To appear in Integrated Computer-Aided Engineering, ©John Wiley & Sons, Inc., December 1997.

Blocked and Free Access Real-Time Splitting Protocols*

Michael J. Markowski† Adarshpal S. Sethi

University of Delaware

Department of Computer and Information Sciences

Newark, Delaware, USA

July 23, 1997

Abstract

Addressing the challenge of packet transmission in a wireless soft real-time system, we present five splitting protocols that take packet deadlines into account. With them, connectionless service with real-time QoS guarantees at the MAC layer can be offered. Three protocols are blocked access and two are free access algorithms. Mathematical models are developed and results compared with simulations. As is the case with non real-time splitting algorithms, the blocked access versions offer higher success rates than the free access versions. We further show that of the two best performing blocked access protocols, under moderate to heavy loads, the Sliding Partition CRA outperforms the Two Cell CRA.

1 Introduction

A group of nodes working in concert to complete some set of tasks by specified deadlines is a *real-time system.* Such systems can be broadly subcategorized into *hard* and *soft* real-time systems. Hard real-time systems are those whose data is so important that all deadlines must be met to avoid catastrophic physical or financial failures. Examples might include aeronautic systems, power plants, or weapons fire control systems. Soft real-time systems, however, can safely afford some amount of lateness or loss of data. Some examples are personal audio or video transmissions, radar based object tracking, or remotely monitored meteorological updates. The nature of the data transferred can also be further categorized. *Streaming* data, like the personal audio and video mentioned above, require longer term, connection-oriented service. In contrast, *bursty* data transfer between nodes in a distributed real-time system tend to be short, connectionless and similar in nature to what is seen in traditional queueing networks. For instance a sensing node might send a new set

*Prepared through collaborative participation in the Advanced Telecommunications/Information Distribution Research Program (ATIRP) Consortium sponsored by the U.S. Army Research Laboratory under the Federated Laboratory Program, Cooperative Agreement DAAL01-96-2-0002.

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

of readings to another node. After processing, it in turn might alert yet another node to some new status. The new features of the traffic, though, are its real-time properties: the time constrained nature of the data being communicated.

We consider here the specific problem of a soft real-time system implemented on a wireless network offering connectionless service. We assume that the network will be used to support the somewhat bursty exchange of data between nodes cooperating to jointly implement a real-time system on a single wireless network. Until recently, there has been little need for the support of real-time data over wireless nets because the high probability of link errors makes radio unattractive for transport of time sensitive data. Hence, most work on wireless communication has focused on long term resource allocation for telephone calls in cellular networks, or, when the case of data transport is considered, the use of protocol tunneling across cellular systems. In both cases, though, the transport of data is connection oriented.

However, in certain environments, *e.g.*, military battlefield communications or coordination of search and rescue vehicles, data is short and bursty, and there is no option but to use wireless channels for real-time communications. When real-time transport of data is needed in random access networks, quality of service guarantees must take into account that not all of the offered load can be transmitted. That is, the higher protocol layers, especially the application layer, must be aware of the capabilities and limitations of the MAC (media access control) layer. Conversely, the MAC layer can make the most of the available resources by providing just the level of quality needed by the application and not more, thus conserving resources for use by other applications or nodes.

Presently, it is often the case that application layer real-time requests are responded to on random access nets by making do with the services provided, ultimately, by the MAC layer and filtering out "late comers." For some applications, this approach is acceptable; the performance of existing protocols provides the quality of service needed. However, on wireless links, usually offering less bandwidth than wire links, to meet functional requirements of system designs, it is necessary to provide media access that better meets the time constraints of the data. It is not new hardware technology that is needed, but new channel access techniques.

2 Background

The most popular random access MAC techniques are Aloha (Abramson 1970, Abramson 1985) and Ethernet (Metcalfe & Boggs 1976) or variants of them. Their shortcoming in the real-time environment is clear: collisions result in a random transmission order of involved packets. Since this offers potentially unbounded access times at worst, and transmission scheduling ignoring time constraints at best, it is undesirable in real-time settings. Aside from straightforward techniques such as priority classes or transmission scheduling based on *a priori* knowledge, approaches to limiting this shortcoming for time constrained communication on random access channels include the use of virtual time clocks and splitting techniques.

In CSMA-CD systems, virtual time clocks were first considered by Molle & Kleinrock (1985) and based on message arrival time. The technique has the advantage of making transmission of queued messages fairer, though it provides no consideration for deadlines. The method was adapted by Ramamritham & Zhao (1987) 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, Stankovic & Ramamritham (1990) proposed a splitting protocol that always performed in simulation at least as well as the virtual time protocols and often better. Consequently, we pursue splitting protocols, though for use on wireless nets rather than CSMA-CD. It should also be noted that while CSMA-CD is not a possibility for wireless LANs since an antenna cannot be used for simultaneous transmission and reception, in this situation, research in CSMA-CD can be considered for use in wireless systems and vice-versa. This is because the CSMA-CD splitting protocols make use of both collision detection and CSMA only in ways peripheral to the splitting algorithm. While Bertsekas & Gallager (1992) show that splitting in conjunction with CSMA-CD offers no improvement over slotted Aloha, the splitting methods we present may, in fact, be attractive for implementation in some CSMA-CD protocols due to the fact that they consider real-time properties of the traffic. The strength of splitting protocols, however, is their ability to efficiently resolve collisions, which of course occur much less frequently when CSMA and CD are used.

Algorithmically simpler CSMA-CD splitting protocols than Zhao et al. (1990) were presented for both hard and soft real-time systems by Arvind (1991), where protocol operations were simulated and some worst case performance analysis was presented. A similar protocol for wireless transmission of hard and non real-time data was analyzed in detail by Papantoni-Kazakos (1992). Hers is a hard real-time variant of the soft real-time protocol presented and analyzed by Paterakis, Georgiadis & Papantoni-Kazakos (1989) described below.

Further examining splitting protocols, they can be grossly classified based on two common properties. Namely, the random variable used for splitting and the splitting methodology used. As a concrete example, the first data communication splitting algorithm proposed, that we call CTM here using the surname initials of the three researchers, was by Capetanakis(1979*a*, 1979*b*) and, independently, by Tsybakov & Mikhailov (1978). In their method, the involved nodes flip coins after a collision. Those who flip tails wait, while those flipping heads retransmit. This continues recursively on both sets of packets, so that a high multiplicity collision will result in a large number of subwindows. The splitting variable is the coin toss outcome, and the splitting method is a fully recursive one.

Gallager (Gallager 1978, Bertsekas & Gallager 1992) and, independently, Tsybakov & Mikhailov (1980) improved on the fully recursive algorithm in a number of ways. In this algorithm, that we name GTM again using surname initials, the splitting variable is queue arrival time. The splitting method uses a sliding partition. That is, there are never more than two split sets; the partition slides to the left, or older, side until zero or one packets are "trapped" there.

The more recent two cell splitting by Paterakis et al. (1989) is similar in many respects to the GTM algorithm except that the splitting variable is again a coin toss outcome, though the splitting technique still never allows more than two sets of packets. There are other differences between the algorithms, but for broad categorization, these two properties are useful for the sake of comparison against those presented below.

Most algorithms presented in this paper use laxity, *i.e.,* time until the packet expires, for the splitting variable. We combine this with the splitting methodologies used by the algorithms described above to develop new splitting protocols for real-time use.

Random access MAC algorithms, real-time or not, can often be characterized and analyzed with regenerative processes (Gallager 1996). In the protocols we present, one "cycle" of the system can be thought of as the amount of time to resolve channel contention by some number of nodes. For the analysis of systems of this sort, Georgiadis, Merakos & Papantoni-Kazakos (1987) developed a straightforward, general method of delay analysis using regenerative properties of random multiple access algorithms. This was elaborated on by Paterakis et al. (1989) where they present and analyze a simple protocol appropriate for limited types of soft real-time systems where all packets initially have the same deadline. While Paterakis *et al.* studied a specific protocol's performance for initial deadlines of up to 30 slots, Panwar, Towsley & Armoni (1993) use the value iteration method to find the optimal splitting algorithm for fixed initial deadlines of up to 4 slots. In (Markowski & Sethi 1995), we extended the technique of Paterakis *et al.* to analyze a somewhat more complex protocol but still used fixed initial deadlines. We included consideration of varying initial deadlines in (Markowski & Sethi 1996). Here, we present a more efficient technique than in (Markowski & Sethi 1996) for the case of varying initial deadlines, and apply it to the analysis of three blocked access protocols. We then present and use still another analytic technique to study the performance of the free access counterparts of the blocked access algorithms.

