Time-Utility Function-Driven Switched Ethernet: Packet Scheduling Algorithm, Implementation, and Feasibility Analysis

Jinggang Wang and Binoy Ravindran, *Member*, *IEEE*

Abstract—We present a MAC-layer, soft real-time packet scheduling algorithm called UPA. UPA considers a message model where message packets have end-to-end timeliness requirements that are specified using Jensen's Time-Utility Functions (TUFs). The algorithm seeks to maximize system-wide, aggregate packet utility. Since this scheduling problem is NP-hard, UPA heuristically computes schedules with a quadratic worst-case cost, faster than the previously best CMA algorithm. Our simulation studies show that UPA performs the same as or significantly better than CMA for a broad set of TUFs. Furthermore, we implement UPA and prototype a TUF-driven switched Ethernet system. The performance measurements of UPA from the implementation reveal its strong effectiveness. Finally, we derive timeliness feasibility conditions of TUF-driven switched Ethernet systems that use the UPA algorithm.

Index Terms—Local-area networks, Ethernet, process control systems, real-time and embedded systems.

1 Introduction

Although the IEEE 802.3 Ethernet standard is unsuited for real-time applications due to the randomness in Ethernet's CSMA/CD protocol, the Ethernet is still attractive for real-time applications due to its wide availability, low cost, and high performance such as that offered by the emerging 10 Gigabit Ethernet standard. This has motivated research on the real-time Ethernet.

The most widely studied timing constraint in real-time Ethernet research is the deadline. Examples include shared real-time Ethernet efforts such as TDMA [1], token-passing techniques [2], [3], Virtual Time Protocols [4], [5], [6], Window Protocols [7], traffic smoothing techniques [8], the CSMA/DDCR protocol [9] and [10], switched real-time Ethernet efforts such as EtheReal [11], SIXNET [12], [13], and [14], and real-time packet-switching efforts [15].

A deadline timing constraint for an application activity essentially implies that completing the activity before the deadline accrues some "utility" to the system and that utility remains the same if the activity were to complete *any time* before the deadline. Furthermore, completing the activity after the deadline yields less utility. With deadline timing constraints, one can specify the hard timeliness optimality criterion of satisfying all deadlines and use hard real-time scheduling algorithms [16] to achieve the criterion.

In this paper, we focus on *supervisory* real-time control systems that are emerging and can be found in defense, industrial automation, and telecommunication domains. Supervisory real-time systems control large, physical systems that are composed of several low-level control systems. Moreover, many timing constraints in such systems are "soft" in the sense that completing an activity at any time will result in some utility to the system and that utility *varies* with activity completion time. Furthermore, supervisory systems often desire a soft timeliness optimality criterion, such as completing as many soft time-constrained activities as possible at their *optimal* completion times.

Another distinguishing feature of supervisory systems is that they are subject to significant runtime uncertainties that are inherent in their application environment. Consequently, upper bounds on timing variables in such systems including duration of computational and communication steps are not known to exist at design time with sufficient accuracy. Thus, the hard timeliness optimality criterion of satisfying all timing constraints is difficult to achieve for supervisory real-time systems.

Jensen's time-utility functions [17] allow the semantics of soft timing constraints to be precisely specified. A Time-Utility Function (or TUF) specifies the utility to the system for completing an application activity as an application or situation-specific function of activity completion time [17]. Fig. 1 shows example TUFs.

Moreover, TUF predicates allow specification of soft timeliness optimality criteria. For example, the objective of completing as many activities as possible at their optimal times can be described as *maximizing summed utility* obtained by activity completions.

In this paper, we present a TUF-driven switched Ethernet network system for supervisory real-time control. We consider a timeliness model where application message packets have end-to-end timeliness requirements that are specified using TUFs. Furthermore, we consider a single-segment switched Ethernet network as the underlying

- *J. Wang is with Embedded System Division, Casabyte, Inc.,2801 Prosperity Dr., PO Box 10127, Blacksburg, VA 24062. E-mail: gwang@embedded.casabyte.com.*
- *B. Ravindran is with the Real-Time Systems Laboratory, The Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061. E-mail: binoy@vt.edu.*

Manuscript received 30 Apr. 2003; revised 4 Aug. 2003; accepted 8 Aug. 2003.
For information on obtaining reprints of this article, please send e-mail to:

1045-9219/04/$17.00 © 2004 IEEE Published by the IEEE Computer Society

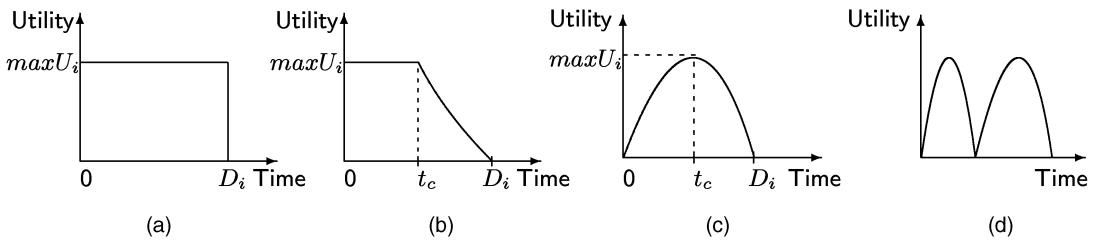


Fig. 1. Soft timing constraints specified using Jensen's time-utility functions: (a) step, (b) soft-step, (c) parabolic, (d) multimodal.

system model. Given such models, our objective is to maximize system-wide, aggregate packet utility.

Toward this objective, we design a packet scheduling algorithm called the **U**tility Accrual **P**acket Scheduling **A**lgorithm (or UPA). The UPA schedules outgoing message packets from source hosts and switches to maximize aggregate packet utility. The packet scheduling problem solved by the UPA is equivalent to the scheduling problem shown to be NP-hard in [18]. In [18], Chen and Muhlethaler present a heuristic algorithm for this problem with a worst-case cost of $O(n^3)$. For convenience, we call this the **C**hen and **M**uhlethaler's **A**lgorithm (or CMA).

Though the UPA heuristically solves the same problem as that of CMA, it only incurs a worst-case cost of $O(n^2)$. Furthermore, our simulation studies show that the UPA performs the same as or significantly better than the CMA for a broad set of TUFs. We also implement the UPA and prototype a TUF-driven switched Ethernet network system using a PC-based platform. Our actual performance measurements from the implementation and experimental comparisons further reveal the strong effectiveness of the algorithm.

Finally, we conduct schedulability analysis and derive timeliness feasibility conditions of switched Ethernet network systems that use the UPA. The feasibility conditions facilitate the design of TUF-driven switched Ethernet systems with guaranteed soft timeliness properties.

Thus, the contribution of the paper includes: 1) the UPA algorithm that seeks to maximize system-wide, aggregate packet utility, 2) construction of a TUF-driven switched Ethernet using the UPA, and 3) timeliness feasibility conditions for constructing switched Ethernets with guaranteed soft timeliness. To the best of our knowledge, we are not aware of any other efforts that solve the problem solved by the UPA (besides the CMA, which the UPA is shown to outperform here).

The rest of the paper is organized as follows: In Section 2, we discuss example supervisory real-time applications to provide the motivating context. We discuss the models of the work in Section 3. In Section 4, we describe the scheduling problem and the objectives. We describe UPA in Section 5. In Sections 6 and 7, we evaluate UPA's performance through simulation studies. We discuss UPA's implementation in Section 8. Section 9 discusses the implementation test bench and Section 10 describes the measurements. We derive UPA's timeliness feasibility conditions in Section 11. The paper concludes in Section 12.

2 Motivating Application Examples

As example supervisory real-time systems, we describe two applications 1) an AWACS (**A**irborne **WA**rning and **C**ontrol **S**ystem) surveillance mode tracker system [19] that was built by The MITRE Corporation and 2) a coastal air defense system [20] that was built by General Dynamics (GD) and Carnegie Mellon University (CMU). Here, we only summarize some of the application timing constraints that are described using TUFs; all other application details can be found in [19], [20], respectively.

The AWACS is an airborne radar system with many missions, including air surveillance. Surveillance missions generate aircraft tracks for command and control. The tracker's most demanding computation, called *association*, associates sensor reports to aircraft tracks. A large number of sensor reports can overload the system, causing sectors of sky to "go blank." The tracker employs two sensors that sweep 180 degrees out of phase with a 10 second period. Thus, association has a "critical time" at the 10 second period. If the computation can process a sensor report for a track in under five seconds (half the sweep period), it will provide better data for the corresponding report from the out-of-phase sensor. Thus, prior to critical time, association's utility decreases as critical time nears.

After the critical time, the utility of association is zero, because newer sensor data has probably arrived. Thus, if the processing load in one sensor sweep period is so heavy that it cannot be completed, probably the load will be about the same in the next period. Thus, there will not be any resources to also process sensor data from the previous sweep.

These semantics establish association's TUF shape: a critical time t_c at sweep period, utility that decreases from a value U_1 to a value U_2 until t_c, and a utility value U_3 after t_c. U_1, U_2, and U_3 are determined using metrics such as: 1) track quality, which is a measure of the amount of sensor data incorporated in a track record; 2) track accuracy, which is a measure of the uncertainty in the estimate of a track's position and velocity; and 3) track importance, which is measure of track attributes such as threat. Fig. 2 shows association's TUF.

Timing constraints of two activities in the GD/CMU coastal air defense system, called *plot correlation* and *database maintenance*, have similar semantics. Correlation is responsible for correlating plot reports that arrive from sensor systems against a tracking database. Maintenance periodically scans the tracking database, purging old and uncorrelated reports so that stale information does not cause tracking errors.

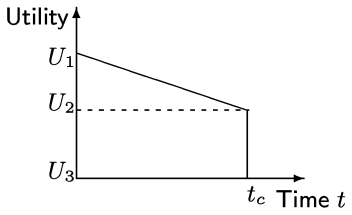


Fig. 2. Track association TUF in MITRE AWACS.

