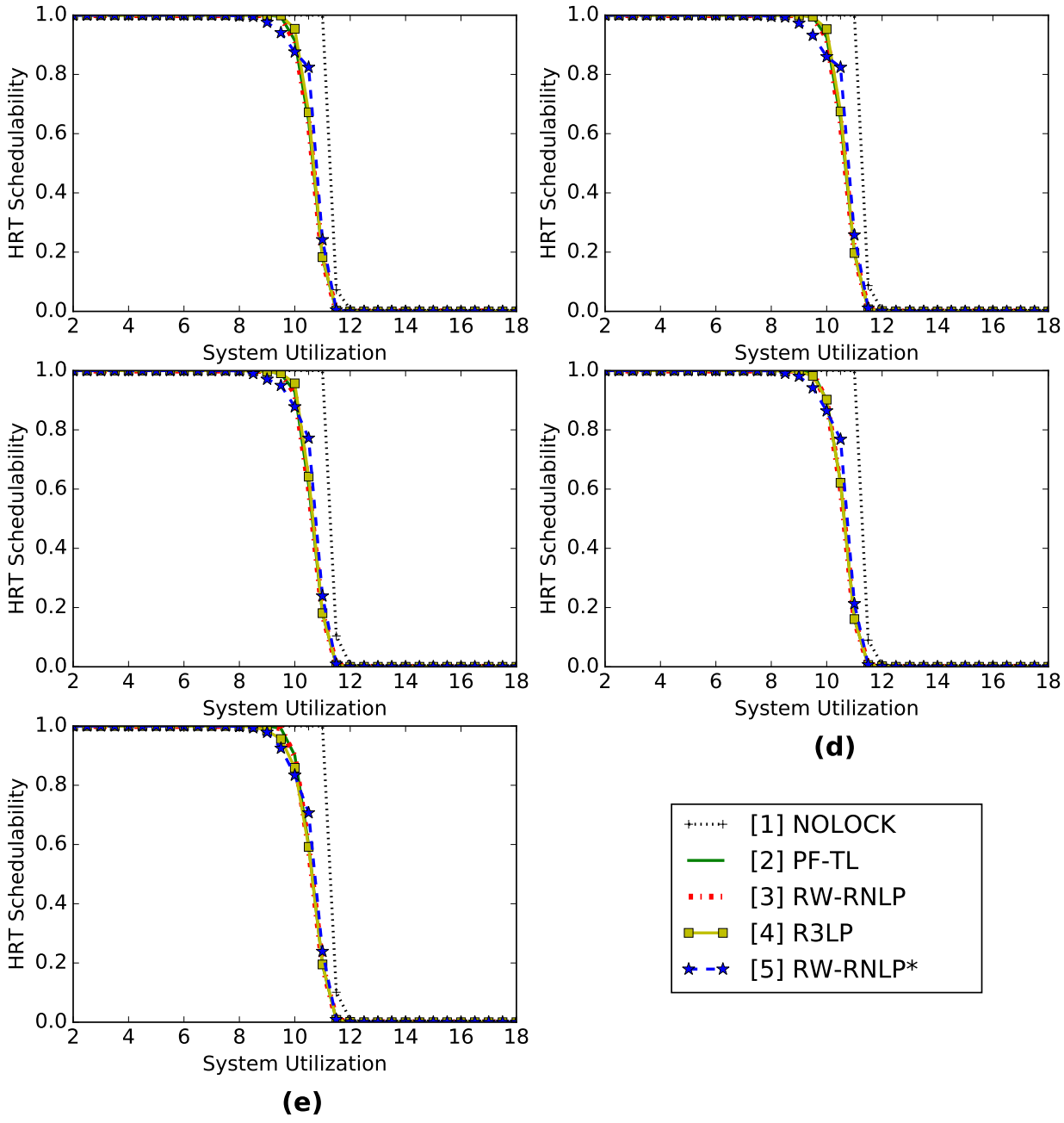



Figure 637: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, long critical-section lengths, and read probability 0.8 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

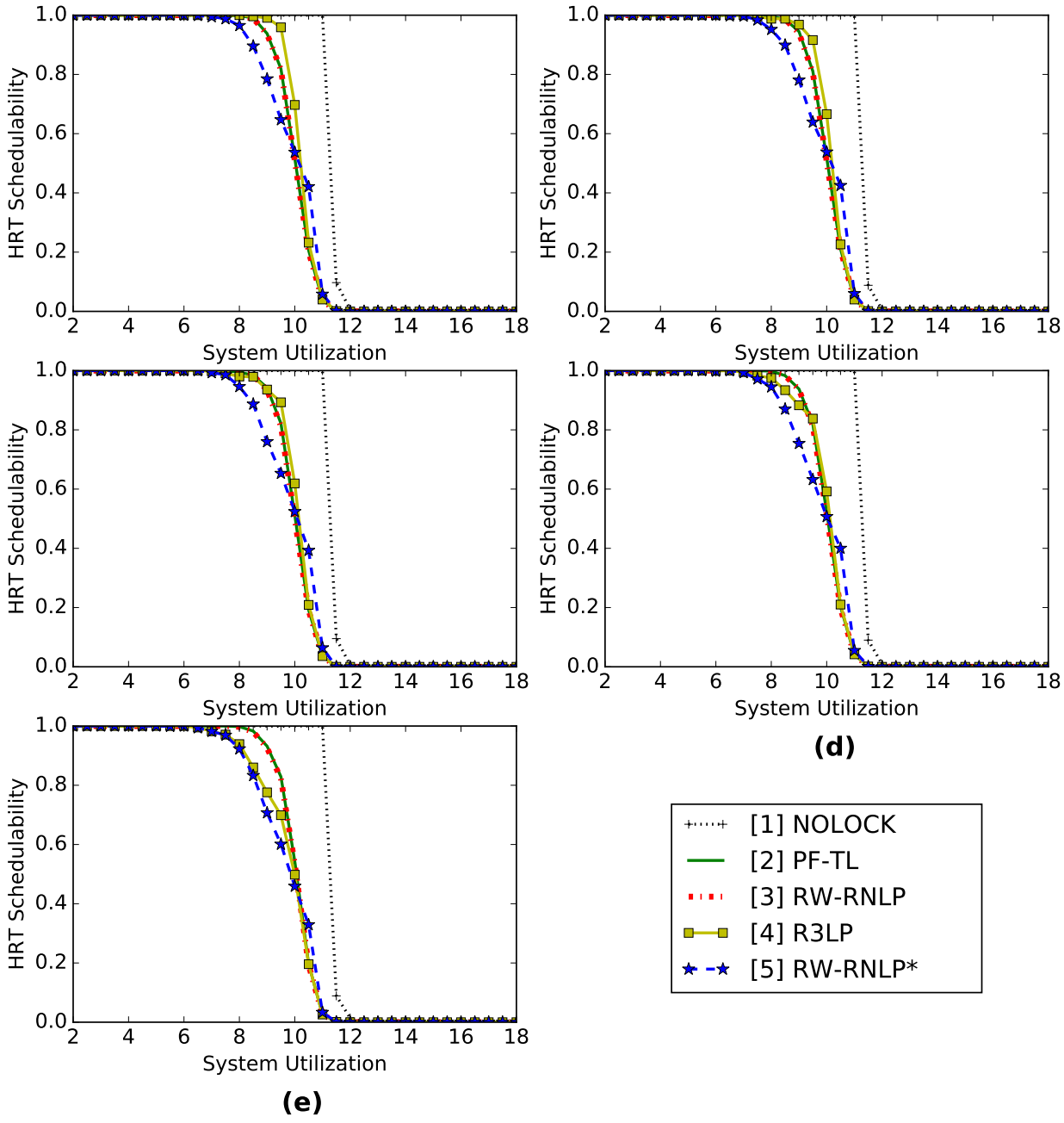



Figure 638: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, long critical-section lengths, and read probability 0.8 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

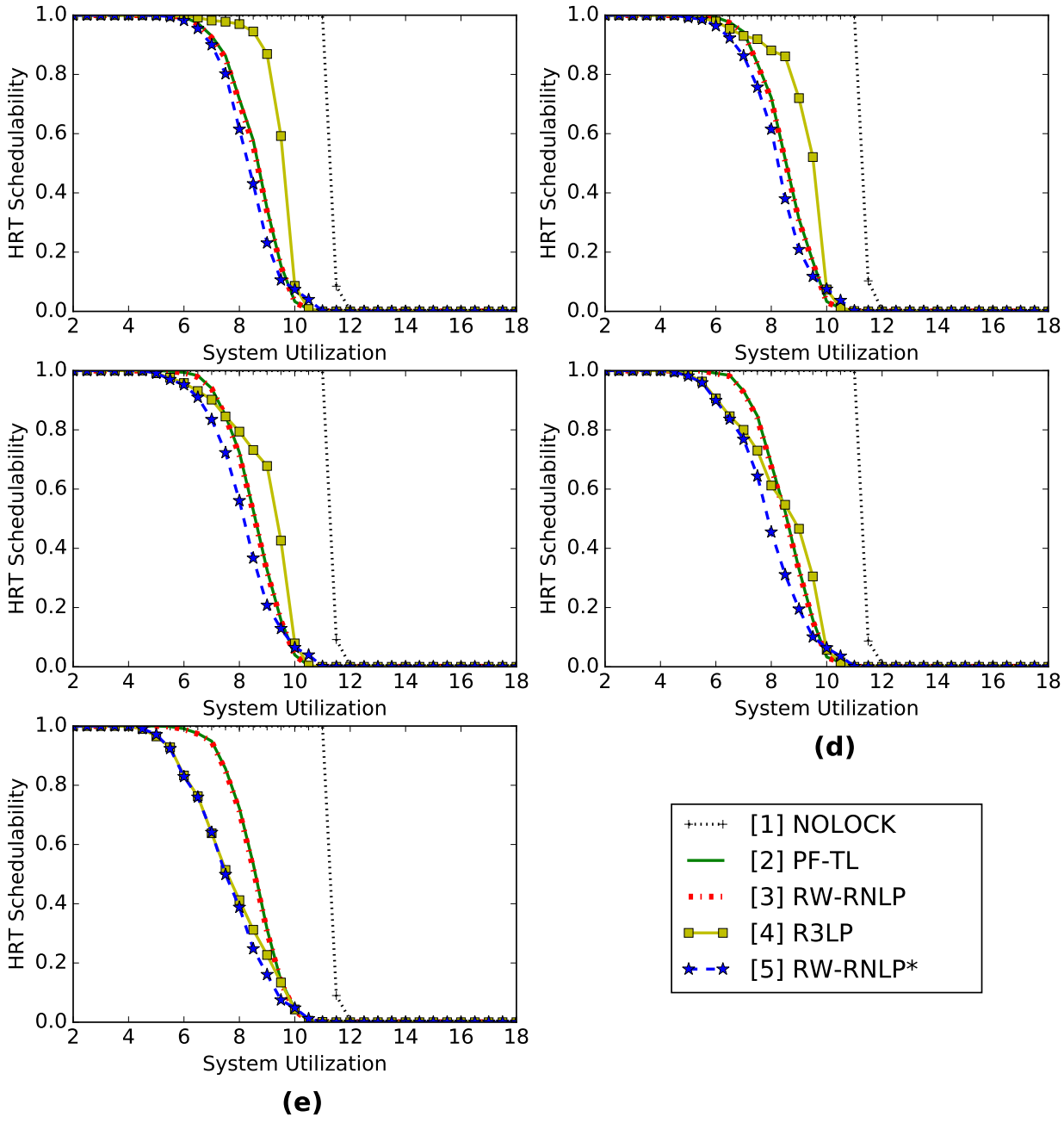



Figure 639: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, long critical-section lengths, and read probability 0.8 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

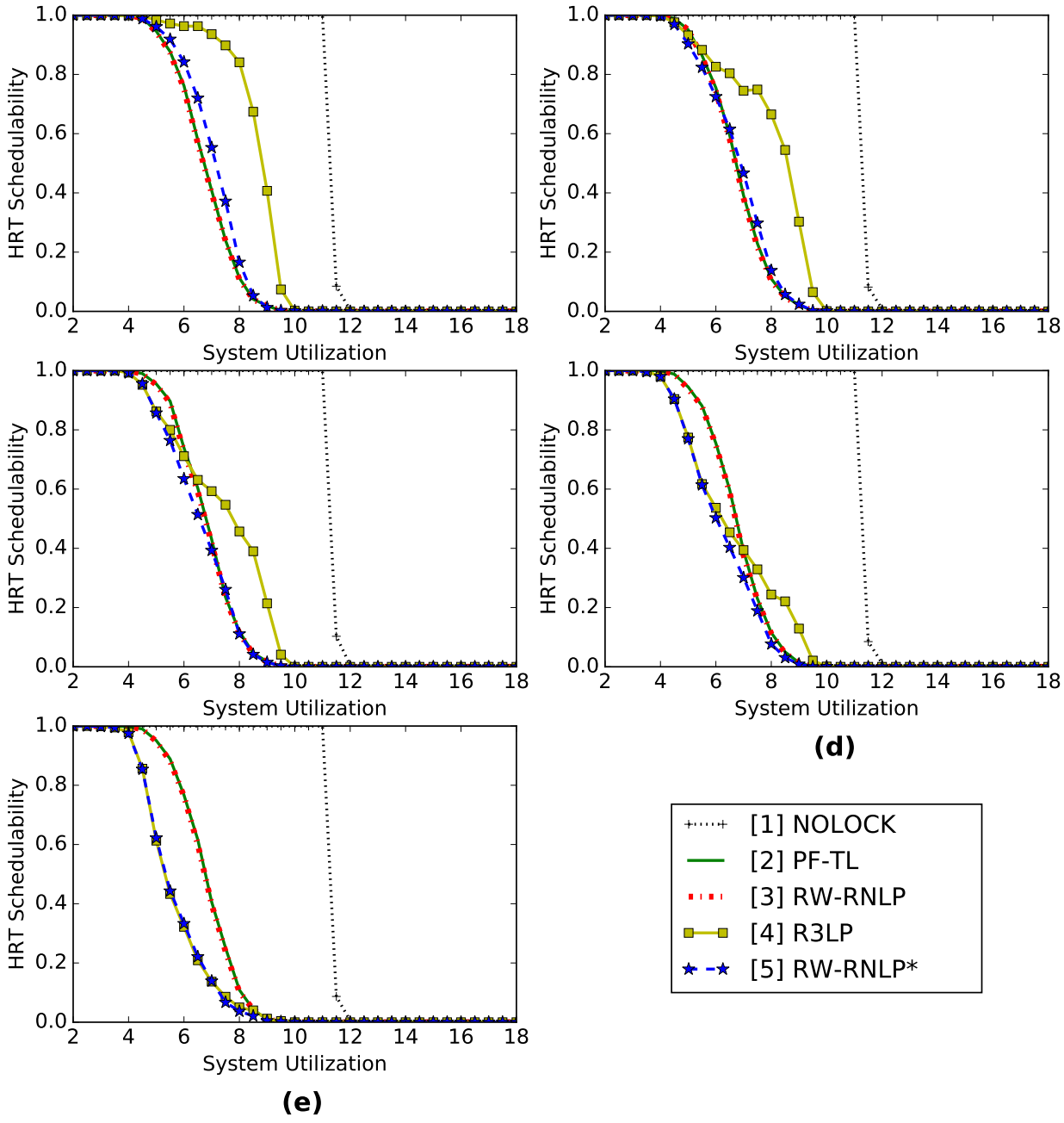



Figure 640: Hard real-time schedulability results with varying nested probabilities for the scenario with medium task utilizations, long periods, long critical-section lengths, and read probability 0.8 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

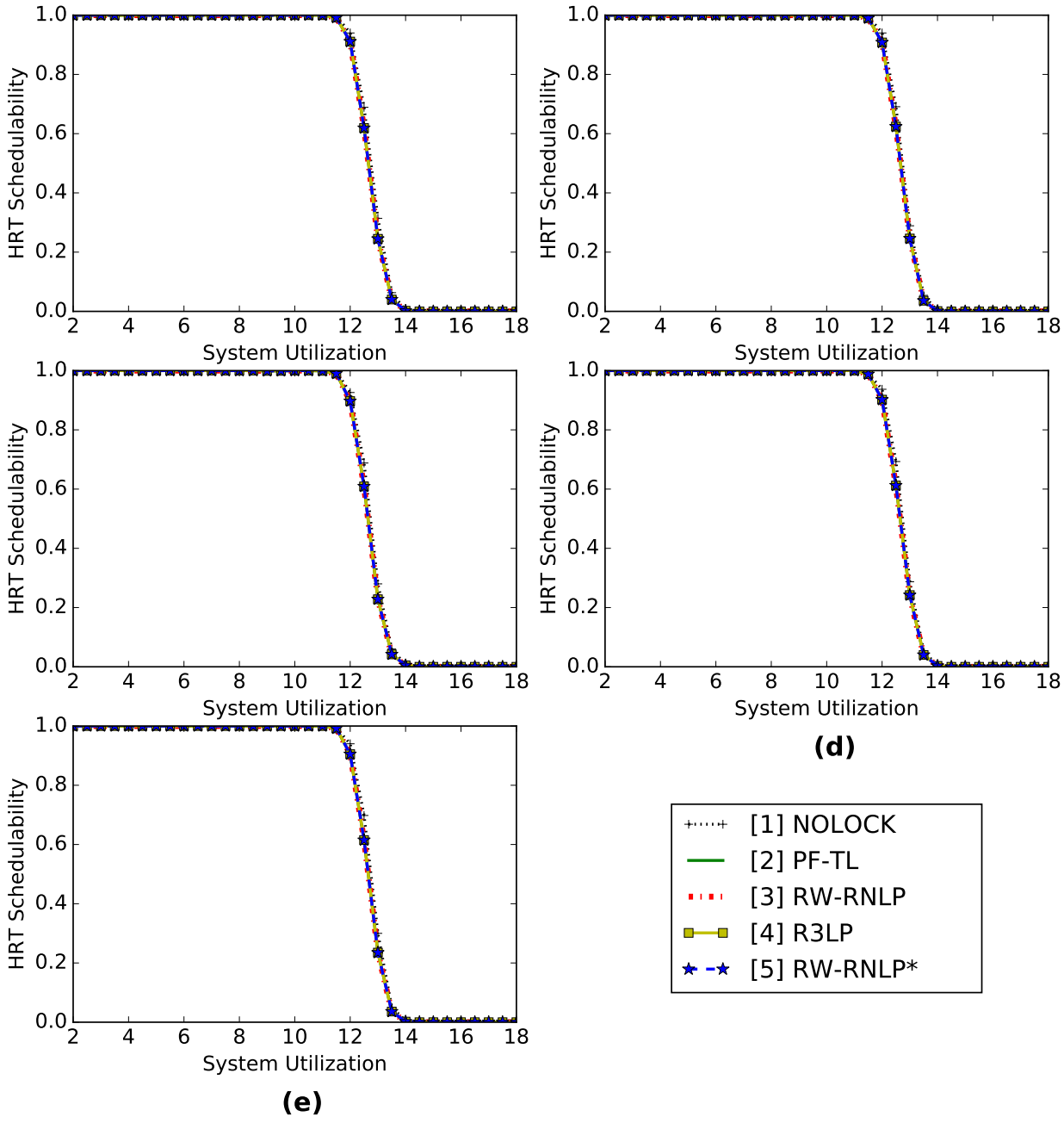



Figure 641: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, short critical-section lengths, and read probability 0 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

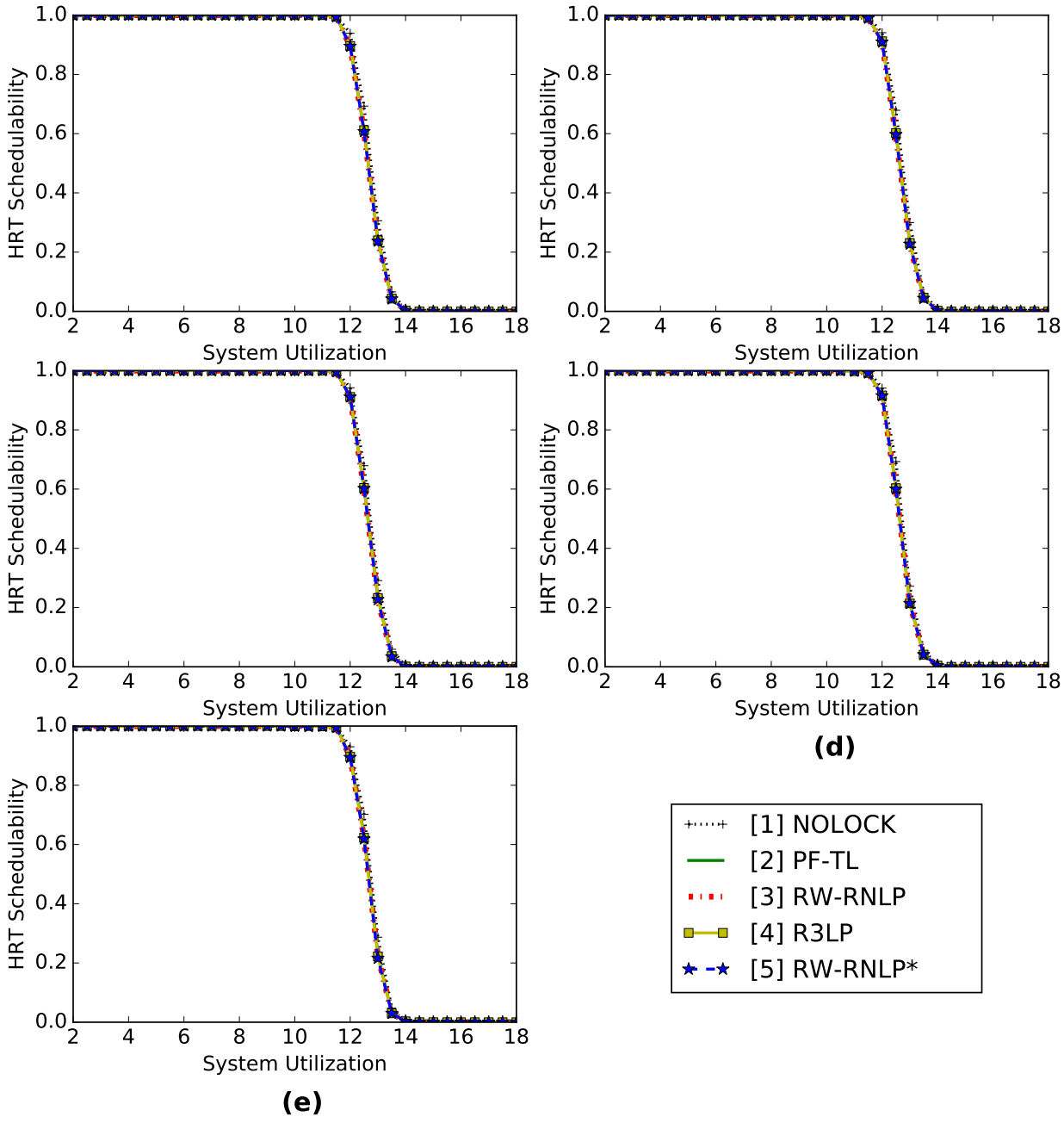



Figure 642: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, short critical-section lengths, and read probability 0 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

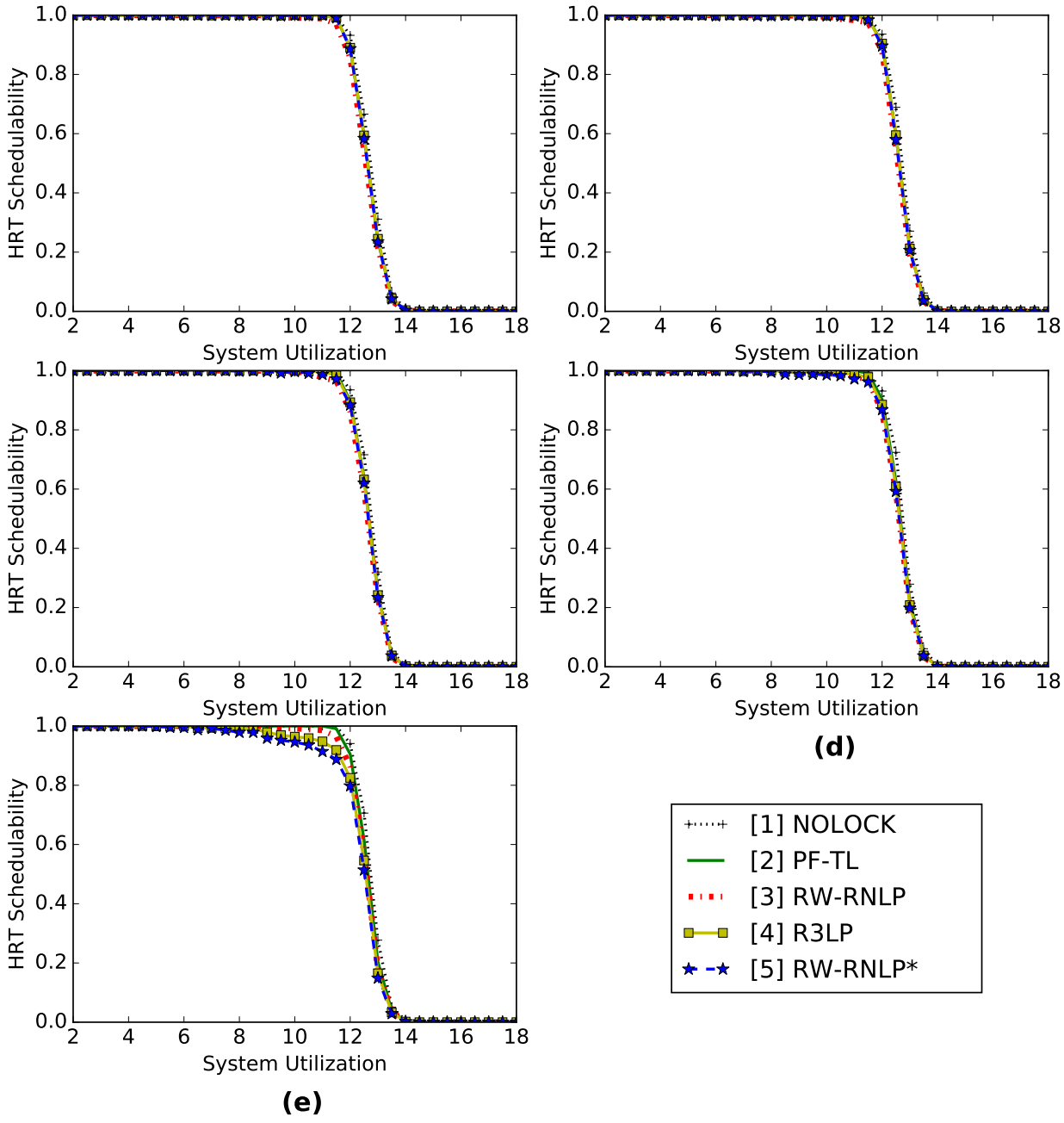



Figure 643: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, short critical-section lengths, and read probability 0 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

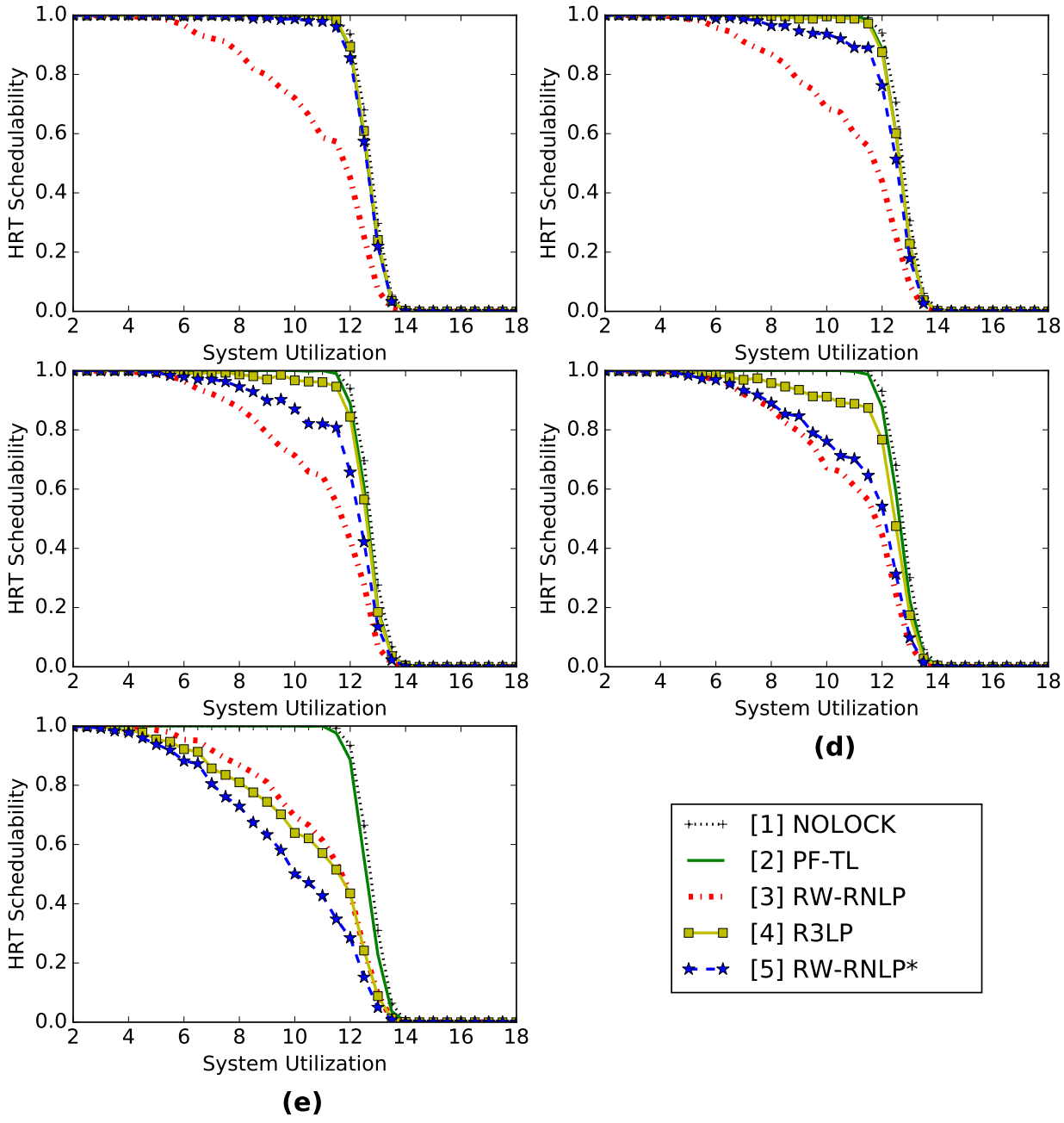



Figure 644: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, short critical-section lengths, and read probability 0 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

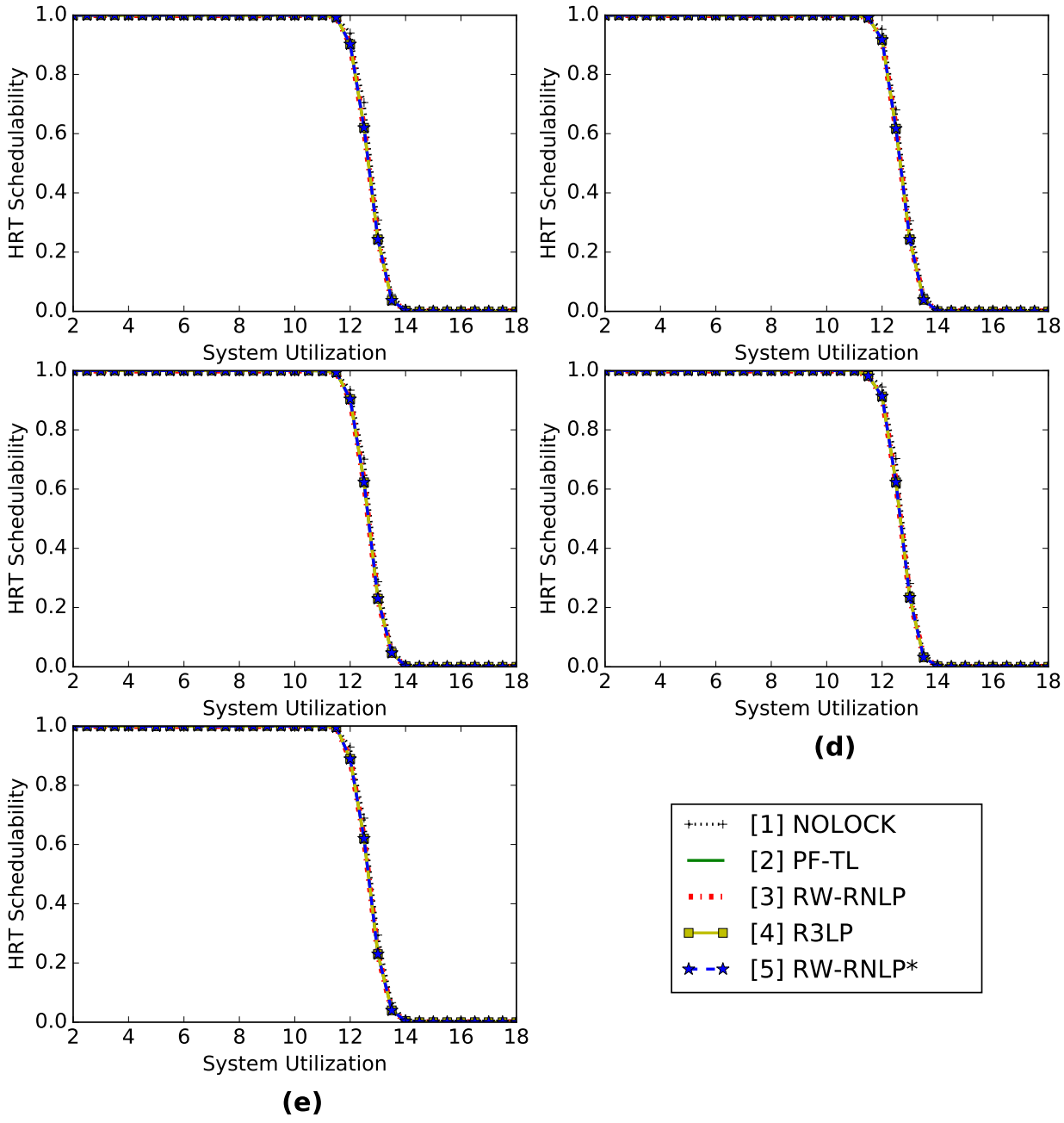



Figure 645: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, short critical-section lengths, and read probability 0.2 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

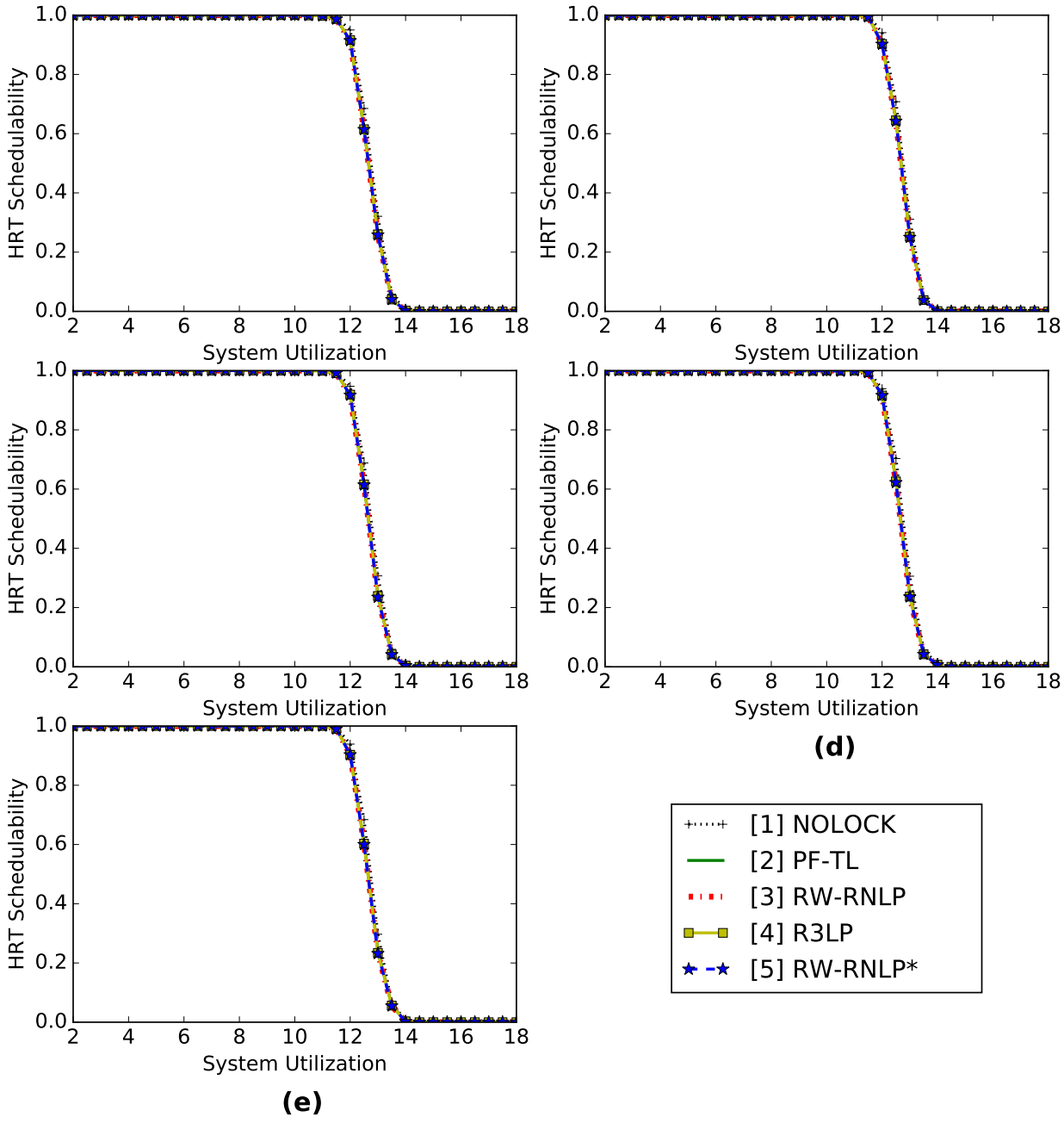



Figure 646: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, short critical-section lengths, and read probability 0.2 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

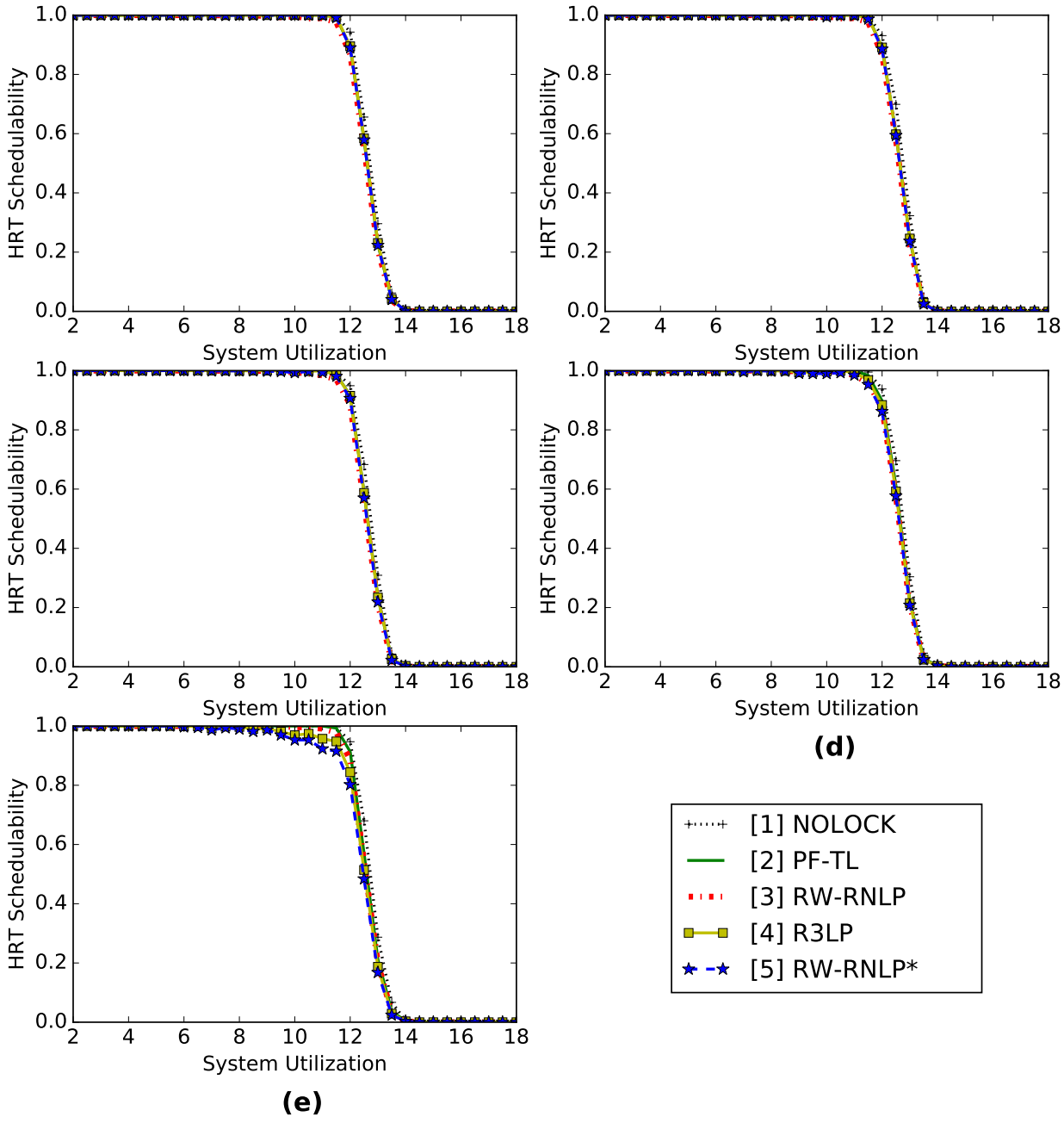



Figure 647: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, short critical-section lengths, and read probability 0.2 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

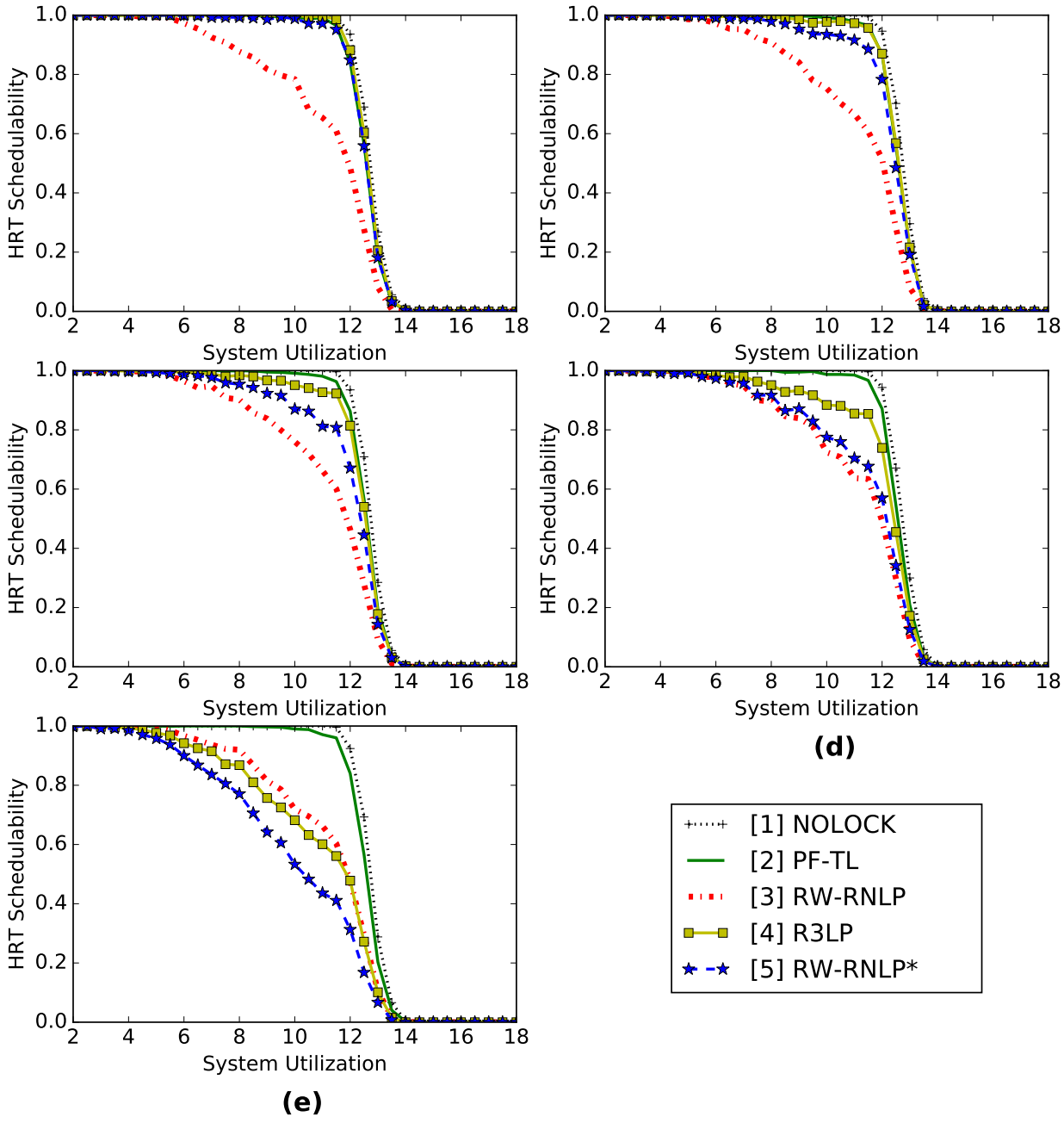



Figure 648: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, short critical-section lengths, and read probability 0.2 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

