A Step toward Realistic Performance Evaluation of High-Speed TCP Variants

Sangtae Ha, Yusung Kim, Long Le, Injong Rhee
Department of Computer Science
North Carolina State University
Raleigh, NC 27695

Lisong Xu
Department of Computer Science and Engineering
University of Nebraska
Lincoln, NE 68588

Abstract: This is a work-in-progress report on our work on designing realistic evaluation suites for testing high-speed TCP variants. In this work, we have created an experimental network model that captures some of the complex characteristics of propagation delays and background traffic [14, 15, 20]. We use our network model to evaluate a large collection of recently proposed TCPs for high-speed networks: BIC TCP, CUBIC, FAST, HSTCP, H-TCP, and STCP. While we do not claim that we have the most realistic experimental network model, we believe that our work is a right step towards improving experimental methodologies for evaluating network protocols. In this report, we show how protocols could behave differently under the presence or absence of background traffic, and point out the danger of drawing conclusions based on testing under an isolated case of no background traffic.

1 Introduction

Congestion control is an important component of a transport protocol in a packet-switched shared network. The congestion control algorithm of the widely used transport protocol TCP is responsible for detecting and reacting to overloads in the Internet and has been the key to the Internet's operational success. However, as link capacity grows and new Internet applications with high-bandwidth demand emerge, TCP performance is unsatisfactory, especially on high-speed and long distance networks. The main reason for this is the conservative behavior of TCP in adjusting its congestion window that governs the senders' transmission rates.

A number of solutions have been proposed to remedy the aforementioned problem of TCP by changing the way in which TCP adapts its congestion window: BIC TCP [1], CUBIC [2], FAST [3], HSTCP [4], H-TCP [5], STCP [6], TCP-Westwood [10], LTCP[27] and TCP-Africa [11]. These new protocols promise to improve TCP performance on high-speed networks significantly and are hence usually called TCPs for high-speed networks.

While the design of TCPs for high-speed networks has received a considerable amount of interest, far less attention has been paid to thorough evaluations of these protocols. For example, Internet measurement studies showed complex behaviors and characteristics of Internet traffic [14, 15, 20]. Unfortunately, existing evaluation work [8] did not capture these behaviors in their testing environments. Since congestion control algorithms are very sensitive to environmental variables such as background traffic and propagation delays, thorough performance evaluations of TCPs for high-speed networks require creating realistic network environments where these protocols are likely to be used.

There are many factors in constructing realistic network testing environments. Most frequently used factors include end-to-end characteristics such as (1) bottleneck bandwidth, (2) round-trip times of protocol flows being observed, (3) the network topology over the path that protocol flows of interest travel through, and (4) queue size at the bottleneck link. These factors are more or less statically captured in a simulation and do not change over the course of the experiment. What is missing in most of existing evaluation work is the considerations of (1) what the protocol flows of interest dynamically (i.e., in a time-varying manner) experience in the middle of the network path, namely the dynamic statistical properties of background traffic over the intermediate links and (2) the impact of background traffic on the statistical properties on a link. These dynamic characteristics include "background" network traffic over both forward and backward directions of these links. These are background traffic because they are not being measured at end points and can still influence the behaviors of the protocol flows being observed at the end points.

There are several reasons why background traffic is important in protocol testing. First, network environments without any randomness in packet arrivals and delays are highly susceptible to the phase effect [25], a commonly observed simulation artifact caused by extreme synchronization of the network flows on the end-to-end path. A good mix of background traffic with diverse arrival patterns and delays reduce the likelihood of the phase effect. Second, a high degree of statistical multiplexing is often assumed in protocol design. For instance, the authors of HSTCP and STCP rely on statistical multiplexing for faster convergence (so criticizing these protocols for slow or no convergence under environments without background traffic is unfair). Today's Internet contains a varying degree of multiplexing and it is very unlikely that a production network does not contain any mix of background traffic. Third, it enables a study on the impact of the protocol flows on the "passing-through" aggregate traffic. This passing-through traffic is not ob-

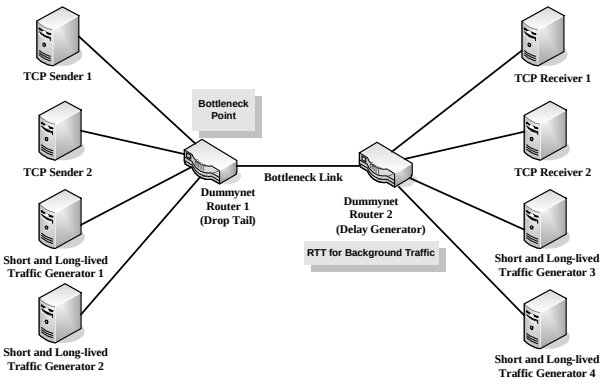


Figure 1: Experimental network setup.

served at the end points of observation as they have different end-points. These aggregate behaviors include queue fluctuations at the bottleneck links, total link utilization fluctuations, traffic distributions, and average response time and throughputs (for short-lived web traffic).