Both activities have "critical times" that correspond to radar frame arrival rate: It is best if both are completed before next data frame's arrival. However, it is acceptable for them to be late by one additional time frame under overloads. Furthermore, plot correlation has a greater utility during overloads. TUFs in Fig. 3 reflect these semantics.

3 The Models

3.1 The Message and Timeliness Models

We consider application message packets that arrive at host MAC-layers for outbound transmission to destination hosts as our message model. The set of packets is denoted $p_i \in P, i \in [1, n]$. Each packet has an end-to-end timing requirement that is specified using a TUF. We denote packet p_i's TUF as $U_i(.)$. Thus, p_i's arrival at its destination host MAC-layer at a time t will yield an utility $U_i(t)$. Fig. 1 shows example TUFs.

Though TUFs can take arbitrary shapes, here we focus on *unimodal* TUFs that are *nonincreasing*. Unimodal TUFs are those that have a *single* optimal completion time interval. Figs. 1a, 1b, and 1c show examples. TUFs that have *multiple* optimal completion time intervals are called *multimodal* TUFs. Fig. 1d shows an example.

Nonincreasing unimodal TUFs are those unimodal TUFs for which utility never increases as time advances. Figs. 1a and 1b show examples. The class of such TUFs allows specifying a broad range of timing constraints; hence, we focus on them.

We define a packet p_i's initial time, denoted as I_i, as the earliest time for which the packet TUF is defined and deadline time, denoted as D_i, as the time at which the TUF drops to zero utility. Further, we assume that $U_i(t) > 0, \forall t \in [I_i, D_i]$ and $U_i(t) = 0, \forall t \notin [I_i, D_i], i \in [1, n]$.

3.2 The System Model

We consider a single-segment switched Ethernet network, where hosts are interconnected through a centralized switch as our target platform (see Fig. 4). Each host is connected to the switch using a full-duplex Ethernet segment (IEEE 802.3) and to a port at the switch that is dedicated for the host. Thus, the link between each host and the switch is a dedicated link for simultaneous two-way communication between the host and the switch. We denote the set of hosts that generate the packet set P as $s_i \in S, i \in [1, z]$.

In single-segment switched Ethernets, packets arrive at the MAC-layer of source hosts where they are generated. Upon arrival, they are queued in the outgoing packet queue of the host. When the network segment from the host to the switch becomes "free" for transmission, the packet scheduling algorithm at the host schedules a packet from the queue for transmission.

The switch maintains a list of packet ready-queues, one queue per host. Each queue stores packets that are destined for a host. When packets arrive at the switch, they are queued in the outgoing packet queue for their destination host. When the network segment from the switch to a host becomes free for transmission, the packet scheduling algorithm at the switch schedules a packet from the queue of destination host for transmission.

The bit length of a packet $p_i \in P$ at the data link layer is denoted as $b(p_i)$. The physical framing overheads increase this size into an actual bit length $b'(p_i) > b(p_i)$ for transmission. Thus, the transmission latency of a packet p_i is given by $l_i = b'(p_i)/\psi$, where ψ denotes the nominal throughput of the underlying network medium (e.g., 10^9 bits/s for Gigabit Ethernet).

We assume that the clocks of hosts and the switch are synchronized using a protocol such as [21]. (We discuss the motivation for clock synchronization in Section 5.)

4 Problem Definition and Objectives

Given the models in Section 3, our objective is to maximize the aggregate utility accrued by the arrival of all packets at their destinations, i.e., $Maximize \sum_{k=1}^{n} U_k(t_k)$, where t_k is the time at which packet p_k arrives at the MAC-layer of its destination host.

In a single-segment switched network, a packet will experience contention for *two* network resources once it arrives at its source MAC-layer. The resources include 1) the network segment from source to switch and 2) the network

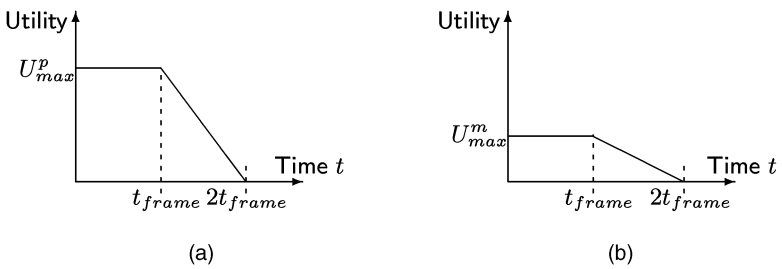


Fig. 3. TUFs of two activities in GD/CMU air defense. (a) Correlation and (b) maintenance.

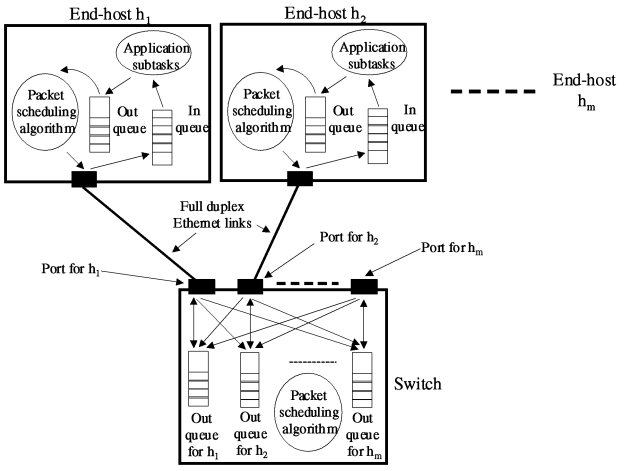


Fig. 4. The switched Ethernet system model.

segment from switch to destination. The contention between packets for the two resources can be resolved in different ways to achieve the objective.

The contention can be *locally* resolved by independently constructing packet schedules for the two network segments such that the aggregate utility accrued on each segment is maximized. Thus, in this approach, MAC-layer packet scheduling algorithms at source hosts and switch will construct local schedules for their respective outgoing network segments such that aggregate utility accrued by the packets on respective segments are maximized. By doing so, the approach seeks to maximize *system-wide*, aggregate accrued utility. The approach thus only seeks to approximate global optimality through independent node scheduling.

The contention can also be *globally* resolved by simultaneously constructing packet schedules for all network segments in a manner that will directly maximize system-wide, aggregate accrued utility. Thus, in this approach, node instances of a logically single scheduling algorithm will execute at MAC-layers of source hosts and switch, interact with each other, and schedule all network segments such that system-wide, aggregate accrued utility is maximized. The approach thus seeks to achieve global optimality.

We consider the local approach in this paper for its simplicity. Our rationale is that the overhead involved in communication and interaction between schedulers for global scheduling may offset its optimality advantage. For example, the time-scales of host/switch MAC-layer scheduling and interhost/switch scheduler communication can differ by orders of magnitude.

We now formalize the objective of (local) packet scheduling at hosts/switch as follows: Let $\mathcal{A} \subseteq P$ denote the set of packets in the outgoing packet queue at a host/switch at a time t. Let $\alpha \subseteq n$ denote the number of packets in set $\mathcal{A}$. Let $\mathcal{S}(\mathcal{A})$ denote all possible sequences of packets of set $\mathcal{A}$, and let $\sigma \in \mathcal{S}(\mathcal{A})$ denote one of the possible packet sequence of the packets in $\mathcal{A}$. Let $\sigma(i)$ denote the packet occupying the ith position in the schedule σ. Then, the scheduling objective is to $Maximize_{\sigma \in \mathcal{S}(\mathcal{A})}\, U(\sigma) = \sum_{k=1}^{\alpha} U_{\sigma(k)}(t + t_k)$, where $t_k = \sum_{i=1}^{k} l_{\sigma(i)}$.

This packet scheduling problem is equivalent to the nonpreemptive task scheduling problem addressed in [18]. In [18], Chen and Muhlethaler show that their task scheduling problem is NP-hard. Further, in [18], they present a heuristic algorithm (referred to as CMA here) to solve this problem, which incurs a worst-case cost of $O(n^3)$. Furthermore, through simulation studies, they show that CMA yields an aggregate utility that is generally close to optimal.

We believe that CMA's $O(n^3)$ cost is too large for an online packet scheduling algorithm. Furthermore, CMA's storage cost is also large, making it practically infeasible for host/switch MAC-layer scheduling. (We discuss this issue in Section 10.1). Thus, in designing UPA, our objective is to compute schedules that 1) are faster than CMA's $O(n^3)$ time, 2) require storage that is appropriate for MAC-layer host/switch scheduling, and 3) produce aggregate utility that is as close as possible to that of the CMA, if not better.

5 The UPA Algorithm: Heuristics and Rationale

5.1 Sort Packets in Decreasing Order of Their "Return of Investments"

The potential utility that can be obtained by spending a unit amount of network transmission time for a packet defines a measure of the "return of investment" for the packet. Thus, by ordering packets in the schedule in the decreasing order of their return of investments, we "greedily" collect as many "high return" packets into the schedule as early as possible. Furthermore, since a packet included in the schedule at any instant in time is always the one with the next "highest-return" packet among the set of nonexamined packets, we increase our chance of collecting as many "high return" packets into the schedule as early as possible. This will increase the likelihood of maximizing the aggregate packet utility as packets yield greater utility if they arrive earlier at their destinations since we only consider nonincreasing unimodal TUFs.

The return of investment for a packet can be determined by computing the slope of the packet TUF. However, computing slopes of arbitrary unimodal TUFs can be computationally expensive. Thus, we determine the return of investment for a packet as simply the ratio of the maximum possible packet utility (specified by the packet TUF) to the packet deadline. This is just a single division, costing $O(1)$ time. We call this ratio the "pseudoslope" of a packet. The slope is "pseudo" as it only gives an approximate measure of the slope. However, the pseudo-slope is an attractive metric since it can be computed with low overhead.