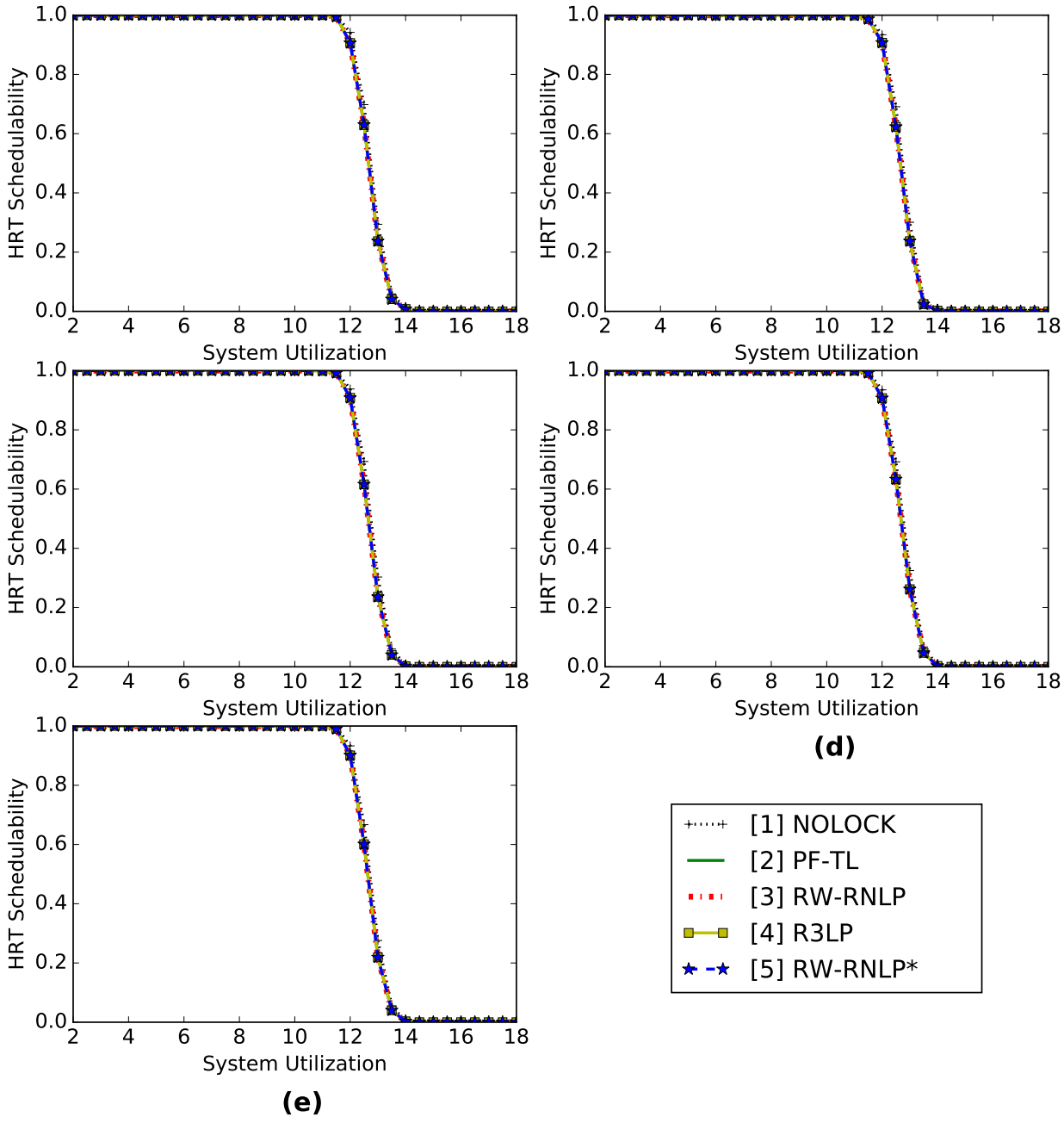



Figure 649: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, short critical-section lengths, and read probability 0.5 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

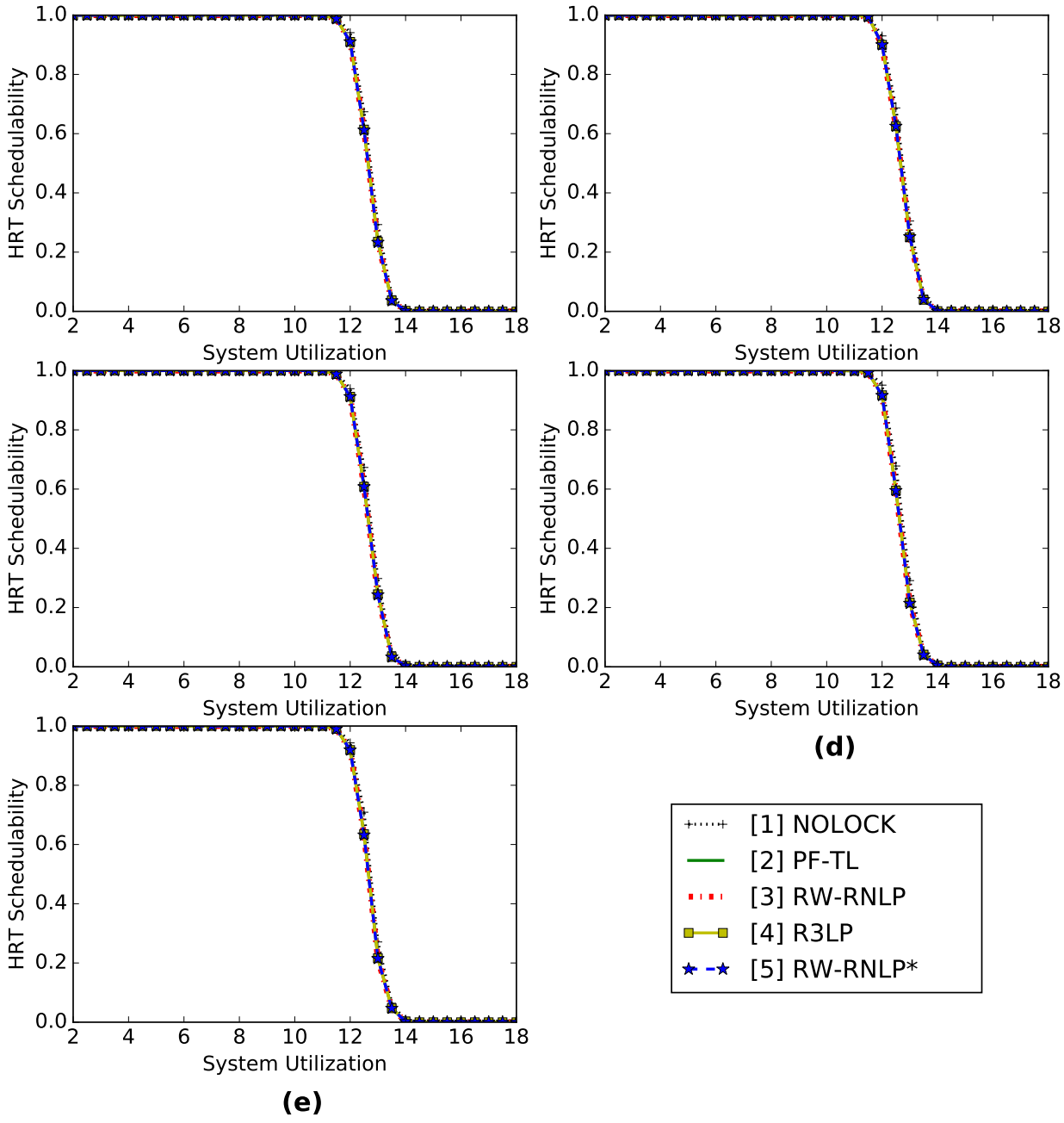



Figure 650: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, short critical-section lengths, and read probability 0.5 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

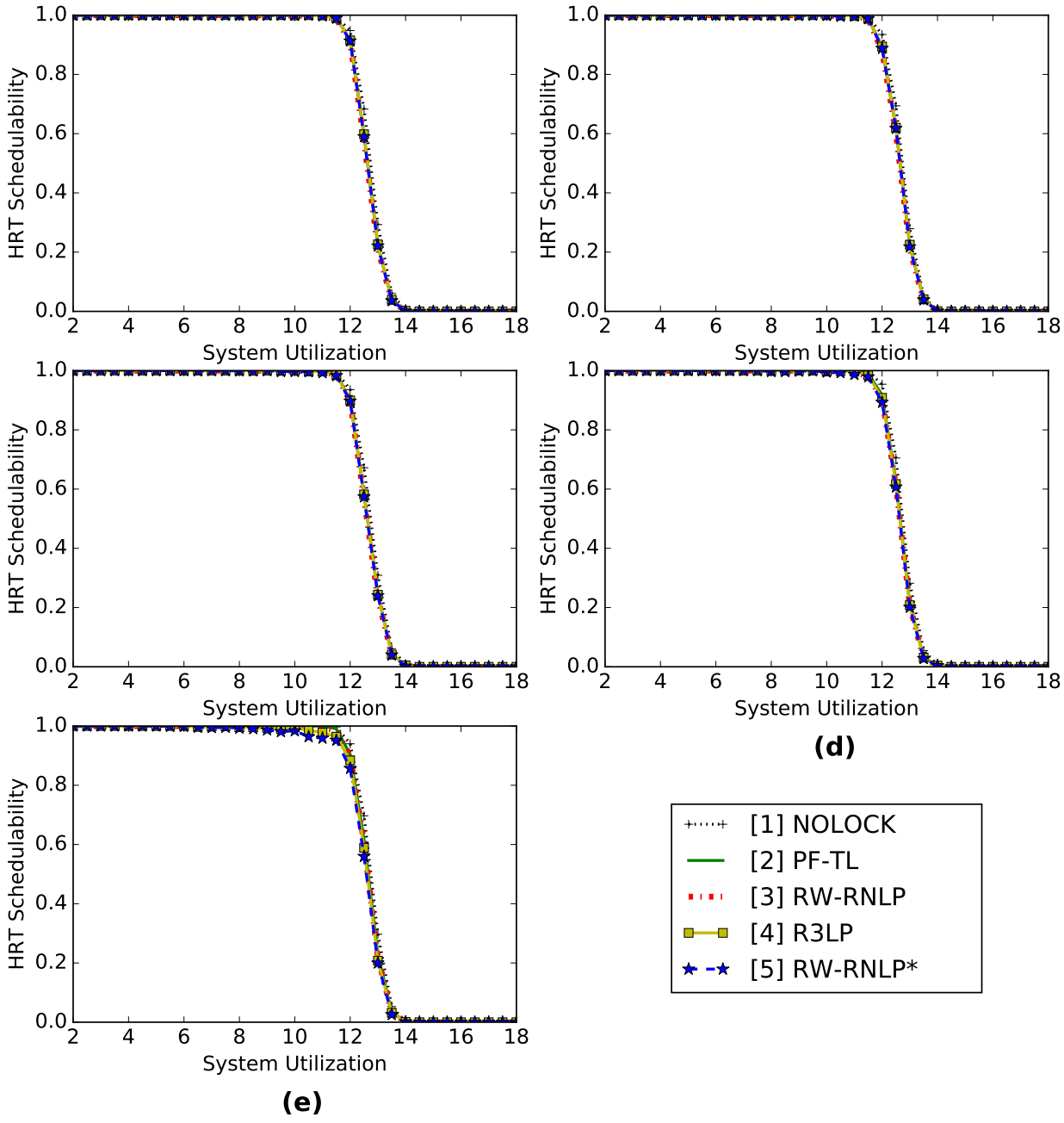



Figure 651: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, short critical-section lengths, and read probability 0.5 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

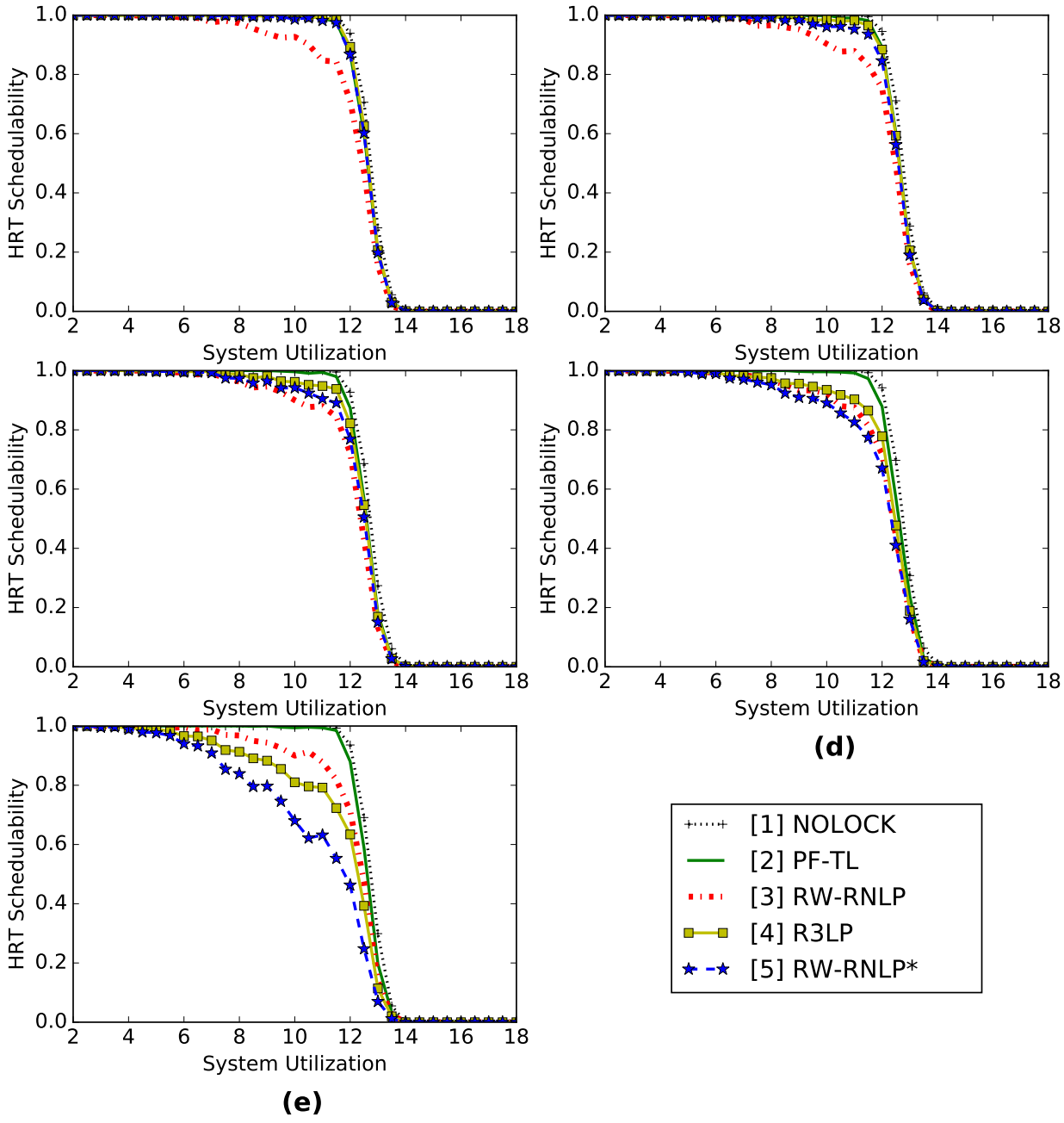



Figure 652: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, short critical-section lengths, and read probability 0.5 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

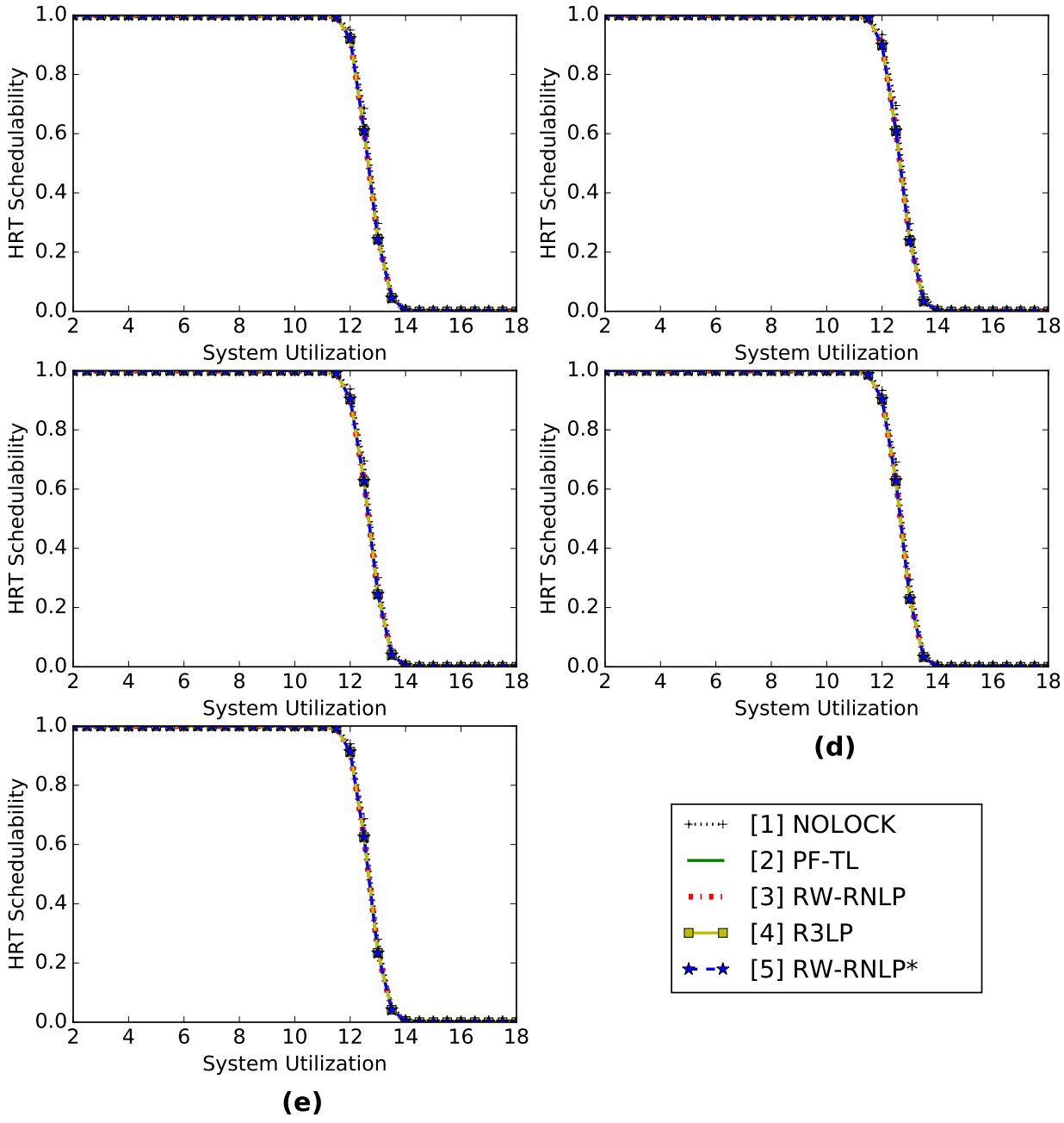



Figure 653: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, short critical-section lengths, and read probability 0.8 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

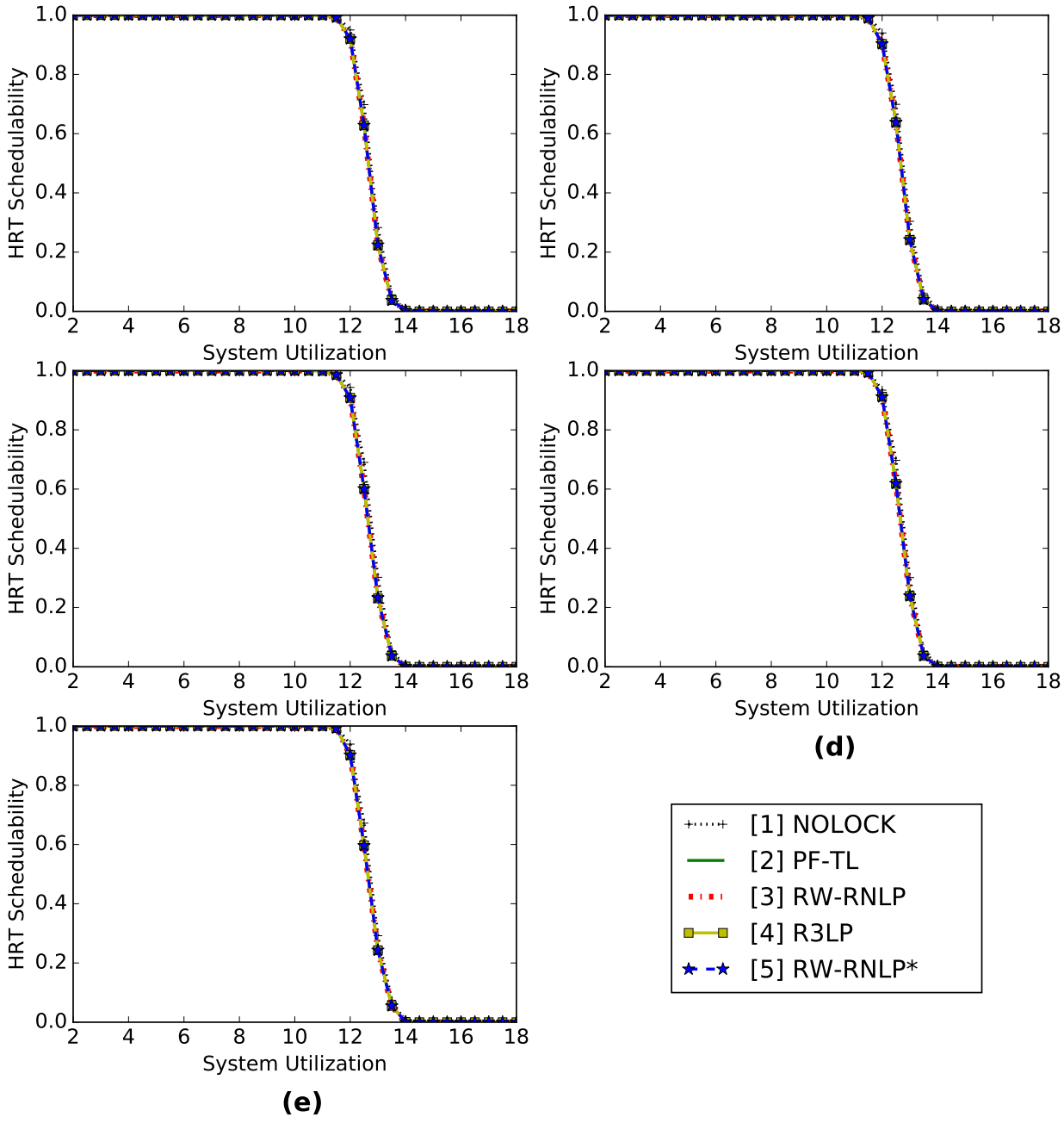



Figure 654: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, short critical-section lengths, and read probability 0.8 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

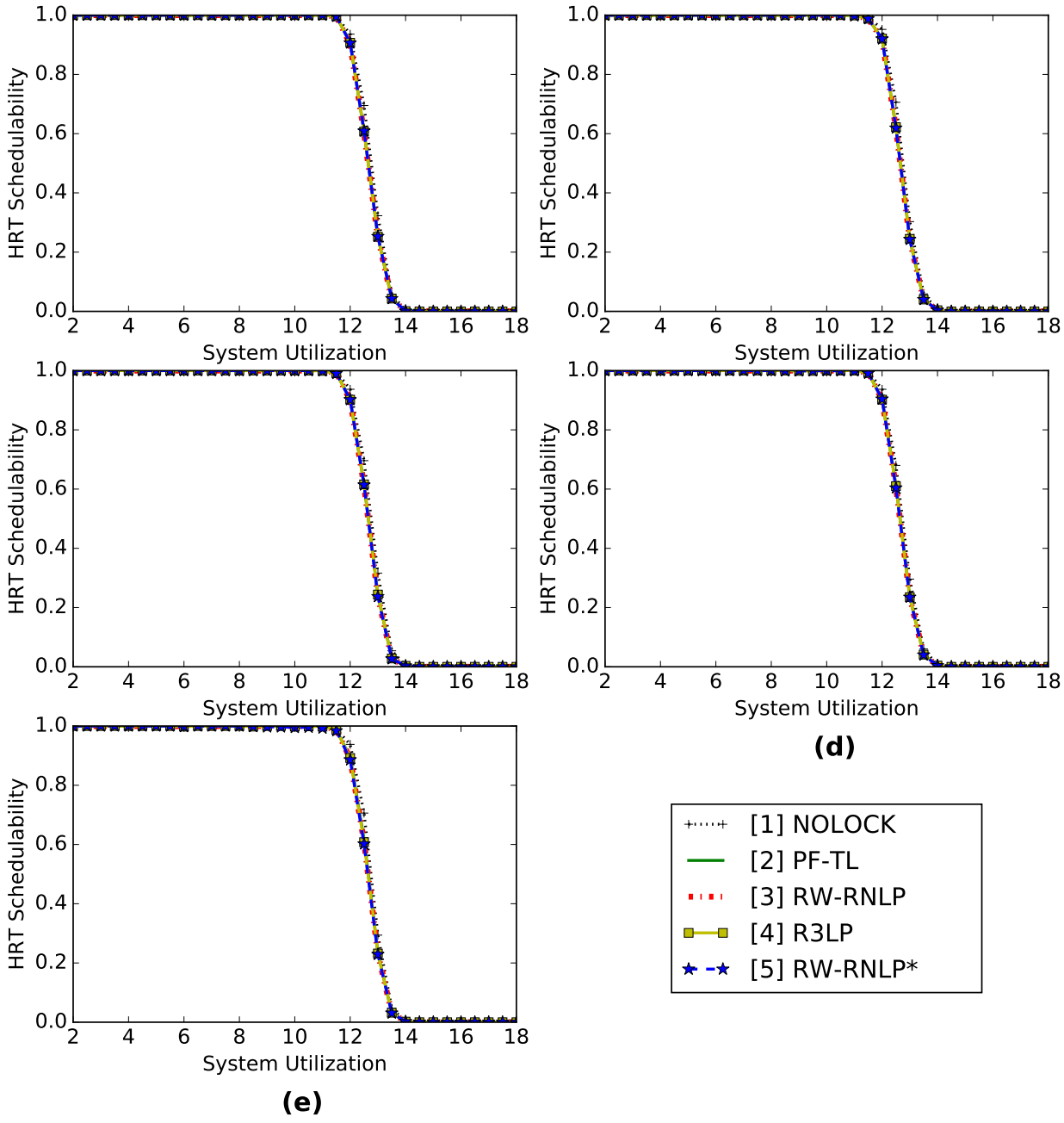



Figure 655: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, short critical-section lengths, and read probability 0.8 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

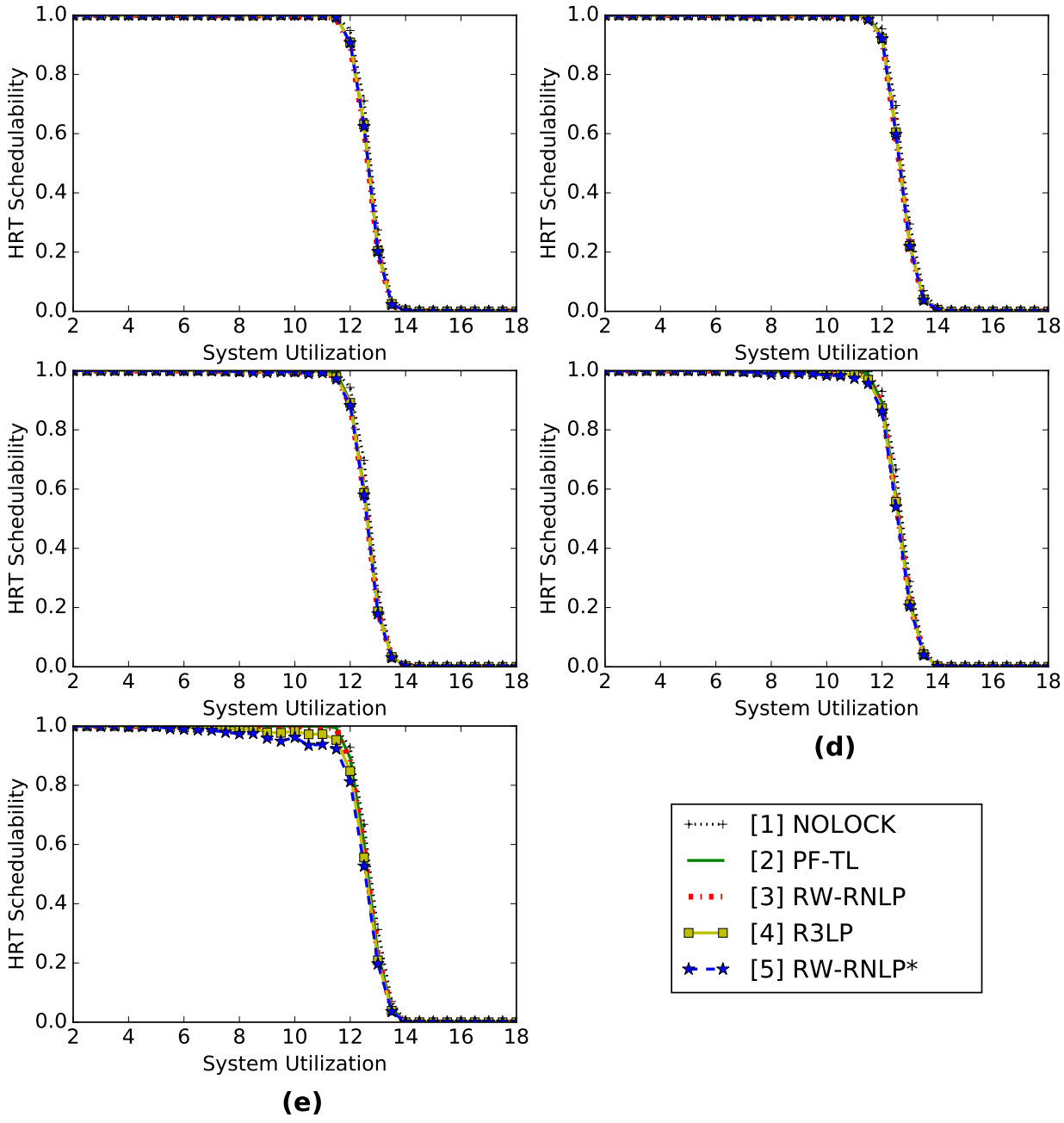



Figure 656: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, short critical-section lengths, and read probability 0.8 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

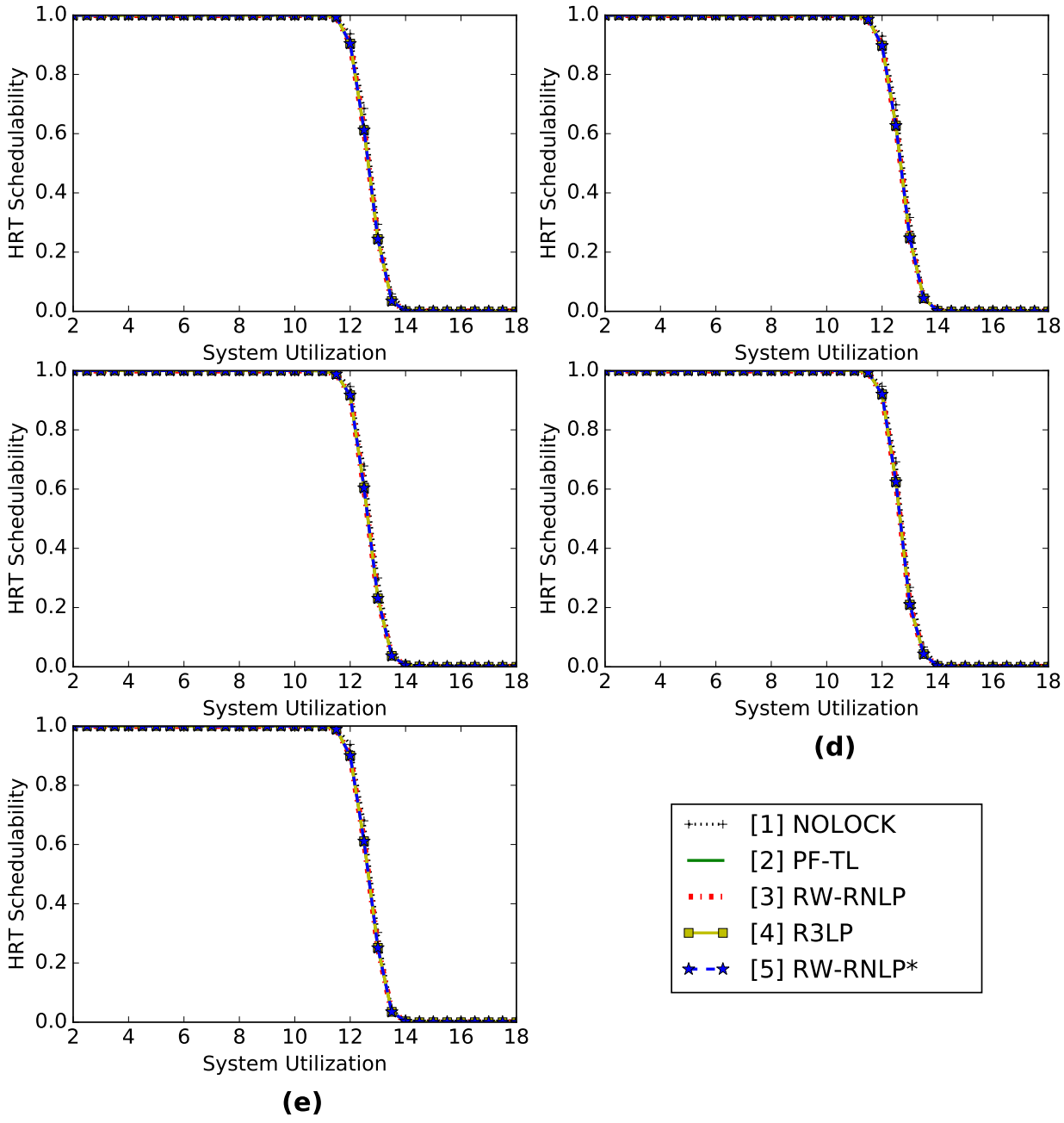



Figure 657: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, short critical-section lengths, and read probability 0 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

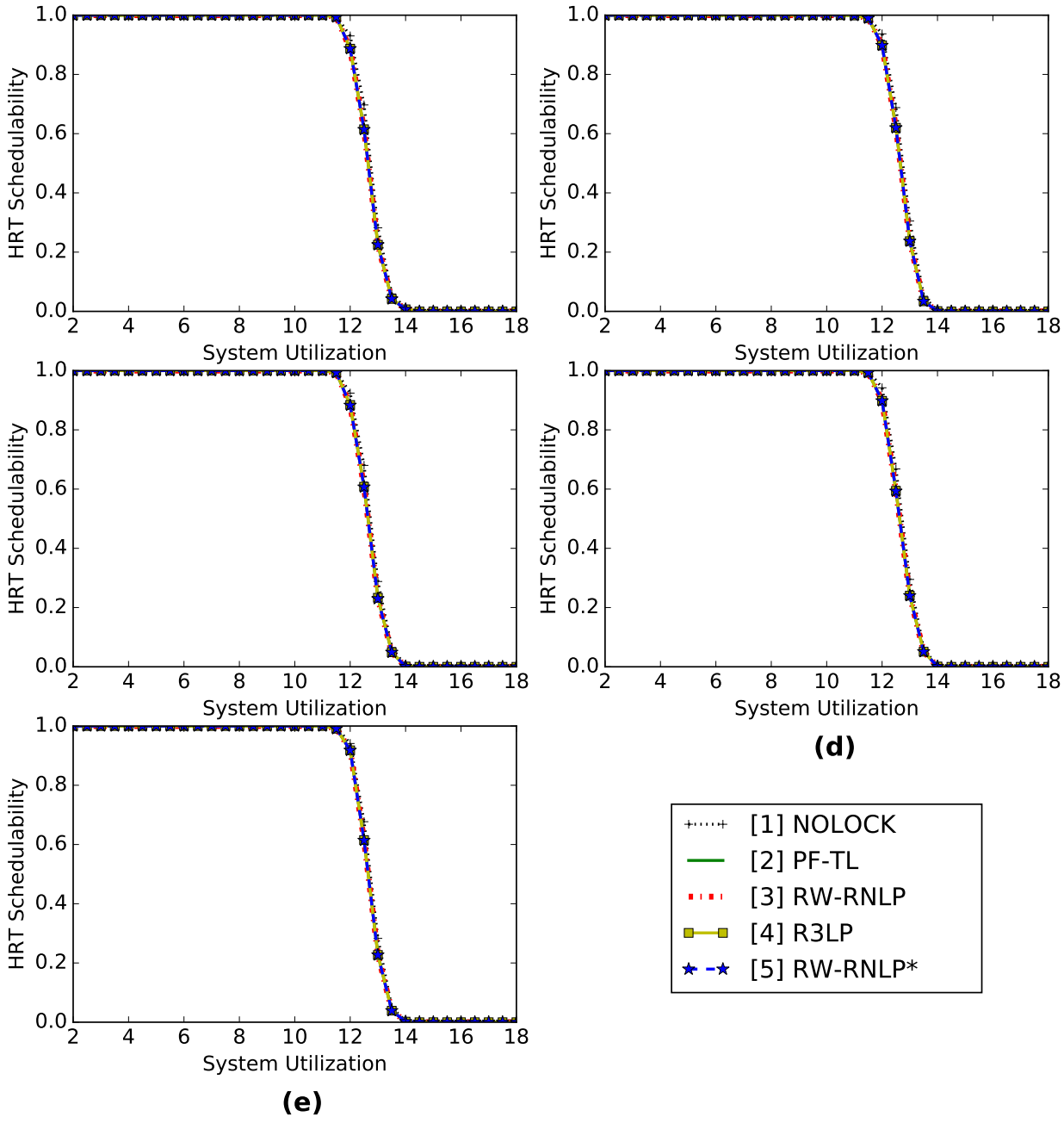



Figure 658: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, short critical-section lengths, and read probability 0 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

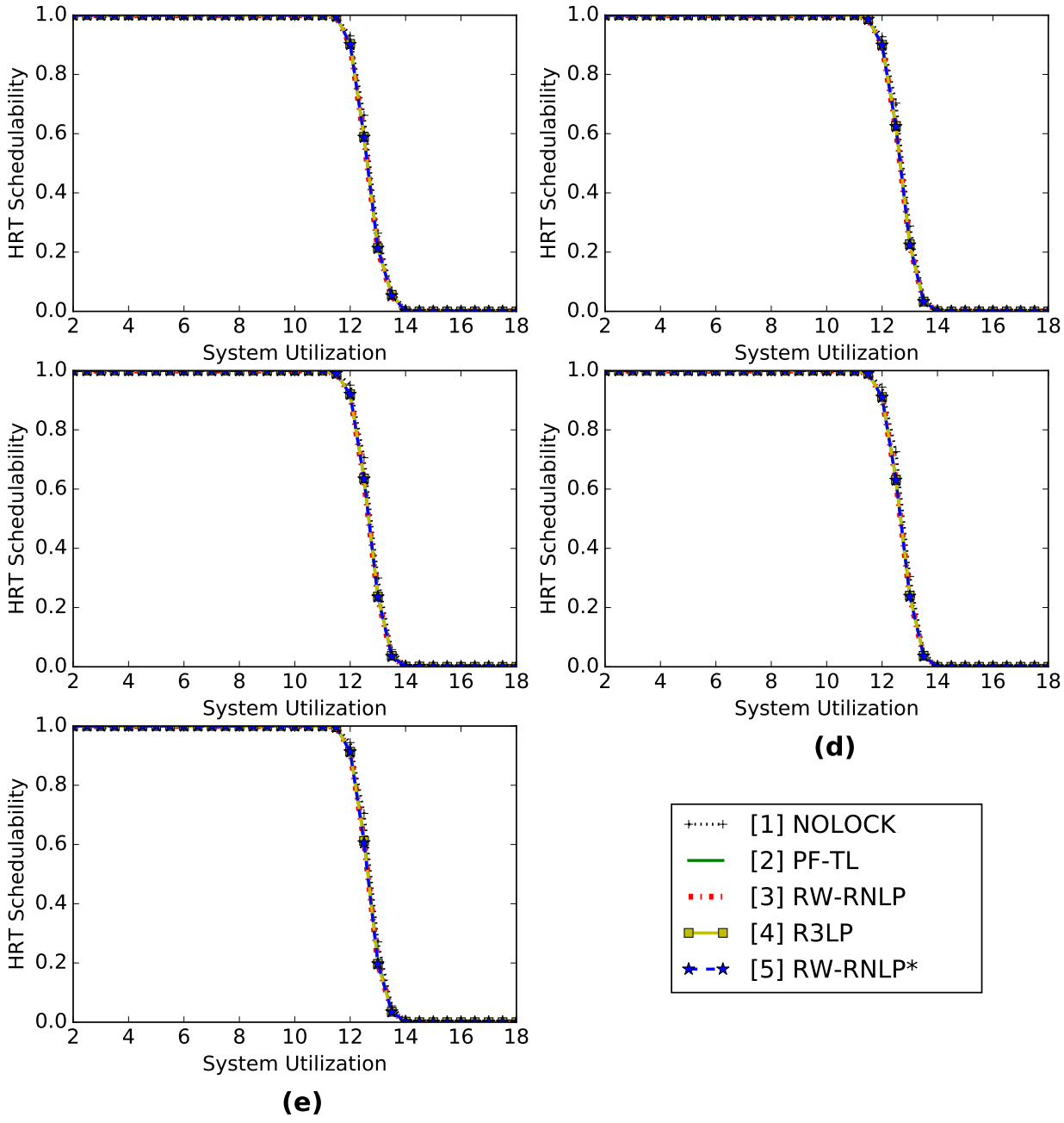



Figure 659: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, short critical-section lengths, and read probability 0 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

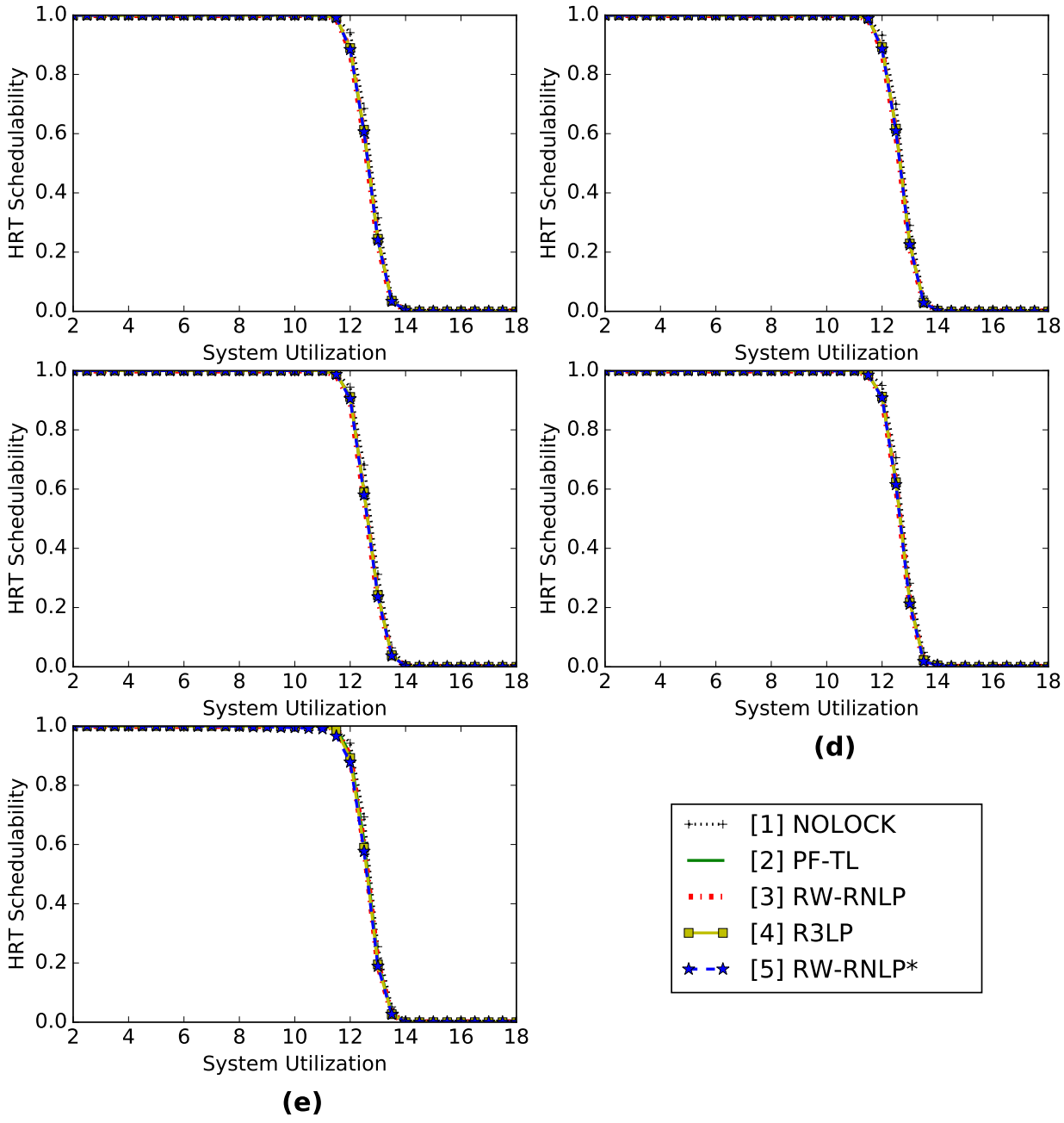



Figure 660: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, short critical-section lengths, and read probability 0 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

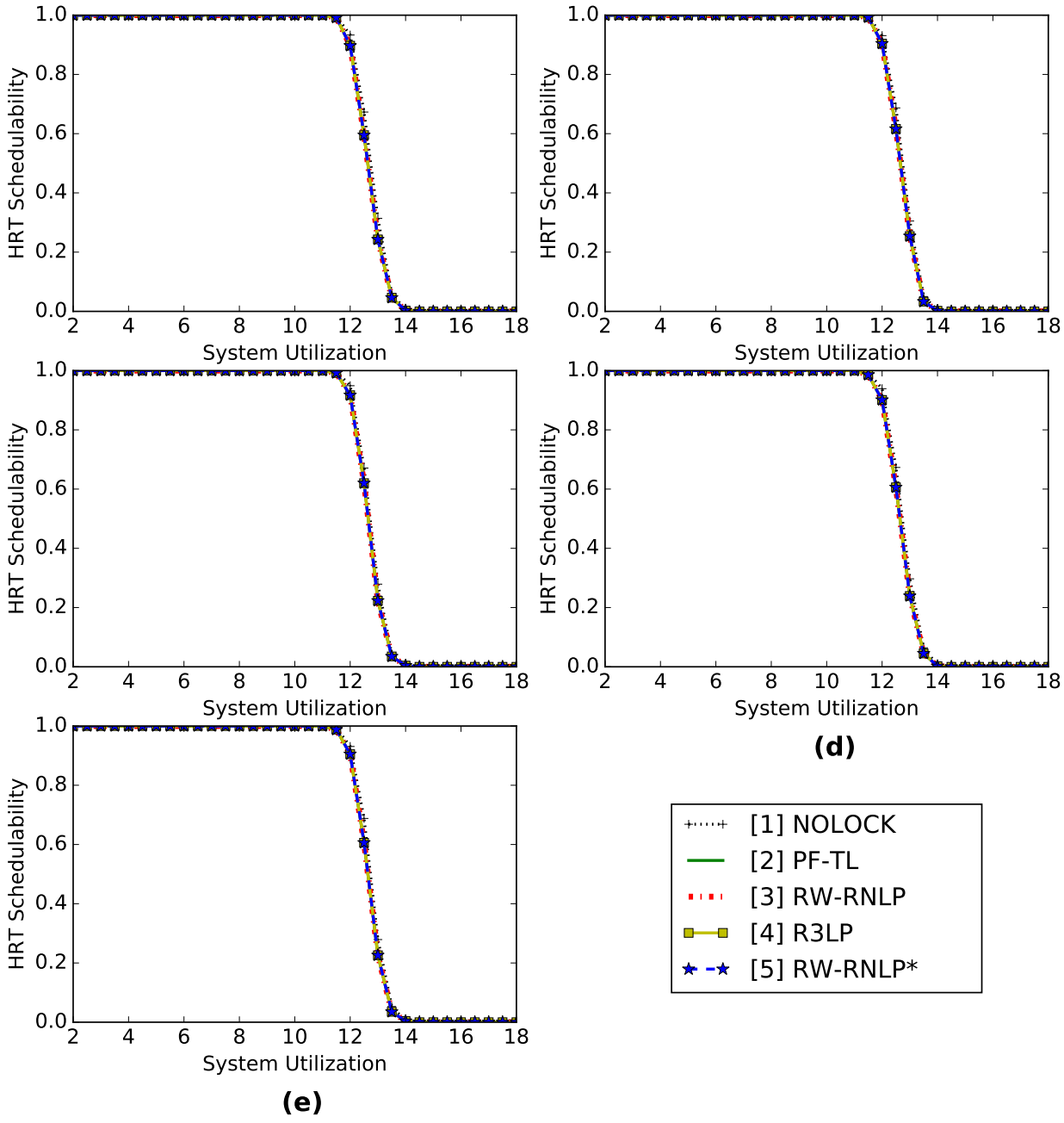



Figure 661: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, short critical-section lengths, and read probability 0.2 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

