Application QoS-Based Time-Critical Automated Resource Management in Battle Management Systems

E. Douglas Jensen
The MITRE Corporation
http://www.real-time.org

Abstract

This paper summarizes some of our unclassified work on concepts and techniques for performing automated run-time time-critical resource management (especially scheduling) in large scale, dynamic, control systems such as for battle management. The approach described here is based on application-level quality of service metrics, such as track quality and weapon spherical error probable. These metrics are used to derive parameters for thread time constraints in the form of time/utility functions. Threads are scheduled according to application-specific optimality criteria that seek to maximize accrued utility to the system. Two worked examples illustrate the cost-effectiveness of this approach for this class of application.

1. Introduction

A system provides services, each of which may have various qualities. Each service usually has multiple dimensions of quality. Generic examples include timeliness, availability, and security. Quality of service (QoS) is traditionally associated only with network communications – e.g., bandwidth, bit error rate, latency and jitter. But QoS is an inherently general concept that can be employed at any levels of a system and enterprise.

In many systems, complex dynamic service – and thus resource – conflicts and dependencies arise. Resolution of the resource contention affects the quality of the provided services to the users and thus the success of the mission and the enterprise. Not all service requests can always be perfectly satisfied; adaptive situation and application-specific resource management decisions are required.

Some of the service requests are time-critical – the quality of the service depends on when it is performed. The timeliness of these services may have potentially drastic impacts on the enterprise and its mission, including survival of property and human lives.

By popular convention, the term "safety critical" is reserved for very simple, static, systems that can be developed in accordance with DO-178B and other similar standards. But many large scale, dynamic, systems – such as for combat platform mission management, and battle management – have at least as much potential for human harm as do those "safety critical" ones, yet they are outside the scope of DO-178B et al., and call for entirely new and different (and intrinsically less predictable) approaches to high assurance. In the absence of substantive results on that problem, assurance must be sought by design methodologies, simulations, and extensive testing.

In this paper we discuss resolving resource contention by adaptively maximizing total application-level quality of service (AQoS) of a set of services according to application- and situation-specific criteria. Multimedia and similar applications popular in the real-time research community have AQoS metrics, but they are isomorphic to the QoS metrics of traditional networking, and so do not present significantly new resource management challenges and opportunities.

We are concerned with managing application, as well as system, resources (hardware and software) – above, as well as at, the network level – using AQoS metrics, such as (in most systems of interest to us) track quality and weapon spherical error probable.

Application designers often think in terms of what we refer to as AQoS metrics, but not in a general and methodological way. Instead, they consider certain metrics and use their domain expertise to attempt to aggregate these into the proper "tuning" of the system. System and resource management (OS, middleware) software designers usually do not think in terms of AQoS. Thus, there are needs for systematizing and quantifying: the expression of AQoS metrics, and the potential trade-offs among them, to resource management software; and managing resources adaptively by employing AQoS to dynamically optimize the system's behavior according to the users' wishes.

We are particularly interested in helping meet those needs in the context of satisfying application timeliness requirements. We are focused on using AQoS metrics in defining how each service's timeliness contributes to its utility to the current state of the mission, and sequencing access to shared resources based in part on maximizing the utility accrued from the set of services.

2. Dynamic Time-Critical Control Systems

Many important time-critical control systems do not fit the "real-time" stereotype of: being small scale, static, periodic, centralized; performing autonomic monitoring and control of simple devices; and having time frames in the microsecond and millisecond range. Instead, they: are large scale in various dimensions (e.g., ten's of millions of lines of source code), dynamic, "mesosynchronous," and distributed; perform closed loop automated control at any level(s) of an enterprise; and typically operate in time frames of hundreds of milliseconds to several minutes.

In mesodynamics [1], the prefix *meso* refers to the middle ground between classical physics and quantum physics. By *mesosynchronous* real-time systems we mean those that are in the middle ground between totally synchronous – in the sense of having only static, periodic, time-driven, activities (e.g., [2]) and totally asynchronous – in the sense of having only dynamic, aperiodic, event-driven, activities. The real world has many important real-time control systems at various points in this mesosynchronous middle ground. The derivation of "mesosynchronous" from "mesodynamics" also reflects that synchronous real-time computing, like classical physics, is relatively well understood – while asynchronous real-time computing, like quantum mechanics, requires a paradigm shift and thus a great deal more research and development on methodologies and formalisms.