5.2 Move Infeasible Packets to Schedule-End

Infeasible packets are packets that cannot arrive at their destinations before their deadlines, no matter what. This is because the *remaining* transmission time of such packets is longer than the time interval between their arrival at a host or the switch and the packet deadlines. Packets that are not infeasible are feasible packets.

By moving infeasible packets to schedule-end, we move as many feasible packets to schedule-beginning as possible. This will increase the likelihood of maximizing aggregate

UPA($\mathcal{A}$, α, t) /* $\mathcal{A}$: set of packets in out queue; α: # of packets in $\mathcal{A}$; t: time of sched. event */

1. $\sigma = \emptyset$; /* Intialize packet schedule to empty */
2. **For** each packet $p_i \in \mathcal{A}$
   - 2.1 PseudoSlope(p_i) $= U_i(0)/D_i$; /* Max utility occurs at time 0 */
3. Sort packets in $\mathcal{A}$ in decreasing order of their pseudo-slopes; /* $\mathcal{A}$ is now sorted */
4. $\sigma = \mathcal{A}$; /* packet schedule σ is set equal to sorted set $\mathcal{A}$ */
5. **For** $k = 1$ to α
   - 5.1 InOrder = TRUE;
   - 5.2 **For** $i = 1$ to $\alpha - 1$
     - 5.2.1 $j = i + 1$; /* p_j is the packet that follows p_i in schedule σ */
     - 5.2.2 **If** $(t + l_i > D_i)$ /* Check for feasibility of packet p_i */
       - Move p_i to end of schedule σ; /* packet p_i is not feasible */
       - **Continue**; /* Skip and continue to another iteration */
     - 5.2.3 **If** $(t + l_j > D_j)$ /* Check for feasibility of packet p_j */
       - Move p_j to end of schedule σ; /* packet p_j is not feasible */
       - **Continue**; /* Skip and continue to another iteration */
     - 5.2.4 $\Delta_{i,j}(t) = [U_i(t + l_i) + U_j(t + l_i + l_j)] - [U_j(t + l_j) + U_i(t + l_j + l_i)]$;
     - 5.2.5 **If** $(\Delta_{i,j}(t) < 0)$ /* Out of order, so swap */
       - $\sigma(i) = p_j$; $\sigma(j) = p_i$; $t = t + l_j$; InOrder = FALSE;
     - 5.2.6 **Else** $t = t + l_i$;
   - 5.3 **If** (InOrder = TRUE) **Break**; /* No swaps; so all packets are inorder */
6. σ is the final schedule; return packet $\sigma(1)$ as the packet selected for transmission;

Fig. 5. High-level pseudocode of the UPA algorithm.

packet utility as feasible packets yield greater utility if they arrive earlier at their destinations since we only consider nonincreasing unimodal TUFs. Furthermore, infeasible packets yield zero utility if they arrive at their destinations after their deadlines. Thus, there is no reason for transmitting them early and jeopardizing the potential utility that can be accrued from feasible packets.

To determine whether a packet is infeasible, the algorithm therefore needs global time. Thus, as discussed previously, we assume that the host and switch clocks are synchronized.

5.3 Maximize Local Aggregate Utility As Much As Possible

We derive the concept of local aggregate utility from the *precedence-relation* property in [18]: Consider two schedules, $\sigma_a = \langle \sigma_1, p_i, p_j, \sigma_2 \rangle$ and $\sigma_b = \langle \sigma_1, p_j, p_i, \sigma_2 \rangle$ of a packet set $\mathcal{A}$, such that $\sigma_1 \neq 0$, $\sigma_2 \neq 0$, $\sigma_1 \bigcup \sigma_2 = \mathcal{A} - \{p_i, p_j\}$, and $\sigma_1 \bigcap \sigma_2 = \emptyset$. Consider a time instant $t = \sum_{k \in \sigma_1} l_k$, when a scheduling decision has to be made, i.e., t is the time instant after all packets in schedule σ_1 has been transmitted. Now, the scheduling decision at time t can be made by computing

$$\Delta_{i,j}(t) = \big[U_i(t + l_i) + U_j\big(t + l_i + l_j\big)\big] - \big[U_j\big(t + l_j\big) + U_i\big(t + l_j + l_i\big)\big].$$

Thus, if $\Delta_{i,j}(t) \geq 0$, then σ_a will yield a higher aggregate utility than σ_b; otherwise, σ_b is better than σ_a.

Now, by examining adjacent packets p_i and p_j in a schedule $\langle \sigma_1, p_i, p_j, \sigma_2 \rangle$ and ensuring that $\Delta_{i,j}(t) \geq 0$, we can maximize the *local* aggregate utility of packets p_i and p_j. If all adjacent packets in the schedule have such locally maximized aggregate utility, this will increase the likelihood of maximizing the global aggregate utility.

The maximization of the local aggregate utility can be done in a manner similar to that of Bubble sort. We can examine adjacent pairs of packets in the schedule, compute Δ, and swap the packets, if the reverse order can lead to higher local aggregate utility. Furthermore, the procedure can be repeated until no swaps are required.

Pseudocode of the UPA at a high-level of abstraction is shown in Fig. 5.

5.4 Computational Complexity of the UPA

The UPA's cost depends upon Step 5's cost. Step 5's cost is dominated by that of Step 5.2; all other substeps of Step 5 take $O(1)$ time. Step 5.2 iterates a maximum of α times and, thus, costs $O(\alpha)$. Step 5 iterates a maximum of α times and, thus, costs $O(\alpha^2)$. Given n packets, UPA's cost is thus $O(n^2)$, which is faster than CMA's $O(n^3)$ cost [18].

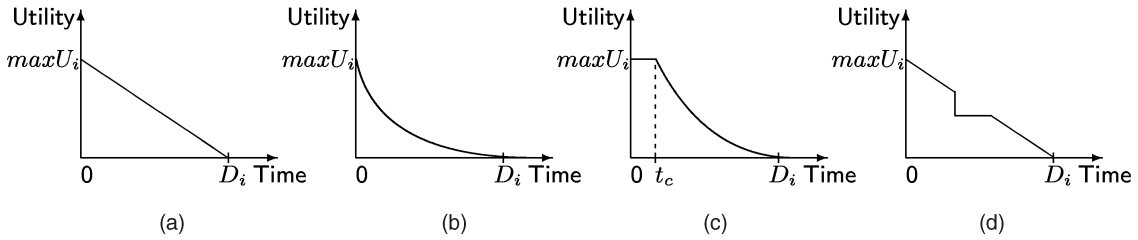


Fig. 6. Four TUFs considered in experimental study: (a) linear, (b) exponential, (c) quadratic, (d) composite.

6 Simulation Study 1: Single Queue Environment

To study how the UPA compares with the optimal algorithm, we consider a single packet queue environment, where packets queue-up in a single queue for outbound transmission. We consider such a single queue environment because that allows us to significantly reduce the problem size, thereby facilitating the UPA's comparison with the optimal algorithm. The optimal algorithm determines optimal schedules through exhaustive search.

6.1 Experiment Setup

To study the UPA's performance in a large data space, we randomly generate all message parameters using probability distribution functions. We determine message transmission time and deadline from an exponential distribution and maximum utility value from a normal distribution.

6.2 Time-Utility Functions

We consider six TUFs in our study. These include the step and soft-step TUFs shown in Fig. 1 and linear, exponential, quadratic, and composite TUFs shown in Fig. 6. Our motivation to consider these six TUFs is that they are close variants of the MITRE AWACS tracker and GD/CMU air defense system TUFs. In fact, in the design of the AWACS TUF, the designers empirically derived the slope of the TUF [19]. Therefore, we consider TUFs that are similar to the AWACS TUF, but with different slopes.

Moreover, the AWACS TUF had to linearly decrease due to an implementation artifact—the scheduling algorithm of the OS (OSF/RI's MK7.3A) allowed only a single critical time. Thus, we consider TUFs that are similar to the AWACS TUF, but that allow multiple critical times such as soft-step and quadratic TUFs (Figs. 1b and 6c, respectively).

We believe that the soft-step and quadratic TUFs nicely "fit" the semantics of AWACS's track association computation. This is because both the TUFs allow two "critical" times: 1) constant utility up to a critical time t_c, after which the utility decreases (with different slopes); and 2) a deadline time D_i, after which the utility is zero.

The time t_c could very well map to half the sweep period of the two sensors, i.e., five seconds, and the time D_i could map to the sweep period length of 10 seconds. This will allow the association computation to gain a constant maximum utility for processing a sensor report for a track *any time* before half the sweep period (t_c), thereby providing better data for the corresponding report from the out-of-phase sensor. Further, after half the sweep period, the utility of the computation will decrease as the sweep period length (D_i) nears. Furthermore, the computation will gain zero utility after the sweep period.

6.3 Normalized Average Performance

Table 1 shows the average (and standard deviation) of the normalized aggregate utility produced by the UPA and CMA for the six TUFs for two traffic sets. We compute the normalized aggregate utility of an algorithm as the ratio of the aggregate utility of the algorithm to that of the optimal algorithm. Fig. 7a shows the average values for one of the traffic sets (the 9-message set).

