A Systems Engineering Approach for Constructing Certifiable Real-Time Distributed Systems

Binoy Ravindran*, Gérard Le Lann†, Jinggang Wang*, and Peng Li*

*Real-Time Systems Laboratory
ECE Dept., Virginia Tech, Blacksburg
VA 24061, USA
{binoy,jiwang5,peli2}@vt.edu

†INRIA
Domaine de Voluceau, B.P. 105
78153 Le Chesnay Cedex, France
Gerard.Le_Lann@inria.fr

Abstract

In this paper, we present a systems engineering methodology for constructing certifiable real-time distributed systems. In the proposed approach, an architectural and algorithmic solution to an application problem is designed by considering the "weakest" models including the weakest asynchronous computational model and multimodal arrival model. Furthermore, timeliness properties are described using Jensen's benefit accrual predicates. Once a system solution is designed, timeliness properties are established by constructing necessary feasibility conditions that are expressed as non-valued predicates. The predicates are quantified and verified to produce the specification of a certified solution. We illustrate the approach by considering a packet transmission problem that desire soft timeliness. We present a certifiable solution to this problem that consists of switched Ethernet, a soft real-time packet scheduling algorithm (that was previously developed), and feasibility conditions.

1. Introduction

Asynchronous real-time distributed systems emerging in many domains are distinguished by the significant run-time uncertainties that are inherent in their application environment, system resource states, and failure occurrences [8]. Consequently, upper bounds on timing variables in such systems including duration of computational and communication steps—manifestations of application workloads and execution environment characteristics—are not known to exist at design time with sufficient accuracy. Furthermore, many of the emerging asynchronous real-time distributed systems are also safety-critical [13, 9]. Therefore, end-users of such systems require guaranteed assurance on the delivery of desired system properties, particularly safety. This defines a *certification requirement*. Asynchronous real-time distributed systems thus raise fundamental issues: "How to build timely systems that operate in the presence of uncertain timeliness? Furthermore, how to certify that such systems will deliver properties including timeliness and safety?"

In this paper, we answer these questions by discussing a proof-based systems engineering (SE) approach that builds upon the previously developed TRDF method [9]. Central to this approach is our argument that in order to maximize the "coverage" of desired system properties such as timeliness, safety, and liveness, asynchronous models and/or asynchronous designs should be favored against synchronous ones. Furthermore, contrary to popular belief, we argue that it is possible to establish timeliness properties including hard and soft timeliness properties for systems constructed using asynchronous solutions (e.g., algorithms, protocols, etc.).

In the subsections that follow, we motivate our approach by first discussing issues that we regard as fundamental to real-time distributed systems, namely, computational models and timeliness optimality. We then overview our approach.

1.1. Computational Models

As defined in [12], computational models range from pure synchronous to pure asynchronous. Pure synchrony means that duration of every computational and communication steps have upper bounds that are known at design time, whereas pure asynchrony means that no such upper bounds are known to exist.

Asynchronous computational models have the well known advantage that properties such as safety and liveness can be established even when the "adversary" embodied in design assumptions

Proceedings of the Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'03)
0-7695-1928-8/03 $17.00 © 2003 IEEE

are violated. Examples include asynchronous consensus algorithms. However, the "curse" of such models is that many problems of interest do not have known deterministic algorithms due to impossibility results [5]. To circumvent this, researchers have augmented the pure asynchrony model with additional semantics, including *timed* semantics and *time-free* semantics. In the pure asynchrony model augmented with timed semantics, called the partially synchronous model, some system modules have pure synchrony semantics and others have arbitrary semantics including pure asynchrony semantics. Pure asynchrony models augmented with time-free semantics are simply called asynchronous models [6]. Examples include unreliable failure detectors [2].

Researchers have also defined the notion of "weakest asynchronous" and "weakest partially synchronous" models [6]. The weakest model is a model that is necessary and sufficient for implementing some given time-free semantics. Thus, a given problem is solved using the weakest partially synchronous model if and only if some minimal set of modules in the solution match pure synchrony assumptions and every other match pure asynchrony assumptions.