The application domain of primary interest to us is defense. Defense systems and applications have always had by far the most challenging time-critical resource management problems – and are MITRE's predominant focus. Examples include: multi-role and multi-mode sensors (e.g., phased array radars such as MP-RTIP [3]), combat and surveillance platform management (e.g., the new E-10A (MC2A) aircraft [4], UAV's and UCAV's [5], the DD(X) class of destroyers [6], the Army Future Combat System vehicle cluster [7], battle management and mission management, (e.g., time-critical targeting [8]), command and control (e.g., AWACS [9]), and network-centric warfare [10].

Time-critical resource management in network-centric warfare is exemplified by the Affordable Moving Surface Target Engagement (AMSTE) system [11], a notional non-classified depiction of which is in Figure 1. AMSTE includes netted airborne and spaceborne Ground Moving Target Indication (GMTI) sensors for tracking a target on the ground, a missile launching fighter aircraft, and a missile guidance aircraft (which is usually one of the sensor aircraft). The targets are moving or alternately moving and stopping. The AMSTE objectives are: to maintain target track from nomination through engagement, from tactically significant standoff ranges; and to provide precision fire control updates to missiles in flight. These objectives require time-critical orchestration of multiple sensors, data links, and the missile. The control loop has about a one second time constraint for sensor to tracker to missile, and about an eight Hz update rate to the missile (the details of an actual AMSTE system are classified).

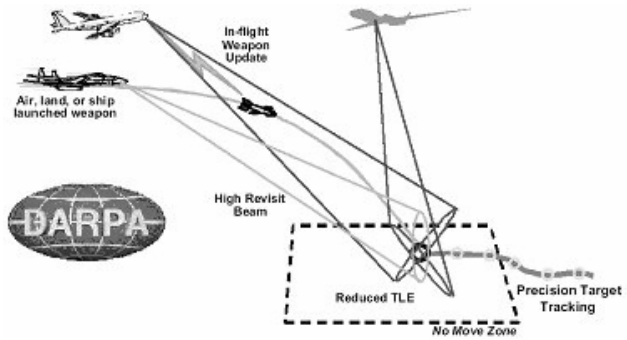

Figure 1. Notional depiction of an AMSTE system

The milliseconds-to-minutes time scale is in a "no man's land" of time-critical resource management theory and practice, between conventional real-time scheduling, and classical logistics (e.g., shop) scheduling with deadlines. Conventional real-time scheduling concepts and techniques: are for static (deterministic), periodic, and (to a degree) sporadic activities, and cannot handle arbitrary asynchrony and dynamic changes in load and resources; and presume time-frames to be in the microseconds to milliseconds range, which limits resource management algorithm capability. Classical logistics scheduling concepts and techniques [12] include: stochastic as well as deterministic models; but time-frames are in the minutes to hours (usually too computationally intensive for on-line automated resource management); the use of lateness (deadline - completion time) as the timeliness metric, the linearity of which severely limits time constraint expressiveness.

3. Time/Utility Functions and Utility Accrual

In 1977 Jensen created a novel scheduling paradigm for dynamic time-critical systems [13] for the U.S. Army Safeguard Command's AN/FPQ-16 Perimeter Acquisition Radar Characterization System [14], located at what is now the USAF Space Command's Cavalier Air Force Station, in Cavalier, North Dakota. The system's radar performance was not up to expectations. Simulated performance with this new scheduling paradigm improved so much that the paradigm was deemed to possibly violate SALT II, and hence was not deployed. From then until now, Jensen and his collaborators have continued to develop, refine, and apply the paradigm. At Carnegie Mellon's Computer Science Department, two of Jensen's

Ph.D. students wrote their theses on this topic [15, 16] – devising new scheduling algorithms, proving theorems about them, simulating them, and implementing them in our Alpha real-time distributed OS kernel [17]. Subsequent related work has also been done by other researchers (e.g., [18, 19, 20, 21]).