From Table 1 and Fig. 7a, we observe that the UPA's performance is very close ($\geq$ 93 percent) to that of the

TABLE 1
The UPA and CMA's Optimal-Normalized Utility

	9-Message Traffic Set				*10-Message Traffic Set*			
	CMA		*UPA*		*CMA*		*UPA*	
	Avg	*Dev*	*Avg*	*Dev*	*Avg*	*Dev*	*Avg*	*Dev*
Linear	0.9944	0.0192	0.9873	0.0394	0.9865	0.0265	0.9802	0.0402
Step	0.8028	0.1774	0.97	0.0549	0.6833	0.1886	0.9457	0.0712
SoftSt.	0.7264	0.2105	0.9361	0.1214	0.6724	0.2312	0.8988	0.1476
Exp	1	8E-6	0.9781	0.1079	1	0	0.9589	0.1449
Quad	0.9976	0.0202	0.9738	0.1146	0.9992	0.0038	0.9534	0.1473
Comp	0.9411	0.0995	0.9462	0.0938	0.8982	0.1253	0.917	0.1108

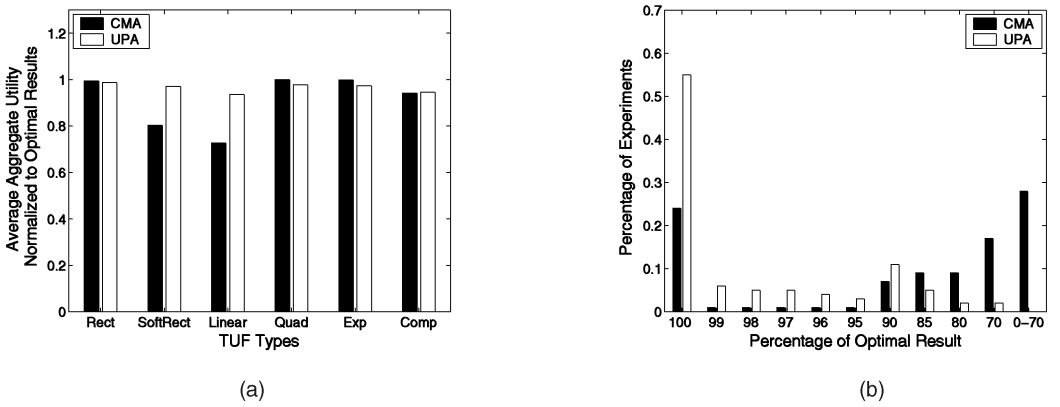


Fig. 7. UPA and CMA's performance with respect to optimal algorithm. (a) Optimal-normalized utility. (b) Utility distribution.

optimal algorithm for all six TUFs. Further, we observe that the CMA performs close to that of the optimal algorithm for some TUFs, such as quadratic, linear, and exponential, but performs poorly for TUFs such as step and soft-step. Furthermore, the UPA performs close to the CMA for linear, exponential, quadratic, and composite TUFs. However, the UPA outperforms the CMA for step and soft-step TUFs for both the traffic sets.

To determine how the UPA's and CMA's performances are distributed over the range of experiments conducted for a given TUF, we plot different percentages of the normalized aggregate utility of the algorithms in terms of the percentage of experiments. Fig. 7b shows the distribution of the algorithms' optimal-normalized aggregate utility for step TUF. From the figure, we observe that more than 55 percent of the UPA's aggregate utility are exactly the same as the optimal value. Furthermore, no results are found to be less than 70 percent of optimal value.

7 Simulation Study 2: Switched Networked Environment

7.1 Experiment Setup

To study the UPA's performance in a large data space, we randomly generate all message parameters using probability distribution functions. We consider a switched network of five hosts, where each host has five processes that generate trans-node messages. We determine the message destination address from a uniform distribution, message length and deadline from an exponential distribution, and maximum utility value and interarrival time from a normal distribution.

Besides the UPA and CMA, we consider the EDF [16] and First-In-First-Out (FIFO) algorithms. We exclude the optimal algorithm for this study as it is found to be computationally intractable for the networked environment. We also consider the six TUFs discussed in Section 6.2.

7.2 Normalized Average Performance

Fig. 8 shows the average of the normalized aggregate utility produced by the UPA, CMA, and EDF for all experiments that were conducted for the algorithms, for each of the six TUFs. For this study, we determine the normalized aggregate utility of an algorithm as the ratio of the aggregate utility of the algorithm to that of FIFO. In Table 2, we show the maximum, minimum, and the standard deviation of the normalized aggregate utilities, besides the average.

From Fig. 8 and Table 2, we observe that the UPA performs the best and the EDF performs the worst, for all six TUFs. The CMA performs in between that of the UPA and EDF. Further, the UPA maintains an average normalized aggregate utility of 2.5. Furthermore, the UPA's and CMA's performance is just about the same for linear, exponential, and composite TUFs. However, the UPA significantly outperforms the CMA for step, soft-step, and quadratic TUFs.

From Table 2, we also observe that EDF performs worse than FIFO in many cases (the minimum value of the EDF's normalized aggregate utility is less than one), although its average performance is always better. This is because the EDF works well for step TUFs that have same maximum utilities (EDF's optimality [22] is true for such a case), which is not the case here. When the maximum utility of step TUFs differs, the EDF performs worse.

7.3 Performance under Increasing Arrival Density

We were also interested to determine how the UPA and CMA perform when the arrival density of packets increases. The arrival density of a packet is the number of times the packet arrives during a time interval. Thus, a larger arrival density implies larger traffic. Our interest in this metric is due to the dynamic nature of supervisory systems that we focus, which are frequently subject to runtime increases in message traffic.

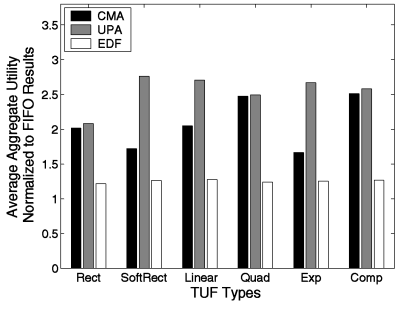


Fig. 8. FIFO-normalized utility of the UPA, CMA, and EDF.

TABLE 2
Performance of the UPA, CMA, and EDF with Respect to FIFO

		Step	*Soft-Step*	*Linear*	*Exp*	*Quad*	*Comp*
CMA	*Max*	6.5460	7.2393	5.4552	10.0590	5.5322	10.4821
	Min	0.9388	0.9882	1.0612	0.9998	0.9426	0.9990
	Avg	1.7228	2.0483	2.0169	2.4754	1.6665	2.5129
	Dev	0.9009	1.1423	0.8728	1.7103	0.8267	1.7569
UPA	*Max*	13.1441	10.4888	5.5426	11.2589	8.6893	9.8991
	Min	1.0029	1.0022	1.0631	1.0004	1.0027	0.9993
	Avg	2.7621	2.7081	2.0806	2.4924	2.6704	2.5827
	Dev	2.0514	1.8661	0.9331	1.7284	1.8234	1.8104
EDF	*Max*	5.2358	3.8456	3.8664	3.4387	4.4670	4.7252
	Min	0.5038	0.3060	0.5204	0.4395	0.3236	0.5102
	Avg	1.2632	1.2758	1.2148	1.2411	1.2525	1.2696
	Dev	0.5327	0.5491	0.4104	0.4837	0.5582	0.5715

We repeated the experiments described in Section 7.2 for 16 message arrival densities that are progressively increasing. The results are shown in Fig. 9 for quadratic TUFs.

In Fig. 9a, we normalize the aggregate utility of the algorithms with respect to FIFO. From the figure, we observe that the UPA performs the best, the EDF the worst, and the CMA in between. Fig. 9b shows the deadline-miss ratio of the UPA, CMA, EDF, and FIFO. Again, we observe that the UPA has the smallest miss ratio, followed by the CMA and EDF. We observed similar consistent results for all other TUFs. So, these are not shown.

8 The UPA Implementation

Our goal in implementing the UPA is simply to prototype a TUF-driven switched Ethernet network in a laboratory setting, conduct experiments using the implementation (by comparing the UPA with other algorithms), and, thus, measure the UPA's actual performance.

Since our goal is only to prototype a network for experimentation and instrumentation (as opposed to actual deployment/application usage), we prototype the switch using a PC. We use a Pentium PC with a 450 MHz processor and 256MB of memory as the central switch. In the switch, we use two ZX346Q 4-port Ethernet adapters from ZNYX [23] for packet switching. The ZX346Q uses PCI bus to communicate with the PC.

Since the computational demand on hosts is small—capability to send/receive packets—we use a single PC to emulate several hosts that send messages. We use another 4-port Ethernet adapter in a Pentium PC to emulate four hosts. For the receiving host, we use a Pentium PC. Thus, in the prototype setup, there are a total of four sending "hosts" and one receiving host.

The four sending ports connect to the switch with 100Mbps full duplex links and the switch connects to the receiving host with a 10Mbps half-duplex link. In our experiments, we assume that real-time messages unidirectionally flow from the four sending "hosts" to the receiving host. Since, in real-time packet transmission, 2-way hand shaking mode is not used, the unidirectional real-time message flow and 10Mbps half-duplex link do not affect our performance studies. We use a 10Mbps link as the

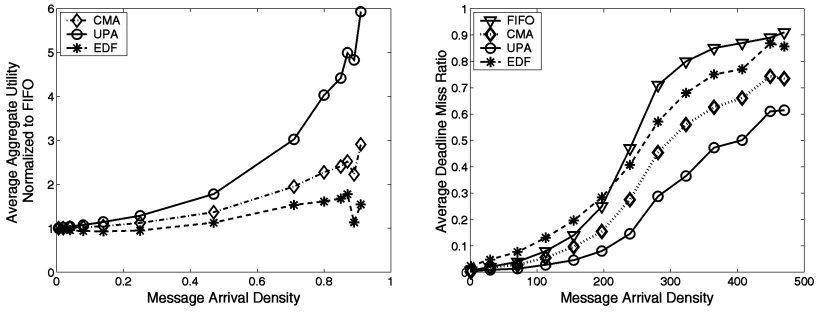


Fig. 9. Performance under increasing arrival density and quadratic TUFs. (a) Normalized aggregate utility. (b) Deadline miss ratio.

