Analysis of a Soft Real-Time Random Access Protocol

Michael J. Markowski
US Army Research Laboratory, APG, MD 21005

Abstract

A communications network that is part of a soft real-time system may need to transmit messages within a bounded delay, but may allow some messages to miss this bound and be dropped within a maximum pre-specified rate of message loss. This paper presents a media access protocol for soft real-time systems implemented on a slotted radio channel with binary feedback. The protocol is based on the Gallager FCFS and Capetanakis splitting algorithms, but incorporates strict delay bounds using packet laxities. Also presented is an analytic model for this protocol by examining the probable lengths of the collision resolution intervals given the current lag at any time. 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.

Keywords. Random access algorithms, Real-time communication protocols, Time constrained communications, Multiple-access protocols.

1 Introduction

Communication on the battlefield often is made up of messages with deadlines. If a message is not delivered to its destination on time, the data is no longer useful. Not only can the lost information have an adverse effect on battlefield management, but current protocols do not take deadlines into account which can result in transmission of out of date messages, making poor use of the channel. Ideally, transmission scheduling will take into account both the needs of applications as well as individual message deadlines. On the battlefield, where bandwidth is limited, this is especially important.

Battlefield applications, since they directly interact with the world, often must respond to changes within some predetermined amount of time. Such systems are classified as *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: aeronautics, nuclear power control, etc. Soft real-time systems can afford some deadlines to slip. The battlefield can be modeled as a loosely coupled soft real-time system because each node functions independently but requires high level coordination between nodes. In such a communication network, quality-of-service constraints may require that a message be transmitted within a bounded delay, but that some messages may miss this bound and be dropped resulting in a certain amount of message loss. For instance, infrequently dropped voice packets don't interfere with a conversation, and similarly, tracking objects doesn't require processing of every return radar signal. In either the hard or soft case, however, all layers of the protocol stack must support the concept of deadlines. Ultimately, it is the lowest layer which provides timely access to the communications medium. In this paper, a media access protocol for soft real-time systems implemented on a slotted radio channel with binary feedback is considered.

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. Two application requirements are considered: minimum fraction of packets which must be successfully transmitted within the specified delay bound, and average delay experienced by a packet. Knowing these for various values of system parameters 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., the safest, lowest loss, and lowest throughput system.

The obvious disadvantage to using a general random access scheme for real-time communication is that the worst case channel access time is unbounded because of packet collisions. The problem is further 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 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 [MK85] 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 [RZ87] 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. [ZSR90] 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 [Arv91] where protocol operations are simulated and some worst case performance analysis is also presented. Paterakis et al. [PGPK89] present a simple protocol appropriate for some soft real-time systems and perform an in-depth analysis of it. Paterakis' technique of analysis is followed in this paper and extended to analyze a more complicated protocol which would also be suitable for use in a soft real-time environment. The protocol, the CG (Capetanakis-Gallager) protocol, is a combination of the Capetanakis splitting algorithm and the Gallager FCFS algorithms [Gal78], [BG92] with binary feedback and our addition of strict delay bounds.

2 Algorithm

In this protocol, 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 (only single-hop radio channels are considered in this analysis). Once a laxity is assigned to a packet, at each tick of the clock it is decremented and the packet discarded should it reach zero. The channel is accessed in a slotted manner with one slot long packets with binary (collision/non-collision) feedback. Collisions are resolved using a the technique described below.

2.1 CG CRA

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. During the collision resolution interval (CRI), a window of initial length Δ is used, and packets within it are ordered, left to right, from lowest laxity to highest. In the slot after a collision, only packets whose laxities fall within the window may transmit. If another collision occurs, the 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 [Gal78] with a few differences: packets in a window following a collision are ordered by laxity rather than by node arrival time, and feedback is binary rather than ternary. Additionally, each window is treated in the recursive manner of the Capetanakis algorithm rather than returning right subwindows to the waiting interval as in the Gallager algorithm. A further constraint is a bound T which is the maximum number of slots a CRI may comprise. If T slot times are exceeded, packets involved in the CRI are dropped, and the algorithm resets.

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. 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 the CRI means that only $T-d$ slots remain before a packet with initial laxity T must be transmitted or dropped. 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.

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. In addition, the position of the left edge of the window at time t is denoted 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 was 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 $h_t = R$.
   - If $f_{t-1} = c$, then $x_t = x_{t-1}, a_t = \frac{1}{2}a_{t-1}$, and $h_t = L$.

