On Recent Advances in Time/Utility Function Real-Time Scheduling and Resource Management

Binoy Ravindran
ECE Dept., Virginia Tech
Blacksburg, VA 24061, USA
binoy@vt.edu

E. Douglas Jensen
The MITRE Corporation
Bedford, MA 01730, USA
jensen@mitre.org

Peng Li
Microsoft Corporation
Redmond, WA 98052, USA
pengli@microsoft.com

Abstract

We argue that the key underpinning of the current state-of-the real-time practice — the *priority* artifact — and that of the current state-of-the real-time art — *deadline-based* timeliness optimality — are entirely inadequate for specifying timeliness objectives, for reasoning about timeliness behavior, and for performing resource management that can dependably satisfy timeliness objectives in many dynamic real-time systems. We argue that time/utility functions and the utility accrual scheduling paradigm provide a more generalized, adaptive, and flexible approach. Recent research in the utility accrual paradigm have significantly advanced the state-of-the-art of that paradigm. We survey these advances.

I. Introduction

Many emerging real-time embedded systems such as robotic systems in the space domain (e.g., NASA Jet Propulsion Laboratory's Mars Rover [1]) and control systems in the defense domain (e.g., phased array radars [2] and battle management systems [3]) operate in environments with dynamically uncertain properties. These uncertainties include transient and sustained resource overloads (due to context-dependent, activity execution times) and arbitrary activity arrival patterns. Nevertheless, such systems' desire the strongest possible assurances on activity timeliness behavior. Consequently, their non-deterministic operating situations must be characterized with stochastic or extensional (rule-based) models.

The most distinguishing property of such systems, however, is that they are subject to time constraints that are "soft" (besides hard) in the sense that completing an activity at any time will result in some (positive or negative) utility to the system, and that utility depends on the activity's completion time. These soft time constraints are subject to optimality criteria such as completing all time-constrained activities as close as possible to their *optimal* completion times—so as to yield maximal collective utility. The optimality of the soft time constraints is generally at least as mission- and safety-critical as that of the hard ones.

Jensen's time/utility functions (or TUFs) [4] allow the semantics of soft time constraints to be precisely specified. A TUF, which is a generalization of the deadline constraint, specifies the utility to the system resulting from the completion of an activity as a function of its completion time.

Figure 1 shows example time constraints from real applications specified using TUFs. Figures 1(a)–1(c) show some time constraints of two applications in the defense domain [5], [6].[1] The classical deadline is a binary-valued, downward "step" shaped TUF; Figure 1(d) shows examples.

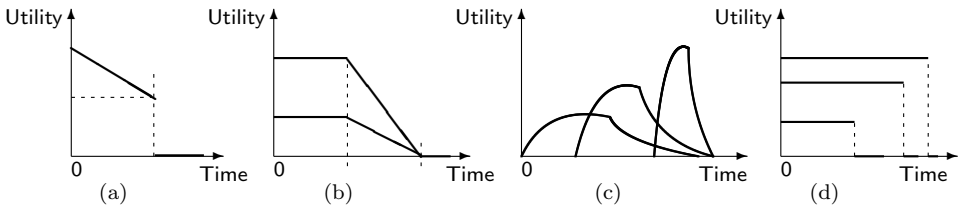

Fig. 1: Example TUF Time Constraints. (a): AWACS *association* TUF [5]; (b-c): Air defense *plot correlation*, *track maintenance*, & *missile control* TUFs [6]; (d): step TUFs.

When activity time constraints are expressed with TUFs, the scheduling optimality criteria are based on accrued activity utility—e.g., maximizing the sum of the activities' attained utilities, assuring satisfaction of lower bounds on activities' maximal utilities. Such criteria are called *utility accrual* (or UA) criteria,

[1] More real-world TUFs exist, including those with more complex shapes, but they appear in classified US DoD systems and hence are not in the public domain.

and sequencing (scheduling, dispatching) algorithms that consider UA criteria are called UA sequencing algorithms.