In this work, we have created an experimental network model that captures some of the complex characteristics of propagation delays and background traffic [14, 15, 20]. We use our network model to evaluate a collection of recently proposed TCPs for high-speed networks: BIC TCP, CUBIC, FAST, HSTCP, H-TCP, and STCP. While we do not claim that we have the most realistic experimental network model, we believe that our work is a right step toward improving experimental methodologies for evaluating network protocols. Since we make no claim about the realism of our background traffic mix, this report has a modest goal of simply contrasting protocol behaviors observed from two different environments created with and without background traffic. Our future work will evolve into testing protocols under more realistic background traffic. Our plan is to use some of the existing traffic generators such as Tmix [24] and Harpoon [23] that use real network traces as seeds for generating synthetic network traffic, and create a standard set of network testing environments where the network community can test and compare protocol behaviors.

2 Related Work

Floyd proposed a framework for evaluating congestion control algorithms [13]. The framework includes a number of metrics such as throughput, packet loss rates, delays, and fairness as well as a range of network environments. Along the same line, Wei et al. [12] proposed that the networking community establish a TCP benchmark suite to leverage comparative performance evaluations of TCP variants. The benchmark includes various scenarios for realistic performance evaluations such as heavy-tail file size distributions and ranges of propagation delays. The frameworks proposed by Floyd and by Wei et al. illustrate the need for realistic performance evaluations of new congestion control algorithms and accentuate the motivation for our work and existing evaluation work that we briefly review below.

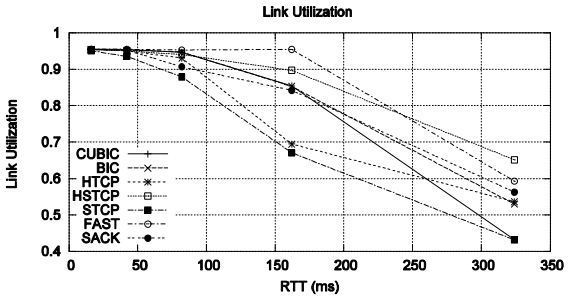
Bullot et al. compared the performance of TCP New Reno with HSTCP, FAST, STCP, HSTCP-LP, H-TCP, and BIC TCP on high-speed production networks [7]. They reported that TCP New Reno gave low and unstable performance and most TCPs for high-speed networks delivered significant improvement over TCP Reno. Bullot et al.'s results are very encouraging. Nevertheless, as their experiments were performed over a real production network path, they did not have any control over the background traffic on the network. They only included UDP background traffic and did not consider the impact of network environments created by various mixes of background traffic on protocol behaviors.

Li et al. [8] performed experiments for STCP, HSTCP, BIC TCP, FAST, and H-TCP in a lab network. They noted that most protocols, especially FAST, STCP, HSTCP and BIC, exhibit substantial unfairness and highlighted the good performance of HTCP. Since Li et al. did not have any background traffic in their experiments, their results may be subject to the deficiencies we point out in the introduction.

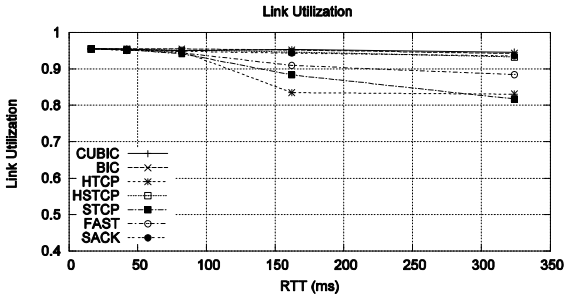
3 Experimental Design

3.1 Testbed setup. The experimental network that we used to perform experiments for TCPs for high-speed networks is shown in Figure 1. At each edge of the network are four machines that have identical hardware configurations. Two machines are used as TCP senders and run *iperf* to simulate high-performance applications that have the demand to transmit a large amount of data to two other machines functioning as TCP receivers on the other side of the network. The TCP senders run a modified version of Linux 2.6.13 kernel that includes the implementations of new congestion control algorithms for high-speed networks.

As pointed out by Li et al. [8], existing implementations of various congestion control algorithms often make changes to parts of the TCP stack that are not directly related to the congestion control algorithm in order to improve their overall performance. To be fair to all congestion control algorithms, we run a modified version of Linux 2.6.13 kernel that separates the implementation of congestion control algorithms from the standard TCP stack (with the exception of FAST that has an implementation in Linux kernel 2.4 because FAST is not yet publicly available in Linux kernel 2.6). Further, we modified the SACK implementation to remove inefficiencies of the standard SACK implementation. Our improved SACK implementation is equally effective for all congestion control algorithms.

