DESIGN AND ANALYSIS OF WIRELESS REAL-TIME DATA LINK LAYER PROTOCOLS

by

Michael J. Markowski

A dissertation submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer and Information Sciences

Spring 1998

DESIGN AND ANALYSIS OF WIRELESS REAL-TIME DATA LINK LAYER PROTOCOLS

by

Michael J. Markowski

Approved: _____

Errol Lloyd, Ph.D.
Chairman of the Department of Computer and Information Sciences

Approved: _____

Conrado M. Gempesaw II, Ph.D.
Vice Provost for Academic Programs and Planning

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Adarshpal Sethi, Ph.D.
Professor in charge of dissertation

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Paul Amer, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Wenbo Li, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Tuncay Saydam, Ph.D.
Member of dissertation committee

ACKNOWLEDGMENTS

I am grateful to everyone who has made this accomplishment possible. I would like to thank my advisor, Professor Sethi, for his guidance, interest, and encouragement during the course of the program. I would also like to thank my committee members for their many suggestions and recommendations that have strengthened my research. In addition, I would like to thank the US Army Research Laboratory for financially supporting me during my pursuit of a PhD.

My friends in the department, and especially Yin Bao and Pramod Kalyanasundaram, have my deepest gratitude for making our time together in the program so enjoyable. I also thank my parents and sister not only for their continual encouragement, but for providing a wonderful home that has made so many other things in life possible.

Most of all, I thank my wife, Alice, and our two young sons, Nathaniel and Jonathan, who sacrificed some of our invaluable time together so that I could complete graduate school. I couldn't have done it without your support!

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This dissertation investigates the design and analysis of real-time protocols at the MAC and LLC layers. In particular, the difficulty of making real-time quality of service guarantees on a random access wireless network is considered.

First, an initial real-time MAC layer protocol is designed. A corresponding technique for its analysis is developed as well. The protocol incorporates two new techniques: transmission ordering based on real-time properties and packet blocking due to deadline expiration.

Second, more protocols are developed and studied. Initially, the focus was on determining how to make use of real-time properties, but here, the goal is to improve on real-time performance. We look at three algorithms and compare their success rates. Additionally, we consider their performance in two very different types of protocols: blocked access and free access. Blocked access protocols are more orderly and of the two types usually offer higher throughput. However, free access protocols are easier to implement.

Third, we develop the first fully distributed, preemptive MAC protocol that makes quality of service guarantees while allowing for the coexistence of hard, soft, and non real-time data. To support full distribution of the algorithm, an additional supporting protocol is developed. With it, each station takes part in the generation of channel feedback data, essential to the proper operation of the MAC protocol itself.

Fourth, we develop a simple LLC layer that uses a real-time scheduler to determine which of its potentially many packets to submit to the MAC layer, where

actual transmission will take place. Several schedulers are compared, but in all cases, scheduling decisions are based not only on queue sizes but also on transmission success/failure results passed back from the MAC layer.

The first and second problems are used as the foundation to finally develop the more general MAC layer protocol of the third. Combined with the LLC development, the final result is a basic data link layer that provides for real-time support of traffic in an *ad hoc* wireless network.

Chapter 1

INTRODUCTION

While the popular ISO OSI model of computer network communications is made up of seven layers, the bottom-most physical layer is the means of actual interconnection and plays a large role in determining what sorts of protocols can be used in the upper six layers. A network designer must take into account not only bandwidth, error rate, and other physical properties of the medium of interconnection, but also cost and application requirements. In most office or laboratory environments, eventual and correct communication of data is all that is necessary because intermittent, failed transmissions are usually resolved much more quickly than human senses can perceive. As a result, Ethernet currently meets the requirements of the bulk of computer users. However, no protocol satisfies the needs of all situations.

Some applications are especially demanding. For instance, scientific visualization requires large bandwidth. Commercial services such as teleconferencing or virtual networking require not only large bandwidth, but a means of reliably providing, monitoring, and charging for the services. The type of system we consider, the *real-time* system, has still different requirements. It interacts with the surrounding environment based on occurrences of real world events and hence has stricter requirements than, say, an email system, on how the time critical data is handled. For instance, if robots on an assembly line are about to crash, the communication protocol must be able to give priority to the *"Halt"* message over the *"Begin spray painting"* message.

Table 1.1: Definitions of real-time message types.

Hard real-time messages	*Vital messages that are guaranteed delivery by their deadlines.*
Soft real-time messages	*Time critical messages that can afford some lateness or loss.*
Non real-time messages	*Messages that are not time critical.*

Real-time traffic itself can also be further classified. *Hard* real-time traffic is that which is so important that its deadlines must be honored once accepted for transmission. More common in wireless systems is *soft* real-time data which, while still time critical, can usually afford some lateness or even loss. A third class, *firm* real-time, has requirements lie somewhere between those of hard and soft deadlines. A certain sensor reading might be able to be lost three times in a row, but must be received no more than four readings apart. And finally, a requirement often overlooked in real-time systems is that non real-time data must be able to coexist with real-time data. Non real-time data, data without deadlines, should not be given such low priority that it has no opportunity to be transmitted. These definitions are summarized in Table 1.1.

While real-time systems on wire-based networks are difficult to implement, implementation on a wireless network is even more complex. A wireless link has a much higher bit error rate (BER), about 10^{-4} or more, than an optical receiver which is usually designed for $\leq 10^{-9}$, or coaxial cable where 10^{-7} is typical. While the typical wireless BER is certainly not desirable, there are situations with no option to use wire or fiber-based communication. Some examples are communication on the battlefield, in forestry service, or to coordinate search and rescue efforts.

For instance, on the battlefield, one or more nodes may be dedicated to searching for radar, infrared, or radio signals. If these are determined to not be

friendly, a message would be sent to other nodes. Those nodes, upon receiving the same messages from other sensory nodes, might alert an artillery station. Similarly, in forestry service, sensors might exist to detect fire, earthquakes, distress beacons, and so on. As with battlefield nodes, when triggered, these nodes might at once notify other nodes. If several sensors report fire, a helicopter dispatcher might be notified. And the same pattern would exist for search and rescue real-time systems. If a distress beacon is received, a search and rescue team might be dispatched. If one search unit spots debris from a ship wreck, its location might be cross referenced with water current data at another station to predict where life rafts might be. At that point, a message could be automatically sent to the nearest rescue unit.

The same theme runs through the examples. Data is generated automatically in a high level feedback system (where the feedback loop is often closed by human decision making), tends to be comprised of short status updating data, and has the potential to trigger an automatic response at other nodes. The data, data delivery times, and response times are all critical factors in a real-time system. The real-time wireless network underlying the system is responsible for getting the data to its destination while the data is still useful.

1.1 Systems of Interest

Like these examples indicate, as real-time systems become more common, there will be greater diversity as well. It is not our intention to develop a real-time wireless data link layer protocol general enough for use by any application. Instead, we focus on a certain class of systems. In these real-time systems, we assume each station is cooperating as part of a larger, overall system. Because it is interacting with the external world by making decisions based on complex rules and heuristics, a real-time system is more complex than a traditional feedback system with a simple PID (Proportional/Integral/Derivative) controller. However, the system does behave in a similar fashion in that a subset of stations tends to be

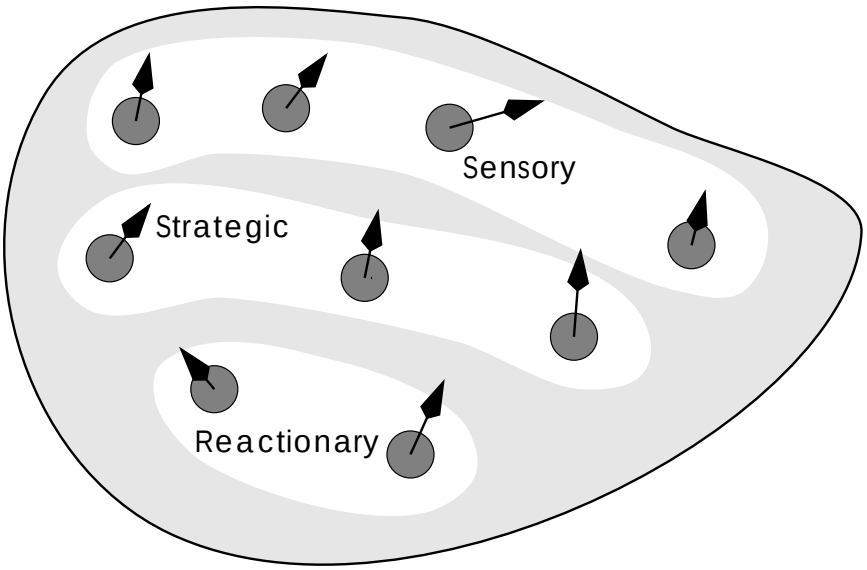


Figure 1.1: Typical wireless real-time system.

sensory in nature and passes on its readings to other nodes. Those nodes interpret the data, make decisions and send messages to other nodes which react in some way. A typical system is illustrated in Figure 1.1, where velocity vectors and system level functions are indicated. The data generated tends to be short in length and bursty in nature, since the system responds to unpredictable changes in its environment.

Additionally, such systems are comprised of mobile nodes that must have the ability to continue operations, if only in a degraded way, should one or more nodes fail. This implies that there can be no central point of authority such as a base station, which in turn means that protocols must be fully distributed. Networks of this type have been popularly dubbed *ad hoc* networks.

1.2 Problem Motivation

Once within the system, the problem, from the point of view of the wireless real-time applications, can be easily stated: messages with deadlines and priorities

are generated and must be transmitted. In the general situation, time critical data is created on one machine and must be forwarded by several other computers to reach the destination. It is difficult when multiple hops are involved to know whether or not a deadline can be met. The end-to-end deadline must, at the network layer, be divided among the series of links between source and destination. Then a determination must be made, in a point-to-point manner, whether or not each local, or link-based, deadline can be met.

This determination is made through interlayer communication at the network and data link layer interface. The network layer, based on the capabilities of its various outgoing links, must determine how to route the packet. This requires the data link layer to have the ability to make decisions based on deadlines and priorities. Such a requirement extends the capabilities of a typical data link layer, whose main goal is implementation of a virtual "bit pipe" where any bits "poured" into one end of the pipe should come out at the other end in an error-free manner.

To use a real-time bit pipe, the network layer has a set of real-time quality of service (QoS) requirements such as minimum interarrival rate, maximum acceptable loss rate, connection-oriented or connectionless service, real-time class required, and so on, and must negotiate with the data link layer to see if the QoS requirements can be met. The data link layer maps network layer requirements onto available resources and responds by either accepting the additional load from the network layer, denying it, or offering a probability, less than one, of service. The network layer completes the negotiation by using the offered data link layer services or indicating to the transport layer that not enough resources are available to support the QoS requirements.

The focus of our work is the development of a real-time data link layer for use in real-time wireless networks. We do not consider the network layer protocol or its interface with the data link layer directly, but work towards building a set of data

link layer services rich enough that higher layer protocols can elaborate on to offer applications flexible and easy to use real-time features. Importantly, though, these real-time features, ultimately supported at the base of the protocol stack, would make maximal use of the channel, rather than make do with the side effects of a best effort service.

Following the common IEEE data link model, we further break down the layer into two sublayers: the logical link control (LLC) and the medium access control (MAC) layers. The relationship between the ISO OSI model's data link layer and the IEEE model is shown in Figure 1.2. The MAC layer is near the base of the protocol stack interfacing directly with the physical layer. To provide the best real-time service at all layers in the protocol stack, it is necessary for real-time services to exist here. In nearly all networks, however this is not the case. Often the application layer sends real-time packets to a destination over a network without real-time services. As a result, packets will be late, lost, duplicated, etc., in unpredictable ways. The common solution is to have some sort of filtering program at the receiver that drops late arrivals, buffers packets, and so on, in the hope of controlling interframe delay. But offering real-time service at the MAC layer makes it possible to build on these and offer still better real-time service at the LLC layer. Building real-time protocols can continue at each level of the stack with a protocol using its immediate lower layer services to create more elaborate real-time services for the next higher layer so that the application, though still needing some basic knowledge of the network's capabilities, simply requests real-time service. Packets are subsequently sent without the application having to deal further with the implementation of the services.

1.3 The Real-Time LLC Layer

The LLC is the top half of the data link layer and responsible for the abstract task of moving a network layer packet across a link. A general LLC layer offers

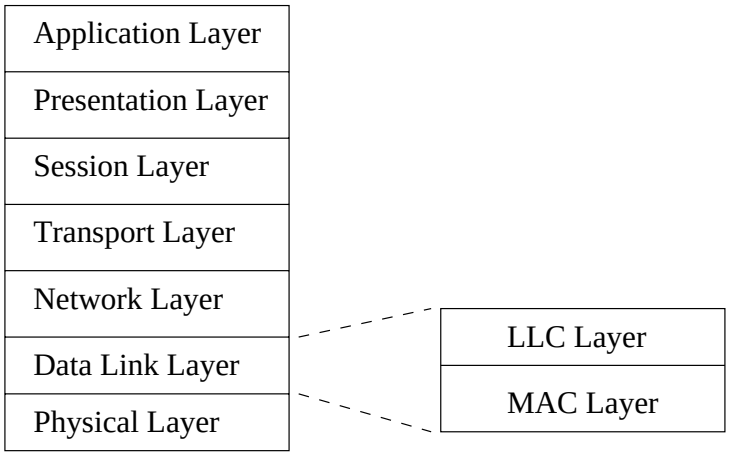


Figure 1.2: ISO OSI stack showing MAC and LLC layers.

connection-oriented service, connectionless service, and acknowledged connectionless service. Connection-oriented services are used for long term data transfer, such as voice packets during telephone calls or transmission of video and large files. Connectionless service is useful for short messages generated in a somewhat bursty manner. This is typical of automated systems and what we are trying to support. All of the messages will be either hard, firm, soft, or non real-time in nature and with deadlines appropriate for the application. The protocols presented in this dissertation are designed for these types of messages. Accordingly, we do not pursue connection-oriented protocols.