II. TUF/UA Paradigm Overcomes Existing Shortcomings

TUFs and UA scheduling overcome many of the shortcomings of the state-of-the-real-time-practice [7] and state-of-the-real-time-art [8]. Most application time constraints are expressed and handled in current real-time practice using the *priority* artifact. Priorities have significant shortcomings, including the following:

(1) Mapping application time constraints to priorities is generally not tractable. Doing so, results in significant loss of information, thereby making it difficult to dependably satisfy time constraints;

(2) Priority assignments are not modular for expressing urgency, because they require global knowledge of all priority assignments, which is often difficult to obtain during system development (e.g., due to organizational boundaries);

(3) Urgency of an application activity is typically orthogonal to the activity's relative importance, but a priority cannot express both (urgency and importance). This causes difficulties in managing resources during resource overloads, when completing activities that are more functionally important than those which are more urgent is often desirable.

Traditional real-time computing theory [8] overcomes these shortcomings by providing application designers with the direct abstraction of time constraints (instead of mapping time constraints to priorities), and using that abstraction for resource management. However, that theory is fundamentally limited to the deadline time constraint. In conventional real-time theory, deadlines are mapped to fixed priorities in algorithms such as RMA [9] and DMA [9] or are directly used for scheduling and resource management in algorithms such as EDF [10]. Deadlines and deadline-based scheduling have the following drawbacks:

(1) A deadline is either (i) a binary-valued expression in the sense that it is either met or not met; or (ii) a linear-valued expression for the penalty of lateness—i.e., the penalty of being late per unit time is constant regardless of the activity's lateness. Thus, deadlines also cannot distinguish between urgency and importance — a serious limitation during overload situations (as explained previously);

(2) Classical deadline-based scheduling [10] suffers from the (counter-intuitive) *domino* effect during overloads [11], as deadline-based algorithms (during overloads) favor activities that have a high likelihood for missing their deadlines over those that have a low likelihood for doing so.

(3) Deadlines cannot express time constraints that are non-binary and non-linear in the sense that the utility attained for activity completion *varies* (e.g., increases, decreases) with the activity completion time.

Timeliness optimality criteria that can be expressed using deadlines thus fall into the following classes: (1) the hard real-time criterion of satisfying all deadlines; (2) criteria that are expressed using number of missed deadlines (e.g., minimize deadline misses, satisfy an upper bound on missed deadlines); and (3) criteria that are expressed using lateness (e.g., minimize maximum lateness, satisfy an upper bound on lateness).

Note that the TUF/UA paradigm overcomes these shortcomings. In particular, UA criteria directly facilitate adaptive behavior during resource overloads, when (optimally or sub-optimally) completing activities that are more important than those which are more urgent is often desirable. UA algorithms that maximize summed utility under downward step TUFs (or deadlines), meet all activity deadlines during under-load situations—i.e., no deadline-misses occur during those situations [11]–[13]. When overloads occur, they favor activities that are more important (since more utility can be attained from them) than those which are more urgent. Thus, deadline scheduling's optimal timeliness behavior is a special-case of UA scheduling.

UA scheduling was first introduced by Jensen [14] in a classified military system. The first public reference was written by Jensen, Locke, and Tokuda in [4] and was later elaborated on in Locke's thesis [11] and Clark's thesis [12]. Subsequently, there have been very few efforts that built upon Locke's and Clark's algorithms. Most of the subsequent efforts focussed on downward step TUFs and sought to improve the timeliness behavior during overloads [15], [16]. Exceptions include [17], [18], which consider non-step TUFs.

While Locke's algorithm allowed almost arbitrary TUF shapes, it was restricted to resource-independent activities. Clark's algorithm, called DASA, advanced Locke's model by allowing activities to mutually exclusively share non-CPU resources under the single-unit resource request model. DASA also provides assurances on timeliness behavior such as optimal timeliness during under-load situations. However, DASA

is restricted to downward step TUFs. Algorithms in [17], [18] allow non-step, but unimodal TUFs. Further, they do not allow sharing of non-CPU resources.