3 System Model

The system studied is an infinite population of similar users, each with a queue of length one, sharing a common channel. Each new packet is assumed to arrive at a new node. The channel is accessed in a slotted manner such that all transmissions begin only at slot boundaries, and a packet is exactly one slot long. It is further assumed that at the end of each slot, binary feedback, *i.e.*, collision or non-collision status, is available describing the slot just ended. Typically, a carrier sensing by the node providing the binary feedback is performed for some short duration at the midpoint of the data slot. At that time, collision/non-collision status is determined, so that there is ample time to provide the feedback immediately following the data slot. Finally, for this initial study, an error-free feedback channel is used.

Two types of random access algorithms (RAAs) are considered: blocked and free access. In a blocked access RAA, after some initial collision between two or more packets occurs, only the packets involved in that collision may contend for the channel. When the collision is resolved, and all packets have been either

transmitted or dropped, only then can other nodes again contend. Conversely, in free access RAAs, there is no such access blocking. That is, regardless of whether or not any collisions have occurred, all nodes are continually able to contend for the channel. Free access RAAs, therefore, generally have lower maximum throughputs than blocked access RAAs due to higher contention.

Blocked access RAAs have two components: the first time transmit rule (FTTR), and the collision resolution algorithm (CRA). The FTTR is used one or more times in succession until a collision occurs, at which time the CRA resolves the contention. The FTTR used by all RAAs described in this paper is to transmit a waiting packet with probability one. When the feedback resulting from some transmission indicates that a collision occurred, the CRA is initiated. The CRA completes when all packets involved in the initial collision have been either transmitted or dropped due to missed deadlines. The length of time from the beginning to the end of the CRA is called the collision resolution interval (CRI), and all subsequent analyses are based on one "cycle" of this regenerative stochastic process.

Splitting CRAs use some random variable to separate the collided users into at least two new sets, after which, one of the sets becomes active and the CRA is applied to that set. Traditionally, the splitting variable is either the outcome of a coin flip or the packet arrival time. In our algorithms, we split into two groups of the same size. However, Wong (1964) showed that this is not always optimal. He derived a functional-minimization equation of the dynamic programming type showing that splitting at the midpoint of a set can be a suboptimal solution for locating a number pulled from a uniform distribution. Though this is the case, for the sake of analysis, we do, however, split equally in our algorithms.

The splitting variable used is tied to the deadline of the packet, namely, 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. At each slot boundary, a packet's laxity is decremented by one and the packet discarded should the laxity reach zero. By splitting based on laxity, the real-time requirements of the system can best be met by making the low laxity group the active one so that contending packets are always transmitted in a laxity ordered manner, low to high, as appropriate in a real-time environment. Initial packet laxities are drawn from a uniform distribution in the interval $[2, T]$. Two is the minimum laxity since it takes one slot to receive feedback plus one more for actual transmission. The maximum initial laxity T is a system parameter and is chosen based on the real-time application's functional requirements.

Another characteristic common to all CRAs analyzed here is 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 CRIs, while individual packets hope for transmission regardless of delay as long their deadlines are met. We balance these views by outright dropping of late packets. A higher protocol layer decides if it is worth retransmitting late or not, especially in the case of multiple MAC layer packets making up a single higher layer message.

4 Blocked Access CRAs

By the nature of blocked access CRAs, the time when contending packets entered the system is known; it is the start of the CRI. At any time during the CRI, an initial collision that happened some while ago is being resolved. This means that there is lag of, say, d slots between "now" and the start of the CRI. The analysis must consider both this lag, and, based on the packet arrival process, the number of packets expected to arrive while the CRI is ongoing. Because the lag can become large when the intensity of the arrival process is high, *windows* are used, as first introduced by Gallager (Bertsekas & Gallager 1992). With large lags, it is likely that when a CRI completes, it will be immediately followed by a collision. To avoid this, a window Δ slots wide is used so that it is less likely to encompass more than one packet.

The window is time based and allows packets whose arrival times fall within the window to transmit. But, should a collision occur, a new window is used where the packets involved are reordered in a laxity window that spans the entire range of possible initial laxities $[2, T]$. This is the window upon which the laxity based CRAs, described shortly, operate. When the CRI ends, the time based window slides forward another Δ slots so that arrivals in that window can then transmit. Figure 4 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. To successfully transmit all packets in the current window, the CRI must complete within the next $T - d$ slots. The variable $l_{u,d}$ is defined in detail later but is simply the length of the CRI from some time t until its end at time t'.

An equivalent approach to using a sliding arrival-time based window might be to use a sliding laxity window in conjunction with the FTTR. The idea is that even when a CRI is not in progress, a laxity window would be used to allow some subset of the full laxity range to transmit. It turns out, however, that there is no appreciable difference between the two. For both the sliding laxity and sliding time windows, the goal is to find a window size reducing the likelihood of collisions while also increasing the likelihood of finding a single packet. In both cases, when a collision does occur, the average case will involve the same number and type of packets. Due to the memoryless nature of the Poisson process, there is no advantage (or disadvantage) to using a sliding window of one type over a sliding window of another. When optimally chosen, likelihood of collisions is the same, and any collision is followed by use of the same laxity based CRA. Simulations confirm this, and we chose a sliding arrival-time based window. While this avoids the added step required with sliding laxity windows of determining optimal window widths for each new laxity set, using a sliding laxity window can be advantageous in practice. It offers the chance to separate the splitting variable from the arrival process. In systems with non-Poisson arrivals, periodic ones, for example, this can be especially useful.

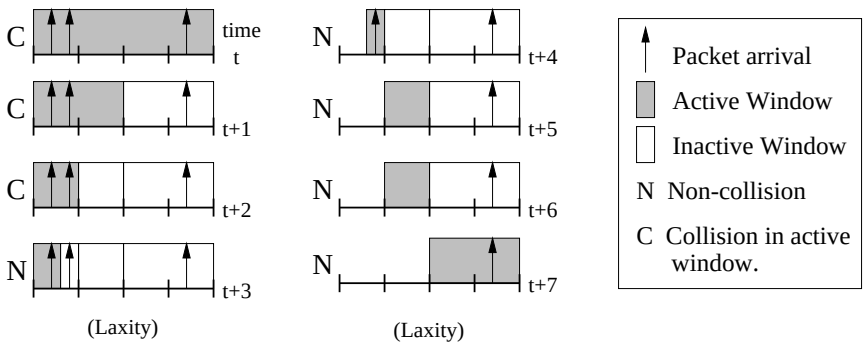

Figure 1: Fully recursive CRA.

4.1 Blocked Access Fully Recursive CRA

The first protocol considered uses packet laxity as the splitting variable and recursively splits each laxity window on subsequent collisions. A sliding time window is used as well in the style of the GTM algorithm to decrease the likelihood that a CRI will be followed by a collision. This is akin to the modification made by Massey (1981) to the original CTM CRA. That is, a sliding arrival-time based window encompasses some range of arrival times. Only packets whose arrival times fall within the window may contend for the channel, so that the likelihood of high multiplicity collisions is lessened.

When a collision does occur, however, the packets enabled in the time based window are reordered in a new laxity window encompassing the full laxity range. That window is split, and the CRA is applied to the left half of the window until no more packets are waiting in it to be transmitted. Then the CRA is applied to the right half of the laxity window. This is illustrated in Figure 1, though the initial sliding time window is not shown. As a result, a splitting tree of arbitrary depth, *i.e.,* a large number of subwindows, can exist at a given point in time in a CRI. The only way to know when the CRI completes is for all nodes to be monitoring channel history since system startup and for there to be no errors in the feedback. Note the many consecutive non-collisions in the figure. Were a node to begin channel monitoring at the fourth slot, the first non-collision, in the CRI, it would have no way of knowing that a CRI is even in progress or that four windows need to be resolved. It would simply see a channel without collisions, essentially idle. While clearly impractical for implementation, it is interesting to consider this CRA for the sake of comparison to other real-time CRAs.

