

# 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

## ABSTRACT

*Addressing the problem of timely packet transmission in a wireless soft real-time system such as one would find on the battlefield, we present five splitting protocols that take packet deadlines into account. Three are blocked access and two are free access algorithms. Mathematical models of the algorithms are developed, and compared with simulations. We show, as expected, that the blocked access algorithms usually offer higher success rates than the free access versions. Of the two best performing blocked access protocols, performance differences are slight on a lightly loaded channel. Under heavy load, however, one is shown to have better performance, and thus would be the best choice for implementation.*

**Keywords— Real-time communication protocols, wireless soft real-time MAC protocols, real-time splitting protocols.**

## 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 military battlefield communications or coordination of search and rescue vehicles.

In support of the problem of timely battlefield com-

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

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

munications, we consider here the specific problem of a soft real-time system implemented on a wireless network. 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 valid; 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.

## BACKGROUND

The most popular random access MAC techniques are Aloha [1] and Ethernet 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 and Kleinrock [2] 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 and Zhao [3] 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.* [4] 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. Algorithmically less complex CSMA-CD splitting protocols were presented for both hard and soft real-time systems by Arvind [5], where protocol operations were simulated and some straightforward 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 [6]. Hers is a hard real-time variant of the soft real-time protocol presented and analyzed by Paterakis *et al.* [7] described below.

We combine the use of laxity, *i.e.*, time until the packet expires, as the splitting variable with the splitting methodologies of traditional algorithms to create new splitting protocols for real-time use.

## SYSTEM MODEL

The system studied is an infinite population of similar users, each with a queue of length one, sharing a common channel. 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. 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 CRAs

After some initial collision occurs initiating the CRA, no other stations may contend for the channel until CRI has completed. The longer a CRI is, the more likely it will be that more than one station has a packet to transmit. To avoid almost certain collisions, a window is used that allows only a subset of waiting packets to be transmitted. Its size is chosen so that it is small enough to effectively reduce the probability of collision while being large to maximize the probability of not being empty.

**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. When a collision occurs, the packets enabled by the arrival time based window are reordered in a 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. As a result, a splitting tree of arbitrary depth, *i.e.*, a large number of subwindows, can exist. 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. While clearly impractical for implementation, it is interesting to consider this CRA for for the sake of comparison to other real-time CRAs.

**Blocked Access Sliding Partition Real-Time CRA.** This CRA differs from the previous 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 midpoint of the current left half. In this way, there are never more than two windows during a CRI. After the left half transmission is a non-collision, the CRA is applied the right half. Due to the nature of the CRA, two consecutive non-collisions indicate the end of a CRI.

**Blocked Access Real-Time Two Cell.** Paterakis *et al.* [7] developed a CRA that uses random splitting as well as an arrival time based sliding window. Again,

there are always only two subsets of split users. Upon collision, nodes flip a coin. On tails, say, nodes do not again transmit until a non-collision feedback is observed. The nodes who flipped heads, however, immediately transmit. This has the advantages of the sliding partition CRA—at most two windows active—with the advantage that the splitting is independent of the arrival process. Like the previous CRA, two consecutive non-collisions signal the CRI completion.

### BLOCKED ACCESS EVALUATION

A soft real-time system can be characterized by at least the following: traffic intensity  $\lambda$ , laxity range  $[2, T]$ , minimum acceptable success ratio  $e_1$ , and average delay desired  $e_2$ . Ideally, a protocol would adapt to these values as they change. 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 maximum laxity  $T$  and success rate  $e_1$ , the initial time window width  $\Delta$  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 given values of  $\Delta$ , however, the problem is simple to solve numerically and, as pointed out by Paterakis *et al.* [7], reduces to:

$$\lambda_{T,e_1}^* = \sup(\lambda : \rho_T(\Delta, \lambda) \geq e_1). \quad (1)$$

We present results only for  $\Delta = 2.5$  though data has been generated for various  $\Delta$  values that, through observation, encompass the maximum throughputs. We also conducted simulation studies of each protocol using Opnet [8], a network simulation package. The simulations model the systems as infinite user populations, thus offering more conservative performance results than the finite user case [9]. Each simulation was run with 95% confidence intervals that the fraction of successful transmissions  $\rho$  was within  $\pm 0.005$  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.

Figure 1 graphs analytical and simulation results for  $\lambda_{10,0.9}^*$ . Figure 2 graphs a similar comparison, but for  $\lambda_{5,0.9}^*$ ,  $\lambda_{10,0.9}^*$ , and  $\lambda_{15,0.9}^*$ . Error bars are not shown since the condition of reaching steady state was made by monitoring  $\rho$ , 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

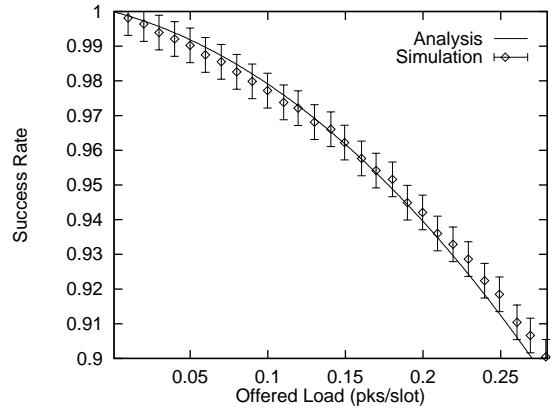


Fig. 1. Blocked access sliding partition.

