Evaluation of Wireless Soft Real-Time Protocols*

Michael J. Markowski† Adarshpal S. Sethi
University of Delaware
Department of Computer and Information Sciences
Newark, Delaware, USA

Abstract

Communication between current military real-time systems and future interconnection of general purpose, embedded real-time systems will often require wireless communications. However, there has been little work undertaken to offer support for real-time applications on wireless networks. We present and evaluate three protocols; variations of two published protocols by Paterakis and Gallager as well as our new one, the Sliding Partition (SP) collision resolution algorithm (CRA). In a real-time setting, the modified Gallager CRA consistently performs worst of the three we consider. We observe that when the deadline range is small, the Sliding Partition CRA performs best. When the deadline range is large, however, the Paterakis CRA performs slightly better than the SP CRA. Both analytic and simulation results are obtained to study the maximum input traffic rates that can be sustained for various laxities, delay bounds, and message loss rates.

1. Introduction

Applications directly interacting with the world often must respond to changes within some predetermined amount of time. Such systems are classified as either *hard* or *soft* real-time systems. A hard real-time system requires that all deadlines are always met because of the importance of the data, e.g., aeronautics and nuclear power control. Soft real-time systems can afford some deadlines to slip.

Data networks supporting real-time systems, whether hard or soft, are usually implemented as nodes interconnected with cable. Using copper or fiber cable is desirable because of their low error rates and high speeds. However, by the nature of their functional requirements, real-time systems are often embedded and autonomous or semi-autonomous. When the need arises to interconnect these for high level coordination, it is not feasible, or even possible in many cases, to use cable. Wireless communication is the only alternative.

As an example, in military environments and especially on the battlefield, data communications have a range of priority levels with messages of each level having some independently derived deadline. Additionally, all nodes are mobile, leaving wireless as the only option.

In these situations, the data shared is often high level for coordination of distinct but cooperating systems rather than instrumental to the interconnection of nodes to build a single real-time system. Occasionally missed deadlines are not detrimental to overall functionality. Thus, this arrangement can be viewed as a loosely coupled soft real-time system. This is convenient because regardless of system requirements, the noisy, error-prone wireless environment is not suitable for implementing a hard real-time system.

Also noteworthy is that wireless systems in general offer lower bandwidth than cable based systems. As a result, packet transmission times are longer. For slotted systems, as we will consider, this means slots are longer as well. This affects transmission scheduling in the sense that while a packet deadline may be far off in terms of time, it might only represent a small number of slots. We take this into account in the models described below.

Ultimately, it is the lowest layer of the protocol stack which provides timely access to the communications medium. We consider several media access protocols for soft real-time systems implemented on a slotted radio channel with binary feedback and assume feedback is provided out of band, e.g., on another frequency.

Because each application has its own requirements for a minimum acceptable level of protocol performance, our interest is, given these requirements, to determine the maximum traffic rate the system can handle. Knowing the performance for a range of traffic rates will allow the protocol to be tailored for use by a variety of soft real-time applications without designing to the lowest common denominator, i.e.,

*This work was supported in part by the U.S. Department of the Army, Army Research Laboratory under Cooperative Agreement DAAL01-96-2-0002 Federated Laboratory ATIRP Consortium.

†M. J. Markowski is with the US Army Research Laboratory, APG, MD, USA.

the safest, lowest loss, and lowest throughput system.

The obvious disadvantage to using existing, general purpose random access schemes for real-time communication is that the worst case channel access time is unbounded due to packet collisions. Typical random access protocols for both wired and wireless LANs react to collisions by assigning a random waiting time to a collided packet. The problem is compounded by the fact that general algorithms do not take packet transmission deadlines into account.

Approaches to limiting or removing these shortcomings for time constrained communication on random access channels have concentrated on two methods: the use of virtual time clocks, and window splitting techniques. Virtual time clocks were first proposed by Molle and Kleinrock [6] and based on message arrival time. This has the advantage of making transmission of queued messages fairer. The method was adapted by Ramamritham and Zhao [9] to take into account various time related properties of a packet for soft real-time systems and shown via simulation to work better than protocols not designed for real-time use. Subsequently, Zhao et al. [10] proposed a window splitting protocol which always performed in simulation at least as well as the virtual time protocols and often better. Less complex window splitting algorithms are presented for both hard and soft real-time systems by Arvind [1] where protocol operations are simulated and some worst case performance analysis is also presented.

In the area of queueing analysis, Georgiadis et al. [4] develop a straightforward, general method of delay analysis using regenerative properties of random multiple access algorithms. This is elaborated on by Paterakis et al. [7] where they present and analyze a simple protocol appropriate for limited types of soft real-time systems. We extend Paterakis' analytic technique in this paper for a more detailed and complex analysis of three protocols suitable for use in wireless soft real-time environments. While we compare published protocols to one we develop, note that even the published ones require modifications, as will be described, to operate in general real-time environments.