4.2 Blocked Access Sliding Partition Real-Time CRA

Figure 2 illustrates how a more practical algorithm operates. This CRA is a modification of the blocked access fully recursive CRA in that the CRA is not applied recursively to each window half. Rather, after a collision, a laxity partition separates the full range into two halves. Packets in the left half then retransmit. If there is a collision, the laxity partition is slid to the midway point of the current left half. In this way,

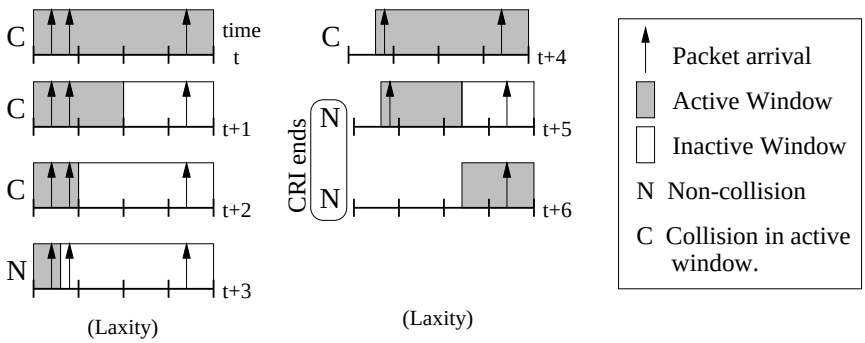

Figure 2: Sliding partition CRA.

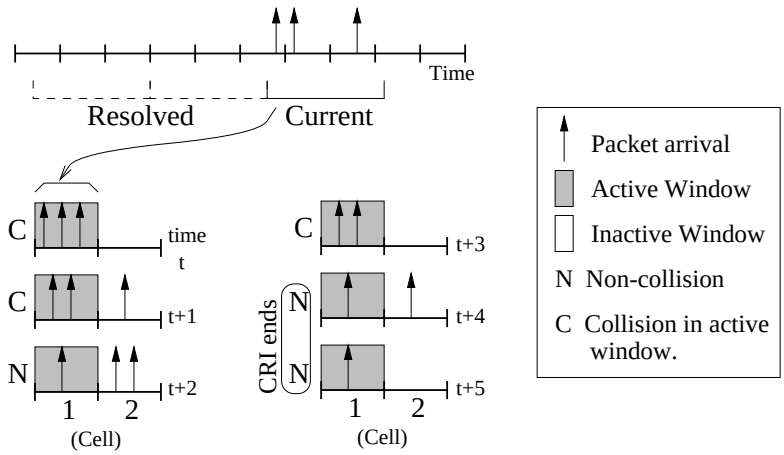

Figure 3: Two cell CRA.

there are never more than two windows during a CRI, as can be seen in the figure. And after the left half transmission is a non-collision, the CRA is applied to the right half. This is similar to, but not quite the same as, the improvement made to the original CTM CRA by Gallager (Gallager 1978)(Bertsekas & Gallager 1992) and Tsybakov & Mikhailov (1980) that yielded the GTM CRA. In the ternary feedback GTM algorithm, after a successful transmission in the left half, the remaining right half is ignored. The window slides beyond the now resolved left half, and begins anew. Due to the fact that this CRA switches from an initial arrival-time based window to a laxity window, in addition to the fact that only binary feedback is available, we are unable to do the same here. However, due to the nature of both this and the GTM CRAs, two consecutive non-collisions still indicate the end of a CRI.

4.3 Blocked Access Real-Time Two Cell

Paterakis et al. (1989) developed a CRA that uses random splitting like the original CTM CRA, but also incorporates windowing in the same manner as the GTM and sliding partition CRAs described above. There

are always only two subsets of split users, those in cell one and those in cell two. Upon collision, nodes flip a fair coin and either remain in cell one or move to cell two. Nodes in cell two never transmit. Cell one nodes, however, always transmit at the next slot boundary. This has the advantages of the sliding partition CRA—at most two windows active—with an advantage also found in the original CTM CRA; namely, the splitting is independent of the arrival process. In Figure 3, the illustration shows the arrival-time based sliding window where the current one happens to allow three packets to contend for the channel. After the initial collision, the rest of the figure shows how the CRI might proceed given that there is a coin flip by all cell one nodes after a collision. Like the previous CRA, two consecutive non-collisions signal the CRI completion. This makes it easy for nodes to join the net any time after system startup, and to resynchronize in the inevitable case of feedback errors.

5 Blocked Access Analysis

In such systems, it is straightforward to describe the performance measures. So that the interested reader can easily perceive our extensions to the analytic technique of Paterakis et al. (1989), we follow their notational scheme where possible. Due to space considerations, we present the analysis only for the real-time sliding partition CRA. Using the modifications described, corresponding expressions for the other protocols have been derived. The intensity of the Poisson arrival process, in packets per slot, is indicated by λ. This is the aggregate arrival rate, or offered load, at the MAC layer. That is, whether or not there is any blocking of packets at higher protocol layers, a system-wide λ packets/slot is assumed to reach the MAC layer, though not all will be ultimately transmitted. Then, if H denotes the expected length of a CRI, and Z the expected number of packets that will be transmitted during that CRI, the fraction of packets that meet their deadlines is

$$\rho = (Z/H)/\lambda. \tag{1}$$

If we next define W as the cumulative expected delay of all packets in the CRI, then the per packet expected delay is

$$D = W/Z. \tag{2}$$

In the following sections, expressions for H, Z, and W are derived. We begin with basic expressions and build upon them to derive expressions for the variables above. The expressions are, essentially, just somewhat complicated counters. That is, H counts the number of slots in a CRI, Z counts the number of packets transmitted, and W counts the number of slots spent waiting. In all cases, the counter value is returned at the end of a CRI.

5.1 Probability of Packet Expiration

During each slot of a CRI, there is some probability that one or more of the contending packets will expire. Recalling that T is the maximum possible initial laxity and that B is the maximum number of slots remaining

in the CRI, $T-B$ is the current length, in slots, of the CRI. At each slot boundary, the laxity of each packet decreases by one. As a result, the window can be thought to slide to left by one slot. Because late packets are dropped, however, the window edges are never less than one. This is illustrated in Figure 5. The probability, then, that one packet expires is conditional on location of the current laxity window. Since packet laxities are drawn from a uniform distribution, the probability of a single packet being dropped when at most B slots remain in the CRI is

$$p_1 = P(\text{pk drop} \mid T, B, x, w) = 1 - \frac{\max(1, x + w - (T - B)) - \max(1, x - (T - B))}{w}. \tag{3}$$

Note that immediately after the collision that initiates the CRI, B will be $T-1$. After the CRI has progressed for $T - B$ slots, the left and right edges of the laxity window move to the left by that much as well. The probability of not being in that window is the probability of a packet being dropped.

The probability of multiple packets being dropped is obtained by treating each single drop probability as a Bernoulli trial. The number of total drops is then described with a binomial distribution, which is

$$\text{bin}(p, n, k) = \binom{n}{k} p^k (1-p)^{n-k},$$

using p_1 as the probability. Therefore, given a window spanning laxities $[x, x + w]$ and with only B slots remaining in the CRI, the probability that d of the k packets will be dropped is

$$\text{bin}(p_1, k, d).$$

5.2 Probable Length of CRI

In most of the subsequent equations, there is consideration of some event occurring within a CRI of length l, which then must be multiplied by the probability that a CRI occurs that is, in fact, of length l. This latter probability is derived below. For a collision involving some finite number of packets, the probable length of the CRI can be expressed recursively. The notation $P_{len}(l \mid k_1, k_2, B, x, w_1, b)$ is used to indicate the conditional probability of the CRI being l slots long given that there are k_1 packets in the left subwindow, k_2 packets in the right, and at most B slots remaining in the CRI. Furthermore, the left edge of the current window is at laxity x, and the left subwindow has width a, and the right has width b. The conditional probability that a CRI is of length l when an initial collision of multiplicity k packets occurs can then be represented by $P_{len}(l \mid k, 0, B, 2, T, 0)$. That is, by convention, we assume a CRI begins with all k packets in a left subwindow where the right subwindow is of zero width.