1.2. Timeliness Optimality

The vast majority of timing constraints and timeliness optimality criteria that are currently used for constructing real-time distributed systems include deadlines and hard timeliness optimality, respectively. We believe that timing constraints and timeliness optimality are best described using Jensen's benefit functions [7] and benefit accrual predicates [7], respectively. The rationale for our belief is due to the following reasons:

1.2.1 Difficulty with Deadlines

Deadline timing constraints have the drawback that they implicitly (or explicitly) indicate that the deadlines are "hard" [7]. Thus, completing a deadline-constrained activity before its deadline implies the accrual of some "benefit" and that benefit remains the same if the activity were to complete *anytime* before the deadline. Furthermore, completing the activity after the deadline implies a timing failure i.e., the accrual of zero benefit.

With deadlines, it therefore becomes difficult to express timing constraints that are not hard, but "soft" in the sense that completing the time-constrained activity at anytime will result in some benefit and that benefit *varies* with the activity completion time. Furthermore, with deadlines, it becomes difficult to specify a timeliness optimality criterion that is not hard, but soft in the sense that completing as many soft time-constrained activities as possible at their *optimal* completion times—completion times that will yield maximal benefit—is what is desirable.

Many timing constraints in real-time distributed systems are soft in the aforementioned sense. Examples include [7, 18, 4]. With deadline-based scheduling algorithms such as EDF [11], such soft timing constraints must be converted to deadlines. This will cause the deadline-based algorithms to seek the completion of the soft activities at anytime before their deadlines, violating their *non-uniform* benefit semantics.

On the contrary, benefit functions allow the semantics of soft timing constraints to be precisely specified. This allows scheduling algorithms that use such specifications to seek the completion of soft activities that are precisely consistent with their timing constraint specifications i.e., seek to complete soft activities at times that will yield their optimal benefit.

1.2.2 Difficulty During Overloads

Given that we cannot predict the future, it is possible that the actual operating conditions of a system may be "stronger" than what was assumed when the system was designed. Thus, it is possible that the models that are used to design a real-time system solution can be violated by the "adversaries" embodied in the models when the system is in operation. When such conditions occur, scheduling is complicated with deadlines and collective timeliness criteria that are specified with deadlines in that they do not indicate what objectives must be sought (during such situations).

On the other hand, benefit accrual predicates allow the specification of timeliness optimality criterion that facilitate application timeliness to be optimized in an application-specific and inherently adaptive way. For example, a highly desirable criterion would be to complete all activities at their optimal completion times if situation permits, such as during conditions when the "adversaries" behave as reasoned and permitted by the design assumptions. Furthermore, when the "adversaries" violate design assumptions, a desirable criterion would be to complete as many activities as possible at their optimal completion times, less at their suboptimal completion times, and thereby facilitate graceful degradation of application timeliness. Such timeliness optimality criteria can be specified with benefit accrual predicates.

1.3. A Systems Engineering Approach

We present a *systems engineering* (SE) approach for constructing certifiable real-time dis-

Proceedings of the Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'03)
0-7695-1928-8/03 $17.00 © 2003 IEEE

tributed systems. We presented the preliminary concepts of this approach in [14].

The approach starts with describing the *weakest possible models* and desired properties of the desired computer-based system.

Models are the set of assumptions regarding the future operational conditions of the desired computer-based system. Models include (1) computational model, where the asynchronous model dominates all others such as partially synchronous and pure synchronous models [10], (2) external event arrival model, where the multimodal arrival model dominates all others such as unimodal, aperiodic, sporadic, and periodic arrival models, and (3) failure model, where the byzantine model dominates all others such as crash and omission models [9]. The domination of one model over another is due to the "strength" of the "adversary" embodied in the model.

Properties are the desired services that the desired computer-based system must provide to end-users. Example properties include timeliness, safety, and liveness. Models and properties are invariants of the design problem.