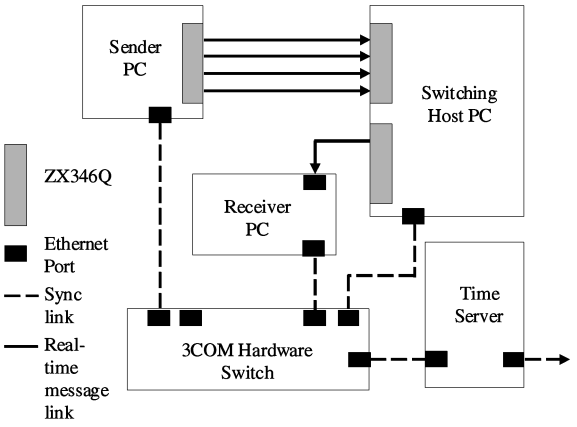


Fig. 10. Prototype TUF-driven switched Ethernet.

receiving link so that sufficiently large packet traffic can be generated on the switch output port.

To synchronize the PC clocks, we use the `xntpd` [24] implementation of the NTP protocol [21]. To accurately synchronize time, we use a dedicated 100Mbps link in all three PCs to connect to the time server in our laboratory through another 3COM hardware-level Ethernet switch. The lab time server is synchronized with an external time server.

We implement UPA at the MAC layer—between the IP and the Ethernet device driver layers—in the Linux kernel for packet scheduling (at hosts and switch). We believe that this is the best location to implement the algorithm because of the accuracy with which scheduling results can be obtained. In the switch, we use Böhme and Buytenhak's Linux Bridge program [25] to conduct packet switching. Fig. 10 shows the prototype system.

9 Test Bench Setup

9.1 Experimental Parameters

For measuring the UPA's performance from the implementation, we generate traffic using the same distributions as those used in our simulation studies. In these experiments, we exclude exponential and composite TUFs. This is because it is computationally expensive to evaluate an exponential TUF in the kernel as it requires math-emulation. Furthermore, composite TUFs are least similar to our motivating TUFs—AWACS and air defense system TUFs.

9.2 Emulating Four Hosts with a Single PC

In our prototype, four sending "hosts" are emulated by four Ethernet ports. In order to send messages through a specific port, we configure each Ethernet port with a different IP address. More importantly, all four IP addresses belong to four different network addresses so that the routing table can be built by the Linux kernel exactly as we desire. However, on the receiver side, it has only one IP address. Therefore, we set three IP alias addresses for the receiver so that the receiver can be accepted by four different network address domains. The IP address configuration details can be found in [26].

9.3 Algorithms for Comparative Evaluation

We implemented six algorithms for a comparative study. The algorithms include the UPA, CMA, FIFO, **EDF** with **N**o **D**eadline **M**iss **C**heck (or EDF_NDMC), and **EDF** with **D**eadline **M**iss **C**heck (or EDF_DMC). Our rationale for considering EDF_DMC and EDF_NDMC is because of the UPA's feasibility test (Steps 5.2.2 and 5.2.3, Fig. 5). Thus, a comparison of the UPA and EDF_DMC will help us evaluate the impact of the UPA's scheduling scheme toward the UPA's performance since both the UPA and EDF_DMC employ the feasibility test. On the other hand, a comparison of EDF_DMC and EDF_NDMC will help us evaluate the impact of the feasibility test itself since EDF_DMC employs the test and EDF_NDMC does not.

9.4 Packet-Level TUFs from Message-Level TUFs

The UPA schedules packets using packet TUFs. However, if a message size is larger than what can be accommodated in a packet, the message will be fragmented into several packets. In such instances, it is reasonable for all packets of the message to inherit the message TUF. To evaluate the effectiveness of this strategy and, thus, the relationship between packet-level and message-level scheduling, we study two cases: 1) "big" message sizes and 2) "small" message sizes. With big message sizes, message fragmentation into packets occurs, whereas, with small message sizes, fragmentation rarely occurs.

9.5 Impact of Maximum Utility Value

To evaluate the impact of *maximum* utility values on the performance of the algorithms, we consider a deterministic assignment of maximum utilities, besides a probabilistic assignment using distribution functions. For example, if an algorithm shows an advantage over others even when the ratio of maximum utility to minimum utility is small, it indicates that the algorithm is quite efficient because it can distinguish a small difference.

Thus, we consider two utility assignments called, UA_L and UA_H. In UA_L, we repeatedly assign the maximum utility values 1, 5, 15, 17, 20, 23, 25, 280, and 1,000 to the messages. In UA_H, we repeatedly assign the maximum utility values 1, 5, 15, 17, 20, 23, 25, 28, and 8,000. Thus, the ratio of maximum utility value to minimum utility value of UA_L is 1,000, while that of UA_H is 8,000. Furthermore, the average of the difference between successive maximum utility values of UA_L is smaller than that of UA_H.

10 Performance Measurement and Analysis

10.1 Infeasibility of the CMA

From our implementation measurements, we observed that the CMA is not practical as a packet scheduling algorithm due to its large storage cost. The CMA uses a precedence matrix to store real-time attributes of each packet [18]. Thus, if the queue size is 4K and each packet requires 14 bytes to store real-time attributes, the total demand for memory would be $4\text{K} \times 4\text{K} \times 14 = 224\text{Mbytes}$. This will exhaust the entire host/switch address space.

For the UPA, we only need a one-dimensional array to store packet real-time attributes, requiring only 56K of

TABLE 3
T0: Big Message Size and Normal Distribution Utility

	Utility Normalized to FIFO				*Average Deadline Miss Ratio*			
	FIFO	*EDF_DMC*	*UPA*	*EDF_NDMC*	*FIFO*	*EDF_DMC*	*UPA*	*EDF_NDMC*
Step	1	1.34	1.61	0.86	0.56	0.50	0.5	0.62
Soft-Step	1	6.35	7.50	1.30	0.58	0.52	0.48	0.54
Linear	1	2.73	5.32	1.22	0.60	0.50	0.44	0.52
Quad	1	4.80	6.35	1.57	0.58	0.51	0.5	0.48

TABLE 4
T1: Small Message Size and Normal Distribution Utility

	Utility Normalized to FIFO				*Average Deadline Miss Ratio*			
	FIFO	*EDF_DMC*	*UPA*	*EDF_NDMC*	*FIFO*	*EDF_DMC*	*UPA*	*EDF_NDMC*
Step	1	1.63	1.84	0.83	0.44	0.28	0.21	0.55
Soft-Step	1	8.45	5.96	1.66	0.47	0.36	0.19	0.42
Linear	1	29.18	38.39	5.89	0.40	0.34	0.17	0.47
Quad	1	4.13	5.53	0.88	0.54	0.51	0.47	0.56

TABLE 5
T2: Small Message Size and UA_L Maximum Utility

	Utility Normalized to FIFO				*Average Deadline Miss Ratio*			
	FIFO	*EDF_DMC*	*UPA*	*EDF_NDMC*	*FIFO*	*EDF_DMC*	*UPA*	*EDF_NDMC*
Step	1	1.59	3.00	1.12	0.50	0.35	0.31	0.45
Soft-Step	1	12.64	20.43	1.46	0.51	0.31	0.26	0.50
Linear	1	3.35	3.52	0.93	0.55	0.29	0.14	0.57
Quad	1	7.31	6.24	1.11	0.56	0.28	0.14	0.57

TABLE 6
T3: Small Message Size and UA_H Maximum Utility

	Utility Normalized to FIFO				*Average Deadline Miss Ratio*			
	FIFO	*EDF_DMC*	*UPA*	*EDF_NDMC*	*FIFO*	*EDF_DMC*	*UPA*	*EDF_NDMC*
Step	1	1.96	3.03	1.03	0.51	0.31	0.26	0.50
Soft-Step	1	6.57	5.90	0.91	0.52	0.31	0.21	0.55
Linear	1	3.40	4.22	0.95	0.53	0.30	0.19	0.57
Quad	1	7.82	6.87	1.26	0.55	0.29	0.14	0.56

memory for auxiliary storage. For the EDF algorithms, there is no need for similar storage. Thus, only the UPA, the EDF algorithms, and FIFO are practical.

10.2 Performance Comparison by Message Size and Maximum Utility Value

In order to evaluate the algorithm performance, we conducted experiments for all four TUFs under all five traffic conditions. Tables 3, 4, 5, and 6 illustrate the results.

From the tables, we observe that, in each traffic type, the UPA always achieves the largest accrued utility, except in some cases of soft-step TUFs where EDF_DMC performs slightly better than the UPA. We believe that such exceptional cases are due to the difference between the pseudoslope approximated by the UPA and the actual slope of soft-step TUFs considered in the experiments.

With different TUFs, we observe that the advantage of the UPA is not that obvious for step TUFs. However, it is

quite obvious for the other three TUFs where the UPA achieves accrued utilities that are 6 to 38 times as that achieved by FIFO.

From Tables 3, 4, 5, and 6, we also observe that the UPA has the minimum deadline-miss ratio among all four algorithms. The difference between the UPA and other algorithms for the deadline-miss ratio metric is quite significant.

Thus, from the results, we observe that the UPA performs very well, even in a software-switching platform. We believe that this is due to the UPA's average-case cost, which is far less than the worst-case cost of $O(n^2)$. Recall that the UPA's most dominating step is the inner for-loop, which can iterate a maximum of n times (see Fig. 5). The outer for-loop also iterates for n times. However, it turns out that the pseudoslope order that the UPA determines prior to the nested for-loop is very close to the final order. Thus, the exact number of iterations performed during the inner for-loop is far less than n. This significantly reduces the UPA's average-case cost. Such results were frequently observed during our simulation studies.

From Tables 3 and 4, we also observe that the UPA achieves similar advantages over FIFO between message-level scheduling and packet-level scheduling except for linear TUFs. This means that, although the UPA is a packet scheduler, the final result at the message-level is also good. Moreover, since the UPA's pseudoslope for linear TUFs is actually the real TUF slope, it is not surprising that, for linear TUFs and small message sizes, the UPA achieves the best accrued utility, e.g., 38 times as that of FIFO.

