Research Statement — UmaMaheswari C. Devi

My current research focus is primarily real-time systems: in particular, design and analysis of processor-scheduling algorithms and development of formal validation techniques (for verifying temporal correctness) for soft real-time systems instantiated on tightly-coupled, identical multiprocessor platforms, typified by SMPs. This work is necessitated by the proliferation of both multiprocessor platforms and applications with soft real-time (or quality-of-service) constraints and workloads that warrant more than one processor. My research has resulted in techniques that can enable systems as described to be implemented more efficiently, using fewer resources, and with better flexibility, than possible previously.

Apart from real-time scheduling and operating systems, I am interested in several other areas within computer science, including computer networking, parallel and distributed computing, computer architecture, and compiler design. I hope to be able to extend my future research to include my other areas of interest, and expect it to be guided by emerging and evolving needs, trends, and applications. My near-term goal is to develop efficient and robust resource-allocation algorithms and infrastructure support that can provide performance guarantees for systems that range in scale from stand-alone embedded systems to large-scale distributed systems. In what follows, I describe my work to date and my current plans for the future.

1 Summary of Dissertation Research

Unlike a hard real-time constraint (or deadline), whose violation can lead to disastrous consequences such as loss of life or property, a soft real-time constraint is less critical. Hence, soft real-time systems can often tolerate deadline misses by bounded durations, i.e., bounded tardiness, and allow timeliness to be traded for improved resource utilization. The goal of my dissertation research has been to explore such trade-offs and develop and enable resource-allocation algorithms that are less complex, more efficient, and more flexible and seamless in the face of runtime changes, while providing timeliness guarantees, for multiprocessor-based soft real-time systems. The need for work in this direction and the results obtained are described below.

1.1 Background and Motivation

One evident trend in the design of both general-purpose and embedded systems is the increase in the use of multiple processing elements that are tightly coupled. In the general-purpose arena, this is evidenced by the availability of reasonably-priced symmetric multiprocessor platforms (SMPs), and the emergence of multicore architectures, which have multiple processing units on a single chip. In the special-purpose and embedded arena, examples of multiprocessor designs include network processors, which are used for packet processing tasks in programmable routers, system-on-chip platforms for multimedia processing in set-top boxes and digital TVs, and automotive power-train systems. If the current shift towards multicore architectures by prominent chip manufacturers such as Intel and AMD is any indication, then in the future, the standard computing platform in many settings can be expected to be a multiprocessor. The need for multiprocessors is due to both architectural issues that impose limits on the performance that a single processing unit can deliver, and the prevalence of, and ever-increasing need for, higher computing demand from applications. Further, the energy consumed by an \( m \)-processor system, ideally, is lower than that consumed by a single-processor system of equivalent capacity by a factor of approximately \( m^2 \) [1].

Some embedded systems, such as set-top boxes and automotive systems, are inherently real-time. Also, a number of emerging real-time applications exist that are instantiated on general-purpose systems and have high workloads. Systems that track people and machines, virtual-reality and computer-vision systems, systems that host web-sites, and some signal-processing systems are a few examples. Timing constraints in several of these embedded- and general-purpose-system applications are predominantly soft. Hence, with the shift towards multiprocessors, the need arises to instantiate soft real-time applications on multiprocessors.
Three basic approaches to real-time scheduling, which is central to ensuring the timing constraints of a real-time system, have been considered on multiprocessors: partitioning, non-Pfair-based global scheduling, and Pfair-based global scheduling [14]. Under partitioning, tasks (i.e., process threads) are statically assigned to processors, and a uniprocessor scheduling algorithm is used on each processor to schedule its assigned tasks. In contrast, under global scheduling, a task may execute on any processor and may migrate across processors.

Most prior research on real-time scheduling on multiprocessors has focused only on hard real-time systems. For such systems, partitioning and non-Pfair-based global scheduling algorithms, such as the global earliest-deadline-first (EDF) algorithm, are not optimal\(^1\) in that they require restrictions on total system utilization, i.e., workload, that can approach roughly 50% of the available processing capacity [14]. Pfair scheduling [5, 15] is currently the only known way of optimally scheduling recurrent real-time task systems on a multiprocessor. However, for optimality, Pfair algorithms schedule tasks one quantum at a time, and as a result, tasks may be preempted and migrate across processors frequently. In addition, optimal Pfair algorithms are complex and impose certain other restrictions also, which can limit their practical implementation, add to runtime overheads, and lower the amount of useful work that is actually accomplished. Thus, the known scheduling algorithms in all the three categories described above are either lacking in validation techniques or may be overkill for soft real-time systems. Hence, to facilitate less-expensive implementations of soft real-time systems, it is necessary that the timeliness versus utilization tradeoff offered by such systems be explored to enable the use of less complex and more flexible algorithms in a resource-efficient manner.