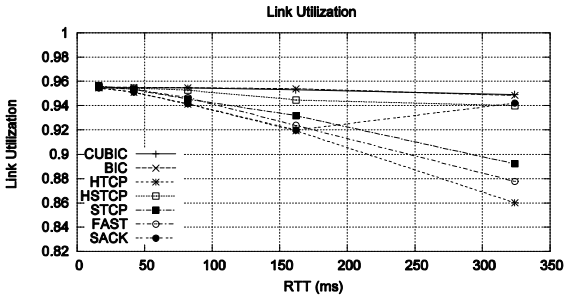


(a) Without background traffic

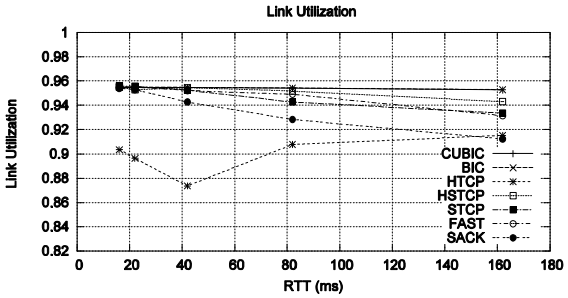


(b) With background traffic

Figure 2. Link utilization with one TCP SACK and one TCP variant flow. Both flows have same RTTs.

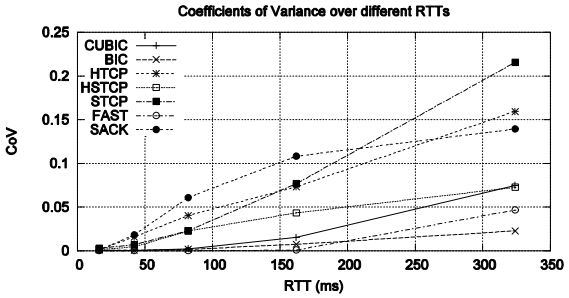


(a) With the same RTT for the two flows

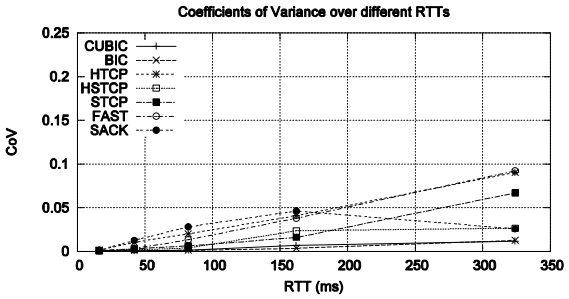


(b) RTT for one flow is set to 162 ms and RTT of the other flow is varied between 16 ms and 164 ms.

Figure 3. Link utilization with two flows of a high-speed TCP variant and background traffic

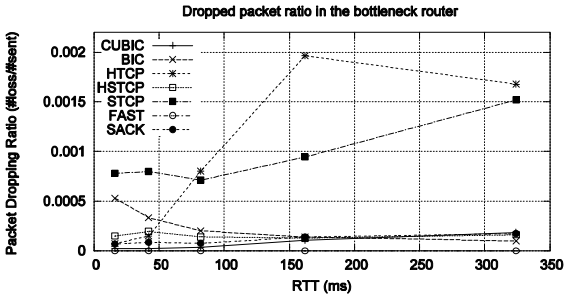


(a) Without background traffic

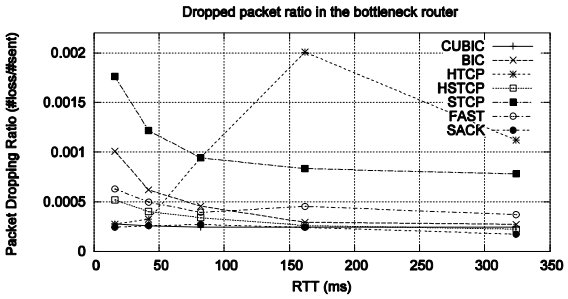


(b) With background traffic

Figure 4. Stability



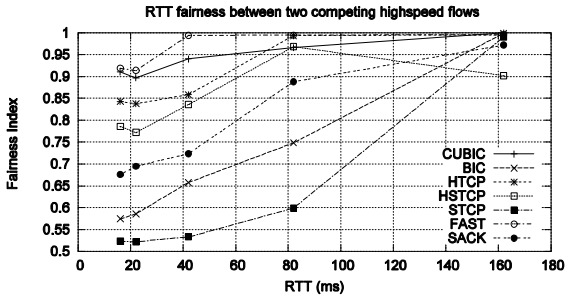
(a) Without background traffic



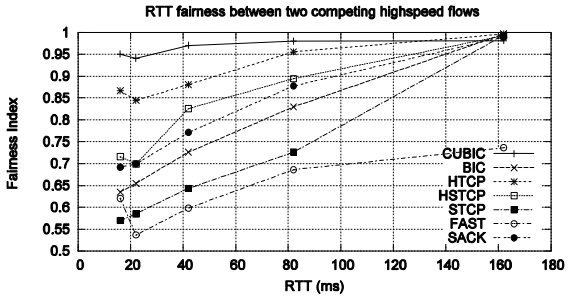
(b) With background traffic

Figure 5. Packet loss rate

At the core of the network are two FreeBSD 5.2.1 machines running *dummynet* software [21]. These machines are tuned to be capable of forwarding traffic close to 1 Gbps. The *dummynet* software is used in the first router to control the bandwidth and buffer size of the bottleneck link. The bandwidth of the bottleneck link is configured to be 400 Mbps. Unless mentioned otherwise, the buffer size of the bottleneck link is fixed to the maximum of 2 Mbytes. While various rules of thumb recommend that the buffer size be proportional to the bandwidth delay product (BDP), we test the