2.2 Paterakis CRA

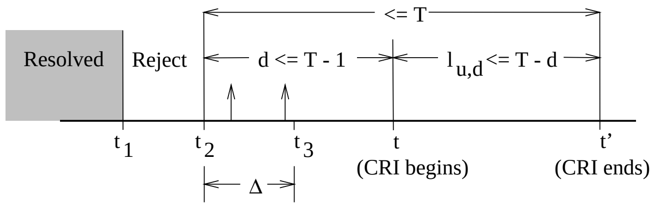
Paterakis et al. [PGPK89] analyze a time bounded CRA which does not take deadlines into account. Their protocol is elegantly simple. The protocol is described here and its performance is later compared to that of the CG CRA. Note that deadlines are not taken into account though the CRI length is bounded. 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 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, the algorithm be can more rigorously describe.

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

2. The transitions are:

[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.



t' corresponds to at least a laxity of 1.

Figure 1: Relation between some variables in algorithm.

- 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.

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. As indicated, the time t_2 corresponds to a laxity of one slot. 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 $t' = T - d$ slots. The variable $l_{u,d}$ will be defined in detail in subsequent sections but is simply the length of the CRI which begins at time t and ends at time t'.

3 Analysis

3.1 Notation

Our notation and analysis technique are based on the methodology presented by Paterakis et al. [PGPK89]. As mentioned, 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 CRI.
$z_{u,d}$:	Sum of delays after time t of packets in $n_{u,d}$.
$\psi_{u,d}$:	Sum of delays of the $n_{u,d}$ within the current window of length u.
$l_{u,d}$:	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 full CRI lengths. The length of a CRI is the number of slots it takes to move from a lag of $d = 1$ to the next $d = 1$ lag. Therefore, 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 the CRI.

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. Thus, the form of the final type of variable is denoted 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, the Poisson arrival process is used.

3.2 Model

In this analysis a simplifying assumption is made, namely that the initial laxities of all packets are identical and equal to T. Because of this, the laxity ordering of the packets is in fact the arrival time plus a constant laxity offset. Later simulation results are presented for the case where initial laxities are not identical.

Considering lags of 1 slot as regeneration points, a single CRI is one cycle of a regenerative stochastic process as pointed out by Georgiadis et al. [GMPK87]. Defining $Z = E\{\alpha_1\}$ as the number of packets successfully transmitted in a collision resolution interval (CRI), and $H = E\{h_1\}$ as the expected length of a CRI, 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

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

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

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

In the following sections, values for H, Z, and W are derived.

3.3 Recursions

In the following sections, let us denote by $E\{\cdot \mid k, B\}$ the expected value, given that k packets are in the current window, and the maximum length of the collision resolution interval is B. Let $P(\cdot \mid k, B)$ denote the conditional probability with given variables as above.

3.3.1 Probable Length of CRI

The probability of a CRI lasting l slots can be expressed recursively. Let $P(l \mid u, d)$ denote the 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, $T - d$.

$$P(l \mid u, d) = \sum_{k=0}^{\infty} P(l \mid k, T - \lceil d \rceil) e^{-\lambda u} \frac{(\lambda u)^k}{k!}. \tag{3}$$

Note that in the expression above, $P(l \mid u, d)$ is in terms of $P(l \mid k, B)$ where k is the number of packets, and B is the laxity bound. Initial conditions are listed below. In the general case of a collision, however, the algorithm states that the current window will be split in two. Therefore, the probability that a CRI is of length l given that a collision occurs and wastes one slot, imposes the constraint that the sum of the CRI lengths of the two subwindows must be $l-1$. Of the k packets involved in the collision, the probability of a packet being in one half versus the other is simply $\binom{k}{i}2^{-k}$. For $k \geq 2$ and $B \geq 2$, this is recursively expressed as

$$\begin{aligned} P(l \mid k, B) &= P(l-j \mid i, B-1) \\ &\quad \cdot P(j-1 \mid k-i, B-1-(l-j)), \qquad \text{with probability } 2^{-k}\tbinom{k}{i}. \end{aligned} \tag{4}$$

This is expressed, for $l > 1, B > 1$, as

$$P(l \mid k, B) = 2^{-k} \sum_{j=1}^{l-1} \sum_{i=0}^{k} \binom{k}{i} \left[P(l-j \mid i, B-1) \cdot P(j-1 \mid k-i, B-1-(l-j)) \right] \tag{5}$$

with the following initial conditions:

$$\begin{aligned} P(0 \mid k, 0) &= 1, & & \\ P(0 \mid k, B) &= 0, & & B > 0 \\ P(l \mid k, 0) &= 0, & & l > 0 \\ P(l \mid k, B) &= 0, & & l > B \\ P(1 \mid k, B) &= 1, & & B \geq 1, 0 \leq k \leq 1, \\ P(1 \mid k, 1) &= 1, & & k \geq 2, \\ P(1 \mid k, B) &= 0, & & B \geq 2, k \geq 2 \end{aligned}$$

Because the number of packets k and the laxity T are finite, the probabilities of CRI lengths can be determined by calculating a finite number of terms.