Implementation of connectionless service is easier in many ways than connection-oriented service. For example, there is no need for sliding windows, piggybacking of acks, or connection establishment and breakdown. In most ways, though, implementing connectionless service is more difficult for real-time use because the traffic is not serviced in a first-in-first-out manner and because of the varying delays induced by random access protocols. The LLC itself will have at least three data queues: one each for hard, soft and non real-time data. While a very low traffic

rate essentially reduces to a first-come-first-served (FCFS) servicing scheme, when the load becomes moderate to heavy, scheduling is difficult. The LLC must be able to quickly pick and submit a best candidate for transmission, since only one packet at a time can be transmitted. However, due to the noisy nature of a radio link as well as potential collisions, *i.e.,* two or more nodes transmitting simultaneously, a submitted packet might not be transmitted successfully. In that case the LLC must determine whether or not the packet can be rescheduled for future submission.

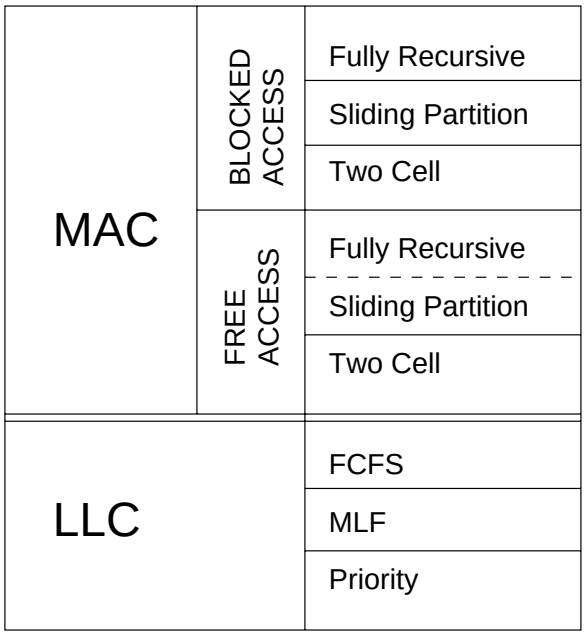
But because the LLC must wait on the distributed MAC layer protocol for packet transmission, LLC performance is tightly coupled to MAC layer performance. The general focus of LLC layer actions are node-based though. Most of its actions in support of real-time services revolve around studying its own queues and deciding, via scheduling algorithms, which packet to submit for transmission.

1.4 The Real-Time MAC Layer

In contrast, the MAC layer is explicitly aware that it is one of several nodes competing for access to the same radio channel. One view of the MAC layer protocol is that it is a distributed server for the channel. The difference between this view and a typical $M/G/1$ system is that the packets in the queue are distributed; nowhere does global knowledge exist of how many items are waiting or in what order. As a result, the queue servicing algorithm must be distributed among the nodes in the system, and must operate in a manner that allows the most deserving packet first service.

Since it is possible for packets to collide, it can take some time for the system to collectively provide channel access to contending packets. During this interval, it is possible for higher priority data streams to want the channel, requiring that the ability to preempt lower priority streams exists. However, the lower priority traffic must still meet any original QoS guarantees it was promised.

Table 1.2: Data link layer algorithms studied.



1.5 Overview of Research

Six major algorithms are studied in this dissertation; three at the MAC layer and three more at the LLC layer. Table 1.2 lists their names and where each fits into the scope of the research. Note that at the MAC layer, the same three algorithms are studied but in two different types of protocols, blocked and free access. Additionally, the Fully Recursive MAC algorithm is studied in Chapters 3 and 4 with packet arrivals having different real-time properties. While these variations affect the mathematical analysis, the algorithms remain mostly unchanged. Due to the number of algorithms, variations and traffic models used, occasionally returning to Table 1.2 is helpful when a reference point is needed.

As we develop increasingly more capable protocols in the subsequent chapters, ultimately a preemptive MAC protocol is presented that can offer QoS guarantees to hard, soft and non real-time data. The preemptive MAC protocol, comprised

of a subset of the MAC algorithms in Table 1.2, is used as the foundation for a simple LLC layer. To decide when and which packets to submit to the MAC layer for transmission, the LLC layer uses a real-time scheduling algorithm requiring the development of capabilities generally not existent in current data link layer protocols for *ad hoc* wireless networks.

In Chapter 2 we present a survey of relevant literature. Building on the existing work, Chapter 3 presents the first protocols we studied. To learn what affects real-time protocols positively and what affects them negatively, we confine our interest to soft real-time systems where all arriving packets have the same initial deadline. We extend Chapter 3's analytic technique in Chapter 4 to consider traffic in soft real-time systems where initial deadlines are drawn from known distributions. In Chapter 5, results from the Chapters 3 and 4 are used to design and analyze a MAC layer protocol capable of transmitting hard, soft, and non real-time traffic in a preemptive manner. The final technical chapter, Chapter 6, first implements via simulation a non-ideal physical radio channel, then a detailed MAC layer, and finally an LLC layer with acknowledgements, retransmissions, segmentation and reassembly, and a real-time scheduler. Chapter 7 summarizes the work.

Chapter 2

RESEARCH DIRECTIONS IN REAL-TIME SYSTEMS

Wireless real-time systems, in their most general forms, will have hard, soft and non real-time data, all of which must access the same, often noisy, channel. For our purposes in the discussion of wireles networks, we refine the system level definitions of Table 1.1 as follows. We define hard real-time to mean deadlines of accepted packets are guaranteed to be met on an error free channel, soft real-time to mean expected, or average, success rates are offered, and non real-time to mean that best effort service is provided. While the term *deadline* is frequently used when describing real-time systems, it is often more convenient to discuss real-time design issues in terms of *laxity*. Laxity is the amount of time remaining before a packet must be transmitted in order to be received by its deadline.

In real-time systems where each station is a cooperating part of a larger system, messages tend to be part of distributed "sense, process, react" transactions and are fairly short, even more so for the special case of hard real-time data. Because of this, protocols for long term resource allocation are unnecessary and are in fact a hindrance due to the added overhead of setting up and breaking down connections.

In this dissertation, we present wireless MAC and LLC layer protocols that provide a flexible means for the coexistence and transmission of heterogeneous real-time data. We begin by offering support for soft real-time deadlines and build on that to eventually offer quality of service guarantees for hard, soft and non real-time data. Splitting protocols with limited feedback sensing are used to support these guarantees.

2.1 Wireless MAC Protocols

Nearly all implementations of wireless systems make use of Aloha [Abr70, Abr85] or CSMA [KT75] techniques. The Aloha protocol has the advantage of being simple to implement while making fair use of bandwidth. In the original implementation at the University of Hawaii, a central node listened at radio frequency 407 MHz for incoming packets. It immediately retransmitted these packets at 413 MHz at 9600 b/s. Thus, a remote station knew its transmission collided with another station's if it did not hear its packet on 413 MHz. After a collision, the remote station rescheduled a transmission after a random delay. In practice, implementations could be simpler still: packets were transmitted without feedback. Only when a higher layer protocol discovered not all data was correctly transmitted is the data resubmitted to the MAC layer. If transmissions always occur at fixed points in time, or slot boundaries, the protocol is called *slotted Aloha,* otherwise it is *unslotted Aloha.* Analysis shows that unslotted Aloha has a maximum throughput of 0.18, while slotted Aloha throughput is 0.36 of the available bandwidth when packets of equal lengths are used [BG92]. This is shown in Figure 2.1.

An important shortcoming with Aloha, as can be seen in the illustration, is that it is unstable. The maximum throughputs occur at particular offered loads. As load increases beyond those points, throughput quickly decreases. In the case of real-time data, there is the added problem that the random back-off following collisions offers no provision for meeting transmission deadlines. As a result, real-time guarantees cannot be made.

It is noteworthy that most cellular telephone systems use versions of reservation Aloha. In this protocol, minislots—often one bit long—are accessed using slotted Aloha. A base station echos back the list of minislots with indications of which minislots were successfully obtained by stations. So if a station tried to write a bit into a minislot and subsequently sees a corresponding reservation in the base

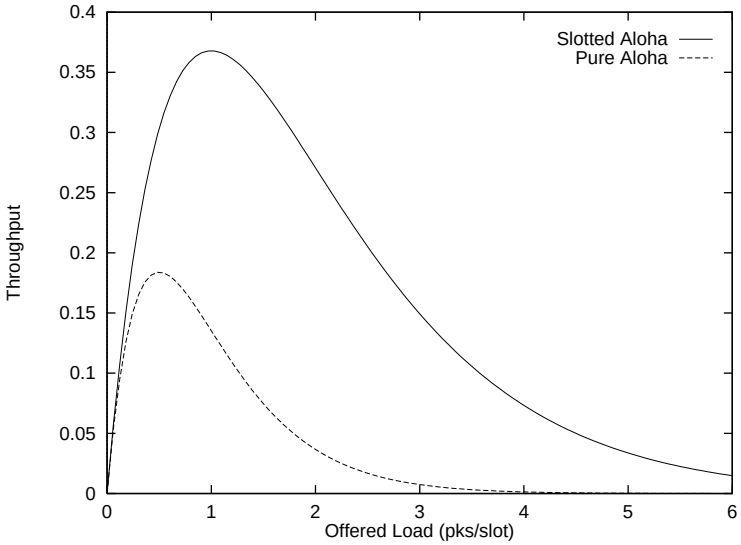


Figure 2.1: Comparison of slotted and pure Aloha.

station's list, the station knows it has successfully reserved upcoming data slots. The data slots are then used in a round robin or TDMA way by the successful stations.

Carrier sense multiple access (CSMA) systems improve upon pure Aloha protocols. Circuitry is used that monitors the radio channel just prior to transmission. If signal strength above a certain level is sensed, the sensing station knows the channel is busy, and the station waits some random time before it again performs a carrier sensing. When the channel is idle, the station transmits its data. However, due to the finite propagation speed of the signal, it is possible that after a station senses an idle channel, it begins transmission only to collide soon after with another station's transmission.

In the case of CSMA protocols, throughput is affected by β, the length of time required to determine that the channel is idle and based on the propagation and detection delays. For slotted systems the throughput is $1/(1+\sqrt{2\beta})$, while unslotted systems have a throughput of $1/(1+2\sqrt{\beta})$ [BG92]. This implies that performance

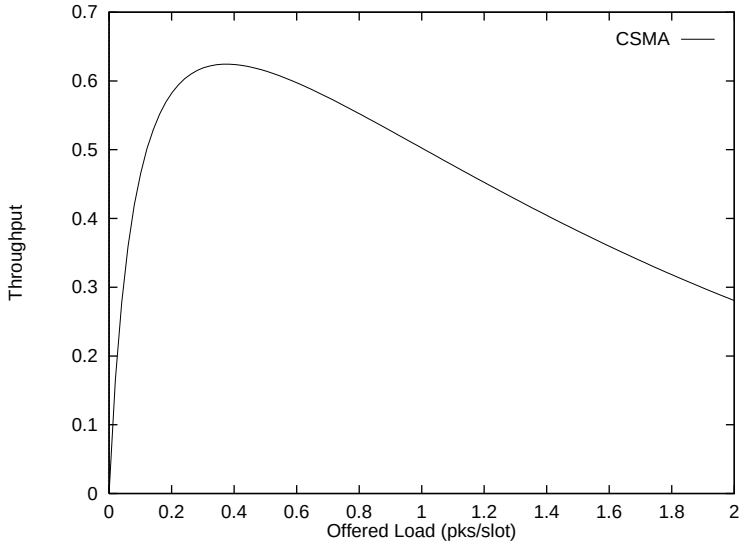


Figure 2.2: CSMA Performance.

degrades as β increases, channel rate increases or packet length decreases. In addition, though less severe than with Aloha, CSMA also has stability problems. So while a typical VHF channel of 16 kb/s with 0.1 s required to detect an idle channel and packets of 1 kb has a maximum throughput of about 0.64, significantly greater than slotted Aloha, we still cannot use the protocol to make real-time guarantees due to the random backoff after collisions. The throughput versus load graph using the above values is shown in Figure 2.2.