III. Recent Advances in TUF/UA Real-Time Research

In spite of the generality and superiority of TUFs and UA scheduling, they have drawbacks that have apparently been impediments to their widespread usage. The most significant drawbacks include: (1) lack of more general timeliness assurances of TUF/UA systems, beyond the special-case assurance of optimal timeliness during under-loads for step TUFs; (2) lack of tool support for composing TUFs and for conducting UA timing analysis; (3) lack of UA algorithms that consider quality of service dimensions of embedded systems other than timeliness, such as power consumption and memory management; and (4) UA algorithms' higher overhead than priority/deadline-based algorithms.

Recent research on UA scheduling has largely overcome these drawbacks. Several new UA scheduling algorithms have been recently developed. Examples include Li's assurance-driven UA scheduling algorithms and protocols [19]–[21], Wu's energy-efficient UA scheduling algorithms [22]–[26], Cho's wait-free protocol for UA scheduling under non-blocking synchronization [27], [28], and Feizabadi's memory-aware UA scheduling algorithm [29], [30]. We overview each of these major advances in the subsections that follow.

A. Assurance-Driven UA Scheduling

Li's UA scheduling algorithms and protocols break significant new ground by providing more general assurances on timeliness behavior of TUF/UA systems such as assurances on individual activity timeliness behavior and system-wide, collective timeliness behavior. Li's algorithms and protocols consider stochastic activity models, where activity execution times and inter-arrival times are stochastically described. In particular, the algorithms and protocols consider a stochastic, activity arrival pattern called PUAM, which is a probabilistic generalization of the unimodal arrival model, and which subsumes most traditional arrival models (e.g., frame-based, periodic, sporadic, unimodal) as special cases. The algorithms and protocols allow activities to be subject to unimodal TUFs, and to mutually exclusively share non-CPU resources under the single-unit request model.

Li's approach includes off-line CPU bandwidth allocation, run-time scheduling, and run-time lock-based and lock-free resource access contention resolution. While CPU bandwidth allocation allocates CPU bandwidth share to activities, scheduling orders activity execution on the CPU. Li's lock-based resource access protocols resolve contention among activities (at run-time) for accessing shared resources, and bound the time needed for accessing those resources. Li's bandwidth allocation algorithms, scheduling algorithms, and resource access protocols collectively provide stochastic assurances on timeliness behavior including probabilistically satisfied lower bounds on each activity's maximal utility, in addition to a lower bound on the sum of activities' maximal utilities, besides maximizing the summed utility.

Lock-free synchronization generally incurs significant, additional time costs due to their "retry loops" [31]. Thus, Li analytically identifies the conditions under which UA scheduling with lock-free synchronization is advantageous with respect to satisfying utility lower bounds than UA scheduling with lock-based protocols, and vice versa. These analytical conditions establish the tradeoffs between UA scheduling with lock-free synchronization and UA scheduling with lock-based synchronization.

Li's algorithms and protocols thus allow application designers to specify lower bounds for individual activity and system-wide maximal utilities and associated probabilities for satisfying them, and analytically verify the probabilistic satisfaction of those bounds. The algorithms and protocols have been folded into a software tool called *Virginia Tech TUF/UA Design Toolkit*, in corporation with Tri-Pacific Software, Inc., for allowing designers to automatically compose TUFs and conduct UA analysis. The toolkit has been transitioned to The MITRE Corporation for use in US DoD programs. Ongoing work is developing a commercial version of this toolkit by integrating it with Tri-Pacific's RapidRMA [32].

Li's algorithms, protocols, and the tool thus directly solve a longstanding problem with TUF/UA scheduling, thereby significantly advancing the TUF/UA paradigm.

B. Energy-Efficient UA Scheduling

Wu's algorithms extend the TUF/UA paradigm in a quality of service dimension that is critical for emerging, mobile and portable, battery-powered embedded systems: *energy consumption*. Wu's algorithms solve, for the first time, the overlapped problem space that intersects: (1) UA scheduling under TUF time constraints, providing general assurances on individual and collective timeliness behavior; (2) scheduling activities under mutual exclusion resource constraints; and (3) CPU scheduling for reduced and bounded system-level energy consumption.