3.3.2 Expected Length of CRI

When the lag d is such that $\Delta < d$, the expected length of a collision resolution interval (CRI) is determined recursively as follows. With a lag of d, there are d slots to examine. However, only Δ slots at a time will be considered. As a result, the expected CRI length, h_d, with lag d is the number of slots it takes to examine Δ slots plus the expected CRI length of now only $d - \Delta$ remaining slots plus the additional number of slots it took to analyze that first window of length Δ. This, as well as when the lag is small and $d < \Delta$, is expressed as

$$h_d = \begin{cases} l_{d,d}, & l_{d,d} = 1, & 1 \leq d \leq \Delta \\ l_{d,d} + h_{l_{d,d}}, & 1 < l_{d,d} \leq T - \lceil d \rceil, & 1 \leq d \leq \Delta \\ l_{\Delta,d} + h_{d-\Delta+l_{\Delta,d}}, & & \Delta < d \leq T-1. \end{cases} \tag{6}$$

The expected value, $l_{u,d}$, of a CRI given length u to be examined and lag d is developed shortly. However, any given CRI is always of integer length. To determine the expected length of the remainder of the CRI after Δ slots are resolved, means giving consideration to each of the possible lengths of the CRI for the first Δ slots. The results of Section 3.3.1 are used for this keeping in mind that the CRI length can range between 1 slot and $T - d$ slots. So, taking expectations and denoting $H_d = E\{h_d\}$, yields

$$H_d = \begin{cases} E\{l_{d,d}\} + \sum\limits_{m=2}^{T-\lceil d \rceil} H_m P(m \mid d, d), & 1 \leq d \leq \Delta, \\ E\{l_{\Delta,d}\} + \sum\limits_{m=1}^{T-\lceil d \rceil} H_{d-\Delta+m} P(m \mid \Delta, d), & \Delta < d \leq T-1. \end{cases} \tag{7}$$

This system of equations is finite because of the bounded nature of the algorithm. See Paterakis [PGPK89] for further mathematical discussion. The system is represented by an $n \times n+1$ matrix where all elements are the conditional probabilities except for the rightmost column. That column is made up of (the negatives of) the expected values.

The expected values of CRI lengths are derived below. Equation 6 above is taken directly from Paterakis et al. [PGPK89] since it is a general description of any bounded CRA. Equation 7, the expected value, addresses the analytical details specific to the bounded CG CRA. So, given a length of u time units to resolve with a current lag of d time units, the expected length of that CRI is derived by first summing against all possible numbers of packet arrivals k in the window u and multiplying, respectively, by their probabilities of occurrence yielding

$$E\{l_{u,d}\} = \sum_{k=0}^{\infty} E\{\, l_{u,d} \mid k, T - \lceil d \rceil \,\} e^{-\lambda u} \frac{(\lambda u)^k}{k!}. \tag{8}$$

Considering each term of the summation individually, when it is given that the window u contains k packets and there is a laxity bound B, initial conditions are straightforward. If 0 or 1 packets are in the window, the CRI length will be one. When $k \geq 2$, the resulting collision takes one slot. Furthermore, though the k packets are each equally likely to be in either the left or right subwindow, the sum of the bounds of the two windows must be at most $B-1$ since the collision used one already. The left subwindow always takes at least one slot, and in some cases can even use all $B-1$ slots. Any remaining slots of the original $B-1$ can be used by the right subwindow. Because a given CRI, as opposed to the average case, can be only an integer number of slots between one and $T - \lceil d \rceil$, each possible length must be considered. The summation with index m Equation 9 loops through the range of possible CRI lengths, recurses on each, and multiplies each results by the probability as given in Equation 5.

Denoting $L_{k,B} = E\{\, l_{u,d} \mid k, B \,\}$, the recursive expression for the length of a CRI based on the operation of the algorithm is

$$L_{k,B} = \begin{cases} 1, & 0 \leq k \leq 1 \\ 1 + L_{i,B-1} + \sum\limits_{m=1}^{B-1} P(m \mid i, B-1) L_{k-i,B-1-m}, & \text{with probability } \binom{k}{i} 2^{-k},\ k > 1. \end{cases} \tag{9}$$

For initial conditions $L_{k,0} = 0$ for all k, and $L_{0,B} = L_{1,B} = 1$ for $B \geq 1$, then

$$L_{k,B} = 2^{-k} \sum_{i=0}^{k} \binom{k}{i} \left[1 + L_{i,B-1} + \sum_{m=1}^{B-1} P(m \mid i, B-1) L_{k-i,B-1-m} \right] \tag{10}$$

Note that the bounded nature of the algorithm guarantees a finite expansion of terms.