Splitting protocols, however, do allow us to make such guarantees while offering a maximum throughput of around 0.5 of the available bandwidth. Initially, splitting protocols were developed as a way to overcome the inherent instability of Aloha-based protocols. Even in overloaded conditions, unlike Aloha or CSMA, splitting protocols continue to offer maximum throughputs. While reservation Aloha protocols can boast throughputs of about 90% for long messages such as telephone calls, for short exchanges the reservation process not only becomes a more significant fraction of the overhead, but the length of time required to reserve a data slot has

the undesirable real-time property of being unbounded. Splitting protocols, as will be described in later chapters, are capable of splitting the set of collided stations into groups based on real-time properties of the traffic. It is then possible to offer real-time guarantees as well as the maximum throughput of the protocols.

Splitting protocols evolved from military medical tests used to determine if new recruits were infected with disease [Wol85]. Since most people are expected to be healthy, many blood samples could be mixed together for a single test. Only if the tests were positive would new samples with half of the original group in each be tested until the infected recruits were isolated, as illustrated in Figure 2.3. Whenever a test shows that a mixture is uninfected, testing of that mixture stops since all contributing recruits must be healthy. Should the test indicate infection, the test is performed on two new mixtures, each made from blood samples of one half of the contributors to the original mixture. Notice that with the unusually high infection ratio of this example, eleven tests were required to test eight recruits. There is a large potential savings in numbers of tests required, however, as infection rate in the population decreases.

In the case of wireless transmission, the "infected recruit" is data lost due to a collision and the "test result" is the feedback indicating whether or not a collision occurred in the previous slot. The "potential savings in numbers of tests" corresponds to how efficiently a collision is resolved. After an initial collision, each collided station uses the current value of some agreed upon random variable to join either the *active* or the *inactive* group. The active group transmits during the next slot, and the splitting continues until transmission by the active group is followed by a noncollision feedback. Then the inactive group becomes the active group and the algorithm is applied to it. The splitting protocols we developed will be described in detail as they are introduced.

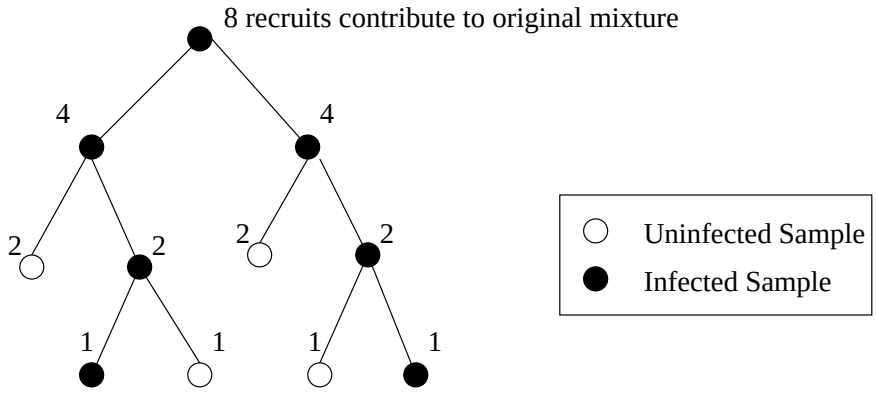


Figure 2.3: Medical testing for infected blood.

2.2 Real-Time Wireless MAC Protocols

While virtually no work has been undertaken to support wireless real-time communication in *ad hoc* networks, a small body of work exists that studies the problem for wire-based real-time CSMA-CD networks. Though CSMA-CD cannot practically be used for wireless networks since generally an antenna cannot be used for simultaneous transmission and reception, in the case of splitting protocols, 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 collision detection and CSMA only in ways peripheral to the splitting algorithm. While Bertsekas and Gallager [BG92] show that splitting in conjunction with CSMA-CD offers no substantial throughput improvement over slotted Aloha, the splitting methods we present may, in fact, be attractive for use in some CSMA-CD protocols because they make use of real-time properties of the traffic. The strength of splitting protocols, however, is their ability to efficiently resolve collisions, which occur much less frequently when CSMA and CD are used. So while throughput will be not be increased appreciably, real-time splitting protocols, even in CSMA-CD systems, do offer the benefit of a worst case bound on channel access times.

In addition to outperforming Aloha, studies in [ZSR90] have shown that window splitting outperforms other random access real-time techniques such as virtual clocks. In non real-time CSMA-CD systems, virtual time clocks were first considered by Molle and Kleinrock [MK85] and based on message arrival time. The technique had the advantage of making transmission of queued messages fairer, though it provides no consideration for deadlines. The method was adapted by Zhao and Ramamritham [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. The underlying protocol is inference avoiding. That is, each transmission attempt is made without reference to past history of the channel or of that particular station. However, the basic multiaccess scheme is augmented by using two clocks at each node. One gives real and the other virtual time. When the channel is busy, the virtual clock stops running and starts again when the channel is idle. When it does run, the virtual clock runs faster than the real clock. How much faster is a value to be determined based on expected traffic patterns. Depending on the variation used, earliest packet transmission (virtual) times are set based on either first come first served, minimum length first, minimum laxity first, minimum deadline first. A message is sent only if its transmission time is less than or equal to the time on the virtual clock.

Subsequently, Zhao, Stankovic, and Ramamritham [ZSR90] proposed a splitting protocol using the last two slots of channel history. It always performed in simulation at least as well as the virtual time protocols and often better. However, this particular splitting protocol introduced laxity abnormalities. When laxities increased, presumably making transmission scheduling easier, system performance was actually degraded. The protocol allowed increased contention when laxities were greater, which in turn caused less packets to be transmitted by their deadlines.

Algorithmically simpler CSMA-CD splitting protocols that used only a single, most recent slot's feedback were presented for both hard and soft real-time systems by Arvind [Arv91]. 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 independently developed and analyzed in depth by Papantoni-Kazakos [PK92]. Hers is a hard real-time variant of the soft real-time protocol presented and analyzed by Paterakis, Georgiadis and Papantoni-Kazakos [PGPK89]. The latter assumes all packets have the same initial deadline [PGPK89]. We explored this type of soft real-time system but studied a more complex fully recursive splitting algorithm in [MS95].

To support a more general soft real-time system, we first extended that work to study protocol performances when initial deadlines are fixed [MS96]. We subsequently extended that to consider performance when initial deadlines are drawn from known distribution [MS97a],[MS97b],[MS98c]. These works presented six protocols and through analysis and simulation determined the one with the best overall performance. Interestingly, they showed that performance reaches an operational maximum as the range of deadlines increases. The existence of an operational maximum implies that splitting protocols perform no better after the maximum initial laxity reaches a certain number of slots, approximately thirty it turns out. The importance of this discovery is its impact on the LLC layer. Since the LLC schedules MAC submissions and those packets often have deadlines much longer than thirty slot times, the operation of the MAC layer's transmission scheduling has a direct effect on the design of the LLC scheduling algorithm.

In general, splitting protocols can be grossly classified based on two common properties: the random variable used for splitting and the splitting methodology used. As a concrete example, the first data communication splitting algorithm proposed, which we call CTM here using the surname initials of the three

researchers, was by Capetanakis [Cap79b],[Cap79a] and, independently, by Tsybakov and Mikhailov [TM78]. In their method, the nodes involved in a collision flip coins afterwards. Those that flip tails wait, while those flipping heads retransmit. This continues recursively on both sets of packets, so a high multiplicity collision, *i.e.,* a collision involving many packets, 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 [Gal78, BG92] and again, independently, Tsybakov and Mikhailov [TM80], improved on the fully recursive algorithm in a number of ways. In the algorithm, which we call GTM again using surname initials, the splitting variable is queue arrival time. The splitting method uses only two groups unlike the unbounded number in CTM. The active group, as defined at the end of Section 2.1, is continually split until a packet is successfully transmitted. The inactive group is then ignored. That is, the algorithm restarts using arrival times just after the newly resolved interval as if the inactive group never existed. Ternary feedback indicating success, idle, or collision is used in the algorithm.

The more recent two cell random splitting by Paterakis, Georgiadis and Papantoni-Kazakos [PGPK89] is similar in many respects to the GTM algorithm except that the splitting variable is a coin toss outcome, and binary feedback indicating collision or noncollision is used, rather than ternary. The advantage of using a coin toss is that it separates splitting from properties of the arrival process. This is helpful if the arrival process is unknown or not well suited to FCFS splitting, *e.g.,* arrivals occurring only at slot boundaries.

2.3 Supporting Heterogeneous Real-Time Data

As mentioned, a general system will support the coexistence of hard, soft and non real-time data. For the case of heterogeneous traffic, other research considers the coexistence of hard and non real-time data [VPK92, DPK93], hard and soft [Arv91],

soft and non real-time data [CS97, CKT89], and also multiple classes of soft real-time traffic [PKLT95]. However, most of these works assume a normalized transmission time unit of one and ignore details of the MAC layer. Our analysis and simulations take into account the workings of the MAC protocol while offering support for all three classes of real-time traffic. Before our work, no published research offered support for all three real-time classes in wireless networks, *ad hoc* or cellular.

2.4 Analytic Techniques for Random Access MAC Protocols

Once a protocol has been described algorithmically, its performance must be studied. Random access MAC algorithms, real-time or not, can often be characterized by regenerative, or renewal, processes [Gal96]. This is a counting process for which interarrival distances, packet arrivals in this case, are positive independent identically distributed (IID) random variables. The advantage to studying regenerative processes is that the embedded renewal intervals let us separate the long term behavior of the process from the behavior of the actual process within a renewal period.

In the protocols we present, one renewal period is the amount of time to fully resolve channel contention by some number of nodes. For the analysis of systems of this sort, Georgiadis, Merakos, and Papantoni-Kazakos [GMPK87] developed a straightforward, general method of delay analysis using regenerative properties of random multiple access algorithms. That method of analysis was elaborated on by Paterakis *et al.* [PGPK89] to analyze their protocol where all packets initially have the same initial deadline. While Paterakis *et al.* took a particular protocol and studied its performance, Panwar, Towsley and Armoni [PTA93] did the opposite. They used the value iteration method to find the optimal splitting algorithm for fixed initial laxities of up to 4 slots. The expressions derived that determine optimal splitting are too complex to consider larger laxities. Hence, we take the more

conventional approach of designing protocols and then analyzing them. This also allows the study of more general systems with varying initial laxities.

2.5 New Contributions

In the current literature, there are three unaddressed problems whose study is necessary for the development of a wireless, real-time, *ad hoc* network. (1) Foremost is the need for explicit deadline support at the MAC layer. Without it, poor use is made of channel bandwidth, an already limited resource. Since for any real-time application, real-time support is needed somewhere in the protocol stack, it is most sensible to offer basic time-oriented services at the base of the stack where fullest control can be exerted over use of the communication medium. (2) It is additionally desirable to be able to guarantee the maximum bandwidth that can be used for successful transmissions. Without this guarantee, it is impossible to bound worst case transmission times, something critical to the implementation of a real-time protocol. (3) It is also necessary to distinguish between classes of real-time services. The related literature, summarized in the preceding sections of this chapter, have either not considered the above problems or have only done so in a piece-meal fashion.

By developing solutions to each of the open areas mentioned above, we have been able to create the first preemptive wireless MAC protocol for use in *ad hoc* networks [MS98a],[MS98b]. That, in turn, lets us tightly couple an LLC layer to it; an LLC that makes decisions based on the real-time MAC services offered. In addition to the particular protocols and analytic methods developed, our work makes new contributions at more conceptual levels as well. It has been traditionally difficult to offer worst case transmission times in random access protocols, where packet delays may be potentially unbounded due to collisions. Through the use of real-time splitting protocols, however, we show that it is indeed possible to make such guarantees. Furthermore, average case delays can be predicted. Splitting protocols

were originally developed with the idea that some central source will provide the channel feedback information. Because our target environment is fully decentralized, we develop a fully distributed feedback protocol eliminating the need for any central point of control. Next, by assuming that a message is dropped whenever its delay exceeds its initial laxity, and by using a minimum rate of successful transmission, we show that not only can window-splitting protocols be modified to work successfully in these environments, their throughput can even exceed the maximum throughputs attained by current non real-time splitting protocols. For still more general use, preemption is introduced so that various classes of real-time data can co-exist. Preemption can occur both within a single node and between nodes.

In the model of our LLC layer, unlike most other published work, we couple protocol actions directly to the real-time services of the MAC layer. While protocol layering is still strictly enforced, the new services can be made use of by LLC schedulers. To get a still better understanding of what performance can be expected in a real implementation, more realistic channels with bit errors are used as well. Together, our solutions to these individual problems let us contribute to the growing need for better real-time services by offering a capable real-time data link layer.

Chapter 3

SOFT REAL-TIME MAC PROTOCOLS WITH FIXED INITIAL DEADLINES

3.1 Introduction

The first real-time system we consider is limited in scope. Instead of allowing the coexistence of hard, soft, and non real-time data, we initially concentrate on a purely soft real-time protocol. The work in this chapter also appeared in [MS96] and [MS95]. Since soft real-time traffic can afford a given level of loss or lateness, soft real-time guarantees offer expected, or average, success rates and delays. Of the three real-time stream types, it is the most complicated, since for soft real-time systems, explicit dropping of packets is required, as will be described shortly. But for both the hard and non real-time streams, the MAC layer attempts to transmit all packets accepted, offering bounded delays for hard real-time and best effort for non real-time data. Any packet dropping is not performed explicitly but is a result of channel errors or loss of synchronization between stations. Soft real-time, though, offers decreasing success rates as offered load increases, so it is important to fully understand this class of traffic before addressing the others.

