To appear in Proceedings IEEE GLOBECOM '97, Phoenix, AZ (Nov. 1997).

Real-Time Wireless Communication using 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 packet transmission in a wireless soft real-time system, we present five splitting protocols that take packet deadlines into account. We show, as in the case of non real-time splitting algorithms, that blocked access versions offer higher success rates than free access ones. Of the two best performing blocked access protocols, performance under moderate to heavy loads further shows the superiority of the Sliding Partition CRA over the Two Cell CRA.

I. 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.

Wireless channels are not typically considered for the transport of real-time data due to their relatively high error rate. However, we consider here the specific problem of a soft real-time system implemented on a wireless network for use in certain environments, *e.g.*, military battlefield communications or coordination of search and rescue vehicles, where wireless communication is the only option.

II. Background

The shortcomings of traditional random access protocols in the real-time environment is clear: collisions result in a random transmission order of involved packets, offering potentially unbounded access times. Aside from straightforward techniques such as priority classes or transmission scheduling based on *a priori* knowledge, approaches to limiting this shortcoming for random access, time constrained communication have included the use of virtual time clocks and splitting techniques.

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.

In CSMA-CD systems, virtual time clocks were first considered by Molle and Kleinrock [1] and based on message arrival time. The method was adapted by Ramamritham and Zhao [2] 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.* [3] 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 the wire based CSMA-CD. Algorithmically less complex than Zhao *et al.*'s CSMA-CD splitting protocols were presented for both hard and soft real-time systems by Arvind [4]. A similar protocol for wireless transmission of hard and non real-time data was analyzed in detail by Papantoni-Kazakos [5]. Paterakis *et al.* [6] presented and analyzed a simple protocol appropriate for limited types of soft real-time systems where all packets initially have the same deadline. While they studied a specific protocol's performance for initial deadlines of up to 30 slots, Panwar *et al.* [7] used the value iteration method to find the optimal splitting algorithm for fixed initial deadlines of up to 4 slots. In [8], 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 [9]. Here, we use a more efficient analytical technique for varying initial deadlines, and apply it to the analysis of three blocked access protocols. The technique is further extended to study performance of the two free access counterparts of the blocked access algorithms.

III. 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. Only when the collision is resolved and all packets have been either transmitted or dropped, 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. A characteristic common to all CRAs analyzed here is that packets are never delivered late. They are either transmitted in a timely manner or dropped.

IV. Blocked Access CRAs

By the nature of blocked access CRAs (Collision Resolution Algorithms), the time when contending packets entered the system is known; it is at the first collision, the beginning of the CRI (Collision Resolution Interval). At any time during the CRI, an initial collision that happened some while ago is being resolved. This means that there is lag of some number of slots between "now" and the actual arrival times of packets that start a 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 [10]. With large lags, it is likely that when a CRI completes, it will be immediately followed by a collision. To avoid this, a window optimized to be some width, in slots, is used so that it is much smaller than large lags and hence less likely to encompass more than one packet. A CRI ends when all involved packets have been transmitted or dropped. Because of space considerations, analysis of the regenerative stochastic systems is not presented here. Detailed presentation, however, can be found in [11].

A. 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. After a collision, the packets within the time based window are reordered by laxity in a new window encompassing the full laxity range. We assume throughout that laxities are uniformly distributed in the interval $[2, T]$, in units of slots. This 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 at a given point in time. 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 impractical for implementation, it is interesting to consider this CRA for the sake of comparison to other real-time CRAs.

B. Blocked Access Sliding Partition Real-Time CRA

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, there are never more than two windows during a CRI. And after the left half transmission is a non-collision, the CRA is applied to the right half. Due to the nature of the algorithm, two consecutive non-collisions indicate the end of a CRI.

C. Blocked Access Real-Time Two Cell

Paterakis *et al.* [6] developed a CRA that uses random splitting but incorporates windowing as in the Sliding Partition CRA. Upon collision, nodes flip a coin. On tails, say, nodes do not again transmit until a non-collision feedback is observed. The nodes that flipped heads, however, immediately transmit. This has the advantages of the Sliding Partition CRA—at most two windows active—with the additional advantage that the splitting is independent of the arrival process. The disadvantage is that real-time properties are not directly considered. 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.