Similar to Li's, Wu's algorithms also consider a stochastic activity model, where activity execution times are stochastically described. Wu's algorithms allow activities to be subject to non-increasing unimodal TUFs, and to mutually exclusively share non-CPU resources under the single-unit request model. However, Wu's algorithms consider a system-level energy consumption model, where each system component's energy consumption is individually modeled and aggregated to obtain system-level energy consumption. For such a model, Wu's considers of the objective of (1) providing stochastic assurances on timeliness behavior, including probabilistically satisfied, lower bounds on each activity's maximal utility and a lower bound on the sum of activities' maximal utilities; and (2) maximize system-level energy efficiency; while ensuring that (a) the system's energy consumption never exceeds the specified energy budget; and (b) resource dependencies that arise due to mutually exclusive resource sharing are respected.

Wu presents a class of polynomial-time, DVS (dynamic voltage scaling)-based, UA scheduling algorithms toward this objective. The algorithms are called EUA [23], ReUA [24], EUA$\star$ [25], and EBUA [26]. EUA provides stochastic assurances on timeliness behavior and maximizes activities' attained utility and system-level energy efficiency. ReUA extends EUA's step TUF model with non-increasing TUFs, and extends EUA's independent task model with mutually exclusive resource sharing. Further, EUA$\star$ extends EUA's and ReUA's periodic arrival model with unimodal arrival; EBUA ensures that the system's energy consumption never exceeds the specified energy budget.

Wu's RUA algorithm [33] makes a major contribution by extending TUF/UA scheduling to mutually exclusive resource sharing under the *multi-unit* resource request model. The multi-unit model generalizes the single-unit model and subsumes the reader/writer lock model and the AND, OR, and AND-OR request models as special cases. Similar to DASA, RUA achieves optimal timeliness during under-loads, and upper bounds the resource access blocking time.

C. UA Scheduling with Wait-Free Synchronization

Most real-time systems use locks for concurrently, but mutually exclusively accessing shared data objects. However, locks have several disadvantages such as reduced concurrency (due to serialized access to shared objects), increased run-time overhead (due to increased context switching between activities blocked on shared objects and activities that hold locks of those objects), potential deadlocks due to failure of lock holders, and the need for a-priori knowledge of the order of accessing shared objects and/or ceilings of locks.

In [28], Cho considers wait-free synchronization for the single-writer, multiple-reader problem in real-time systems that are subject to TUF time constraints and UA optimality criteria. The motivation to consider wait-free synchronization for UA scheduling is to reap their advantages (e.g., greater concurrency, reduced run-time overhead, fault-tolerance) toward better optimization of UA criteria, such as increased attained utility during under-loads and overloads, and increased possibility for satisfying lower bounds on activities' maximal utilities. However, wait-free synchronization generally incurs significant, additional space costs due to their usage of multiple buffers.

In [28], Cho considers the single-writer, multiple-reader problem. Further, the work considers two UA optimality criteria: (1) the single criterion of maximizing activities' attained utilities; and (2) the dual criteria of (a) satisfying lower bounds on each activities' maximal utilities, and (b) maximizing activities' attained utilities. In [28], Cho presents a wait-free protocol for UA scheduling, building upon the the space-optimal, wait-free protocol for the single-writer, multiple-reader scenario developed earlier in [27]. Cho establishes that this protocol's buffer requirements is the absolute minimum that is needed to establish the safety and orderliness properties needed for wait-free synchronization. Furthermore, Cho analytically identifies the conditions under which UA scheduling with wait-free synchronization is advantageous (with respect to the targeted UA criteria) than UA scheduling with lock-based protocols, and vice versa. These analytical

conditions establish the tradeoffs between UA scheduling with wait-free synchronization and UA scheduling with lock-based synchronization.