The first analyzed is a variation of that introduced by Paterakis. His analysis, though, makes the assumption that all packets are assigned the same deadline upon arrival at the transmission queue. This is unrealistic for general real-time systems, so we make the change of assigning a deadline from a distribution. We do the same for the other two protocols. The next considered is a variation of the famous FCFS Gallager algorithm [3], [2] modified by us with binary feedback, strict delay bounds, and laxity ranking. This is similar to one studied by Arvind [1] but his does not incorporate a delay bound. Finally, we present a new protocol, the Sliding Partition (SP) CRA, that takes deadlines into account yet remains algorithmically simple.

2. Algorithm

In these protocols, a packet has a single property of interest, its laxity. Laxity is the maximum amount of time that can elapse prior to transmission, after which the packet will not reach its destination on time (we only consider single-hop radio channels in this analysis). Once a laxity is assigned to a packet, at each slot boundary it is decremented and the packet discarded should the laxity reach zero. The channel is accessed in a slotted manner by one slot long packets with binary (collision/non-collision) feedback. Collisions are resolved using algorithms described below.

When two or more nodes transmit at once and collide, the collision resolution algorithm (CRA) commences and only nodes involved in the collision may contend for the channel. Only after all collided packets have been either successfully transmitted or else discarded due to missed deadlines, can other nodes again contend for the channel. Note that packets are never delivered late. They are either transmitted in a timely manner or dropped. At this low level, there are two competing viewpoints: the system as a whole would like short collision resolution intervals (CRIs), while individual packets want to be transmitted regardless of delay as long their deadlines are met. We balance these views by outright dropping of late packets. A higher protocol layer should decide if it is worth retransmitting late or not, especially in the case of multiple packets making up a single higher layer message.

Simple notation is used to describe each protocol. Letting f_t denote the feedback corresponding to slot t, $f_t = c$ if there was a collision in slot t, otherwise $f_t = nc$ if there was no collision in that slot. A system parameter T limits the length of a CRI. If the CRI is ongoing for T slots, all involved packets are dropped and the CRA is reset. During a CRI, any packet whose laxity drops to 0 is discarded.

2.1. Paterakis CRA

The protocol put forth by Paterakis et al. [7] is elegantly simple. Each node has a counter whose value may be either 1 or 2, but can only transmit when the counter value is 1. Upon collision, a collision resolution interval (CRI) begins. During the CRI, after each collision all colliding nodes flip coins. Based on the outcome, counters are assigned a new value of 1 on, say, heads, and 2 on tails. After each noncollision slot, all nodes in the CRI set the counter to 1 and transmit. A CRI ends when there are two consecutive noncollision slots.[1]

A node's counter at time t is denoted by r_t. With this notation, we can more rigorously describe the algorithm.

[1] Most window splitting techniques split based on some parameter such as arrival time, laxity, etc. When two packets *tie*, i.e., have the same value for that parameter, randomness is introduced to artificially separate the values. Paterakis simply uses this technique at the outset.

1. A packet is successfully transmitted if and only if $r_t = 1$ and $f_t = nc$.

2. The transitions are:

   - If $f_{t-1} = nc$ and $r_{t-1} = 2$, then $r_t = 1$.
   - If $f_{t-1} = c$ and $r_{t-1} = 2$, then $r_t = 2$.
   - If $f_{t-1} = c$ and $r_{t-1} = 1$, then $r_t = 1$ with probability 0.5 and $r_t = 2$ with probability 0.5.

2.2. Modified Gallager CRA

During a CRI in our modified version of the Gallager CRA, an arrival time based window of initial length Δ slots is used. If a collision occurs, the packets are ordered, left to right, from lowest laxity to highest, in a laxity window. Only packets whose laxities fall within the laxity window may transmit. If another collision occurs, the laxity window is split in half, and the CRA recurses first on the left half and then on the right. The CRA behaves similarly to the classical FCFS splitting algorithm [2] with a few differences: packets in a window following a collision are ordered by laxity rather than by queue arrival time, and feedback is binary rather than ternary. We further impose a bound T which is the maximum number of slots a CRI may comprise. If T slot times are reached, all packets involved in the CRI are dropped, and the algorithm is reset. This CRA is analyzed in detail in [5] for constant initial laxities.

Because of the laxity ordering, the first successful transmission during a CRI will be the lowest laxity packet, the second will be the second lowest, and so on, guaranteeing a laxity ordered transmission schedule appropriate in a real-time setting.