Another interesting observation is that EDF_NDMC presents almost no advantage over that of FIFO. This is not surprising since, during overload situations, EDF_NDMC does no deadline-miss checks. This wastes network bandwidth similar to that of FIFO. But, the EDF-ordering wastes more network bandwidth than that of FIFO due to the EDF's domino effect [22]. This explains why EDF_NDMC performs worse than FIFO.

This hypothesis is further validated by EDF_DMC's performance, which, to some extent, is close to that of the UPA. Although EDF_DMC only does deadline-ordering, it has a lesser cost than that of the UPA. This advantage of EDF_DMC becomes significant when the UPA's pseudoslope is not close to the real slope or when utility-ordering is close to deadline-ordering.

Tables 5 and 6 show results for small message size and the deterministic UA_L and UA_H maximum utility assignments for step, soft-step, and quadratic TUFs, respectively. The final utility results are better than small message size and normal distribution utility cases, while, for linear TUFs, the latter is much better than former.

We observed that the results for big message size and UA_H utility assignment-case is similar to that of small message size and UA_H maximum utility-case shown in Table 6. So, those results are not shown. This demonstrates the effectiveness of packets inheriting message TUFs.

10.3 Performance Comparison by Time-Utility Function Types

The average aggregate utility of the four algorithms with respect to FIFO under different TUFs are shown in Fig. 11. From the figure, we observe that the TUF order where the UPA performs the best to the worst when compared with FIFO is: linear > soft-step > quadratic > step. From this order, we conclude that closer the pseudoslope of the UPA to the real TUF slope, greater the advantages the UPA has over that of FIFO.

Another conclusion we draw is about the most favored traffic types of the UPA under each TUF. For example, we see from Fig. 11a that, when step TUF is used, the UPA is good at traffic types T2, T3, and T4. Furthermore, we observe that the UPA is good at traffic type T1 for linear TUF, type T2 for soft-step TUF, and type T4 for quadratic TUF.

We observed similar consistent results for the algorithms for the deadline-miss ratio metric for different TUFs. So, these are not shown. We observed that the UPA always generates the smallest deadline-miss ratio. Furthermore, among all TUFs for the UPA, the linear TUF gives the smallest miss ratio. This is expected since the pseudoslope of linear TUF is its actual slope.

10.4 Performance Comparison Based on Increasing Traffic Arrival Density

All the performance comparisons described so far are based on averaged results from tens of experiments. In order to compare all four algorithms at a much finer level, we repeated all individual experiments according to the ascending order of traffic density.

In Fig. 12, we show the results for step TUF under traffic type T3. We observed very similar results for step TUF under all other traffic types. So, these are not shown. From Fig. 12 and other similar measurements for traffic types T0, T1, T2, and T4, we draw the following conclusions: First, we conclude that the higher the traffic density, the more advantages the UPA has over that of FIFO. This conclusion is exactly the same as that obtained from our simulation results. This also means that UPA is effective for overloads.

Second, we conclude that, when FIFO's deadline-miss ratio is small, the UPA's and EDF's deadline-miss ratios are larger (than that of FIFO) due to their higher computational cost. However, when FIFO's deadline-miss ratio is large, the computational overhead of the UPA and EDF is completely traded off; thus, both the UPA and EDF achieve lower deadline-miss ratio than FIFO, although the advantage of EDF_NDMC over FIFO is quite small.

Third, we conclude that the UPA achieves the largest aggregate utility except in very few special cases. Furthermore, the UPA achieves the smallest deadline-miss ratio in all cases.

Finally, we conclude that the accrued utility advantage of the UPA over EDF_DMC is significant in some cases such as under traffic types T2, T3 (shown in Fig. 12), and T4. However, in some other cases, this advantage is not too significant and, in yet other few cases, the advantage even disappears. (We discuss the reasons for this in Section 10.2.)

We observed similar results for soft-step, linear, and quadratic TUFs under all five traffic types. So, these are not shown here. Thus, the actual performance measurements clearly demonstrate the UPA's superiority over the other algorithms.

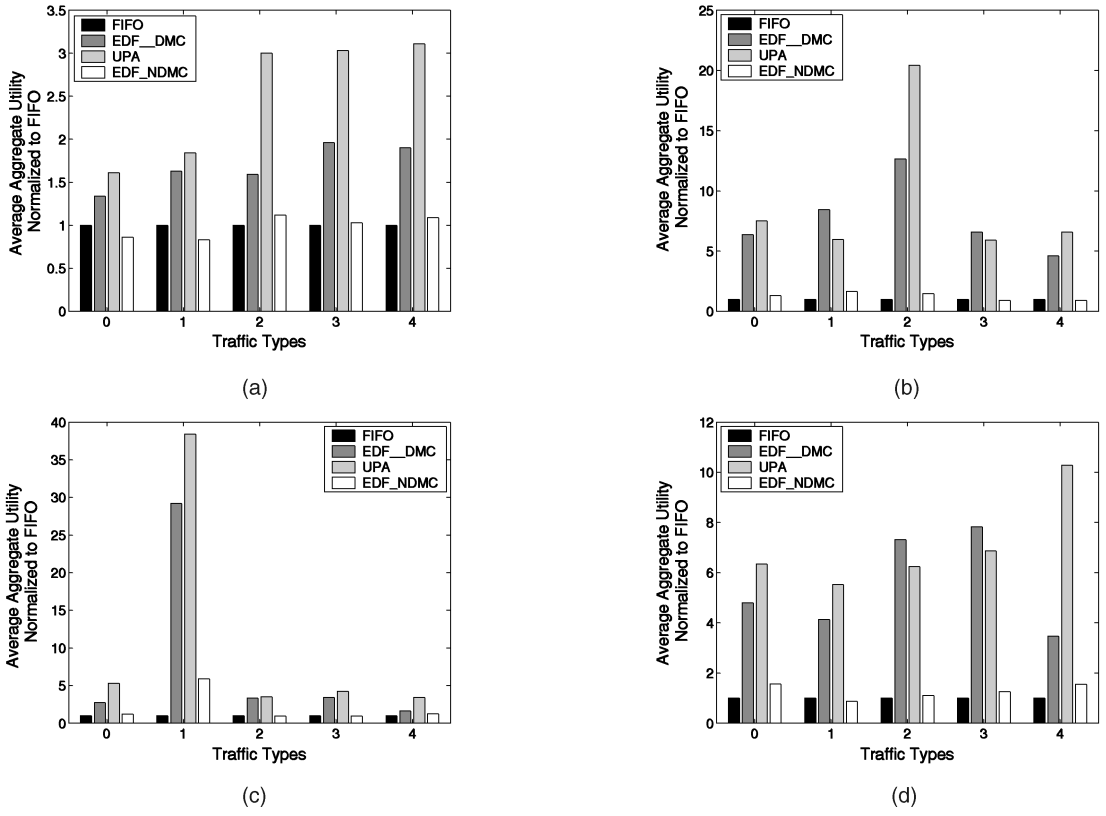


Fig. 11. Average aggregate utility with respect to FIFO under five traffic types for different TUFs. (a) Step, (b) soft-step, (c) linear, (d) quadratic.

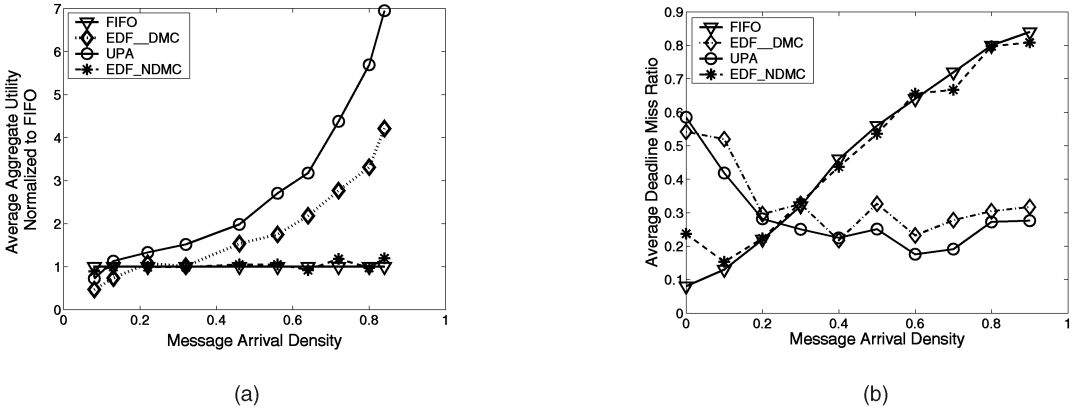


Fig. 12. Performance under traffic Type T3 for Step TUF. (a) Normalized aggregate utility and (b) deadline miss ratio.

11 Timeliness Feasibility Conditions

We use a Systems Engineering (SE) approach in deriving the UPA's timeliness feasibility conditions. An SE problem specifies the models and properties of a system that is desired to be constructed [27]. Models are assumptions on the operational conditions of the desired system. Properties are desired services that the system must provide to end-users. For a real-time system, the fundamental property is timeliness.

A solution to an SE problem specifies the "blueprint" of the desired system, i.e., the architectural and algorithmic solutions that constitute the system. Furthermore, such a system solution must deliver the desired properties under the assumed models. For a real-time system, this is ensured by timeliness feasibility conditions.

We first describe the models and properties. We then derive the feasibility conditions. In describing the models, we only augment our description in Section 3.

11.1 Models

The size n of the packet set P and the number of host sources z are unrestricted. Further, we assume that the packet subset $P_j \subseteq P$ is mapped onto source $s_j \in S, j \in [1, z]$. Furthermore, any packet $p_j \in P$ may be mapped onto any source $s_j \in S$.

Packets arrive according to the *multimodal model* [27], which is a generalization of the *unimodal model*. For a

packet p_i, the unimodal model defines the size of a sliding time window $w(p_i)$ and the maximum number of arrivals $a(p_i)$ that can occur during any $w(p_i)$. Multimodal arrival is a finite, ordered sequence of unimodal arrivals. Thus, the sequence $\langle (a_\pi(p_i), w_\pi(p_i)), \pi \in [1,k] \rangle, i \in [1,n]$ is defined, where k is unrestricted.