D. Memory-Aware UA Scheduling

Feizabadi's MSA algorithm extend the TUF/UA paradigm in yet another critical, quality of service dimension for embedded systems: *memory management*. Embedded real-time systems are traditionally limited to main memory, static allocation, and dedicated fixed-size partitions, all of which contribute to predictability. However, dynamic memory allocation and dynamic, automatic memory deallocation (i.e., garbage collection) is highly attractive for dynamic, embedded real-time systems whose worst-case memory requirements cannot often be statically estimated.

The MSA algorithm allows activities to be subject to unimodal TUFs, and to mutually exclusively share non-CPU resources under the single-unit request model. The algorithm considers the scheduling objective of maximizing the summed utility. Further, the algorithm allows memory to be dynamically allocated, which are considered as explicit scheduling points. Similar to CPU overloads, when "memory overloads" occur (i.e., when memory demands of all activities cannot be simultaneously satisfied), MSA schedules those subset of activities from whom greater summed utility can be accrued. Similar to DASA, the algorithm provides the timeliness assurance of optimal timeliness during (CPU and memory) under-load situations with step TUFs.

E. Summary of Advances

Prototype implementations of many of these new UA algorithms have also been made. Implementations using a real-time POSIX-compliant framework [34] have shown that the algorithms have reasonable overheads even at a middleware level. For example, in [34], we show that DASA's overhead is in the magnitude of sub-milliseconds, even for a ready queue of over 60 threads on a 500MHz Pentium platform.

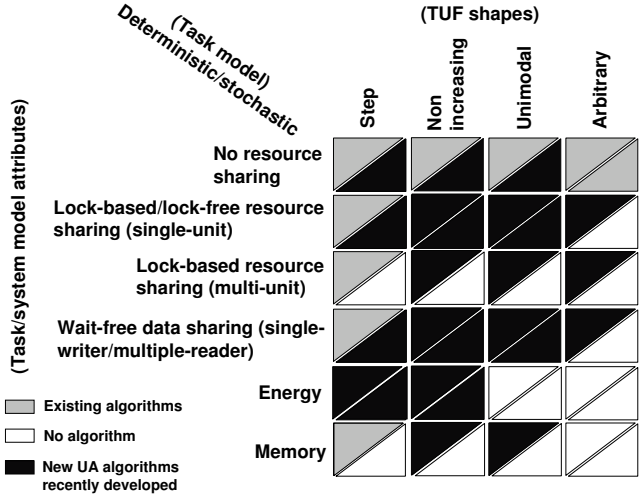

Fig. 2: Coverage of TUF/UA Scheduling

In Figure 2, we summarize the new TUF/UA scheduling advances using a *three*-dimensional matrix. Each matrix row represents an application/system model attribute such as no resource sharing, lock-based/lock-free/wait-free resource sharing, power, and memory. Each matrix column represents a class of TUF shape (e.g., step, non-increasing, unimodal, arbitrary). Further, each matrix element (identified by a row and a column position) is partitioned into deterministic and stochastic task models — the top left triangle of the element represents the deterministic model and the bottom right triangle represents the stochastic model. Algorithms developed in past research and those developed recently are now placed in the appropriate matrix elements. Observe that while traditional real-time algorithms are confined to the first column (for step TUFs), TUF/UA algorithms span all columns, illustrating their wider coverage.

IV. Conclusions

We conclude that the TUF/UA paradigm is important in dynamic real-time systems, and that the state of the art has recently significantly advanced. A variety of new scheduling and resource management algorithms have been recently devised. These algorithms overcome many of the traditional drawbacks of the original approaches, and expand the coverage of TUF/UA scheduling. We are unaware of any other timeliness model and scheduling techniques that can satisfy the timeliness requirements of dynamic non-deterministic real-time systems as well as TUF/UA techniques can.

We believe that emerging real-time systems can significantly benefit from using the TUF/UA paradigm. This will allow such real-time systems to leverage the adaptivity and flexibility of the paradigm, yet reap

the advantages of traditional real-time theory (e.g., timeliness optimality during under-loads, statistical timeliness assurances).