3.3.3 Packets Successfully Transmitted

The next value to be determined is the expected number of *successful* transmissions between times with a lag of 1, that is, a single CRI. This requires knowledge of the expected length of a CRI as developed in the preceding section. Given a current lag of d slots with $d \geq \Delta$, there will be some number of packets transmitted during examination of the current window. This examination removes Δ slots from the lag d but introduces $l_{\Delta,d}$ more slots. Thus, the total expected number of transmitted packets would be $n_{\Delta,d} + \alpha_{d-\Delta+l_{\Delta,d}}$. To also account for the situation where $d < \Delta$, meaning the lag is very near the current moment in time, the following expression is more general:

$$\alpha_d = \begin{cases} n_{d,d} & l_{d,d} = 1, & 1 \le d \le \Delta, \\ n_{d,d} + \alpha_{l_{d,d}}, & 1 < l_{d,d} \le T - \lceil d \rceil, & 1 \le d \le \Delta, \\ n_{\Delta,d} + \alpha_{d-\Delta+l_{\Delta,d}}, & & \Delta < d \le T-1. \end{cases} \tag{11}$$

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

$$A_d = \begin{cases} E\{n_{d,d}\} + \sum\limits_{m=2}^{T-\lceil d \rceil} A_m P(m \mid d,d), & 1 \le d \le \Delta, \\ E\{n_{\Delta,d}\} + \sum\limits_{m=1}^{T-\lceil d \rceil} A_{d-\Delta+m} P(m \mid \Delta, d), & \Delta < d \le T-1 \end{cases} \tag{12}$$

Finally, the number of packets transmitted during a single CRI is

$$E\{n_{u,d}\} = \sum_{k=0}^{\infty} E\{\, n_{u,d} \mid k, T - \lceil d \rceil \,\} e^{-\lambda u} \frac{(\lambda u)^k}{k!} \tag{13}$$

Denoting $N_{k,B} = E\{\, n_{u,d} \mid k, B \,\}$, to subsequently calculate the expected number of packets successfully transmitted during the period that it takes to move from a lag of d to a lag of 1, the algorithm induces the relationship below. The left subwindow can use up to $B-1$ slots to transmit the i packets in its window. In a manner identical to that for obtaining the expected CRI length, each possible CRI length is considered because however many slots a given left window CRI requires, the right window's CRI has that many fewer slots. Because individual cases are being considered, the expected value of a CRI length cannot be used as a subscript on the last factor of the last term.

$$N_{k,B} = \begin{cases} 0, & k = 0, B \ge 1, \\ 1, & k = 1, B \ge 1, \\ N_{i,B-1} + \sum\limits_{m=1}^{B-1} P(m \mid i, B-1) N_{k-i,B-1-m}, & k > 1, \text{with probability } \binom{k}{i} 2^{-k} \end{cases} \tag{14}$$

For initial conditions $N_{k,0} = 0$ for all k, and $N_{0,B} = 0, N_{1,B} = 1$ for $B > 1$, then

$$N_{k,B} = 2^{-k} \sum_{i=0}^{k} \binom{k}{i} \left[N_{i,B-1} + \sum_{m=1}^{B-1} P(m \mid i, B-1) N_{k-i,B-1-m} \right]. \tag{15}$$

As before, this is a finite expansion because of the properties of the CRA.

3.3.4 Delay Experienced by Packets Successfully Transmitted

Determining the cumulative delay experienced by successfully transmitted packets is perhaps the most complex of the calculations introduced thus far. The cumulative delay is the sum of the delays experienced within the window u, the delays experienced by the current lag, and the delays during the CRI. That is,

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

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

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

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

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

Therefore, the cumulative delay of packets transmitted during a single CRI is

$$E\{z_{u,d}\} = \sum_{k=0}^{\infty} E\{\, z_{u,d} \mid k, T - \lceil d \rceil \,\} e^{-\lambda u} \frac{(\lambda u)^k}{k!} \tag{18}$$

The initial conditions are easy: a window with no packets experiences no delay, and a window with one packet experiences a delay of one. It is more involved, however, when a collision has occurred. After splitting, each packet in the left window experiences an additional delay of one slot due to the collision. The packets in the right window, though, will experience the *total* delay of the left window's resolution in addition to a similar one slot delay/packet for the initial collision which caused the split. Similar to the development of previous expressions, the number of slots used by the left subwindow are subtracted from the maximum available slots for use by the right subwindow. This is expressed mathematically as

$$Z_{k,B} = \begin{cases} 0, & k = 0, \\ 1, & k = 1, \\ (N_{i,B-1} + Z_{i,B-1}) + \sum_{m=1}^{B-1} P(m \mid i, B-1) & k \geq 2, \text{with} \\ \quad [(m+1) N_{k-i,B-1-m} + Z_{k-i,B-1-m}], & \text{probability } \binom{k}{i} 2^{-k}. \end{cases} \tag{19}$$