In addition to restricting our attention to soft real-time traffic, we also assume all nodes are generating similar traffic and all packets have the same initial deadline. These assumptions would be typical for systems involved with transmission of audio, of sensor readings (temperature, radar, etc.), or mobile user position updates, and so on. The data in these systems all have the same useful lifetime and an occasional

loss is not harmful to overall performance. From the viewpoint of characterizing the system mathematically, it also provides us the opportunity to make an initial study of real-time properties without being overburdened by the detail in more general systems. Later chapters build on these findings to create a more general mathematical model.

3.2 An Example Algorithm

Before delving into the detail of the first algorithms studied, it is helpful to consider an example. We use a simplified version of the Two Cell splitting protocol by Paterakis *et al.* [PGPK89]. A slotted system is assumed and after each slot time, feedback is provided by the system indicating that either a collision occurred or that a non-collision occurred. If a collision occurred, each collided node flips a coin.[1] Depending on the outcome, the node either joins the active group or the inactive group. Active group members all will transmit at the next slot boundary. Inactive group members never transmit but instead wait until seeing a non-collision feedback, at which time they transit to the active group.

If a two-way collision were to occur, it is easy to see that there are three possible group memberships: (1) one active and one inactive member, (2) two active members, or (3) two inactive members. These memberships are depicted in Figure 3.1, where the left tree branches indicate the active group, and the right tree branches, the inactive. The root node is the initial multiplicity two collision. Since only the active group transmits, the channel feedback of collision (C) or noncollision (NC) is shown under the active group. We want to determine the expected number of slots it takes to resolve the collision. If there is one member in each group, the resolution is three slots long: initial collision + active group transmission + inactive

[1] That is, a random number in $[0, 1)$ is generated at each node. The value "heads" represents outcomes in one half of the interval, and "tails" in the other.

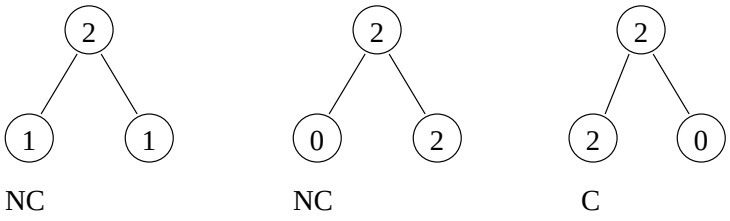


Figure 3.1: Possible outcomes of multiplicity two collision.

group transits to active group and transmits. However if both nodes join the inactive group, one slot is used with the initial collision followed by an idle slot. Then the algorithm is applied to the followup collision. Similarly, if both members join the active group after the initial collision, there is one slot for the initial collision followed by recursing on that and finally the last noncollision for the zero member inactive group. If l_k is the number of slots required to resolve a k-way collision, a multiplicity two collision is characterized by the expression

$$l_2 = \begin{cases} 3 \text{ slots}, & \text{when each group contains 1 member} \\ 2 + l_2 \text{ slots}, & \text{when both members are inactive members} \\ 2 + l_2 \text{ slots}, & \text{when both members are active members} \end{cases} \tag{3.1}$$

Taking expected values, where the notation $L_k = E\{l_k\}$ is the expected number of slots required to resolve a collision of multiplicity k, the analysis proceeds as follows. When an active group contains zero or one members, it clearly takes only one slot to resolve. The last expression includes the three cases above as well as the probability of each occurring.

$$\begin{aligned} L_0 &= 1 \text{ slot} \\ L_1 &= 1 \text{ slot} \\ L_2 &= \frac{1}{2}(1 + L_1 + L_1) + \frac{1}{4}(1 + L_0 + L_2) + \frac{1}{4}(1 + L_2 + L_0) \end{aligned} \tag{3.2}$$

Straightforward algebraic manipulations tell us that $L_2 = 5$ slots. It is easy to extend this to the case for any $k \geq 0$, where, incidentally, induction can be used to show that L_k is linearly bounded for a Poisson arrival process.

This general technique is extended so that time constrained transmissions can be analyzed. The departure process in the real-time environment is more involved because departure is due not only to transmissions but also failure due to deadline expiration.

3.3 Algorithm

With the example providing an indication of the analytic style used, our first real-time protocol can be studied. In it, a packet has a single property of interest, its laxity. Laxity is the maximum amount of time that can elapse prior to transmission, after which the packet will not reach its destination on time (we consider a single-hop radio network). Once a laxity is assigned to a packet, at each slot boundary the laxity is decremented and the packet discarded should its laxity reach zero. The channel is accessed in a slotted manner, *i.e.,* transmissions begin only at slot boundaries, with one slot long packets that are immediately followed by binary, or collision/non-collision, feedback.

To improve throughput, a sliding window is used that enables a given portion of the arrival process. That is, only packets whose arrival times fall within the bounds of the window may contend for the channel. After all arrivals in the sliding window have been transmitted or discarded, the window slides forward enabling the next set of arrivals. The idea is that occasionally it will take much longer than average to resolve a collision. During the long collision resolution interval (CRI), many other packets have arrived and are waiting for the CRI to complete so that they can be transmitted. It is likely that after a long CRI, the next slot will contain a high multiplicity collision. Use of a sliding window, as shown in Figure 3.2, limits contention. The width of the window should be long enough to avoid wasting

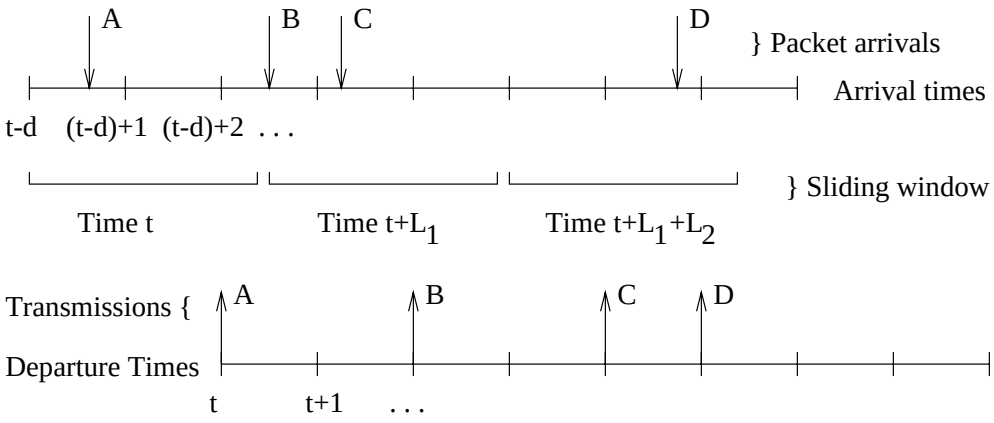


Figure 3.2: Arrival time-based sliding window.

time resolving empty windows, yet long enough to avoid too many collisions. In the illustration, the departure process axis is labeled beginning at time t. Resolving prior collisions means that the interval of the arrival process currently enabled actually arrived some time ago. And even at system startup, it takes one slot to receive feedback, so that the sliding window is always at least one slot in the past and often more. The difference between the current time and the leftmost edge of the sliding window is called the lag, d. Because of the always positive system lag d, the arrival process axis is labeled beginning at time $t - d$. Additionally, the sliding window is shown at three different times. The window's first position at time t, encompasses one packet (A) and so takes $L_1 = 1$ slot to be resolved. The window's second position, at time $t + L_1$, allows two packets (B, C) to contend resulting in a collision. It takes L_2 slots on average to resolve, using the notation of the simple example in the previous section. So the window slides forward to its third position in the figure $(t + L_1) + L_2$ slots later.

When two or more nodes transmit simultaneously 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. Within the sliding window of initial length Δ, any packets involved in a collision are reordered by laxity, left to right, from lowest laxity to highest. The left half of the laxity range will be considered the active group, while the right half will be one of the possibly many inactive groups. In the slot after a collision, only packets whose laxities fall within the active group may transmit. If another collision occurs, the group is split in half, and the CRA recurses first on the left half and then on the right. After each active group is resolved, the most recently created inactive group becomes the active group. The CRA behaves similarly to the classical FCFS splitting algorithm [Cap79b] with a few differences: packets in a group following a collision are ordered by laxity rather than by node arrival time, and feedback is binary rather than ternary. We further impose a bound T which is the maximum number of slots a CRI may comprise. If T slot times are exceeded, packets involved in the CRI are dropped, and the algorithm resets. This is equivalent to all packets having an initial laxity of T.

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 minimum 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.

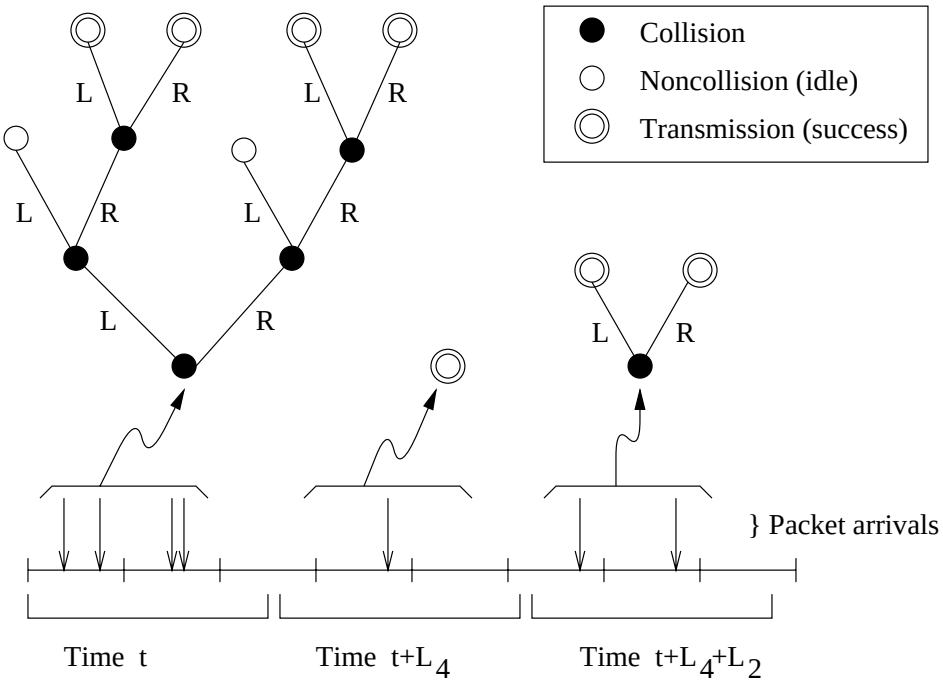


Figure 3.3: Relation between sliding window and CRA.

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, we denote the position of the left edge of the window at time t as x_t and its length in slots as a_t. Finally, h_t can take on the value of L or R indicating whether the collision resolution algorithm 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 $h_t = L$.

Figure 3.3 shows how the algorithm operates. It is much like Figure 3.2, except to simplify the illustration, the departure process is not shown. At the bottom, again are drawn the various positions of the sliding window as time progresses. Whenever a collision occurs, the window does not slide forward until the collision is resolved. The first multiplicity four collision requires L_4 slots for resolution. The window then slides forward enabling the single packet to be transmitted. And finally it slides forward again, encompassing a quickly resolved two-way collision. Figure 3.4 illustrates some of the variables of the analysis 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. Due to

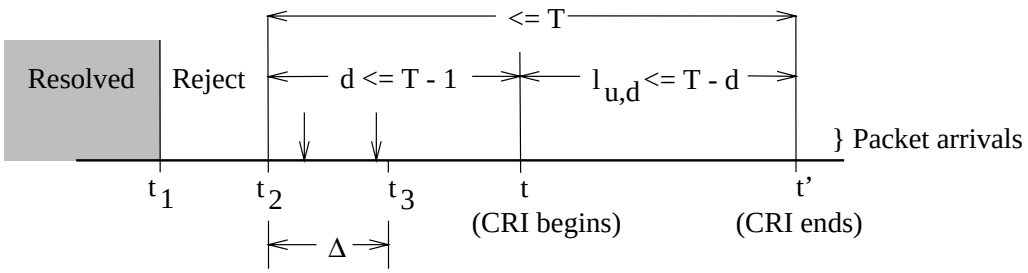


Figure 3.4: Relation between some variables in algorithm.

lag, anything with a smaller laxity than t_2 cannot be transmitted before its deadline and so is rejected and 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'.

3.4 Analysis

This system can be modeled as a finite Markov chain whose state names represent a possible lag value from 1 to T. However, due to the difficulty of directly analyzing this chain, as mentioned in the last chapter, we use the theory of regenerative processes, where a lag of 1 slot in the random multiple access algorithm (RMAA) is the start of a renewal period.

3.4.1 Notation

When possible, we use the notation in [GMPK87], where the usefulness of studying RMAAs with renewal theory is put forth. As mentioned, two considerations vital to successful transmission of a packet are the current lag d and the window width u to be examined. Appropriately, the following variables are subscripted with

this information.