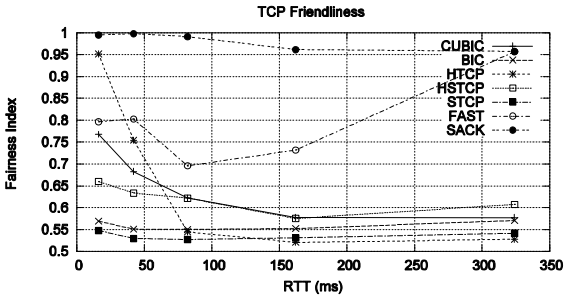


(a) Without background traffic

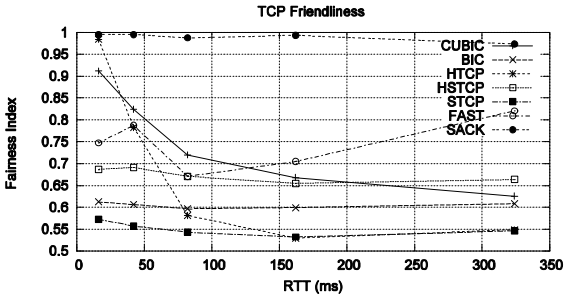


(b) With background traffic

Figure 6. RTT Fairness.

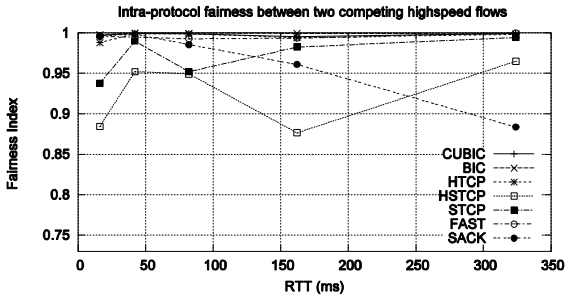


(a) Without background traffic

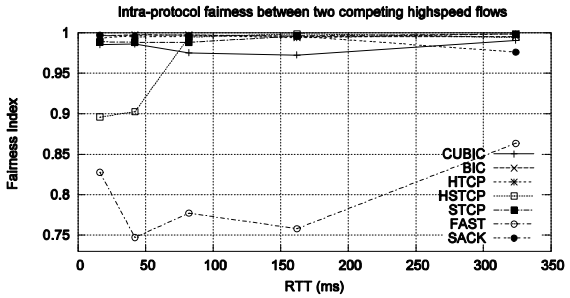


(b) With background traffic

Figure 7. TCP Friendliness.

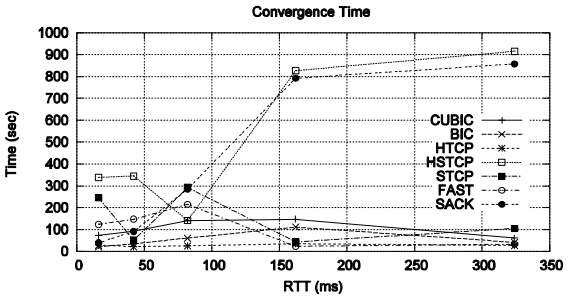


(a) Without background traffic

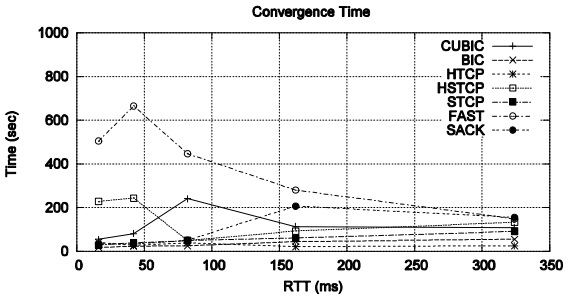


(b) With background traffic

Figure 8. Intra-protocol Fairness.



(a) Without background traffic



(b) With background traffic

Figure 9. Convergence.

protocols under a smaller router buffer size than BDP which is a likely trend in high-speed routers. This trend is in line with recent research results showing that the buffer size of a bottleneck link with a high degree of multiplexing of TCP connections can be much less than the bandwidth delay product [16, 17].

3.2 Model for propagation delays. An extension of *dummynet* is used in the second router to assign *per-flow delays* to background traffic flows (long-lived and short-lived). This configuration gives all packets from a flow the same amount of delay that is randomly sampled from a distribution obtained from a measurement study [14]. This allowed us to obtain results as if our experiments would have

been performed on an actual wide-area network where different flows passing through a router experience different amount of delays.