### 1.2 Contributions (Dissertation Research)

Our\(^2\) contributions towards meeting the goal stated earlier are largely concerned with deriving tardiness bounds with no restriction on total system utilization (other than not exceeding the available processing capacity) for suboptimal algorithms that are less expensive (i.e., have low runtime overheads). Thus, if bounded tardiness is tolerable, then the system utilization level can be increased. The algorithms considered differ in their characteristics and target applications and can be classified into (i) global EDF and related algorithms, (ii) relaxed Pfair algorithms, and (iii) a partitioned-EDF-based algorithm. In deriving tardiness bounds for these algorithms, insights were discerned that led to the development of new analysis techniques and extensions of existing techniques.

**Global EDF and related algorithms.** EDF (earliest-deadline-first) schedules jobs\(^3\) on the basis of their deadlines, and when deployed in a global setting, is a good choice even for applications with moderately high migration costs or those that are comprised of dynamic tasks. This is because, under global EDF, each job contributes to at most one system-wide preemption and migration, and because, unlike partitioning algorithms, tasks are not pinned to processors, repartitioning is not necessary when the task composition changes. However, as already mentioned, under global EDF, nearly half the processing capacity may have to be wasted if every deadline must be met. To reclaim such wasted capacity for soft real-time systems, we have derived tardiness bounds under both preemptive (EDF) and non-preemptive (NP-EDF) global EDF [10]. (This paper received a best-paper award at the 26th IEEE Real-Time Systems Symposium, in December 2005.) Despite being subject to a higher tardiness bound than EDF, NP-EDF has the advantage of eliminating job preemptions and migrations, and thus, may be preferable when the cost of migrating any given job is high, but not much state is carried between jobs. Also, on each processor, at most one job may be in a started but unfinished state, and hence, NP-EDF has the added benefit of lowering the total stack size, which may be especially desirable in embedded systems.

Tasks in real-time systems are not always independent, but may share resources and be constrained by precedence relations. Towards developing a generic resource-sharing framework for EDF, we have proposed efficient techniques for synchronizing accesses to simple shared data structures in [13].

---

\(^{1}\)An algorithm is said to be optimal if it can meet all the deadlines of every task system whose total system utilization does not exceed the available processing capacity, which, in our case, is the number of processors.

\(^{2}\)This is joint work with my advisor, Prof. James H. Anderson.

\(^{3}\)Tasks in real-time applications are generally recurrent in nature. Each invocation of a recurrent task is referred to as its job.
The tardiness bounds derived for EDF are applicable to every task in a task system. However, tasks in some real systems are hybrid, i.e., may have varying tolerances to tardiness and may include hard tasks. In [12], we have presented a new algorithm based on global EDF for supporting such hybrid systems.

**Relaxed Pfair algorithms.** Pfair algorithms, which schedule tasks one quantum at a time and hence allow fine-grained adaptivity, may be better suited than EDF for applications with low-migration costs and those that are highly dynamic. In Pfair terminology, each quantum is referred to as a subtask, subtasks are assigned deadlines, known as pseudo-deadlines, and are scheduled on an earliest-pseudo-deadline-first (EPDF) basis. For optimality, non-trivial tie-breaking rules are used for resolving ties among subtasks with the same deadline [4]. Though necessary for optimality, tie-breaking entails overheads that may be unnecessary or unacceptable for soft or dynamic real-time systems. For such systems, the EPDF algorithm, which resolves ties arbitrarily, may be a better choice.

We have extended the existing validation analysis for EPDF in the following ways. First, we showed that the prevailing conjecture that a tardiness of one quantum holds for EPDF is false. Next, we improved the known sufficient per-task utilization restrictions needed for bounded tardiness under EPDF [8]. Third, we derived non-trivial sufficient restrictions along an orthogonal dimension, namely, total system utilization, for schedulability under EPDF, thereby extending the applicability of EPDF for hard real-time systems [9]. Finally, we have considered extending these restrictions to allow bounded tardiness [9].