$n_{u,d}$: Number of packets in window of width 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 width 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, our regeneration cycle, 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 the designated variable's value when returning from the current lag of d slots to a lag of one slot. 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, we denote the form of the final type of variable as:

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

Finally, to compute probabilities of a given number of packets arriving in some interval u, we use the Poisson arrival process. This is reasonable because we are modeling real-time systems where the transfer of short data messages is primarily undertaken in an automated manner, and so is bursty in nature.

3.4.2 Model

As mentioned in the introduction of this chapter, we make the simplifying assumption that initial laxities of all packets are identical and equal to T. Considering a single CRI as one cycle of the regenerative stochastic process, we define $Z = E\{\alpha_1\}$ as the number of packets successfully transmitted in a collision resolution interval (CRI). If $H = E\{h_1\}$ is 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

$$\rho = \frac{Z/H}{\lambda}. \tag{3.3}$$

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

$$D = \frac{W}{Z}. \tag{3.4}$$

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

3.4.3 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.

3.4.3.1 Probable Length of CRI

In Figure 3.3, a four way collision and one possible resolution are illustrated. But of all the possible resolutions for a four way resolution, what, for example, is the probability that a CRI will take eleven slots, as in the illustration? Or more generally, what is the probability for *any* multiplicity collision, that a CRI will require eleven slots? The latter question, especially, is important when deriving expressions

for expected values of CRI lengths, number of packets transmitted during a CRI, and other values of interest.

To calculate the probability that a CRI is of some length l, the first step is to consider the arrival process. We need to know the probability that n packets will arrive, and then, secondly, use that in conjunction with the probability that the multiplicity n collision will require exactly l slots to be resolved.

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, based on the arrival process, that the u slots contain 0, 1, ..., ∞ packets and that these packets can be successfully transmitted within the remaining time, $T - d$.

$$P(l \mid u, d) = \sum_{k=0}^{\infty} P(l \mid k, T - \lceil d \rceil) e^{-\lambda u} \frac{(\lambda u)^k}{k!} \tag{3.5}$$

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 with 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 $\binom{k}{i} 2^{-k}$. For $k \geq 2$ and $B \geq 2$, this is recursively expressed as

$$\begin{aligned} P(l \mid k, B) \;=\; & P(l - j \mid i, B - 1) \\ & \cdot P(j - 1 \mid k - i, B - 1 - (l - j)), \qquad \text{with probability } 2^{-k}\binom{k}{i} \end{aligned} \tag{3.6}$$

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

$$P(l \mid k, B) = 2^{-k} \sum_{j=1}^{l-1} \sum_{i=0}^{k} \binom{k}{i} \left[P(l - j \mid i, B - 1) \cdot P(j - 1 \mid k - i, B - 1 - (l - j)) \right] \tag{3.7}$$

with the following initial conditions:

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

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.

3.4.3.2 Expected Length of CRI

With an expression for the probable length of a CRI, it is possible to now derive an expression for the expected length of a CRI. This is the variable H in Equation 3.3.

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 at most 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 $d - \Delta$ 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

$$h_d = \begin{cases} l_{d,d}, & l_{d,d} = 1, & 1 \leq d \leq \Delta \\ l_{d,d} + h_{l_{d,d}}, & 1 < l_{d,d} \leq T - \lceil d \rceil, & 1 \leq d \leq \Delta \\ l_{\Delta,d} + h_{d-\Delta+l_{\Delta,d}}, & & \Delta < d \leq T - 1 \end{cases} \tag{3.8}$$

The expected value, $l_{u,d}$, of a CRI with u slots to be examined and current lag of d is developed shortly. However, a 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 3.4.3.1 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

$$H_d = \begin{cases} E\{l_{d,d}\} + \sum_{m=2}^{T-\lceil d \rceil} H_m P(m \mid d, d), & 1 \leq d \leq \Delta \\ E\{l_{\Delta,d}\} + \sum_{m=1}^{T-\lceil d \rceil} H_{d-\Delta+m} P(m \mid \Delta, d), & \Delta < d \leq T-1 \end{cases} \tag{3.9}$$

The first term is the average number of slots it will take to resolve the current window. The second term in each equation takes into consideration that sometimes it will take one slot, sometimes two slots, etc., on up to the maximum of $T - \lceil d \rceil$ slots, each with their respective probabilities, as developed in the last section. Depending on the case, the resolution removes either d or Δ slots from the unresolved interval. However, it took m slots to achieve that resolution, so the lag is incremented by that much.

This system of equations generated in Equation 3.9 is finite because of the bounded nature of the algorithm. 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. When the system of equations is solved, the variable of interest is H_1. Its value is the expected number of slots required, when the system is at a lag of one slot, to return to a lag of one slot—one renewal interval. In Equation 3.3, the variable H is in fact H_1.

Equation 3.8 is presented in Paterakis *et al.* [PGPK89] as a general description of any bounded CRA. Equation 3.9, the expected value of Equation 3.8, addresses the analytic details specific to the CRA presented in this chapter. 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

$$E\{l_{u,d}\} = \sum_{k=0}^{\infty} E\{\, l_{u,d} \mid k, T - \lceil d \rceil \,\} e^{-\lambda u} \frac{(\lambda u)^k}{k!} \tag{3.10}$$

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 in Equation 3.11 loops through the range of possible CRI lengths, recurses on each, and multiplies each result by the probability as given in Equation 3.7.

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

$$L_{k,B} = \begin{cases} 1, & 0 \leq k \leq 1 \\ 1 + L_{i,B-1} & \text{with probability } \binom{k}{i} 2^{-k},\ k > 1 \\ \quad + \sum_{m=1}^{B-1} P(m \mid i, B-1) L_{k-i,B-1-m} & \end{cases} \tag{3.11}$$

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

$$L_{k,B} = 2^{-k} \sum_{i=0}^{k} \binom{k}{i} \left[1 + L_{i,B-1} + \sum_{m=1}^{B-1} P(m \mid i, B-1) L_{k-i,B-1-m} \right] \tag{3.12}$$

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

3.4.3.3 Successfully Transmitted Packets

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:

$$\alpha_d = \begin{cases} n_{d,d} & l_{d,d} = 1, \quad 1 \leq d \leq \Delta \\ n_{d,d} + \alpha_{l_{d,d}}, & 1 < l_{d,d} \leq T - \lceil d \rceil, \quad 1 \leq d \leq \Delta \\ n_{\Delta,d} + \alpha_{d-\Delta+l_{\Delta,d}}, & \Delta < d \leq T-1 \end{cases} \tag{3.13}$$

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

$$A_d = \begin{cases} E\{n_{d,d}\} + \sum_{m=2}^{T-\lceil d \rceil} A_m P(m \mid d, d), & 1 \leq d \leq \Delta \\ E\{n_{\Delta,d}\} + \sum_{m=1}^{T-\lceil d \rceil} A_{d-\Delta+m} P(m \mid \Delta, d), & \Delta < d \leq T-1 \end{cases} \tag{3.14}$$

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

$$E\{n_{u,d}\} = \sum_{k=0}^{\infty} E\{\, n_{u,d} \mid k, T - \lceil d \rceil \,\} e^{-\lambda u} \frac{(\lambda u)^k}{k!} \tag{3.15}$$

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. Similar to obtaining the

expected CRI length in the previous section, 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.

$$N_{k,B} = \begin{cases} 0, & k = 0, B \geq 1 \\ 1, & k = 1, B \geq 1 \\ N_{i,B-1} + \sum_{m=1}^{B-1} P(m \mid i, B-1) N_{k-i,B-1-m}, & k > 1, \text{with probability } \binom{k}{i} 2^{-k} \end{cases} \tag{3.16}$$

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

$$N_{k,B} = 2^{-k} \sum_{i=0}^{k} \binom{k}{i} \left[N_{i,B-1} + \sum_{m=1}^{B-1} P(m \mid i, B-1) N_{k-i,B-1-m} \right] \tag{3.17}$$

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

3.4.3.4 Experienced Packet Delay

Determining the cumulative delay experienced by successfully transmitted packets is the most complex of the calculations. It is the sum of the delays experienced within the window u, the delays experienced by the current lag, and the delays during the CRI itself. That is,

$$w_d = \begin{cases} \psi_{d,d} + z_{d,d}, & l_{d,d} = 1, & 1 \leq d \leq \Delta \\ \psi_{d,d} + z_{d,d} + w_{l_{d,d}}, & 1 < l_{d,d} \leq T - \lceil d \rceil, & 1 \leq d \leq \Delta \\ \psi_{\Delta,d} + z_{\Delta,d} & & \\ \quad + (d - \Delta) n_{\Delta,d} + w_{d-\Delta+l_{\Delta,d}}, & & \Delta < d \leq T - 1 \end{cases} \tag{3.18}$$

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

$$W_d = \begin{cases} E\{\psi_{d,d} + z_{d,d}\} + \sum_{m=2}^{T-\lceil d \rceil} W_m P(m \mid d, d), & 1 \le d \le \Delta \\ E\{\psi_{\Delta,d} + z_{\Delta,d} + (d-\Delta)n_{\Delta,d}\} & \Delta < d \le T-1 \\ \quad + z_{\Delta,d} + (d-\Delta)n_{\Delta,d}\} + \sum_{m=1}^{T-\lceil d \rceil} W_{d-\Delta+m} P(m \mid \Delta, d) & \end{cases} \tag{3.19}$$

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

$$E\{z_{u,d}\} = \sum_{k=0}^{\infty} E\{\, z_{u,d} \mid k, T - \lceil d \rceil \,\} e^{-\lambda u} \frac{(\lambda u)^k}{k!} \tag{3.20}$$

Considering the initial conditions, 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

$$Z_{k,B} = \begin{cases} 0, & k = 0 \\ 1, & k = 1 \\ (N_{i,B-1} + Z_{i,B-1}) + \sum_{m=1}^{B-1} P(m \mid i, B-1) & k \ge 2, \text{with} \\ \quad [(m+1)N_{k-i,B-1-m} + Z_{k-i,B-1-m}], & \text{probability } \binom{k}{i} 2^{-k} \end{cases} \tag{3.21}$$

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

$$\begin{aligned} Z_{k,B} \;=\; & \sum_{i=0}^{k} \binom{k}{i} \{N_{i,B-1} + Z_{i,B-1} \\ & + \sum_{m=1}^{B-1} P(m \mid i, B-1)[(m+1)N_{k-i,B-1-m} + Z_{k-i,B-1-m}] \Big\} \end{aligned} \tag{3.22}$$

and is a finite expansion.

3.5 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 for a significantly smaller average delay. We call these design parameters `SuccFraction`, denoted by e_1 in equations, for the fraction of packets transmitted successfully, and `AvDelay`, denoted by e_2, for the delay experienced by an average packet.

While it is the application that drives the choices for maximum laxity T, `SuccFraction` and `AvDelay`, the initial window width Δ is a basic design parameter of the protocol itself. That is, the value of Δ is chosen from the optimization of the analysis of the previous section. The window width can be anywhere in $(1, T-1]$. For this range we would like to find the maximum system traffic rate, λ^*, that allows successful transmission of at least `SuccFraction` 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:

$$\lambda^*_{T,e_1} = \sup(\lambda \,:\, \rho_T(\Delta, \lambda) \geq e_1) \tag{3.23}$$

We present results for $\Delta = 3$ though data has been generated for various Δ values which, through observation, encompass the maximum throughputs.

We also conducted a simulation study of this protocol using Opnet [Mil96], 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

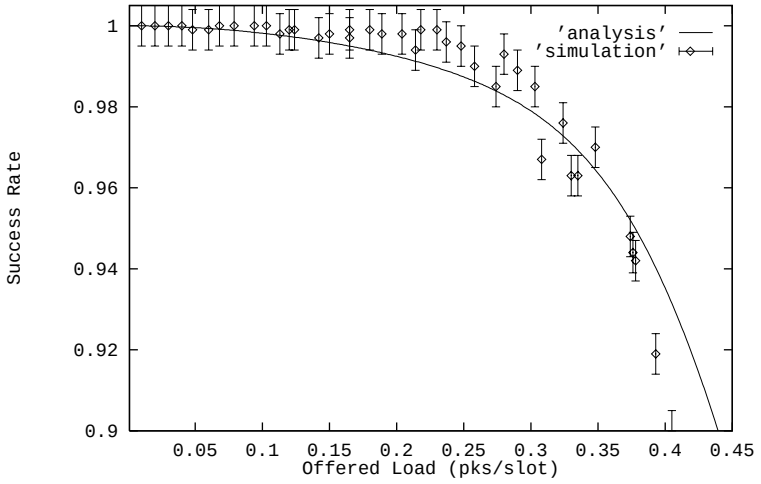


Figure 3.5: Comparison of simulated and analytic results for fraction successful when $\Delta = 3.0$ and $T = 20$

window size $\Delta = 3.0$ and laxity $T = 20$, Figures 3.5 and 3.6 show overlaid graphs of analytic and simulated results. Because of the closeness of results, subsequent graphs do not include simulated data for easier readability. Simulation results, however, have been obtained for all presented graphs and show similar closeness.