Realizing that a collision uses one slot, the probable length of the CRI can then be calculated using an equation of the form $P_{len}(l \mid \cdot) = \ldots P_{len}(l-1 \mid \cdot) \ldots$ However, because packets have initial laxities drawn from a range of laxities, there is some probability that one or more packets will expire during the current slot of the CRI. Thus, a summation over the number of packets lost multiplied by the probability of the

likelihood of such an event is needed. The full expression, with initial conditions omitted for brevity, is

$$P_{len}(l \mid k_1, k_2, B, x, w_1, w_2) = \begin{cases} \sum_{d=0}^{k_2} P_{len}(l-1 \mid k_2-d, 0, B-1, x+w_1, w_2, 0) & \\ \quad \cdot P_{drop}(k_2, d \mid T, B, x+w_1, w_2) & k_1 = 0, 1, \\ \sum_{i=0}^{k_1} \mathrm{bin}(w_1/(2w_1+2w_2), k_1, i) \sum_{d=0}^{i} & \\ \quad P_{len}(l-1 \mid i-d, k_2+k_1-i, B-1, x, w_1/2, w_2+w_1/2) & \\ \quad \cdot P_{drop}(i, d \mid T, B, x, w_1/2) & \text{otherwise.} \end{cases} \tag{4}$$

The above gives the probability in terms of specific cases for values of k_1 and k_2. In the general case for a Poisson arrival process, the probability of a CRI lasting l slots is again expressed recursively. Let $P_{len}(l \mid u, d)$ denote the conditional probability that a CRI is l slots long given that u slots must be examined, and that the current lag is d slots. This probability is the sum of the probabilities that the u slots contain 0, 1, ..., ∞ packets and that these packets can be successfully transmitted within the remaining time, $T - d$.

$$P_{len}(l \mid u, d) = \sum_{k=0}^{\infty} P_{len}(l \mid k, 0, T - \lceil d \rceil, 2, T-1, 0) e^{-\lambda u} \frac{(\lambda u)^k}{k!}. \tag{5}$$

5.3 Expected Length of CRI

An expression for the expected length of a CRI is developed in a manner similar to that used for the probable length of a CRI. Given an initial collision involving k packets and at most B slots till the end of the CRI, we denote the expected length as $L_{k_1,k_2,B,x,w_1,b}$. We again must consider the probabilities of one or more of the k packets expiring as the CRI. For $k > 1$, *i.e.*, initial conditions not listed, we get

$$L_{k_1,k_2,B,x,w_1,w_2} = \begin{cases} 1 + \sum_{d=0}^{k_2} L_{k_2-d,0,B-1,x+w_1,w_2,0} \cdot P_{drop}(k_2, d \mid T, B, x+w_1, w_2) & k_1 \le 1, \\ 1 + \sum_{i=0}^{k_1} \mathrm{bin}(w_1/(2w_1+2w_2), k_1, i) \sum_{d=0}^{i} & \\ \quad L_{i-d,k_2+k_1-i,B-1,x,w_1/2,w_2+w_1/2} & \\ \quad \cdot P_{drop}(i, d \mid T, B, x, w_1/2) & \text{otherwise.} \end{cases} \tag{6}$$

For the general case where u slots must be resolved with a current lag of d, the expected CRI length is

$$E\{l_{u,d}\} = \sum_{k=0}^{\infty} E\{\, l_{u,d} \mid k, T - \lceil d \rceil \,\} e^{-\lambda u} \frac{(\lambda u)^k}{k!}. \tag{7}$$

With these expressions, we can now use the high level relationship induced by time-constrained, blocked access CRAs, as put forth by Paterakis et al. (1989):

$$h_d = \begin{cases} l_{d,d}, & l_{d,d} = 1, & 1 \le d \le \Delta \\ l_{d,d} + h_{l_{d,d}}, & 1 < l_{d,d} \le T - \lceil d \rceil, & 1 \le d \le \Delta \\ l_{\Delta,d} + h_{d-\Delta+l_{\Delta,d}}, & 1 < l_{d,d} \le T - \lceil d \rceil, & \Delta < d \le T-1. \end{cases} \tag{8}$$

Taking expectations and denoting $H_d = E\{h_d\}$ yields

$$H_d = \begin{cases} E\{l_{d,d}\} + \sum_{m=2}^{T-\lceil d \rceil} H_m P_{len}(m \mid d, d), & 1 \le d \le \Delta, \\ E\{l_{\Delta,d}\} + \sum_{m=1}^{T-\lceil d \rceil} H_{d-\Delta+m} P_{len}(m \mid \Delta, d), & \Delta < d \le T-1. \end{cases} \tag{9}$$

This is a finite system of linear equations that can be easily solved with standard methods, *e.g.*, Gauss-Jordan elimination. The solution for the variable H_1 represents the expected number of slots required to begin with a lag of 1 slot and to return to a lag of 1 slot. This is the expected value for the CRI length dependent on arrival process, initial laxity range, and window width Δ.

5.4 Expected Number of Transmissions During CRI

The expression for the number of packets transmitted during a CRI follows the same recursive pattern as does the expression for expected CRI length. However, where the expected length expression counts the number of slots in the CRI, the expected number of transmissions counts packets. After each non-collision, the counter is incremented by the number transmitted; either zero or one.

$$N_{k_1,k_2,B,x,w_1,w_2} = \begin{cases} k_1 + \sum_{d=0}^{k_2} N_{k_2-d,0,B-1,x+w_1,w_2,0} \cdot P_{drop}(k_2, d \mid T, B, x+w_1, w_2) & k_1 = 0, 1, \\ \sum_{i=0}^{k_1} \mathrm{bin}(w_1/(2w_1+2w_2), k_1, i) \sum_{d_1=0}^{i} & \\ \quad N_{i-d_1,k_2+k_1-i,B-1,x,w_1/2,w_2+w_1/2} & \\ \quad \cdot P_{drop}(i, d_1 \mid T, B, x, w_1/2) & \text{otherwise.} \end{cases} \tag{10}$$

As before, the general case of resolving a window of u slots with a d slot lag is

$$E\{n_{u,d}\} = \sum_{k=0}^{\infty} E\{\, n_{u,d} \mid k, T - \lceil d \rceil \,\} e^{-\lambda u} \frac{(\lambda u)^k}{k!}. \tag{11}$$

Similar to the CRI length expression, the relationship induced by the CRA is:

$$\alpha_d = \begin{cases} n_{d,d}, & l_{d,d} = 1, & 1 \le d \le \Delta \\ n_{d,d} + \alpha_{l_{d,d}}, & 1 < l_{d,d} \le T - \lceil d \rceil, & 1 \le d \le \Delta \\ n_{\Delta,d} + \alpha_{d-\Delta+l_{\Delta,d}}, & 1 < l_{d,d} \le T - \lceil d \rceil, & \Delta < d \le T-1. \end{cases} \tag{12}$$

Taking expectations and denoting $A_d = E\{\alpha_d\}$ yields

$$A_d = \begin{cases} E\{n_{d,d}\} + \sum_{m=2}^{T-\lceil d \rceil} A_m P_{len}(m \mid d, d), & 1 \le d \le \Delta, \\ E\{n_{\Delta,d}\} + \sum_{m=1}^{T-\lceil d \rceil} A_{d-\Delta+m} P_{len}(m \mid \Delta, d), & \Delta < d \le T-1. \end{cases} \tag{13}$$

$Z = A_1$ is the expected number of packets transmitted during a CRI dependent on arrival process, initial laxity range, and window width Δ.

5.5 Expected Cumulative Delay During CRI

The cumulative delay, Z, of all packets transmitted during the CRI is obtained by combining the previous two expressions to sum the number transmitted and slots spent waiting. The number transmitted happens

to also be the number of slots required to transmit them, so that for a given number of packets, the expected cumulative delay is

$$Z_{k_1,k_2,B,x,w_1,w_2} = \begin{cases} k_1 + \sum_{d=0}^{k_2}(Z_{k_2-d,0,B-1,x+w_1,w_2,0} + N_{k_2-d,0,B-1,x+w_1,w_2,0}) & \\ \quad \cdot P_{drop}(k_2, d \mid T, B, x+w_1, w_2) & k_1 = 0,1, \\ \sum_{i=0}^{k_1} \mathrm{bin}(w_1/(2w_1+2w_2), k_1, i) \sum_{d=0}^{i} & \\ \quad (Z_{i-d,k_2+k_1-i,B-1,x,w_1/2,w_2+w_1/2} & \\ \quad + N_{i-d,k_2+k_1-i,B-1,x,w_1/2,w_2+w_1/2}) & \\ \quad \cdot P_{drop}(i, d \mid T, B, x, w_1/2) & \text{otherwise.} \end{cases} \tag{14}$$

Therefore, the cumulative delay of packets transmitted during a single CRI (excluding delay due to lag and window width) is

$$E\{z_{u,d}\} = \sum_{k=0}^{\infty} E\{\, z_{u,d} \mid k, T - \lceil d \rceil \,\} e^{-\lambda u} \frac{(\lambda u)^k}{k!}. \tag{15}$$