We consider the weakest possible models so that impossibility results if any, can be circumvented, and system solutions that have the maximum possible "coverage" of the desired properties can be designed. Examples include the *weakest asynchronous computational model* i.e., the pure asynchronous model that is augmented with time-free semantics such as [2], as such models have the well known advantage that properties such as safety and liveness can be established even when the "adversaries" embodied in the design assumptions are violated. Furthermore, we specify timeliness optimality using benefit accrual predicates such as a user-desired lower bound on system-wide, aggregate (e.g., total) timeliness benefit.

With such computational models and timeliness properties, an architectural and algorithmic solution to a given application problem is then designed. Once a system solution is designed, safety and liveness are first provably correctly established for the solution. Timeliness properties are later established by constructing *timeliness feasibility conditions* for the solution that are proven to be necessary, and if tractable, sufficient as well. Such conditions are constructed as non-valued functions that will embody all possible scenarios that can be deployed by the "adversaries" considered in the design models.

The timeliness feasibility conditions are then *quantified* by assigning numerical values to unvalued variables in the solution such as processor speeds and network throughputs. The resulting quantified instance of the solution is then verified to ensure that the solution satisfies the feasibility conditions.

The quantification and verification of the feasibility conditions thus produce the specification of a computer-based system solution that can be certified to exhibit the desired timeliness, safety, and liveness properties. Thus, the fundamental philosophy of the approach is that synchrony need to be considered only for establishing timeliness properties.

The rest of the paper is organized as follows:

To illustrate our SE approach, we consider a design subproblem that is concerned with the transmission of message packets, which must exhibit soft timeliness properties. We call this problem the Soft Real-time Packet Transmission (or SRPT) problem. We discuss SRPT in Section 2 by simply sketching the models and properties of the problem. In Section 3, we present a solution to SRPT using our SE approach. The solution comprises of (1) switched Ethernet, (2) a soft real-time packet scheduling algorithm called BPA, and (3) timeliness feasibility conditions.

The timeliness feasibility conditions of SRPT that we present in Section 3 is the main contribution of the paper. To the best of our knowledge, we are not aware of any such feasibility conditions for guaranteeing soft timeliness property.

Finally, the paper concludes by summarizing the work and discussing future work in Section 4.

2. The SRPT Design Subproblem

2.1. Models of SRPT

The set of application packets is denoted as $p_i \in P, i \in [1, n]$. The size n of the set P is unrestricted.

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).

Packets are generated from a set of sources, $s_i \in S, i \in [1, z]$. The number of sources z is unrestricted. The packet subset $P_j \subseteq P$ is mapped onto source $s_j \in S, j \in [1, z]$. The mapping is unrestricted i.e., any packet $p_j \in P$ can be mapped onto any source $s_j \in S$.

The arrival law for packets is the multimodal arbitrary arrival model [9]. Multimodal arrival model is a finite, ordered sequence of unimodal arrivals. For a packet p_i, the unimodal arrival 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 window $w(p_i)$. Thus, the sequence

Proceedings of the Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'03)
0-7695-1928-8/03 $17.00 © 2003 IEEE

$\langle (a_\pi(p_i), w_\pi(p_i)), \pi \in [1,k] \rangle, i \in [1,n]$ is defined, where the number of sequences k is unrestricted.

Each source $s_i \in S$ is equipped with a clock synchronization module, denoted C_i. (We discuss the motivation for clock synchronization in Section 3.2.) The clock synchronization module generates clock synchronization packets periodically with a period θ. Such packets are assumed to be always transmitted before application packets are transmitted.

The bit length of a clock synchronization packet at the data link layer is a constant and is denoted as b_c. The physical framing overheads increase this size into an actual bit length $b'_c > b_c$.

Thus, we are considering a weak asynchronous model. The only synchronous assumption that is needed to implement this model is the synchrony of clocks.