With the initial conditions above, i.e., $Z_{k,0} = 0$, $Z_{0,B} = 0$, and $Z_{1,B} = 1$,

$$Z_{k,B} = \sum_{i=0}^{k} \binom{k}{i} \left\{ N_{i,B-1} + Z_{i,B-1} + \sum_{m=1}^{B-1} P(m \mid i, B-1) [(m+1) N_{k-i,B-1-m} + Z_{k-i,B-1-m}] \right\}. \tag{20}$$

This, again, is a finite expansion.

4 Evaluation

In real-time systems it is important that the application's requirements be met by all layers of the protocol stack. As previously 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. Similarly, while there is a maximum laxity value, it is also desirable to design to a significantly smaller average delay. Continuing to follow the notation of Paterakis et al. [PGPK89], these design parameters are called e_1, for the fraction of packets transmitted successfully, and e_2, for the delay experienced by an average packet.

While it is the application which drives the choices for maximum laxity T, and for e_1 and e_2, the initial 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

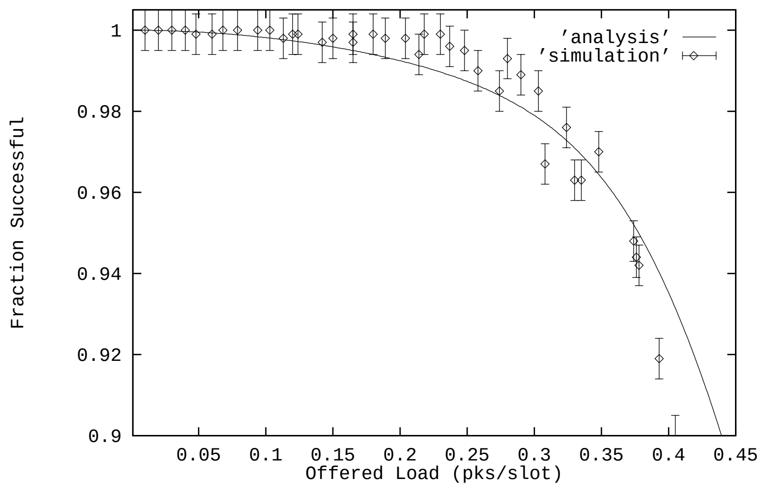


Figure 2: Comparison of Simulated and Analytical Results for Fraction Successful when $\Delta = 3.0$ and $T = 20$

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 reduces to:

$$\lambda^*_{T,e_1} = \sup(\lambda \,:\, \rho_T(\Delta, \lambda) \geq e_1). \tag{21}$$

Because of space considerations, results are presented only for $\Delta = 3$ though data has been generated for various Δ values which, through observation, encompass the maximum throughputs.

A simulation study of this protocol was also conducted using Opnet, a network simulation package. The simulations accurately modeled the overall system functionality including the channel behavior, but did not explicitly model individual stations. Each simulation was run with 95% confidence intervals that the fraction of successful transmissions ρ was within ± 0.005 of the steady state value. Input traffic rates, in packets/slot, ranged from 0.050 to 0.600 in increments of 0.005. When window size $\Delta = 3.0$ and laxity $T = 20$, Figures 2 and 3 show overlaid graphs of analytical and simulated results. Because of the close correlation, subsequent graphs do not include simulated data for easier readability. Simulation results, however, have been obtained for all presented graphs and show similar close correlations.

In Figure 4, three curves are graphed showing the maximum sustainable input traffic rates attainable when window size $\Delta = 3.0$ and the indicated values of e_1 are used. It can be seen from these figures that the maximum input rate increases as initial laxity is increased and also as the threshold for success rate is decreased.

Given a minimum success fraction, an additional constraint can be added to the problem posed in Equation 21 above so that an average delay is met in addition to meeting the maximum laxity T, yielding:

$$\lambda^*_{T,e_1,e_2} = \sup(\lambda \,:\, \rho_T(\Delta, \lambda) \geq e_1, D_T(\Delta, \lambda) \leq e_2). \tag{22}$$