From the memoryless nature of the Poisson process, the average packet position in a window is the middle. Therefore each packet in the window experiences, on average, a delay of half the window length. The expected delay within a window, then, is simply,

$$E\{\psi_{u,d}\} = \frac{1}{2} u E\{n_{u,d}\}.$$

The time spent waiting is made up of three parts: a packet's expected position in a window, the amount of time before the window encompasses the packet, and the number of slots required for the CRA to transmit the packet. This is expressed as

$$w_d = \begin{cases} \psi_{d,d} + z_{d,d}, & l_{d,d} = 1, & 1 \le d \le \Delta \\ \psi_{d,d} + z_{d,d} + w_{l_{d,d}}, & 1 < l_{d,d} \le T - \lceil d \rceil, & 1 \le d \le \Delta \\ \psi_{\Delta,d} + z_{\Delta,d} & & \\ \quad +(d-\Delta) n_{\Delta,d} w_{d-\Delta+l_{\Delta,d}}, & 1 < l_{d,d} \le T - \lceil d \rceil, & \Delta < d \le T-1. \end{cases} \tag{16}$$

Taking expectations where $W_d = E\{w_d\}$, we see that

$$W_d = \begin{cases} E\{\psi_{d,d} + z_{d,d}\} + \sum_{m=2}^{T-\lceil d \rceil} W_m P(m \mid d, d), & 1 \le d \le \Delta, \\ E\{\psi_{\Delta,d} + z_{\Delta,d} + (d-\Delta) n_{\Delta,d}\} + \sum_{m=1}^{T-\lceil d \rceil} W_{d-\Delta+m} P(m \mid \Delta, d), & \Delta < d \le T-1 \end{cases} \tag{17}$$

The solution for the variable $W = W_1$ represents the expected cumulative delay of the packets transmitted during a CRI. With W, it is now possible to substitute in values to Equations 1 and 2 for performance evaluation.

6 Blocked Access Evaluation

A soft real-time system can be characterized by at least the following: traffic rate λ, laxity range $[2, T]$, minimum acceptable success ratio e_1, and average delay desired e_2. Ideally, a protocol would adapt to a

changing network by modifying these values while still meeting quality of service guarantees. More simply, though, if a system's real-time requirements are known at design time, it is easy to determine protocol performance.

While it is the application that drives the choices for laxity range $[2, T]$ and success rate e_1, the initial time window width Δ is a design parameter of the protocol itself. That is, it is chosen from the optimization of the analysis of the previous section. Because of the complexity of the expressions developed, deriving closed form solutions is difficult. For a given value of Δ, however, the problem is simple to solve numerically and, as pointed out by Paterakis et al. (1989), reduces to:

$$\lambda^*_{T,e_1} = \sup(\lambda : \rho_T(\Delta, \lambda) \geq e_1). \tag{18}$$

We present results only for $\Delta = 2.5$ though data has been generated for various Δ values that, through observation, encompass the maximum throughputs. We also conducted simulation studies of each protocol using Opnet (Mil 3, Inc. 1996), a network simulation package. The simulator models the system as an infinite user population, thus offering more conservative performance results than the finite user case (Paterakis, Georgiadis & Papatoni-Kazakos 1987). Each simulation was run with 0.95 probability that the fraction of successful transmissions ρ was within ± 0.005 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. It is important to note that the simulator is not numerically solving the recursions of the analysis. It performs the various splitting techniques in the same manner that an implementation would, and samples are gathered with statistics generated upon reaching steady state. The analytic curves, in contrast, are generated solely from numerical solutions to Equations 1 and 2.

For the blocked access sliding partition CRA, Figure 6 graphs analytical and simulation success rate results for $\lambda^*_{10,0.9}$ and illustrates the close correlation of data, even though the two methods used to acquire the performance results are very different. Figure 7 graphs a similar comparison with the blocked access sliding partition CRA, but for delays corresponding to $\lambda^*_{5,0.9}, \lambda^*_{10,0.9}$, and $\lambda^*_{15,0.9}$. Error bars are not shown since the condition of reaching steady state was made by monitoring ρ, not the delay. Because of the close correlation between the two sets of results, subsequent graphs show only one or the other set for the sake of readability. With simulation results, error bars will not be shown but in all cases are based on the ± 0.005 tolerance of ρ. Analytical results for the sliding partition algorithm are graphed in Figure 8, showing expected CRI lengths for various values of T and λ. As expected, when the intensity is high, CRI length begins to increase dramatically, and especially when T is large.

For maximum initial laxity $T = 10$, analytical results for the three protocols are compared in Figure 9. It is interesting that the fully recursive CRA outperforms the others. In the original ternary feedback versions, and even when windowing was added to the fully recursive CRA, it performed worst. In the soft real-time environment, it appears that the finer window splitting of that CRA offers some advantage. However, when $T = 30$, the advantage is lost as illustrated in Figure 10. While splitting finely allows perhaps quicker

isolation of contending packets, it is then necessary for the recursion to spend another slot doubling the window size, thus losing valuable time.

As one might expect, the sliding partition CRA, which takes packet laxities into account, performs better than the random splitting of the two cell CRA. As laxity range increases, the performance differences between the two become more significant. This is shown in Figure 11. Furthermore, as seen in Figure 12, the CRA is seen to approach some operational maximum as the value of T is increased. From this, it appears that results for, say, $T = 30$ could be used to approximate still wider laxity ranges. More importantly, it indicates that the LLC (Logical Link Control) layer that submits packets to the MAC layer must be aware that though deadlines of its frames might extend relatively far into the future, MAC peak performance is reached with deadlines no more than 30 slots away. This will undoubtedly have some effect on design of the LLC scheduling algorithm.

7 Free Access CRAs

Because free access CRAs allow packets to contend for the channel at any time, there is no way of knowing at what point during a CRI a given packet arrived. At the beginning of each slot, there is some probability that the arrival process will add zero or more packets to the system, and some probability that the departure process—a combination of the CRA and expiring packets—will remove zero or more packets from the system. Therefore, a time based window is also unnecessary.

7.1 Free Access Real-Time Sliding Partition/Fully Recursive CRA

The blocked access sliding partition CRA is easily modified for use in a free access manner. After the time based window is removed, the only other change is that the left edge, the low laxity window edge, never moves. By doing so, regardless of the progress in the CRI, lower laxity packets are always admitted. Higher laxity packets, though, must wait for the lower laxity ones to resolve any collisions. So the algorithm does not implement free access in the purest sense, but does so in a way that seemingly best supports soft real-time transmissions. Interestingly, by similarly modifying the fully recursive laxity splitting algorithm, its free access variant is identical to the free access sliding partition CRA.

7.2 Two Cell Random Splitting

The two cell CRA is also modified by removing the time based window. It is further modified such that all new arrivals join the waiting group and react to the feedback exactly as if they were in the waiting group during the previous slot. That is, they transmit after a non-collision, and remain in the waiting group after a collision.

8 Analysis

For free access analysis, the expressions for H, Z, and W must again be derived for use in Equations 1 and 2. Because there is no need for a time-based window or to track its location relative to the current lag, the top level equations for the free access analysis are more straightforward than for blocked access. However, because it is no longer known when a given packet enters the CRI, the expression for the packet drop probability becomes more involved. Because a CRI is not guaranteed to be shorter than the largest initial laxity, as in the blocked access case, the notation changes slightly. Rather than have a bound B counting down from the maximum laxity of T slots, we start a counter P from zero and increment it at each new slot.

8.1 Probability of Packet Drop

Given that there is a packet arrival within the last P slots and that the arrival process is Poisson, it is equally likely that the packet arrived in any of the last P slots. When $P < T$, the probability of a packet drop is the same as the blocked access case. For $P \geq T$, though, since T is the maximum initial laxity, we know that the packet must have arrived within the last T slots. Therefore, the probability that a single packet will be dropped is:

$$p_1 = P(\text{pk drop} \mid T, P, x, w) = \sum_{i=2}^{\min(P,T-1)} \frac{1}{T-1} \cdot \frac{\max(1, x+w-(T-i)) - \max(1, x-(T-i))}{w}. \quad (19)$$

As in the blocked access derivation, the probability that d packets out of an existing k packets are dropped is a binomial distribution utilizing the probability above. This group drop probability is again denoted by

$$\text{bin}(p_1, k, d).$$

8.2 Expected Length of CRI

The expected length of a CRI is obtained in a straightforward manner, though the notation is somewhat involved. At the beginning of each slot, the Poisson arrival process determines the likelihood of new arrivals, while the CRA determines the likelihood of departures. With these probabilities, the expected CRI length is expressed as

$$L_{k_1,k_2,P,w_1,w_2} = \begin{cases} 1 + \sum_{a=0}^{\infty} \sum_{d=0}^{k_2} L_{k_2-d+a,0,P+1,w_1+w_2,0} & \\ \quad \cdot P_{drop}(k_2, d \mid T, P, w_1+w_2) e^{-\lambda} \frac{\lambda^a}{a!}, & k_1 \leq 1 \\ & \\ 1 + \sum_{i=0}^{k_1} \text{bin}(w_1/(2w_1+2w_2), k_1, i) \sum_{d_1=0}^{i} \sum_{d_2=0}^{k_2+k_1-i} \sum_{a_1=0}^{\infty} \sum_{a_2=0}^{\infty} & \\ \quad L_{i-d_1+a_1,k_2+k_1-i-d_2+a_2,P+1,w_1/2,w_2+w_1/2} & \\ \quad \cdot P_{drop}(i, d_1 \mid T, P, w_1/2) & \\ \quad \cdot P_{drop}(k_2+k_1-i, d_2 \mid T, P, w_1/2, w_2+w_1/2) & \\ \quad \cdot e^{-\lambda} \frac{\lambda^{(a_1+a_2)}}{(a_1+a_2)!} \left(\frac{w_1}{w_1+w_2}\right)^{a_1} \left(\frac{w_2}{w_1+w_2}\right)^{a_2} & \text{otherwise.} \end{cases} \quad (20)$$