2.2. Properties of SRPT

The only property of interest is timeliness. The timeliness property is defined using benefit functions and benefit accrual predicates as follows:

Each packet $p_i \in P$ has a *unimodal* benefit function that is *non-increasing*, denoted as B_i. Unimodal benefit functions are those benefit functions for which any decrease in benefit cannot be followed by an increase in benefit [7]. Benefit functions, which are not unimodal are called *multimodal*.

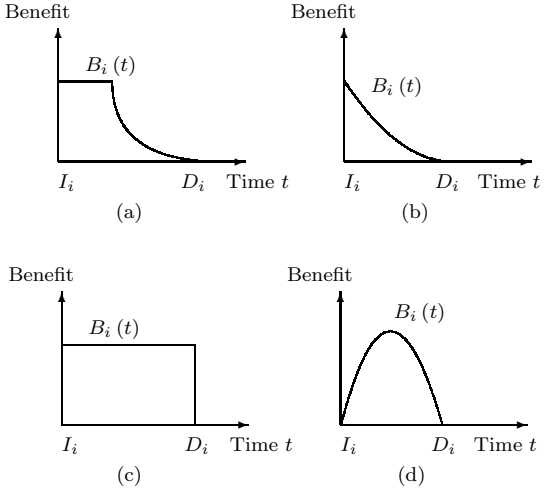

Figure 1. Example Unimodal Functions

Example unimodal functions are shown in Figure 1. Note that the classical "hard" deadline can be expressed as a "rectangular" unimodal function such as the one shown in Figure 1(c), where the arrival of a packet at its destination at anytime before its deadline will result in uniform benefit; the arrival of the packet after the deadline will result in zero benefit. All non-rectangular benefit functions express soft timing requirements [7].

Example multimodal benefit functions are shown in Figure 2.

Unimodal functions that are non-increasing are simply those benefit functions for which benefit never increases when time advances. Figures 1(a), 1(b), and 1(c) show examples. The class of non-increasing unimodal functions allow the specification of a broad range of timing constraints, including hard constraints and a majority of soft constraints. Therefore, we focus on non-increasing unimodal benefit functions here.

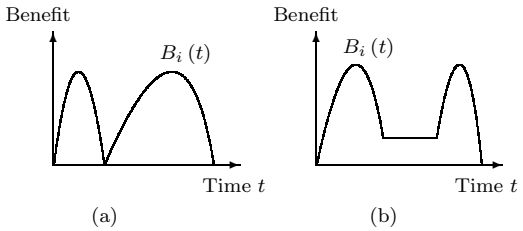

Figure 2. Example Multimodal Functions

All benefit functions $B_i, i \in [1,n]$ have an initial time I_i and a deadline time D_i. Initial time is the earliest time for which the function is defined and deadline time is the time at which the function drops to a zero value. Furthermore, $B_i(t) \geqslant 0, \forall t \in [I_i, D_i], i \in [1,n]$ and $B_i(t) = 0, \forall t \notin [I_i, D_i], i \in [1,n]$.

The timeliness property is a desired lower bound on system-wide, aggregate timeliness benefit denoted ATB_l, where the system-wide, aggregate timeliness benefit denoted ATB, is defined as the sum of the individual benefit accrued by the arrival of packets at their destinations. Thus, $ATB = \sum_{i=1}^{n} B_i(t_i) \geqslant ATB_l$, where t_i is the absolute time at which packet p_i arrives at its destination end-host, since its release at its source end-host. The functions $B_i, i \in [1,n]$ and the value of ATB_l are unknown.

3. A Solution Using Switched Real-Time Ethernet

A solution to SRPT involves two components: (1) an algorithmic solution that is associated with some architectural solution; and (2) timeliness feasibility conditions. Our solution to SRPT consists of a single-segment switched real-time Ethernet, a MAC-layer, real-time packet scheduling algorithm called Best-Effort Packet Scheduling Algorithm (or BPA), and timeliness feasibility conditions.