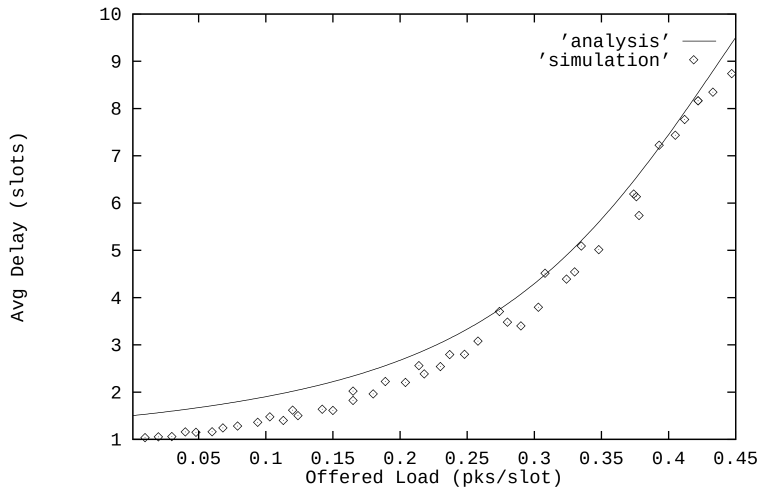


Figure 3: Comparison of Simulated and Analytical Results for Average Delay when $\Delta = 3.0$ and $T = 20$

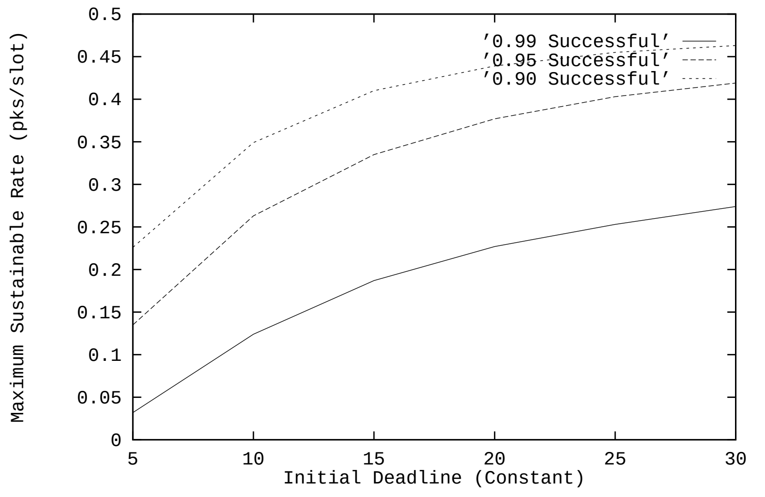


Figure 4: Maximum Input Traffic Rate vs Initial Laxity with Minimum Success Constraints.

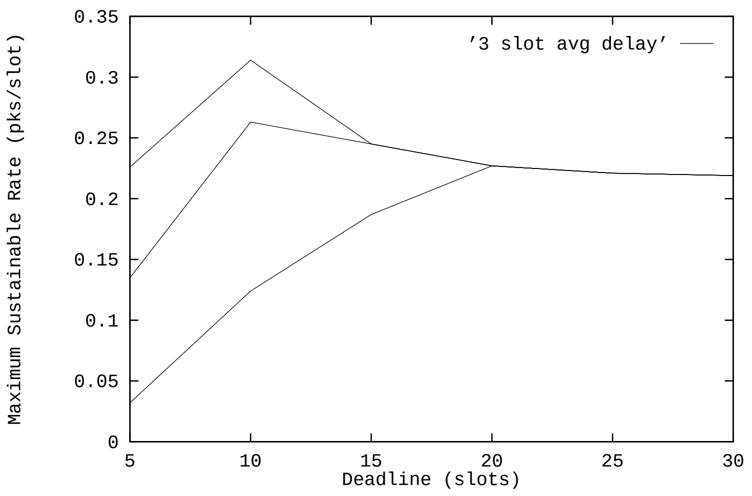


Figure 5: Maximum Input Traffic Rate vs Initial Laxity with Minimum Success Constraints $e_1 = 0.99, 0.95, 0.90$, Average Delay Constraint $e_2 = 3$.

When an average delay of $e_2 = 3$ slots is met, the results are shown in Figures 5. Similar graphs are shown for 5 and 7 slots in Figures 6 and 7 respectively. All three figures represent performance with window $\Delta = 3.0$. It is interesting to note that with the new constraint e_2, the curves on each of these graphs converge whereupon the function slowly decreases. This is expected when increasing traffic rate but not relaxing the delay constraint.

It could be expected that because the CG CRA makes decisions based on packet deadlines that it would outperform the Paterakis CRA. However, the CG CRA has several shortcomings when used in an environment where packets have fixed initial laxities. Packets with very close laxity values cannot be quickly separated, so that the CRA must continually split until the windows are small enough to separate the packets. Not only is this time consuming, but the CRA must then recursively double the window lengths until again reaching the original length, Δ. Because of the time bounded nature of the CRA, the CRI often expires before packets with similar laxity values can be transmitted. On the other hand, the coin flipping of the Paterakis CRA introduces randomness at each step of a CRI making it more likely that packets will be more quickly separated into two groups. Figure 8 illustrates the performance differences. It shows two sets of three curves; one set is the performance of the Paterakis and the other of the CG CRA representing maximum sustainable offered loads for $e_1 = 0.99, 0.95, 0.90$. The Paterakis CRA always outperforms the CG CRA, even at low traffic rates.