The two keystone concepts in our paradigm are *time/utility functions* (originally called *time/value* and then *benefit*, functions) and *utility accrual* sequencing (scheduling, dispatching) optimality criteria [13, 22].

A time/utility function (TUF) expresses the utility to the system – derived from AQoS metrics – of completing an activity (e.g., service) as an application- or situation-specific function of when it completes. Deadlines are a simple special case, where maximum utility is achieved if the activity completes by its deadline, and zero (or some negative) utility is achieved if it completes after its deadline. Four example TUF's are depicted in Figure 2.

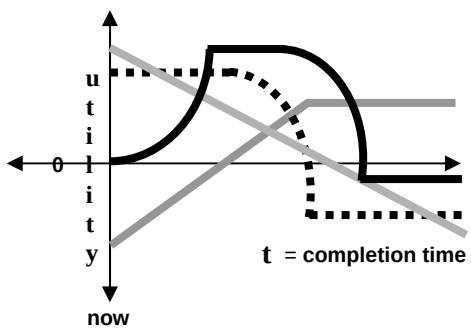

Figure 2. Four example time/utility functions

Utility accrual (UA) based sequencing algorithms sequence activities according to optimality criteria based on accruing utility – such as maximizing the sum of the utilities (plus satisfying dependencies such as precedence, resource constraints, etc.). The algorithms take into account the known or expected durations of the activities, and may be either deterministic or non-deterministic (e.g., stochastic).

3. Worked Examples

To illustrate the use of this paradigm, we summarize two significant (unclassified) demonstration applications we implemented successfully in collaboration with application domain experts.

3.1 AWACS Surveillance Tracker

The first demonstration application is an AWACS surveillance tracker, implemented jointly by the MITRE Corporation and the Open Group [23]. AWACS is an airborne radar system with varied missions, including air surveillance. Surveillance missions generate aircraft tracks for battle management and command and control. Sensor reports arrive at the tracker periodically, and the number of sensor reports can vary from radar sweep to radar sweep. When new sensor reports arrive, a tracker associates the information contained in the sensor reports with the current estimated states of the tracks in its track file to produce updated track records that represent the tracking system's estimate of the state of the airspace covered by its sensors [24].

A typical surveillance tracking application is comprised of several stages. First it applies a divide-and-conquer technique consisting of a pair of algorithms called *gating* and *clustering* to split the association problem into smaller, mutually exclusive, subproblems, called *clusters*. Then data association solves each of these subproblems individually by correlating sensor reports in a cluster with tracks in that cluster. The final stages of a tracker – *prediction* and *smoothing* – are used to predict the next position, velocity and other parameters for each tracked object based upon its track history and the results of data association.

The tracker's workload is a function of both the current track states and the new sensor-report data. It is very common for there to be too many radar reports for the computing system to process, which in the current AWACS system causes sectors of the sky to "go blank". Higher performance computing only helps somewhat, because the potential workload and the need for ever-better tracking algorithms will always exceed the computing capacity. Currently, AWACS operators have knowledge-intensive manual work-arounds for certain overload situations. This project's goal was to improve the tracker's – and hence the surveillance mission's – performance by enabling the tracker to do a better job of selecting the most useful subset of incoming data that it could feasibly process.

Our approach to achieving that goal was to employ a computing system (in this case, operating system) scheduling facility instead of mechanisms internal to the tracker as extant AWACS surveillance trackers do.

This led to the experimental tracker being multithreaded, including a separate thread per data association (i.e., per cluster). The association algorithms are the most computationally demanding ones, and were encapsulated in an object that was extracted from the rest of the tracker. This object can be – and has been – run on other nodes in a distributed tracking system. Moreover, there is provision in the tracker to select the association server to be used on a cluster-by-cluster basis. For brevity and clarity, in this paper we consider only the association threads and make the simplifying assumption that each

cluster has only a single track.

There are two sensors (radar and IFF) sweeping 180º out of phase with a 10-second period, which suggests the association TUF has a *critical time* at the 10-second period length, at least two distinct non-zero utilities before the critical time, and a third distinct, lower, utility after the critical time.