We denote the position of the left edge of the window at time t as x_t and its length, in slots, as a_t. Finally, h_t can take on the value of L or R indicating whether the collision resolution algorithm is in the left or right subwindow. By convention, the algorithm is initially in the right half. At time $t = 0$ the system is empty and that slot has a feedback value of $f_0 = nc$. Subsequently,

1. If collision resolution is not in progress, then a node with a packet which arrived in slot $t-1$ may transmit it in the current slot t.

2. If collision resolution is in progress:

   - If $f_{t-1} = nc$ and $h_{t-1} = L$, then $x_t = x_{t-1} + a_{t-1}, a_t = a_{t-1}$, and $h_t = R$.
   - If $f_{t-1} = nc$ and $h_{t-1} = R$, then $x_t = x_{t-1} + a_{t-1}, a_t = \min(2a_{t-1}, \min(d, \Delta))$, and $h_t = R$. Lag d is discussed shortly in the Analysis section.
   - If $f_{t-1} = c$, then $x_t = x_{t-1}, a_t = \frac{1}{2}a_{t-1}$, and $h_t = L$.

2.3. Sliding Partition CRA

The Sliding Partition (SP) protocol has some properties of each of the above CRAs. During the CRI, packets are ordered by laxity, lowest to highest, from left to right. Only packets whose laxities fall within the current laxity window can transmit. However, after a collision the window is not recursively split. The left window is shrunk by half with the other portion returned to the right window. Until a noncollision occurs, the left window is successively split in two while the right window successively expands. After a noncollision, all packets in the right window transmit. When Δ is appropriately chosen, we expect most collisions to be of multiplicity two. As in the Paterakis CRA but unlike the Gallager CRA, a CRI end is signaled by two consecutive noncollision slots or when the time bound is reached.

We use the same notation as with the Gallager CRA. By convention, the algorithm is initially in the right half.

1. If collision resolution is not in progress, then a node with a packet which arrived in slot $t-1$ may transmit it in the current slot t.

2. If collision resolution is in progress:

   - If $f_{t-1} = nc$ and $h_{t-1} = L$, then $x_t = x_{t-1} + a_{t-1}, a_t = \min(\Delta, d) - a_{t-1}$, and $h_t = R$.
   - If $f_{t-1} = nc$ and $h_{t-1} = R$, then CRI terminates.
   - If $f_{t-1} = c$, then $x_t = x_{t-1}, a_t = \frac{1}{2}a_{t-1}$, and $h_t = L$.

3. Analysis

Regardless of which CRA is used, at any time instant during a CRI, an initial collision that happened some while ago is being resolved. This means that there is a lag of d units between "now" and the period of time currently being examined by the CRA. The lag d is important in a real-time environment because the lag induced by a CRI means that, for the subsequent CRI, only $T - d$ slots remain before the CRA is reset dropping all untransmitted packets. In addition, the width u of a window plays a crucial role in the length of a CRI. Short initial windows waste time because many will be empty, while long ones potentially encompass several packets leading to still more collisions.

Figure 1 illustrates some of these variables and how they are related. In the figure, the current moment of the illustration is time t and because of some previous collision, the current lag is d slots. The previous CRI completed at time t_1. Anything with a smaller laxity cannot be transmitted before its deadline and so is rejected, or discarded, outright. Similarly, to successfully transmit all packets in the current window, the CRI must complete within the next $t' = T - d$

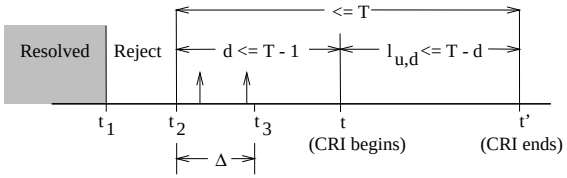

Figure 1. Relation between some variables in algorithm.

slots. The variable $l_{u,d}$ is defined in detail in the Appendix but is simply the length of the CRI which begins at time t and ends at time t'.

In the subsequent paragraphs we present the technique for analysis of time bounded CRAs with packets having variable laxity assignments. See Paterakis et al. [7] for the presentation of the original, fixed laxity assignment technique used to analyze their protocol, or [5] for our first extensions to analyze the more involved fixed initial laxity, time bounded, and laxity ordered Gallager CRA.

3.1. Notation

In this analysis, we make the assumption that initial packet laxities are drawn from a uniform distribution between 2 and T, inclusive. Two is the smallest since there is always a one slot feedback delay plus the one slot to transmit. Considering a Markov chain whose state space is the set of integers representing possible *lag* values, we use lags of one slot as regeneration points. Subsequent analysis is based on one cycle of the regenerative stochastic process.