A second requirement of optimal Pfair algorithms is synchronized scheduling, that is, tasks need to be allocated processor time in fixed-sized quanta that align on all processors. Apart from leading to wasted processor time, this restriction entails additional implementation overhead. In [11], we showed that if scheduling is desynchronized, then under an otherwise optimal Pfair scheduling algorithm, deadlines are missed by at most the maximum size of one quantum. Further, this result can be extended to most prior results on Pfair scheduling: in general, tardiness bounds guaranteed under EPDF and other sub-optimal Pfair algorithms are worsened by at most one quantum when scheduling is desynchronized. Finally, the same analysis can be used to show that if another Pfair restriction, which requires task periods to be integral, is relaxed, then deadlines can be missed by less than two quanta.

**Partitioning-EDF-based restricted-migration algorithm.** Finally, for use with applications with high inter- and intra-job migration costs, we designed an algorithm based on partitioned-EDF, called EDF-fm [2]. A pure partitioning algorithm (i.e., a no-migration algorithm), offers no scope for guaranteeing bounded tardiness, and hence, for improving processor utilization, for soft real-time systems that cannot be partitioned among available processors. Thus, migrations are necessary in such systems. Hence, in designing EDF-fm, our focus was on minimizing required migrations only to the extent needed for guaranteeing bounded tardiness. Specifically, under EDF-fm, on an m-processor system, the capability to migrate is required for at most m − 1 tasks, and it is sufficient that every such task migrates between two processors and at job boundaries only.

**Prototype development.** All the work described above is concerned with analytically determining the soft real-time guarantees that can be provided by various algorithms. In practical implementations, the relative merits of the algorithms will depend on the characteristics of the underlying platform and the application per se. While it may be possible to match applications to algorithms to some extent, real implementations are nonetheless needed to understand subtle issues, identify best implementation idioms, even pitfalls and fallacies, and evaluate the tradeoffs involved. Furthermore, if applications with varying characteristics are to be multiplexed, then it is not straightforward to identify the best scheduling paradigm. Hence, we are in the process of developing prototype implementations of the concerned algorithms and incorporating them in the kernel of Linux 2.6 running on an SMP with four processors and empirically evaluating the algorithms using representative applications.

# Other Research Contributions

In addition to research directed towards my dissertation, I have worked on the following.

---

4The minimum interval between consecutive invocations of a recurrent task is given by its period.
• Developing an improved polynomial-time schedulability test for hard real-time recurrent task systems scheduled under EDF on a uniprocessor [7].

• Analytically determining the real-time guarantees of a cache-cognizant scheduling algorithm for systems instantiated on multicore platforms [3].

3 Future Research Plan

Some problems that I plan on exploring in the future are described below.

3.1 Open Problems in Real-Time Scheduling on Multiprocessors

Scheduling algorithms and validation techniques for uniprocessor-based real-time systems have been investigated for over 25 years and are reasonably well understood. In comparison, despite significant advances on some fronts, techniques for multiprocessors are still considered nascent and are under development. With multiprocessors expected to be widely deployed in the future, such platforms will be used both in the design of stand-alone systems and as building blocks of complex distributed systems. Further, in many cases, distributed systems have real-time constraints. Hence, it is imperative that work on multiprocessor-based real-time scheduling be equally mature as that for uniprocessors. Towards this end, I plan on addressing some of the problems discussed below.

Improving current results. Though the problem of showing that tardiness under global EDF on multiprocessors is bounded has been closed as a part of my dissertation, the bounds derived have not been shown to be tight, and are somewhat high when the average task utilization is high, say, above 0.8. While systems with such high-utilization tasks may be quite rare in practice, the problem of tightening the bounds is, nevertheless, theoretically important, and appears to be necessary for proposing effective extensions for use with other task models and distributed systems. So, one of my near-term goals is to derive tighter tardiness bounds under EDF. Another goal along the same lines would be to more efficiently support hybrid task systems (described in Sec. 1.2).