Prior to the critical time, processing a sensor report for one of these tracks in under five seconds (half the sweep period) would provide better data for the corresponding report from the out-of-phase sensor. So the utility decreases with time. The TUF had to decrease linearly due to an implementation artifact in this experimental system – the OS (OSF/RI's MK7 [25]) TUF scheduling algorithm allowed only one critical time. The slope, and thus U_2, were derived empirically (systematizing this application-specific process is one of our current research topics).

After the critical time, utility U_3 is zero, because newer sensor data has probably arrived if the processing load in one sensor sweep period is so heavy that it couldn't be completed, probably the load will be about same in next period, so there will be no capacity to also process data from the previous sweep. The resulting TUF shape is shown in Figure 3.

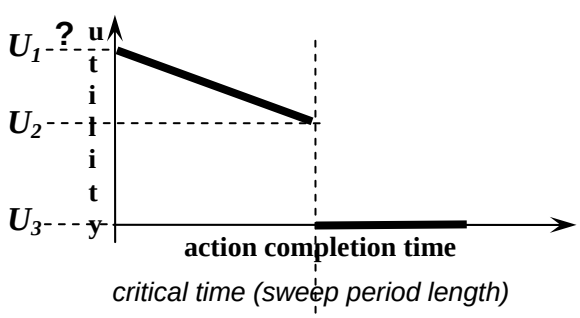

Figure 3. Association time/utility function shape

Next, to determine the utility value U_1, we used well-established surveillance tracker domain knowledge that the tracker should perform preferential processing for tracks that:

- are in danger of being dropped
- the user has identified as "important"
- have poor state (position and velocity) estimates
- are maneuvering
- potentially pose a high threat
- are moving at high speed.

Conventionally, tracker domain experts aren't provided with computation technology that would give them incentives to use their knowledge to understand and express behavioral options and perform trade-offs to resolve conflicts among these preferences in the face of dynamic uncertainties (i.e., gracefully handling overloads) that plague current trackers. Our intention was to provide technology that would utilize tracking domain expertise sufficient to automate the adaptive assignment of the right resources to the right tasks at the right time.

We chose three (to limit the complexity of the experimental system) AQoS metrics for an AWACS surveillance tracking application (a decision with which the surveillance radar domain experts concurred):

- *Quality* – 0 to 7: a traditional measure of the amount of recent sensor data incorporated in a track record, and incremented or decremented after each radar scan
- *Accuracy* – "high" or "low": a measure of the uncertainty of the estimate of a track's position and velocity and derived from Kalman filter processing
- *Importance* – "high" or "low": traditionally, operator-identified based on geography, threat, and other characteristics.

The twelve combinations of these three AQoS metrics are diagrammed in Figure 4.

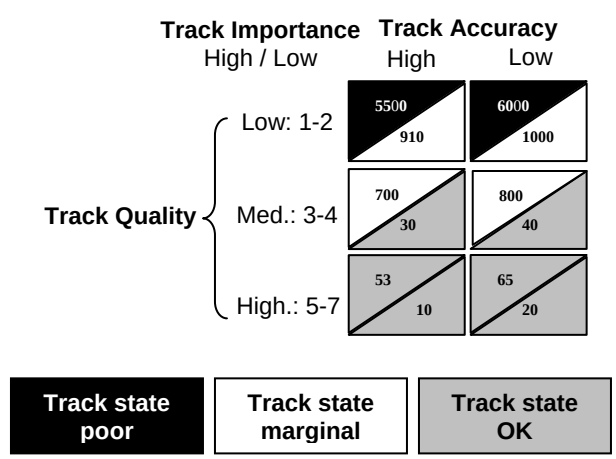

Figure 4. The 12 Combinations of track AQoS's and their relative utilities U_1

The initial utility U_1 of an association for a track report is derived from track AQoS metrics by gedanken experiments (again, systematizing this application-specific process is one of our current research topics). The utilities range from a low of 10 to a high of 6000 – meaning that 600 times more utility is gained by performing an association for a low accuracy, high

importance, poor quality track, than for a high accuracy, low importance, high quality track.