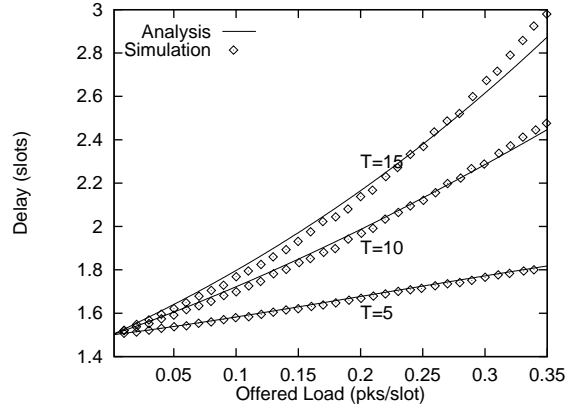


Fig. 2. Blocked access sliding partition delay.

readability. With simulation results, error bars will not be shown but in all cases are based on the  $\pm 0.005$  tolerance of  $\rho$ .

For maximum initial laxity  $T = 10$ , analytical results for the three protocols are compared in Figure 3. It is interesting that the fully recursive CRA outperforms the other others. In the original ternary feedback versions, and even when windowing was added to the fully recursive CRA, it algorithm performed worst. In the soft real-time environment, it appears that the finer window splitting of that CRA offers some advantage. However, as  $T$  increases, the advantage is lost. The fully recursive CRA performs progressively worse until the other protocols far overtake it in success rate. While splitting finely allows perhaps quicker isolation of contending packets, it is then necessary for the recursion to spend a 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 as load increases than the random splitting of the two cell

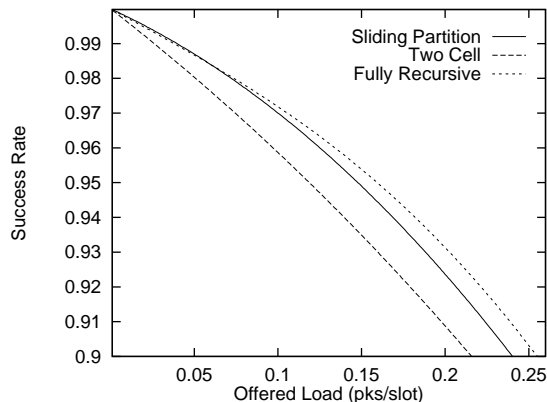


Fig. 3. Blocked access protocols,  $T = 10$ .

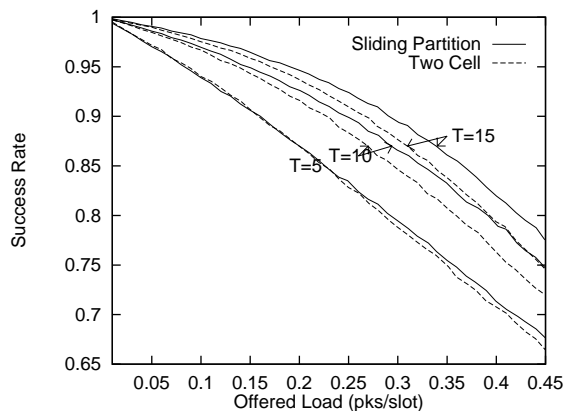


Fig. 4. Blocked access protocols,  $T = 5, 10, 15$ .

CRA. As laxity range increases, the performance differences between the two become more significant. This is shown in Figure 4. Furthermore, as seen in Figure 5, the CRA is seen to approach some operational maximum as the value of  $T$  is increased. The corresponding delay curve is graphed in Figure 2. From this, it appears that results for, say,  $T = 30$  could be used to approximate still wider laxity ranges.

### 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. Therefore, a time based window is not used in these CRAs.

**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.

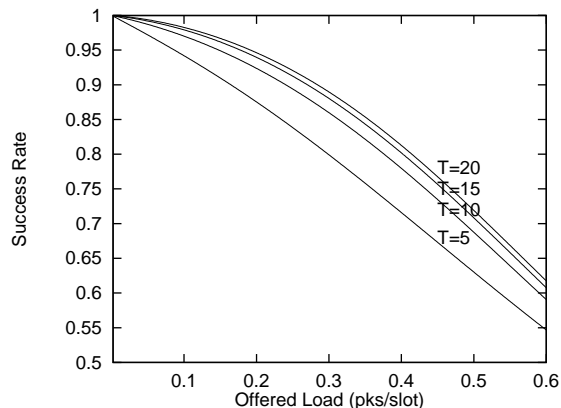


Fig. 5. Blocked access sliding partition,  $T = 5-20$ .