Recall that in the soft real-time, free access version of the sliding partition CRA, that the left edge of the laxity is always 2. This means that after a successful transmission, the laxity window is fully expanded to the full range. For the case above, where $k_1 \leq 1$, any new arrivals thus will be members of the subsequently full-sized laxity window. In the case where $k_1 > 1$, however, the probabilities of arrivals entering either half of the window are considered.

8.3 Expected Number of Packets Transmitted During CRI

The expected number of packets successfully transmitted during a CRI is similarly derived. At the beginning of each slot, the Poisson arrival process determines the likelihood of new arrivals, while the CRA determines the likelihood of departures. With these probabilities, the expected CRI length is expressed as

$$Z_{k_1,k_2,P,w_1,w_2} = \begin{cases} k_1 + \sum_{a=0}^{\infty} \sum_{d=0}^{k_2} Z_{k_2-d+a,0,P+1,w_1+w_2,0} & \\ \quad \cdot P_{drop}(k_2, d \mid T, P, w_1 + w_2) e^{-\lambda} \frac{\lambda^a}{a!}, & k_1 \leq 1 \\ \\ \sum_{i=0}^{k_1} \text{bin}(w_1/(2w_1 + 2w_2), k_1, i) \sum_{d_1=0}^{i} \sum_{d_2=0}^{k_2+k_1-i} \sum_{a_1=0}^{\infty} \sum_{a_2=0}^{\infty} & \\ \quad Z_{i-d_1+a_1,k_2+k_1-i-d_2+a_2,P+1,w_1/2,w_2+w_1/2} & \\ \quad \cdot P_{drop}(i, d_1 \mid T, P, w_1/2) & \\ \quad \cdot P_{drop}(k_2 + k_1 - i, d_2 \mid T, P, w_1/2, w_2 + w_1/2) & \\ \quad \cdot e^{-\lambda} \frac{\lambda^{(a_1+a_2)}}{(a_1+a_2)!} \left(\frac{w_1}{w_1+w_2}\right)^{a_1} \left(\frac{w_2}{w_1+w_2}\right)^{a_2} & \text{otherwise.} \end{cases} \tag{21}$$

8.4 Expected Cumulative Delay

The expected cumulative packet delay during a CRI is slightly more complex. In addition to counting the slots delayed during the CRI itself, on average an arriving packet waits half a slot, due to the free access, before attempting its first transmission. Therefore, the expected delay is

$$W_{k_1,k_2,P,w_1,w_2} = \begin{cases} k_1/2 + \sum_{a=0}^{\infty} \sum_{d=0}^{k_2} (W_{k_2-d+a,0,P+1,w_1+w_2,0} & \\ \quad Z_{k_2-d+a,0,P+1,w_1+w_2,0}) & \\ \quad \cdot P_{drop}(k_2, d \mid T, P, w_1 + w_2) e^{-\lambda} \frac{\lambda^a}{a!}, & k_1 \leq 1 \\ \\ \sum_{i=0}^{k_1} \text{bin}(w_1/(2w_1 + 2w_2), k_1, i) \sum_{d_1=0}^{i} \sum_{d_2=0}^{k_2+k_1-i} \sum_{a_1=0}^{\infty} \sum_{a_2=0}^{\infty} & \\ \quad (W_{i-d_1+a_1,k_2+k_1-i-d_2+a_2,P+1,w_1/2,w_2+w_1/2} & \\ \quad + Z_{i-d_1+a_1,k_2+k_1-i-d_2+a_2,P+1,w_1/2,w_2+w_1/2}) & \\ \quad \cdot P_{drop}(i, d_1 \mid T, P, w_1/2) & \\ \quad \cdot P_{drop}(k_2 + k_1 - i, d_2 \mid T, P, w_1/2, w_2 + w_1/2) & \\ \quad \cdot e^{-\lambda} \frac{\lambda^{a_1+a_2)}}{(a_1+a_2)!} \left(\frac{w_1}{w_1+w_2}\right)^{a_1} \left(\frac{w_2}{w_1+w_2}\right)^{a_2} & \text{otherwise.} \end{cases} \tag{22}$$

9 Free Access Evaluation

The performance evaluation of the free access protocols is done using the same top level equations just as for the blocked access ones. Because at each slot boundary, free access analysis must consider the probability of new arrivals, and due to the exponential nature of the equations, it quickly becomes difficult to obtain analytic results as CRI bound T and Poisson arrival intensity increase. Therefore, while Figure 13 shows the close correlation between the analytical and simulation results for maximum initial laxity $T = 5$ in the sliding partition free access protocol, for higher T values, simulation results are more quickly obtained. The simulation results of the two free access protocols are compared in Figure 14. For each algorithm, six curves are plotted for initial laxity ranges $[2, T]$ where $T = 5, 10, 15, 20, 25, 30$. The two cell version significantly outperforms the sliding partition based CRA. This is not surprising since new arrivals always join the waiting cell avoiding collisions, on average, that are unvoidable in the free access sliding partition CRA. Conversely, while delay levels off at a later point in the free access two cell CRA, we can see in Figure 15 that delay levels off in the free access sliding partition CRA fairly fast and at a lower point, due to the increased contention, than in the two cell CRA.

In the cases of both blocked and free access algorithms, meeting performance requirements of the application layer are dependent on the MAC traffic intensity not exceeding a rate corresponding to the desired success/delay criteria. We make the simplifying assumption that this λ is either the by-product of the soft real-time system implemented on the wireless net, or else some admission control is in force in the LLC (Logical Link Control) layer just above the MAC layer.

In the case of traditional free access CRAs, free access algorithms have lower throughput because of the increased channel contention. In the soft real-time case, though, it is interesting to reconsider these algorithms because of the more involved departure process. For small laxity ranges, it turns out the the two types of algorithms perform similarly. Figure 16 compares the performance of the sliding partition blocked and free access versions when $T = 5$. The difference is slight. However, when higher ranges are compared, the performances quickly diverge. Figure 17 graphs success rates for initial laxities in $[2, T]$. We can see that as T becomes larger, the blocked access algorithm performs increasingly better, whereas the free access version's performance becomes increasingly worse. The more practical range of success rates for $e_1 = 0.9$ is illustrated in Figure 18. For both sets of analytically derived curves, values of $T = 5, 10, 15$ are graphed left to right. While both algorithms show increased success rates when T is increased, it is easily seen that even in this restricted range of success rates, the blocked access CRAs perform better. The corresponding delay curve is graphed in Figure 19.

10 Conclusions

A real-time system meets its functional requirements not just by generating correct results, but by generating correct and timely results. Deadlines can be fully supported only if the concept of time is incorporated into

all layers of the protocol stack. This has been traditionally difficult to do in random access protocols, where packet delays are potentially unbounded due to collisions. Higher layers have therefore been forced to accept lower performance than possible simply because the foundation of the protocol stack offers no support for deadlines. With the five algorithms presented, we have shown that window-splitting protocols can be modified to work successfully in soft real-time environments.

In addition to the algorithms, we have presented several analytic models of them that can be used to determine operational parameter values when given quality of service requirements. Supporting the analytic techniques, simulations of the algorithmic actions yield results nearly identical, within the given tolerance, with the numerical solutions. Results indicate that for general use, the blocked access sliding partition algorithm offers the best performance.

Addressing the practicality of the protocols studied, it was mentioned that the fully recursive algorithm, while interesting to study, cannot be implemented since it requires an error free channel and that all nodes must monitor the channel from system start up. Both the two cell and sliding partition algorithms, however, have the desirable property that CRIs are always complete when two consecutive non-collisions are observed. This allows stations to easily join the net by simply waiting for two non-collisions before their first transmissions.