Two considerations vital to successful transmission of a packet are the current lag d and the number of slots u to be examined. Appropriately, the following variables are subscripted with this information.

- $n_{u,d}$: Number of packets in window u with lag d which are successfully transmitted during the cycle.
- $z_{u,d}$: Sum of delays after transmission of packets in $n_{u,d}$.
- $l_{u,d}$: Number of slots needed to examine u slots when current lag is d slots.

In order to compute the expected values for the fraction of traffic transmitted within the laxity bound and for the delay experienced by successful transmissions, the variables above must be incorporated into expressions which reflect not just what occurs for given window/lag values but for a full regenerative cycle. The subscript d on the following variables represents what happens between the current lag of d slots and when the lag next returns to 1. When $d \neq 1$, the algorithm is at some intermediate point in a regeneration cycle.

- h_d : Given current lag of d, the number of slots to return to lag of 1.
- w_d : Cumulative delay of packets transmitted during the h_d slots.
- α_d : Number of packets transmitted during the h_d slots.

Also, given that some event occurs during a CRI of length l, we must multiply that event by the probability that the CRI is actually of that length. We denote the form of the final type of variable as:

- $P(l \mid u, d)$: Given that u slots are to be examined and the lag is presently d slots, the probability that the CRI will be l slots long.

Finally, to compute probabilities of a given number of packets arriving in some interval u, we assume a Poisson arrival process.

Defining $Z = E\{\alpha_1\}$ as the number of packets successfully transmitted in a cycle, and $H = E\{h_1\}$ as the expected length of a cycle, then Z/H is the traffic rate of successful packets. This rate must be less than or equal to the original rate λ. Therefore, the fraction, ρ, of generated packets which are successfully transmitted is

$$\rho = \frac{Z/H}{\lambda}.$$

If we next define $W = E\{w_1\}$ as the cumulative expected delay of all packets in the cycle, then the expected per packet delay is simply

$$D = \frac{W}{Z}.$$

Taking expectations of expressions whose derivations are omitted here, the following is obtained:

$$X_d = \begin{cases} E\{\theta_{d,d}\} + \sum_{m=2}^{T-\lceil d \rceil} & 1 \leq d \leq \Delta, \\ \quad X_m P(m \mid d, d, \ldots), & \\ E\{\theta_{\Delta,d}\} + \sum_{m=1}^{T-\lceil d \rceil} & \Delta < d < T. \\ \quad X_{d-\Delta+m} P(m \mid \Delta, d, \ldots), & \end{cases}$$

where $X_d = E\{x_d\}$ and x_d is one of the random variables h_d, w_d, or α_d; d is the current lag in slots; and Δ is the maximum window width. The ellipses in the probability terms indicate that for different protocols, more parameters may be required. These are elaborated on in the Appendix. Note that the system of equations is finite due to the bound T. The recursions for $E\{\theta\}$ and $P(l \mid \ldots)$ are also given in the Appendix.

Compared to fixed laxity assignments, it is more complex to analyze the variable laxity case because, during each slot of a CRI involving k packets, there is a possibility of one or more packets expiring due to their variable initial deadlines.

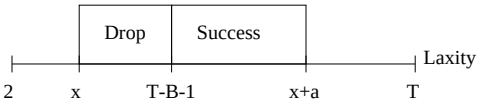

Figure 2. Relation between some variables determining drop probability.

Incorporation of that requires expressions for the probable deadline values assigned. For instance, as a CRI proceeds, it becomes more and more likely that some of the collided packets will expire before transmission. Consider either the bounded Gallager or Sliding Partition CRAs. If the current laxity window's left edge is at laxity x and is of length a, the range $[x, x+a]$ is a subset of the maximum range $[2, T]$. Furthermore, if, due to a current or previous CRI, only B slots remain before the CRA resets, we must determine the probability that a given packet will be dropped at the end of the current slot. Looking at the interesting case of $T-B-1 \in [x, x+a]$, i.e., only some packets in the current laxity window can survive, Figure 2 shows the relationships described. Packets whose initial laxities r_0 were such that $r_0 \in [T-B-1, x+a]$ are not dropped. Packets with $r_0 \in [x, T-B-1)$, however, are dropped at the end of the current slot, so simply because a packet is in the current laxity window does not mean it has a transmission probability of 1.

4. Evaluation

In real-time systems it is important that the application's requirements be met by all layers of the protocol stack. As mentioned, at the media access layer of a soft real-time system, this translates to a guarantee that some minimum fraction of traffic is in fact transmitted by its deadline. Calling this design parameter e_1, for the fraction of packets transmitted successfully, results are presented below.