We had presented BPA, its implementation, and construction of a switched real-time Ether-

Proceedings of the Sixth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'03)
0-7695-1928-8/03 $17.00 © 2003 IEEE

net in [17]. To discuss our solution to SRPT, we first overview switched Ethernets and the BPA algorithm. We then derive the timeliness feasibility conditions.

3.1. Switched Ethernet Networks

In switched Ethernet networks, end-hosts are interconnected through switches using full-duplex Ethernet segments. Furthermore, the switch determines the destination host of incoming packets and schedules them on the appropriate outgoing network segment. Thus, unlike in shared Ethernets, where a hub simply broadcasts every packet that it receives and causes collisions, a switch "directs" the traffic by scheduling packets on the "right" outgoing network segment.

The switched real-time Ethernet approach, which is gaining wide support in the real-time industry is presented in EtheReal [16], SIXNET Industrial Ethernet Switch [15], and [1].

We consider a single-segment switched Ethernet network, where end-hosts are interconnected through a single switch as our architectural solution to the design subproblem.

In single-segment switched Ethernets, packets first arrive at the MAC-layer of source end-hosts where they are generated. Upon arrival, they are queued in the outgoing packet queue of the host. Furthermore, when the network segment from the host to the switch becomes "free" for transmission, it triggers the packet scheduling algorithm at the end-host, which then executes and schedules a packet from the packet queue for transmission.

The switch maintains a list of packet ready-queues, one queue per host. Each queue stores the packets that are destined for the corresponding host. When packets arrive at the switch, they are queued in the outgoing packet queue at the switch for the corresponding destination host. When the network segment from the switch to an end-host becomes free for transmission, it triggers the packet scheduling algorithm at the switch, which then executes and schedules a packet from the packet queue of the destination host for transmission (on the corresponding output port).

Thus, time-constrained application packets (of the SRPT problem) must be scheduled at the MAC-layer of source end-hosts as well as that at the switch using an appropriate real-time scheduling algorithm (at the end-host and at the switch, respectively) such that the system-wide, trans-node, end-to-end timeliness property is satisfied.

3.2. Overview of BPA

BPA is a packet scheduling algorithm that executes at the MAC-layer of end-hosts and the switch for selecting packets for outbound transmission. The algorithm considers a packet model, where packets have *non-increasing, unimodal benefit function constraints* and seeks to *maximize* the aggregate benefit that is accrued when packets arrive at their destinations i.e., *Maximize* $\sum_{i=1}^{n} B_i(t_i)$, where t_i is the absolute time at which packet p_i arrives at its destination. This optimization problem is $\mathcal{NP}$-hard [17]. In [17], we show that BPA is the "best" heuristic algorithm for this problem, outperforming the previously known best algorithm presented in [3].

BPA is an asynchronous algorithm i.e., it is "time-free." The algorithm is invoked whenever the outgoing network link (from an end-host or that from the switch) becomes "free" for transmission. Thus, the only scheduling event of the algorithm is the release of the network resource by a packet.

3.2.1 BPA Heuristics

BPA first constructs a tentative schedule by sorting packets in decreasing order of their "return of investments." The return of investment for a packet is the potential timeliness benefit that can be obtained by spending a unit amount of network transmission time for the packet. Thus, "high return" packets will appear early in the tentative schedule. The return of investment for a packet is determined by computing the slope of the packet benefit function.

From this tentative schedule, packets that are found to be infeasible are moved to the end of the schedule. 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 are longer than the time interval between their arrival at an end-host or the switch and the packet deadlines. Packets that are not infeasible are feasible packets.

To determine whether a packet is infeasible, the algorithm therefore needs global time. Thus, as discussed previously, we assume that the end-hosts and the switch have access to synchronized clocks.