3.3 Models for background traffic. Since high-speed variants of TCP are unlikely to run alone in dedicated networks, we need to generate background traffic to make our results realistic as in real-world scenarios. Two types of flows are used to generate background traffic: long-lived and short-lived. The long-lived flows are generated by *iperf* and used to simulate regular long-lived TCP flows such as ftp. The amount of traffic generated by these flows is controlled by the number of *iperf* connections. Short-lived flows are used to simulate web sessions and are generated using Lognormal (Body) and Pareto (Tail) distribution for their file sizes [12, 15, 20]. The inter-arrival between two successive short-lived flows follows an exponential distribution and is used to control the amount of short-lived flows [12, 20]. The chosen distributions of file sizes and inter-arrival times are considered representative of Internet traffic characteristics [12, 20]. Further, we also generate reverse traffic consisting of both short-lived and long-lived flows to achieve the effects of ACK compression and to reduce the phase effect [22].

Each experiment with high-speed variants of TCP is run for 1200 seconds. The long-lived and short-lived background flows start at time 0. We have two high-speed TCP flows in each experiment. The first flow starts at 30 seconds and the second flow starts at 130 seconds. We took measurements after the first 135 seconds to eliminate the start-up phase.

4 Experimental Results

We performed a suite of experiments where propagation delays for the two high-speed TCP flows were set to 16, 42, 82, 162, and 324 ms to simulate different network scenarios. Note that while the propagation delays for high-speed TCP flows were set to these values, propagation delays for background traffic were randomly sampled from a realistic model for propagation delays [14] as described in section 3. We simulated scenarios where the two high-speed TCP flows either experienced the same or different propagation delays. Further, we performed experiments with and without background traffic to contrast experimental results and protocol behaviors between these scenarios. The high-speed variants of TCP are evaluated based on the following properties: fairness, convergence time, RTT fairness, TCP friendliness, link utilization, stability of throughput, and packet loss rates.

4.1 Utilization. We measure the utilization ratio of the bottleneck link capacity. We found that the utilization ratio shows high sensitivity to the queue size of the bottleneck router, the characteristics of background traffic and the behavior of protocols being tested.

Figure 2a shows experimental results where we run one high-speed TCP flow together with one flow of TCP-SACK. In each experiment, both flows have the same RTT. We vary the RTTs from 16 to 324 ms. The buffer size of the bottleneck router is fixed to 2 Mbytes. With a RTT of 324 ms, this buffer size amounts to roughly 12% of bandwidth and delay product (BDP) of the network. All protocols drop their utilization below 65% in the 324ms RTT run. This is because the window size (cwnd) and its fluctuations of the high speed TCP flow (except for TCP-SACK) are too large to be accommodated by such buffer. With 324 ms RTT, the maximum window size of high-speed flows is larger than 10,000 packets (around 20 Mbytes).

As we added background traffic (both short and long lived flows) to the same experiments, the utilization ratio improved dramatically. Figure 2b shows the utilization results of the same experiments as in Figure 2a but with background traffic. The utilization ratio with a RTT of 324 milliseconds improved up to 90-95% for HSTCP, CUBIC, BIC, FAST and TCP-SACK. This change in utilization confirms the recent work by Appenzeller et al. [16] that high statistical multiplexing in the bottleneck link (i.e., high randomness) permits use of small buffers. Note that there is enough background traffic to consume all the link capacity even without high speed TCP flows. These background flows are generated by TCP and their transmission rates are elastic to the amount of traffic in the network to the extent limited by the maximum allowed by 64KB receiver buffer size.

Even with background traffic, the utilization of HTCP, FAST and STCP with RTTs of 160 and 324 milliseconds is significantly lower than that of other protocols. We conjecture that this problem is related to the inherent protocol behavior of HTCP in the way that HTCP ties its window reduction to the estimated buffer size of the network. For STCP, it is being simply too aggressive. For FAST, its behavior becomes less predictable with presence of background traffic due to noise in RTT estimation. Detailed examinations of these protocol behaviors are available in [26].

The previous experiments investigated protocol behaviors of high-speed TCP variants when they competed with a TCP-SACK flow (with and without background traffic). Below we study protocol behaviors when two flows of a high-speed TCP variant competed with each other (the two flows experienced either the same RTT or different RTTs). Figures 3a and 3b show two different types of experiments, all with background traffic: (1) two flows of a TCP variant with the same RTT and (2) two flows of a TCP variant, but the RTT of one flow is set to 162 ms and the RTT of the other flow is varied from 16 to 162 ms.

Experiments in Figures 3a and 3b are conducted with two flows of the same TCP variant. But the difference lies in the RTT values of each flow. When flows have the same RTT, their average transmission rate is approximately the same (if they ensure intra-protocol fairness) and also lower than

when they have different RTTs in which case one flow tends to have a higher transmission rate than the other. We observe that the utilization of HTCP is generally lower than that of the other protocols. Note that the experiments inject enough background traffic to consume the entire network capacity even without high speed flows. In the second experiment (Figure 3b), HTCP shows lower utilization even with small RTTs. This suggests that HTCP may also have lower utilization when it competes with heterogeneous traffic of different RTTs. More details about the dynamic protocol behaviors are provided in our technical report [26].