While it is the application which drives the choices for maximum CRI length T and e_1, the initial time window width Δ is a basic design parameter of the protocol itself. That is, it is chosen from the optimization of the analysis of the previous section. The window can range between zero and $T-1$. For this range we would like to find the maximum system traffic rate, λ^*, which allows successful transmission of at least e_1 of the traffic. Because of the complexity of the expressions developed, analytic optimization is difficult. For given values of Δ, however, the problem is simple to solve numerically and, as pointed out by Paterakis et al. [7], reduces to:

$$\lambda^*_{T,e_1} = \sup(\lambda : \rho_T(\Delta, \lambda) \geq e_1).$$

We present results only for $\Delta = 2.5$ though data has been generated for various Δ values which, through observation, encompass the maximum throughputs. We also conducted simulation studies of each protocol using Opnet, a network simulation package. The simulations model the systems with infinite user populations, thus offering more conservative performance results than the finite user case [8]. Each simulation was run with 95% confidence intervals that the fraction of successful transmissions ρ was within ± 0.001 of the steady state value. Input traffic rates, in units of packets/slot, ranged from 0.050 to 0.600 in increments of 0.01. When window size $\Delta = 2.5$, Figure 7 shows overlaid graphs of analytic and simulation results for $\lambda^*_{20,0.9}$. Because of the close correlation, other graphs show either only analytic or only simulation results for the sake of readability. Furthermore, due to the exponential nature of the equations, it quickly becomes exceedingly difficult to obtain analytic results as CRI bound T increases.

In Figure 3, three curves are graphed showing $\lambda^*_{5,0.9}$. For a given offered load, the CRAs can be ranked, best to worst, as SP, Paterakis, and Gallager. The corresponding delays for these curves can be seen in Figure 4.

However, as the CRI bound increases to approximately $T = 8$, the three CRAs perform similarly. This is seen for $T = 10$ in Figure 5 and corresponding delay curves in Figure 6. As T increases still more, the Gallager CRA performs progressively worse as Figures 8 and especially 9 show. Interestingly, at the same time, the Paterakis CRA begins to slightly outperform the SP CRA. Once the laxity range is greater than eight slots or so, it appears that the window splitting is time consuming enough that the Paterakis CRA does better because, due to the random splitting, it tends to more quickly separate packets. This is impressive considering that the Paterakis CRA does not take deadlines into account. However, because deadlines are not considered in the Paterakis CRA, the performance gap between it and the SP CRA is not seen to widen as T further increases. The advantage gained by the randomness of the Paterakis CRA is equally offset by the advantage gained using deadlines in the SP CRA.

5. Conclusions

To support a soft real-time system, a network must incorporate the concept of deadlines in all layers of the protocol stack. This has been traditionally difficult to do in random access protocols, where packet delays may be potentially unbounded due to collisions. By assuming that a message is dropped whenever its delay exceeds its initial laxity, and by using a minimum rate of successful transmission, we have shown that window-splitting protocols can be modified to work successfully in these environments. We have presented an analytic model for such protocols which can be used to determine proper operating parameters for specified quality-of-service constraints.

Based on the results graphed for these specific protocols, when packets have initial laxities encompassing a wide range, the Paterakis CRA would be the best choice because it is the simplest to implement. While it does not take deadlines into account, its simplicity allows it to outperform more complex real-time CRAs like the Gallager CRA and, in some cases, the SP CRA. In systems, however, where a small range of laxities are expected, the SP CRA performs best. The Gallager CRA, aside from the fact that it does not perform well, cannot be considered for implementation because of its unrealistic assumptions, i.e., that nodes never lose synchronicity and that the feedback channel is error free. In contrast, the Paterakis and Sliding Partition CRAs operate in ways allowing practical implementation in a wireless environment. Two consecutive noncollision slots signal that a CRI is not in progress. If nodes lose synchronization due to noisy feedback, or if they have not been monitoring the channel because of temporary loss of connectivity, they simply wait until observing two noncollision slots.

It would be interesting to modify these protocols to operate in a free access manner. That is, newly arriving packets during a CRI could be transmitted, presumably allowing still more low laxity packets to be successfully transmitted. Also, the CRAs currently drop all packets in a window once the left edge of the window reaches a laxity of one. This is not necessary, and perhaps a performance improvement would be seen if the CRI were modified to accommodate this. Also interesting would be a hybrid version of the SP algorithm which, when the laxity window is less than one unit wide, would revert to the Paterakis CRA until the CRI completes. Especially interesting to study, though, is the initial laxity window for the SP CRA. Since it clearly outperforms the other CRAs with laxity ranges of eight or less, perhaps if the initial window were already split to that size, it would perform still better. After a study of these variations, the most promising CRA will be implemented for use in military battle environments.