In Figure 3.7, three curves are graphed showing the maximum sustainable input traffic rates attainable when window size $\Delta = 3.0$ and the indicated values of `SuccFraction` 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 3.23 above so that an average delay is met in addition to meeting the maximum laxity T, yielding:

$$\lambda^*_{T,e_1,e_2} = \sup(\lambda \,:\, \rho_T(\Delta,\lambda) \geq e_1, D_T(\Delta,\lambda) \leq e_2) \tag{3.24}$$

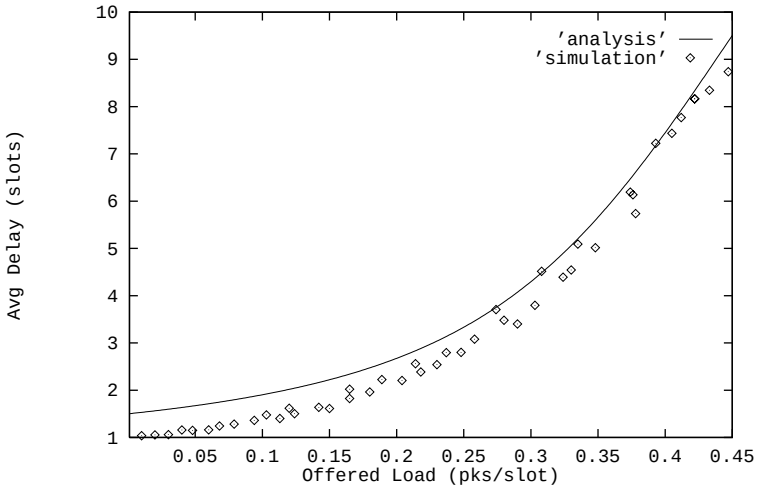


Figure 3.6: Comparison of simulated and analytic results for average delay when $\Delta = 3.0$ and $T = 20$

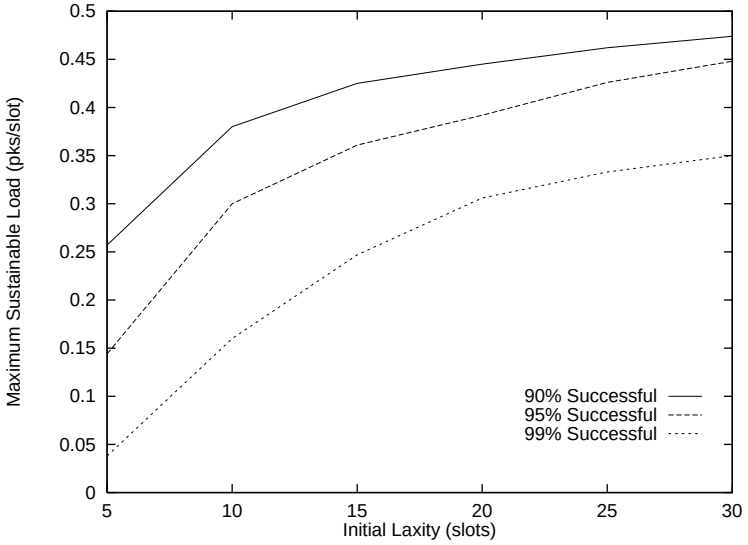


Figure 3.7: Maximum input traffic rate vs initial laxity with minimum success constraints.

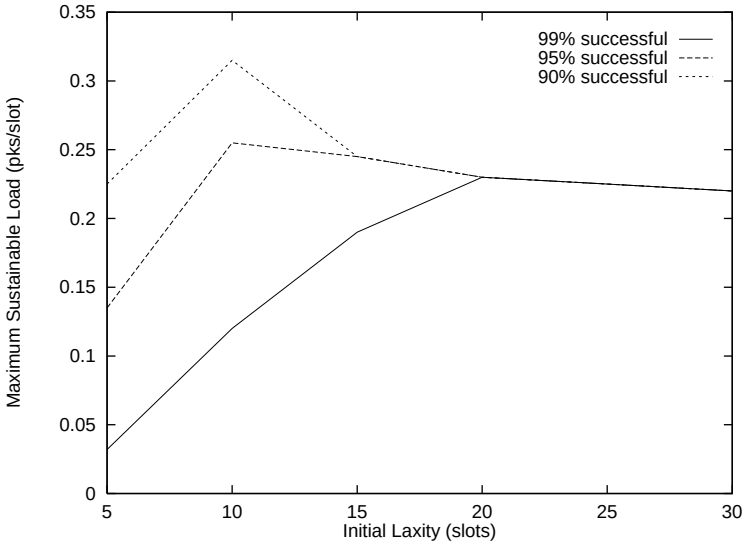


Figure 3.8: Maximum input traffic rate vs initial laxity with `SuccFraction` = $0.99, 0.95, 0.90$, `AvDelay` = 3.

When an average delay of `AvDelay`= 3 slots is met, the results are shown in Figure 3.8. Similar graphs are shown for 5 and 7 slots in Figures 3.9 and 3.10 respectively. All three figures represent performance with window $\Delta = 3.0$. It is interesting to note that with the new constraint `AvDelay`, the curves on each of these graphs converge whereupon the function slowly decreases. This is expected when the traffic rate is increased, hence increasing average delay, but not relaxing the average delay desired.

3.6 Summary

In this chapter we have studied a soft real-time MAC protocol for use in wireless systems. While it is suitable for use in some environments, it is far from general purpose. Recall, however, that the main goals of the work presented are to take the first steps towards the design and analysis of a general real-time MAC protocol. Equally important, if not more so at this stage, is finding a model to characterize

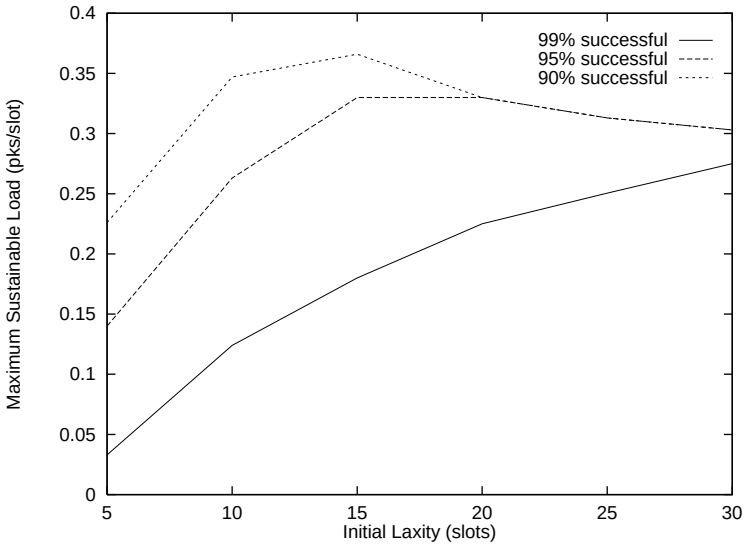


Figure 3.9: Maximum input traffic rate vs initial laxity with `SuccFraction` $= 0.99, 0.95, 0.90$, `AvDelay` $= 5$.

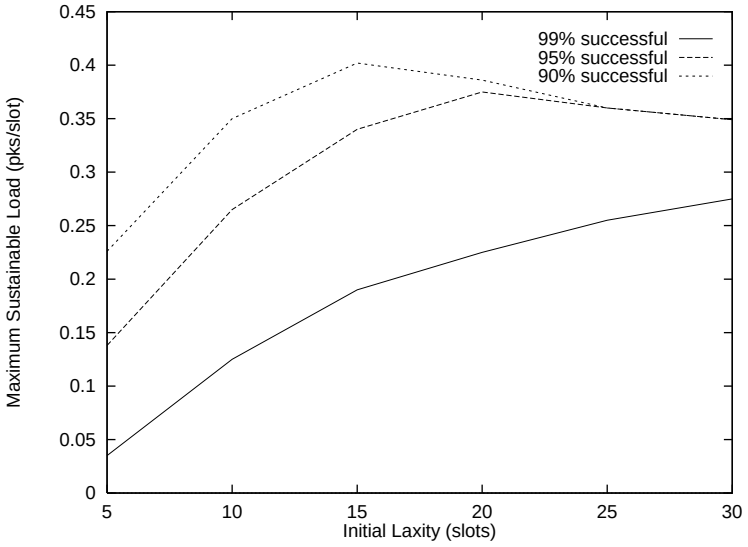


Figure 3.10: Maximum input traffic rate vs initial laxity with `SuccFraction` $= 0.99, 0.95, 0.90$, `AvDelay` $= 7$.

and analyze a real-time MAC protocol. We did this while also presenting a new splitting protocol that makes explicit use of real-time properties.

We find that as initial laxity increases, success rate does as well. However, the increase in success rate is marginal beyond initial laxities of about 25. The small increase suggests that with fixed initial laxities, there is no significant performance advantage in a system that handles wider laxity ranges. And further, for a system requiring an average delay, choices of initial laxity are limited. Maximum throughput is attained for particular initial laxities when an average delay constraint is specified. In order for an application to achieve the highest possible throughput with a desired average delay, the system designer must consider the MAC layer capabilities.

Chapter 4

BLOCKED AND FREE ACCESS REAL-TIME SPLITTING PROTOCOLS

4.1 Introduction

Further addressing the challenge of packet transmission in a wireless soft real-time system, we present five splitting protocols in this chapter that take packet deadlines into account. The studies presented in this chapter have also been published as [MS98c], [MS97a], and [MS97b]. The protocols offer connectionless service with real-time QoS guarantees at the MAC layer. Three of the protocols are blocked access and two are free access algorithms. Mathematical models are developed followed by a comparison between numerical and simulation results.

As in the previous chapter, we again consider the problem of a soft real-time system implemented on a wireless network offering connectionless service. Here, however, we are interested in protocol performance when the arriving traffic has initial laxities drawn from a known distribution. When soft 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. The higher protocol layers, and especially the application layer, should be aware of the general capabilities and limitations of the MAC (media access control) layer. Additionally, the LLC 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 MAC resources for use by other applications or nodes.

Presently, it is commonly 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, or else buffering can be used to somewhat shield user awareness from unpredictable lower layer performance. However, on wireless links that usually offer less bandwidth than wire links, media access that better meets the time constraints of the data is needed to meet functional requirements of system designs. It is not new hardware technology or physical layer improvement that is needed, but new channel access techniques.

4.2 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 a 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. Additionally, an error-free feedback channel is assumed.

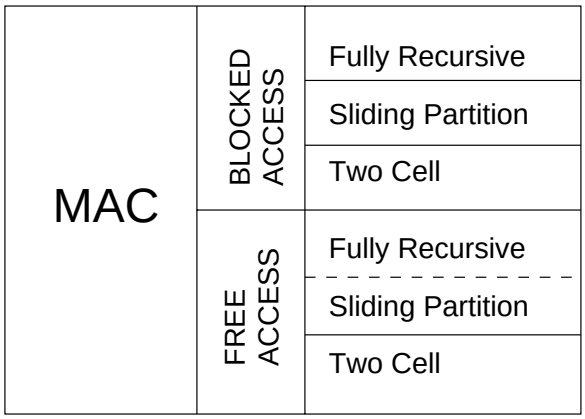
The slotted channel assumption is the most difficult of those listed above to realize in an implementation, and especially so in a fully distributed case. Synchronization, however, is possible with stable clocks, receiver feedback, and guard time between the end of one transmission of the start of the next [BG92]. The assumption of infinite users is not realistic but when used in analysis, provides us with worst case packet delay. Finally, in systems where transmitters are far from each other,

feedback immediately after a data slot is not possible. However, delayed feedback causes no problems for the algorithm. Several interleaved delayed feedback systems can used to make fullest use of the bandwidth.

Two types of random access algorithms (RAAs) are considered: blocked and free access. In a blocked access RAA, after an initial collision between two or more packets occurs, only the nodes whose packets were 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. Regardless of prior collisions, all nodes with packets to transmit always contend for the channel. Free access RAAs, therefore, traditionally have had lower maximum throughputs than blocked access RAAs due to higher contention, but they are easier to implement. In real-time systems, though, the departure process is more complex. In addition to departing due to transmissions, packets can be dropped because of deadline expiration as well. We thought, in light of this, that it would be worthwhile to consider them in a real-time environment. Table 4.1, the MAC portion of Table 1.2, shows which algorithms will be studied in this chapter. Section 4.3 describes blocked access versions, and Section 4.6 describes free access versions of the three soft real-time MAC algorithms. Subsections within each of those major sections describe the particular algorithms of interest.

Blocked access RAAs have two components: the first time transmit rule (FTTR), and the collision resolution algorithm (CRA). The FTTR is used in succession until a collision occurs, at which time the FTTR is deactivated, and 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 feedback from a transmission indicates that a collision occurred, the CRA is initiated. The CRA completes when all packets involved in the initial collision either have been transmitted or dropped

Table 4.1: MAC algorithms studied.



due to missed deadlines. The length of time until the initial collision is resolved is called the collision resolution interval (CRI), and all subsequent analyses are based on one "cycle" of this regenerative stochastic process.

Splitting CRAs use a 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 random number generator or the packet arrival time. In our algorithms, we split into two groups of the same size. However, Wong [Won64] 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 and development of easy to implement algorithms, we do, in fact, split intervals equally in our algorithms.