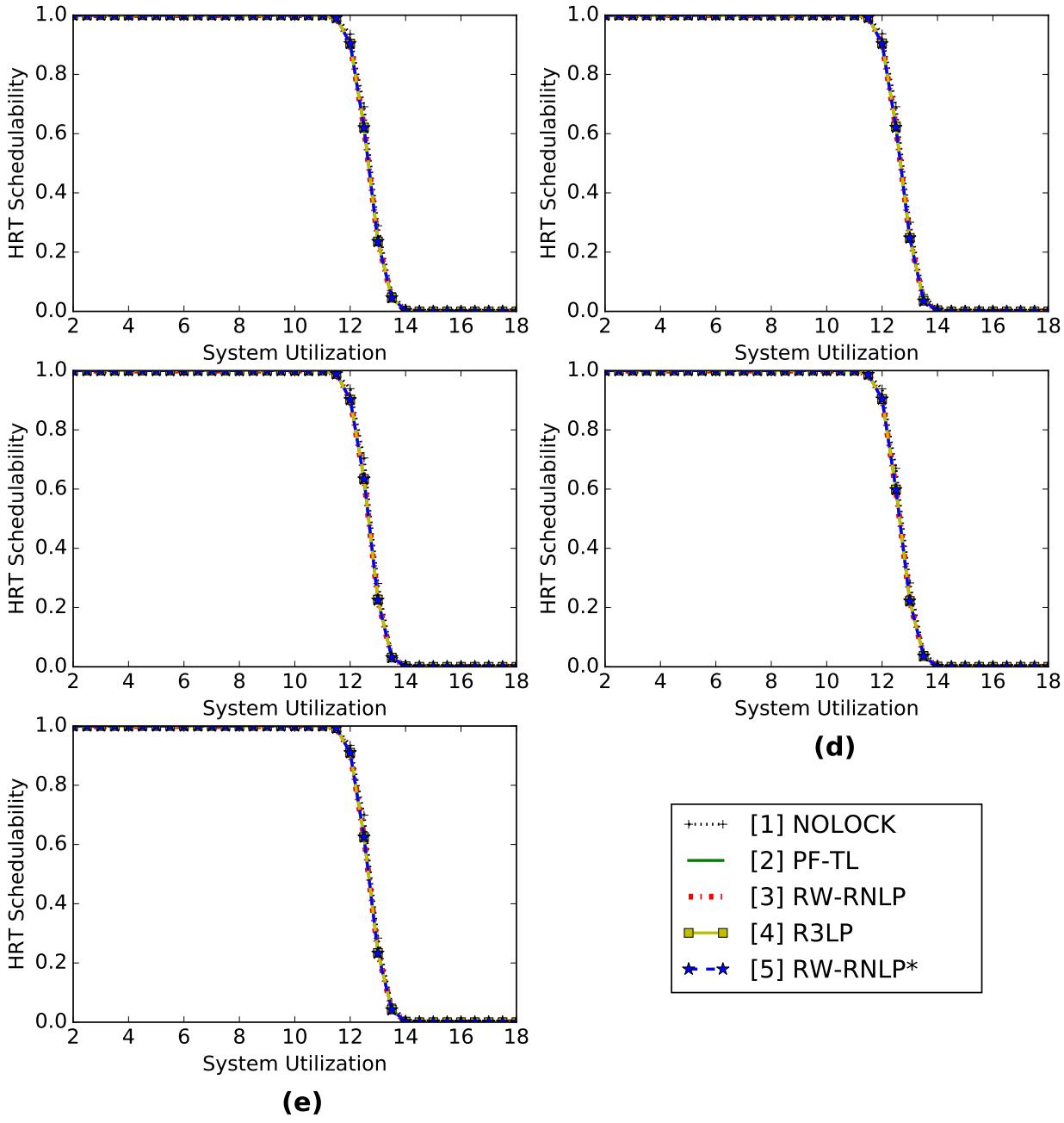



Figure 662: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, short critical-section lengths, and read probability 0.2 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

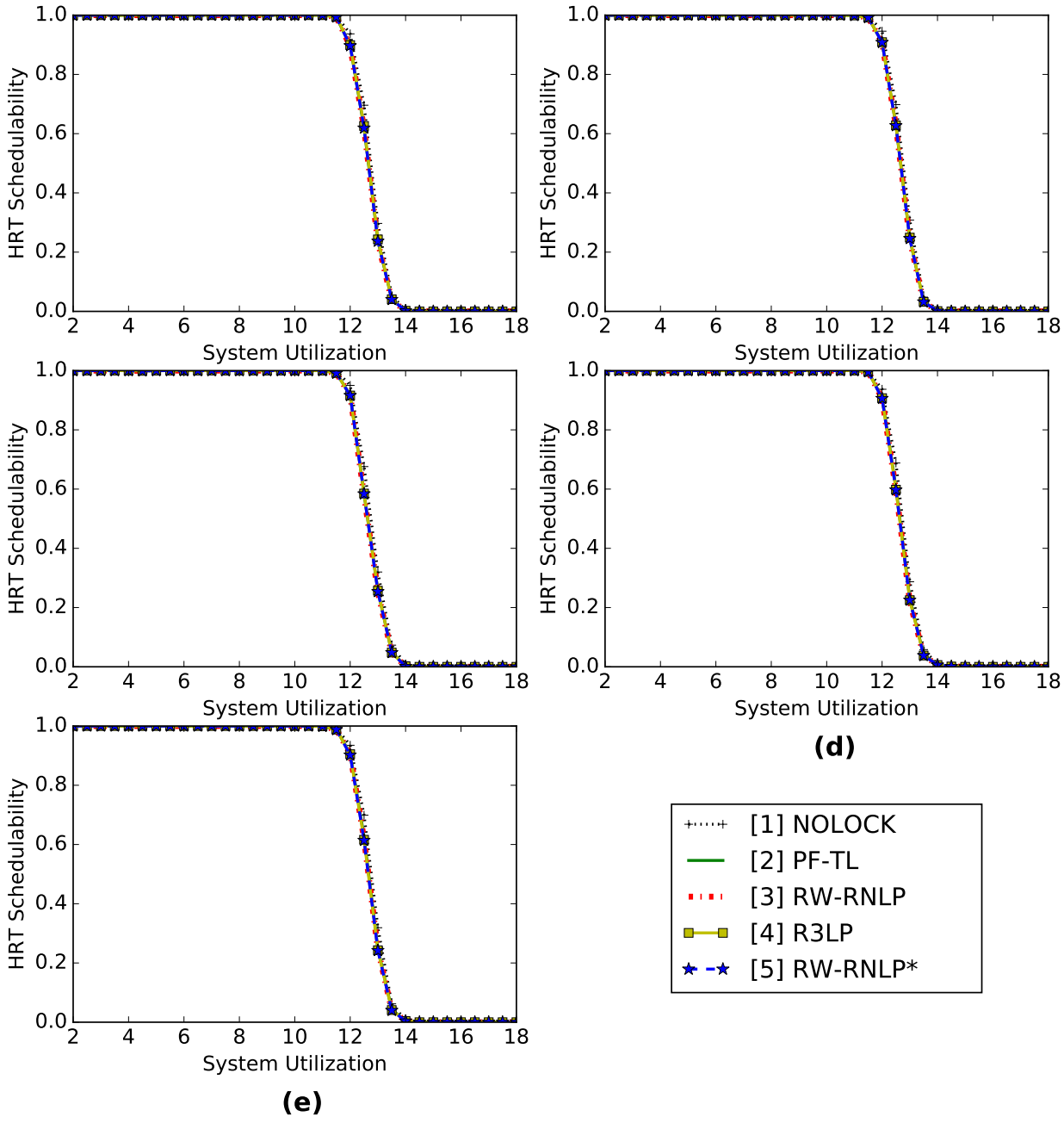



Figure 663: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, short critical-section lengths, and read probability 0.2 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

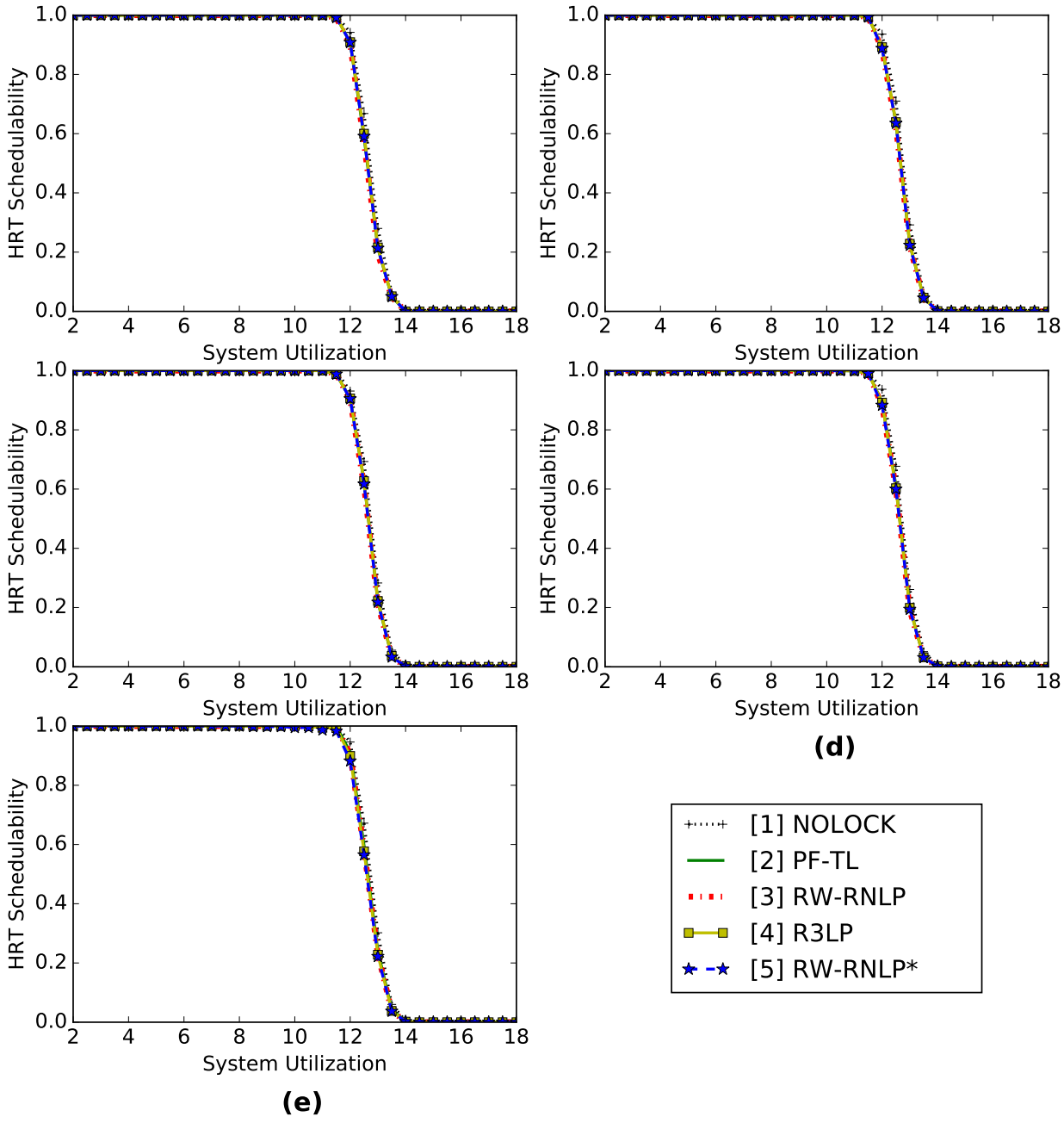



Figure 664: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, short critical-section lengths, and read probability 0.2 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

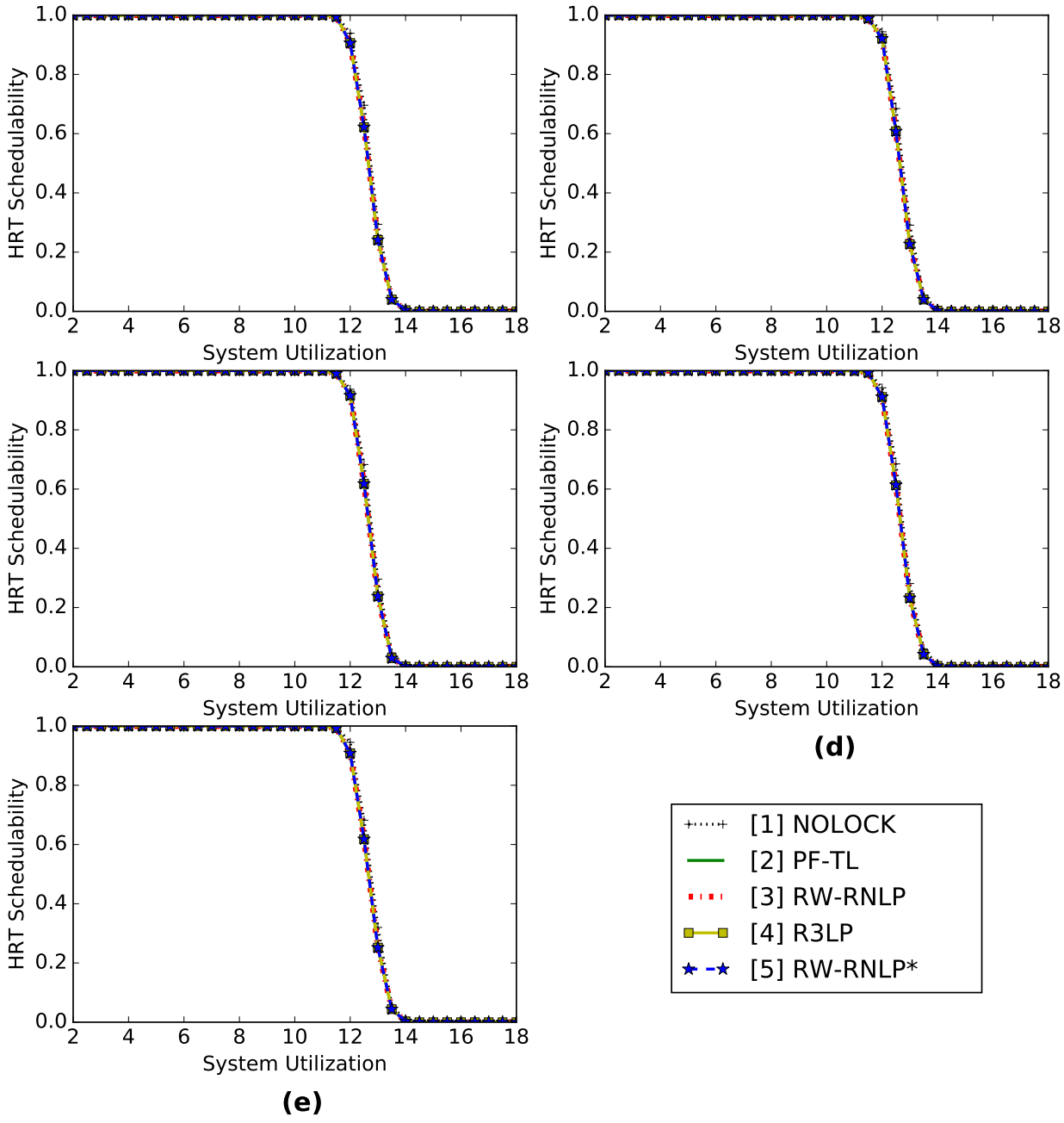



Figure 665: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, short critical-section lengths, and read probability 0.5 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

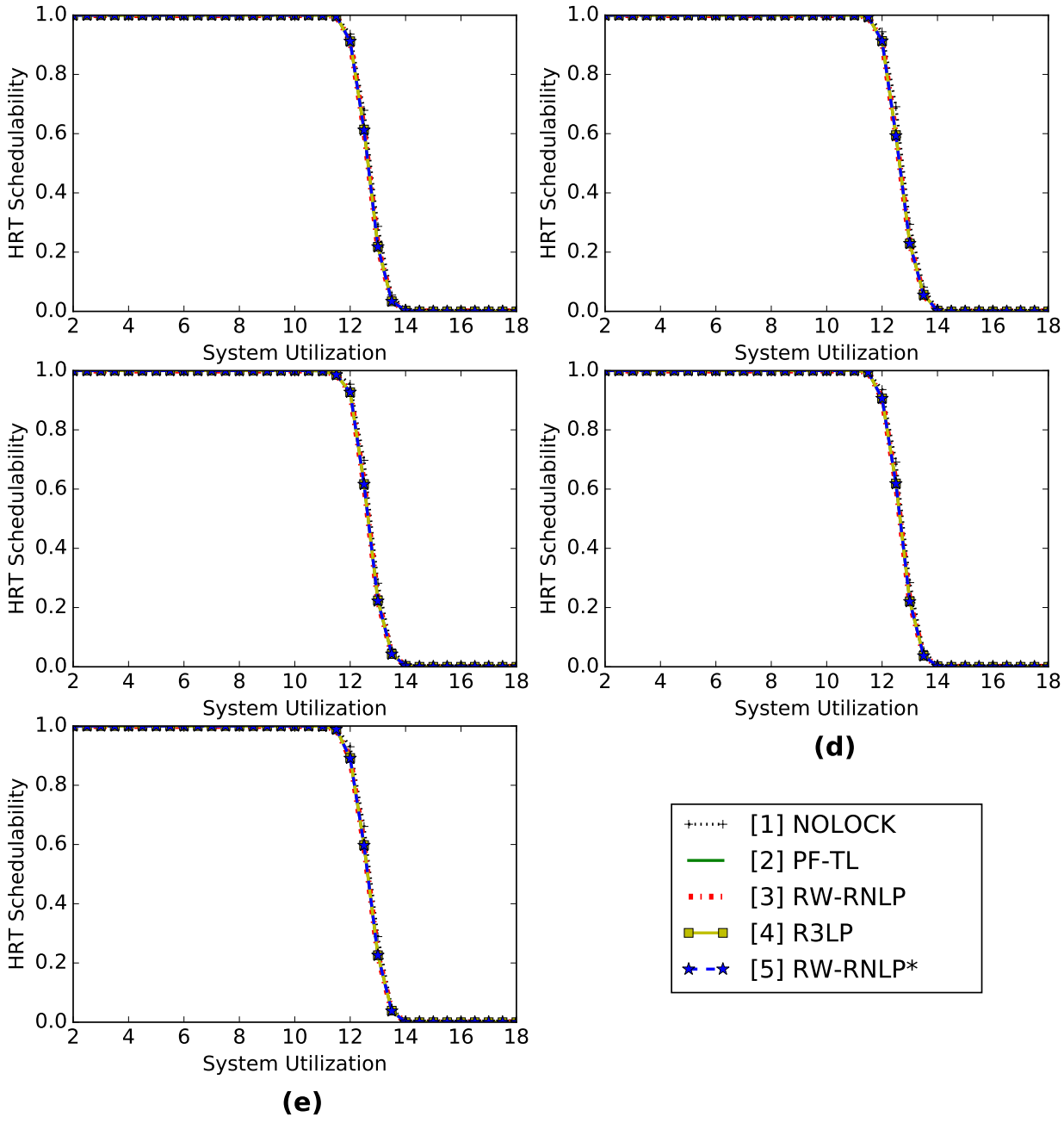



Figure 666: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, short critical-section lengths, and read probability 0.5 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

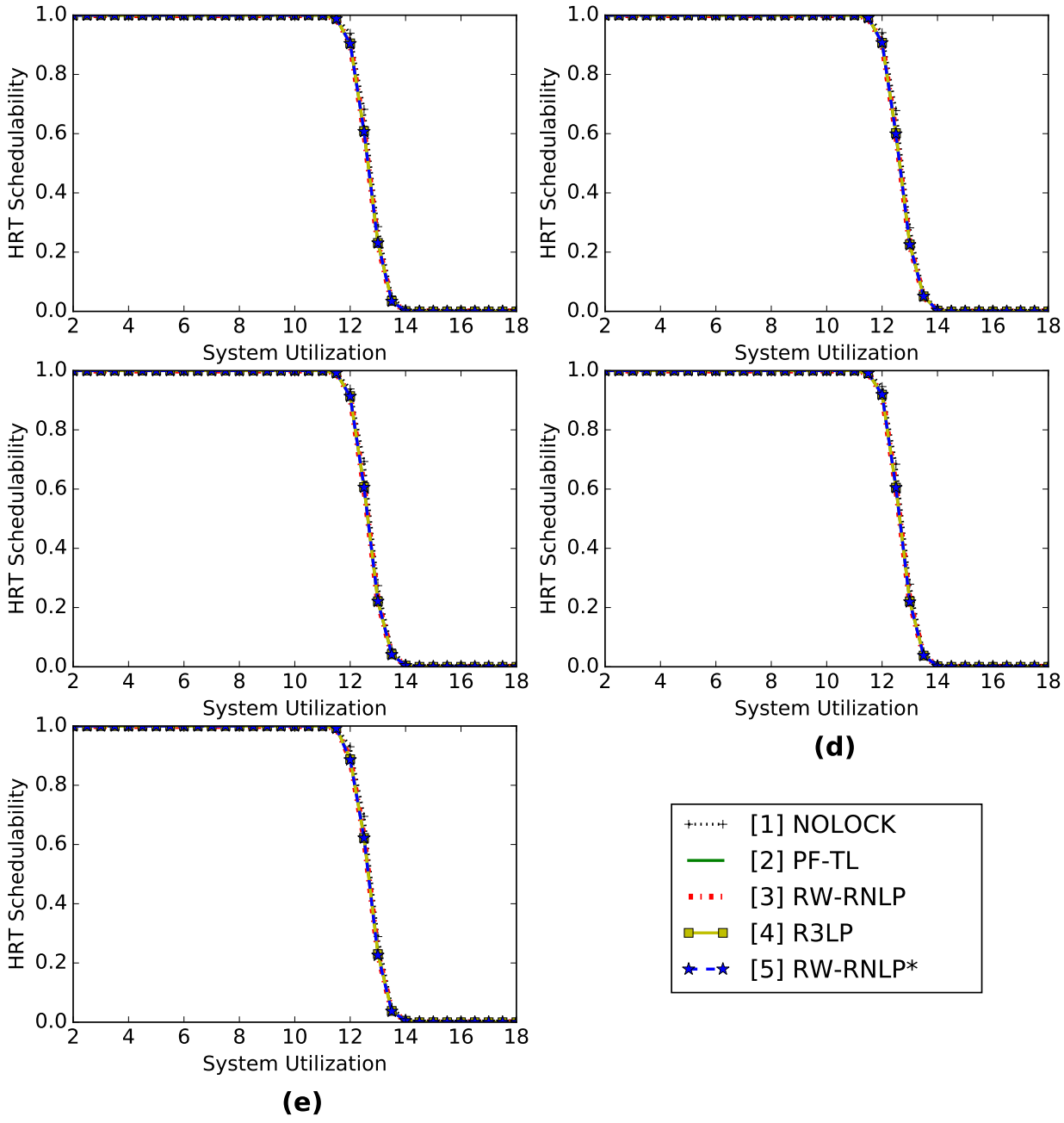



Figure 667: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, short critical-section lengths, and read probability 0.5 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

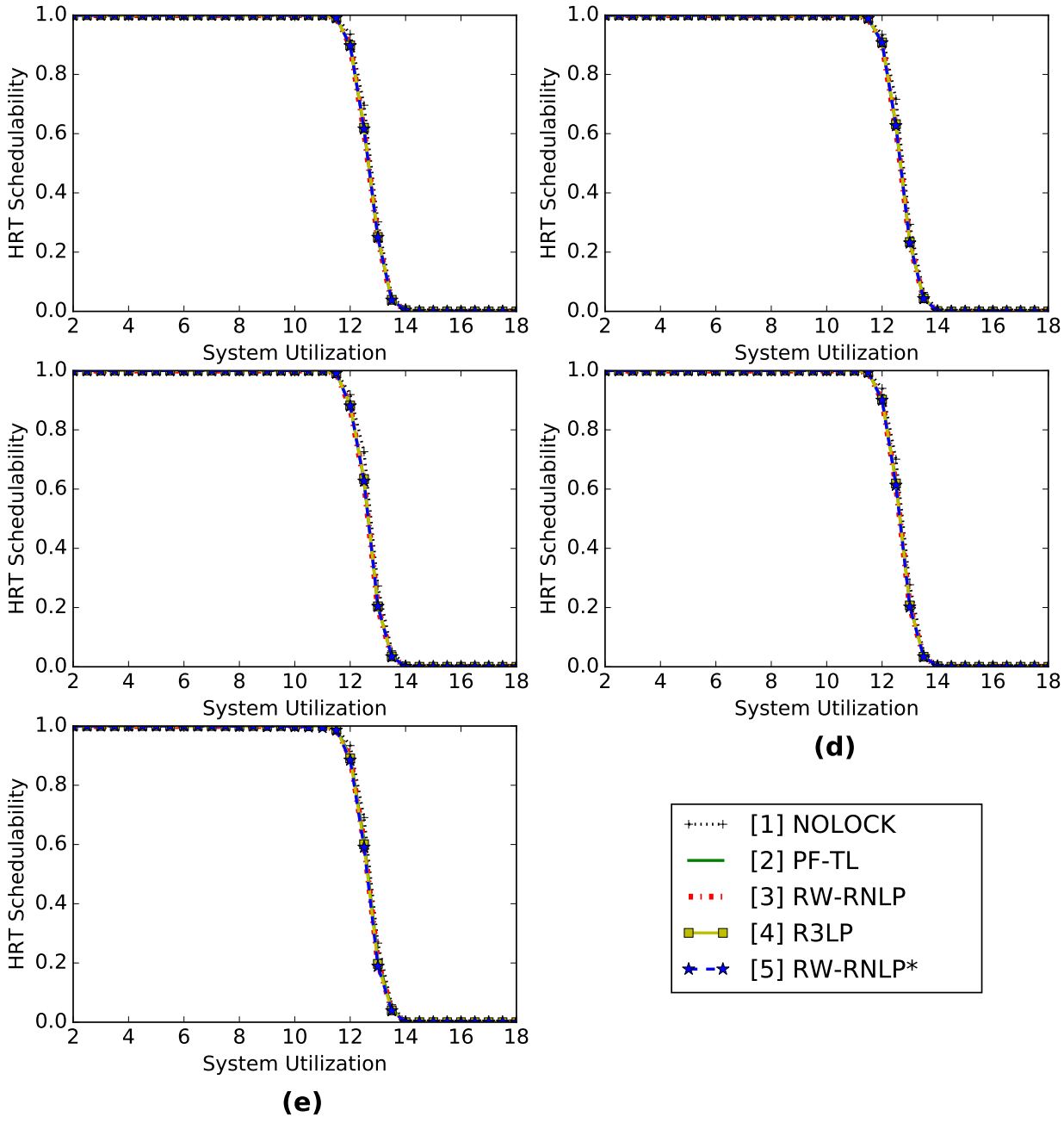



Figure 668: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, short critical-section lengths, and read probability 0.5 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

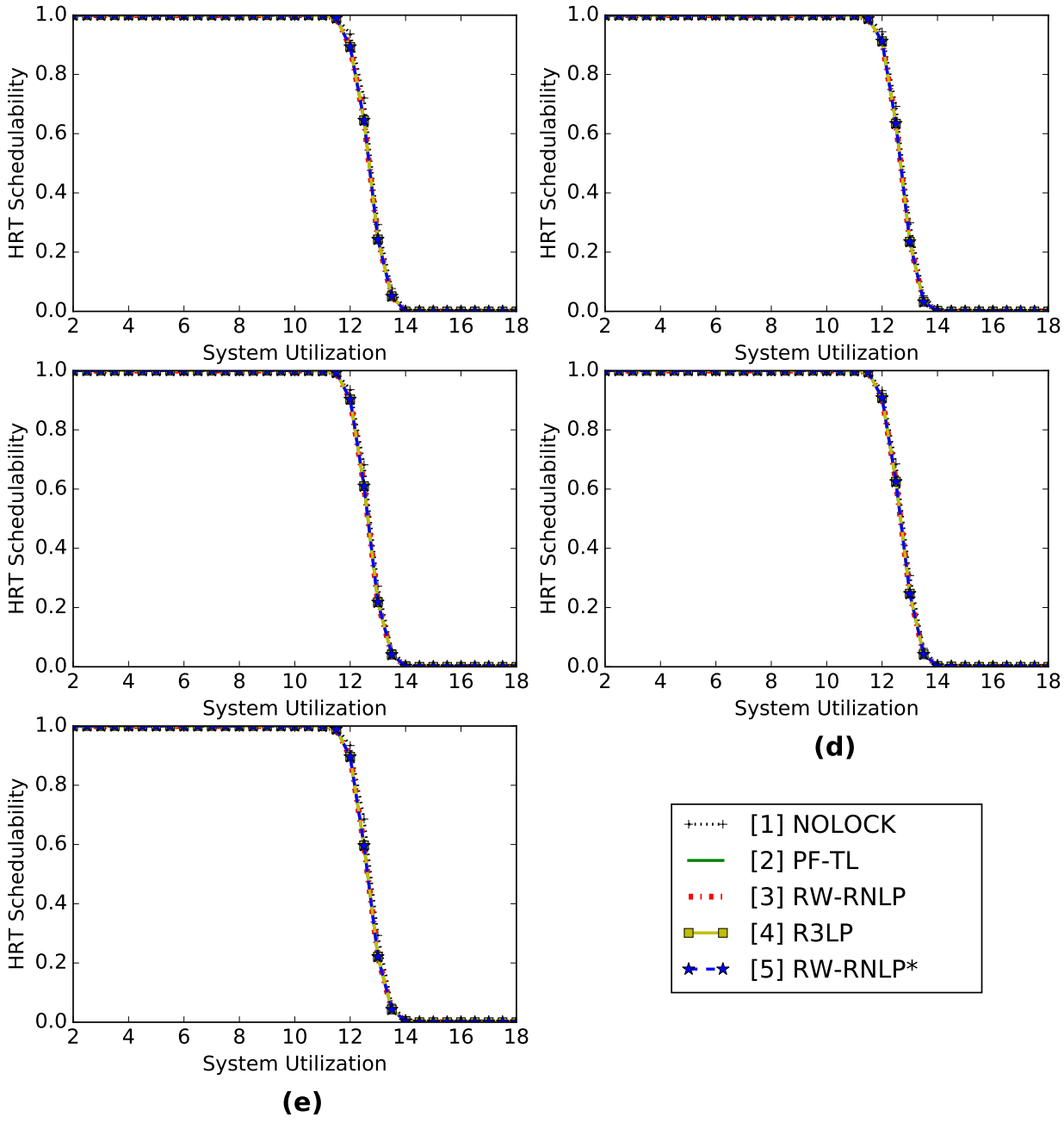



Figure 669: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, short critical-section lengths, and read probability 0.8 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

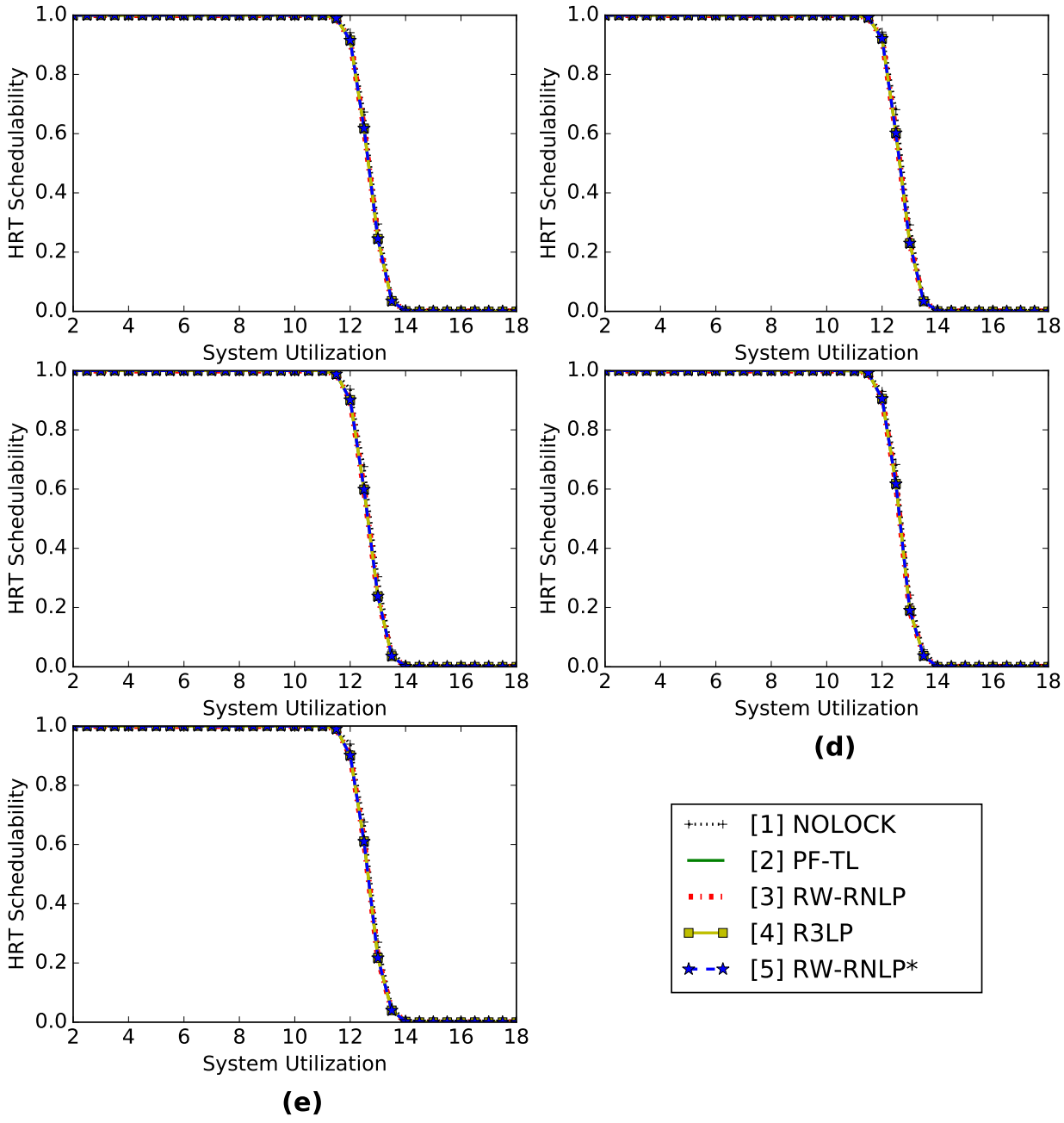



Figure 670: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, short critical-section lengths, and read probability 0.8 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

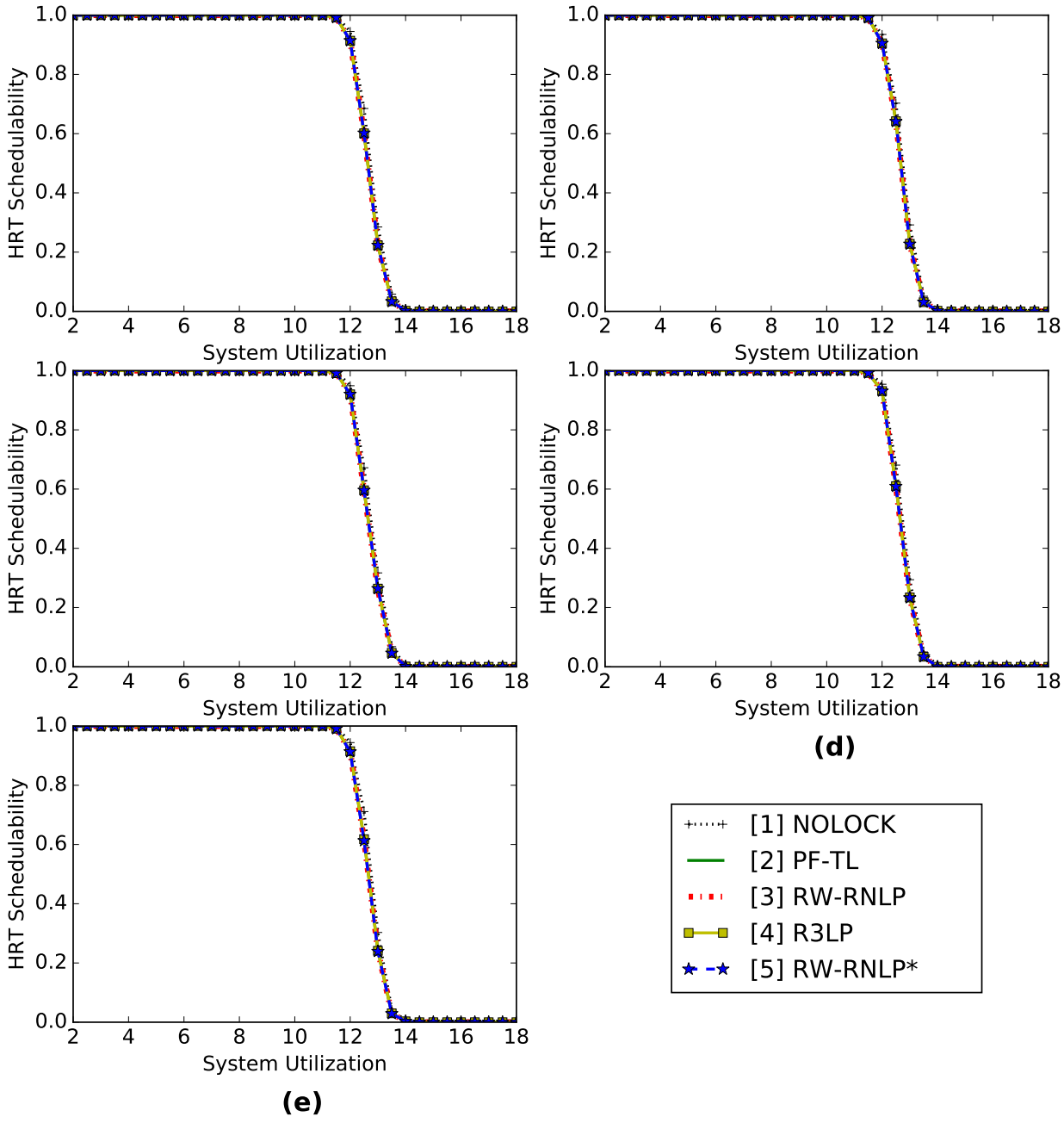



Figure 671: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, short critical-section lengths, and read probability 0.8 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

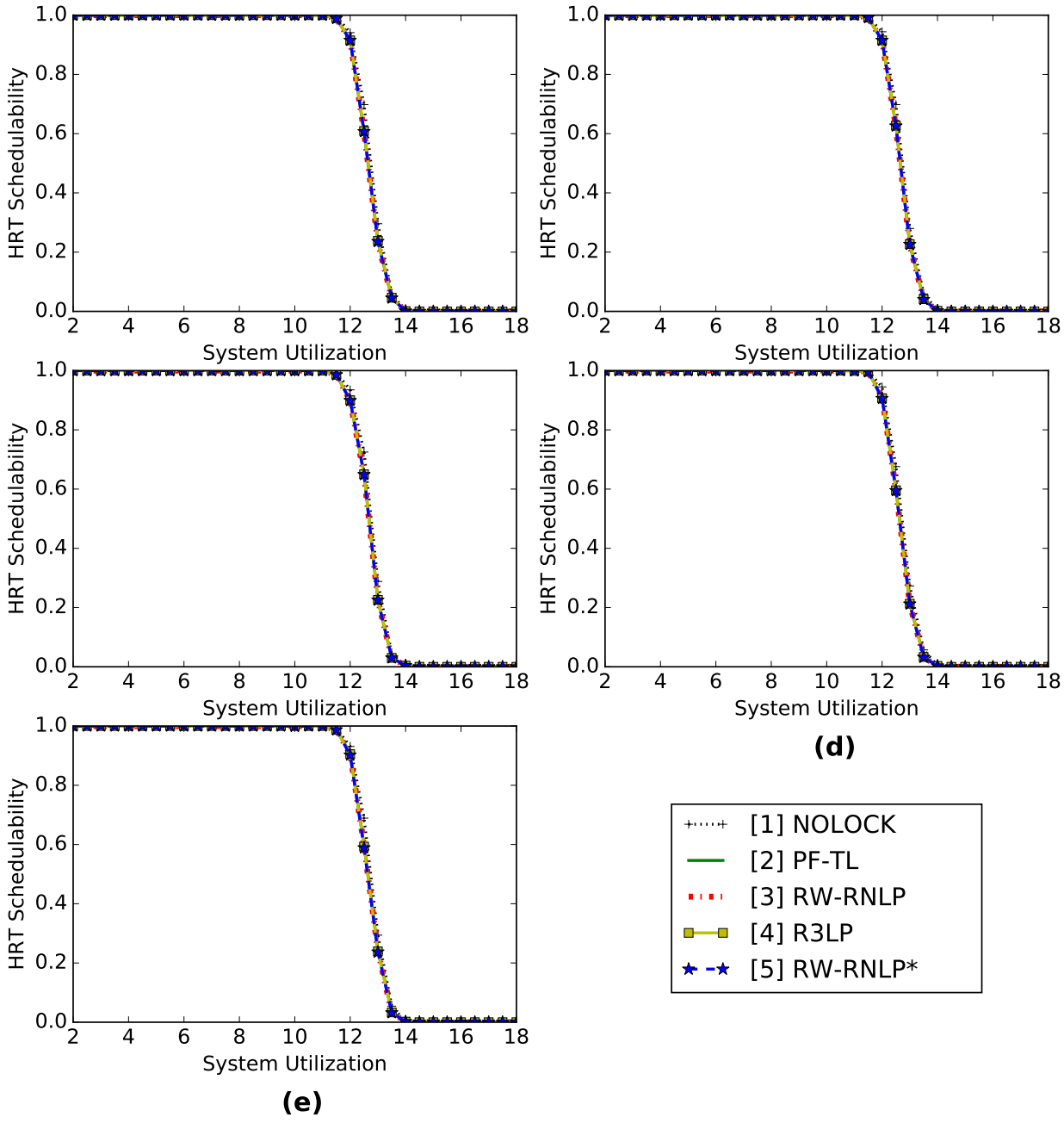



Figure 672: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, short critical-section lengths, and read probability 0.8 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

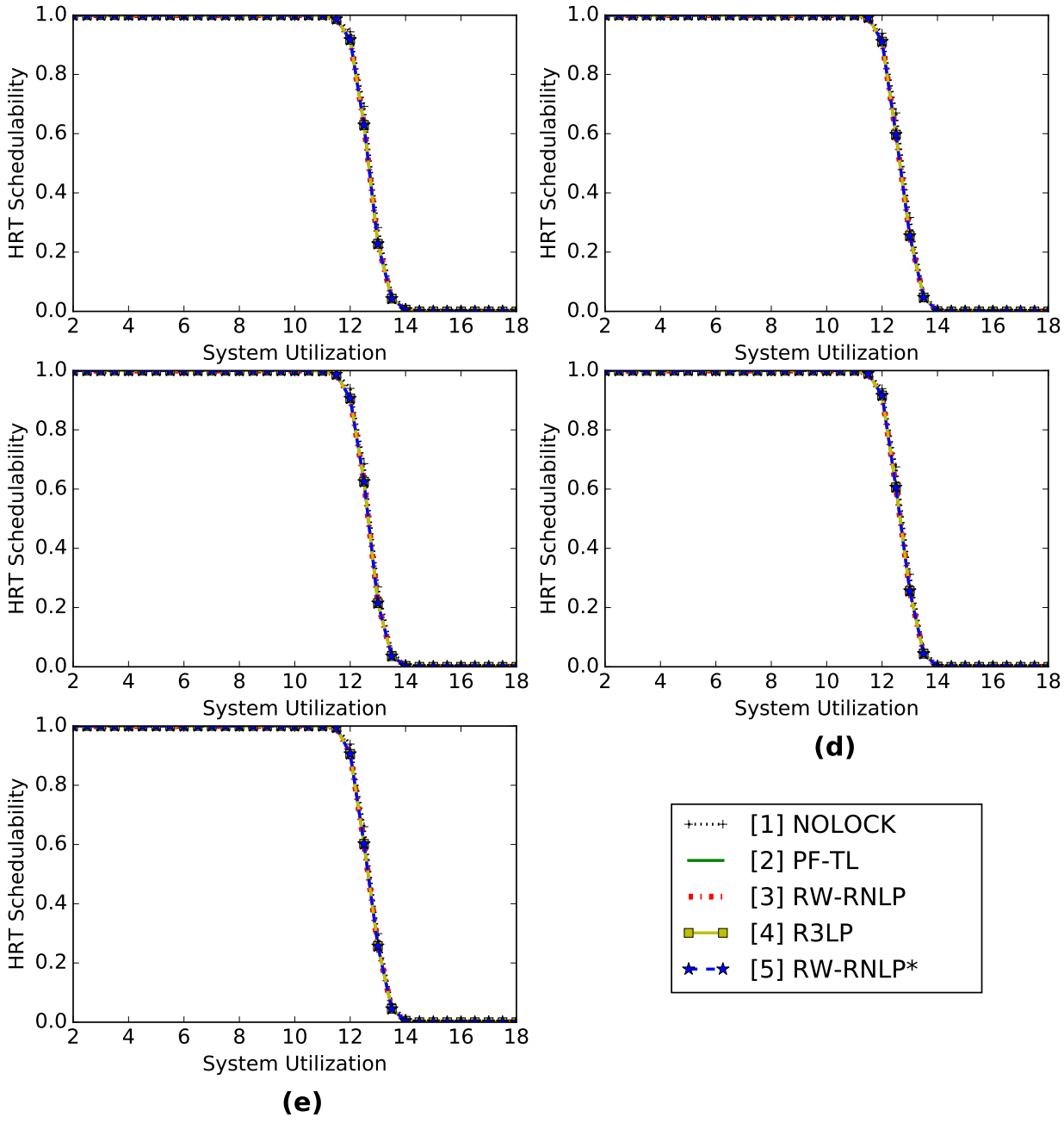



Figure 673: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, short critical-section lengths, and read probability 0 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

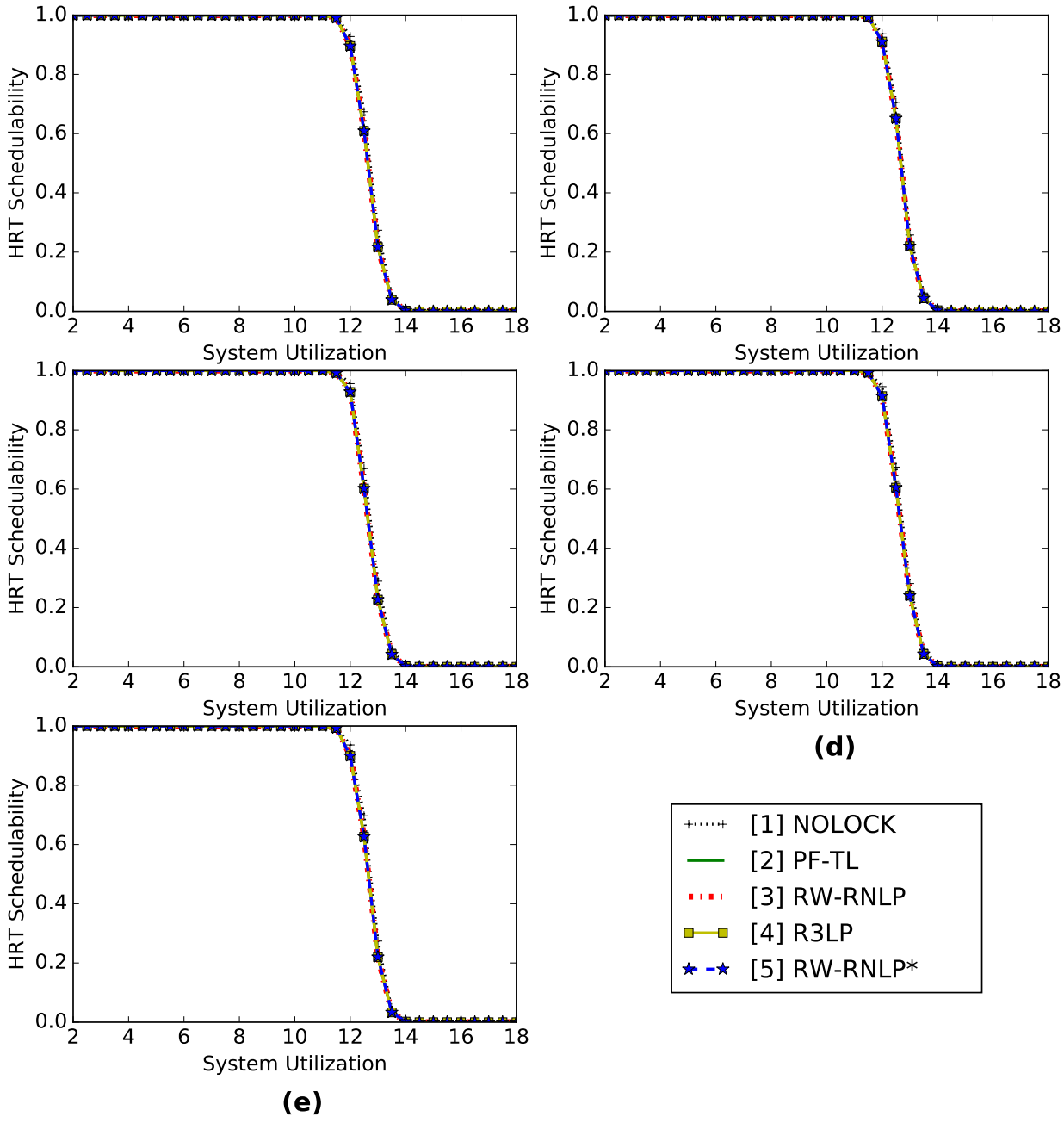



Figure 674: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, short critical-section lengths, and read probability 0 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