4.2 Stability. We measure the stability of a protocol by the coefficient of variance (CoV) that was also used in [3]. We take samples of transmission rates of the protocol flows at periodic intervals. Each sample is the arithmetic average of the transmission rate during that interval. We are interested in the stability (or instability) that high-speed protocol flows induce to the entire network traffic. Thus, here we measure the total throughput at the bottleneck router instead of the transmission of a protocol. We compute the average CoV of the average values of the total throughput at the bottleneck router measured at every 10-second interval (we also have results for other time intervals in [26]). We measure CoV after the first 200 seconds in each run of 1200 seconds.

Figure 4a shows the average CoV of the various protocol flows when two flows of a high-speed TCP variant run with the same RTT. No background traffic is added. In our experiment, 0.1 CoV indicates high instability. We can see HTCP, STCP and TCP-SACK cause high fluctuations in the bottleneck router capacity usage. Figure 4b shows the same metric of the runs with the same setup as the above but with background traffic. We observe that the CoVs of all the protocols have reduced and conclude that as we add more background traffic, the stability of protocols gets improved. But HTCP still shows very high CoV values. FAST also shows gradually increasing CoV values as RTT increases.

4.3 Packet loss rate. In this section, we examine packet loss observed at the bottleneck router as high-speed TCP flows compete for the bottleneck capacity. We measure the total packet loss at the bottleneck link and do not distinguish flows that experience packet losses. Like the stability measurement, this performance metric also measures the impact of high-speed TCP flows on the background traffic.

The packet loss rate of various protocols is plotted in Figure 5a. In this experiment, we do not add any background traffic and only two flows of the same TCP variant protocol with the same RTT run at the same time. We observe that HTCP has much higher packet loss rates than the other protocols in the runs with RTTs 80ms and longer. STCP also shows high packet loss rates as RTTs increase. This is because STCP sees the effect of small buffer sizes with high RTTs. FAST shows the least packet loss ratio among all the protocols. Figure 5b shows the results with background traffic. We see packet loss rates for most protocols slightly increase. But STCP has a lower loss rate with 324ms RTT. We conjecture that this is because the randomness in the network improves the stability and robustness of STCP (and all protocols as well). HTCP still induces significantly more packet losses even with background traffic than the other flows.

4.4 RTT Fairness. We measure the fairness in sharing the bottleneck bandwidth among competing flows that have different RTTs. There are several notions of "RTT fairness". One notion is to achieve the equal bandwidth sharing where the two competing flows may share the same bottleneck bandwidth even if they have different RTTs. This property may not be always desirable because long RTT flows tend to use more resources than short RTT flows since they are likely to travel through more routers over a longer path. Another notion is to have bandwidth shares inversely proportional to the RTT ratios. This proportional fairness makes more sense in terms of the overall end-to-end resource usage. Although there is no commonly accepted notion of RTT-fairness, it is clear that the bandwidth share ratio should be within some reasonable bound so that no flows are being starved because they travel a longer distance. Note that RTT-fairness is highly correlated with the amount of randomness in packet losses (or in other words, the amount of loss synchronization) [1]. In more random environments, protocols tend to have better RTT-fairness.

Figure 6a shows the RTT fairness of various protocols without any background traffic. Two flows are tested; we fix the RTT of one flow to 162ms and vary the other flow from 16ms to 162ms. FAST has the best fairness index and achieves the equal RTT fairness among the two FAST flows regardless of their RTTs. CUBIC has RTT fairness linearly proportional to the inverse of the RTT ratio (i.e., the short RTT flow having proportionally more bandwidth share than the long RTT flow). So its RTT fairness is slightly lower than FAST. As discussed above, we question whether this equal sharing property of FAST regardless of delays is desirable. HTCP and HSTCP have similar RTT fairness. BIC's RTT fairness is lower than HSTCP but higher than STCP. This behavior is expected as explained in [1]. BIC is known to follow the same RTT fairness as TCP-SACK under a very large BDP network [1]. In the current testing environment, BIC's RTT fairness is targeted to be in between those of TCP and STCP. The argument is that in a network of this size, there will be enough multiplexing so RTT unfairness would not be so severe. We also found that HTCP allows the long RTT flow to have more bandwidth share.

Figure 6b shows the same metric as in Figure 6a but with background traffic. In this experiment, we expect more asynchrony in packet losses. In the test, we found that background traffic has the biggest impact on FAST while in general, most protocols improve their fairness compared to the

cases without background traffic. As suggested earlier, BIC's fairness has been improved substantially close to TCP-SACK as we add background traffic.

4.5 TCP Friendliness. We measure how TCP-friendly the high-speed protocols are by running experiments with one high-speed flow and one regular TCP flow with the same RTT over the same bottleneck link. These experiments were performed with and without background traffic. We measured TCP friendliness by Jain's fairness index [18] using the throughput of the high-speed flow and of the regular TCP flow. Jain's fairness index is a normalized number between 0 and 1 (1 being the greatest fairness). Jain's fairness indices for various high speed TCP variants are shown in Figure 7a and 7b as we vary the RTTs from 16 to 324 ms.