The splitting variable used is related with the deadline of the packet, namely, its laxity. At each slot boundary, a packet's laxity is decremented by one and the packet dropped 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 system'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 to increase throughput, while individual packets hope for transmission regardless of delay as long as 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.

4.3 Blocked Access CRAs

By the nature of blocked access CRAs, the time when contending packets enter the system is known; it is the start of the CRI. At any time during the CRI, an initial collision that happened previously 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 rate of the arrival process is high, *windows* are used, as first introduced by Gallager [BG92]. With large lags, it is likely that when a CRI completes, it will be immediately followed by a collision. To avoid this, an arrival-time based window Δ slots wide is used to limit channel contention.

Assumptions used for the analysis in this chapter are much the same as in the previous chapter. The window is arrival-time based and allows only those nodes

whose packet arrival times fall within the window to be transmitted. 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 3.4 can again be used as illustration of the association between some of the more important variables in the analysis.

An equivalent approach to using a sliding arrival-time based window, as shown in the illustration, might be to use a sliding laxity window as part of the FTTR. The idea is that even when a CRI is not in progress, a laxity window would be used to allow a subset of the full laxity range to transmit. It turns out, however, that there is no 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. This avoids the added step when using sliding laxity windows of determining optimal window widths for each new laxity set. However, 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, *e.g.,* periodic ones, this can be especially useful.

4.3.1 Blocked Access Fully Recursive CRA

The first algorithm considered is the blocked access Fully Recursive CRA, introduced in Chapter 3. The Fully Recursive algorithm uses packet laxity as the

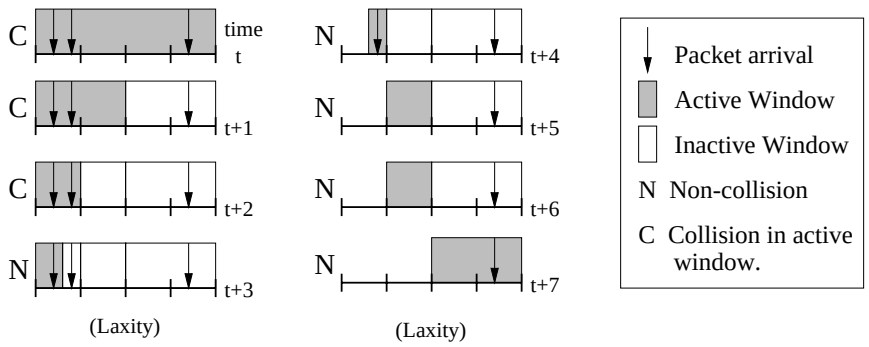


Figure 4.1: Blocked access fully recursive CRA.

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. Use of a sliding window is akin to the modification made by Capetanakis [Mas81] to the original CTM CRA. That is, an arrival-time based window encompasses a 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 in it are waiting to be transmitted. Then the CRA is applied to the right half of the laxity window. An example splitting sequence is illustrated in Figure 4.1, though the initial sliding time window is not shown. With the blocked access Fully Recursive CRA, 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 Figure 4.1. Were a node to begin channel monitoring

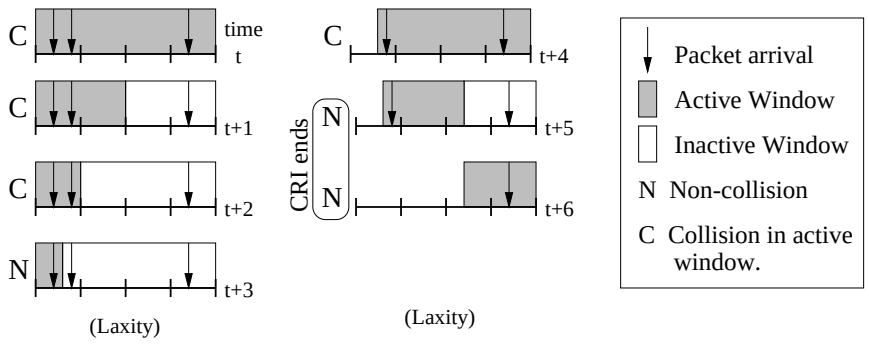


Figure 4.2: Blocked access sliding partition CRA.

at the fourth slot, *i.e.,* time $t + 3$, 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. Hence, the node is unable to know with certainty when it may transmit. While clearly impractical for implementation, it is interesting to consider such a straightforward CRA for the sake of comparison to other real-time CRAs.

4.3.2 Blocked Access Sliding Partition Real-Time CRA

The blocked access Sliding Partition algorithm, illustrated in Figure 4.2, is a more practical algorithm. 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, we use a laxity partition to separate the full range into two halves. Packets in the left half, the active set, then retransmit. If there is a collision, the laxity partition is slid to the midpoint of the current active set. In this way, there are never more than two windows during a CRI, as can be seen in Figure 4.2. And after the left half transmission is a non-collision, the blocked access Sliding Partition CRA is applied to the right half, the unresolved interval. Though the initial collision lets us know that at least one more packet still remains to be transmitted, the memoryless nature of the Poisson arrival process offers no information regarding location or number of additional packets in the unresolved

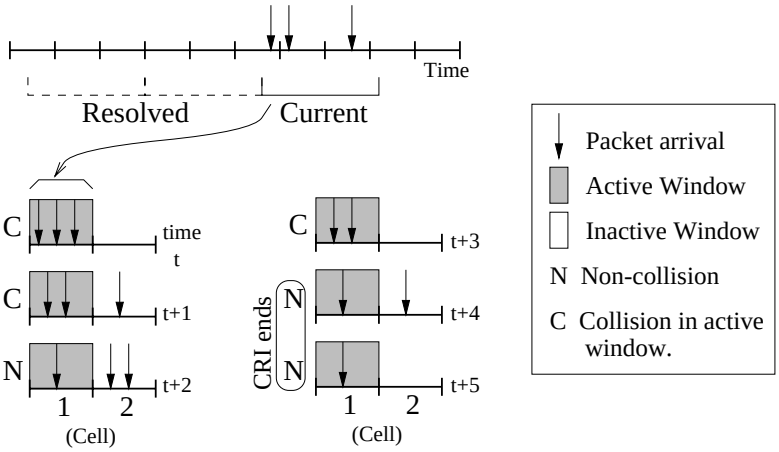


Figure 4.3: Blocked access two cell CRA.

interval. Therefore, there is no gain by performing the more complex recursive splitting. Additionally, having only two groups offers a practical advantage. When both are resolved, the departure process feedback is two consecutive noncollisions, one for each window. Every time this sequence is observed, it indicates that a CRI is no longer in progress. For a node just joining a net, waiting for two consecutive noncollisions is an easy way to know when to activate its own FTTR.

4.3.3 Blocked Access Real-Time Two Cell

Paterakis *et al.* [PGPK89] developed the blocked access Two Cell 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, that the splitting is independent of the arrival process. Figure 4.3 illustrates

the arrival-time based sliding window in the blocked access Two Cell CRA where the current window happens to allow three packets to contend for the channel. After the initial collision, the rest of Figure 4.3 shows how the CRI might proceed given that there is a coin flip by all cell one nodes after a collision. Like the Sliding Partition CRA, two consecutive non-collisions signal the CRI completion. This easy to recognize event makes it easy for nodes to join the net any time after system startup, and to resynchronize in the inevitable case of feedback errors.

4.4 Blocked Access Analysis

The top level performance measures are the same as those of the previous chapter, and we follow the same notational scheme when possible. While expressions have been derived for all protocols studied, we present the analysis only for the blocked access Sliding Partition CRA, which turns out to be the best performing algorithm. We also present results for the other protocols studied. 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 can be ultimately transmitted. Repeating Equations 3.3 and 3.4, as an aid to the reader, 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

$$\rho = \frac{Z/H}{\lambda} \tag{4.1}$$

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

$$D = \frac{W}{Z} \tag{4.2}$$

As before, 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 the same, somewhat complicated counters as before. 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. Characterizing the events during an epoch of this system, because of the more realistic initial packet laxities, is more complicated than the case of fixed initial laxities.

4.4.1 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. Note, however, that this does not mean application layer packets cannot be delivered late. At the LLC layer, deadlines in units of seconds or minutes are mapped into deadlines of slots. Nothing prevents that mapping from introducing late arrival. The MAC layer laxity window is illustrated in Figure 4.4. 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

$$p_1 = P(\text{pk drop} \mid T, B, x, w) = 1 - \frac{\max(1, x + w - (T - B)) - \max(1, x - (T - B))}{w} \tag{4.3}$$

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

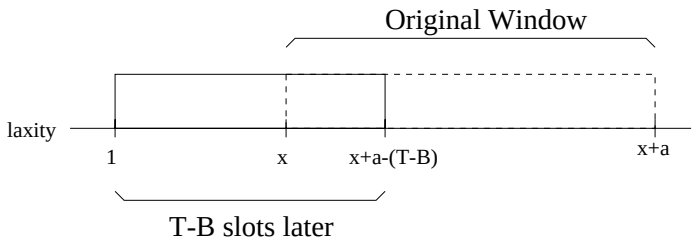


Figure 4.4: Sliding laxity window.

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 characterized by a binomial distribution,

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

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

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

4.4.2 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. The latter probability is derived in this section. For a collision involving a 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 is needed over the number of packets lost multiplied by the probability of the likelihood of such an event. Note that the probability of packet drop in the inactive window with the k_2 packets is not considered. This is unnecessary because while in the inactive window, the packets cannot contend for the channel and hence have no effect on throughput. Once the inactive window becomes the active window, however, the drop probability is then calculated. While it would not be incorrect to model packet dropping in the inactive window, it has no effect on the results, and unnecessarily increases the numerical analysis code by an order of magnitude. Thus, the full expression is,

$$
\begin{aligned}
&P_{len}(l \mid k_1, k_2, B, x, w_1, w_2) \qquad (4.4)\\
&= \begin{cases}
\sum_{d=0}^{k_2} P_{len}(l-1 \mid k_2 - d, 0, B-1, x+w_1, w_2, 0) & \\
\quad \cdot P_{drop}(k_2, d \mid T, B, x+w_1, w_2) & k_1 = 0, 1 \\
\\
\sum_{i=0}^{k_1} \mathrm{bin}(w_1/(2w_1+2w_2), k_1, i) \sum_{d=0}^{i} & \\
\quad P_{len}(l-1 \mid i-d, k_2+k_1-i, B-1, x, w_1/2, w_2+w_1/2) & \\
\quad \cdot P_{drop}(i, d \mid T, B, x, w_1/2) & \text{otherwise}
\end{cases}
\end{aligned}
$$

with initial conditions

$$P_{len}(l = 1 \mid k_1, k_2, B, x, w_1, w_2)$$

$$
= \begin{cases}
0, & \text{if } l > B \\
0, & \text{if } k_1 = k_2 = 0 \\
1, & \text{if } B = 1 \\
1, & \text{if } k_1 = 1, k_2 = 0 \\
P_{drop}(k_2, k_2 \mid T, B, x + w_1, w_2), & \text{if } k_1 \leq 1, k_2 > 0 \\
P_{drop}(k_1, k_1 \mid T, B, x, w_1), & \text{if } k_1 > 1, k_2 = 0 \\
P_{drop}(k_2, k_2 \mid T, B, x + w_1, w_2), & \text{if } k_1 = 1, k_2 > 0 \\
P_{drop}(k_1, k_1 \mid T, B, x, w_1) & \\
\quad \cdot P_{drop}(k_2, k_2 \mid T, B, x + w_1, w_2), & \text{if } k_1 > 1, k_2 > 0
\end{cases}
$$

To improve readability, note that initial condition specifications are read sequentially. That is, consider the first two initial conditions above. The result is zero when $l > B$ regardless of other variable values. This implies in the second initial condition that the outcome is again zero if $k_1 = k_2 = 0$, *and* if $l \leq B$. In the first case of the recursion, 0 or 1 packets were in the active window. Accordingly, in preparation for the next slot, the k_1 and k_2 parameters show that the inactive window containing k_2 packets now becomes the active window. For the moment, there is an inactive window of zero width. The second case of the equation, though, shows what happens after a collision. Of the $k_1 > 1$ packets that collide in the active window, each possible split must be considered with each multiplied by the probability of such a split.

The initial conditions are much more involved than the corresponding ones of the previous chapter when packets with fixed initial laxities were used. The first four, though, are straightforward enough. If the limit has expired or if there are no packets to transmit, the CRI can clearly not be the length of interest. If the bound is one, or if there is one packet to transmit, then the probability of the CRI being one slot long is one. For the others, more care must be taken. The only way that a CRI can complete in one slot when there is more than one packet to transmit is if

all deadlines happen to expire. That is the set of events that the latter four initial conditions capture.