All the association (and other) threads are scheduled based on their utility functions. For the association threads, the tracking application selects the established TUF from the scheduler's library of shapes. The tracking application does a look-up in the utility U_1 table for each association thread before calling the scheduler. A utility accrual based processor scheduling discipline (in the OS, in this system) schedules threads according to a heuristic that attempts to maximize total accrued (in this case, summed) utility.

This surveillance tracking application was implemented on a research OS (the Open Software Foundation's MK7) having a scheduling framework that allowed different disciplines to be used. Figure 5 shows two critical tracker AQoS metrics, the number of dropped tracks and track quality, for more and less important threads, using three scheduling disciplines: FIFO (the discipline used in the currently deployed AWACS tracker); priority (the predominant discipline in real-time computing systems); and a utility accrual discipline.

In the measured cases shown, under non-overload conditions, the tracker handled all tracks as expected and delivered high AQoS in both metrics for all tracks. In an overload scenario, such as when the tracker can only process about 33% of the input, the system delivered essentially perfect track quality for the more important tracks, while delivering a reasonable track quality for the less important tracks. (In some respects, this overload level can be likened to a system where the probability of detecting an airborne object is about 33%.) When the tracker was further constrained so that it could only process about 10% of the input, the system finally dropped some tracks – from the less important track class. No important tracks have been dropped during our demonstrations. In addition, the demonstrations have shown that the tracker also adapts when new resources are added. In that case, which we demonstrate by loosening the time constraints on association processing, the prototype automatically delivers approximately the maximum achievable AQoS.

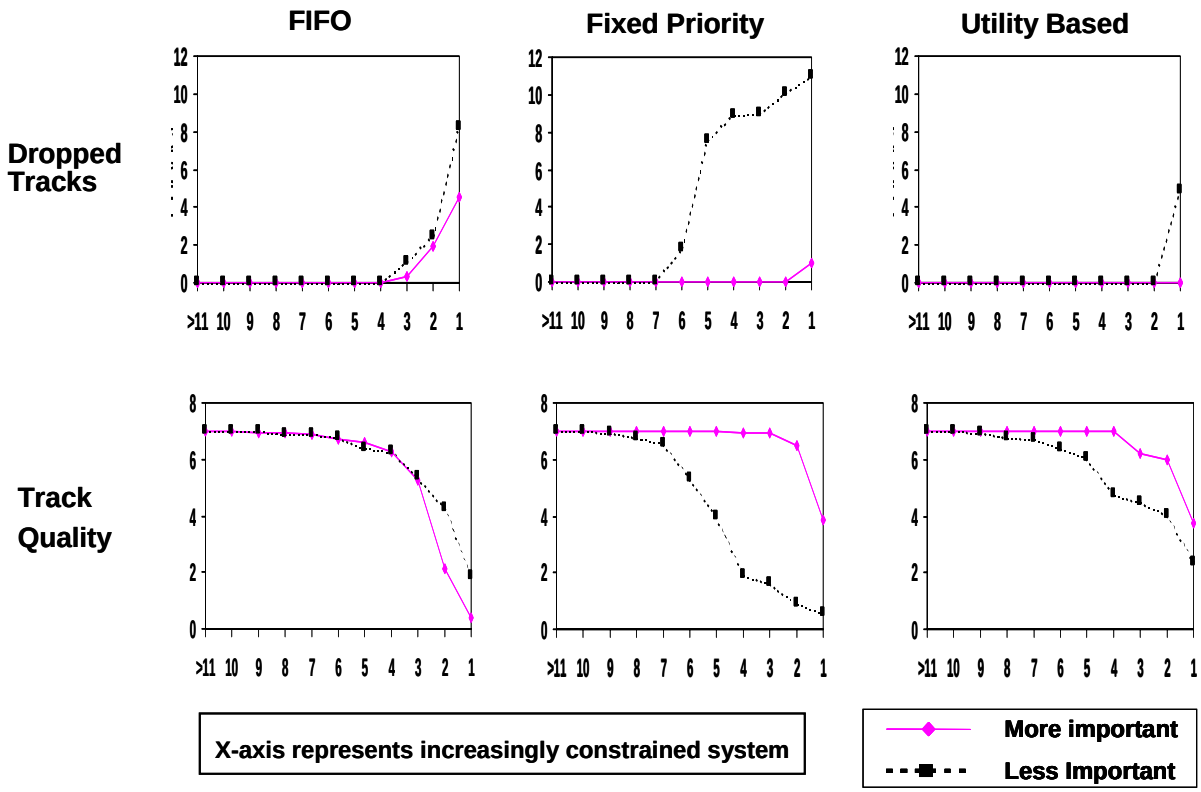

Figure 5. Comparison of AWACS surveillance tracker AQoS metrics for different scheduling disciplines