A. SP Recursions

To calculate $E\{\theta\}$ and $P(\ldots)$, the expected value, given that k packets are within the current window, B slots is the maximum length of the CRI, x is the left edge of the window, and a its length, is

$$E\{\zeta_{u,d}\} = \sum_{k=0}^{\infty} E\{\zeta_{u,d} \mid k, T-d, x, a\} e^{-\lambda u} \frac{(\lambda u)^k}{k!}$$

with similar notation for $P(\ldots)$, and where ζ is one of n, z, or l. Note that the Paterakis recursions have no need for x and a since a laxity window is not used. Because the algorithms are independent of the window width u and current lag d, they are dropped from subsequent notation. The following denotations are used:

$$L_{k,B,x,a} = E\{l_{u,d} \mid k, B, x, a\},$$

and similarly for $N_{k,B,x,a}$ and $Z_{k,B,x,a}$.

A.1. Blocking Probability

The probability of a single packet being dropped can be considered a Bernoulli trial. For a multiplicity k collision, trials for each of the k packets yield a binomial distribution. The variable p_1 is the probability of dropping a single packet.

$$\text{bin}(k, b \mid T, B) = \binom{k}{b} p_1^b (1-p_1)^{k-b}.$$

In the following sections, it can be seen that most terms take into account the fact that at each slot, 0 or more packets can be dropped. For each summation involving the subscript b, for "blocked packet," that result is multiplied by the probability, above, of that number of packets being blocked, or dropped.

With maximum CRI length T and at most B slots remaining in CRI, the probability that a given packet's initial laxity r_0 will cause it to be dropped at the end of the current slot is

$$p_1 = P(r_0 <= T - B - 1) = \frac{T-B}{T-1}.$$

Notice that there is no dependency on the laxity window.

A.2. Probable Length of CRI

$$\begin{aligned} P(l \mid k, B, x, a) &= 2^{-k} \textstyle\sum_{j=1}^{l-1} \binom{k}{i} \\ &\cdot \textstyle\sum_{b=0}^{i} Q(l-j \mid i-b, B-1, x, a/2) \\ &\cdot \text{bin}(i, b \mid T, B) \\ &\cdot \textstyle\sum_{b=0}^{k-i} P(j-1 \mid k-i-b, B-1-(l-j), \\ &\quad x + a2^{-(l-j)} a - a2^{-(l-j)}) \\ &\cdot \text{bin}(k-i, b \mid T, B-m) \end{aligned}$$

for $k > 1$ and with initial conditions $P(\ldots) = 0$, for $B = 0$; $P(\ldots) = 1$, for $k \leq 1, B \geq 1$; $P(\ldots) = 1$, for $B = 1$; $P(\ldots) = 0$, for $k \leq 1, B > 1$.

$$\begin{aligned} Q(l \mid k, B, x, a) = 2^{-k} &\textstyle\sum_{j=1}^{l-1} \binom{k}{i} \textstyle\sum_{b=0}^{i} Q(l-1 \mid i-b, \\ &B-1, x, a/2) \text{bin}(i, b \mid T, B) \end{aligned}$$

for $k > 1$ with the same initial conditions as for $P(\ldots)$.

A.3. Expected Length of CRI

$$\begin{aligned} L_{k,B,x,a} &= 1 + \textstyle\sum_{b=0}^{i} M_{i-b,B-1,x,a/2} \\ &\cdot \text{bin}(i, b \mid T, B) \\ &+ \textstyle\sum_{m=1}^{B-1} \sum_{b=0}^{k-i} P(m \mid i, B-1, x, a/2) \\ &\cdot L_{k-i-b,B-1-m,x+a2^{-m},a-a^{-2m}} \\ &\cdot \text{bin}(k-i, b \mid T, B-m) \end{aligned}$$

with probability $\binom{k}{i}2^{-k}$, for $k > 1$, and initial conditions $L_{k,B,x,a} = 1$, for $k = 0, 1$.

$$M_{k,B,x,a} = 1 + \sum_{b=0}^{i} M_{i-b,B-1,x,a/2} \cdot \text{bin}(i, b \mid T, B)$$

with probability $\binom{k}{i}2^{-k}$, for $k > 1$, and initial conditions $L_{k,0} = M_{k,0} = 0$ for all k, and $L_{0,B} = L_{1,B} = L_{k,1} = M_{0,B} = M_{1,B} = M_{k,1} = 1$ for $B \geq 1$.