There are often cases where soft real-time systems generate packets with variability in initial laxities unlike the simplying assumption made in this analysis. Using the simulation model to study such a situation, each packet was assigned an initial laxity uniformly distributed in the range $[2, T]$. Figure 9 presents these results, showing the maximum input rate achieved as the average laxity is increased. Because of the variability, the maximum input rates are now smaller than the rates achieved for constant laxities. Figure 10 shows the average delays experienced for the corresponding success rates. The results shown represent performance for a window size of 3.0 slots. There are no observable differences when a constraint of e_2 is imposed.

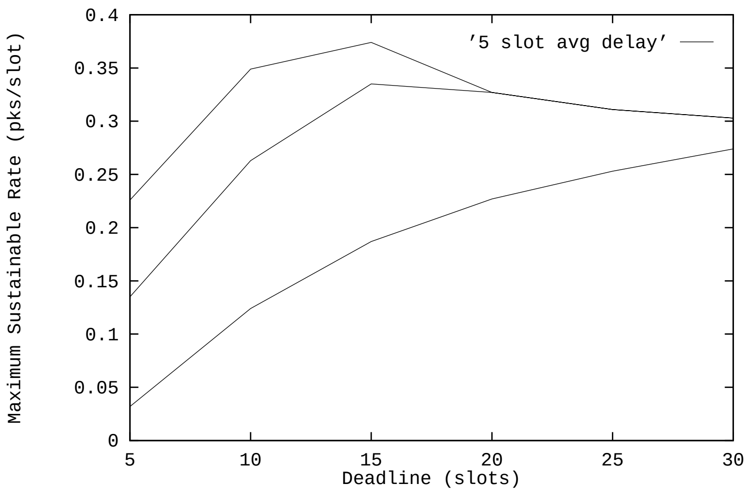


Figure 6: Maximum Input Traffic Rate vs Initial Laxity with Minimum Success Constraints $e_1 = 0.99, 0.95, 0.90$, Average Delay Constraint $e_2 = 5$.

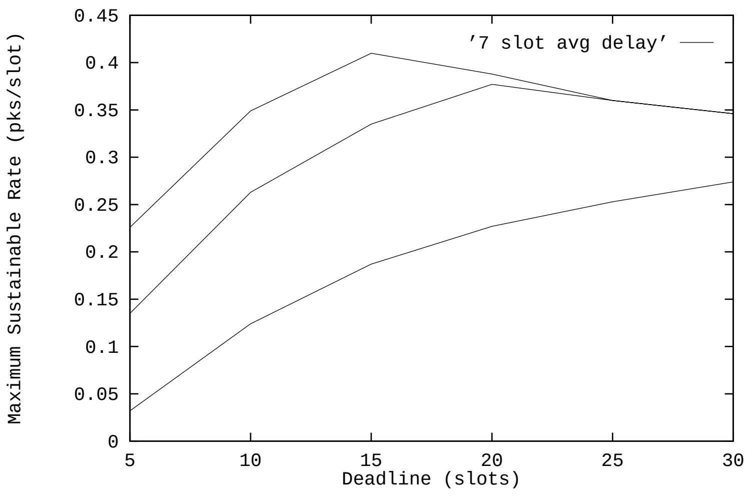


Figure 7: Maximum Input Traffic Rate vs Initial Laxity with Minimum Success Constraints $e_1 = 0.99, 0.95, 0.90$, Average Delay Constraint $e_2 = 7$.

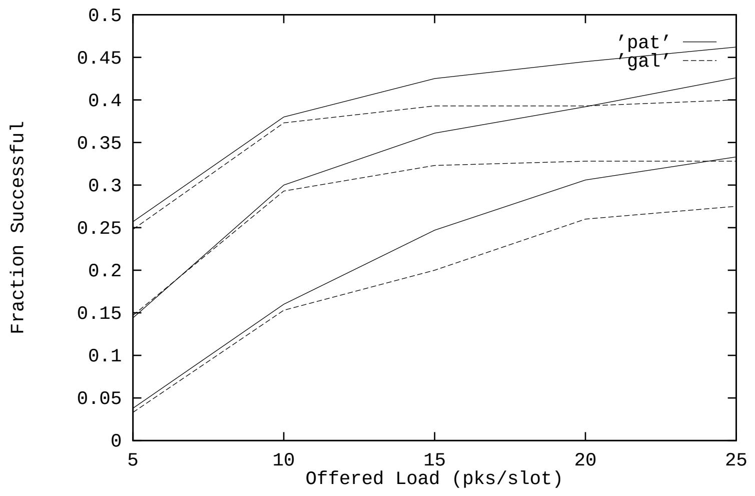


Figure 8: Maximum Input Traffic Rates vs Initial Laxity with Minimum Success Constraints $e_1 = 0.99, 0.95, 0.90$.