Acknowledgements

The work summarized in this paper was sponsored by the US Office of Naval Research under Grant N00014-00-1-0549 and The MITRE Corporation under Grant 52917.

References

[1] R. K. Clark, E. D. Jensen, and N. F. Rouquette, "Software organization to facilitate dynamic processor scheduling," in *IEEE Workshop on Parallel and Distributed Real-Time Systems*, April 2004.

[2] GlobalSecurity.org, "Multi-Platform Radar Technology Insertion Program," http://www.globalsecurity.org/intell/systems/mp-rtip.htm/.

[3] ——, "BMC3I Battle Management, Command, Control, Communications and Intelligence," http://www.globalsecurity.org/space/systems/bmc3i.htm/.

[4] E. D. Jensen, C. D. Locke, and H. Tokuda, "A Time-Driven Scheduling Model for Real-Time Systems," in *IEEE Real-Time Systems Symposium*, December 1985, pp. 112–122.

[5] R. Clark, E. D. Jensen, A. Kanevsky, J. Maurer, P. Wallace, T. Wheeler, Y. Zhang, D. Wells, T. Lawrence, and P. Hurley, "An Adaptive, Distributed Airborne Tracking System," in *IEEE Workshop on Parallel and Distributed Real-Time Systems*, ser. LNCS, vol. 1586. Springer-Verlag, April 1999, pp. 353–362.

[6] D. P. Maynard, S. E. Shipman, R. K. Clark, J. D. Northcutt, R. B. Kegley, B. A. Zimmerman, and P. J. Keleher, "An Example Real-Time Command, Control, and Battle Management Application for Alpha," Department of Computer Science, Carnegie Mellon University, Archons Project TR-88121, December 1988, http://www.real-time.org.

[7] IEEE and OpenGroup, "The open group base specifications issue 6," 2001.

[8] J. W. S. Liu, *Real-Time Systems*. New Jersey: Prentice Hall, 2000.

[9] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[10] W. Horn, "Some Simple Scheduling Algorithms," *Naval Research Logistics Quaterly*, vol. 21, pp. 177–185, 1974.

[11] C. D. Locke, "Best-Effort Decision Making for Real-Time Scheduling," Ph.D. dissertation, Carnegie Mellon University, 1986, CMU-CS-86-134, http://www.real-time.org.

[12] R. K. Clark, "Scheduling Dependent Real-Time Activities," Ph.D. dissertation, Carnegie Mellon University, 1990, CMU-CS-90-155, http://www.real-time.org.

[13] H. Wu, B. Ravindran, E. D. Jensen, and U. Balli, "Utility Accrual Scheduling under Arbitrary Time/utility Functions and Multi-unit Resource Constraints," in *RTCSA*, August 2004, pp. 80–98.

[14] M. G. Gouda, Y.-W. Han, E. D. Jensen, W. D. Johnson, and R. Y. Kain, "Radar scheduling: Section 1, the scheduling problem," in *Distributed Data Processing Technology, Applications of DDP Technology to BMD: Architectures and Algorithms*. Minneapolis, MN: Honeywell Systems and Research Center, September 1977, vol. IV, no. National Technical Information Service (NTIS) ADA047475, ch. 3.

[15] G. Koren and D. Shasha, "D-Over: An Optimal On-line Scheduling Algorithm for Overloaded Real-Time Systems," in *IEEE Real-Time Systems Symposium*, December 1992, pp. 290–299.

[16] D. Mosse, M. E. Pollack, and Y. Ronen, "Value-Density Algorithm to Handle Transient Overloads in Scheduling," in *IEEE Euromicro Conference on Real-Time Systems*, June 1999, pp. 278–286.

[17] K. Chen and P. Muhlethaler, "A Scheduling Algorithm for Tasks Described by Time Value Function," *Journal of Real-Time Systems*, vol. 10, no. 3, pp. 293–312, May 1996.