Once infeasible packets are moved to the schedule-end, the algorithm maximizes the *local* aggregate benefit in the resulting schedule. The local aggregate benefit is maximized by observing that given 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 $\sigma_1 \bigcap \sigma_2 = \emptyset$, the scheduling decision at a time t, where $t = \sum_{k \in \sigma_1} l_k$, that will lead to maximum local aggregate benefit is determined by computing $\Delta_{i,j}(t)$, where $\Delta_{i,j}(t) = [B_i(t + l_i) + B_j(t + l_i + l_j)] - [B_j(t + l_j) + B_i(t + l_j + l_i)]$. Thus, if $\Delta_{i,j}(t) \geq$

0, then schedule σ_a will yield a higher aggregate benefit than σ_b.

BPA maximizes local aggregate benefit by examining adjacent pairs of packets in the schedule, computing Δ, and swapping the packets, if the reverse order can lead to higher local aggregate benefit. The procedure is repeated until no swaps are required. The packet that appears first in the resulting schedule is then selected for transmission.

In [17], we show that given n application packets, BPA has a worst-case complexity of $O\left(n^2\right)$, which is faster than the $O\left(n^3\right)$ cost of the algorithm presented in [3].

It is important to note that the superiority of BPA is not affected by the accuracy and precision of global time that is achieved through synchronized clocks (i.e., clock drift rates). Given that any *implemented* solution to the SRPT problem must rely on global time (in order to behave as closely as possible to optimality), BPA's superiority is not affected, nor is it jeopardized when considering multi-segment switched Ethernets.

3.3. Timeliness Feasibility Conditions

In order to establish the timeliness feasibility conditions, we must now "bind" BPA to a synchronous model that matches the switched Ethernet system model that we are considering. That is, synchrony assumptions are now needed to establish the feasibility conditions. This "late binding" process permits to translate the time-free or event-based activation conditions of BPA into timed conditions.

To establish the desired timeliness property, we need to determine the worst-case lower bound on the system-wide, aggregate timeliness benefit that is possible under the models of SRPT. This can be determined by first determining the lower bound on the individual benefit that is accrued by the arrival of any given packet at its destination. Aggregation of the individual benefit lower bounds will yield the lower bound on the system-wide benefit.

To determine the lower bound on individual benefit, we need to determine an upper bound on the packet delay. We thus seek to construct a computable function denoted $R(s_i, p)$, that gives an upper bound on the delay incurred by any packet p to arrive at its destination, since its arrival at the MAC-layer of any source $s_i, i \in [1, z]$ for outbound transmission.

In a single-segment switched network, a packet will experience contention for *two* network resources once it arrives at the MAC-layer of its source. The resources include (1) the network segment from the source to the switch and (2) the network segment from the switch to the destination. Thus, we define $R(s_i, p) = R_1(s_i, p) + R_2(p)$, where $R_1(s_i, p)$ 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.

3.3.1 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 the arrival time of the packet p at the MAC-layer of the source s_i and let $d(p)$ denote its relative deadline (i.e., relative to the arrival time $A(p)$). Let $I(p)$ denote the time 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 subset P_i that will be scheduled for outbound transmission (by BPA) on source s_i, over any interval $I(p)$, *before* packet p is transmitted. We denote this upper bound as $u_1^i(p)$.

The bound $u_1^i(p)$ must be established considering the strongest possible "adversary" that is embodied in the arrival model i.e., assuming that packet arrivals occur at their bounded densities over all time windows over $I(p)$.

3.3.2 Upper bound $u_1^i(p)$

Upper bound $u_1^i(p)$ is established by observing that any packet q will be scheduled by BPA on source s_i before packet p, only if packet 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 packet q were to arrive before $A(p) - d(q)$, then its absolute deadline will occur before the arrival of packet p. Thus, when packet p arrives, BPA has either already scheduled packet q for transmission or has dropped packet q because it has become infeasible.

Similarly, if packet q were to arrive after $A(p) + d(p) - \frac{b'(p)}{\psi}$, then at that time, BPA would have either already scheduled packet p for transmission 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 non-preemptive.