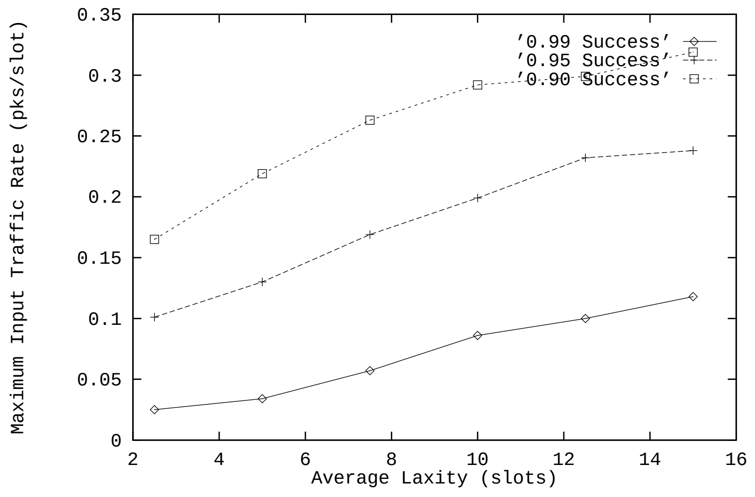


Figure 9: Maximum Input Traffic Rate vs Average Laxity with Minimum Success Constraints e_1 = 0.99, 0.95, and 0.90.

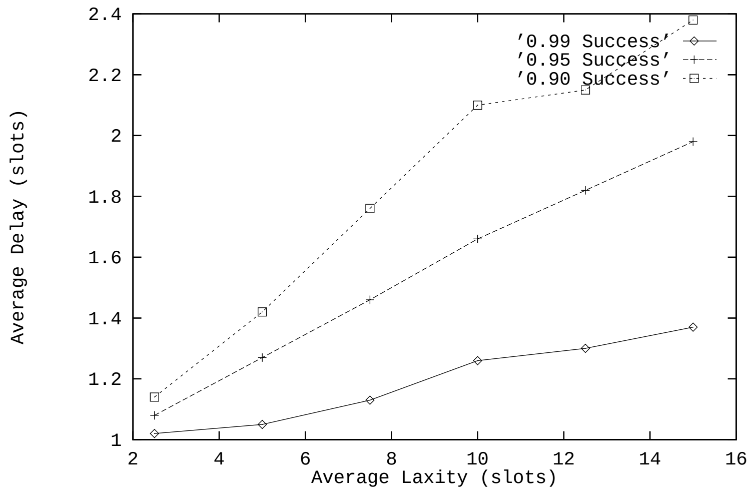


Figure 10: Maximum Input Traffic Rate vs Average Laxity with Minimum Success Constraints $e_1 = 0.99$, 0.95, and 0.90.

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 specified laxity, and by using a minimum rate of successful transmission, the analysis presented has shown that window-splitting protocols can be modified to work successfully in these environments. An analytic model of such a protocol has been presented which can be used to determine proper operating parameters for specified quality-of-service constraints.

A comparison of the CG CRA with the Paterakis CRA yielded surprising results. It showed that even though the Paterakis CRA does not take deadlines into account, its extreme simplicity allows it to perform better than the CG CRA. It seems likely that a variant of the Gallager CRA would perform similarly to the Paterakis CRA. It would also be interesting to see if a hybrid protocol can be developed including features of both the Gallager and Paterakis CRAs. Such a protocol could be of significant benefit to battlefield communication. The protocol could be tailored to the needs of applications while still basing transmissions on message deadlines.

References

[Arv91] K. Arvind. *Protocols for Distributed Real-Time Systems.* PhD thesis, University of Massachusetts at Amherst, 1991.

[BG92] Dimitri Bertsekas and Robert Gallager. *Data Networks.* Prentice Hall, second edition, 1992.

[Gal78] R. G. Gallager. Conflict resolution in random access broadcast networks. *Proceedings of the AFOSR Workshop in Communications Theory Applications*, pages 74–76, 1978.

[GMPK87] 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.

[MK85] 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.

[PGPK89] 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.

[RZ87] K. Ramamritham and W. Zhao. Virtual time CSMA protocols for hard real-time communication. *IEEE Transactions on Software Engineering*, 13(8):938–952, 1987.

[ZSR90] 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.