V. Blocked Access Evaluation

A soft real-time system can be characterized by at least traffic rate λ, laxity range $[2, T]$ and minimum acceptable success ratio e_1. 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.

In addition to developing analytic techniques characterizing the algorithms, we also conducted corresponding simulation studies of each protocol using Opnet [12], a network simulation package. The simulations model the systems as infinite user populations, thus offering more conservative performance results than the finite user case [13]. 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 units of packets/slot, ranged from 0.050 to 0.600 in increments of 0.01.

For the blocked access Sliding Partition CRA, Figure 1 graphs success rate results, in packets/slot, when $T = 10$ and $e_1 = 0.9$. Figure 2 graphs a similar comparison but for delay when $e_1 = 0.9$ and T is $5, 10,$ and 15.

Figure 3 offers a performance comparison of the three protocols. It is interesting that for smaller T values, the Fully Recursive CRA outperforms the others. In the original non real-time ternary feedback versions, and even when windowing was added to the Fully Recursive CRA, it performs worst. In the soft real-time environment, it appears that the finer window splitting of that CRA, in some cases, is an advantage. However, when $T = 30$, the advantage is lost as illustrated in Figure 4. 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, 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 5. Furthermore, as seen in Figure 6, 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.

VI. 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. The analysis, therefore, is complicated by the more involved arrival process, yet simplified by the removal of a sliding time based window. Due to space considerations, free access analytical equations are not presented.

Interestingly, the free access versions of the Fully Recursive and Sliding Partition are identical. To always allow lower laxity packets to contend, the multiple low laxity windows of the blocked access case, are now a single free access low laxity window. The Two Cell CRA allows new entries at all times, but if a collision occurred in the previous slot, they automatically join the waiting cell. Therefore, neither of the free access CRAs are purely free access, since mandatory waiting is required for some newly arrived packets. This is unavoidable, though, when packets are dealt with in a real-time manner.

VII. Evaluation

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 7 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.

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 8 compares the performance of the Sliding Partition blocked and free access versions when $T = 5$. The difference is slight. However, as wider laxity ranges are used, the performances quickly diverge. Figure 9 graphs success rates for initial laxities in $[2, T]$. We can see that as T increases, whereas the free access version's performance worsens, the blocked access algorithms perform comparatively better. The more practical range of success rates for $e_1 = 0.9$ is illustrated in Figure 10. For both sets of analytically derived curves, values of $T = 5, 10$, and 15 are graphed left to right. While both algorithms show increased success rates when T is increased, it's easily seen that even in this restricted range of success rates, the blocked access CRAs perform best.

VIII. 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 real-time environments. We have presented several analytic models for such protocols that can be used to determine proper operating parameters for specified quality-of-service constraints.

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. It requires that all stations remain synchronized and that all have the same system startup time. Otherwise, due to the recursive window splitting, there is no way to determine when a CRI is in progress. 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 differences between the Sliding Partition and Two Cell CRAs are 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, but at high or overloaded conditions, the difference becomes significant.

Splitting protocols, due to the contention required for each packet transmission, 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.

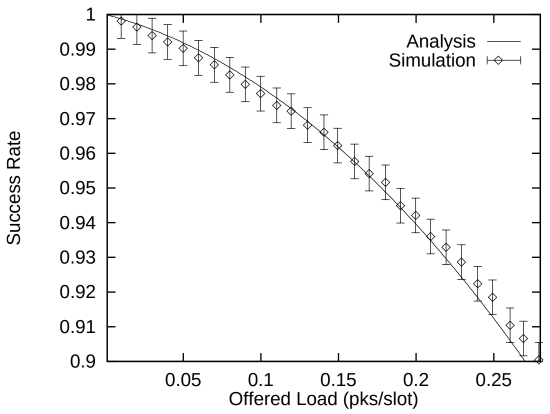

Fig. 1. Blocked access Sliding Partition.