Note that if packet q were to arrive after $A(p) + d(p) - d(q)$ but before $A(p) + d(p) - \frac{b'(p)}{\psi}$, the absolute deadline of packet q will occur after that of p. Under an EDF packet scheduler, this will cause packet q to be scheduled after packet p, since packet q has a longer absolute deadline than that of p. However, under BPA, it is quite possible that packet q can be scheduled before packet p. Thus, with EDF, the latest arrival time of q after which q cannot be scheduled before p will occur

at $A(p)+d(p)-d(q)$, whereas with BPA, this will occur at $A(p)+d(p)-\frac{b'(p)}{\psi}$.

Since $\left[A\left(p\right)+d\left(p\right)-\frac{b'(p)}{\psi}\right]-\left[A\left(p\right)-d\left(q\right)\right]=d\left(p\right)+d\left(q\right)-\frac{b'(p)}{\psi}$, it follows that:

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

Now, $R_1(s_i,p)$ is given by the sum of (1) the time needed to physically transmit $u_1^i\left(p\right)$ packets at throughput ψ and (2) the upper bounds on aggregate worst-case execution times for BPA to make scheduling decisions for $u_1^i\left(p\right)$ packets. The worst-case execution time of BPA will depend upon factors such as processor speed and memory access latencies. Given a COTS product that guarantees a worst-case execution time δ_h for BPA at an end-host, $R_1(s_i,p)$ is given by $R_1\left(s_i,p\right)=$

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

3.3.3 Construction of $R_2(p)$

$R_2(p)$ is the upper bound on the delay incurred by any packet p to arrive at its destination, since its arrival at the switch. This upper bound can be established in a manner similar to that of $R_1(s_i,p)$. The only difference is that we now need to consider all packets q that can arrive from all sources $s_i, i\in[1,z]$ such that they will contend for the same outgoing network segment at the switch as that of packet p. Since the destination of packets is not specified in the models of SRPT, we will consider all packets $q\in P$.

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

3.3.4 Upper bound $u_2\left(p\right)$

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

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

Now, $R_2(p)$ is given by the sum of (1) the time needed to physically transmit $u_2\left(p\right)$ packets at throughput ψ and (2) the upper bounds on aggregate worst-case execution times for BPA to make scheduling decisions for $u_2\left(p\right)$ packets. Again, given a COTS product that guarantees a worst-case execution time δ_s for BPA at the switch, $R_2(p)$ is given by $R_2\left(p\right)=$

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

3.3.5 Timeliness Feasibility Conditions

The feasibility conditions are therefore:

$$ATB=\sum_{i=1}^{z}\sum_{p_j\in P_i}B_j\left(R\left(s_i,p_j\right)\right)\geqslant ATB_l,$$

where $R\left(s_i,p_j\right)=R_1\left(s_i,p_j\right)+R_2\left(p_j\right).$

Now, to dimension a computer-based system solution to the SRPT problem, an assignment of values to unvalued variables in models of SRPT including number of packets n, packet sizes $b(p_i), i\in[1,n]$, number of sources z, mappings $P_j, j\in[1,z]$, number of arrival sequences k, and arrival densities $\langle(a_\pi(p_i),w_\pi(p_i)),\pi\in[1,k]\rangle, i\in[1,n]$; and properties of SRPT including benefit functions $B_i, i\in[1,n]$ and lower bound on system-wide, aggregate timeliness benefit ATB_l must be made.

The resulting quantified instance of the SRPT problem must then be verified using the feasibility conditions presented here. If a feasible solution exists (to the problem instance), a quantified system that will have values assigned to unvalued variables in the solution including network throughput ψ and BPA's worst-case execution times δ_h and δ_s can be obtained (using the feasibility conditions).

The quantification of the feasibility conditions and the associated verification of system solutions

can be automatically performed by constructing *system dimensioning tools*. The dimensioning tools simply encapsulate the timeliness feasibility conditions and verify whether a quantified instance of the problem satisfies the desired properties under the assumed models.

4. Conclusions, Future Work

In this paper, we present an SE approach for constructing certifiable solutions for soft real-time distributed systems by building upon the TRDF method. Central to the approach is our argument that the coverage of desired system properties including timeliness properties are maximized by favoring asynchronous models and solutions in the design of such systems. Given that many domains can only be modelled with asynchrony assumptions, this is good news.