Splitting protocols, though, are not well suited for the transmission of streaming data, which is best handled by connection-oriented services. Yet for wireless networks where short messages are passed between nodes that are cooperating as parts of a larger soft real-time system, splitting protocols are appropriate. In these situations, real-time splitting protocols avoid the overhead of setting up and breaking down connections yet still offer quality of service guarantees. With such guarantees made at the MAC layer, a strong foundation is provided for building real-time services throughout the protocol stack.

References

Abramson, N. (1970), "Another Alternative for Computer Communications", *Proceedings of the Fall Joint Computer Conference, AFIPS Conference*, Vol. 37, 281–285.

Abramson, N. (1985), "Development of ALOHANET", *IEEE Transactions on Information Theory*, **5**(2), 28–42.

Arvind, K. (1991), *Protocols for Distributed Real-Time Systems*, PhD thesis, University of Massachusetts, Amherst, MA.

Bertsekas, D. & Gallager, R. (1992), *Data Networks*, second edn, Prentice Hall.

Capetanakis, J. (1979*a*), "Generalized TDMA: The Multiple Access Tree Protocol", *IEEE Transactions on Communications*, **27**(10), 1476–1484.

Capetanakis, J. (1979*b*), "Tree Algorithms for Packet Broadcast Channels", *IEEE Transactions on Information Theory*, **25**(5), 505–515.

Gallager, R. G. (1978), "Conflict Resolution in Random Access Broadcast Networks", *Proceedings of the AFOSR Workshop in Communications Theory Applications*, 74–76.

Gallager, R. G. (1996), *Discrete Stochastic Processes*, Kluwer Academic Publishers.

Georgiadis, L., Merakos, L. F. & Papantoni-Kazakos, P. (1987), "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.

Markowski, M. J. & Sethi, A. S. (1995), "Analysis of a Soft Real-Time Protocol", Technical Report 96-02, Dept. of Computer and Information Sciences, University of Delaware, Newark, DE.

Markowski, M. J. & Sethi, A. S. (1996), "Evaluation of Wireless Soft Real-Time Protocols", *IEEE Proceedings Real-Time Technology and Applications Symposium*, Boston, MA, 139–146.

Massey, J. (1981), "Collision Resolution Algorithms and Random Access Communications", I. G. Longo, ed., *Multi-User Communication Systems, CISM Courses and Lectures*, Springer–Verlag, New York, 73–137.

Metcalfe, R. M. & Boggs, D. R. (1976), "Ethernet: Distributed Packet Switching for Local Computer Networks", *Communications of the ACM*, 395–404.

Mil 3, Inc. (1996), "Opnet Modeler, v3.0". Washington, DC.

Molle, M. L. & Kleinrock, L. (1985), "Virtual Time CSMA: Why Two Clocks are Better Than One", *IEEE Transactions on Communications*, **COM-33**(9), 919–933.

Panwar, S. S., Towsley, D. & Armoni, Y. (1993), "Collision Resolution Algorithms for a Time-Constrained Multiaccess Channel", *IEEE Transactions on Communications*, **41**(7), 1023–1026.

Papantoni-Kazakos, P. (1992), "Multiple-Access Algorithms for a System with Mixed Traffic: High and Low Priority", *IEEE Transactions on Communications*, **40**(3), 541–555.

Paterakis, M., Georgiadis, L. & Papantoni-Kazakos, P. (1989), "Full Sensing Window Random-Access Algorithm for Messages With Strict Delay Constraints", *Algorithmica*, **4**, 318–328.

Paterakis, M., Georgiadis, L. & Papatoni-Kazakos, P. (1987), "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.

Ramamritham, K. & Zhao, W. (1987), "Virtual Time CSMA Protocols for Hard Real-time Communication", *IEEE Transactions on Software Engineering*, **13**(8), 938–952.

Tsybakov, B. & Mikhailov, V. (1978), "Free Synchronous Packet Access in a Broadcast Channel with Feedback", *Problemy Peredachi Informassi*, **14**(4), 259–280.

Tsybakov, B. & Mikhailov, V. (1980), "Random Multiple Access of Packets: Part-and-Try Algorithm", *Problemy Peredachi Informassi*, **16**(4), 65–79.

Wong, E. (1964), "A Linear Search Problem", *SIAM Review*, **6**(2), 168–174.

Zhao, W., Stankovic, J. A. & Ramamritham, K. (1990), "A Window Protocol for Transmission of Time-Constrained Messages", *IEEE Transactions on Computers*, **39**(9), 1186–1203.

List of Figures

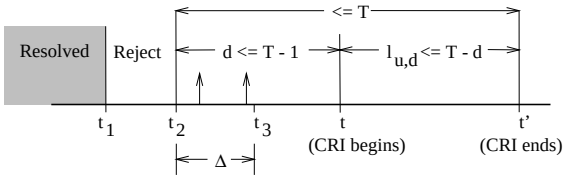

Figure 4: Relation between some variables in a CRA.

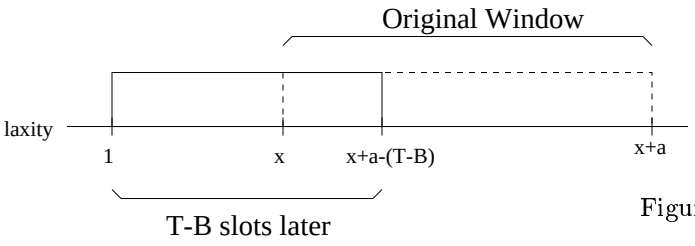

Figure 5: Sliding laxity window.

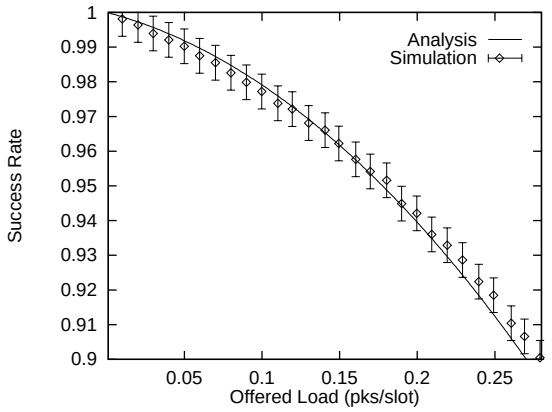

Figure 6: Blocked access sliding partition.

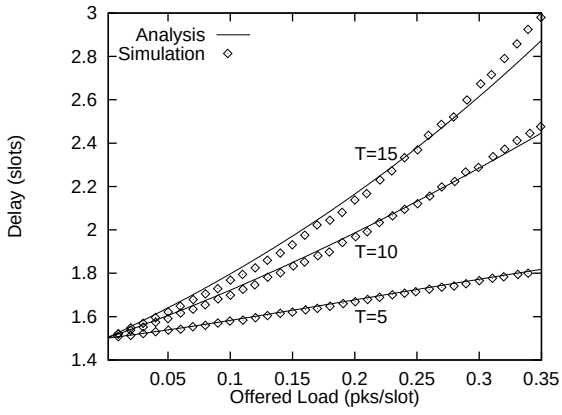

Figure 7: Blocked access sliding partition delay.

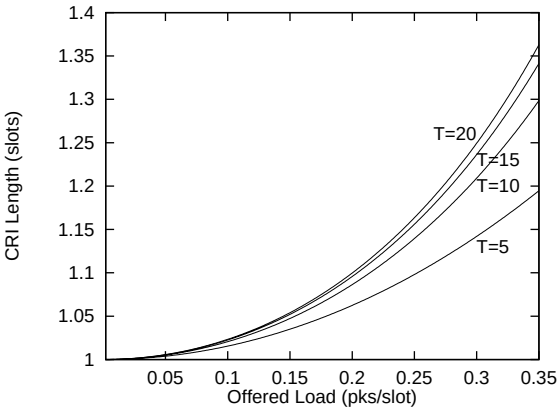

Figure 8: Blocked access sliding partition CRI lengths.

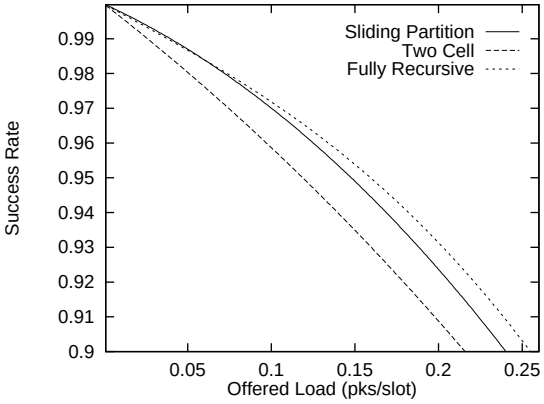

Figure 9: Blocked access protocols, $T = 10$.