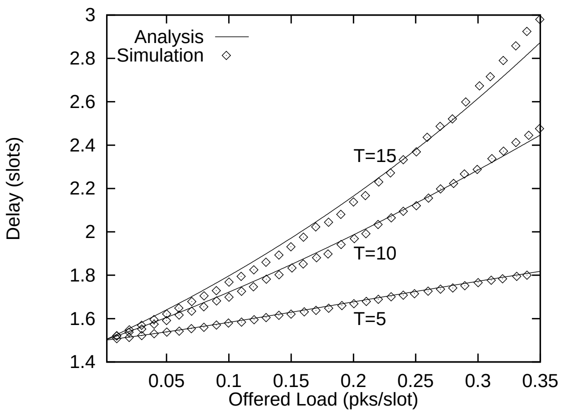

Fig. 2. Blocked access Sliding Partition delay.

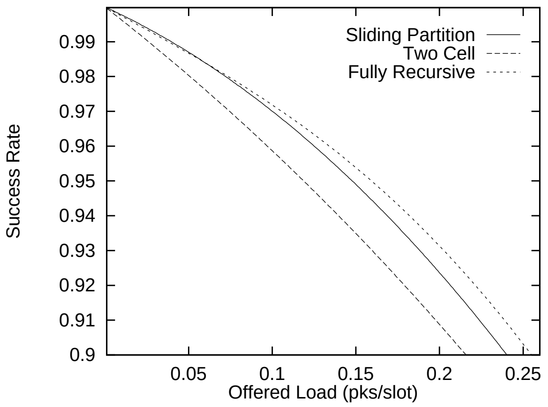

Fig. 3. Blocked access protocols, $T = 10$.

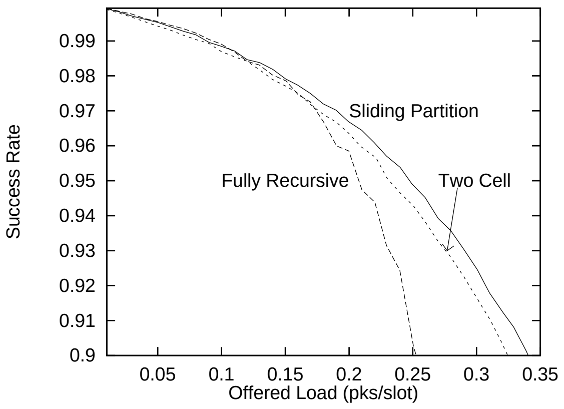

Fig. 4. Blocked access protocols, $T = 30$.

REFERENCES

[1] 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.

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

[3] 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.

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

[5] 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.

[6] 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.

[7] S. S. Panwar, D. Towsley, and Y. Armoni. Collision resolution algorithms for a time-constrained multiaccess channel. *IEEE Transactions on Communications*, 41(7):1023–1026, July 1993.

[8] M. J. Markowski and A. S. Sethi. Analysis of a soft real-time protocol. Technical Report 96-02, Dept. of Computer and Information Sciences, University of Delaware, Newark, DE, November 1995.

[9] M. J. Markowski and A. S. Sethi. Evaluation of wireless soft real-time protocols. In *IEEE Proceedings Real-Time Technology and Applications Symposium*, pages 139–146, Boston, MA, June 1996.

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

[11] M. J. Markowski and A. S. Sethi. Blocked and free access real-time splitting protocols. *Integrated Computer-Aided Engineering*, to appear. John Wiley & Sons, Inc.

[12] Mil 3, Inc. Opnet modeler, v3.0, 1996. Washington, DC.

[13] 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.

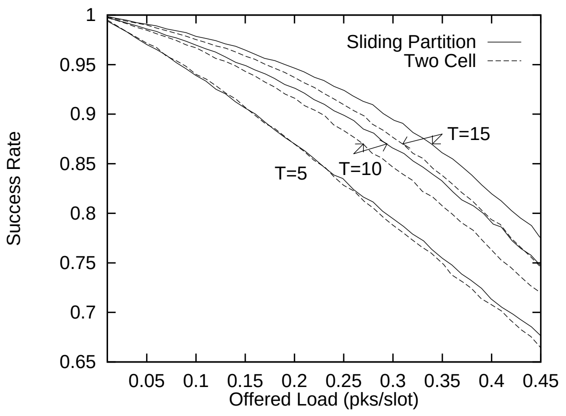

Fig. 5. Blocked access protocols, $T = 5, 10, 15$.