Note that these performance results are not easily obtainable with conventional static priority tracking systems. In priority-based trackers, track priorities might reasonably be set according to track importance, where high importance implies high priority. In our tracker, scheduling decisions are based on both importance and timeliness, and even relatively unimportant tracks can have very high application utility in a surveillance

mission – a valuable outcome that would not be possible with straightforward priority-based scheduling.

3.2 Battle Management for Coastal Air Defense

The second application is a notional battle management system for coastal defense from cruise missiles and bombers [26]. It was implemented collaboratively by the General Dynamics Corporation and Jensen's Archons Project at Carnegie Mellon University's Computer Science Department. It is included here to further illustrate the dynamic adaptivity possible with time/utility functions, which would be difficult to achieve with priorities or even deadlines.

Here we will consider a subset of this system's mission – to destroy incoming hostile cruise missiles (for brevity we will disregard the hostile bombers), by using guided interceptor missiles (for brevity we will ignore the defensive surface-to-air missiles). There may be more cruise missiles and decoys than can be intercepted, due to the number of cruise missiles, insufficient interceptors, and insufficient computational resources for battle management. Cruise missiles maneuver during flight, but do not try to evade the interceptors. Interceptors are guided by airborne defenders using airborne (AWACS, etc.), spaceborne, and ground based, sensor platform data.

The cruise missile defense (CMD) application quality of service metric here is the weapon spherical error probable (WSEP). Spherical error probable (SEP) is the radius in meters of a sphere centered on a point, within which the true value of an estimated point will lie with a probability of 0.5. The WSEP is an SEP radius around the cruise missile chosen such that if the interceptor gets within that distance, the interceptor's terminal guidance sensor system can take over and cause the interceptor to either impact the cruise missile or detonate close enough to the cruise missile to destroy it.

In addition to minimizing WSEP, other CMD AQoS metrics include (but are not addressed here):

- intercept before the cruise missile reaches its target
- intercept as far as possible from the cruise missile's target (due to EMP, etc.)
- intercept away from an adverse geographical location (e.g., over a populated area)

Several factors must be taken into account for this CMD application, including:

- guidance updates to the interceptor are repetitive but not necessarily periodic or even sporadic (an interceptor's path need not be straight and smooth), and need to occur more often as the interceptor gets closer to the cruise missile
- it becomes more important for interceptor guidance updates to use the most recent information about the positions and velocities of the missiles as the distance between the cruise missile and the interceptor missile decreases
- some cruise missile targets, and thus cruise missiles, are more important than others.

This experimental coastal air defense application consists of a number of activities, as depicted in Figure 6.

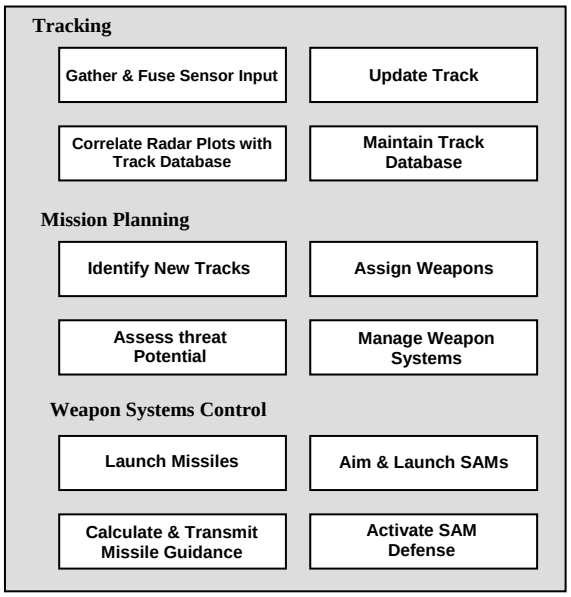

Figure 6. The coastal air defense battle management activities

Some activities resemble corresponding activities in the AWACS surveillance tracker. The plot correlation and track database maintenance threads have critical times corresponding to the radar frame arrival rate. In both cases, it is better if the processing is completed before the next frame of sensor data arrives. It is acceptable for the processing to slip as much as one additional time frame under extreme overload situations. The plot correlation activity has a much greater utility to the system under overload conditions. The TUF's for those two threads are shown in Figures 7 and 8.