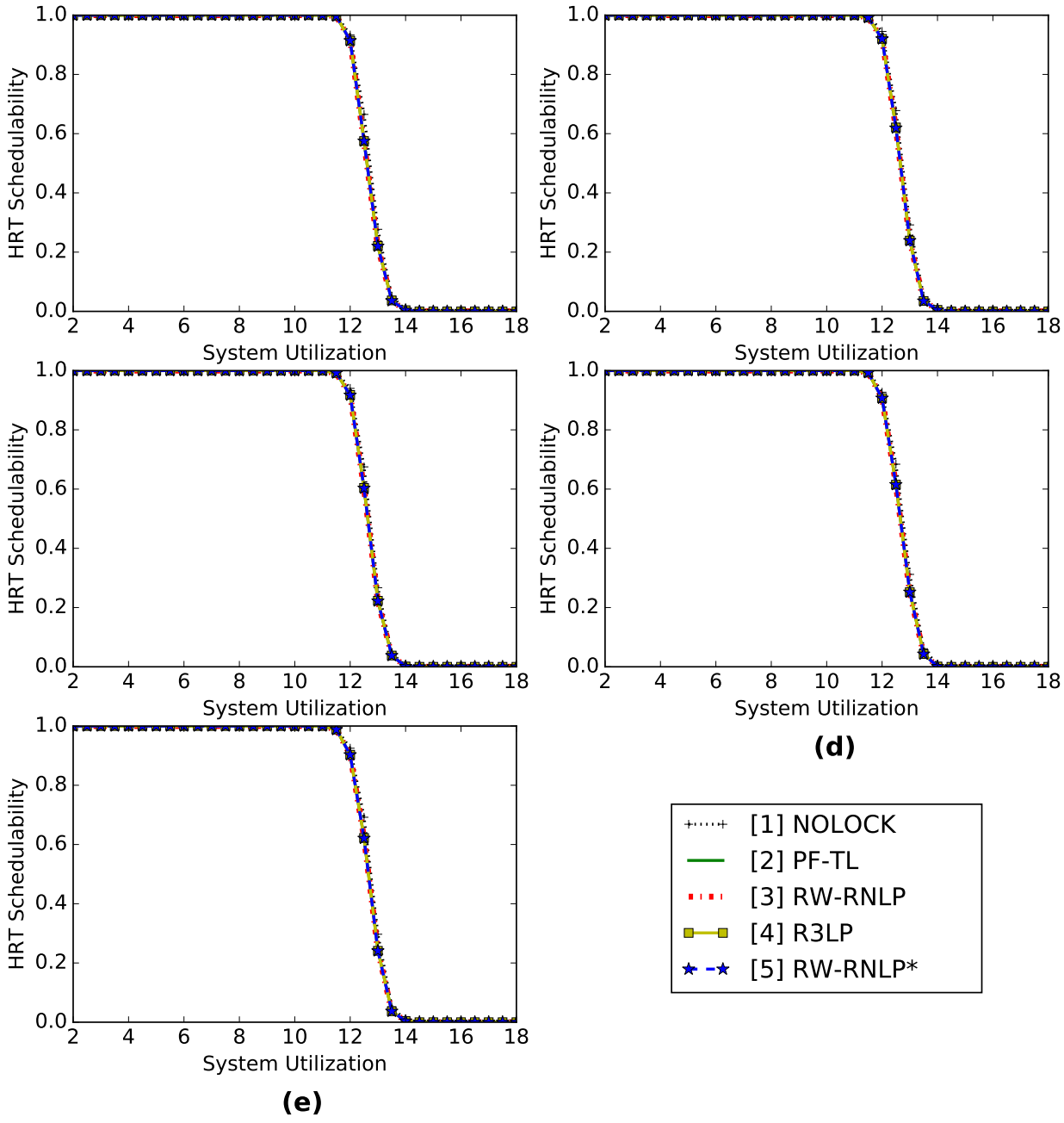



Figure 675: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, short critical-section lengths, and read probability 0 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

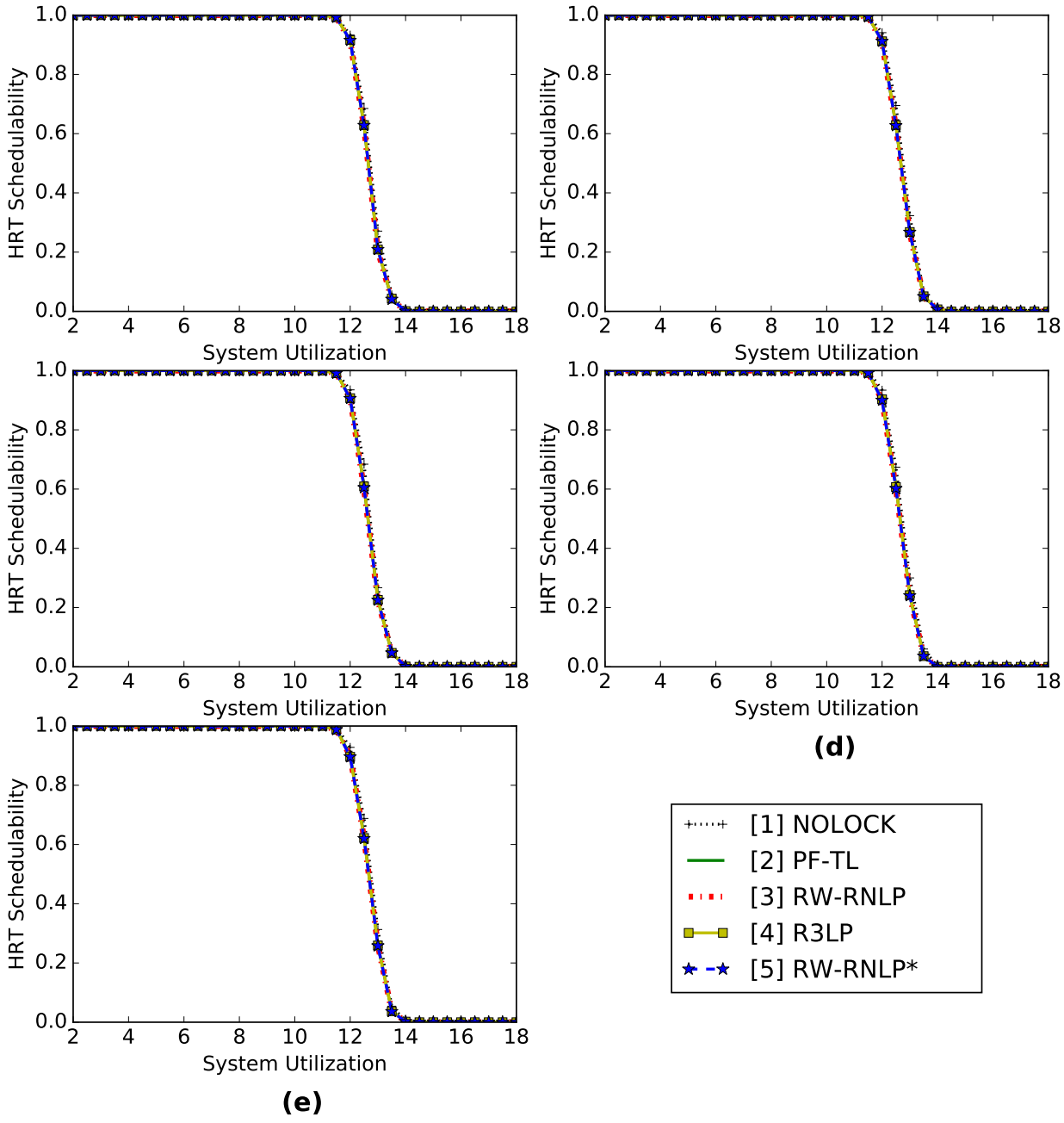



Figure 676: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, short critical-section lengths, and read probability 0 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

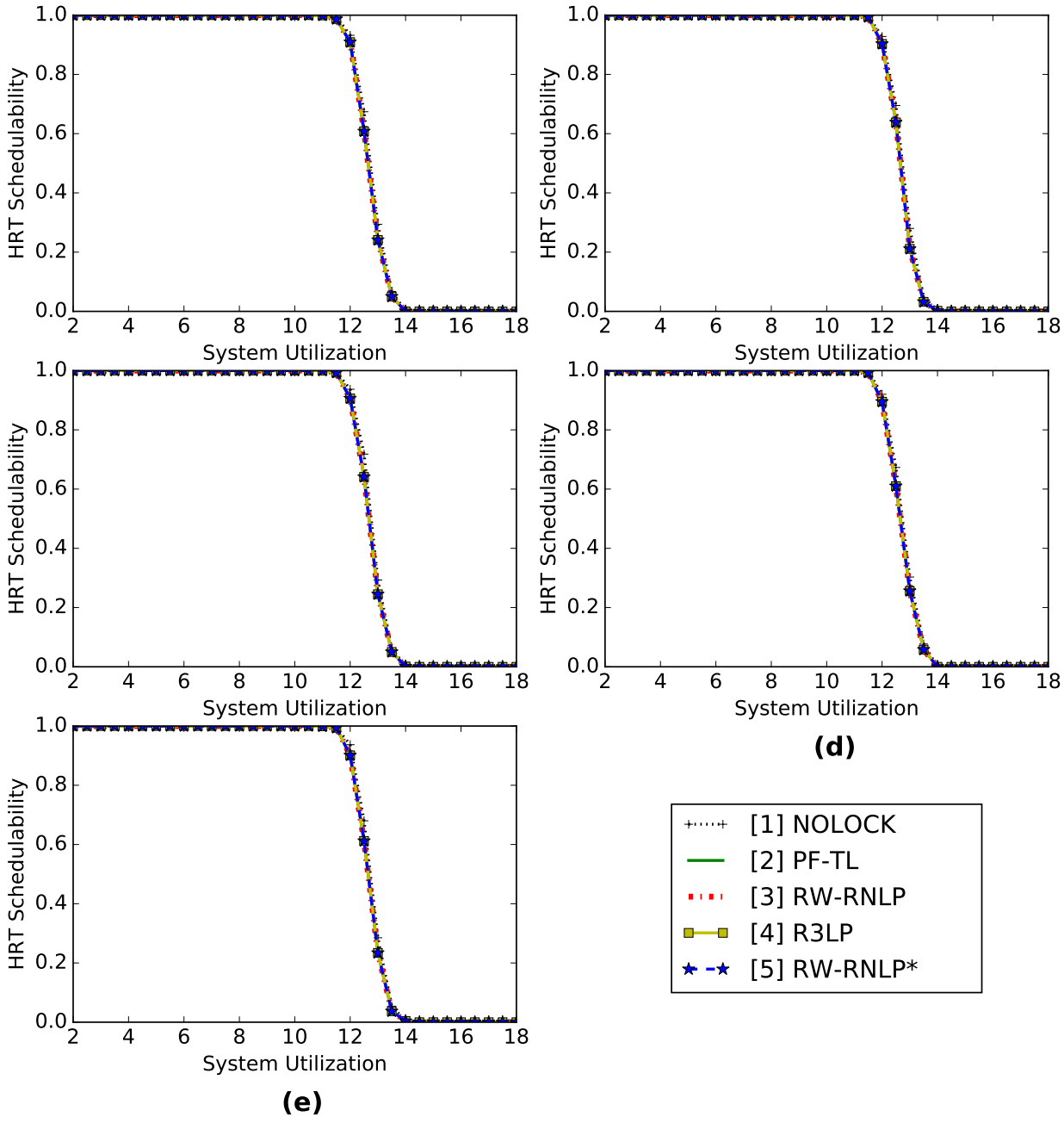



Figure 677: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, short critical-section lengths, and read probability 0.2 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

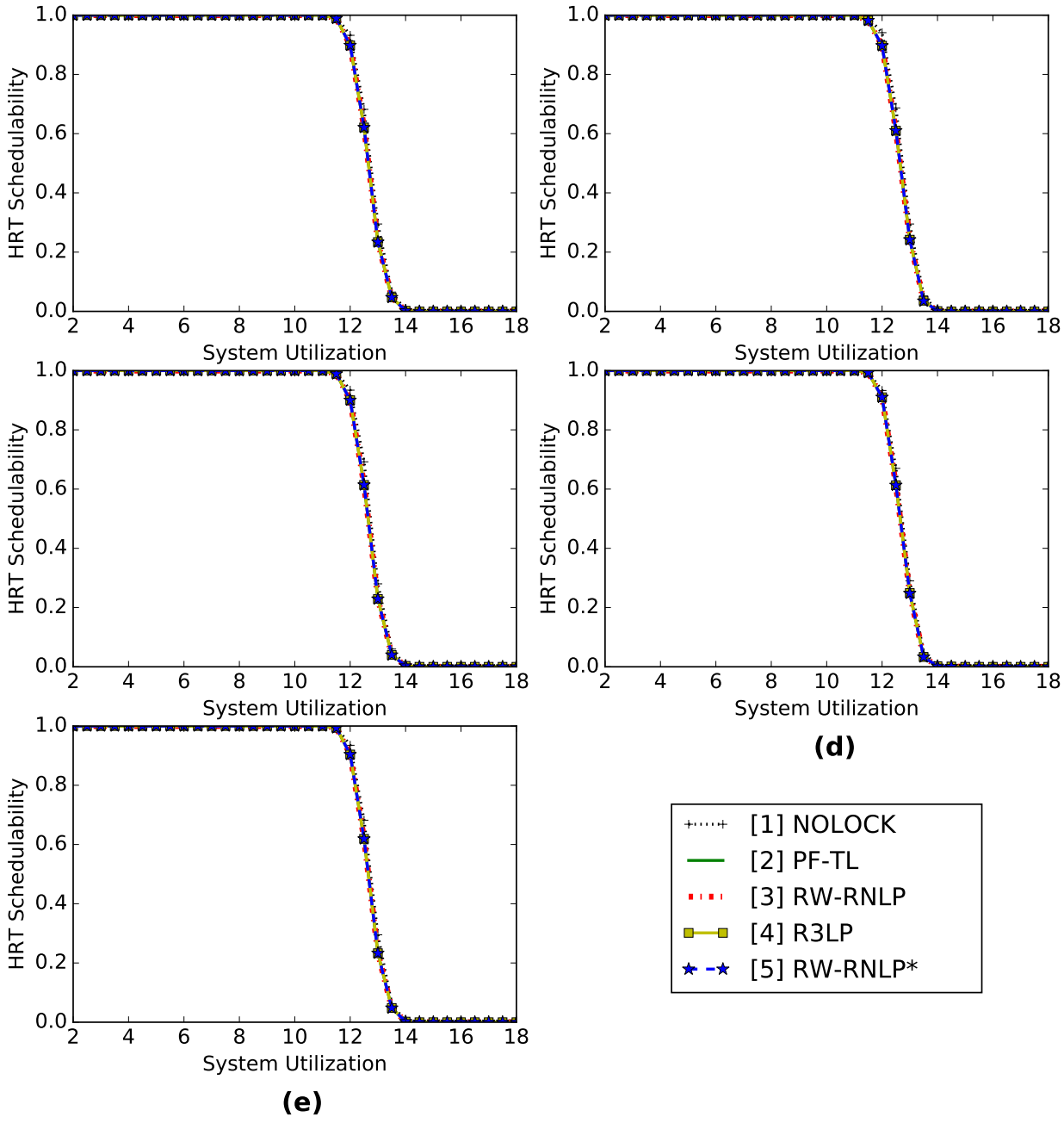



Figure 678: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, short critical-section lengths, and read probability 0.2 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

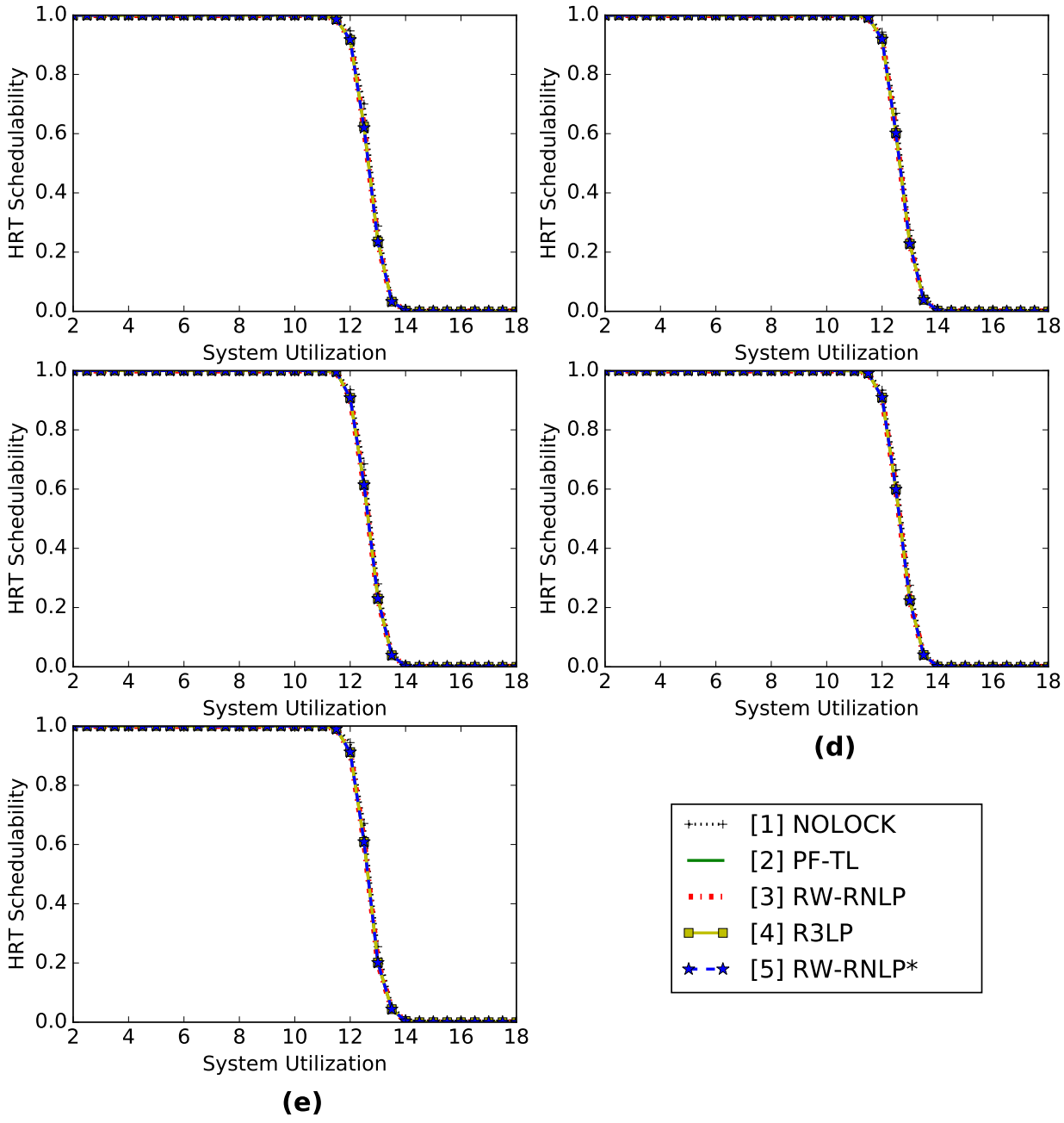



Figure 679: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, short critical-section lengths, and read probability 0.2 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

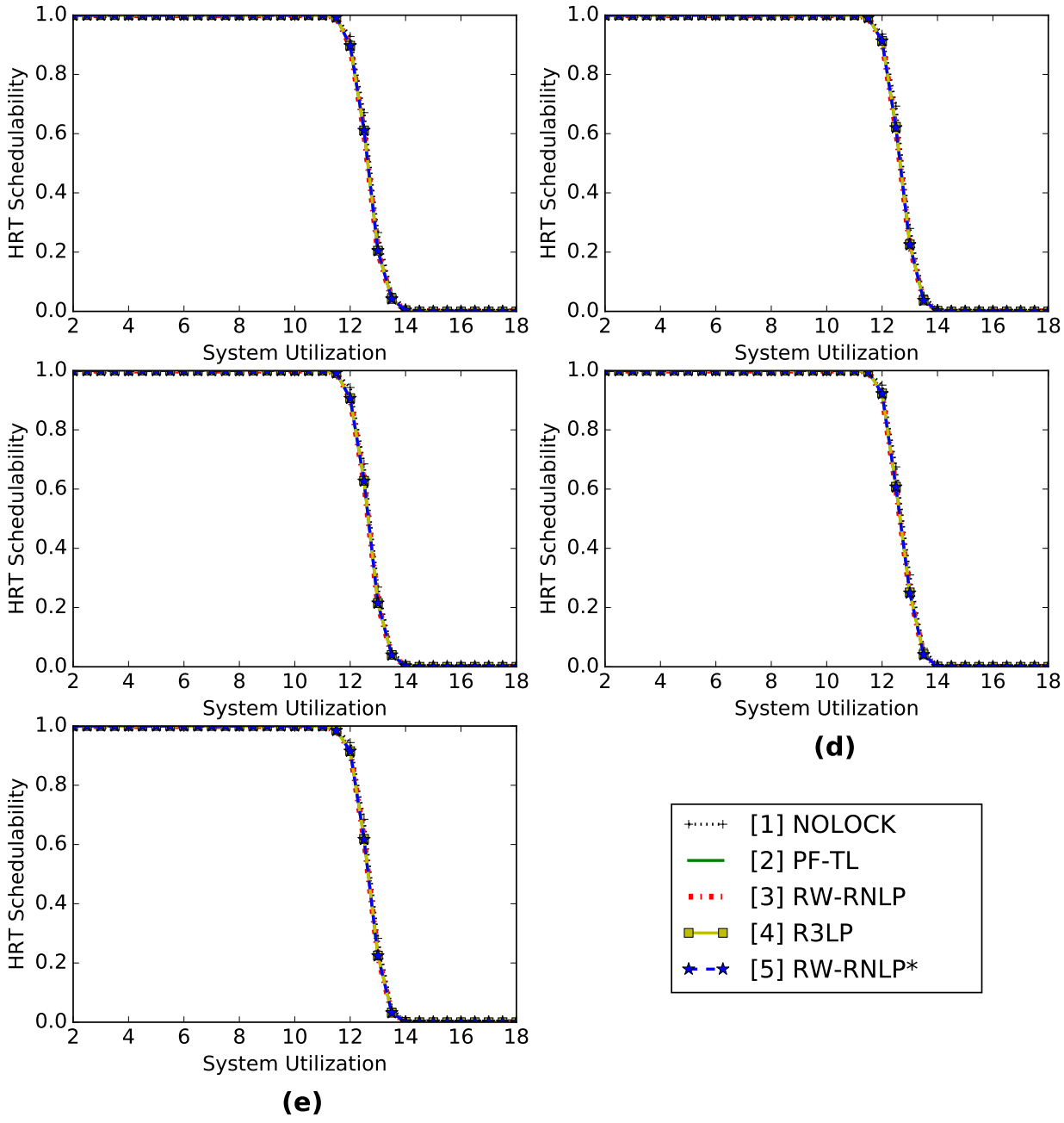



Figure 680: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, short critical-section lengths, and read probability 0.2 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

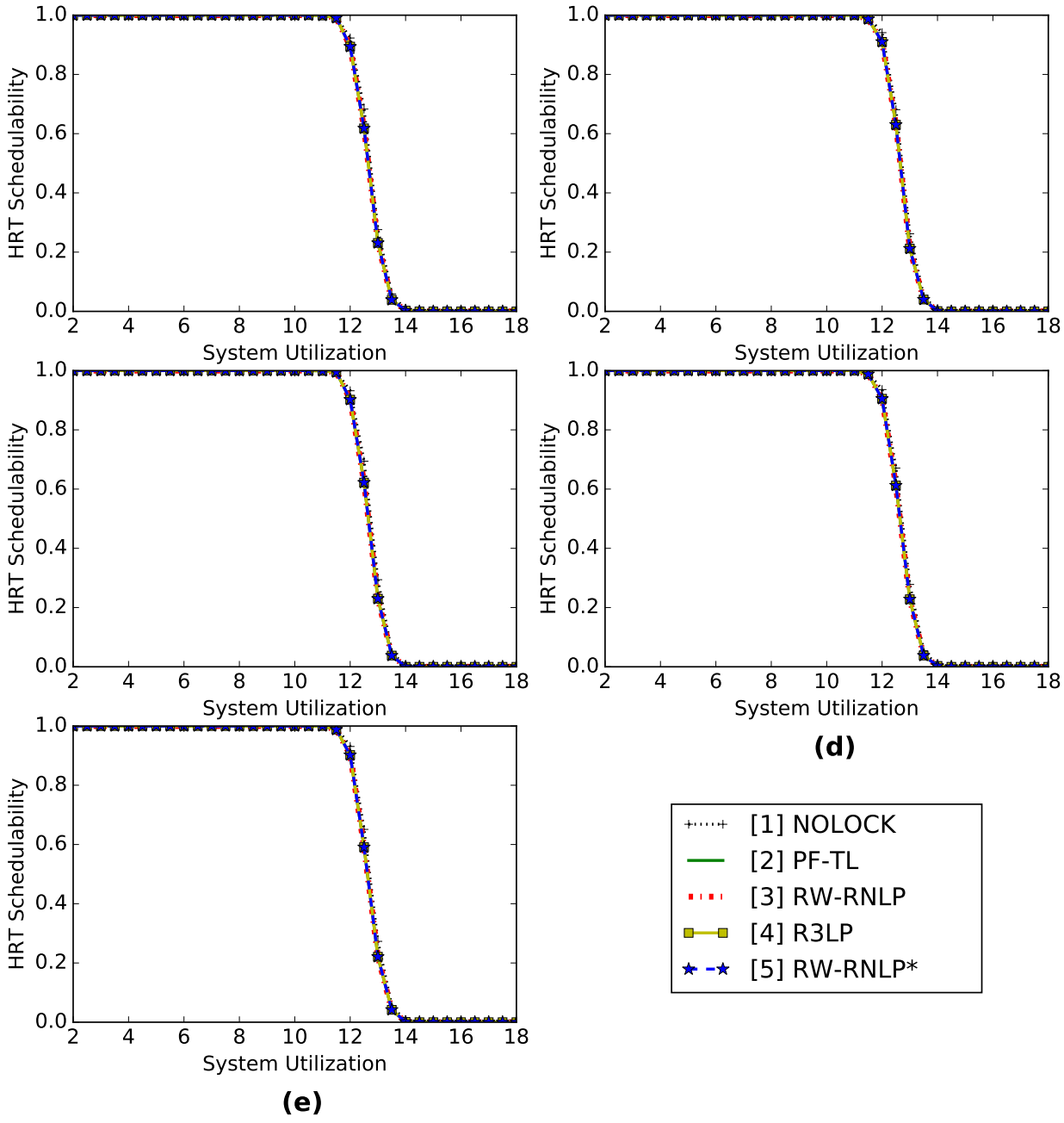



Figure 681: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, short critical-section lengths, and read probability 0.5 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

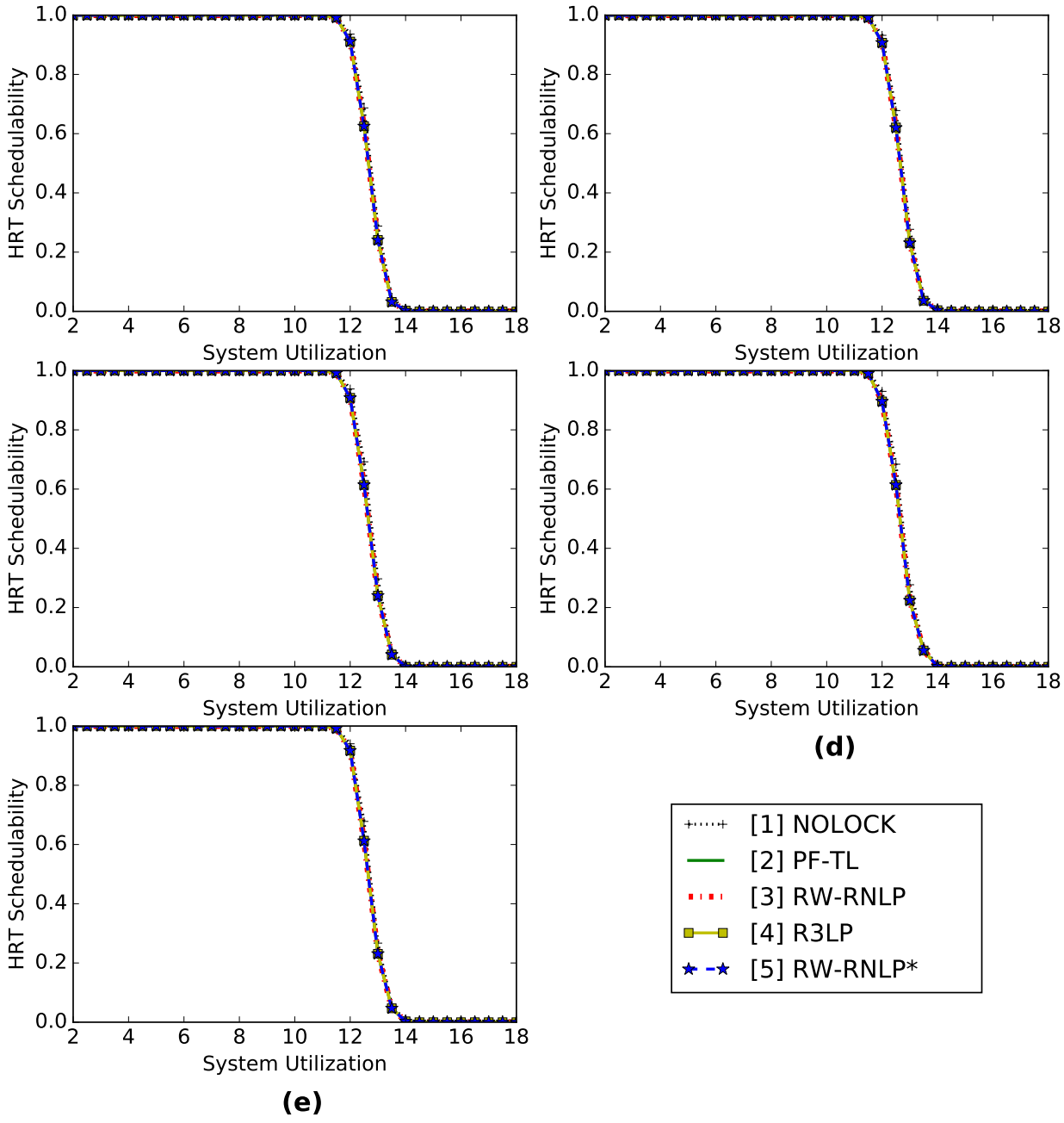



Figure 682: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, short critical-section lengths, and read probability 0.5 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

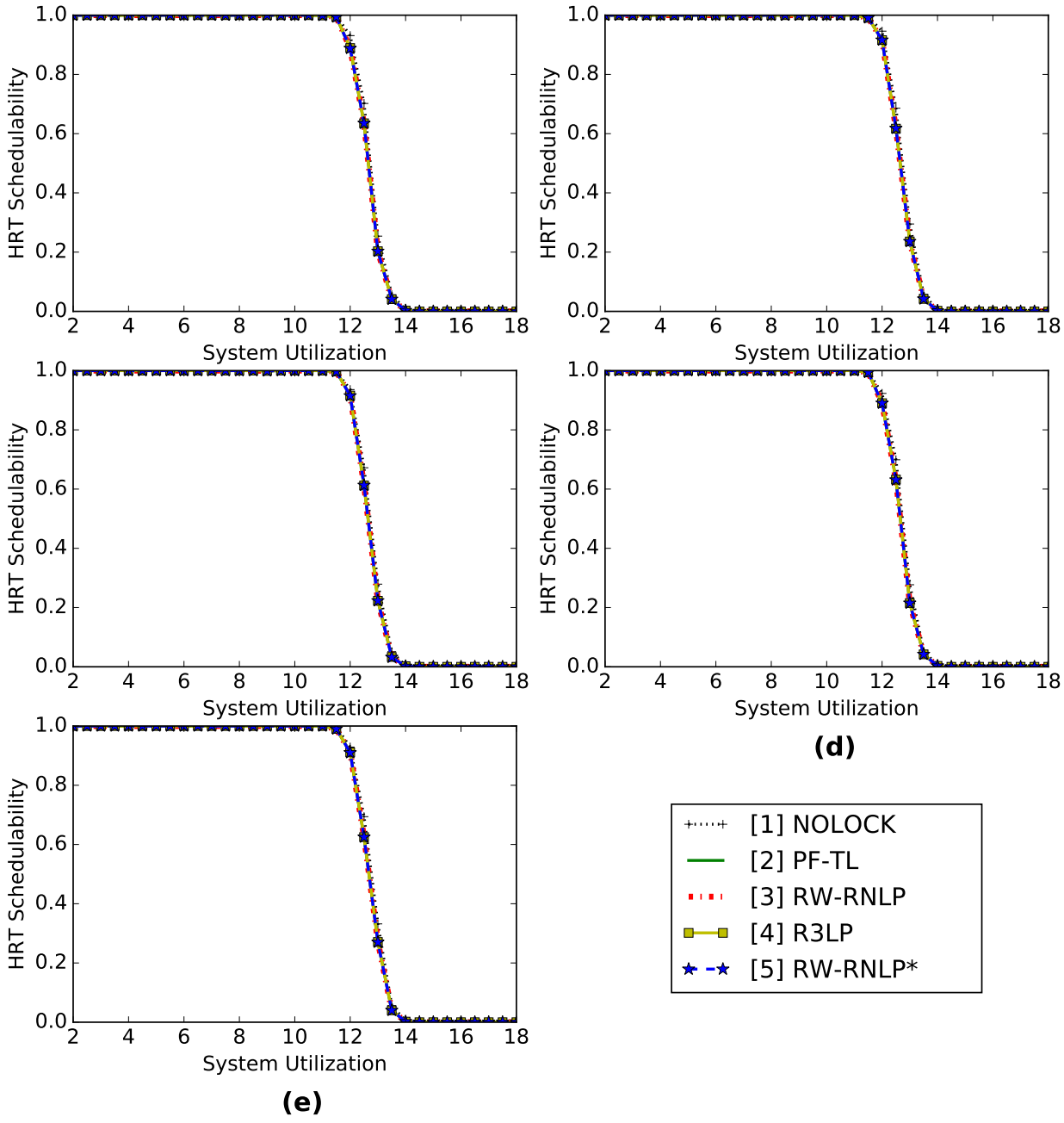



Figure 683: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, short critical-section lengths, and read probability 0.5 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

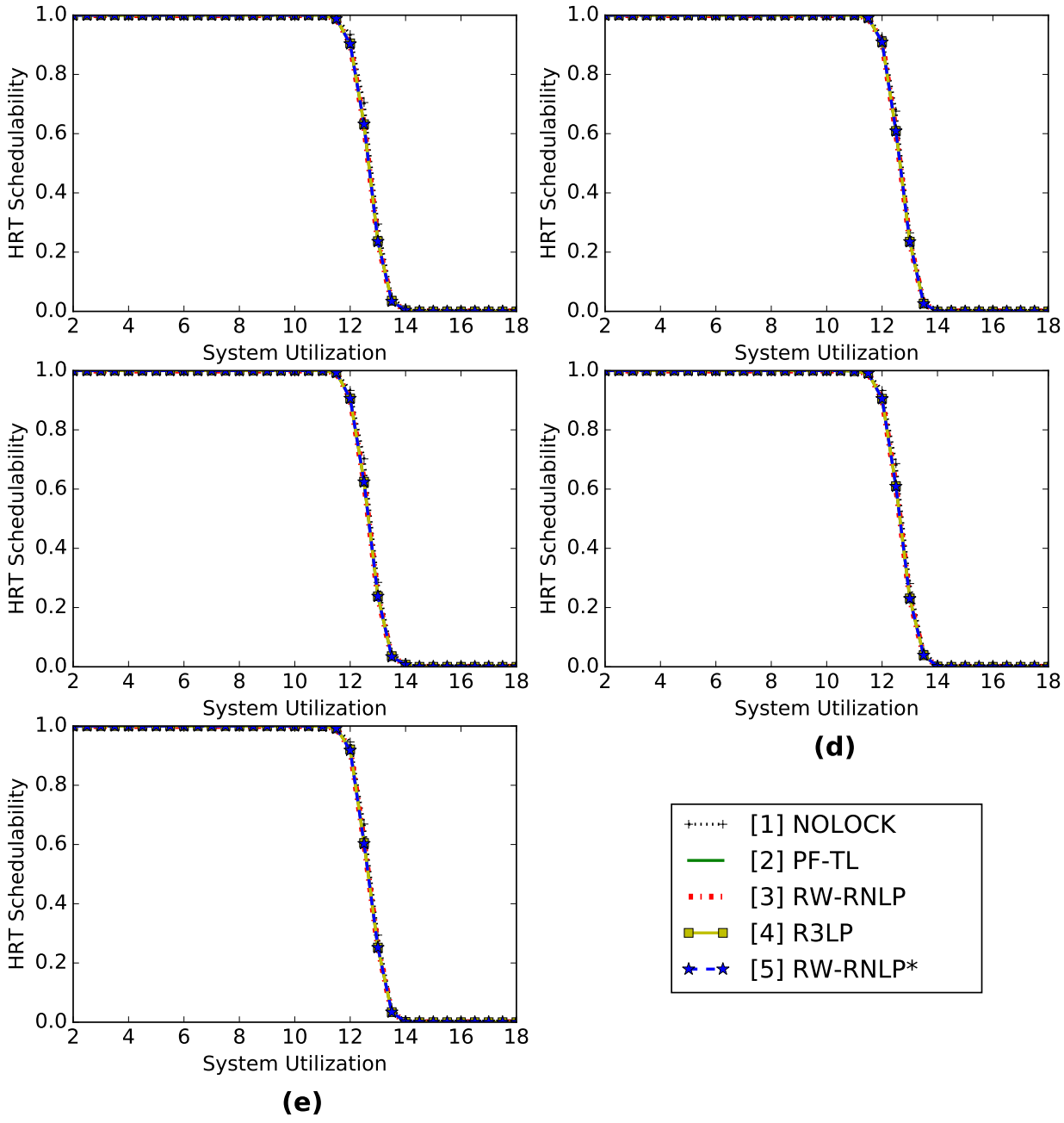



Figure 684: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, short critical-section lengths, and read probability 0.5 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

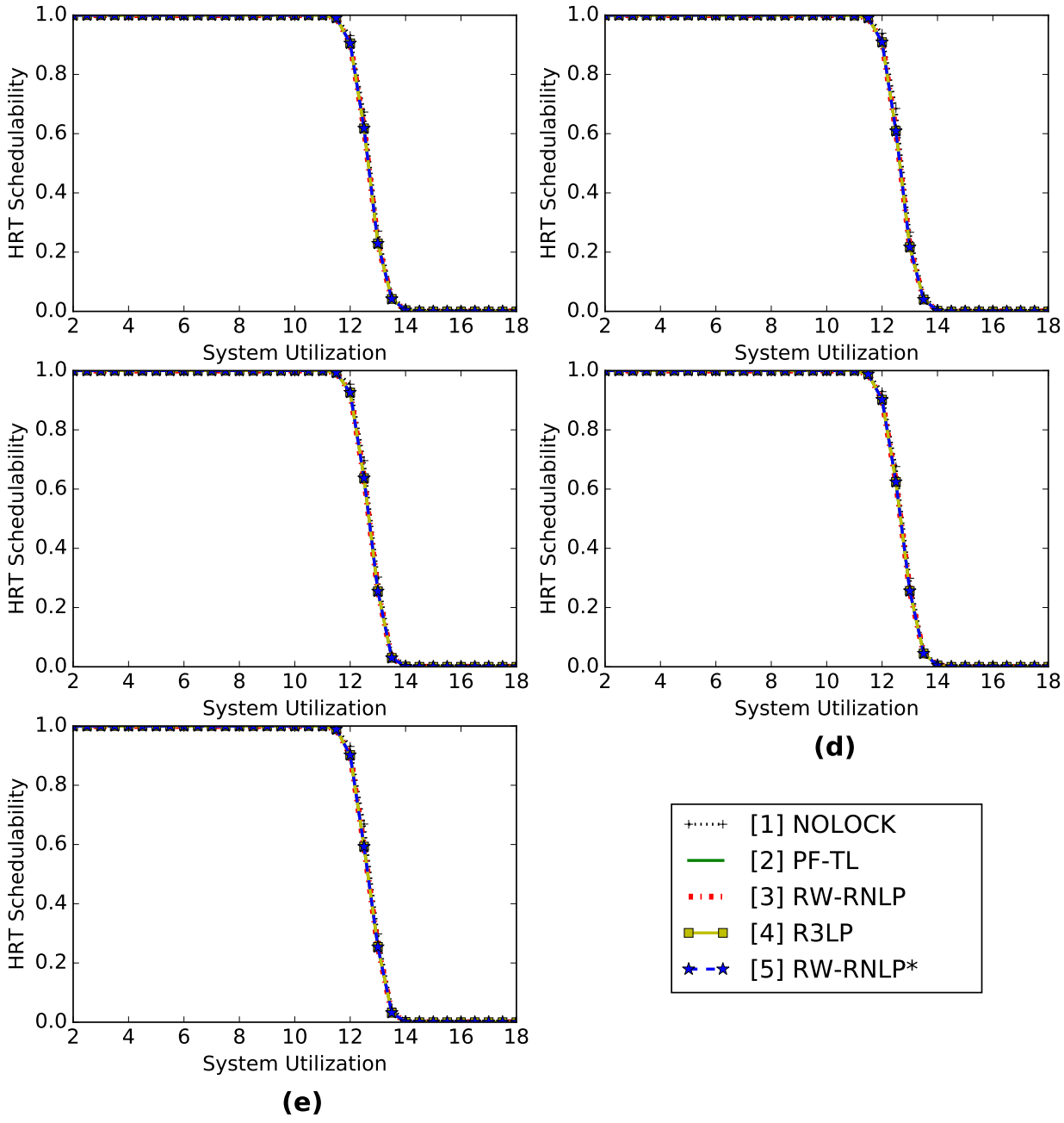



Figure 685: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, short critical-section lengths, and read probability 0.8 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

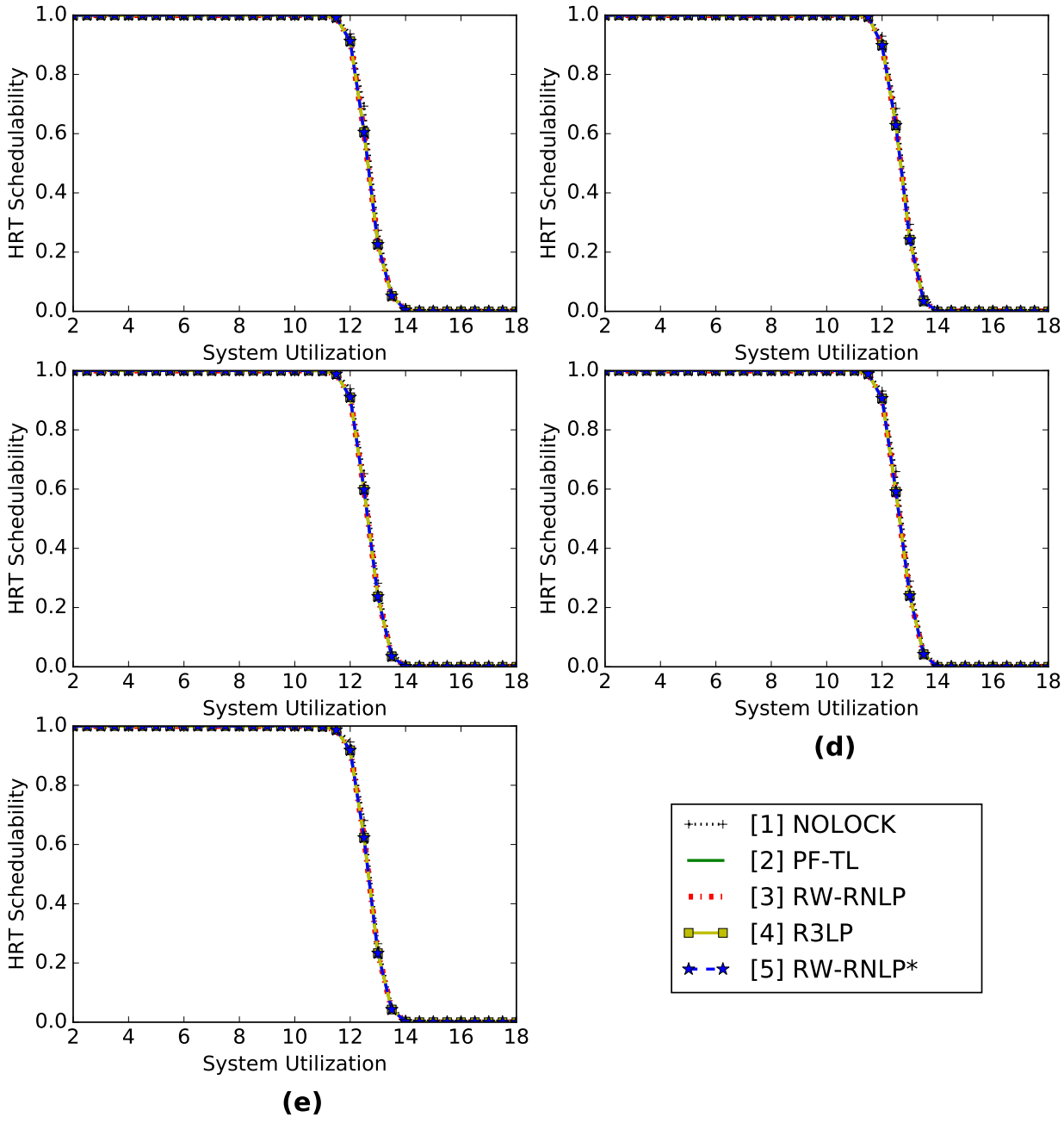



Figure 686: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, short critical-section lengths, and read probability 0.8 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