A.4. Packets Transmitted During CRI

$$N_{k,B,x,a} = \sum_{b=0}^{i} J_{i-b,B-1,x,a/2}\text{bin}(i, b \mid T, B) + \sum_{m=1}^{B-1}\sum_{b=0}^{k-i} P(m \mid i, B-1, x, a/2) \cdot N_{k-i-b,B-1-m,x+a2^{-m},a-a^{-2m}} \cdot \text{bin}(k-i, b \mid T, B-m)$$

with probability $\binom{k}{i}2^{-k}$, for $k > 1$, and initial conditions $N_{k,B,x,a} = k$ for k=0,1.

$$J_{k,B,x,a} = \sum_{b=0}^{i} J_{i-b,B-1,x,a/2}\text{bin}(i, b \mid T, B)$$

with probability $\binom{k}{i}2^{-k}$, for $k > 1$, and initial conditions $N_{k,0} = J_{k,0} = 0$ for all k, and $N_{1,B} = J_{1,B} = 1$ for $B \geq 1$, and $N_{0,B} = N_{k,1} = J_{0,B} = J_{k,0} = 1$ for $B \geq 1$, $k > 1$.

A.5. Cumulative Delay Experienced During CRI

$$Z_{k,B,x,a} = \sum_{b=0}^{i}(J_{i-b,B-1,x,a/2} + Y_{i-b,B-1,x,a/2}) \cdot \text{bin}(i, b \mid T, B) + \sum_{m=1}^{B-1}\sum_{b=0}^{k-i} P(m \mid i, B-1, x, a/2) \cdot [(m+1)N_{k-i-b,B-1-m,x+a/2,a/2} + Z_{k-i-b,B-1-m,x+a/2,a/2}] \cdot \text{bin}(k-i, b \mid T, B-m),$$

with probability $\binom{k}{i}2^{-k}$, for $k > 1$, and initial conditions $Z_{k,B,x,a} = k$ for k=0,1.

$$Y_{k,B,x,a} = \sum_{b=0}^{i}(J_{i-b,B-1,x,a/2} + Y_{i-b,B-1,x,a/2}) \text{bin}(i, b \mid T, B)$$

with probability $\binom{k}{i}2^{-k}$, for $k > 1$, and initial conditions $Y_{k,B,x,a} = k$ for k=0,1.

A.6. Paterakis and Gallager Recursions

The expected changes are made to the modified Paterakis and Gallager expressions for L, N and Z. Due to space considerations, they are not presented.

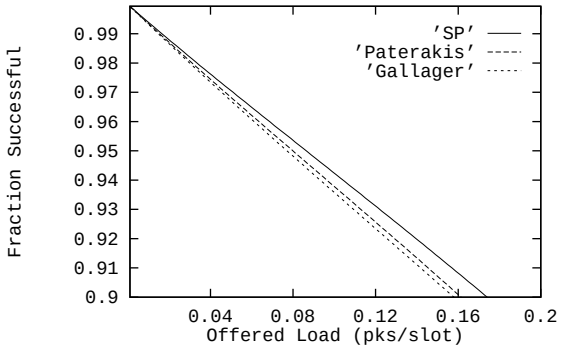

Figure 3. Analytic results for $\lambda^*_{5,0.9}$.

References

[1] K. Arvind. *Protocols for Distributed Real-Time Systems*. PhD thesis, University of Massachusetts at Amherst, 1991.

[2] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, second edition, 1992.

[3] R. G. Gallager. Conflict resolution in random access broadcast networks. *Proceedings of the AFOSR Workshop in Communications Theory Applications*, pages 74–76, 1978.

[4] L. Georgiadis, L. F. Merakos, and P. Papantoni-Kazakos. A method for the delay analysis of random multiple-access algorithms whose delay process is regenerative. *IEEE Journal on Selected Areas in Communications*, SAC-5(6):1051–1062, July 1987.

[5] M. J. Markowski and A. S. Sethi. Analysis of a soft real-time protocol. Technical Report 96-02, University of Delaware, November 1995.

[6] M. L. Molle and L. Kleinrock. Virtual time CSMA: Why two clocks are better than one. *IEEE Transactions on Communications*, COM-33(9):919–933, September 1985.

[7] M. Paterakis, L. Georgiadis, and P. Papantoni-Kazakos. Full sensing window random-access algorithm for messages with strict delay constraints. *Algorithmica*, 4:318–328, 1989.

[8] M. Paterakis, L. Georgiadis, and P. Papatoni-Kazakos. On the relation between the finite and the infinite population models for a class of raa's. *IEEE Transactions on Communications*, COM-35(11):1239–1240, November 1987.

[9] K. Ramamritham and W. Zhao. Virtual time CSMA protocols for hard real-time communication. *IEEE Transactions on Software Engineering*, 13(8):938–952, 1987.

[10] W. Zhao, J. A. Stankovic, and K. Ramamritham. A window protocol for transmission of time-constrained messages. *IEEE Transactions on Computers*, 39(9):1186–1203, September 1990.