But the timeliness requirements for the interceptor missile control threads are more complex because they vary over the course of an interception engagement. After an interceptor is launched, the guidance control threads must issue timely aperiodic course updates to ensure a successful intercept. The required timeliness of these updates, and the importance of completing the course corrections at the desired time, change as the distance

decreases between the interceptor and the cruise missile, and between the cruise missile and the coastline. Figure 9 shows three snapshots of an interceptor missile control thread as it adapts its shape, progressing from the right-most (launch phase) to the left-most (intercept) phase; the number of these adaptations is variable. To accomplish this adaptation, three TUF shape parameters of each interceptor's control thread are changed during the engagement, as seen in Figure 10.

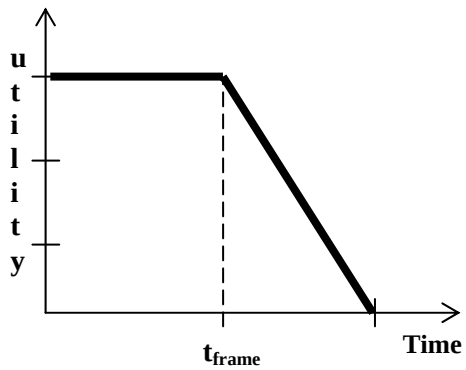

Figure 7. The plot correlation time/utility function

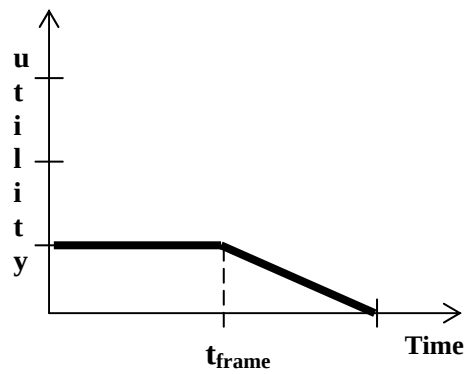

Figure 8. The track database maintenance time/utility function

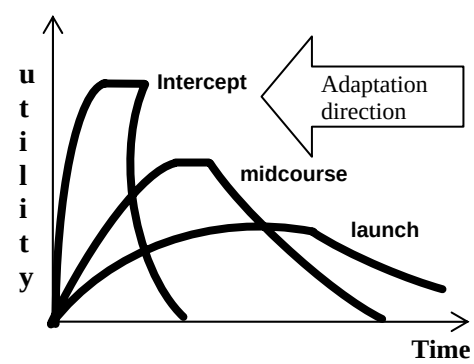

Figure 9. The interceptor time/utility function shape is adapted during the engagement

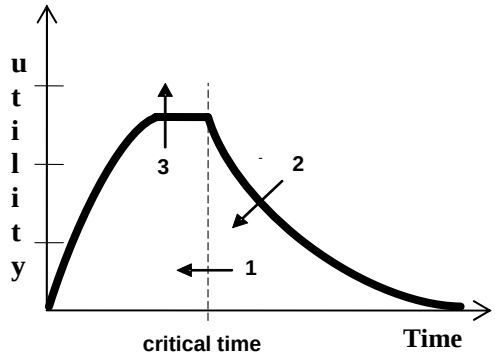

Figure 10. Three shape parameters of the interceptor time/utility function are adapted during the engagement

This richly expressive adaptation is extremely difficult to achieve by manipulating priorities or deadlines.