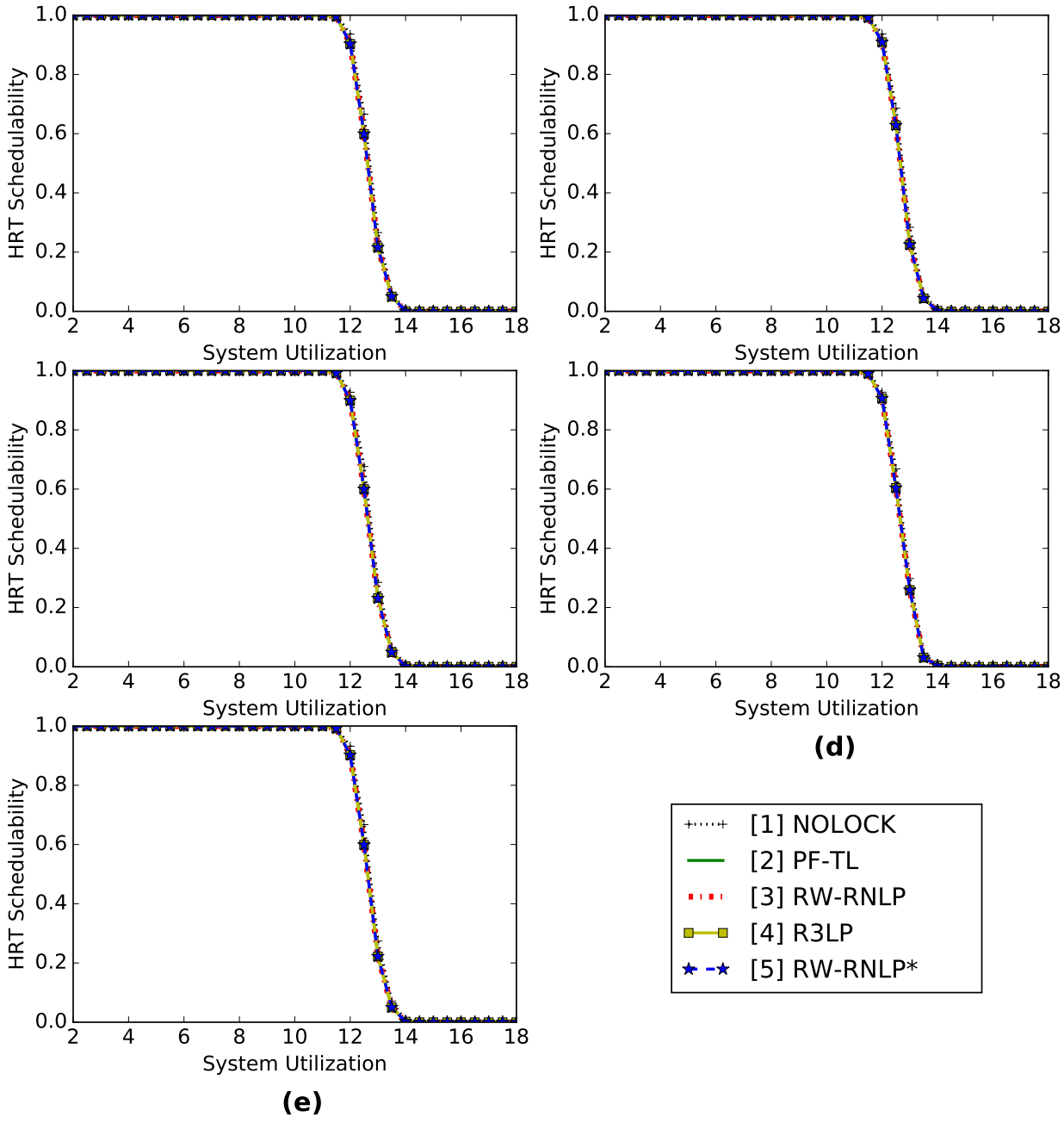



Figure 687: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, short critical-section lengths, and read probability 0.8 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

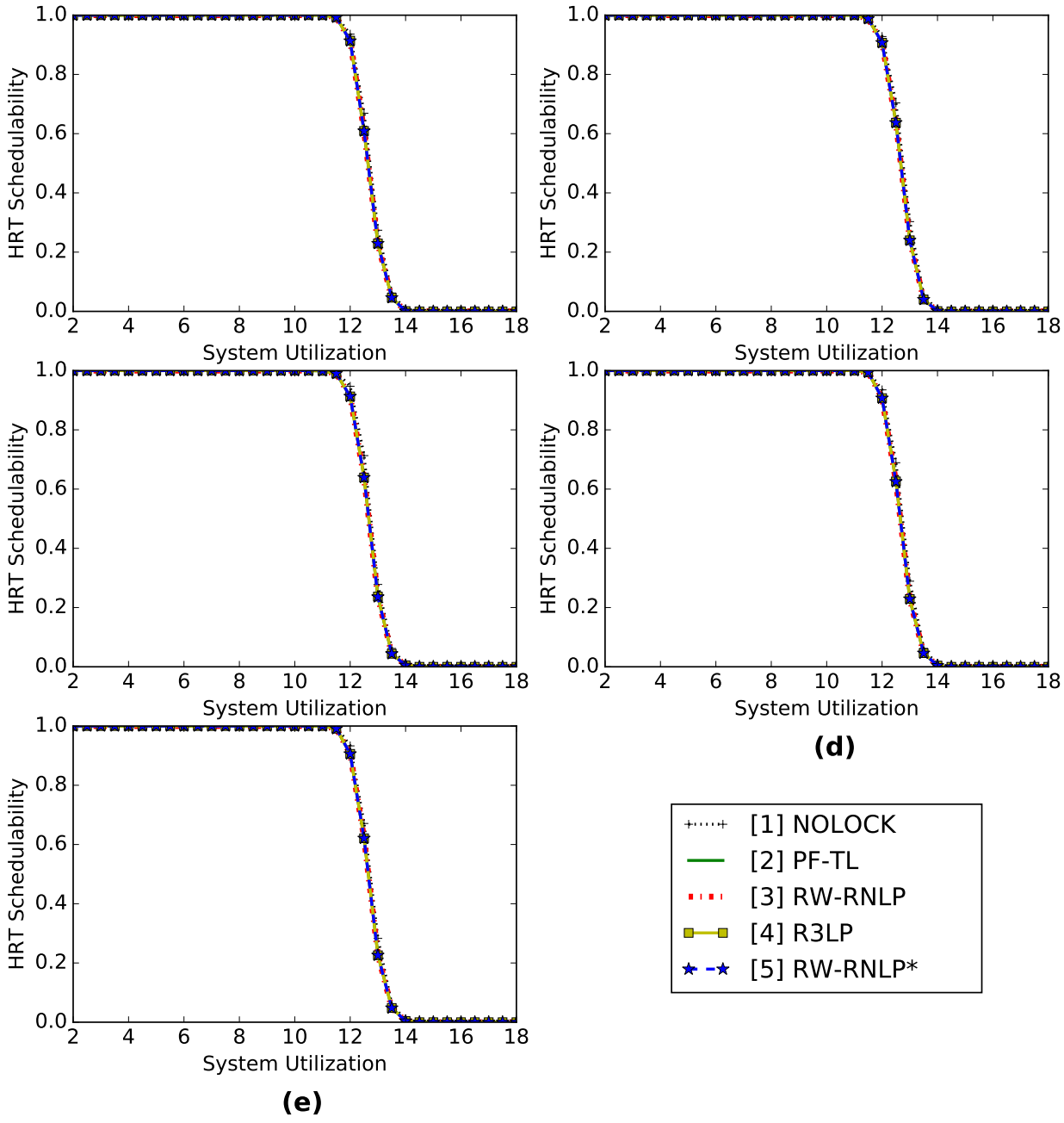



Figure 688: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, short critical-section lengths, and read probability 0.8 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

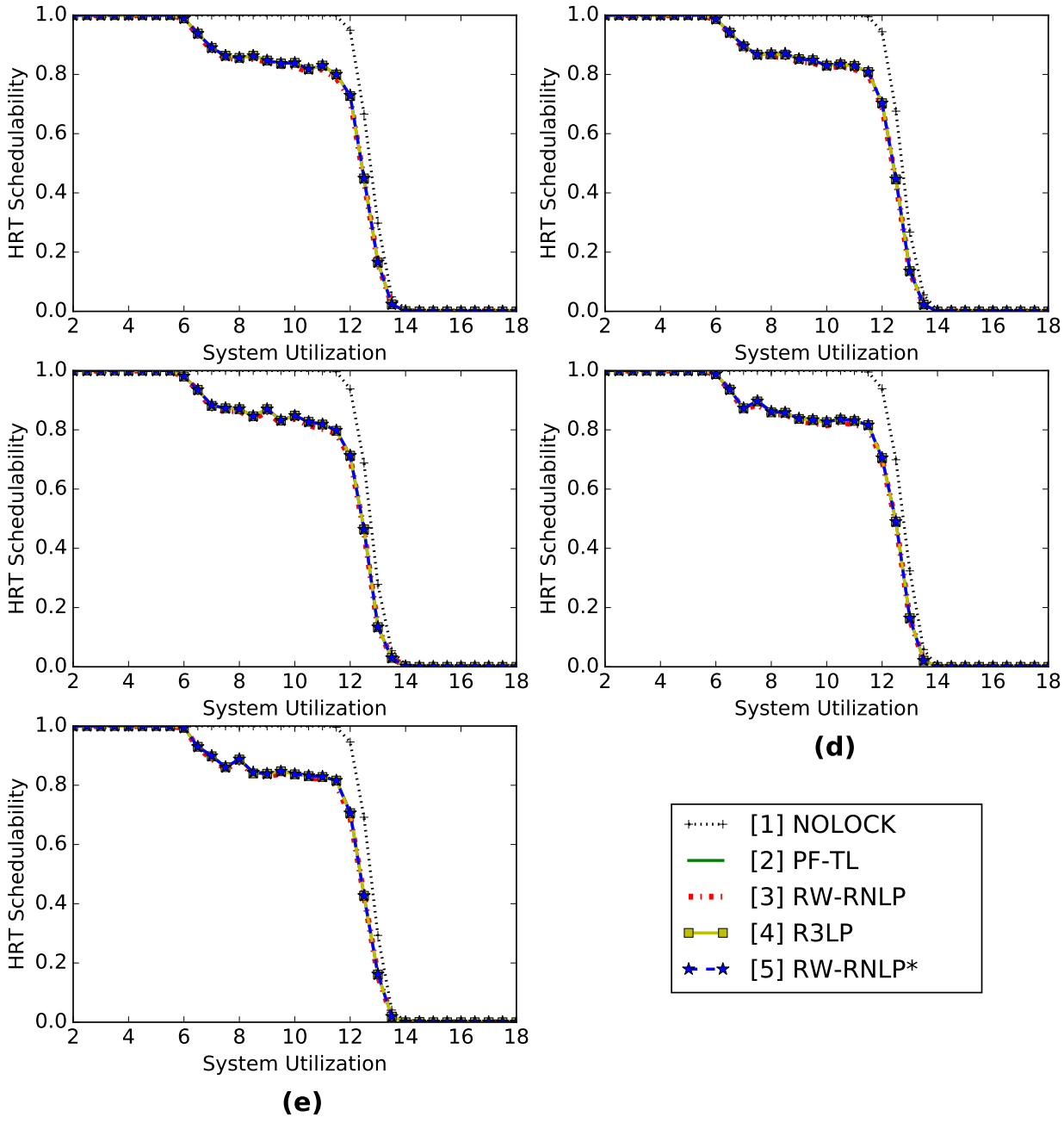



Figure 689: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, long critical-section lengths, and read probability 0 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

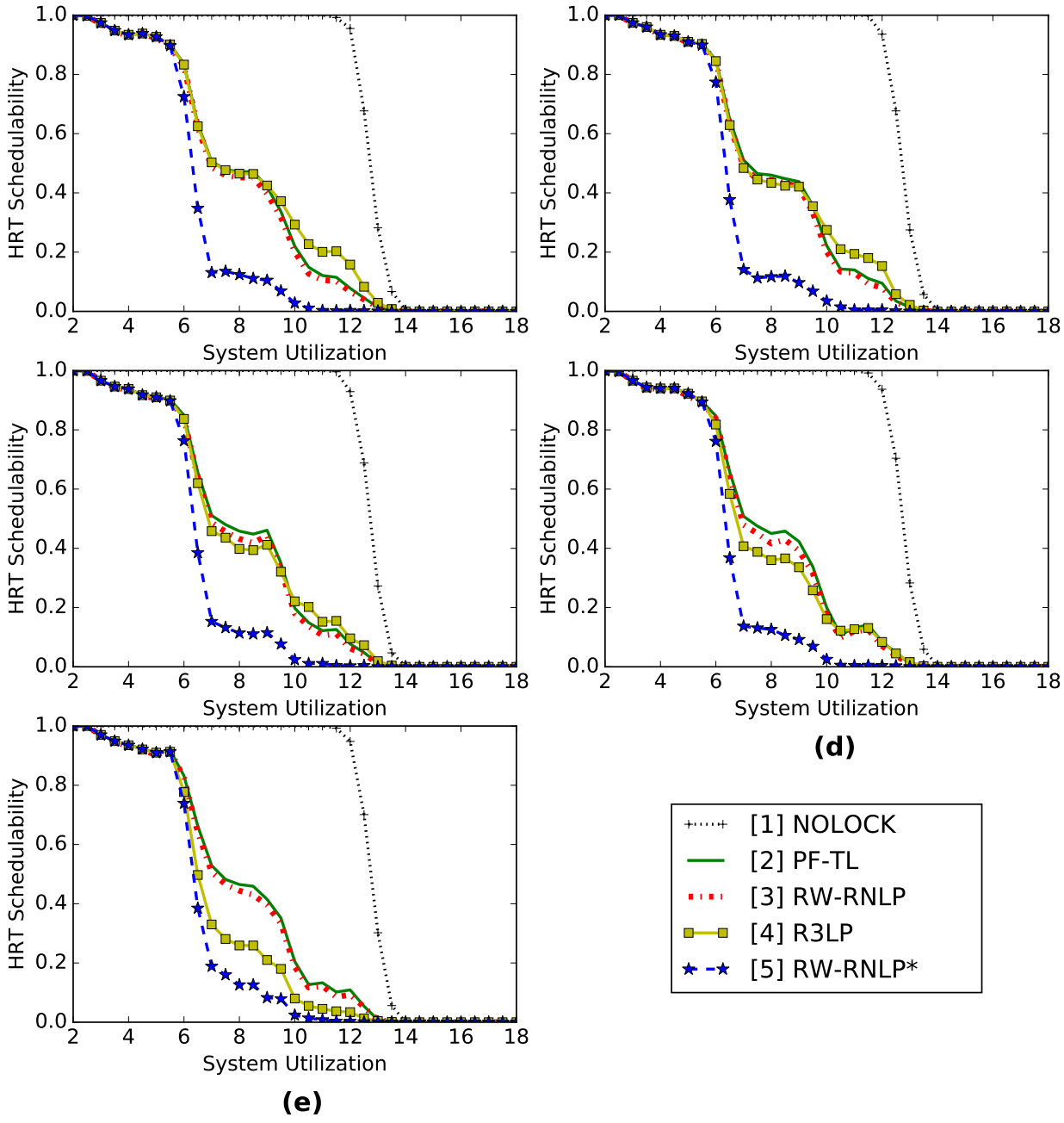



Figure 690: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, long critical-section lengths, and read probability 0 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

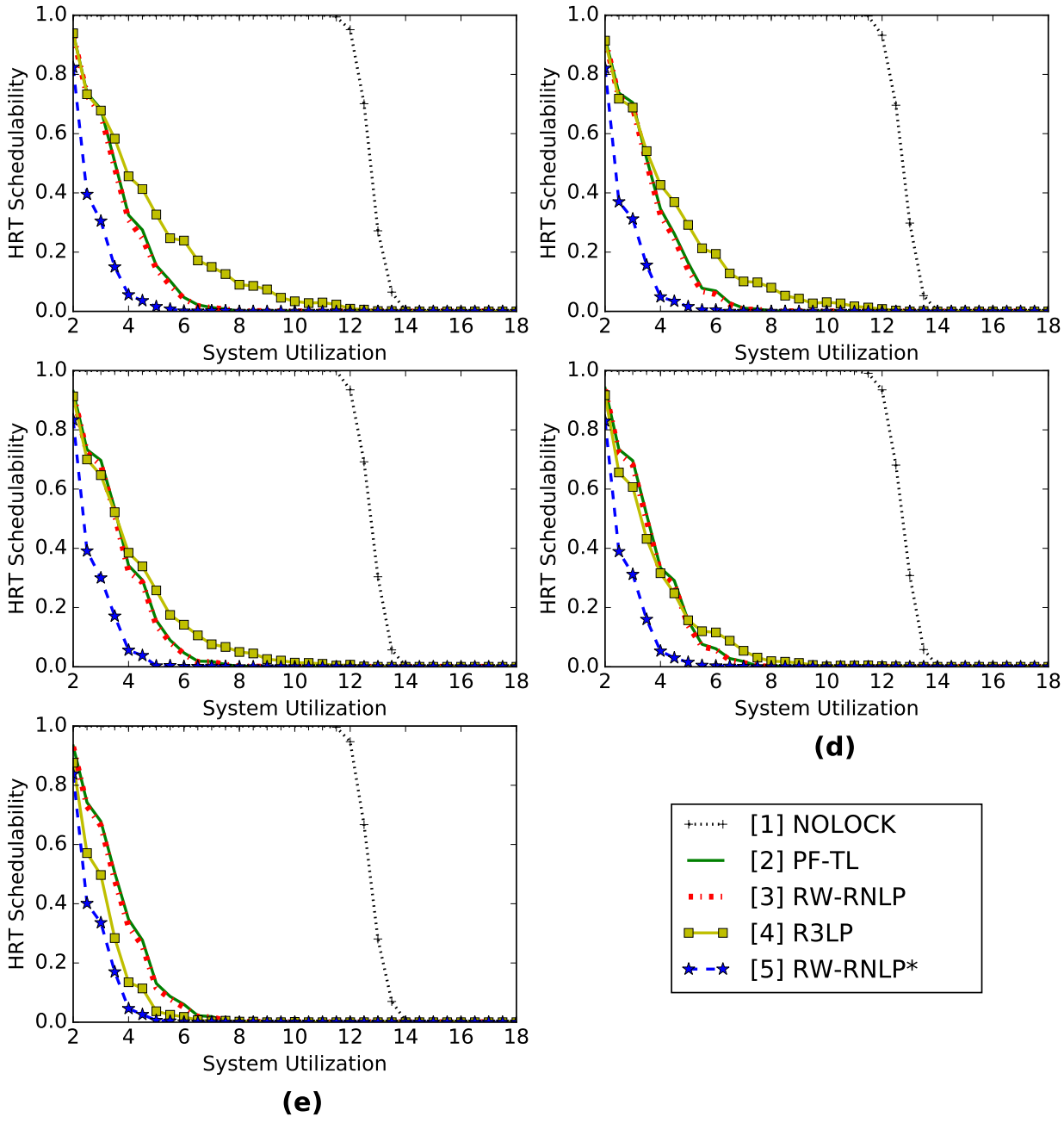



Figure 691: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, long critical-section lengths, and read probability 0 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

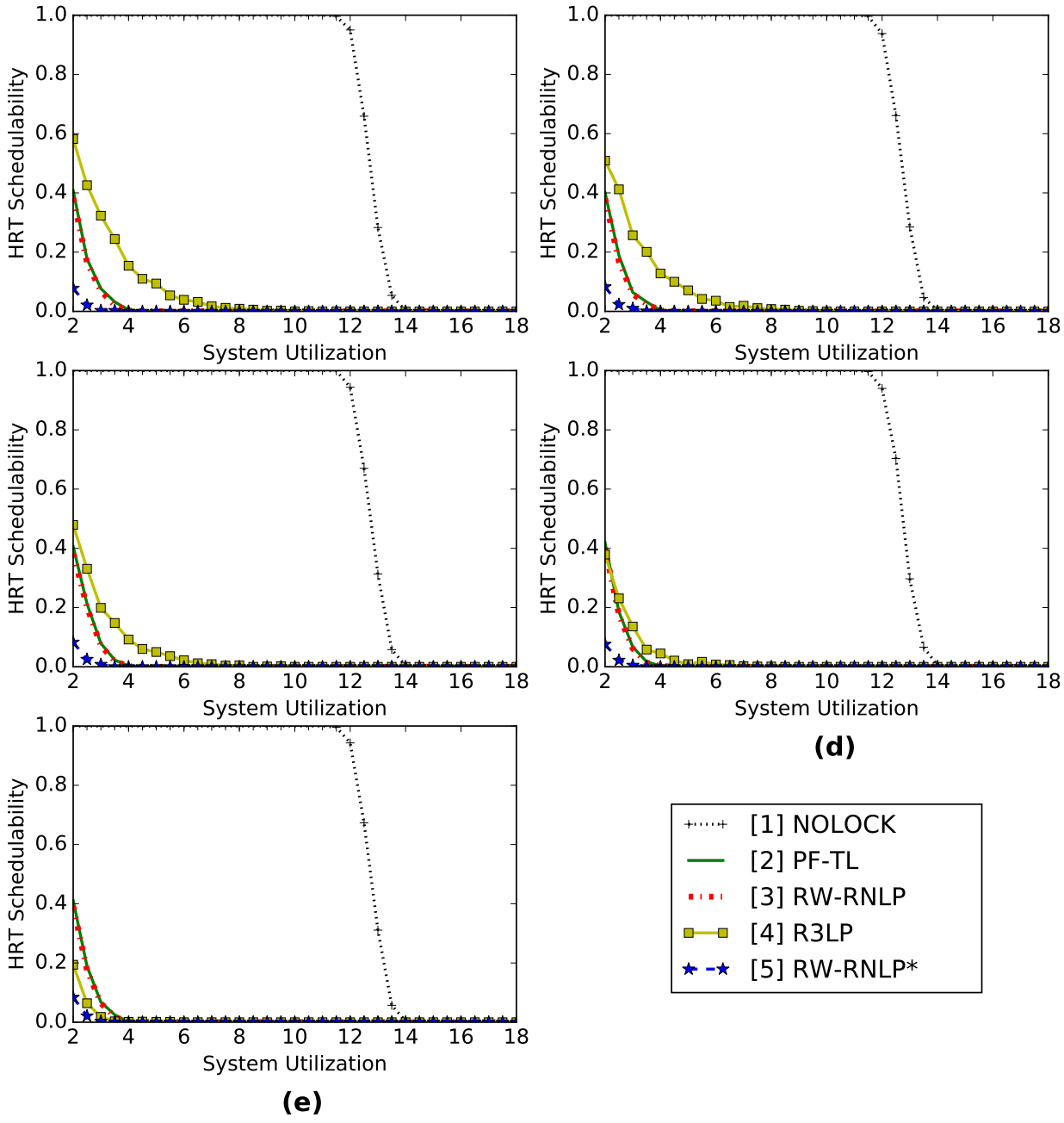



Figure 692: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, long critical-section lengths, and read probability 0 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

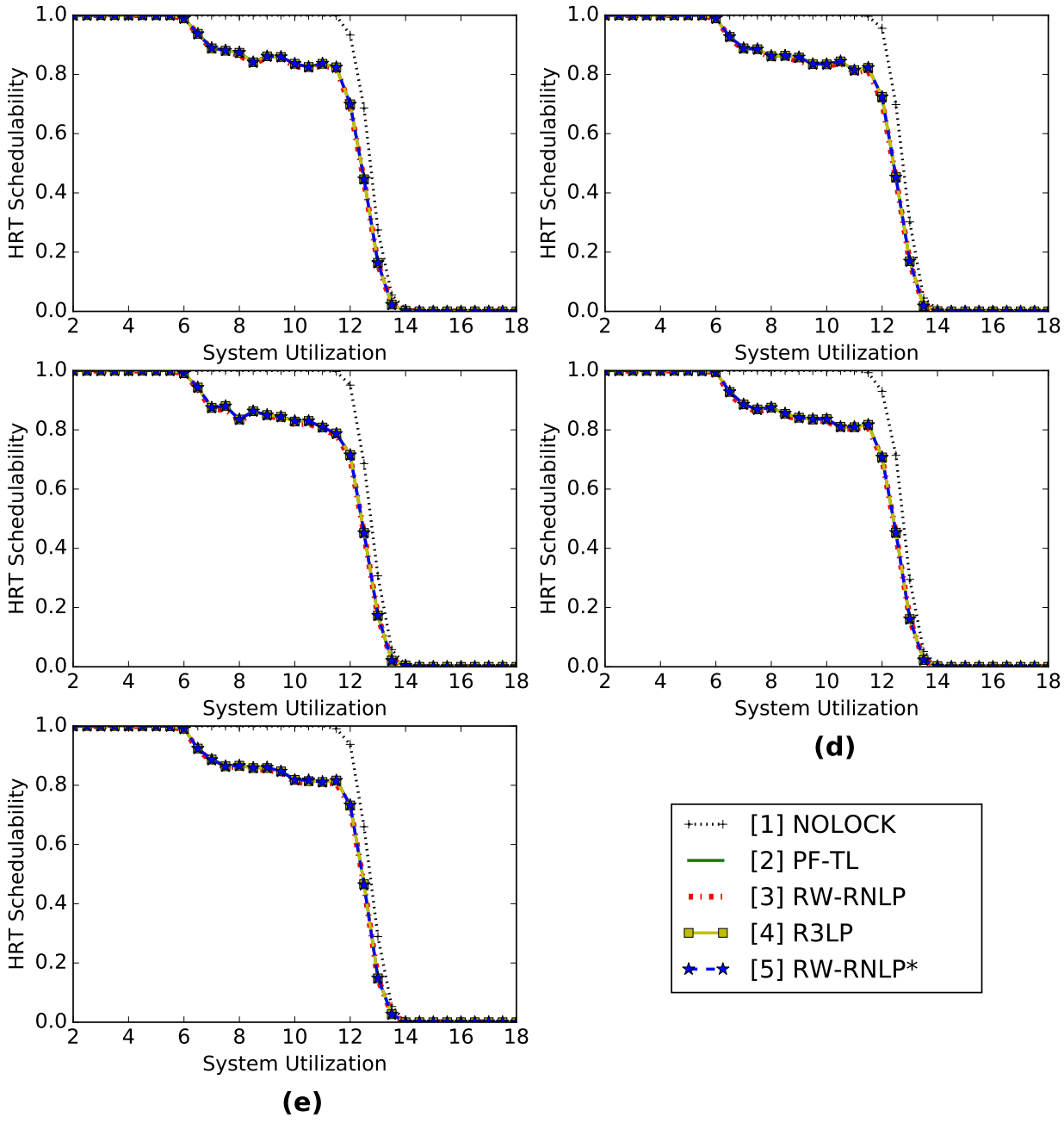



Figure 693: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, long critical-section lengths, and read probability 0.2 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

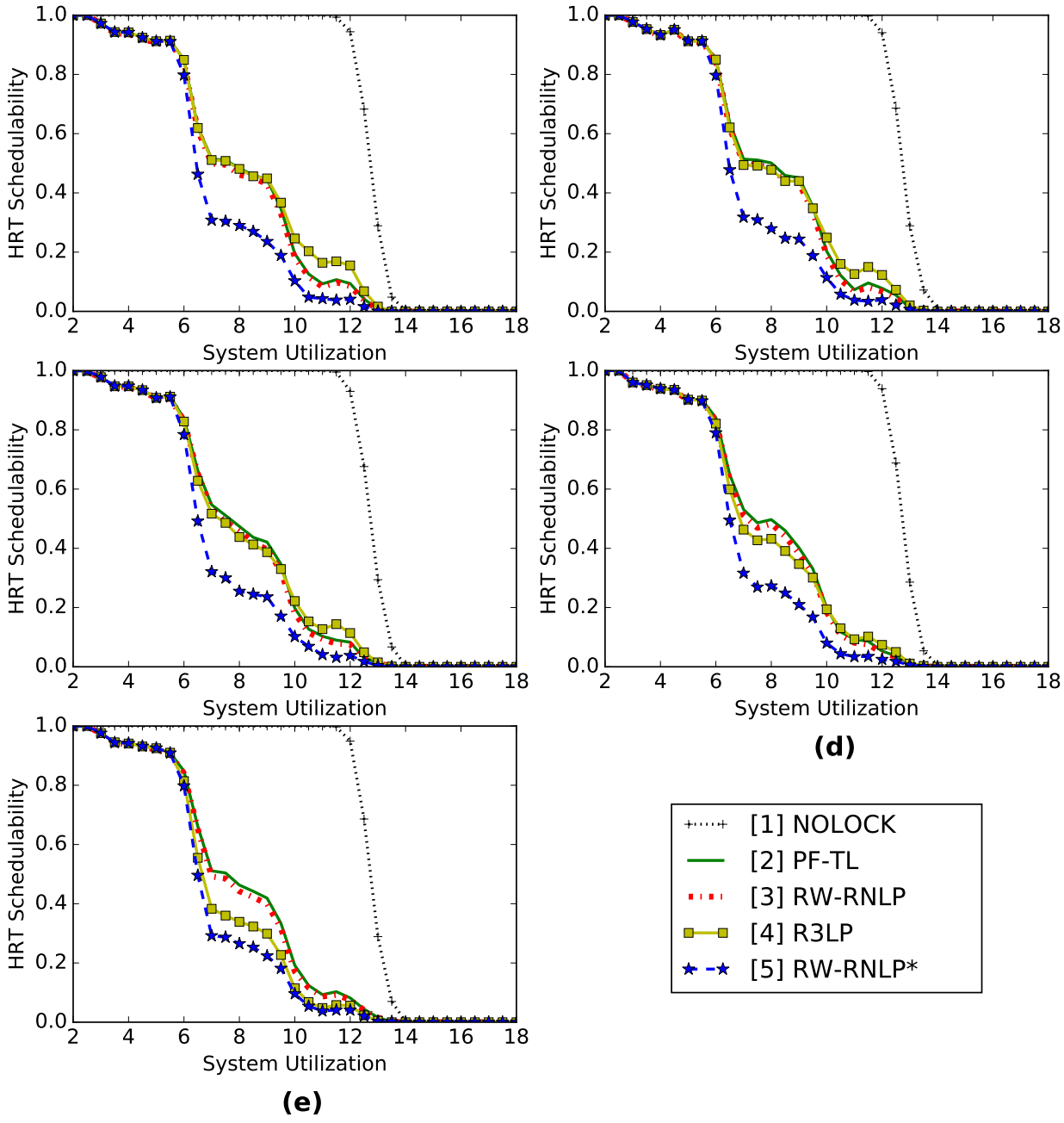



Figure 694: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, long critical-section lengths, and read probability 0.2 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

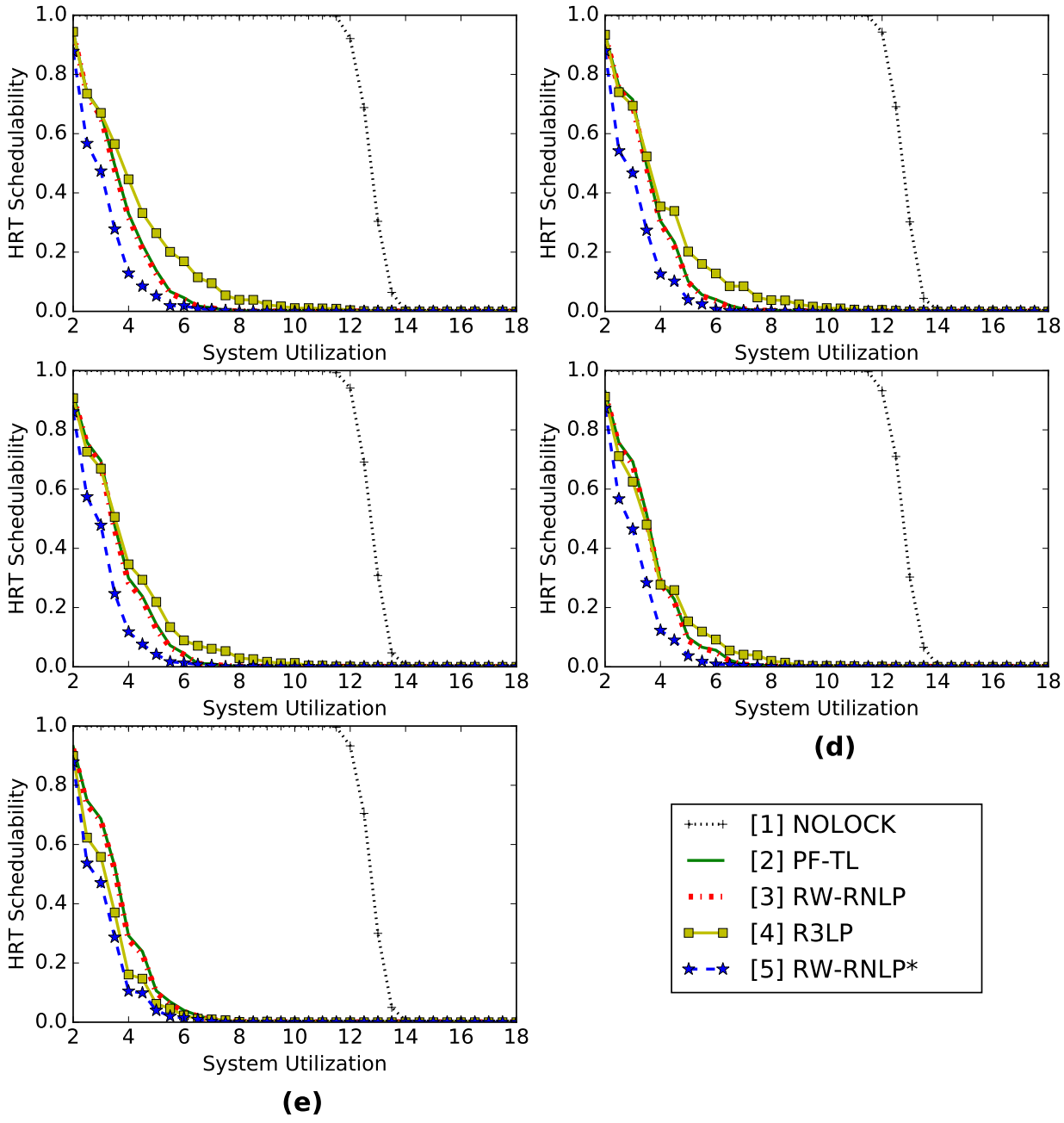



Figure 695: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, long critical-section lengths, and read probability 0.2 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

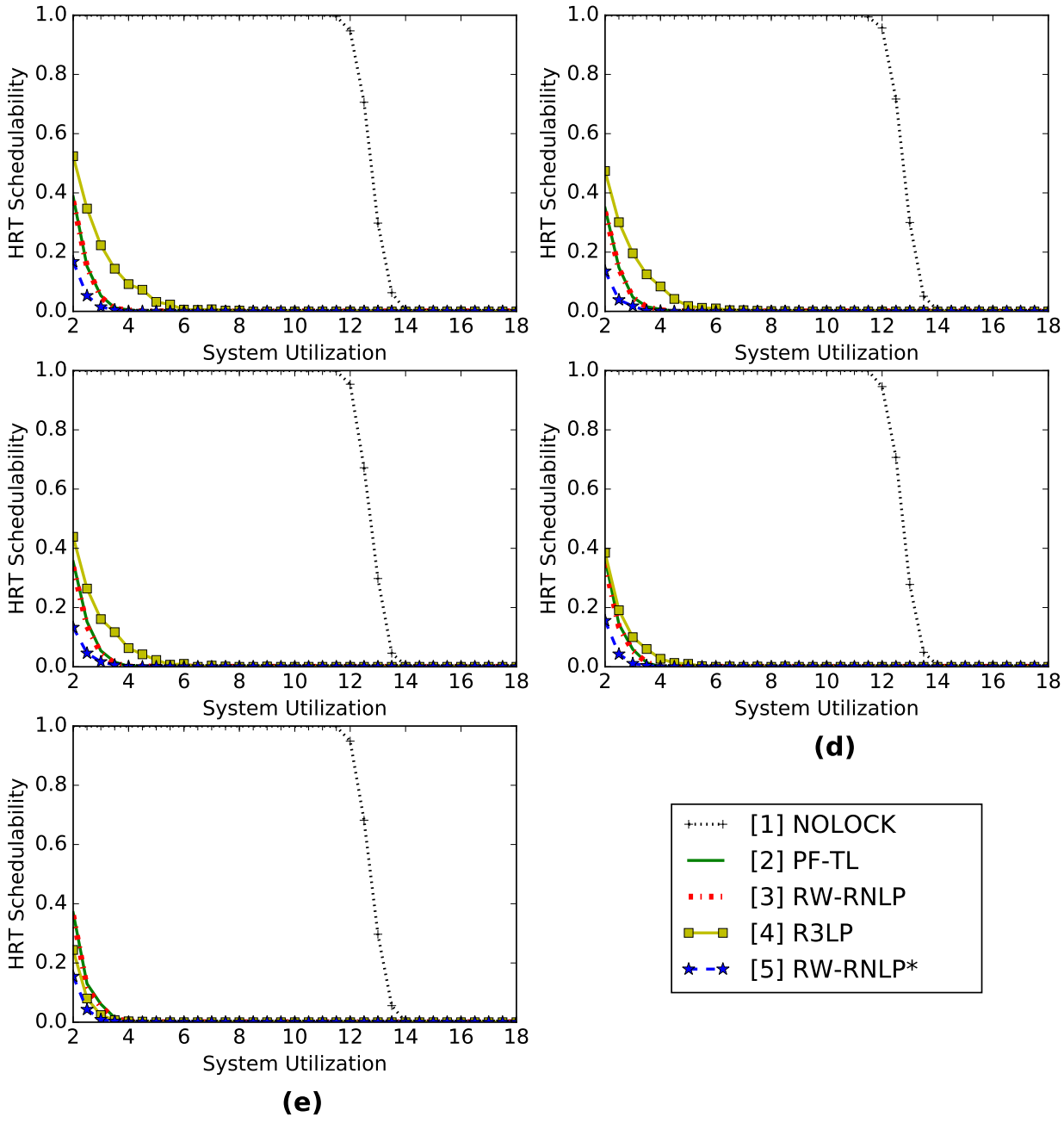



Figure 696: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, long critical-section lengths, and read probability 0.2 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

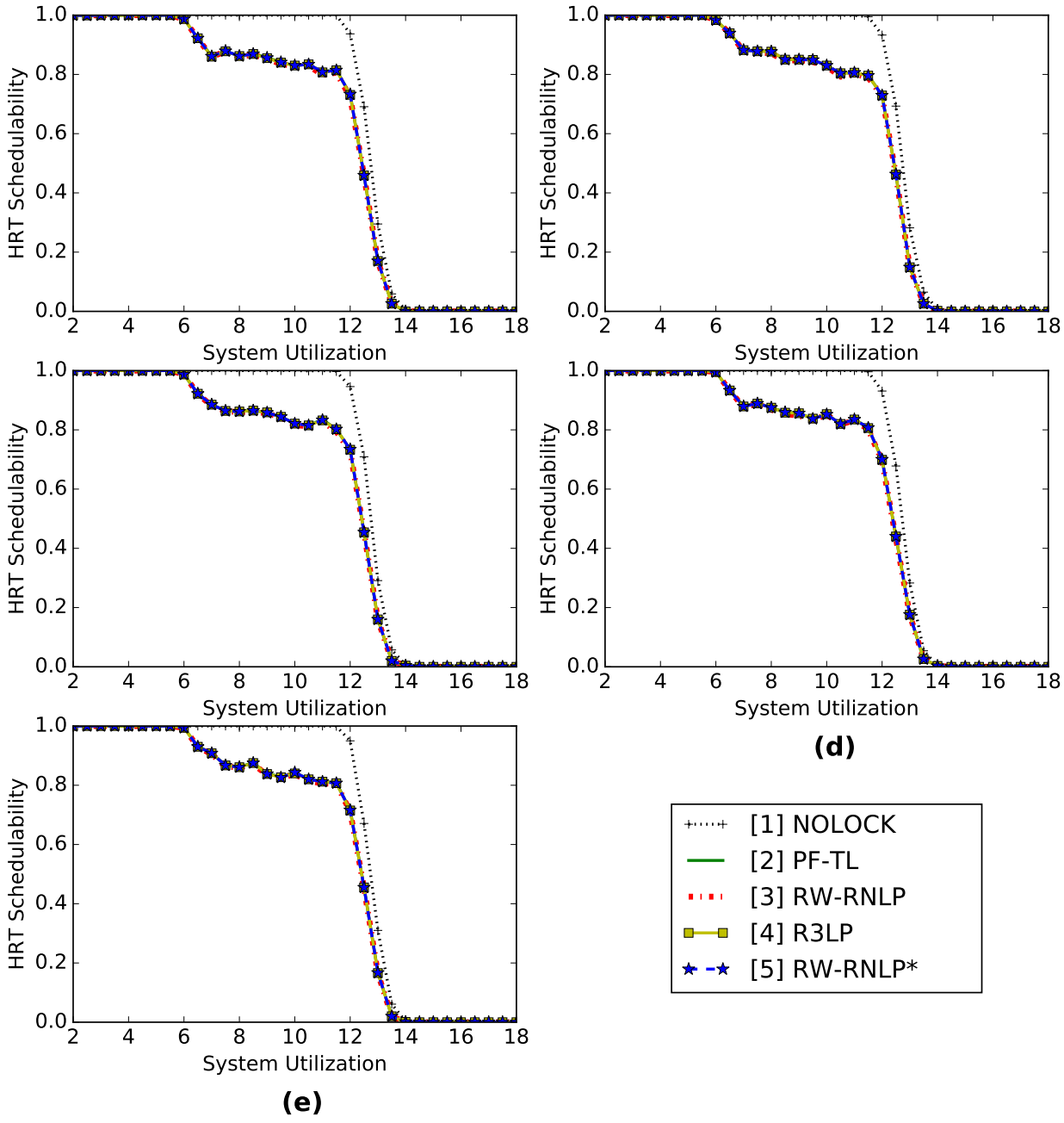



Figure 697: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, long critical-section lengths, and read probability 0.5 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

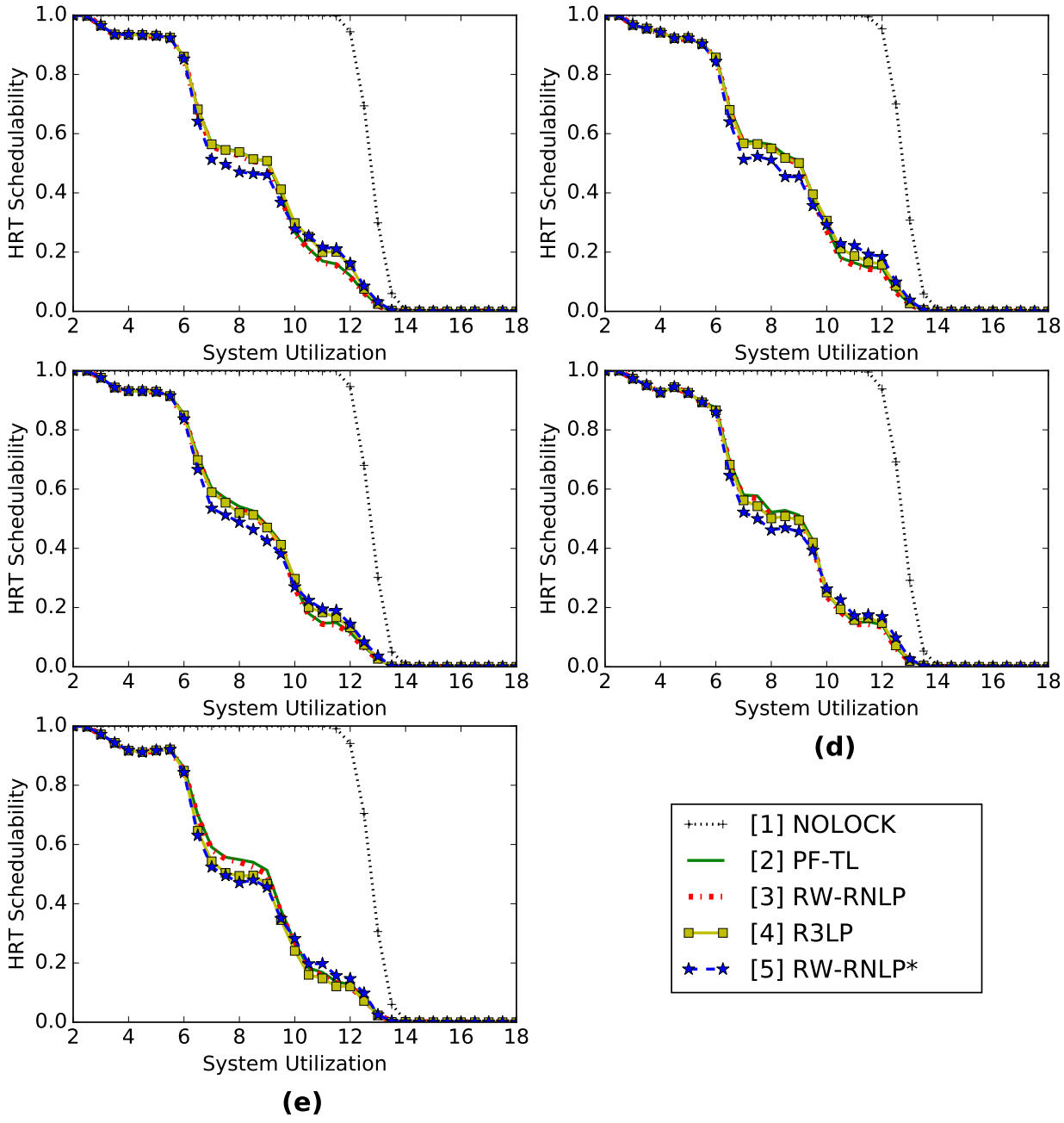



Figure 698: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, long critical-section lengths, and read probability 0.5 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

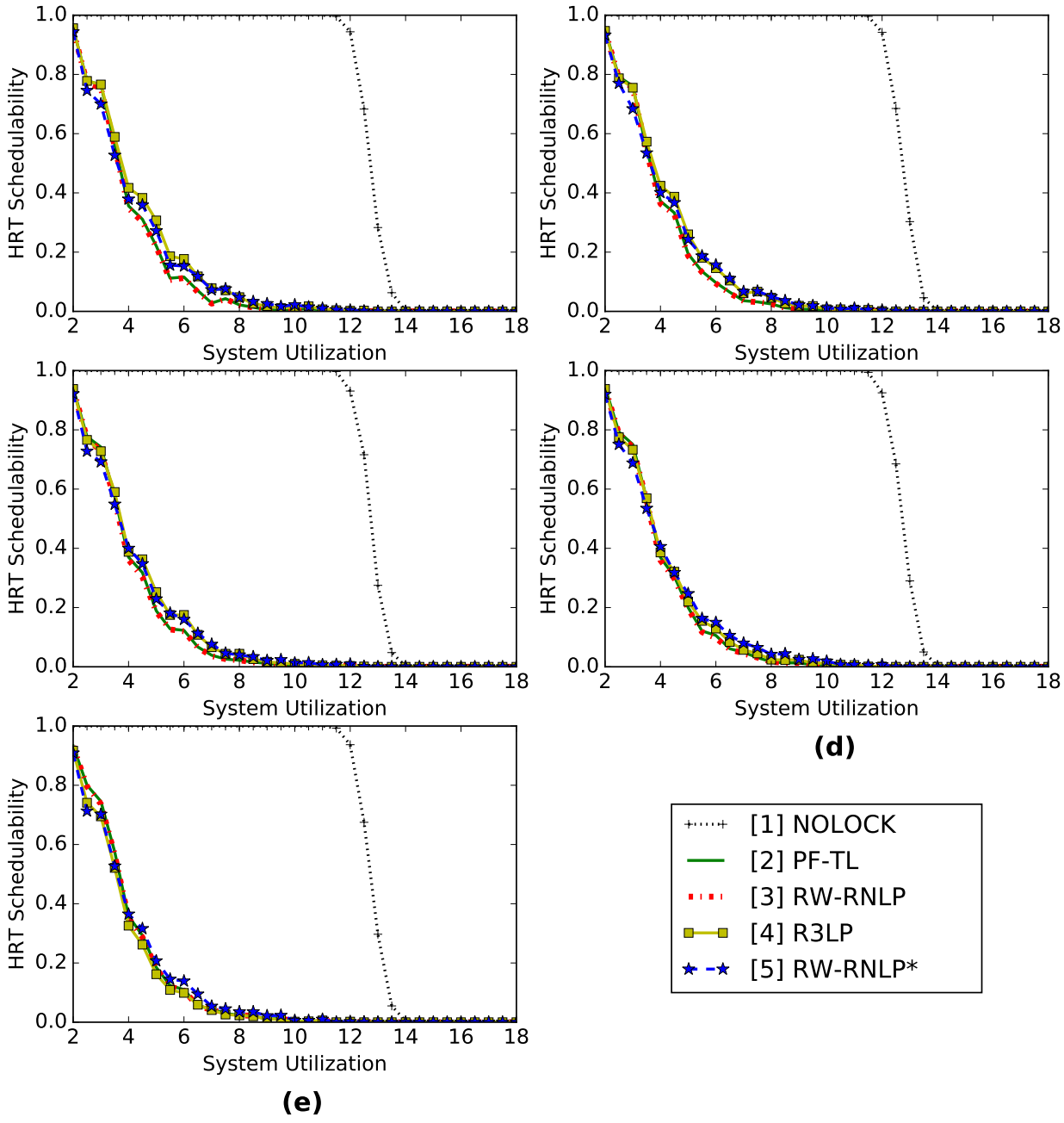



Figure 699: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, long critical-section lengths, and read probability 0.5 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