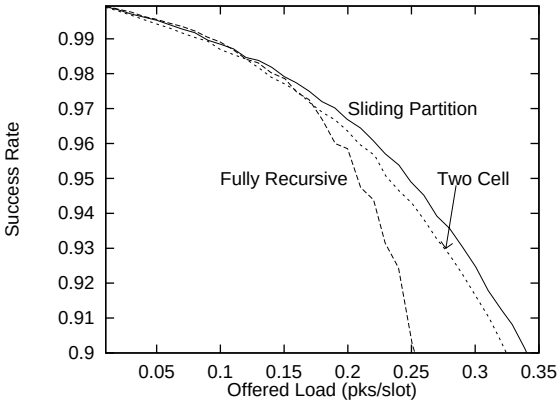

Figure 10: Blocked access protocols, $T = 30$.

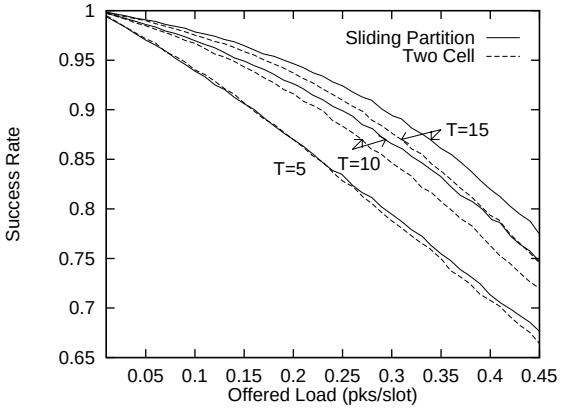

Figure 11: Blocked access protocols, $T = 5, 10, 15$.

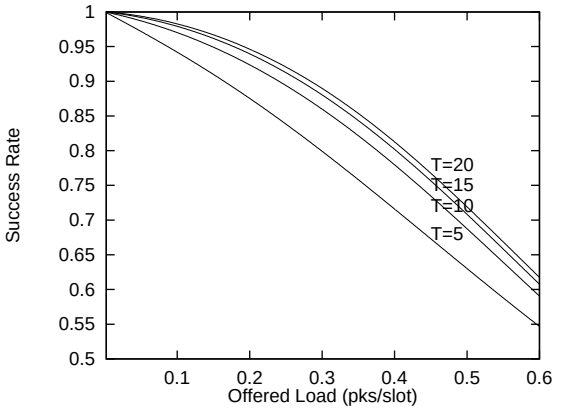

Figure 12: Blocked access sliding partition, $T = 5, 10, 15, 20$.

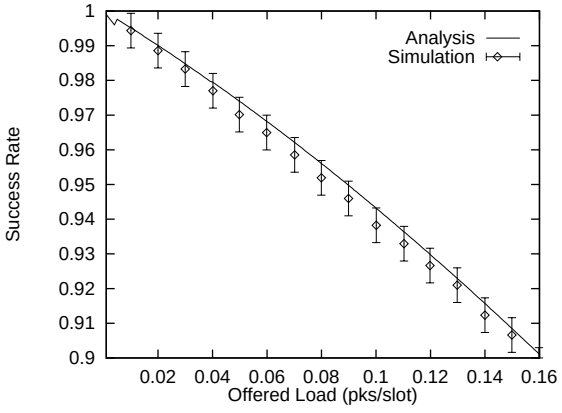

Figure 13: Free access sliding partition.

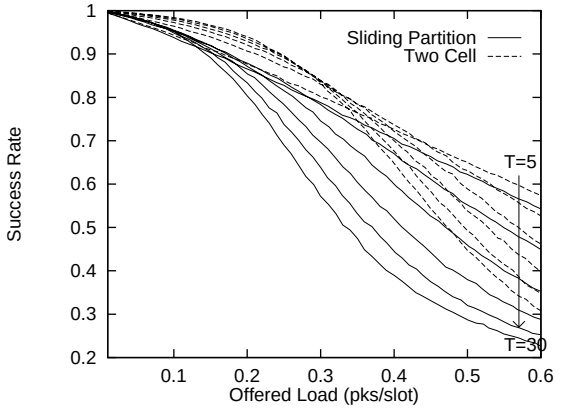

Figure 14: Free access sliding partition and two cell.

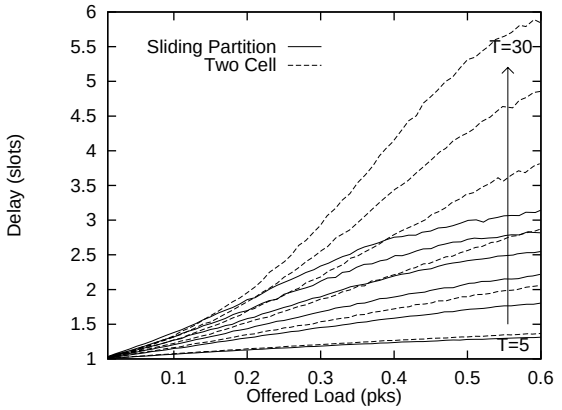

Figure 15: Free access sliding partition and two cell delay.

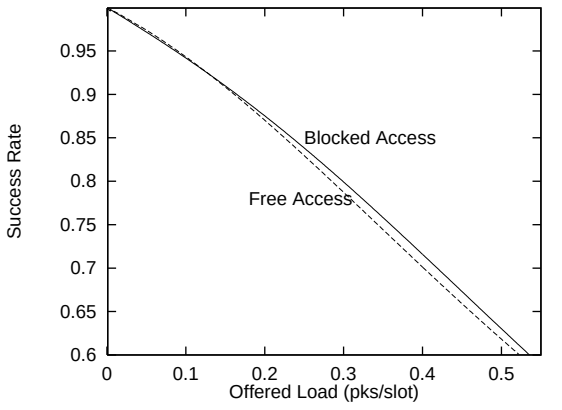

Figure 16: Blocked and free sliding partition, $T = 5$.

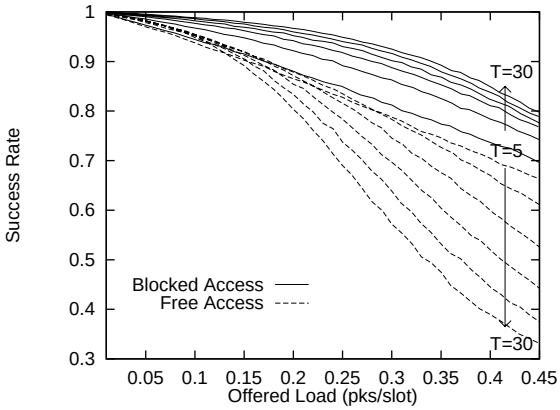

Figure 17: Blocked and free sliding partition.

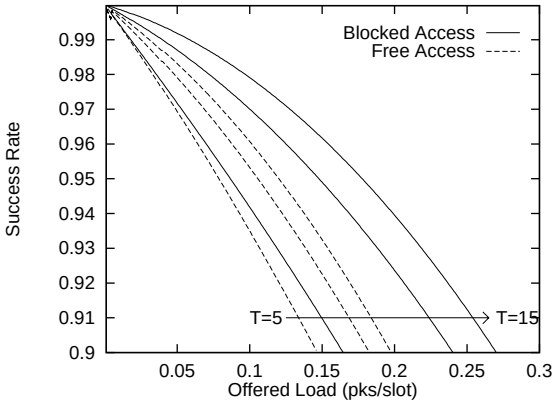

Figure 18: Blocked and free access sliding partition, $e_1 = .9$.

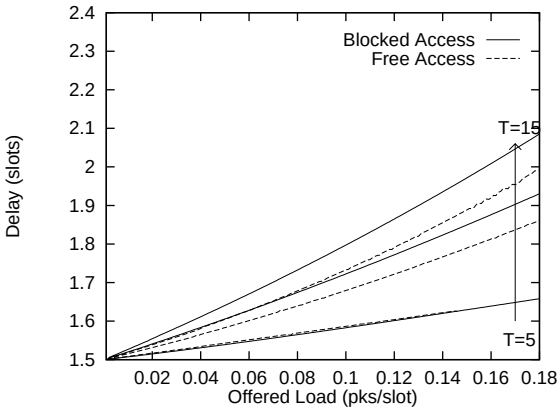

Figure 19: Blocked and free access sliding partition delay, $e_1 = .9$.