We demonstrate our SE approach by considering a packet transmission problem with soft timeliness properties. We show how necessary timeliness feasibility conditions can be constructed for this problem. The feasibility conditions that we present constitute the main contribution of the paper. To the best of our knowledge, we are not aware of any such feasibility condition for a soft real-time solution.

Several aspects of the work are currently being studied. The feasibility conditions presented here are necessary, but not sufficient. Constructing necessary and sufficient conditions is a difficult problem, especially for best-effort scheduling algorithms such as BPA as they make scheduling decisions at each scheduling event that are functions of the remaining packet transmission times at the event (unlike EDF).

Further, constructing feasibility conditions for satisfying application-level soft timeliness properties is a challenging problem due to the presence of multiple software layers such as that of the OS and middleware systems and the interactions between the layers. Furthermore, extending the system model for wide-area network systems that include multiple switches and routers and deriving feasibility conditions for such systems broadens the scope of the certification approach. All these issues are currently being studied.

5. Acknowledgements

This work was supported by the U.S. Office of Naval Research under Grant N00014-00-1-0549.

References

[1] C. Baek-Young, S. Sejun, N. Birch, and J. Huang. Probabilistic approach to switched ethernet for real-time control applications. In *Proc. of The IEEE RTCSA*, pages 384 –388, 2000.

[2] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *JACM*, 43(2):225–267, 1996.

[3] K. Chen and P. Muhlethaler. A scheduling algorithm for tasks described by time value function. *Journal of Real-Time Systems*, 10(3):293–312, May 1996.

[4] 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 *Proc. of The WPDRTS*, volume 1586 of *LNCS*, pages 353–362. Springer-Verlag, April 1999.

[5] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *JACM*, 32(2):374–382, April 1985.

[6] J.-F. Hermant and G. Le Lann. Fast asynchronous uniform consensus in real-time distributed systems. *IEEE Trans. on Computers*, 51(8):931–944, August 2002.

[7] E. D. Jensen. Asynchronous decentralized real-time computer systems. In *Real-Time Computing*, Proceedings of the NATO Advanced Study Institute. Springer Verlag, October 1992.

[8] E. D. Jensen and B. Ravindran. Guest editor's introduction to special section on asynchronous real-time distributed systems. *IEEE Trans. on Computers*, 51(8):881–882, August 2002.

[9] G. Le Lann. Proof-based system engineering and embedded systems. In G. Rozenberg and F. Vaandrager, editors, *LNCS*, volume 1494, pages 208–248. Springer-Verlag, October 1998.

[10] G. Le Lann. Asynchrony and real-time dependable computing. In *Proc. of The IEEE WORDS*, January 2003.

[11] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *JACM*, 20(1):46–61, 1973.

[12] N. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[13] M. W. Masters. Challenge problem: System certification for real-time systems that employ dynamic resource management. `http://wpdrts.cs.ohiou.edu/challenge_prob.html`, WPDRTS, 2003.

[14] B. Ravindran, G. Le Lann, and P. Li. Constructing high assurance asynchronous real-time distributed systems: A proof-based system engineering approach. In *Proc. of The IEEE HASE*, pages 89–90, October 2002.

[15] SIXNET. The sixnet industrial ethernet switch. `http://www.sixnetio.com/`.

[16] S. Varadarajan and T. Chiueh. Ethereal: A host-transparent real-time fast ethernet switch. In *Proceedings of The IEEE ICNP*, October 1998.

[17] J. Wang. Soft real-time switched ethernet: Best-effort packet scheduling algorithm, implementation, and feasibility analysis. Master's thesis, Virginia Tech, September 2002.

[18] L. R. Welch, B. Ravindran, B. Shirazi, and C. Bruggeman. Specification and modeling of dynamic, distributed real-time systems. In *Proc. of The IEEE RTSS*, pages 72–81, December 1998.