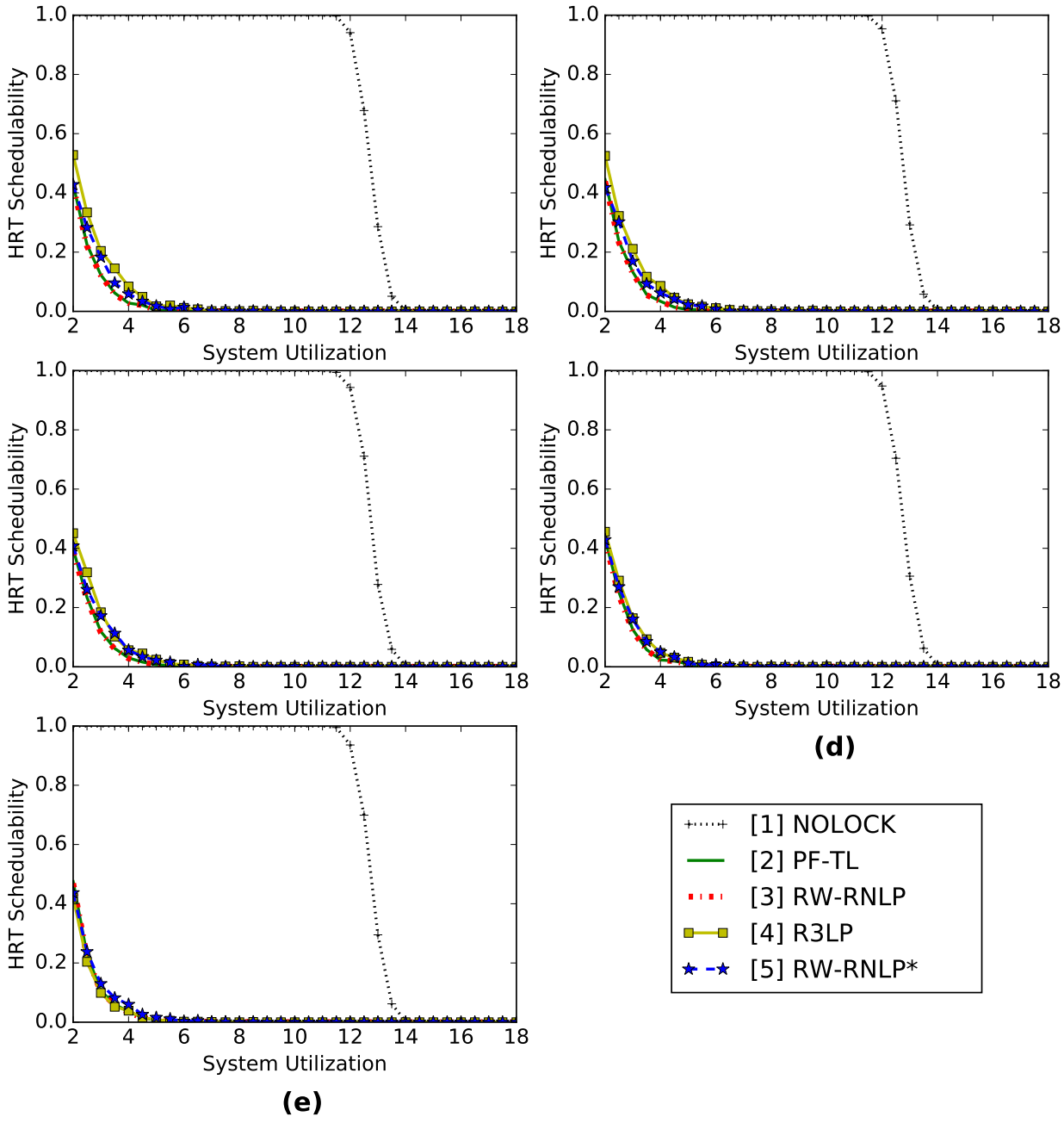



Figure 700: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, long critical-section lengths, and read probability 0.5 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

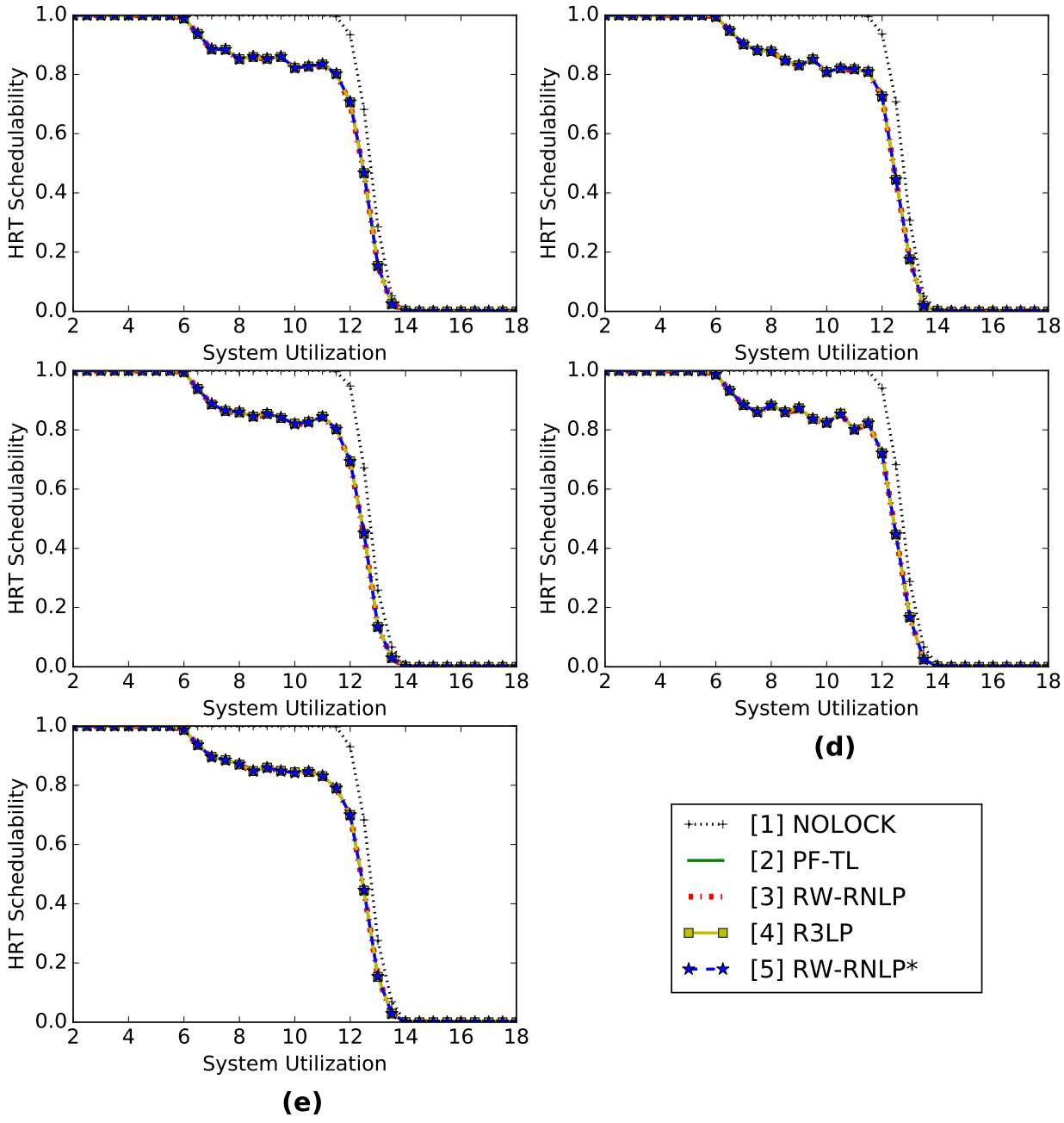



Figure 701: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, long critical-section lengths, and read probability 0.8 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

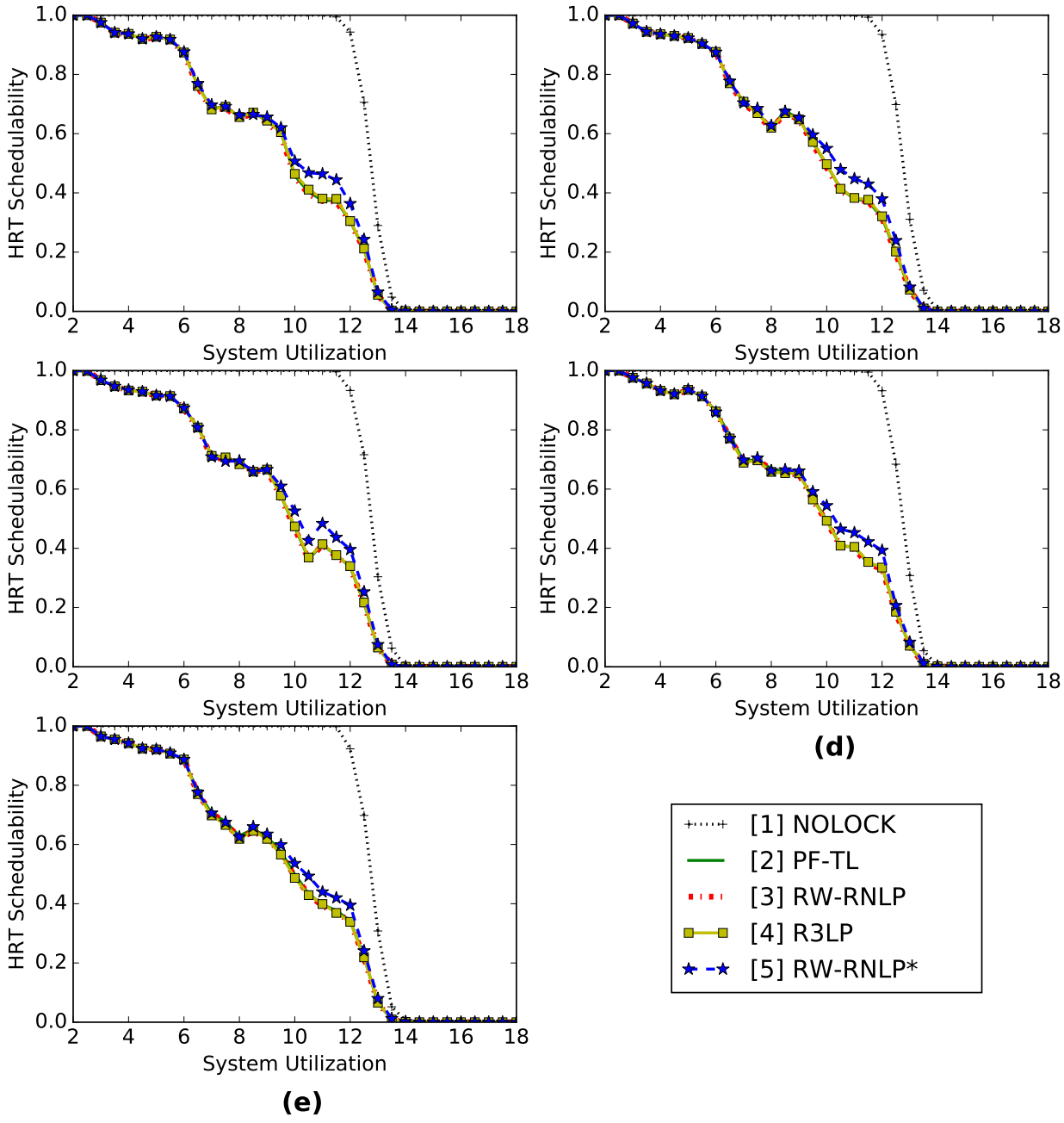



Figure 702: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, long critical-section lengths, and read probability 0.8 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

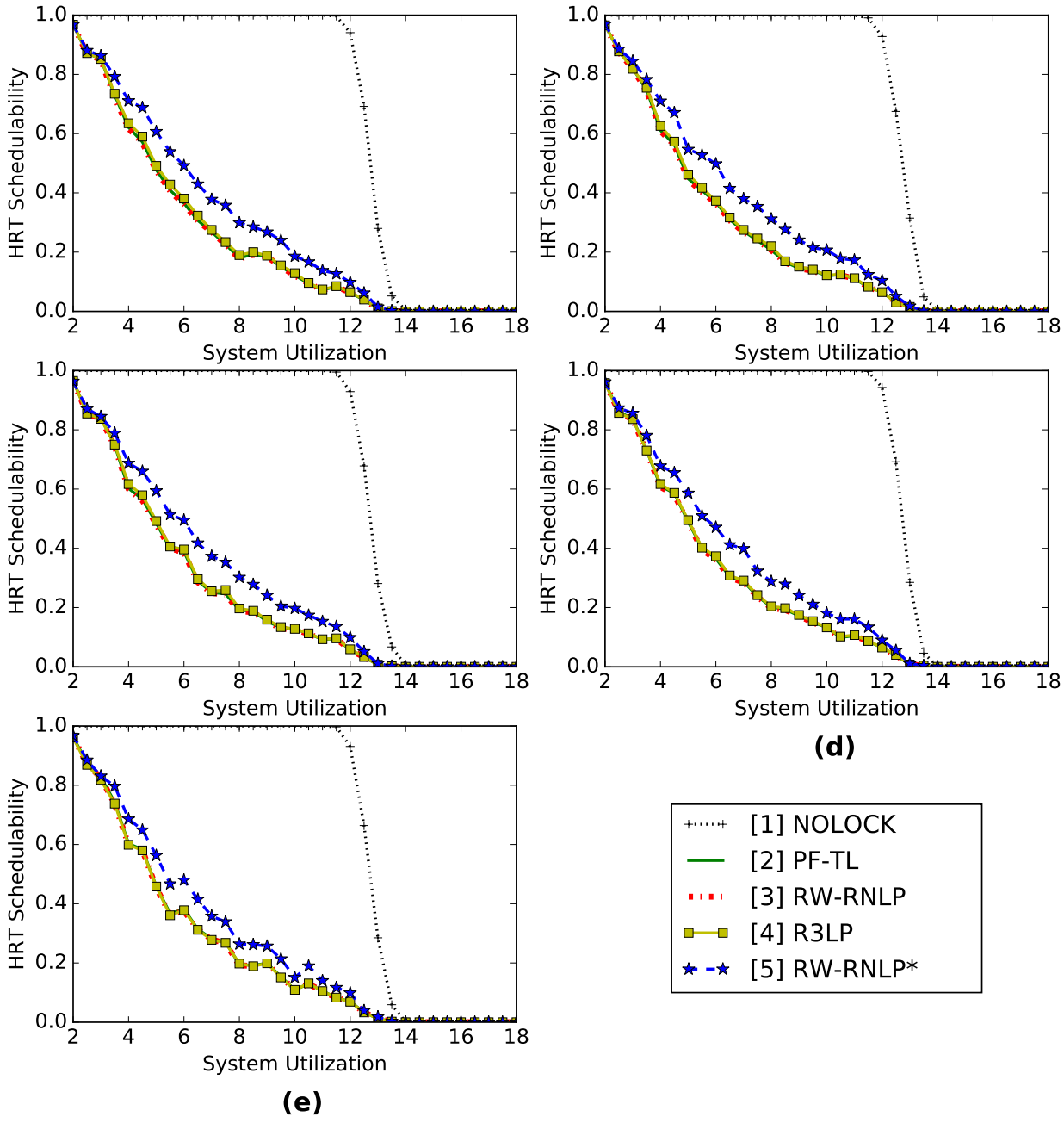



Figure 703: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, long critical-section lengths, and read probability 0.8 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

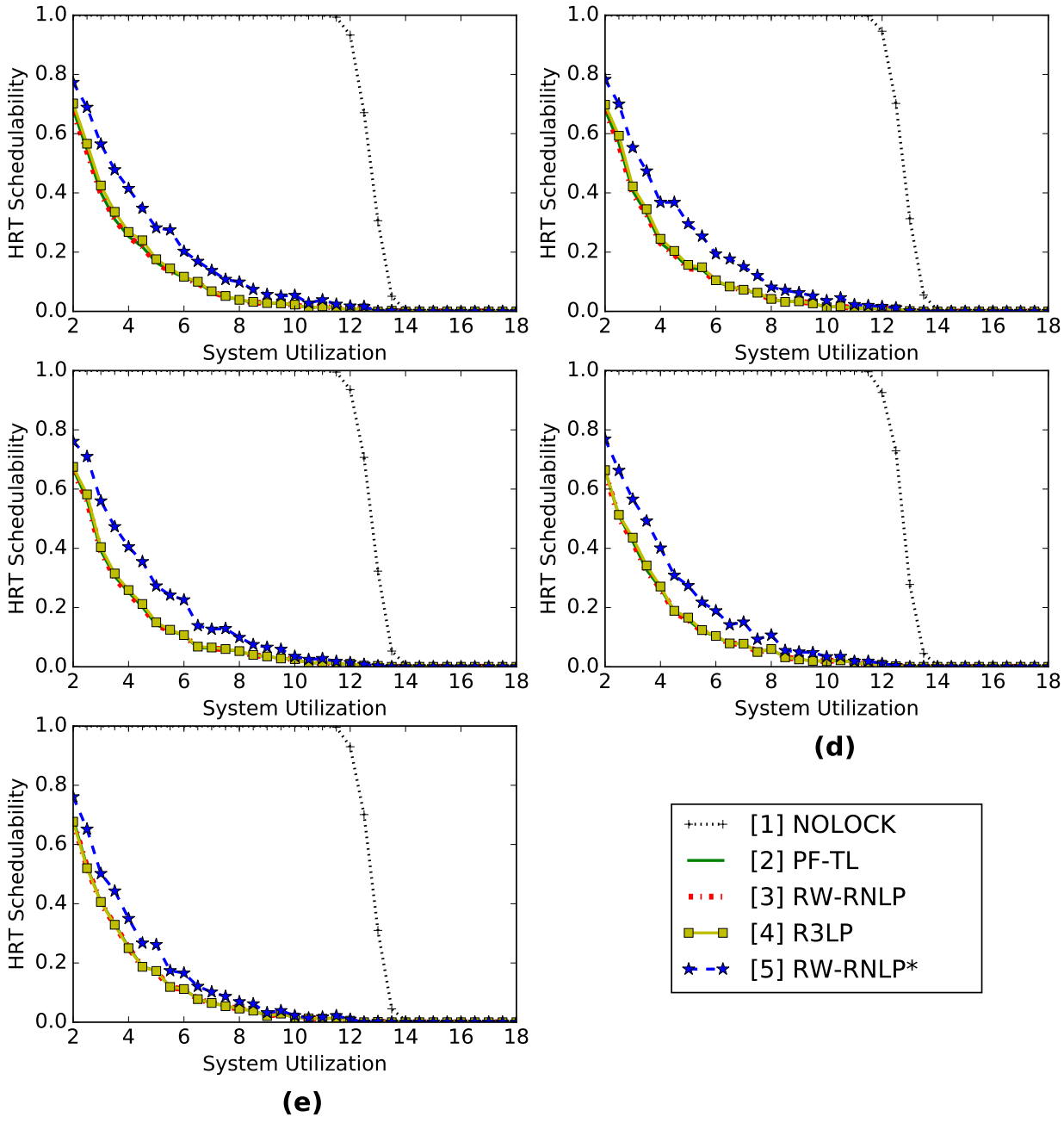



Figure 704: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, short periods, long critical-section lengths, and read probability 0.8 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

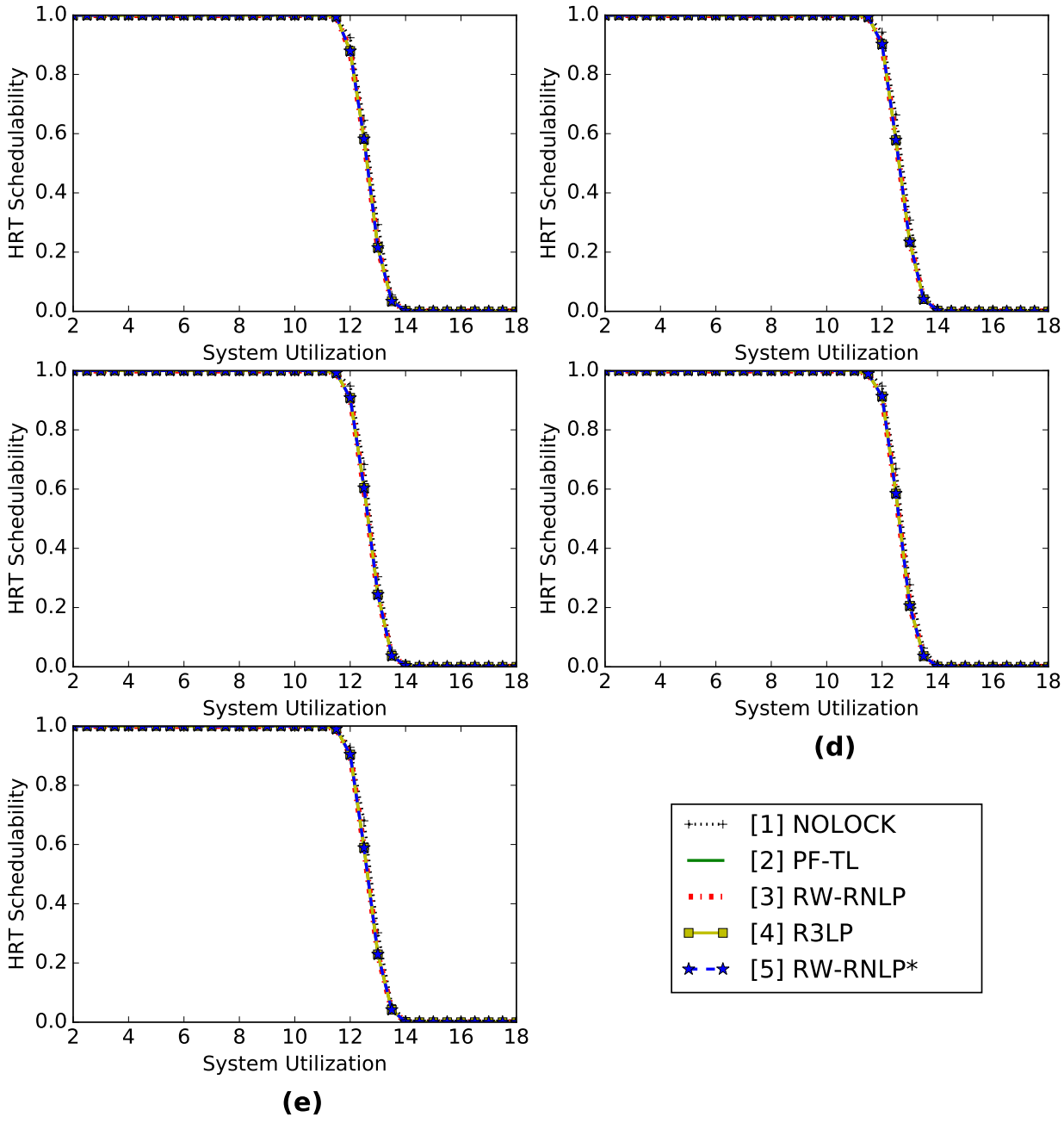



Figure 705: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, long critical-section lengths, and read probability 0 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

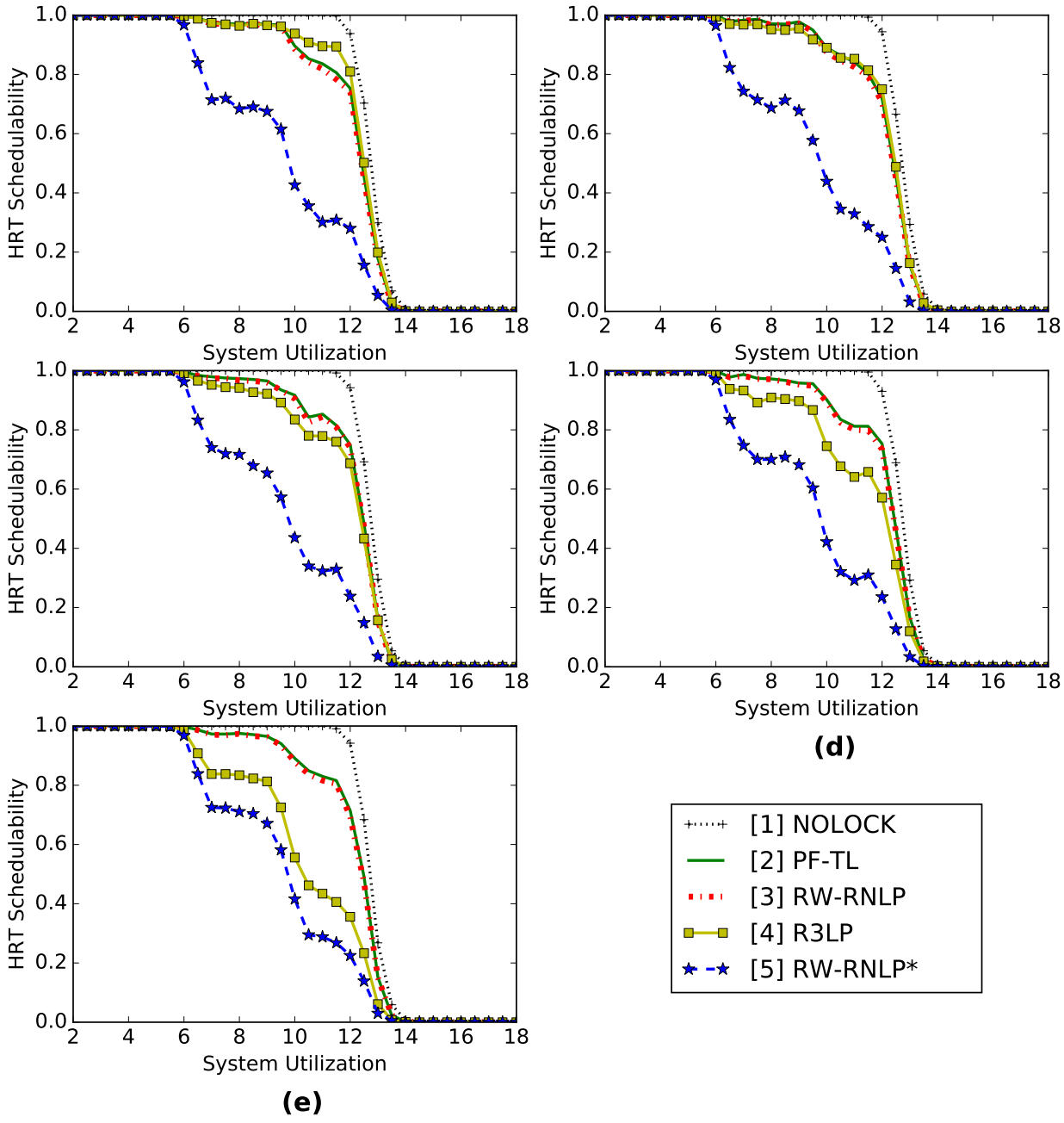



Figure 706: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, long critical-section lengths, and read probability 0 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

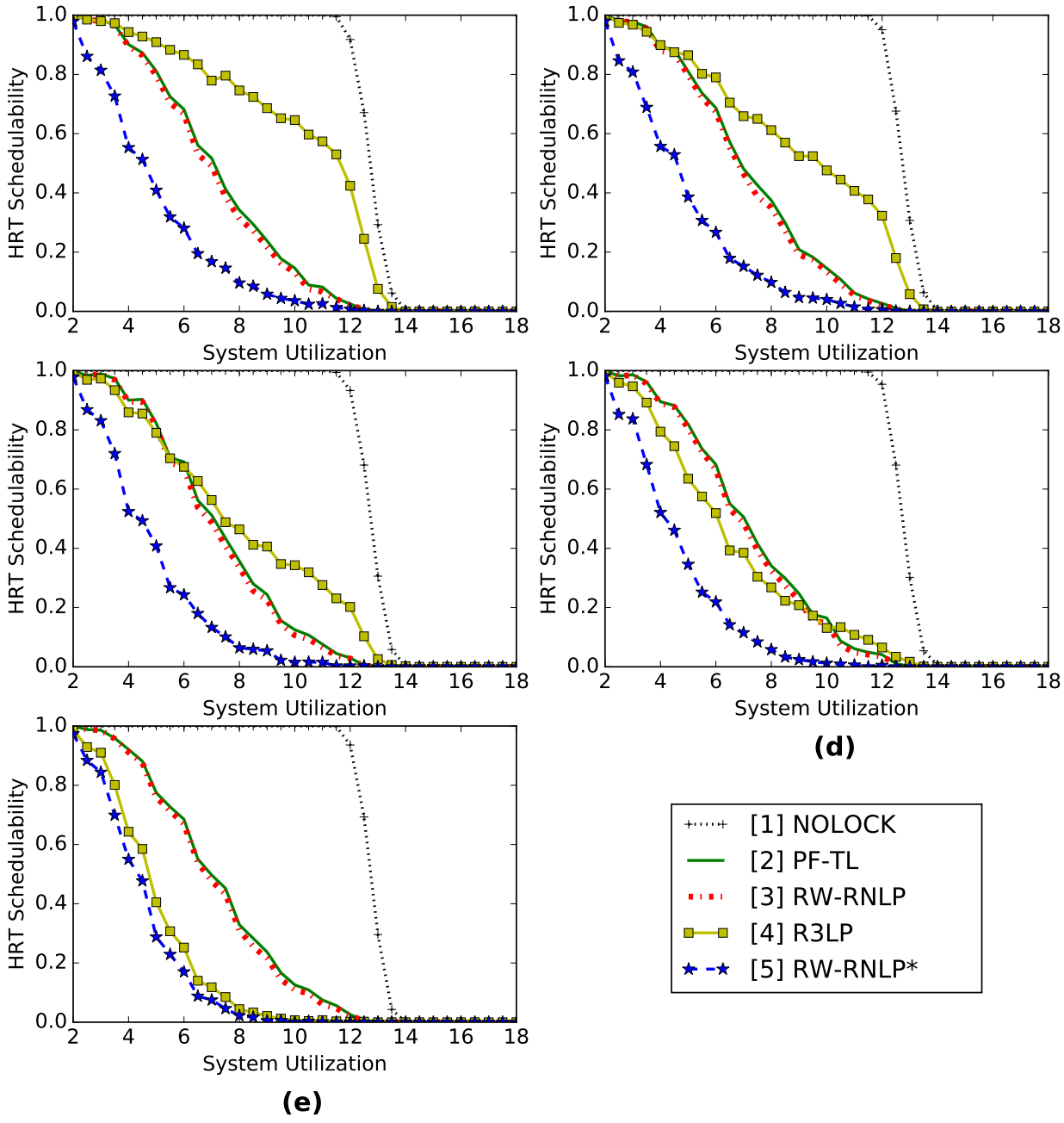



Figure 707: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, long critical-section lengths, and read probability 0 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

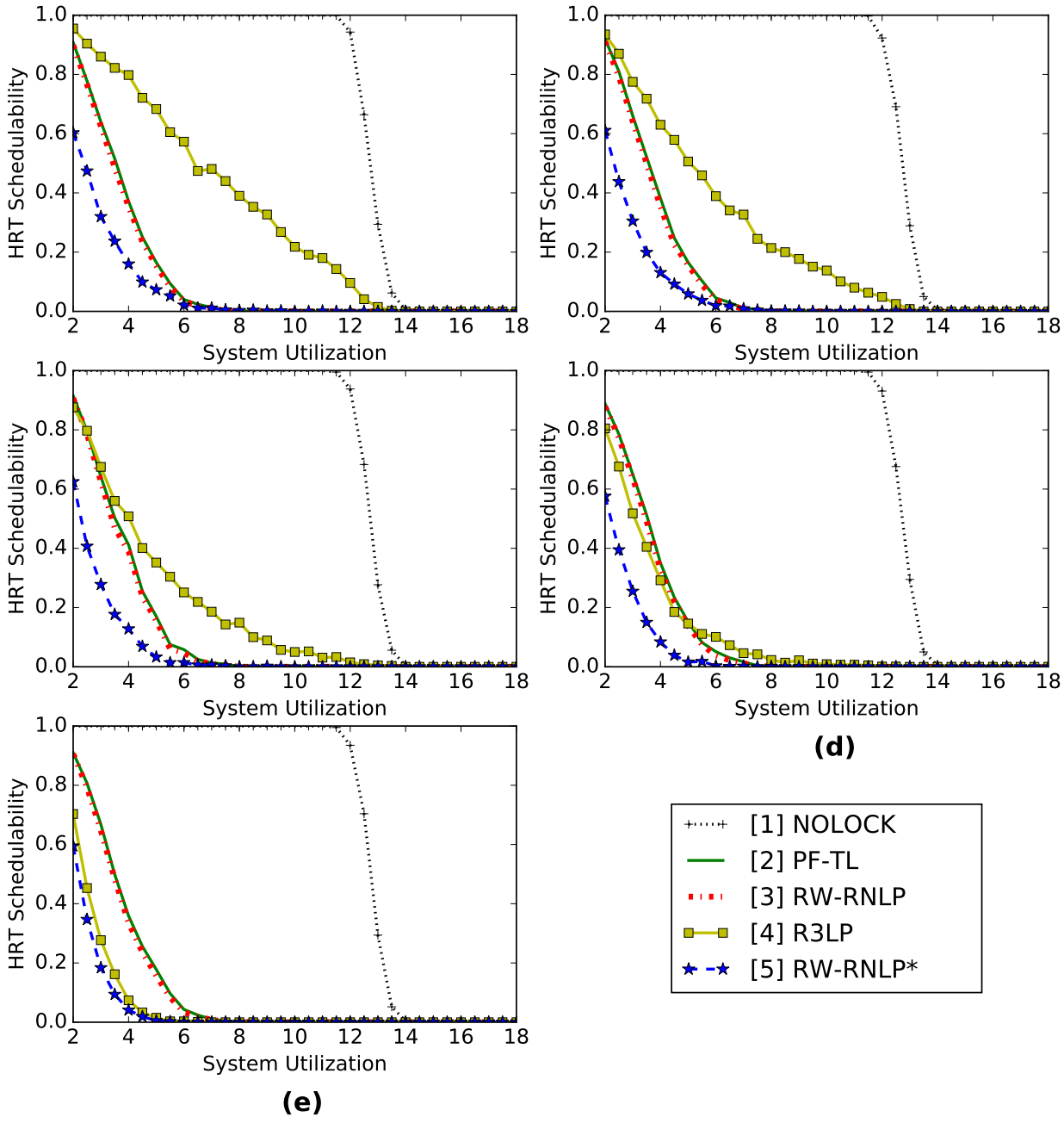



Figure 708: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, long critical-section lengths, and read probability 0 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

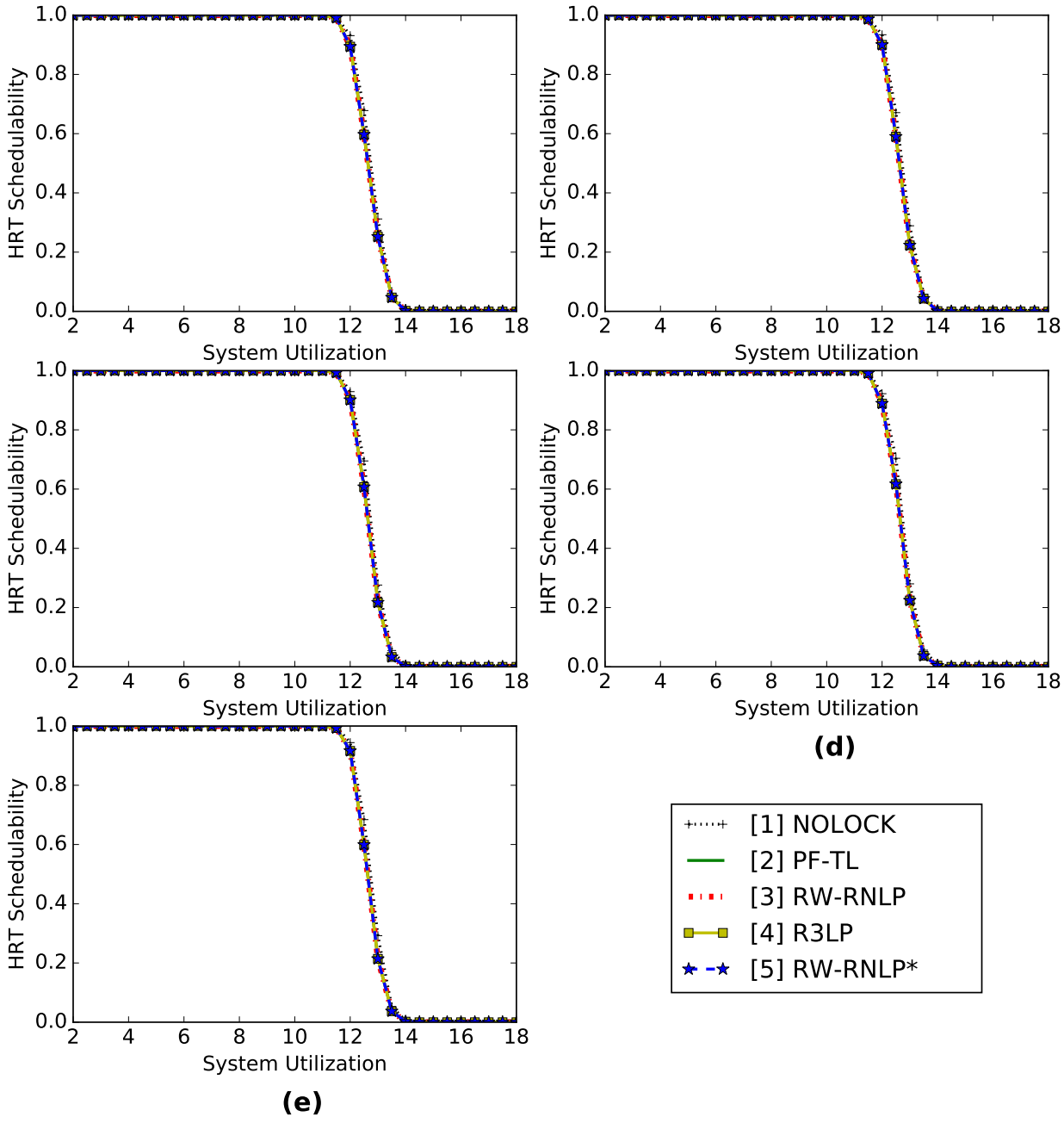



Figure 709: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, long critical-section lengths, and read probability 0.2 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

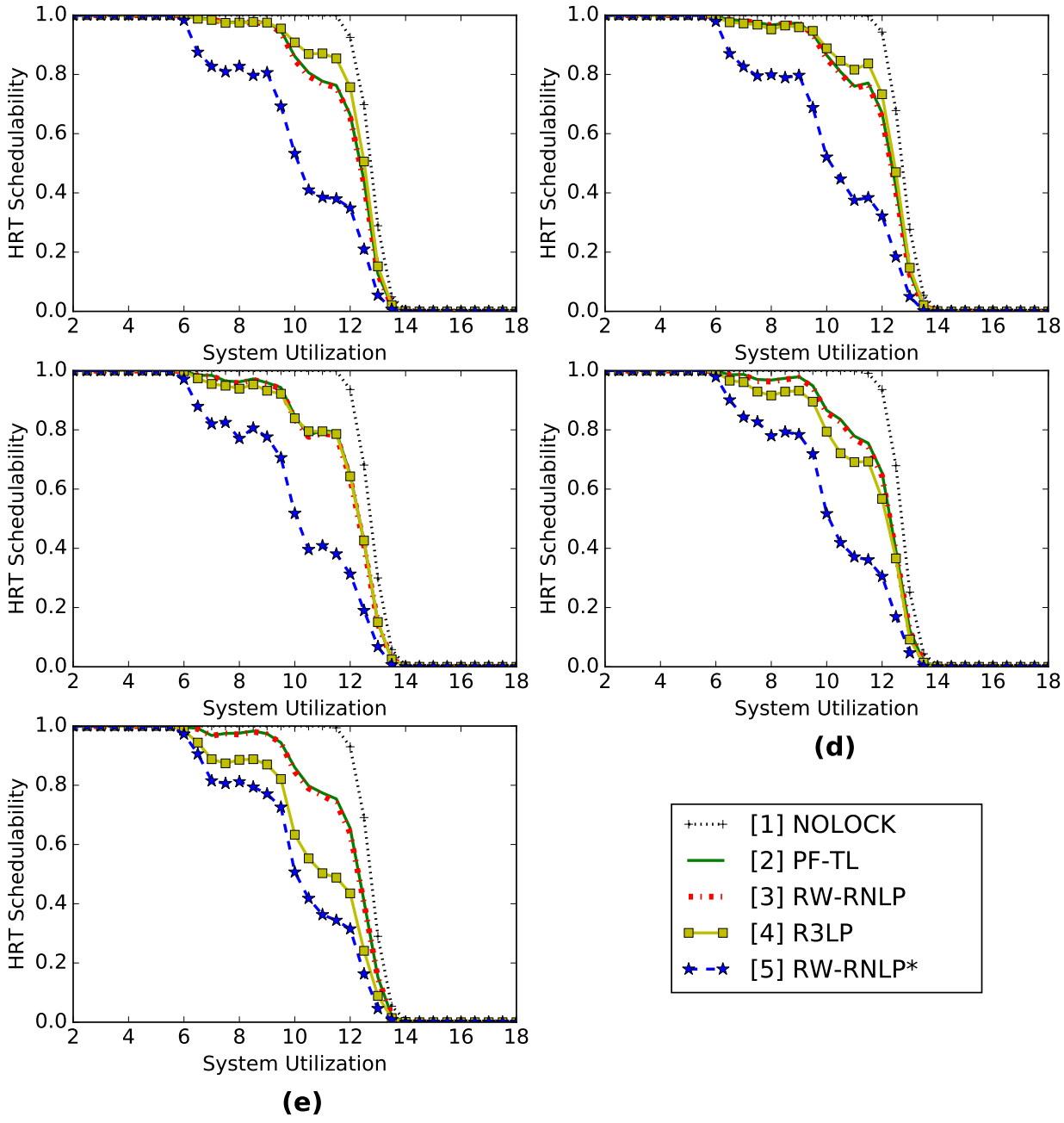



Figure 710: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, long critical-section lengths, and read probability 0.2 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

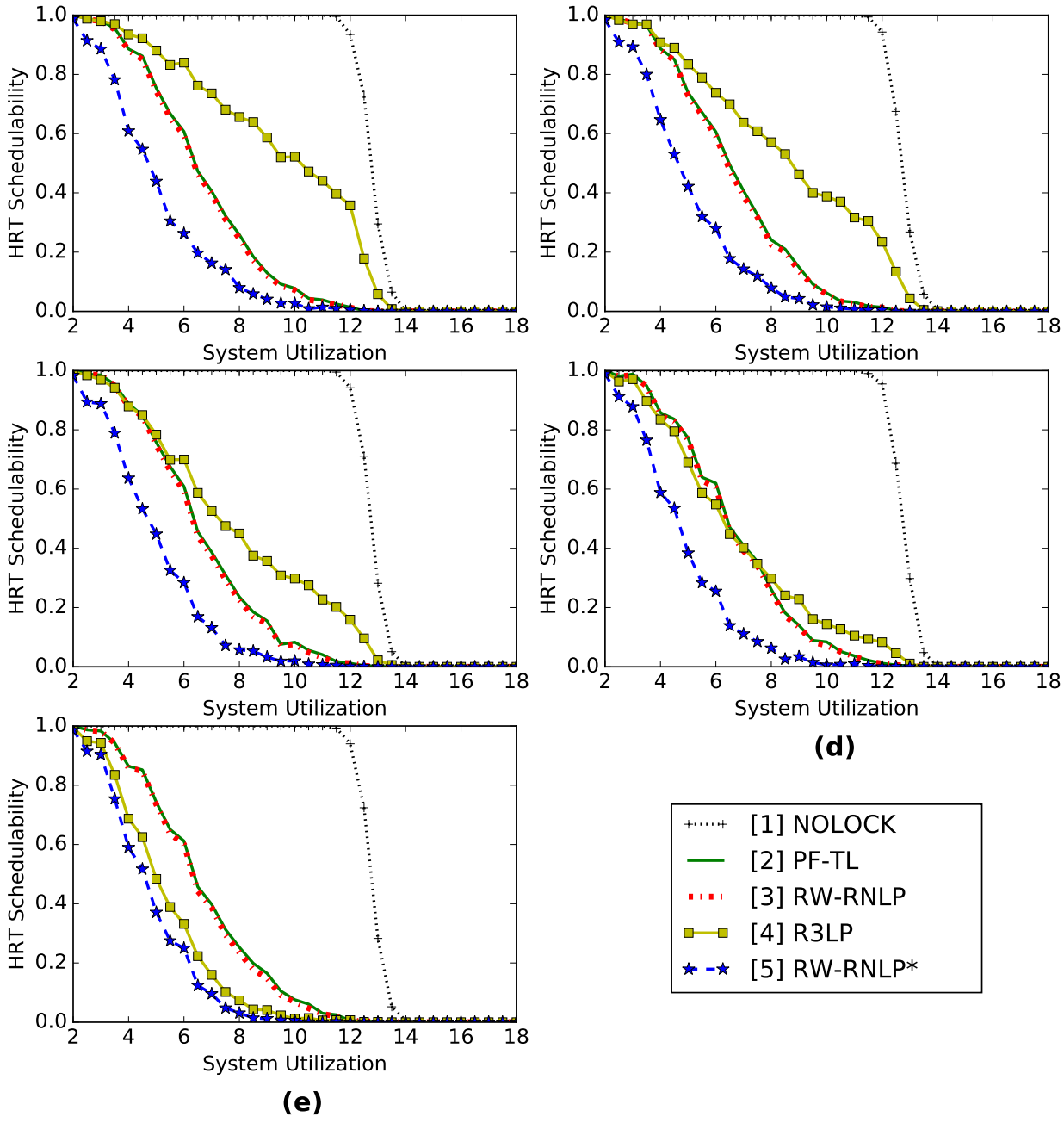



Figure 711: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, long critical-section lengths, and read probability 0.2 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