Our motivation to consider the multimodal model is due to the strength of the "adversary" embodied in the model, i.e., the set of worst-case scenarios allowed by the model. The "adversary" embodied in the multimodal model is stronger than those in the unimodal, aperiodic, sporadic, and periodic models [27]. The weaker the arrival model (due to weakness in assumption), the greater is the likelihood that the properties hold true.

The clock synchronization (clock-sync, for short) module at hosts and switch periodically generate clock-sync packets at a period θ. Clock-sync packets are always transmitted before transmitting application packets. The bit length of a clock-sync packet at the data link layer is a constant and is denoted b_c. Physical framing overheads increase this to $b_c' > b_c$.

11.2 Timeliness Property

The timeliness property is a desired lower bound on system-wide, accrued utility, which is the sum of the utility accrued by the arrival of packets at their destinations. Thus, $ATB = \sum_{i=1}^{n} U_i(t_i) \geq ATB_l$, where t_i is the absolute time at which packet p_i arrives at its destination. The functions $U_i, i \in [1,n]$ and the lower bound ATB_l are unknown.

11.3 Construction of Timeliness

To establish the desired timeliness property, we need to determine the worst-case lower bound on system-wide accrued utility under the design models. This can be determined by computing a lower bound on the individual utility accrued by each packet. To determine such a lower bound, we need to determine a packet delay-upper bound. We thus seek to construct a computable function $R(s_i, p)$ that gives an upper bound on the delay incurred by any packet p to arrive at its destination since its arrival at its source host MAC-layer.

Since a packet will experience contention for *two* network resources (in a single-segment switched network) once it arrives at the MAC-layer of its source, we define $R(s_i, p) = R_1(s_i, p) + R_2(p)$. $R_1(s_i, p)$ is the upper bound on the delay incurred by any packet p to arrive at the switch since its arrival at the MAC-layer of any source $s_i, i \in [1,z]$ and $R_2(p)$ is the upper bound on the delay incurred by any packet p to arrive at its destination since its arrival at the switch.

11.3.1 Construction of $R_1(s_i, p)$

Consider a packet p that arrives at the MAC-layer of a source s_i. Let $A(p)$ denote p's arrival time, $d(p)$ denote p's relative deadline, and $I(p)$ denote the interval $[A(p), A(p) + d(p)]$. To determine $R_1(s_i, p)$, we need to determine an upper bound on the number of packets belonging to set P_i that will be scheduled for outbound transmission on s_i, over any interval $I(p)$, *before* p is transmitted. We denote this upper bound as $u_1^i(p)$. $u_1^i(p)$ must be established assuming that packet arrivals occur at their bounded densities over all time windows over $I(p)$.

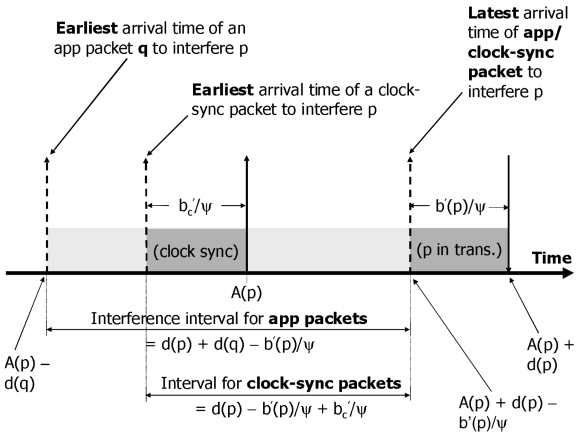


Fig. 13. Packet interference under the UPA.

Upper bound $u_1^i(p)$. $u_1^i(p)$ is established by observing that any application packet q will be scheduled by the UPA on source s_i before packet p, **only if** q arrives no sooner than $A(p) - d(q)$ and no later than $A(p) + d(p) - \frac{b'(p)}{\psi}$. This is because, if q were to arrive before $A(p) - d(q)$, then its absolute deadline will occur before p's arrival. Thus, when p arrives, the UPA has either scheduled q or has dropped q because it has become infeasible.

Similarly, if application packet q were to arrive after $A(p) + d(p) - \frac{b'(p)}{\psi}$, then, at that time, the UPA would have either scheduled p or has dropped p because it has become infeasible. Note that q cannot be scheduled before p once p has been scheduled and is in transmission (even if q were to arrive before $A(p) + d(p)$) since packet transmission is nonpreemptive.

Note that, if q arrives after $A(p) + d(p) - d(q)$ but before $A(p) + d(p) - \frac{b'(p)}{\psi}$, q's absolute deadline will occur after that of p. Under the EDF, q will then be scheduled after p since q has a longer absolute deadline than that of p. However, under the UPA, it is quite possible that q can be scheduled before p. Thus, with the EDF, the latest arrival time of q after which q cannot be scheduled before p will occur at $A(p) + d(p) - d(q)$; with the UPA, this will occur at $A(p) + d(p) - \frac{b'(p)}{\psi}$.

Further, UPA will schedule a clock-sync packet q on s_i before packet p **only if** q arrives no sooner than $A(p) - \frac{b'(c)}{\psi}$ and no later than $A(p) + d(p) - \frac{b'(p)}{\psi}$. If q were to arrive before $A(p) - \frac{b'(c)}{\psi}$, then the maximum transmission time that it needs will occur before p's arrival. Thus, when p arrives, UPA has already scheduled q for transmission.

Fig. 13 illustrates the packet interference situation under UPA. It follows that:

$$u_1^i(p) = \sum_{q \in P_i} \sum_{\pi=1}^{k} \left\lceil \frac{\left[d(p) + d(q) - \frac{b'(p)}{\psi} \right]}{w_\pi(q)} \right\rceil a_\pi(q) + \left\lceil \frac{d(p) - \frac{b'(p)}{\psi} + \frac{b'(c)}{\psi}}{\theta} \right\rceil.$$

Now, $R_1(s_i,p)$ is given by the sum of 1) the time needed to physically transmit $u_1^i(p)$ packets at throughput ψ and 2) the upper bounds on aggregate Worst-Case Execution Times (WCETs) for the UPA to schedule $u_1^i(p)$ packets. Given a WCET δ_h for the UPA at a host, $R_1(s_i,p)$ equals:

$$\sum_{q\in P_i}\sum_{\pi=1}^{k}\left\lceil\frac{\left\lceil d(p)+d(q)-\frac{b'(p)}{\psi}\right\rceil}{w_\pi(q)}\right\rceil a_\pi(q)\left\lceil\frac{b'(q)}{\psi}+\delta_h\right\rceil + \left\lceil\frac{d(p)-\frac{b'(p)}{\psi}+\frac{b'(c)}{\psi}}{\theta}\right\rceil\left\lceil\frac{b'_c}{\psi}+\delta_h\right\rceil.$$

11.3.2 Construction of $R_2(p)$

$R_2(p)$ can be established similarly to $R_1(s_i,p)$. The only difference is that we need to consider all packets q that can arrive from all sources such that they will contend for the same outgoing network segment at the switch as p. Since packet destinations are not specified in the models, we consider all packets $q \in P$.

Similarly to $R_1(s_i,p)$, to determine $R_2(p)$, we need to determine $u_2(p)$, i.e., an upper bound on the number of packets belonging to set P that will be scheduled for outbound transmission by the UPA on the switch over any interval $I(p)=[A(p),A(p)+d(p)]$, where $A(p)$ is packet p's arrival time at the switch MAC-layer and $d(p)$ is its relative deadline.

Upper bound $u_2(p)$. Upper bound $u_2(p)$ is established by observing that any application packet q will be scheduled by the UPA on the switch before packet p **only if** q arrives no sooner than $A(p)-d(q)$ and no later than $A(p)+d(p)-\frac{b'(p)}{\psi}$. Furthermore, any clock-sync packet q will be scheduled by the UPA on the switch before packet p **only if** q arrives no sooner than $A(p)-\frac{b'(c)}{\psi}$ and no later than $A(p)+d(p)-\frac{b'(p)}{\psi}$. The rationale for these is exactly the same as that for establishing the bound $u_1^i(p)$. It follows that:

$$u_2(p)=\sum_{q\in P}\sum_{\pi=1}^{k}\left\lceil\frac{\left\lceil d(p)+d(q)-\frac{b'(p)}{\psi}\right\rceil}{w_\pi(q)}\right\rceil a_\pi(q) + \left\lceil\frac{d(p)-\frac{b'(p)}{\psi}+\frac{b'(c)}{\psi}}{\theta}\right\rceil.$$

Now, $R_2(p)$ is given by the sum of 1) the time needed to physically transmit $u_2(p)$ packets at throughput ψ and 2) the upper bounds on aggregate WCETs for the UPA to schedule $u_2(p)$ packets and for moving the packets (inside the switch) from switch input port to output port. Given a WCET and input-to-output port transfer time δ_s for the UPA at the switch, $R_2(p)$ equals:

$$\sum_{q\in P}\sum_{\pi=1}^{k}\left\lceil\frac{\left\lceil d(p)+d(q)-\frac{b'(p)}{\psi}\right\rceil}{w_\pi(q)}\right\rceil a_\pi(q)\left\lceil\frac{b'(q)}{\psi}+\delta_s\right\rceil + \left\lceil\frac{d(p)-\frac{b'(p)}{\psi}+\frac{b'(c)}{\psi}}{\theta}\right\rceil\left\lceil\frac{b'_c}{\psi}+\delta_s\right\rceil.$$

11.3.3 Feasibility Conditions

Thus, the conditions are $ATB=\sum_{i=1}^{z}\sum_{p_j\in P_i}U_j\big(R(s_i,p_j)\big) \geq ATB_l$, where $R(s_i,p_j)=R_1(s_i,p_j)+R_2(p_j)$. Now, to dimension a system to an SE problem, an assignment of values to *unvalued variables* in models and properties must be made. Unvalued variables in models include:

1. number of packets n,
2. packet sizes $b(p_i), i\in[1,n]$,
3. number of sources z,
4. mappings $P_j, j\in[1,z]$,
5. number of arrival sequences k, and
6. arrival densities $\langle(a_\pi(p_i),w_\pi(p_i)),\pi\in[1,k]\rangle, i\in[1,n]$.