We observe that HTCP has the best TCP-friendliness in very low RTT networks (where TCP-friendliness is important because TCP-SACK does not have much performance problem in these networks) with or without background traffic. However, as RTT increases beyond 16ms, HTCP's fairness to TCP drops rapidly in both cases. In general, we note that all TCP variants (except FAST) improved their TCP friendliness when background traffic is added to the experiments. This is mostly because of two reasons. First, increased background traffic takes away bandwidth from high speed TCP variants so they become less aggressive as their average window sizes become less than without background traffic. The other reason is that with background traffic, randomness in packet losses increases. As we can see that HSTCP, CUBIC, and BIC improve their TCP fairness indices considerably with background traffic, background traffic breaks loss synchronization and allows flows to adapt their transmission rates more asynchronously.

We conducted the same experiments on a 100-Mbps bottleneck link and observed that the phenomenon mentioned above also occurs more vividly in a smaller bandwidth network (most notably with HSTCP, CUBIC and BIC). We observe that FAST shows the best TCP friendliness in high RTT networks. This is not necessarily desirable in such networks because TCP-SACK is too conservative in high BDP networks. In addition, the TCP friendliness of FAST has been affected the most by the presence of background traffic. Details of these results can be found in our report [26].

4.6 Intra-protocol Fairness. We measure the intra-protocol fairness of protocols by performing experiments with two flows of a high-speed protocol with the same RTT. These two flows' throughput is used as input to compute Jain's fairness index. These experiments are conducted when RTTs are varied between 16 and 324 milliseconds. Figure 8a and 8b show intra-protocol fairness of protocols with and without background traffic.

Without background traffic, HSTCP, TCP-SACK and STCP show lower fairness indices than the other protocols. However, as we add background traffic, we find that all the protocols (except FAST) show very good fairness. The reason for this result could be that since FAST is delay-based, background traffic introduced more dynamics and fluctuations in the bottleneck queue and made it more difficult for FAST flows to estimate their fair shares of bandwidth based on delay information. We also observe that HSTCP obtained a higher fairness index in the presence of background traffic (as noted by its author, HSTCP relies on statistical multiplexing for faster convergence).

4.7 Convergence. Figure 9a and 9b show the convergence time of two high-speed protocol flows with and without background traffic that are started at different times. The convergence time is defined to be the elapsed time when the timed average throughput of the second flow reaches 80% of the first flow (recall from section 2 that the two flows are started at 30 and 130 seconds). The average throughput is obtained at one-second intervals. The convergence time shown in Figure 9a and 9b allows us to discuss quantitatively the dynamic behaviors of protocols that we already qualitatively pointed out above.

CUBIC and BIC show up to 150 second convergence time when running without background traffic. Under 300ms RTT, their convergence times reduce. This is because the small buffer size (of 2MB) allows the second flow to perturb the network significantly, when it goes into slow start, to force the first flow to drop its bandwidth share quickly. This case can be seen from STCP – because STCP is very aggressive, the second flow always forces the first flow to come down quickly. Thus STCP shows fairly short convergence times with high RTTs. (However, in most cases, STCP does not show good convergence beyond 80% with no background traffic.) On the other hand, HSTCP shows very slow convergence time and also in most case, their convergence beyond 80% is not possible. As can be seen in Figure 9b, HSTCP reduces its convergence time considerably when running with background traffic. Further, STCP also showed improved convergence behavior with background traffic. On the other hand, FAST increased their convergence time noticeably in the presence of background traffic. With low RTTs, FAST flows do not converge. The other protocols showed a rather short convergence time both with and without background traffic.

While the convergence time sheds some light on the dynamic behavior of the protocols, it does not give the complete view on the convergence behavior of the protocols. For example, although the convergence time measures the time that the second high-speed flow takes until it reaches 80% throughput of the first flow, it does not provide any information about the dynamic behavior of these flows after that. Another metric that can be used to investigate the dynamic behavior of protocols is the average fairness index over different time scales. Results for this performance metric is reported in our technical report due to space limitation

[26]. We only note here that most protocols improved their convergence behavior in the presence of background traffic.

5 Conclusions

We presented results of an evaluation study of a collection of high-speed TCP variants. Due to space limitation, we can only report a subset of our results here and encourage readers to read our technical report [26]. We used different metrics such as fairness, convergence time, packet loss rates, link utilization, RTT fairness, TCP friendliness, and stability of throughput to evaluate these protocols. It is known that background traffic may affect the protocol behavior, but little is known "how" it is going to affect the behavior. Our study sheds some light on the problem. Further study will show more interesting properties with background traffic.

We do not declare any winner in our evaluation but simply show contrasting results and protocol behaviors when experiments were conducted with and without background traffic. Our results demonstrated that different conclusions can be drawn when protocol evaluations were conducted with and without background traffic. Thus, evaluating a new protocol without background traffic can be dangerous and a thorough evaluation needs to look at a variety of testing scenarios to make a valid observation about the behavior of a protocol. While we do not claim that our models for traffic and propagation delays are the most realistic, we believe that evaluations of a new protocol without background traffic are likely unrealistic. We propose that evaluations of a new protocol should use diverse scenarios that involve many different models for traffic and propagation delays.