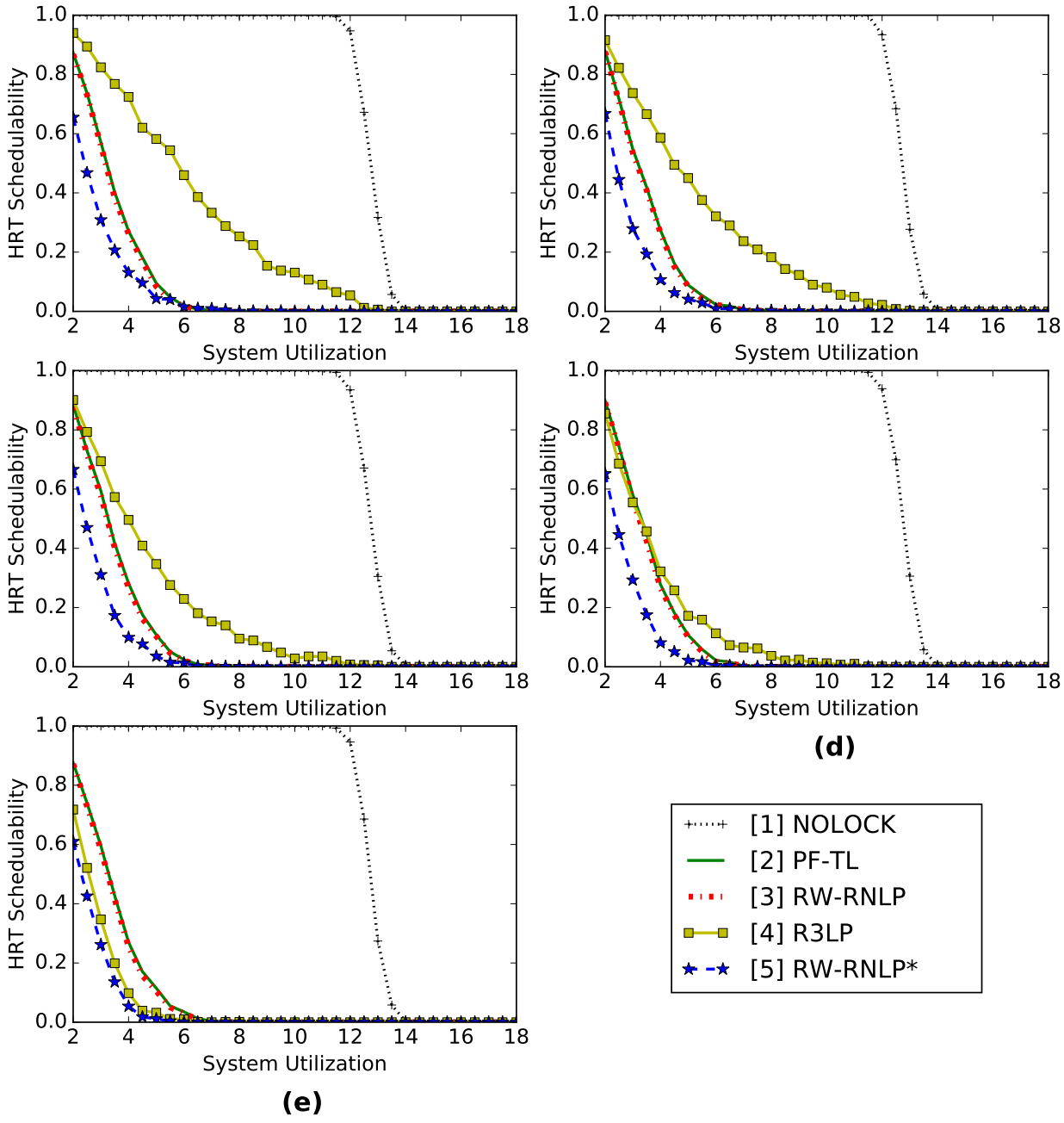



Figure 712: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, long critical-section lengths, and read probability 0.2 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

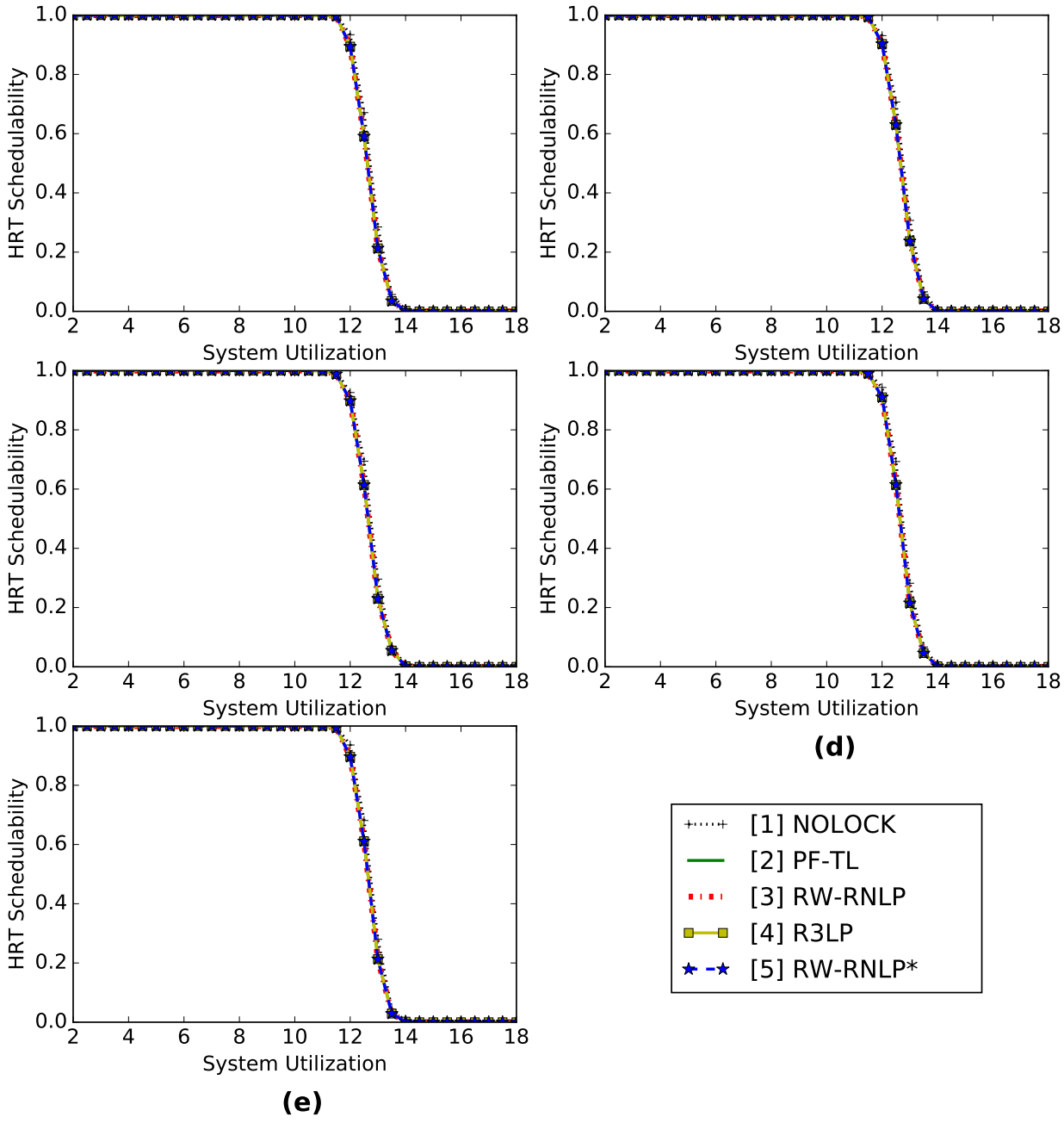



Figure 713: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, long critical-section lengths, and read probability 0.5 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

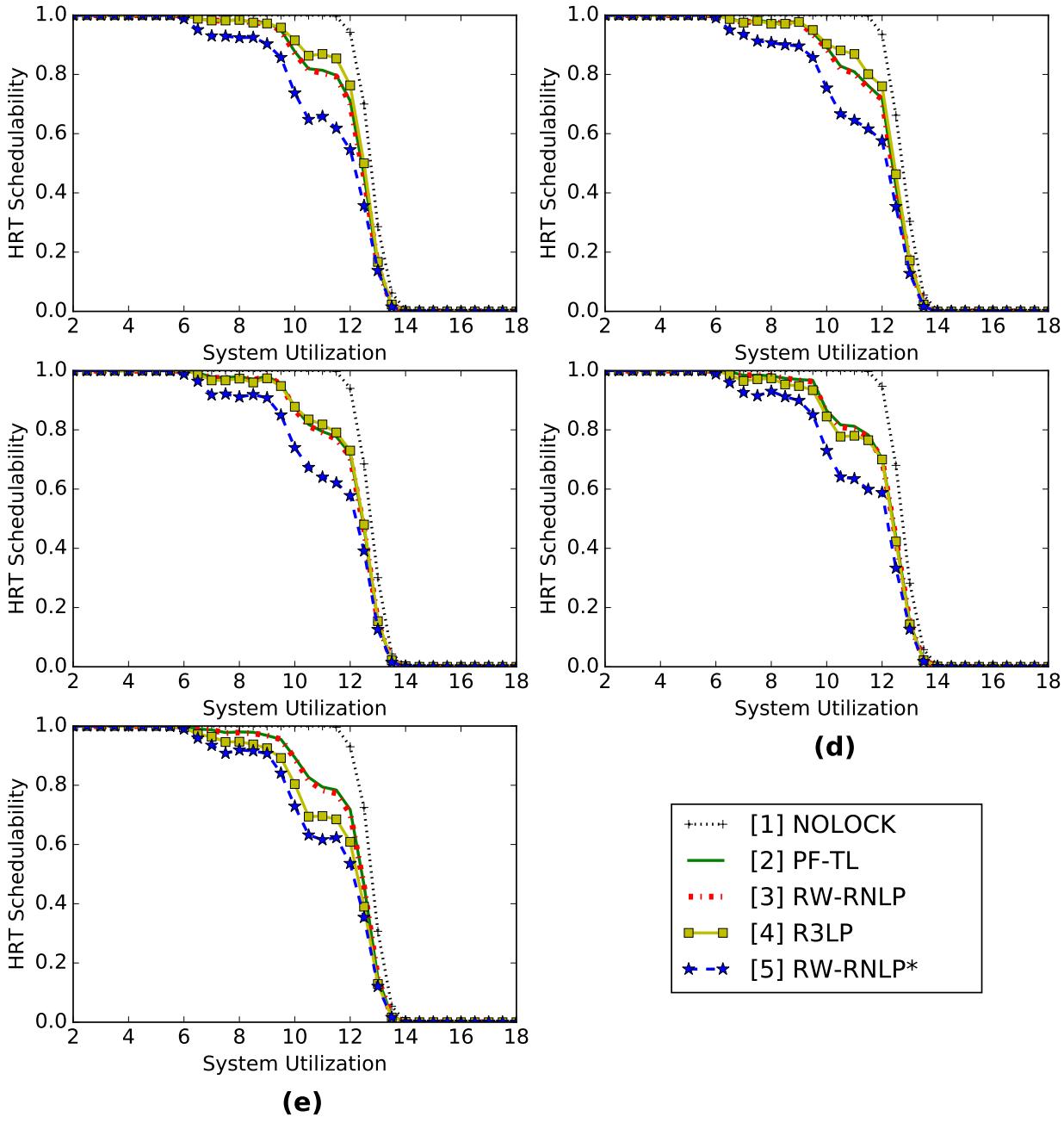



Figure 714: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, long critical-section lengths, and read probability 0.5 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

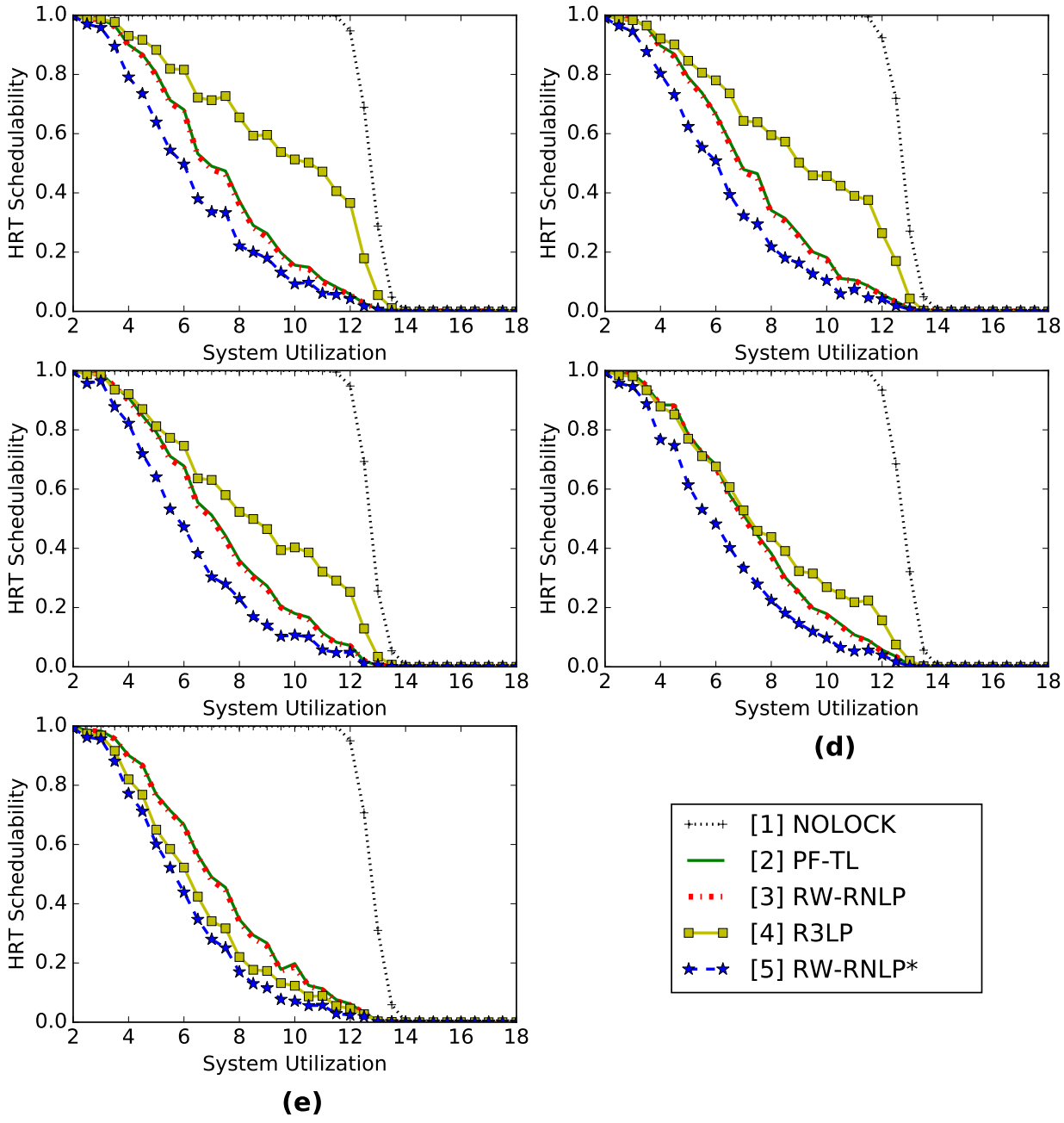



Figure 715: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, long critical-section lengths, and read probability 0.5 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

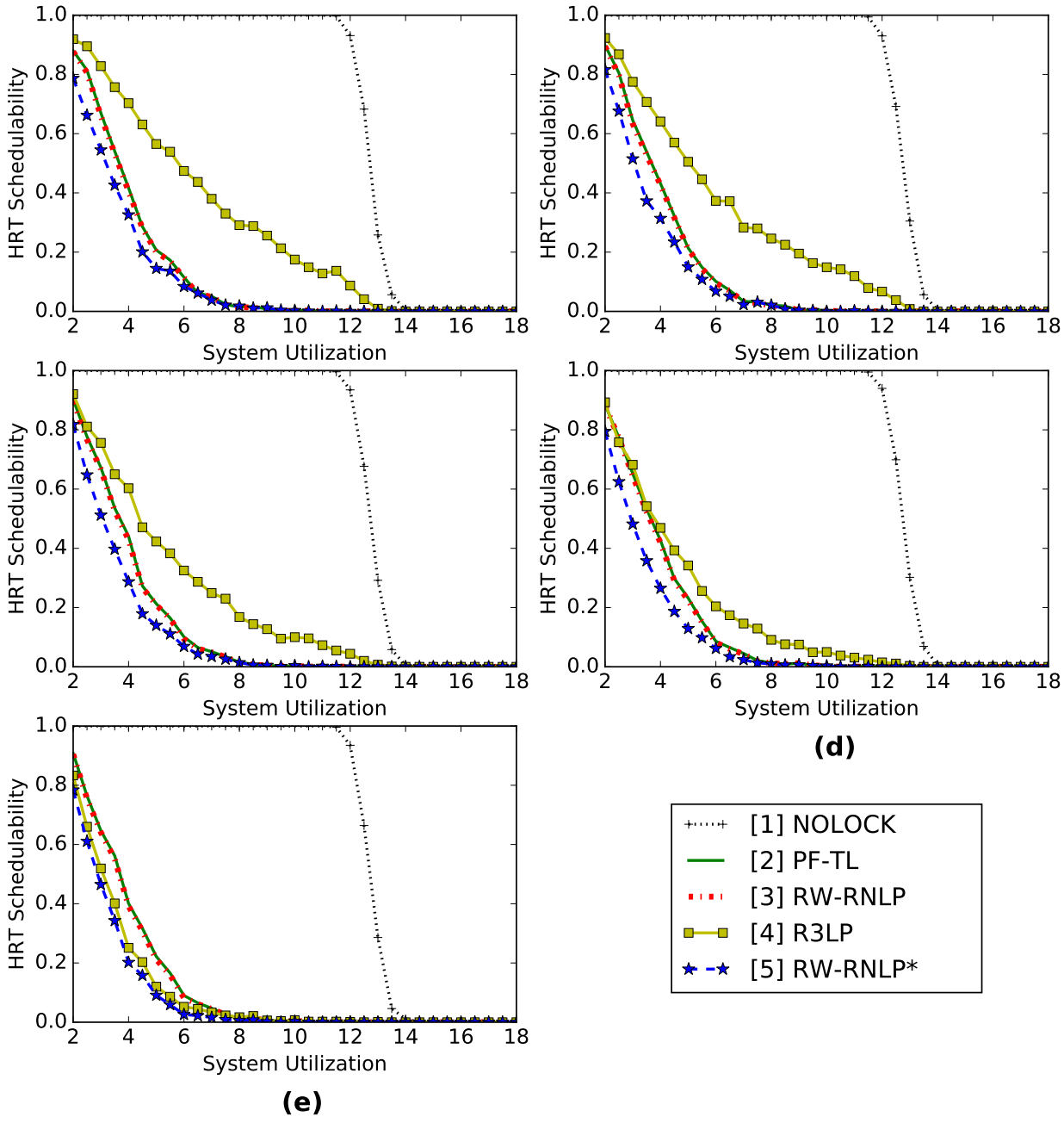



Figure 716: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, long critical-section lengths, and read probability 0.5 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

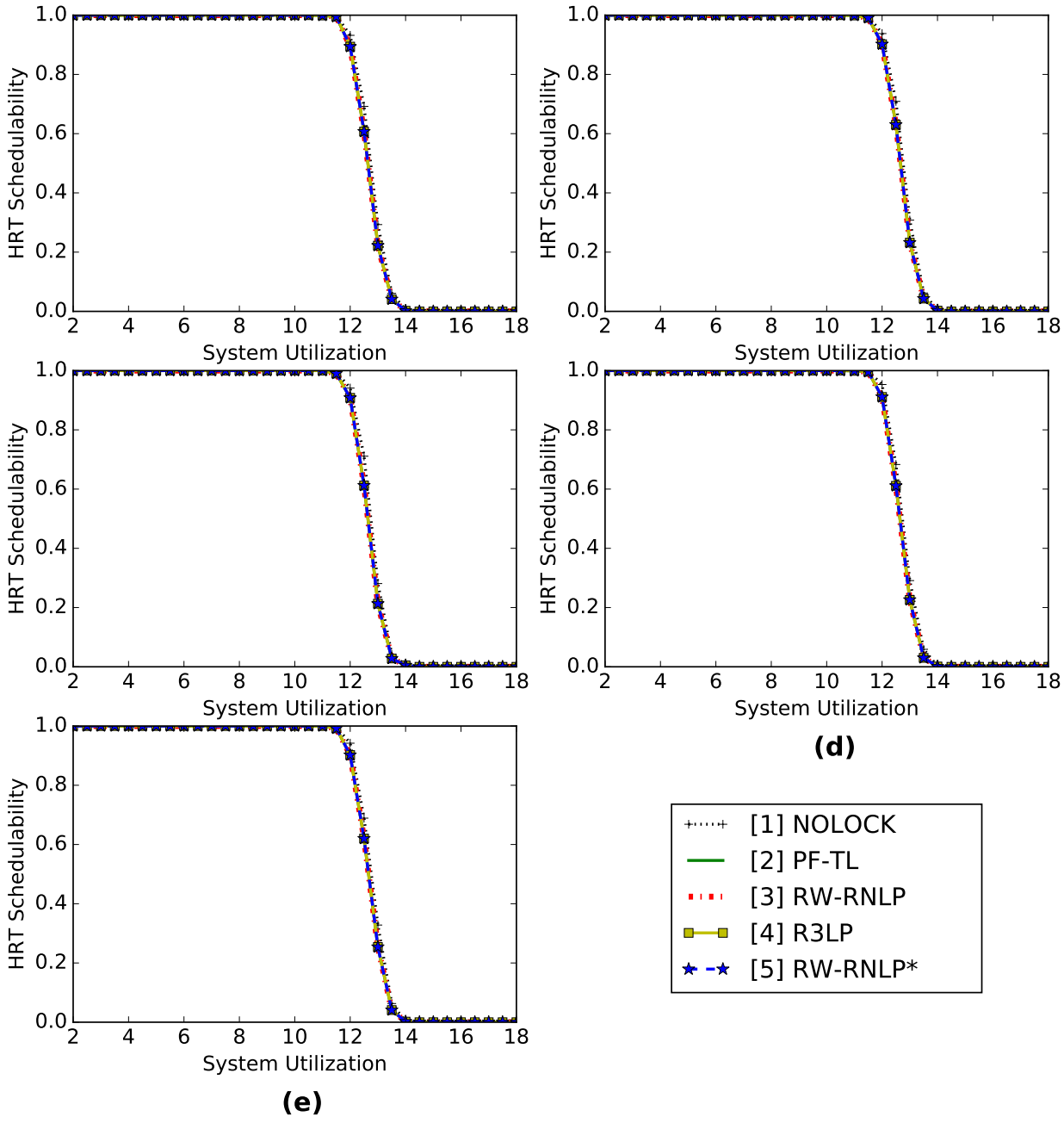



Figure 717: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, long critical-section lengths, and read probability 0.8 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

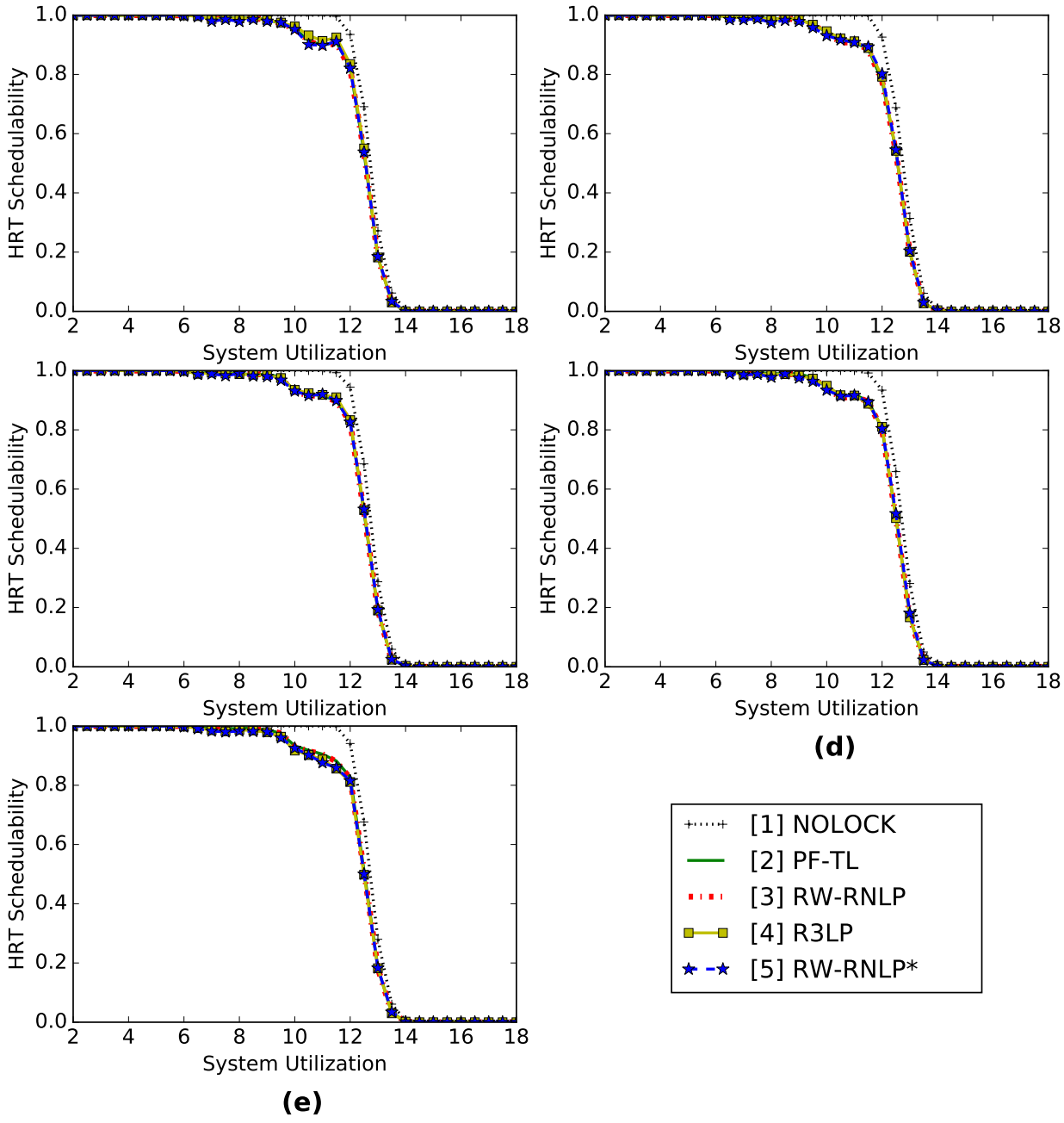



Figure 718: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, long critical-section lengths, and read probability 0.8 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

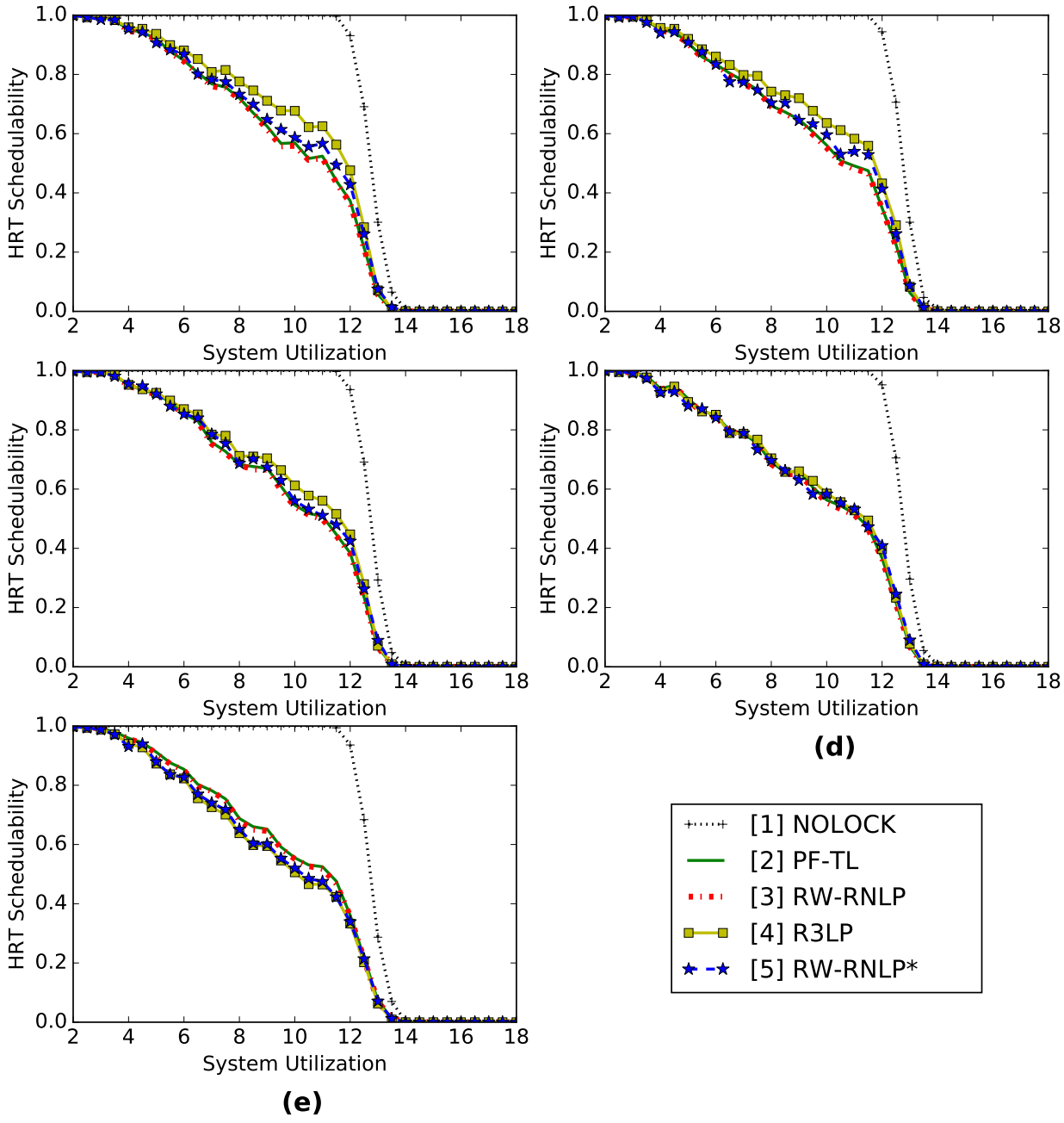



Figure 719: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, long critical-section lengths, and read probability 0.8 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

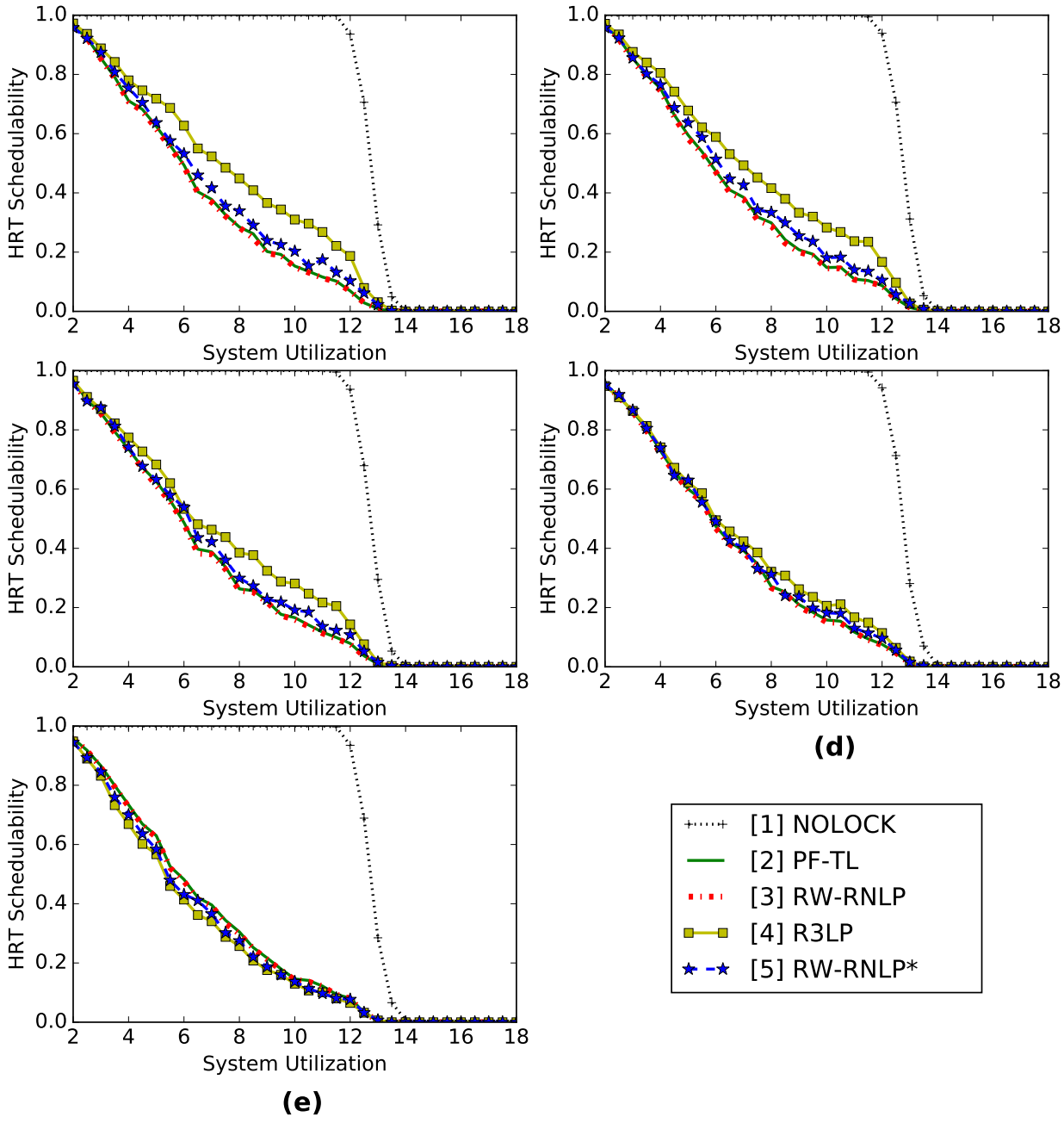



Figure 720: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, moderate periods, long critical-section lengths, and read probability 0.8 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

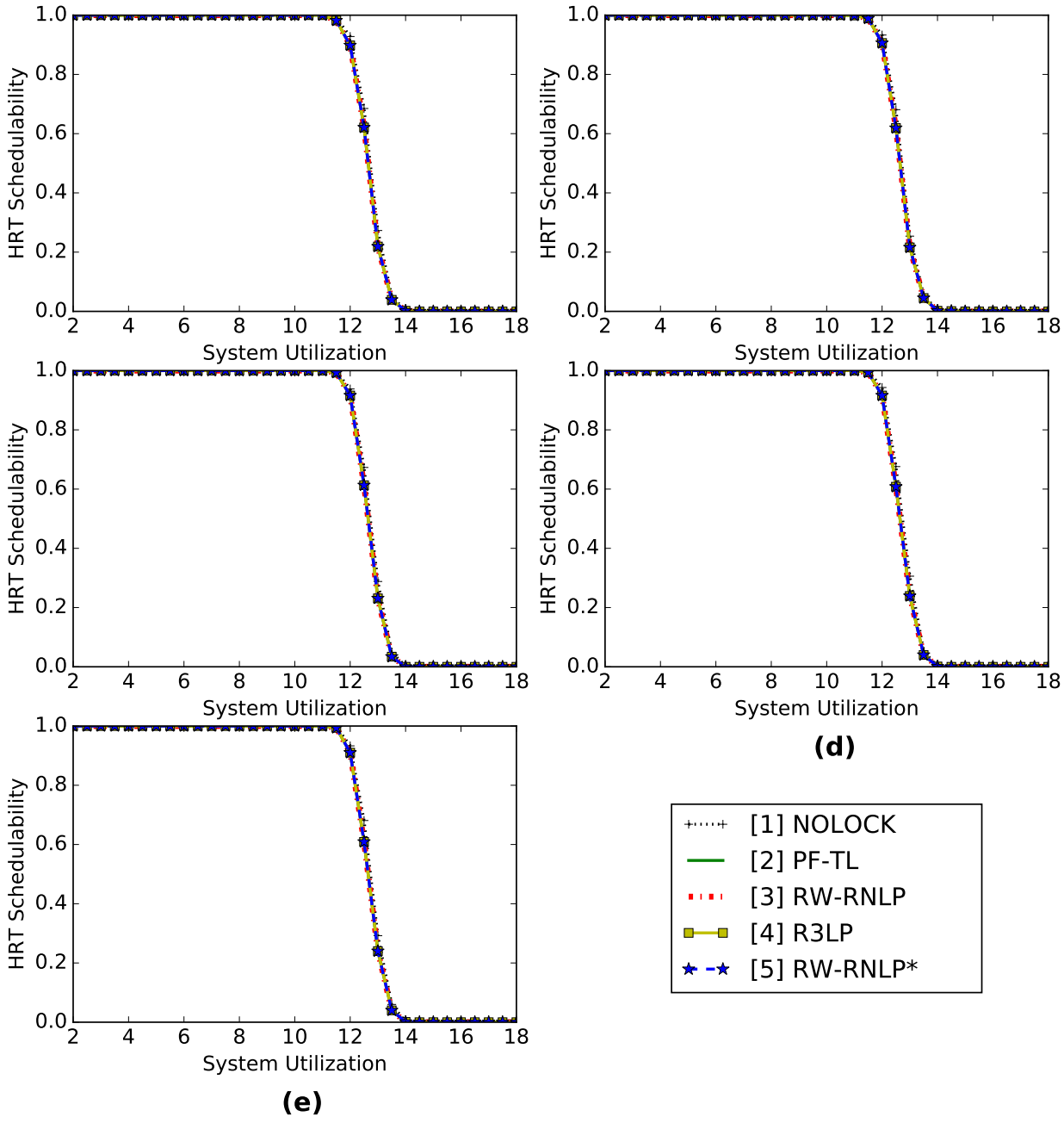



Figure 721: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, long critical-section lengths, and read probability 0 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

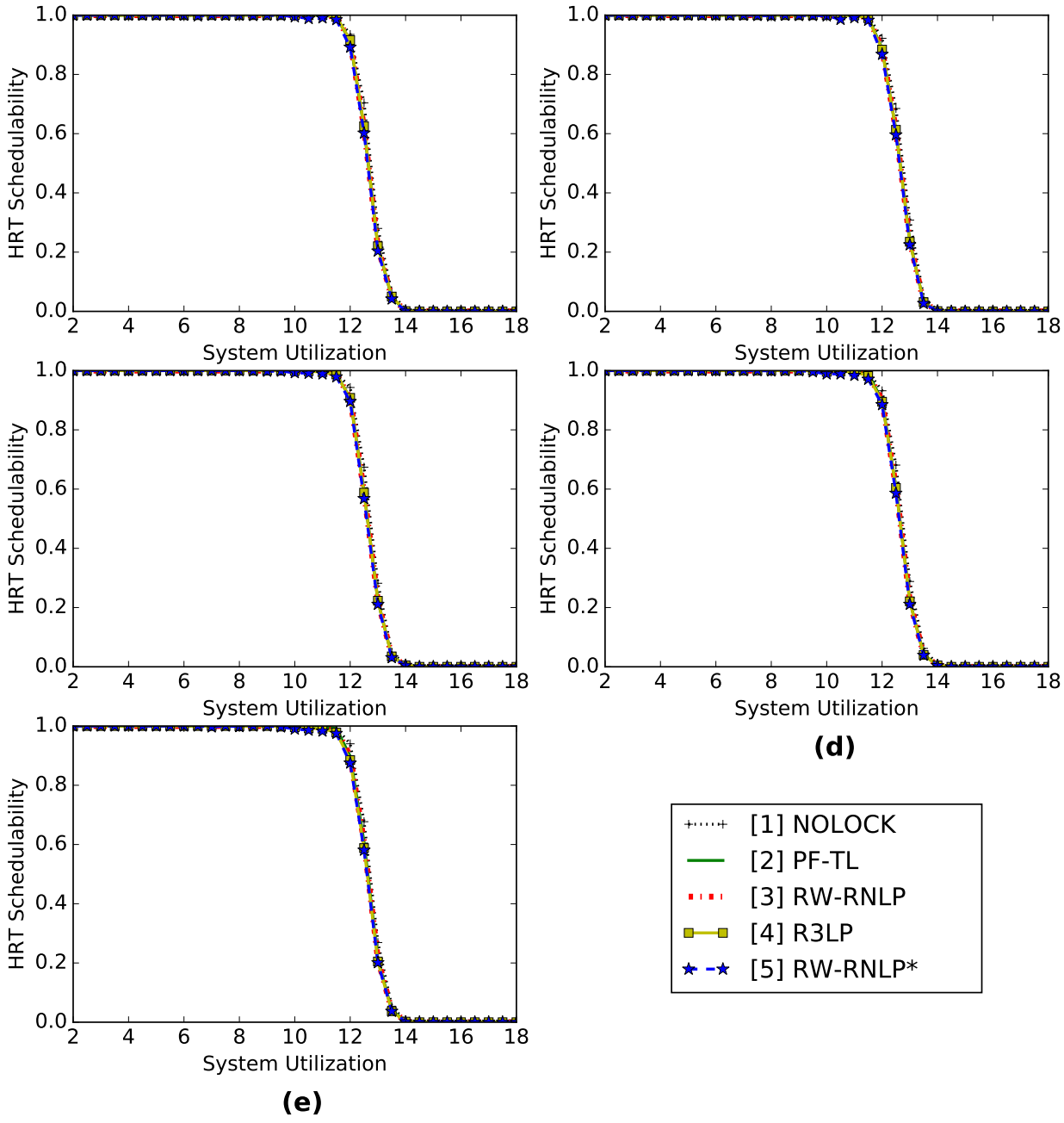



Figure 722: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, long critical-section lengths, and read probability 0 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

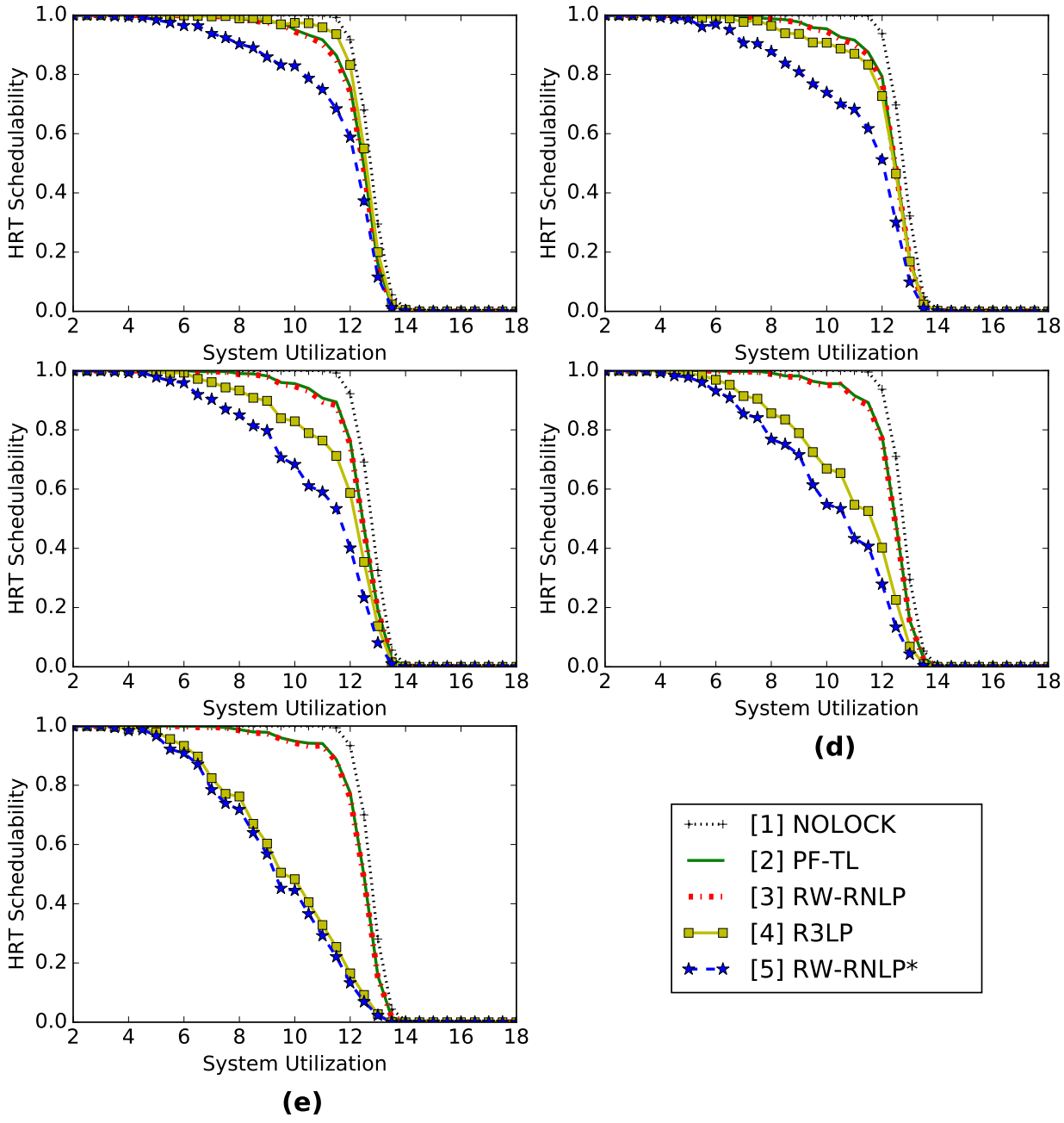



Figure 723: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, long critical-section lengths, and read probability 0 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

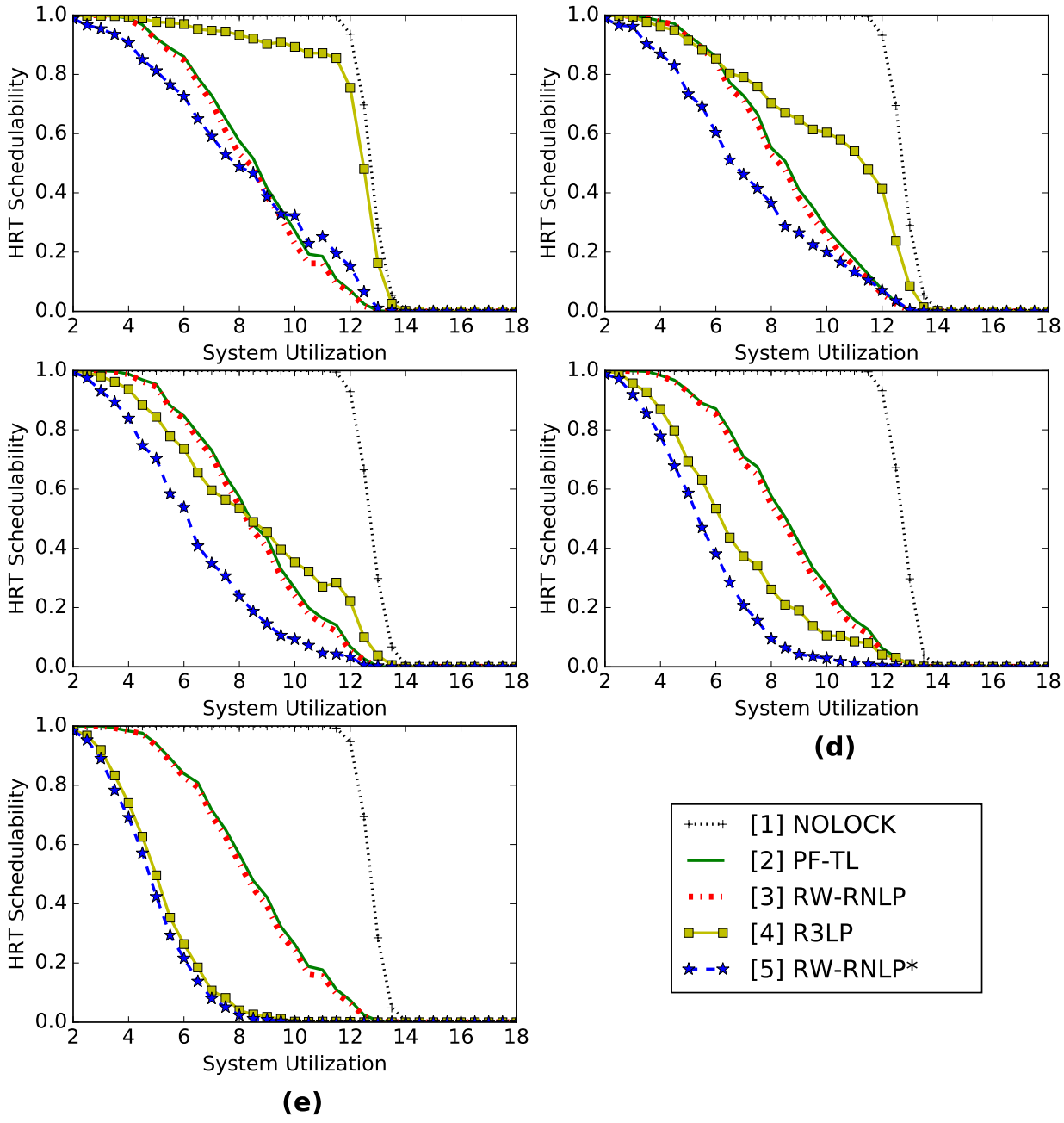



Figure 724: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, long critical-section lengths, and read probability 0 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