[18] J. Wang and B. Ravindran, "Time-Utility Function-Driven Switched Ethernet: Packet Scheduling Algorithm, Implementation, and Feasibility Analysis," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 2, pp. 119–133, February 2004.

[19] P. Li, "Utility Accrual Real-Time Scheduling: Models and Algorithms," Ph.D. dissertation, ECE Dept., Virginia Polytechnic Institute and State University, 2004, http://scholar.lib.vt.edu/theses/available/etd-08092004-230138/.

[20] P. Li, H. Wu, B. Ravindran, and E. D. Jensen, "On Utility Accrual Resource Management with Assured Timeliness Behavior," in *IEEE Euromicro Conference on Real-Time Systems (under review)*, July 2005.

[21] P. Li, B. Ravindran, and E. D. Jensen, "Utility Accrual Resource Access Protocols with Assured Timeliness Behavior for Real-Time Embedded Systems," in *ACM Conference on Languages, Compilers, and Tools for Embedded Systems (under review)*, June 2005.

[22] H. Wu, "Energy-Efficient Utility Accrual Real-Time Scheduling," PhD Dissertation Proposal, ECE Dept., Virginia Polytechnic Institute and State University, 2004, http://www.ee.vt.edu/~realtime/wu-proposal04.pdf.

[23] H. Wu, B. Ravindran, E. D. Jensen, and P. Li, "Cpu scheduling for statistically-assured real-time performance and improved energy-efficiency," in *IEEE/ACM/IFIP Conference on Hardware/Software Codesign and System Synthesis*, September 2004, pp. 110–115.

[24] ——, "Energy-Efficient, Utility Accrual Scheduling under Resource Constraints for Mobile Embedded Systems," in *ACM EMSOFT*, Sept. 2004, pp. 64–73.

[25] H. Wu, B. Ravindran, and E. D. Jensen, "Energy-Efficient, Utility Accrual Real-Time Scheduling Under the Unimodal Arbitrary Arrival Model," in *ACM Design, Automation, and Test in Europe (DATE)*, March 2005.

[26] ——, "Utility Accrual, Real-Time Processor Scheduling with Energy Bounds," in *ACM Design Automation Conference (under review)*, June 2005.

[27] H. Cho, B. Ravindran, and E. D. Jensen, "A space-optimal, wait-free real-time synchronization protocol," in *IEEE Euromicro Conference on Real-Time Systems (under review)*, July 2005.

[28] ——, "Utility accrual real-time scheduling under shared data: Lock-based versus non-blocking synchronization," 2005, in preparation.
[29] S. Feizabadi, "Utility Accrual Real-Time Memory Management: Algorithms for Automatic, Dynamic, Explicit Memory Management and Garbage Collection," PhD Dissertation Proposal, Department of Computer Science, Virginia Polytechnic Institute and State University, 2004, http://www.ee.vt.edu/~realtime/feizabadi-proposal04.pdf.
[30] S. Feizabadi, B. Ravindran, and E. D. Jensen, "Msa: A memory-aware utility accrual scheduling algorithm," in *ACM Symposium On Applied Computing (Track on Embedded Systems)*, March 2004, to appear.
[31] J. H. Anderson, S. Ramamurthy, and K. Jeffay, "Real-time computing with lock-free shared objects," *ACM TOCS*, vol. 15, no. 2, pp. 134–165, May 1997.
[32] Tri-Pacific Software, Inc., "Rapid rma," http://www.tripac.com/NEW/prod-fact-rrm.html, last updated, October 1999.
[33] H. Wu, B. Ravindran, E. D. Jensen, and U. Balli, "Utility accrual scheduling under arbitrary time/utility functions and multiunit resource constraints," in *IEEE Real-Time and Embedded Computing Systems and Applications*, August 2004.
[34] P. Li, B. Ravindran, S. Suhaib, and S. Feizabadi, "A formally verified application-level framework for real-time scheduling on posix real-time operating systems," *IEEE Transactions on Software Engineering*, vol. 30, no. 9, pp. 613 – 629, September 2004.