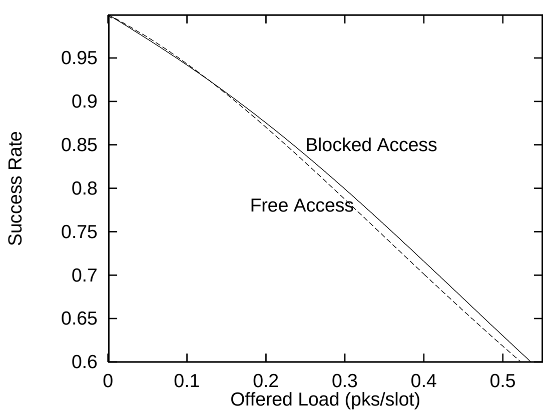

Fig. 8. Blocked and free Sliding Partition, $T = 5$.

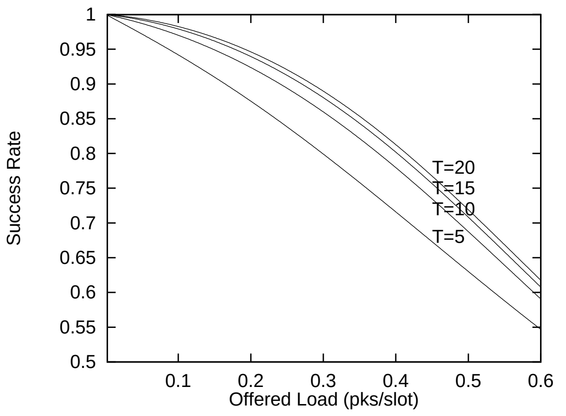

Fig. 6. Blocked access Sliding Partition, $T = 5$–20.

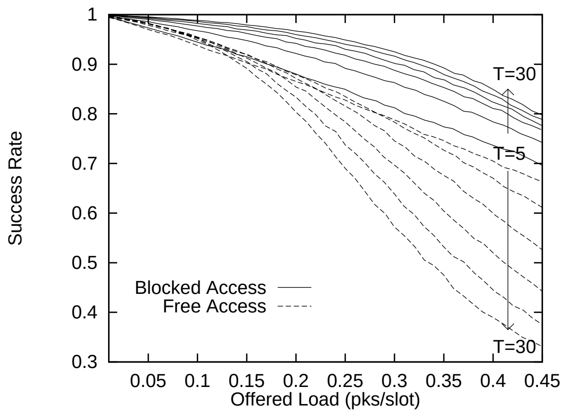

Fig. 9. Blocked and free Sliding Partition.

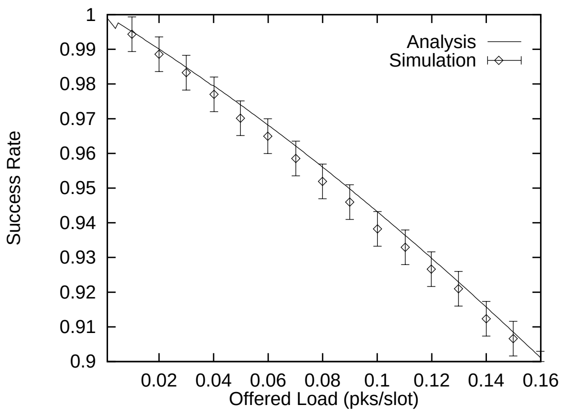

Fig. 7. Free access Sliding Partition.

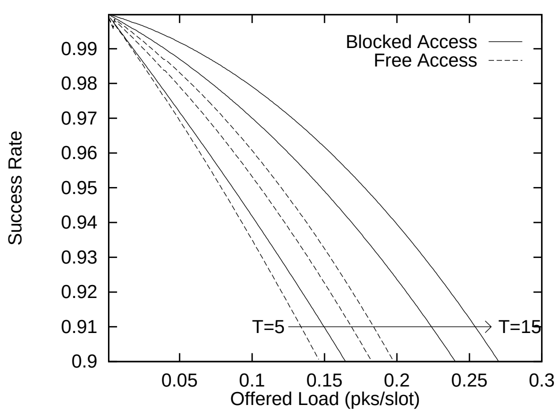

Fig. 10. Blocked and free access Sliding Partition, $e_1 = .9$.