Average-case analysis. My dissertation research is concerned with improving processor utilization on multiprocessors for soft real-time systems when the system workload is specified for the worst case. However, for some applications, workload can vary widely over time. For instance, in certain tracking and video-conferencing systems, the worst-case workload can be up to ten times the average load, and for such systems, reserving resources based on a worst-case specification would be extremely wasteful. On uniprocessors, much research on soft real-time systems has focused on allowing workload specifications that are less pessimistic than the worst case, and providing deterministic or probabilistic bounds on the deadline misses possible during times of transient overload. An obvious next step in the progression of the theory of soft real-time scheduling on multiprocessors is to consider analyzing systems with average-case specifications, and I expect this to be one of my focus areas.

Hierarchical scheduling. For reasons described in Sec. 1.1, developing a multiprocessor-based open-systems infrastructure [6], which would allow independently-authored applications with real-time needs to be multiplexed on a multiprocessor, such that each application is temporally isolated from other real-time or non-real-time applications becomes important. One way of providing isolation to applications is through hierarchical scheduling. Most prior work on hierarchical scheduling and open systems has focused mainly on uniprocessors or applications that require the processing capacity of at most one processor. Extending this work to encompass applications that exceed the capacity of a single processor is another possible avenue for future work.

Designing new algorithms. Though quite a few multiprocessor scheduling paradigms have been studied, there is clearly no one favorable algorithm for every kind of application, and the possibility of a not-yet-developed algorithm exhibiting better performance is not ruled out. Hence, I would like to analyze a few promising algorithms.
3.2 Real-Time Operating Systems

Though real-time operating systems is an active area of research, there is a considerable disparity between theory and practice in the real-time systems discipline. The rate at which theoretical results are produced is far greater than that at which those results are empirically evaluated and incorporated into production systems. For instance, despite the well-known theoretical superiority of EDF on uniprocessors, very few operating systems, if any, support EDF even on a uniprocessor. The disparity is more so in the case of multiprocessor systems, where more options, and hence, more interesting tradeoffs, are available for scheduling and other kernel activities. It is my plan to expand my initial work described earlier on the empirical evaluation of multiprocessor scheduling algorithms to a full-fledged plug-and-play test platform for evaluating new techniques proposed, identifying implementation bottlenecks, and investigating efficient and predictable multiprocessor-based real-time inter-process communication and synchronization mechanisms, and implementation techniques, including designing kernel data structures.

3.3 Execution-Time Analysis for Multiprocessors

A crucial part in validating the timing constraints of a real-time system lies in appropriately characterizing the workload, for which a priori knowledge of the worst-case or the average-case execution time of a program on a given hardware platform is needed. Estimating the execution time of a program is referred to as timing analysis and is generally performed in one of two ways: (i) measurements through simulations or (ii) static analysis of the program. While the former may seem simpler, exhaustively covering all paths in a program is often cumbersome and infeasible. In the latter, code is statically analyzed automatically using programmer-supplied hints (which are necessary in some cases due to connections to the halting problem), and a model of the microarchitecture of the underlying hardware, with the objective of arriving at safe, yet tight, bounds on execution times. On modern processors, architectural features provided for improving performance, such as pipelines, caches, and branch prediction and out-of-order processing mechanisms, lead to unpredictability and pose challenges in realizing this objective effectively. Most research on overcoming such challenges has been confined to uniprocessors, and has been very limited on multiprocessors. On multiprocessors, the timing-analysis problem is exacerbated due to additional features such as hardware threads, simultaneous multithreading, cache-coherence issues, and shared caches, and is in need of significant attention (especially in light of the ongoing shift toward multicore technologies). Therefore, one of my plans is to develop adequate expertise to fill the need for research in this area. Because timing analysis cuts across multiple areas of computer science that I am interested in, such as computer architecture, compilers, and programming languages, I view it as an ideal choice for diversifying research.

3.4 Resource Allocation in Real-Time Distributed Systems and Communication Networks

Apart from an increase in the demand for raw computation power, there is a continuous increase in application complexity as well. Though many real-time applications already exist that are distributed in nature, as with multiprocessor-based real-time systems, analysis techniques for distributed real-time systems are still under development, especially when the processing nodes are multiprocessor-based. For distributed real-time systems, in addition to resource-allocation algorithms that efficiently account for data and control dependencies among the processing, communication, and I/O resources, efficient and predictable distributed synchronization mechanisms are also required. I am interested in contributing to research focused on developing an integrated scheduling and analysis framework for such systems.

References