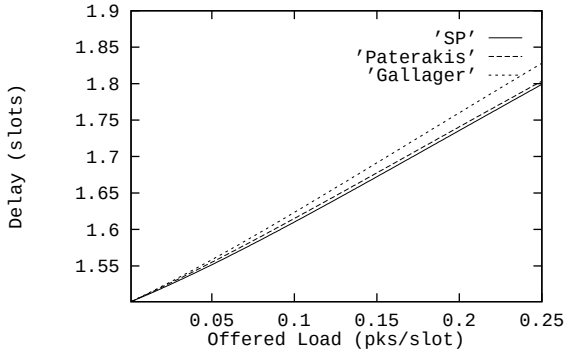

Figure 4. Analytic results for $\lambda^*_{5,0.9}$.

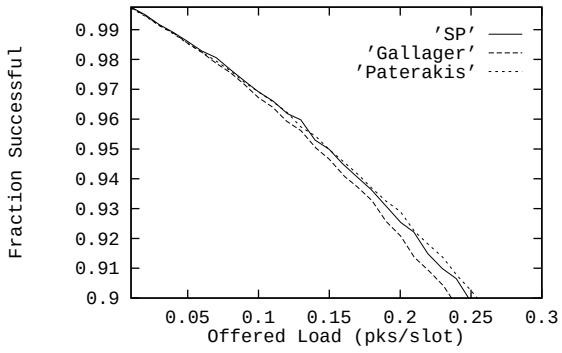

Figure 5. Simulation results for $\lambda^*_{10,0.9}$.

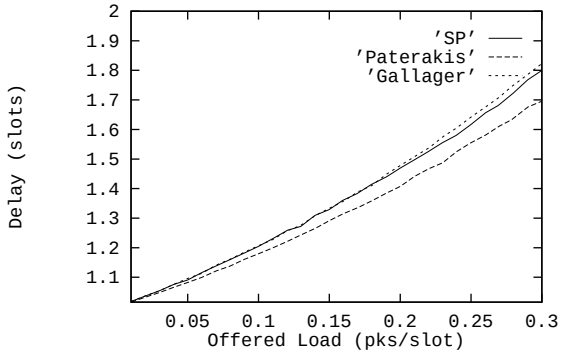

Figure 6. Simulation results of delay for $\lambda^*_{10,0.9}$.

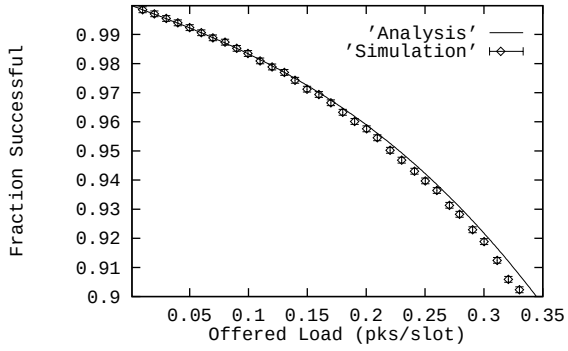

Figure 7. Simulation and analytic results for $\lambda^*_{20,0.9}$ for the Paterakis CRA.

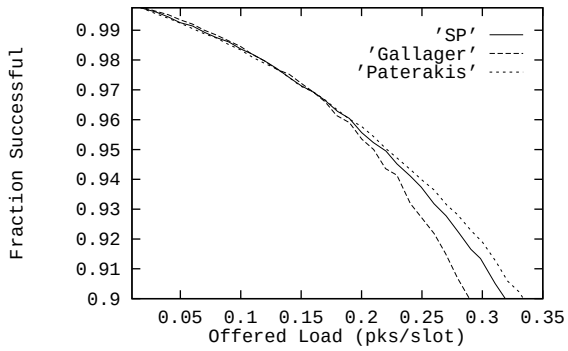

Figure 8. Simulation results for $\lambda^*_{20,0.9}$.

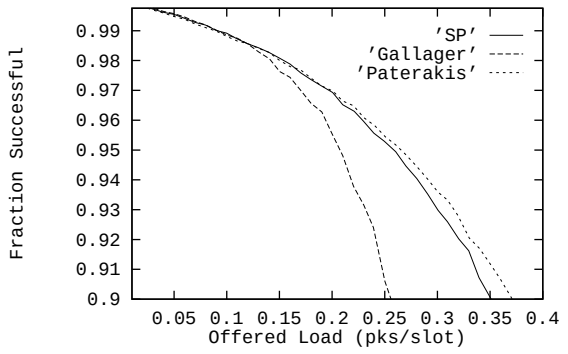

Figure 9. Simulation results for $\lambda^*_{30,0.9}$.