Unvalued variables in properties include TUFs $U_i, i\in[1,n]$ and utility lower bound ATB_l.

The resulting quantified problem instance must then be subject to feasibility analysis using the conditions. If a feasible solution exists, a quantified system with values assigned to unvalued variables in the solution including network throughput ψ and WCET/port-to-port transfer times δ_h and δ_s can be obtained. Thus, the feasibility conditions facilitate the design of TUF-driven switched Ethernets with guaranteed soft timeliness properties.

12 Conclusions and Future Work

Our main conclusion is that the UPA can achieve significantly higher accrued utility than the CMA and EDF for a broad set of TUFs. UPA's advantage is significant when the TUF pseudoslope used by the algorithm matches the actual TUF slope. Thus, the algorithm performs the best for linear TUFs, followed by soft-step and quadratic TUFs. UPA's performance is the least significant for step TUFs. This is good news as TUFs such as linear, soft-step, and quadratic are closest variants of realistic TUFs (e.g., AWACS).

The advantage of the UPA is also significant at larger message traffic. At smaller traffic, when packet contention for network segments is small, the algorithm has lesser advantages due to its larger overhead compared with simple algorithms such as FIFO. However, this is also good news as supervisory real-time systems—our target systems—are frequently subject to significant runtime increases in message traffic.

Several aspects of the work are directions for further research. The UPA's timeliness feasibility conditions that are presented here are sufficient, but not necessary. So, one direction is to develop necessary and sufficient, tractable feasibility conditions. Another interesting direction is to extend the system model for multihop networks that include multiple switches and routers, develop utility accrual algorithms, and derive feasibility conditions.

Acknowledgments

This work was supported by the US Office of Naval Research under Grant N00014-99-1-0158 and N00014-00-1-0549. The authors thank Peng Li and Dr. Douglas Jensen for the many insightful discussions and the anonymous reviewers for their helpful comments.

REFERENCES

[1] H. Kopetz, A. Damm, C. Koza, M. Mulazzani, W. Schwabl, C. Senft, and R. Zainlinger, "Distributed Fault-Tolerant Real-Time Systems: The Mars Approach," *IEEE Micro,* vol. 9, no. 1, pp. 25-40, Feb. 1989.

[2] C. Venkatramani and T.-C. Chiueh, "Supporting Real-Time Traffic on Ethernet," *Proc. IEEE Real-Time Systems Symp.,* pp. 282-286, Dec. 1994.

[3] D.W. Pritty, J.R. Malone, S.K. Banerjee, and N.L. Lawrie, "A Real-Time Upgrade for Ethernet Based Factory Networking," *Proc. IEEE/IECON Conf. Industrial Electronics, Control, and Instrumentation,* pp. 1631-1637, 1995.

[4] W. Kim and J. Srivastava, "New Virtual Time CSMA/CD Protocols for Real-Time Communication," *Proc. IEEE Conf. Comm. for Distributed Applications and Systems,* pp. 11-22, 1991.

[5] W. Zhao and K. Ramamritham, "A Virtual Time CSMA/CD Protocol for Hard Real-Time Communication," *Proc. IEEE Real-Time Systems Symp.,* pp. 120-127, Dec. 1986.

[6] M.N. El-Derini and M.R. El-Sakka, "A CSMA Protocol under a Priority Time Constraint for Real-Time Communication," *Proc. IEEE Workshop Future Trends of Distributed Computing Systems,* pp. 128-134, 1990.

[7] W. Zhao, J.A. Stankovic, and K. Ramamritham, "A Window Protocol for Transmission of Time-Constrained Messages," *IEEE Trans. Computers,* vol. 39, no. 9, pp. 1186-1203, Sept. 1990.

[8] S.K. Kweon and K.G. Shin, "Achieving Real-Time Communication over Ethernet with Adaptive Traffic Smoothing," *Proc. IEEE Real-Time Technology and Applications Symp.,* pp. 90-100, 2000.

[9] J.-F. Hermant and G. LeLann, "A Protocol and Correctness Proofs for Real-Time High-Performance Broadcast Networks," *Proc. IEEE Conf. Distributed Computing Systems,* pp. 360-369, 1998.

[10] D. Kim, Y. Doh, and Y. Lee, "Table Driven Proportional Access Based Real-Time Ethernet for Safety-Critical Real-Time Systems," *Proc. IEEE Pacific Rim Symp. Dependable Computing,* pp. 356-363, 2001.

[11] S. Varadarajan and T.-C. Chiueh, "Ethereal: A Host-Transparent Real-Time Fast Ethernet Switch," *Proc. IEEE Conf. Network Protocols,* pp. 12-21, Oct. 1998.

[12] SIXNET, "The Sixnet Industrial Ethernet Switch," http://www.sixnetio.com, Year?

[13] H. Hoang, M. Jonsson, U. Hagström, and A. Kallerdahl, "Switched Real-Time Ethernet and Earliest Deadline First Scheduling-Protocols and Traffic Handling," *Proc. IEEE Workshop Parallel and Distributed Real-Time Systems,* pp. 94-99, Apr. 2002.

[14] C. Baek-Young, S. Sejun, N. Birch, and J. Huang, "Probabilistic Approach to Switched Ethernet for Real-Time Control Applications," *Proc. IEEE Conf. Real-Time Computing Systems and Applications,* pp. 384-388, 2000.

[15] H. Zhang, "Service Disciplines for Guaranteed Performance Service in Packet Switching Network," *Proc. IEEE,* vol. 83, no. 10, pp. 1374-1396, Oct. 1995.

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

[17] E.D. Jensen, "Asynchronous Decentralized Real-Time Computer Systems," *Real-Time Computing,* W.A. Halang and A.D. Stoyenko, eds., NATO Advanced Study Inst., Oct. 1992.

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

[19] E.D. Jensen, A. Kanevsky, J. Maurer, T. Wheeler, Y. Zhang, D. Wells, T. Lawrence, and P. Hurley, "An Adaptive, Distributed Airborne Tracking System," *Proc. IEEE Workshop Parallel and Distributed Real-Time Systems,* Apr. 1999.

[20] 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," technical report, Computer Science Dept., Carnegie Mellon Univ., Dec. 1988, Archons Project Technical Report 88121.

[21] D.L. Mills, "Improved Algorithms for Synchronizing Computer Network Clocks," *IEEE/ACM Trans. Networking,* vol. 3, pp. 245-254, June 1995.

[22] J.W.S. Liu, *Real-Time Systems.* Prentice Hall, 2000.

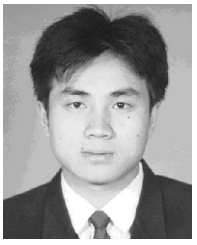
[23] ZNYX Networks, "Zx340q Series," http://www.znyx.com/products/netblaster/zx340q.htm, Year?

[24] D. Mills, "xntpd," http://www.eecis.udel.edu/ntp/database/html_xntp3-5.90/xntpd.html, Year?

[25] U. Böhme and L. Buytenhenk, "Linux bridge-stp-howto," http://www.tldp.org/HOWTO/BRIDGE-STP-HOWTO, Year?

[26] J. Wang, "Soft Real-Time Switched Ethernet: Best-Effort Packet Scheduling Algorithm, Implementation, and Feasibility Analysis," master's thesis, Virginia Tech., Sept. 2002.

[27] G. LeLann, "Proof-Based System Engineering and Embedded Systems," *Lecture Notes in Computer Science,* G. Rozenberg and F. Vaandrager, eds., vol. 1494, pp. 208-248, Oct. 1998.



Jinggang Wang received the BS degree from Beijing University of Aeronautics and Astronautics (BUAA), China, in 1992, and the MS degree from the Virginia Polytechnic Institute and State University, in 2002, both in electrical engineering. Previously, he was employed by Industrial and Commercial Bank of China (ICBC), Huizhou, China, where he developed and managed several large-scale, commercial software projects. Currently, he is a senior software engineer with the Embedded System Division, Casabyte Inc., a company that focuses on world-wide 3G wireless network QoS. His research interests include resource management in wireless network and embedded mobile systems, embedded real-time operating systems, real-time networking, and self-driven computing.

Binoy Ravindran received the master's degree from the New Jersey Institute of Technology (NJIT) in 1994 and the PhD degree from the University of Texas at Arlington (UT Arlington) in 1998, both in computer science. He is an assistant professor in the Bradley Department of Electrical and Computer Engineering at Virginia Tech, Blacksburgh. He received two "Student Achievement Awards" from NJIT for outstanding research by a graduate student (1994, 1995) and a "Doctoral Dissertation Award" from UT Arlington for outstanding doctoral research (1998). His current research focus includes real-time distributed systems having application-level, end-to-end quality of services (e.g., timeliness, survivability, security). Recent sponsors of Dr. Ravindran's research include The MITRE Corporation, US Office of Naval Research, and NASA. During 1999, he was an invited speaker at INRIA, France, and at the ARTES (Swedish National Strategic Research Initiative in Real-Time Systems) Real-Time Week, Sweden. He served as the program chair of the 1999 IEEE International Workshop on Parallel and Distributed Real-Time Systems. Dr. Ravindran was a coguest editor for *IEEE Transactions on Computers* (2002) for a special issue on "asynchronous real-time distributed systems." He is a member of the IEEE and the IEEE Computer Society.

▷ **For more information on this or any computing topic, please visit our Digital Library at** http://computer.org/publications/dlib.