Further, we also conclude that high-speed protocols have rather complex behaviors and a thorough evaluation of these protocols need to investigate all aspects of their behaviors. It appears that there will probably be no "perfect" high-speed protocol that would be a clear winner in all different (and sometimes conflicting) aspects of protocol behaviors.

6 References

[1] L. Xu, K. Harfoush, and I. Rhee, "Binary Increase Congestion Control for Fast Long-Distance Networks", INFOCOM 2004.

[2] Injong Rhee and Lisong Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant", PFLDnet 2005.

[3] C. Jin, D. X. Wei and S. H. Low, "FAST TCP: motivation, architecture, algorithms, performance", INFOCOM 2004.

[4] Sally Floyd, "HighSpeed TCP for Large Congestion Windows", IETF RFC 3649, December 2003.

[5] Douglas Leith and Robert Shorten, "H-TCP Protocol for High-Speed Long Distance Networks", PFLDnet 2004.

[6] T. Kelly, "Scalable TCP: Improving Performance on Highspeed Wide Area Networks", ACM CCR, April 2003.

[7] H. Bullot, R. L. Cottrell, and R. Hughes-Jones, "Evaluation of Advanced TCP Stacks on Fast Long-Distance Production Networks", PFLDnet 2004.

[8] Y. Li, D. Leith, and R. Shorten, "Experimental Evaluation of TCP Protocols for High-Speed Networks", Technial report, Hamilton Institute, 2005.

[9] C. Jin, D. Wei, S. Low, G. Buhrmaster, J. Bunn, D. Choe, R. L. A. Cottrell, J. C. Doyle, W. Feng, O. Martin, H. Newman, F. Paganini, S. Ravot, S. Singh, "FAST TCP: From Theory to Experiments", IEEE Network, January/February 2005.

[10] R. Wang, K. Yamada, M. Yahya Sanadidi, and M. Gerla, "TCP with sender-side intelligence to handle dynamic, large, leaky pipes", IEEE Journal on Selected Areas in Communications, 23(2):235-248, 2005.

[11] R. King, R. Riedi, and R. Baraniuk, "Evaluating and Improving TCP-Africa: an Adaptive and Fair Rapid Increase Rule for Scalable TCP", PFLDnet 2005.

[12] David X. Wei, P. Cao, and Steven H. Low, "Time for a TCP Benchmark Suite?", Technical report, 08/2005, available at www.cs.caltech.edu/~weixl/research/technicals/benchmark/summary.ps .

[13] S. Floyd, Metrics for the Evaluation of Congestion Control Mechanisms, August 2005, Internet draft, draft-irtf-tmrg-metrics-00.txt .

[14] Jay Aikat, Jasleen Kaur, F. Donelson Smith, and Kevin Jeffay, "Variability in TCP Roundtrip Times", ACM IMC 2003.

[15] Paul Barford and Mark Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation", ACM SIGMETRICS 1998.

[16] G. Appenzeller, I. Keslassy, and N. Mckeown, "Sizing router buffers", in Proceeding of ACM SIGCOMM'04.

[17] D. Barman, G. Smaragdakis, and I. Matta, "The Effect of Router Buffer Size on HighSpeed TCP Performance", IEEE Globecom 2004.

[18] D. Chiu and R. Jain. "Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks." Journal of Computer Networks and ISDN, 17(1):1–14, 1989.

[19] E. Altman, K. Avrachenkov, B.J. Prabhu, "Fairness in MIMD congestion control algorithms", IEEE INFOCOM, 2005.

[20] S. Floyd and V. Paxson, "Difficulties in Simulating the Internet", ACM/IEEE Transactions on Networking, August 2001.

[21] Luigi Rizzo, "Dummynet: A simple approach to the evaluation of network protocols", ACM CCR, January 1997.

[22] L. Zhang, S. Shenker, and D. Clark, "Observations on the Dynamics of a Congestion Control Algorithm: the Effects of Two-Way Traffic", SIGCOMM 1991.

[23] J. Sommers, P. Barford, and H. Kim, "Harpoon: A Flow-Level Traffic Generator for Router and Network Tests", extended abstract, ACM SIGMETRICS 2004.

[24] F. Hernández-Campos, F. D. Smith, and K. Jeffay, "Generating Realistic TCP Workloads", in Proceedings of CMG 2004.

[25] S. Floyd and E. Kohler, "Internet Research Needs Better Models", HotNets-I, October 2002.

[26] S. Ha, Y. Kim, L. Le, I. Rhee, and L. Xu, "A Step toward Realistic Evaluation of High-Speed TCP Protocols", http://www.csc.ncsu.edu/faculty/rhee/export/bitcp/asteppaper.htm .

[27] S. Bhandarkar, S. Jain and A. L. N. Reddy, "Improving TCP Performance in High Bandwidth High RTT Links Using Layered Congestion Control, PFLDNet 2005.