Higher laxity packets, though, must wait for the lower laxity ones to resolve their contention. 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.

**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 remaining in the waiting group after a collision.

### EVALUATION

In the case of traditional, non real-time 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 of  $[2, 5]$ , it turns out the the two types of algorithms perform nearly identically. However, when higher ranges are compared, the performances quickly diverge. Figure 6 graphs success rates for initial laxities in  $[2, T]$ . We can see that as  $T$  increases, the blocked access algorithms perform better, whereas the free access version's performance worsens.

### 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

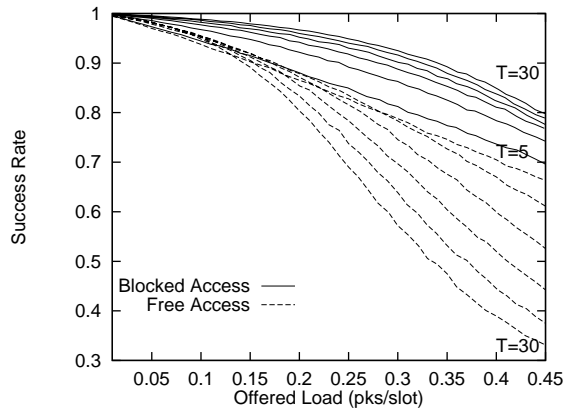


Fig. 6. Blocked and free sliding partition.

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 real-time environments.

While there are certain situations where the blocked access fully recursive CRA outperforms the other blocked access CRAs, it so happens that the fully recursive algorithm is not practically implementable as previously mentioned. Both the sliding partition and two cell CRAs, however, are practically implementable. When the CRA is viewed as the channel server for the infinite user population, it turns out that the service rate, *i.e.*, packet delay, is small enough that the performance difference between the sliding partition and two cell CRAs is never more than 5%. For implementation, the best choice would be the sliding partition CRA. It always outperforms the two cell CRA, though only by a little when lightly loaded (where all splitting algorithms perform similarly), but at moderate to high loads, the difference becomes significant.

Comparing the blocked access CRAs to their free access counterparts, we find, not surprisingly, that the blocked access algorithms perform best. However, if the application's requirements are met by a free access CRA, the free access protocols are easier to implement. They only require that a node listen to the feedback for the slot of its arrival and then decide which subwindow to join. In contrast, the blocked access versions demand that any new node wait until two consecutive non-collision

slots appear. Of course, if the laxity range is small, we find that both blocked and free versions perform almost identically.

It is possible that these protocols can be modified for still better performance. Currently, in the blocked access CRAs, the sliding time window moves forward only after all packets in the window have been transmitted. Gallager has shown [10], however, that after the first non-collision, higher throughput (in FCFS servicing) is obtained by immediately sliding forward the window. It might be possible to similarly modify these protocols. To improve performance of the free access CRAs, it would be interesting to see if the addition of more complex decision making on which subwindow is joined by new arrivals, could improve their performance. And finally, this paper has focused on soft real-time transmissions in an error free channel. It would be useful to have a distributed MAC algorithm that combines hard, soft, and non real-time transmissions and to study it in a non-ideal environment.<sup>1</sup>

## REFERENCES

- [1] N. Abramson. Another alternative for computer communications. In *Proceedings of the Fall Joint Computer Conference, AFIPS Conference*, volume 37, pages 281–285, 1970.
- [2] 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.
- [3] K. Ramamritham and W. Zhao. Virtual time CSMA protocols for hard real-time communication. *IEEE Transactions on Software Engineering*, 13(8):938–952, 1987.
- [4] 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.
- [5] K. Arvind. *Protocols for Distributed Real-Time Systems*. PhD thesis, University of Massachusetts, Amherst, MA, 1991.
- [6] P. Papantoni-Kazakos. Multiple-access algorithms for a system with mixed traffic: High and low priority. *IEEE Transactions on Communications*, 40(3):541–555, March 1992.
- [7] M. Paterakis, L. Georgiadis, and P. Papantoni-Kazakos. Full sensing window random-access algorithm for messages with strict delay constraints. *Algorithmica*, 4:318–328, 1989.
- [8] Inc. Mil 3. *Opnet modeler*, v3.0, 1996. Washington, DC.
- [9] M. Paterakis, L. Georgiadis, and P. Papatoni-Kazakos. On the relation between the finite and the infinite population models for a class of RAA's. *IEEE Transactions on Communications*, COM-35(11):1239–1240, November 1987.
- [10] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, second edition, 1992.

<sup>1</sup>The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government.