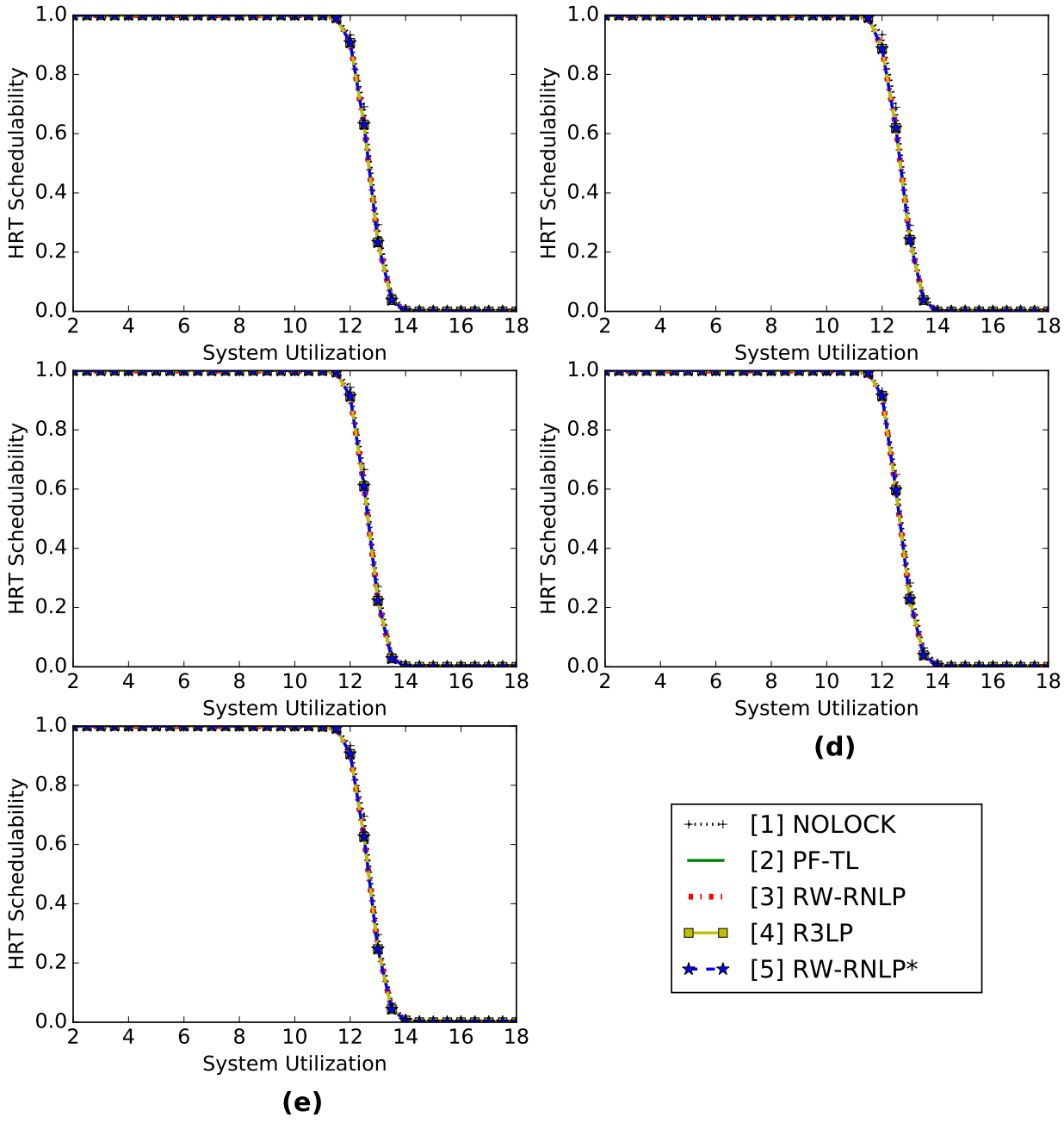



Figure 725: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, long critical-section lengths, and read probability 0.2 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

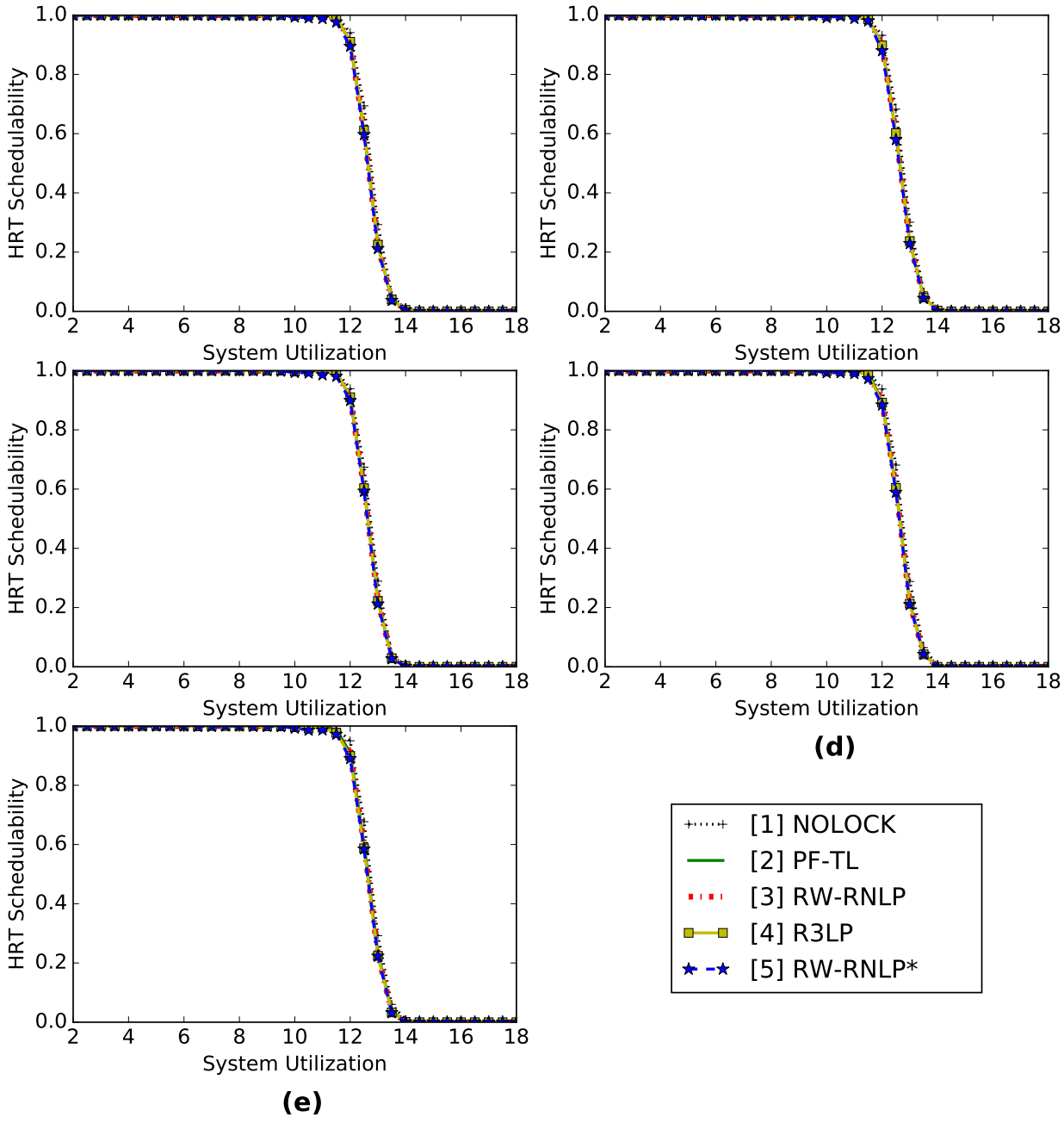



Figure 726: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, long critical-section lengths, and read probability 0.2 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

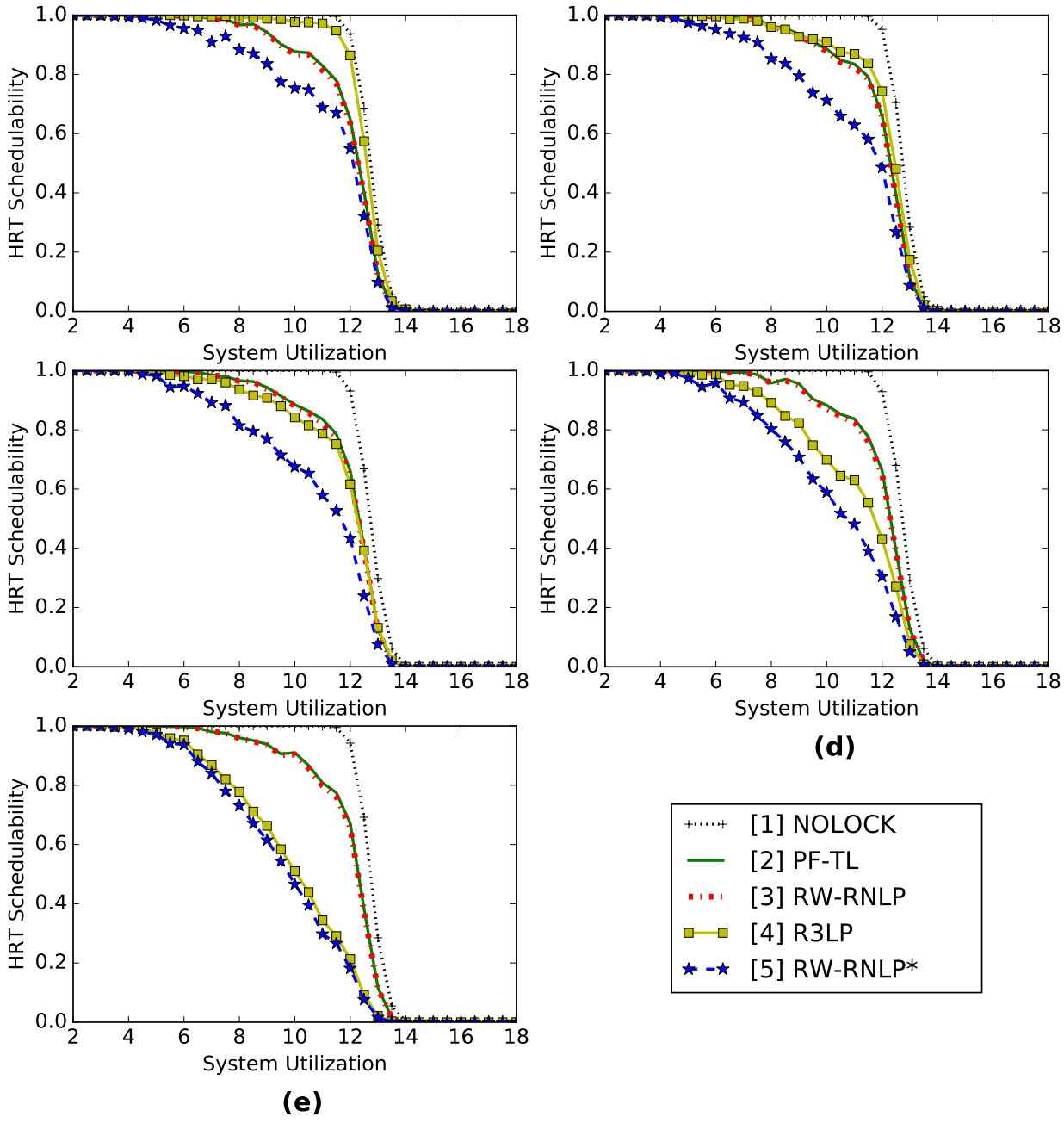



Figure 727: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, long critical-section lengths, and read probability 0.2 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

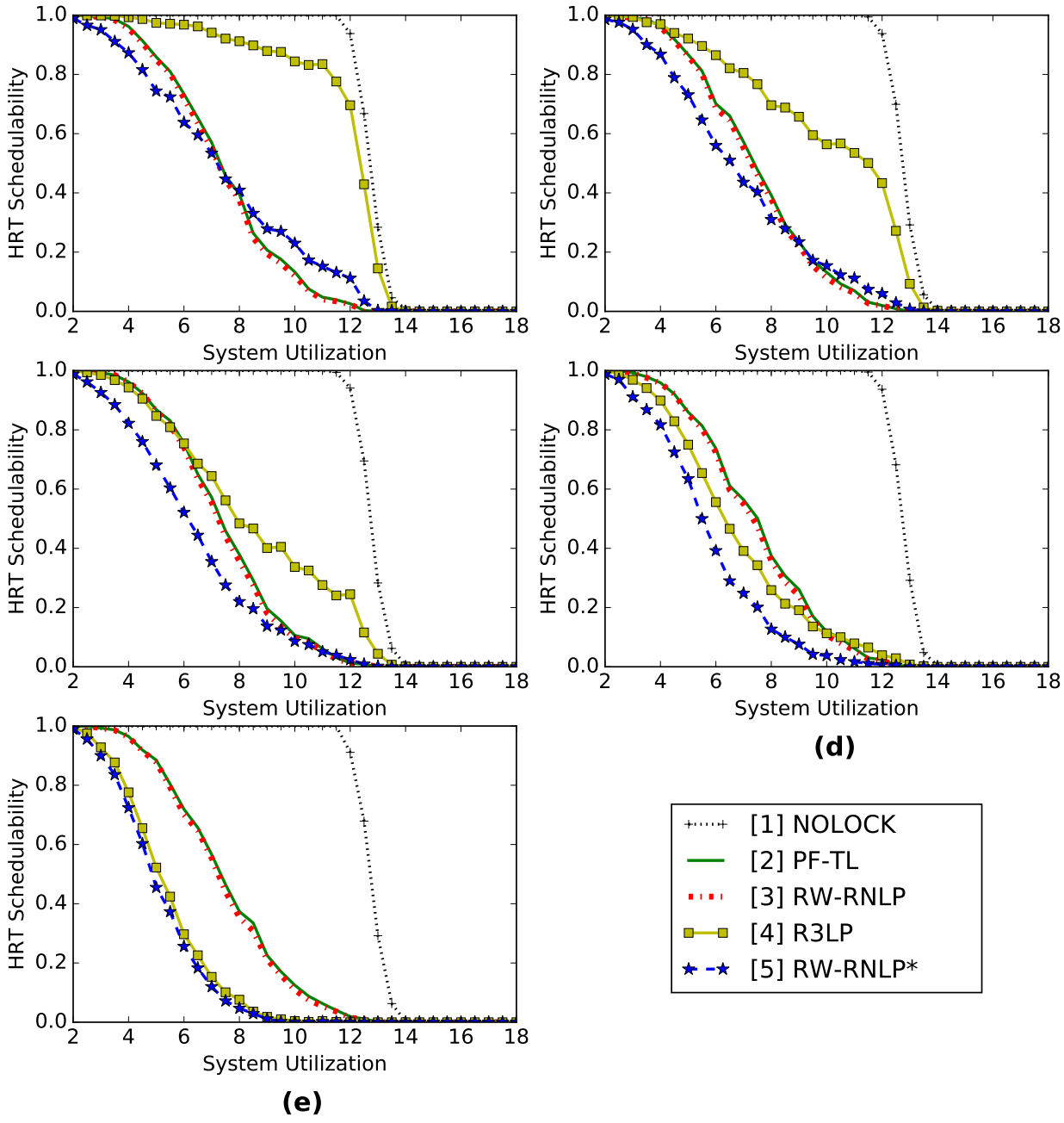



Figure 728: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, long critical-section lengths, and read probability 0.2 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

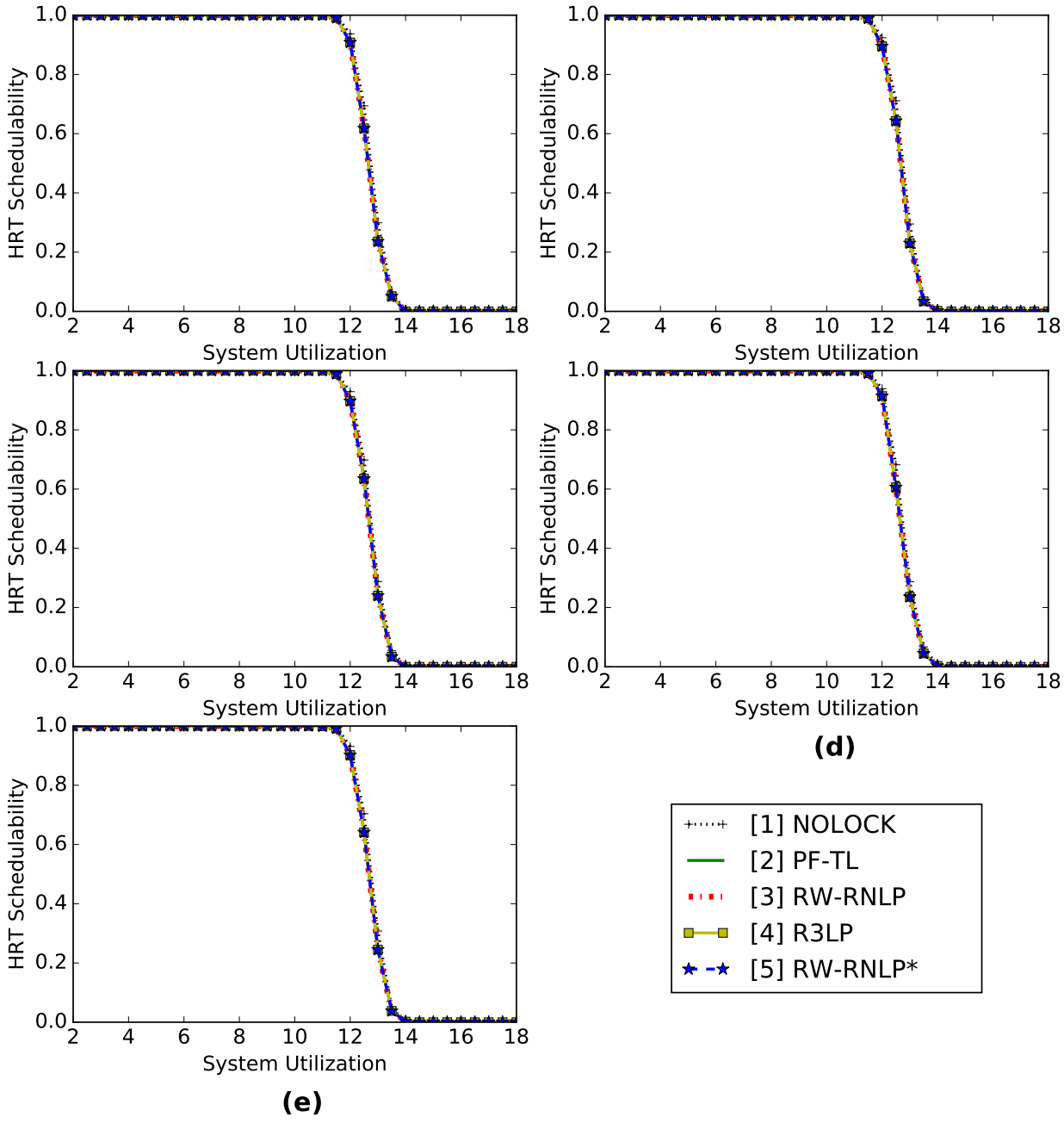



Figure 729: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, long critical-section lengths, and read probability 0.5 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

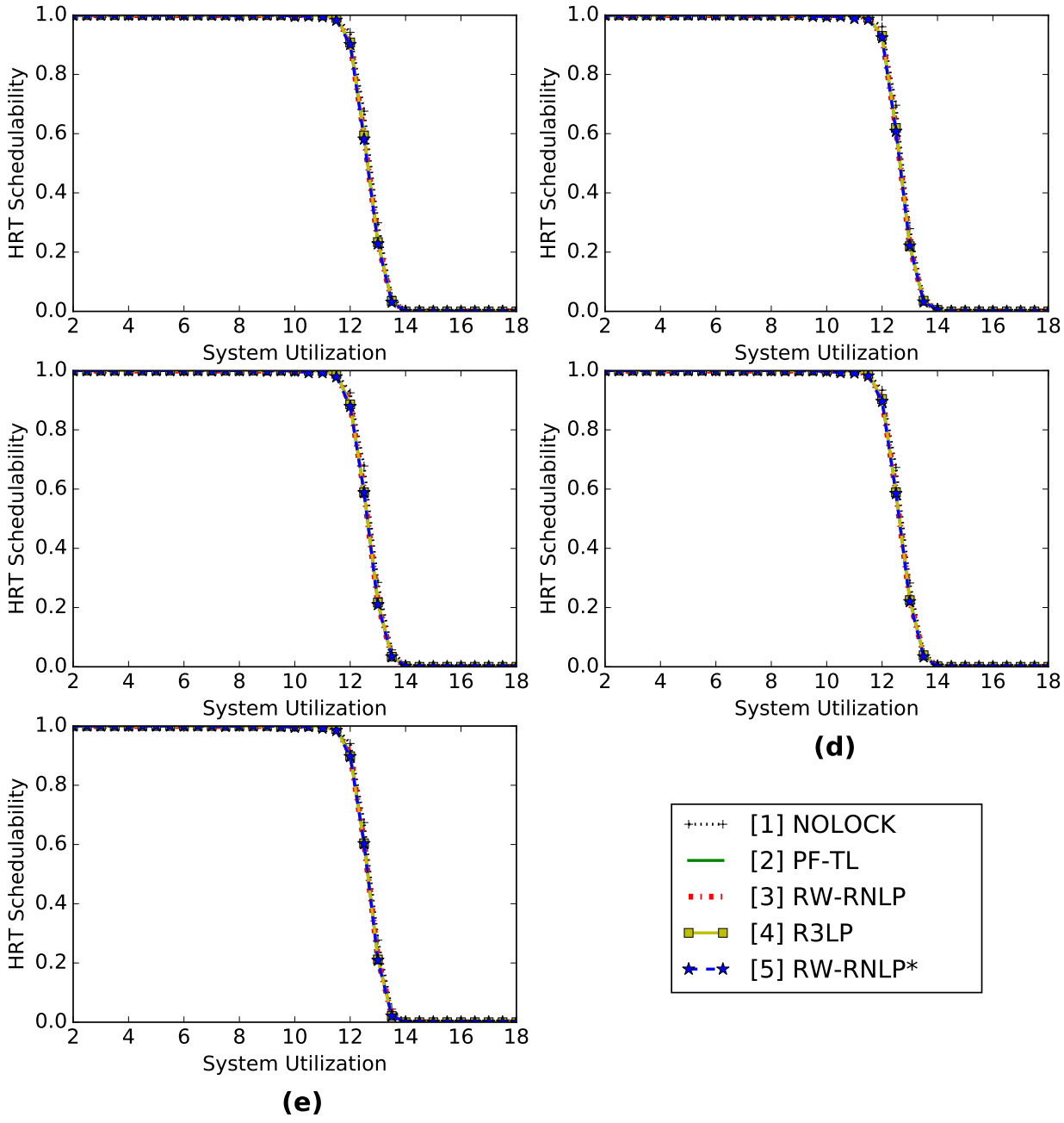



Figure 730: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, long critical-section lengths, and read probability 0.5 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

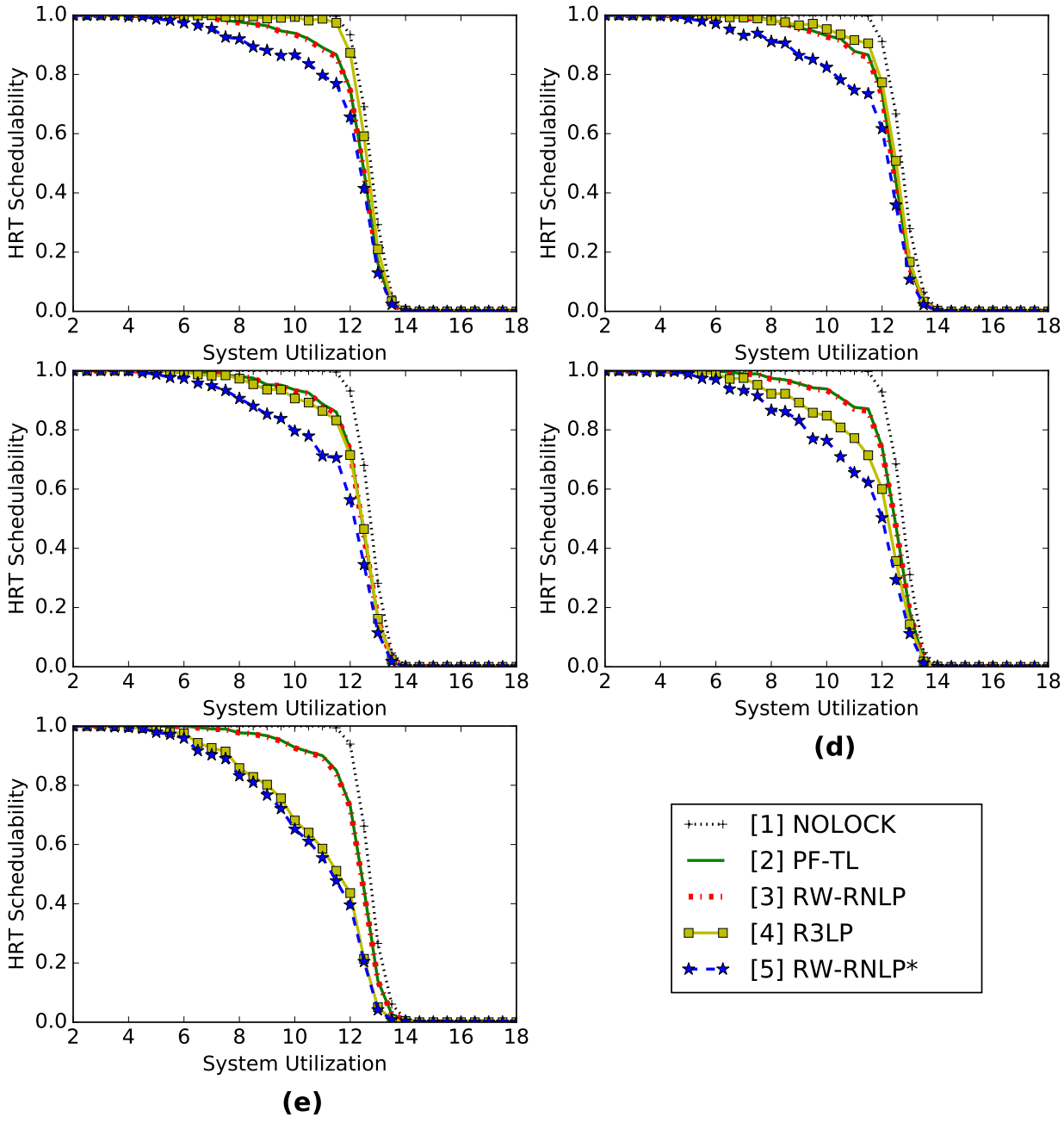



Figure 731: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, long critical-section lengths, and read probability 0.5 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

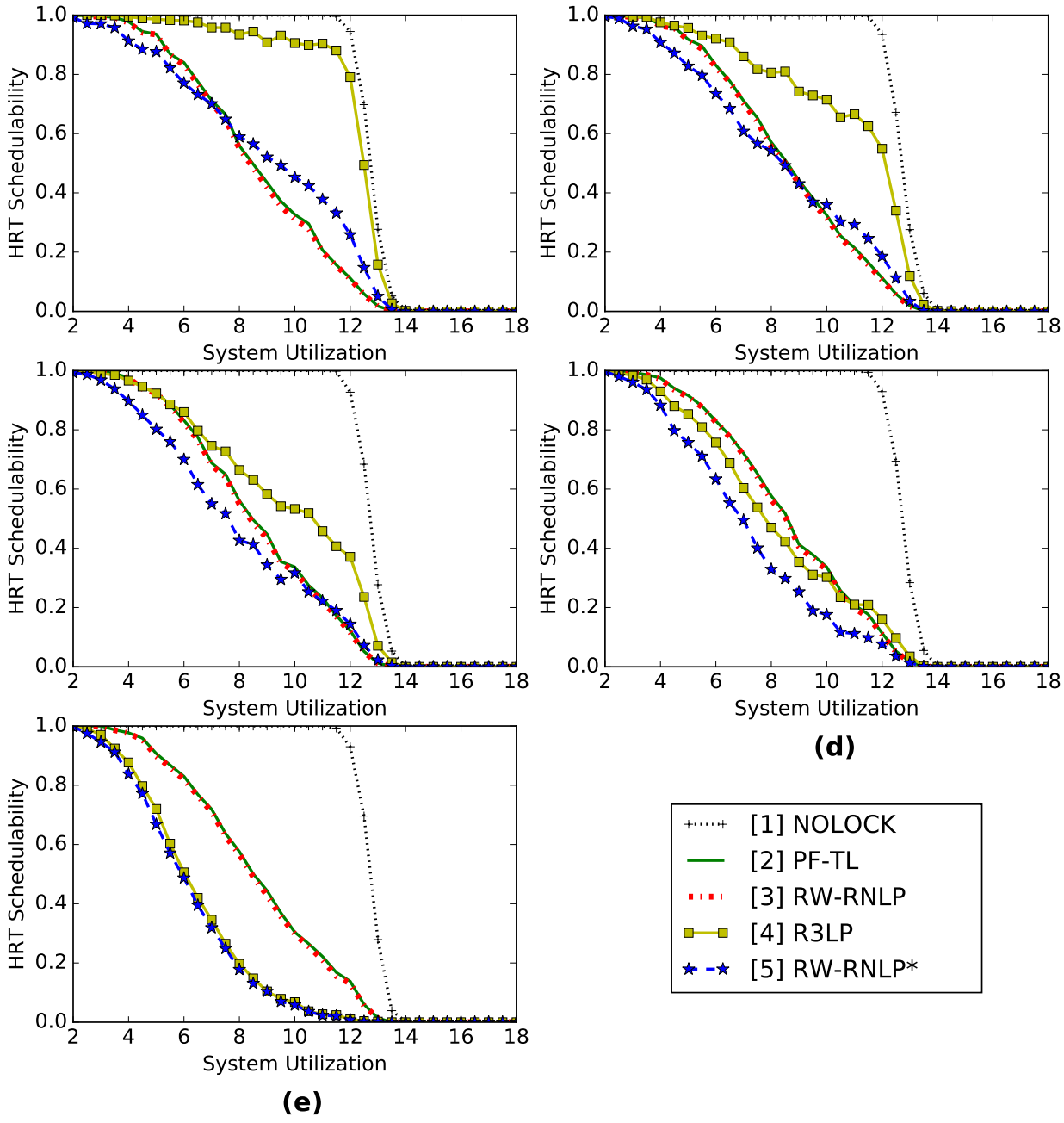



Figure 732: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, long critical-section lengths, and read probability 0.5 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

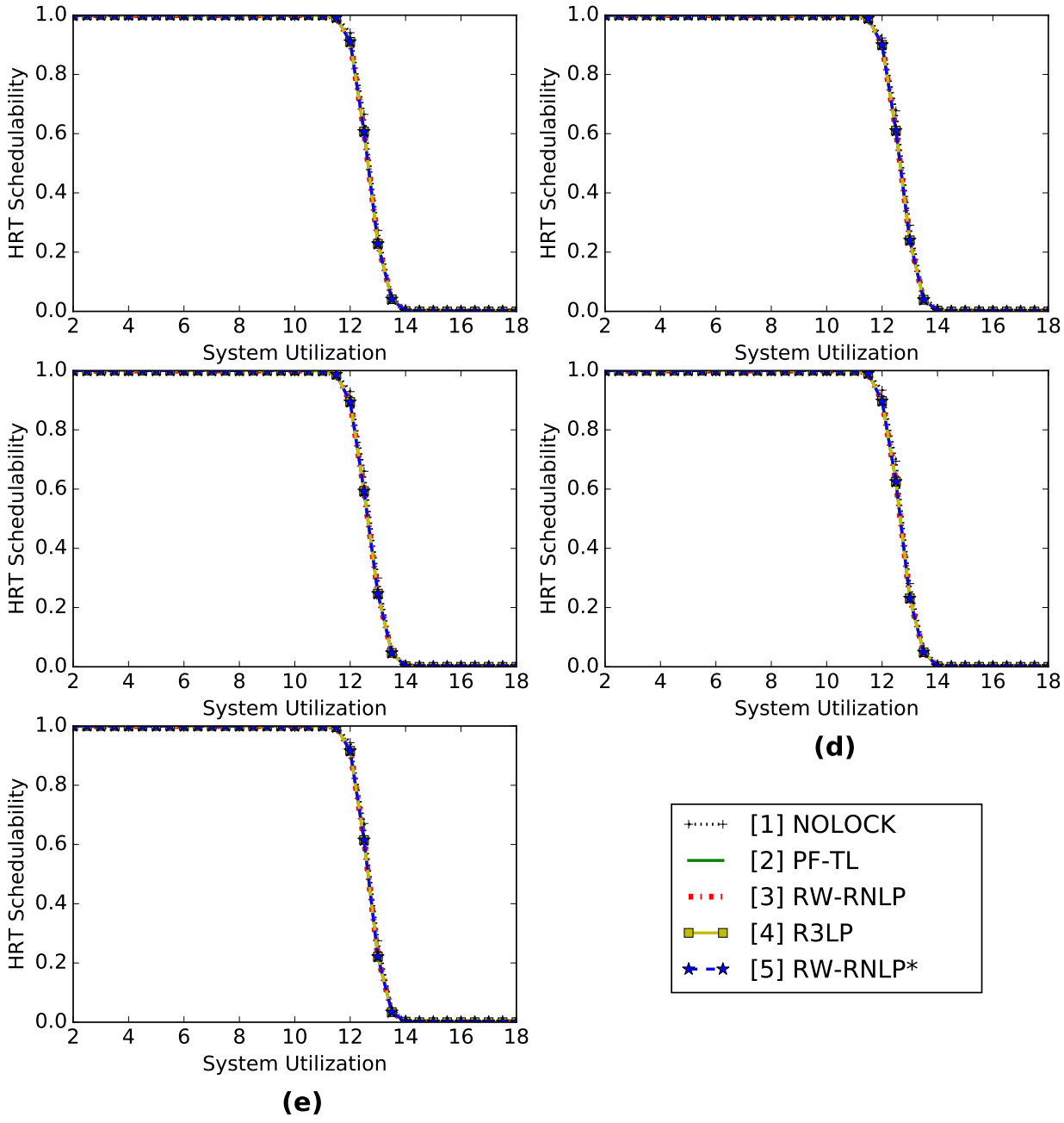



Figure 733: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, long critical-section lengths, and read probability 0.8 in which a task has probability 0.1 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

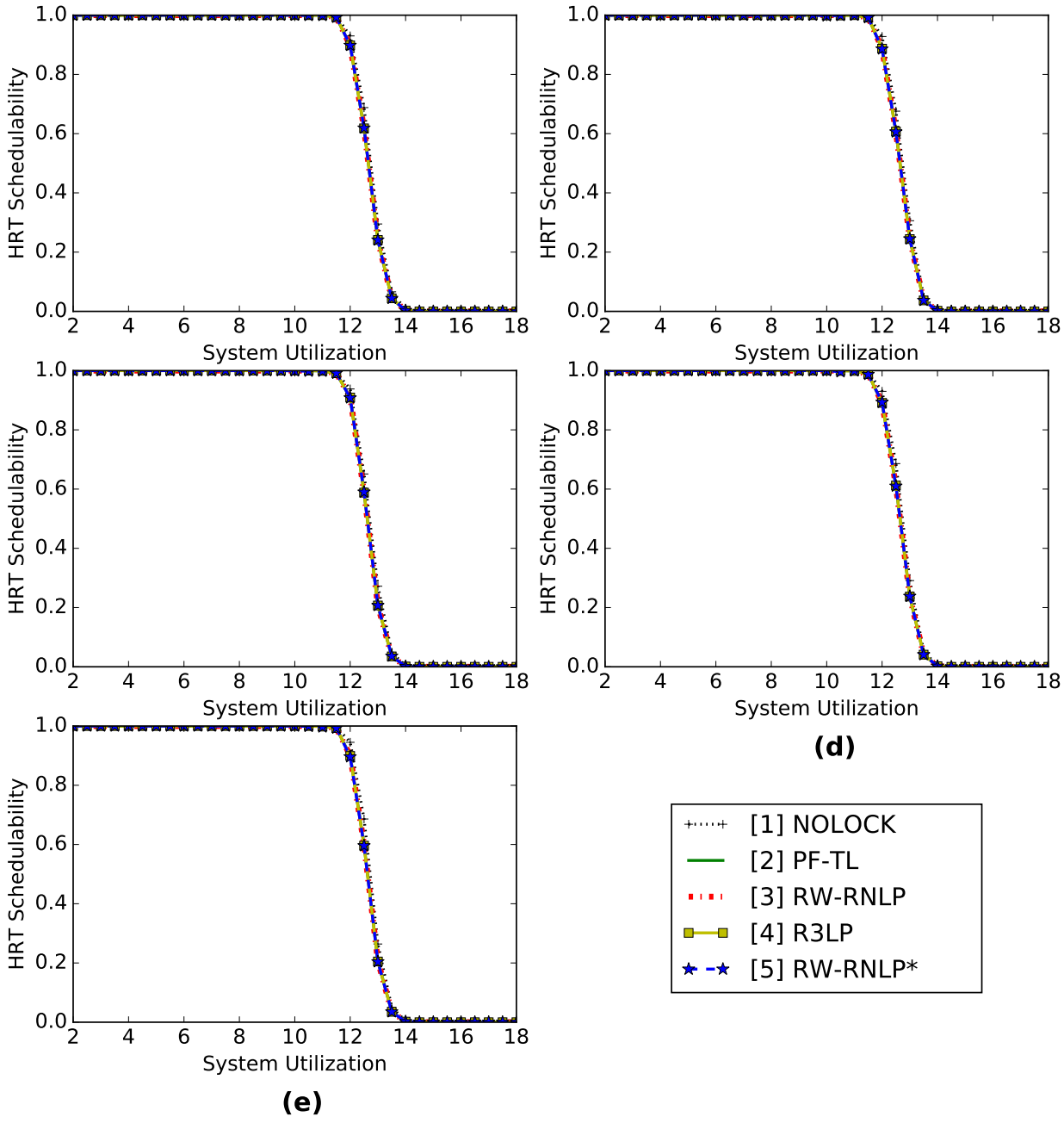



Figure 734: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, long critical-section lengths, and read probability 0.8 in which a task has probability 0.2 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

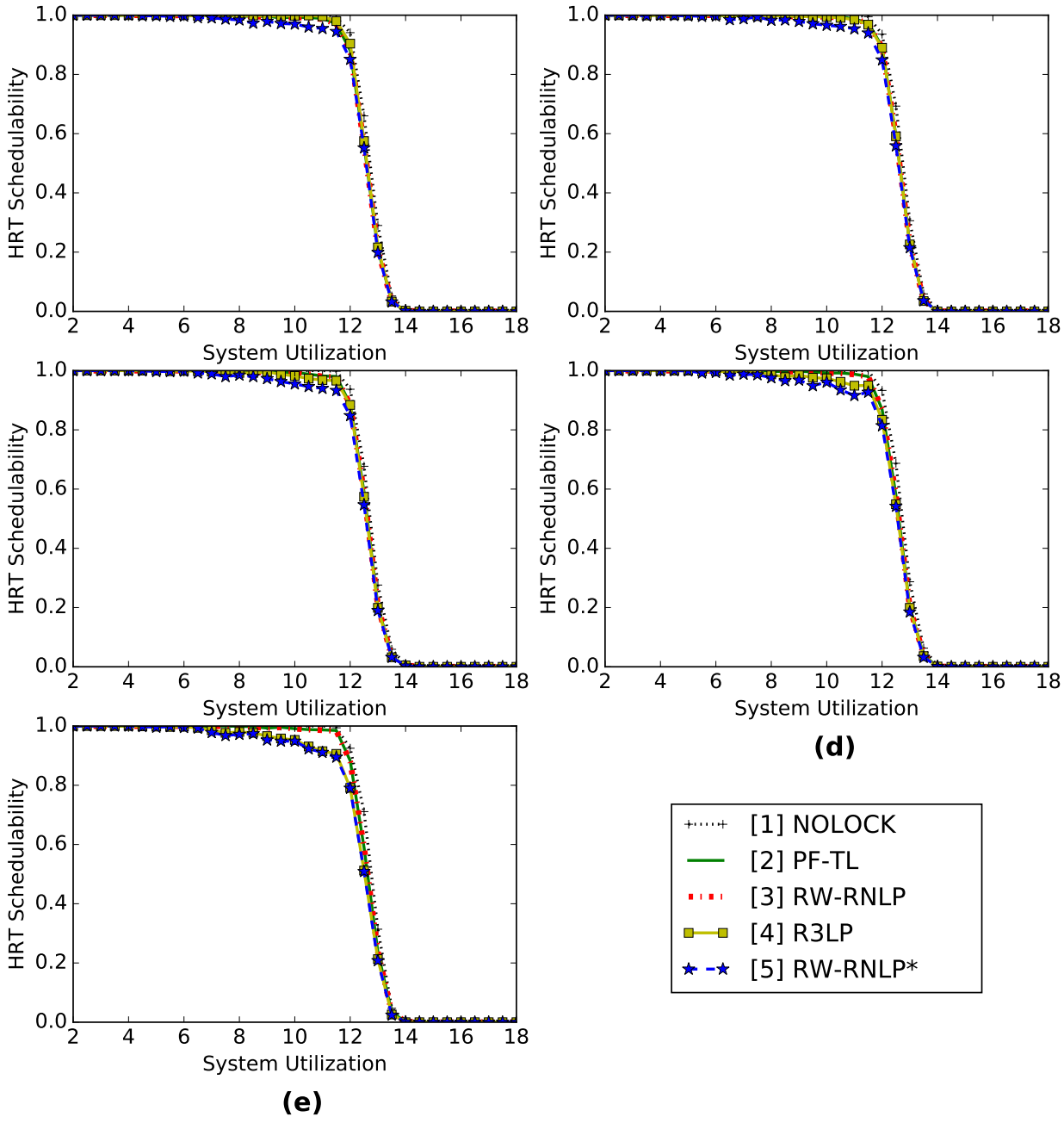



Figure 735: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, long critical-section lengths, and read probability 0.8 in which a task has probability 0.5 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.

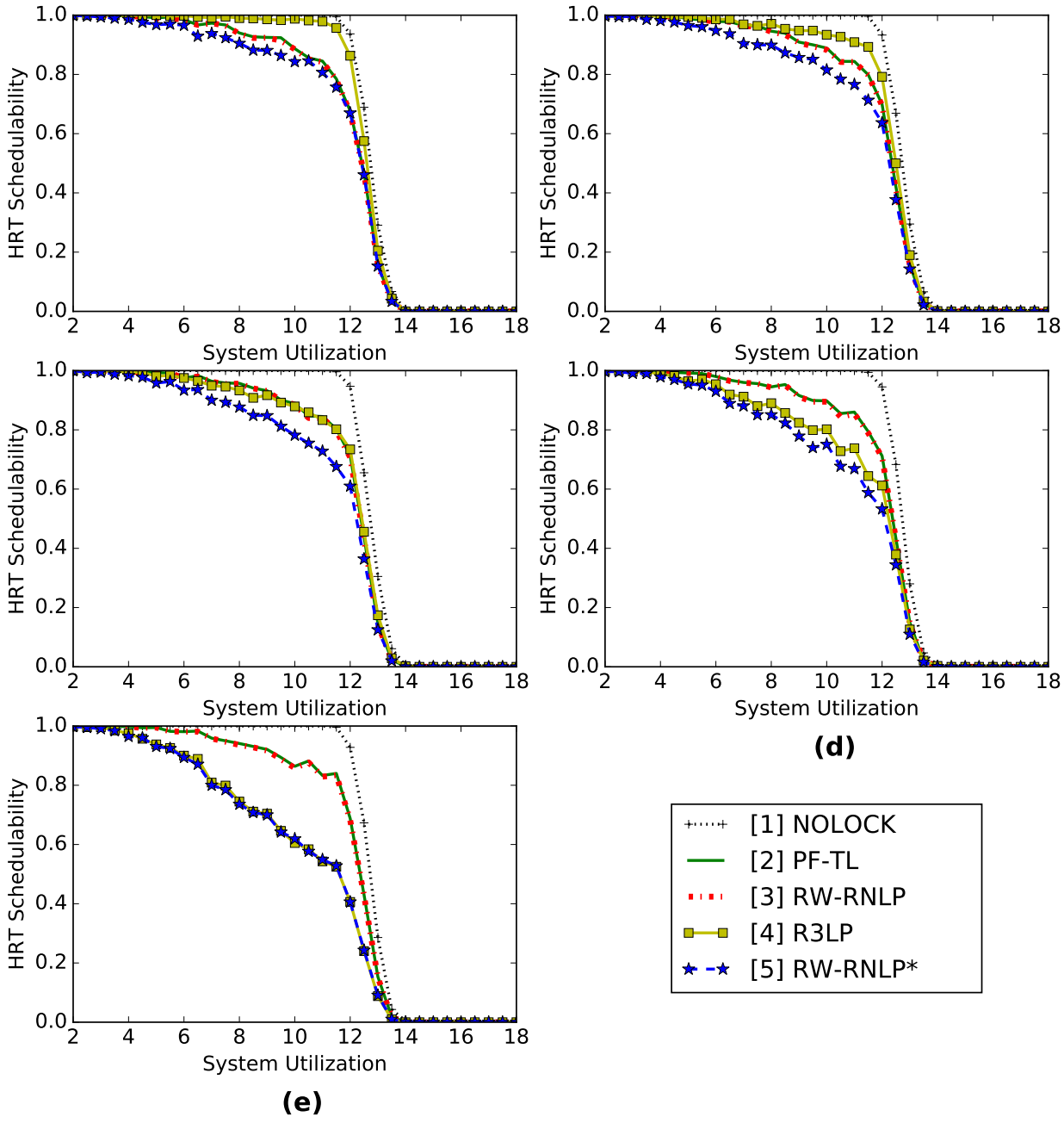



Figure 736: Hard real-time schedulability results with varying nested probabilities for the scenario with heavy task utilizations, long periods, long critical-section lengths, and read probability 0.8 in which a task has probability 1.0 of issuing a request. Nested probabilities are **(a)** 0.01, **(b)** 0.05, **(c)** 0.1, **(d)** 0.2, and **(e)** 0.5.