4. Conclusion

These two unclassified experimental demonstrations of using time/utility functions and utility accrual-based scheduling optimality criteria have been very successful. Our current research objective on this topic is to advance the concepts and techniques, and the applications, of this paradigm. We are devising new resource management algorithms, proving timeliness and other properties, doing simulations of them, building and measuring experimental implementations of them, devising a methodology for users to apply the paradigm, and implementing them in a number of DoD and other (e.g., NASA) programs.

5. References

[1] R.D. Peters, http://mathforum.org/library/view/16558.html

[2] H. Kopetz, *Real-Time Systems, Design Principles for Distributed Embedded Applications*, Chapter 8, "The Time-Triggered Protocols," Kluwer Academic Publishers, 1997.

[3] http://www.globalsecurity.org/intell/systems/mp-rtip.htm

[4] http://esc.hanscom.af.mil/esc-mc2a/

[5] http://www.globalaircraft.org/planes/?type=uav

[6] http://peoships.crane.navy.mil/ddx/

[7] http://www.darpa.mil/fcs/

[8] http://www.globalsecurity.org/space/systems/bmc3i.htm

[9] http://www.globalsecurity.org/military/systems/aircraft/e-3.htm

[10] http://www.dodccrp.org/ncwPages/ncwPage.html

[11] http://www.globalsecurity.org/military/systems/munitions/amste.htm

[12] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 2nd edition, Prentice Hall, 2002, ISBN 0-130-28138-7.

[13] M. G. Gouda, Y-W. Han, E. D. Jensen, W. D. Johnson, and R.Y. Kain, *Distributed Data Processing Technology, Vol. IV, Applications of DDP Technology to BMD: Architectures and Algorithms*, Honeywell Systems and Research Center, Minneapolis, MN. September 1977.

[14] http://srmsc.org/

[15] C. D. Locke, *Best-Effort Decision Making for Real-Time Scheduling*,, Ph.D. Thesis, CMUCS-86-134, Department of Computer Science, Carnegie Mellon University, 1986.

[16] R. K. Clark, *Scheduling Dependent Real-Time Activites*, Ph.D. Thesis, CMUCS-90-155, School of Computer Science, Carnegie Mellon University, 1990.

[17] R. K. Clark, E. D. Jensen and F. D. Reynolds, "An Architectural Overview of the Alpha Real-Time Distributed Kernel", *USENIX Workshop on Microkernels and other Kernel Architectures*, pp 200-208, 1993.

[18] P. Li, B. Ravindran, H. Wu, and E. D. Jensen, A utility accrual scheduling algorithm for real-time activities with mutual exclusion resource constraints, *IEEE Transactions on Computers*. Submitted August 2003. Available at: http://nile.ece.vt.edu/submissions/GUS-TOC03.zip.

[19] D. Mosse, M. E. Pollack, and Y. Ronen, "Value-density algorithm to handle transient overloads in scheduling," in *Proc. Euromicro Conference on Real-Time Systems*, June 1999, pp. 278{286.

[20] W. T. Strayer, "Function-driven scheduling: A general framework for expressing and analysis of scheduling," Ph.D. dissertation, University of Virginia, May 1992, department of Computer Science.

[21] S. A. Aldarmi and A. Burns, "Dynamic value-density for scheduling real-time systems," in *Proc. of Euromicro Conference on Real-Time Systems*, June 1999, pp. 270{277.

[22] E. D. Jensen, C. D. Locke, and H. Tokuda, "A Time-Driven Scheduling Model for Real-Time Systems," *Proc. Real-Time Systems Conference*, IEEE, December 1985.

[23] 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 *Proceedings of IEEE International Workshop on Parallel and Distributed Real-Time Systems*, ser. Lecture Notes in Computer Science, vol. 1586. Springer-Verlag, April 1999, pp. 353{362.

[24] S. Blackman, "Multiple-Target Tracking with Radar Applications," Artrech House, ISBN 0-89006-179-3, 1986.

[25] D. Wells, "A Trusted, Scalable, Real-Time Operating System," *Dual-Use Technologies and Applications Conference Proceedings*, pp II 262-270, Utica, NY, 1994.

[26] D. P. Maynard, S. E. Shipman, and R. K. Clark. et al., "An example real-time command, control, and battle management application for alpha," Technical Report Archons Project TR-88121, CMU Computer Science Department, December 1988.