The above gives the probability based on specific initial values of k_1 and k_2. More often, we are not interested in particular cases but want the overall probability that the CRI will be of a length l given the current lag and the current window width. 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, the current window, 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, $T - d$. Since these are transmissions made while the FTTR is active, the laxity window is at its full width, $[2, T)$. Also because of the FTTR transmission, there is no inactive window, though the lag might be more than one if the system is not yet at a renewal point. This is expressed below where one can also see the Poisson arrival probability. To consider other arrival processes, this term would be changed to reflect the process of interest.

$$P_{len}(l \mid u, d) = \sum_{k=0}^{\infty} P_{len}(l \mid k, 0, T - \lceil d \rceil, 2, T - 1, 0) e^{-\lambda u} \frac{(\lambda u)^k}{k!} \tag{4.5}$$

4.4.3 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,w_2}. We again must consider the probabilities of one

or more of the k packets expiring as the CRI progresses. For $k > 1$,

$$L_{k_1,k_2,B,x,w_1,w_2} = \begin{cases} 1 + \sum_{d=0}^{k_2} L_{k_2-d,0,B-1,x+w_1,w_2,0} & \\ \quad \cdot P_{drop}(k_2, d \mid T, B, x + w_1, w_2) & k_1 \leq 1 \\ & \\ 1 + \sum_{i=0}^{k_1} \text{bin}(w_1/(2w_1 + 2w_2), k_1, i) \sum_{d=0}^{i} & \\ \quad L_{i-d,k_2+k_1-i,B-1,x,w_1/2,w_2+w_1/2} & \\ \quad \cdot P_{drop}(i, d \mid T, B, x, w_1/2) & \text{otherwise} \end{cases} \tag{4.6}$$

and with initial conditions $L_{\ldots B \ldots} = 1$, if $B = 1$, and $L_{k_1,k_2,\ldots} = 1$, if $k_1 \leq 1, k_2 = 0$.

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

$$E\{l_{u,d}\} = \sum_{k=0}^{\infty} E\{\, l_{u,d} \mid k, T - \lceil d \rceil\, 2, T - 1, 0\} e^{-\lambda u} \frac{(\lambda u)^k}{k!} \tag{4.7}$$

The effect of lag d on the expression is that it reduces the number of slots remaining until the CRA resets and involved packets are discarded. The effect of window width u is, as its width increases, to increase the likelihood of more arrivals. With these expressions, we can now use the high level relationship induced by time-constrained blocked access RAAs, expressed in Equation 3.9, to calculate a value for H_1, the expected number of slots it takes to leave and return to a system state of a one slot lag.

4.4.4 Expected Number of Transmissions During CRI

In a similar manner, an expression for the number of packets transmitted during a CRI can be derived. However, where the expected length expression counts the number of slots in the CRI, the expected number of transmissions counts only packets. After each non-collision, the counter is incremented by the number transmitted;

either zero or one.

$$N_{k_1,k_2,B,x,w_1,w_2} = \begin{cases} k_1 + \sum_{d=0}^{k_2} N_{k_2-d,0,B-1,x+w_1,w_2,0} \\ \quad \cdot P_{drop}(k_2, d \mid T, B, x + w_1, w_2) & k_1 \leq 1 \\ \\ \sum_{i=0}^{k_1} \text{bin}(w_1/(2w_1 + 2w_2), k_1, i) \sum_{d_1=0}^{i} \\ \quad N_{i-d_1,k_2+k_1-i,B-1,x,w_1/2,w_2+w_1/2} \\ \quad \cdot P_{drop}(i, d_1 \mid T, B, x, w_1/2) & \text{otherwise} \end{cases} \tag{4.8}$$

with initial conditions $N_{k_1,k_2,\ldots} = 0$, if $k_1 = k_2 = 0$, $N_{k_1,k_2,\ldots} = 1$, if $k_1 = 1, k_2 = 0$, and $N_{\ldots B \ldots} = 0$, if $B = 1$. Notice that the value of N is incremented only after a noncollision that contained 1 packet, a successful transmission.

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

$$E\{n_{u,d}\} = \sum_{k=0}^{\infty} E\{\, n_{u,d} \mid k, T - \lceil d \rceil \,\} e^{-\lambda u} \frac{(\lambda u)^k}{k!} \tag{4.9}$$

Substituting the above expressions into Equation 3.14 lets us calculate $Z = A_1$, the expected number of packets transmitted during a CRI. With the expressions of previous sections for derivation of H, we can now get values for transmission success rate, ρ, in Equation 4.1.

4.4.5 Expected Cumulative Delay During CRI

The cumulative delay, Z, of all packets transmitted during the CRI is obtained in a more complicated manner than CRI length or number of packets transmitted. This expression must count the number of slots spent waiting, but only by packets that are eventually transmitted. The most straightforward way of doing this is to augment the expression for N. The number of packets ultimately transmitted, due to our assumption of having one slot long packets, also happens to be the number of slots required to transmit them. In addition, though, other waiting packets are

delayed by packet transmissions occurring earlier in the CRI, hence the two terms in the recursion:

$$Z_{k_1,k_2,B,x,w_1,w_2} = \begin{cases} k_1 + \sum_{d=0}^{k_2}(Z_{k_2-d,0,B-1,x+w_1,w_2,0} + N_{k_2-d,0,B-1,x+w_1,w_2,0}) \\ \quad \cdot P_{drop}(k_2, d \mid T, B, x+w_1, w_2) & k_1 = 0, 1 \\ \\ \sum_{i=0}^{k_1} \mathrm{bin}(w_1/(2w_1+2w_2), k_1, i) \sum_{d=0}^{i} \\ \quad (Z_{i-d,k_2+k_1-i,B-1,x,w_1/2,w_2+w_1/2} \\ \quad + N_{i-d,k_2+k_1-i,B-1,x,w_1/2,w_2+w_1/2}) \\ \quad \cdot P_{drop}(i, d \mid T, B, x, w_1/2) & \text{otherwise} \end{cases} \tag{4.10}$$

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

$$E\{z_{u,d}\} = \sum_{k=0}^{\infty} E\{\, z_{u,d} \mid k, T - \lceil d \rceil \,\} e^{-\lambda u} \frac{(\lambda u)^k}{k!} \tag{4.11}$$

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. These are used in Equation 3.19. The solution for the variable $W = W_1$ represents the expected cumulative delay of the packets transmitted during a CRI. With W calculated, the per packet delay can be obtained using Equation 4.2.

4.5 Blocked Access Evaluation

In Chapter 3, performance of the blocked access Fully Recursive CRA when arriving traffic had the same initial laxity was studied based on success rate, average delay, and worst case delay. We evaluate the blocked access Fully Recursive, Sliding Partition, and Two Cell CRAs in the same manner here, though arriving traffic has laxities drawn from a uniform distribution.

The soft real-time protocols studied in the previous sections can be characterized by at least the following: traffic rate λ, laxity range $[2, T]$, minimum acceptable success ratio `SuccFraction`, denoted by e_1 in equations, and average delay desired `AvDelay`, denoted by 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.

For a given minimum acceptable success rate `SuccFraction`, the initial sliding time-based window of width Δ yields different results. The highest success rates are obtained when the best value for Δ is used.

$$\lambda^*_{T,e_1} = \sup(\lambda \,:\, \rho_T(\Delta, \lambda) \geq e_1) \tag{4.12}$$

We have generated data for a range of Δ values and through observation, $\Delta = 2.5$ is best. Therefore, we present results only for that value. The simulation studies of each protocol were conducted using an Opnet [Mil96] model. The simulation models the system as an infinite user population, thus offering more conservative performance results than the finite user case [PGPK87]. Each simulation was run with 0.95 confidence 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 simulation is not numerically solving the recursions of the analysis. The simulation performs the various

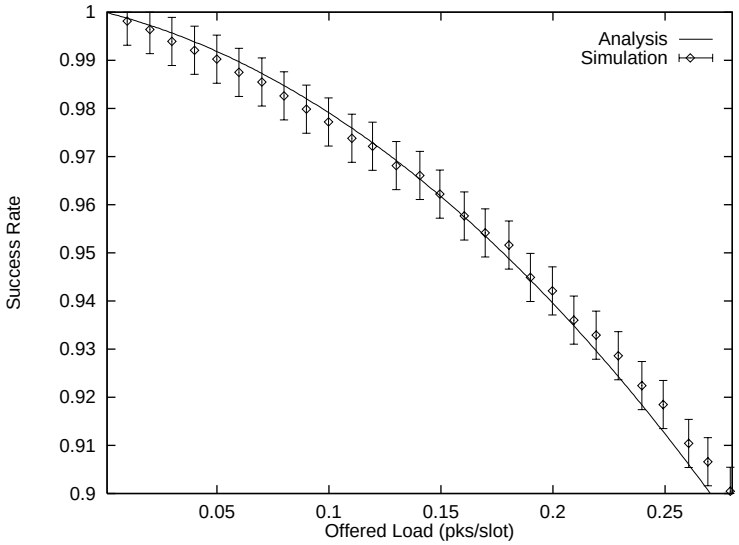


Figure 4.5: Blocked access Sliding Partition success rate.

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 4.1 and 4.2.

For the blocked access Sliding Partition CRA, Figure 4.5 graphs analytic and simulation success rate results for $\lambda^*_{10,0.9}$ and illustrates the closeness of data, even though the two methods used to acquire the performance results are very different. Figure 4.6 graphs a similar comparison again using 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 agreement 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. Analytic results for the Sliding Partition algorithm are graphed in Figure 4.7, showing expected CRI lengths for various values of T and λ.

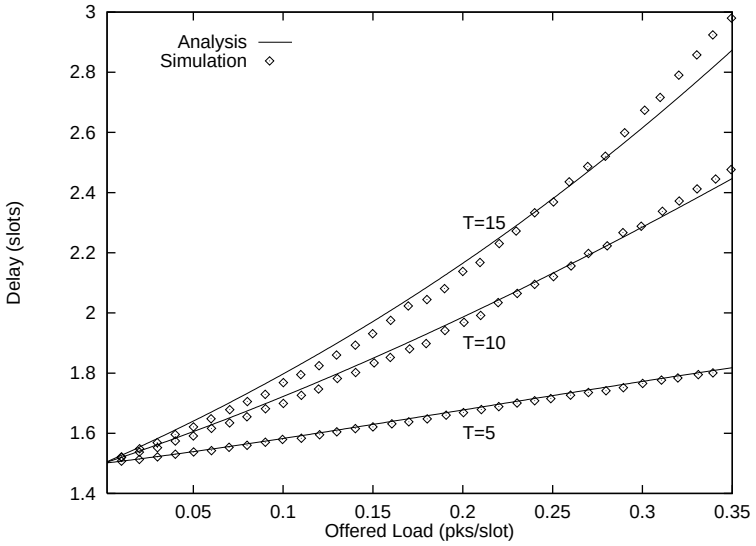


Figure 4.6: Blocked access Sliding Partition delay.

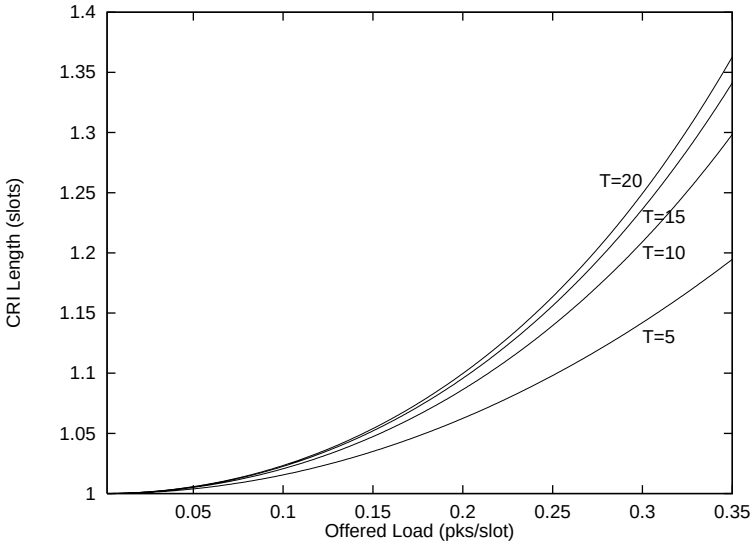


Figure 4.7: Blocked access Sliding Partition CRI lengths.

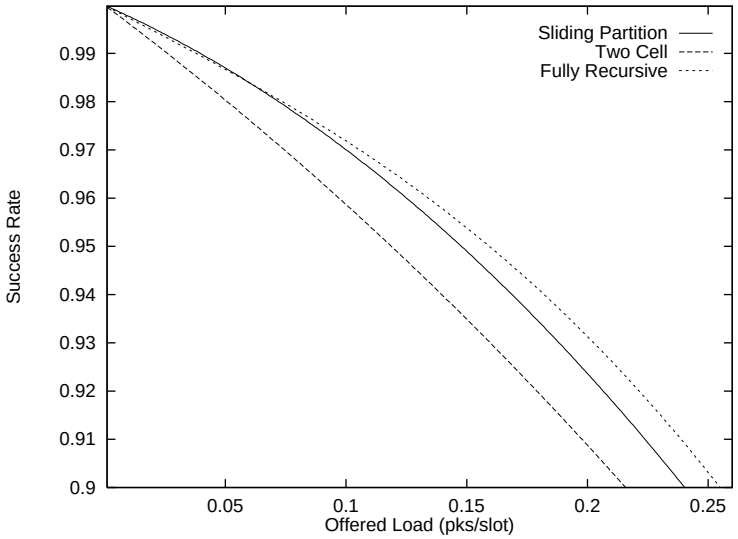


Figure 4.8: Blocked access protocols, $T = 10$.

As expected, when the arrival intensity is high, CRI length increases accordingly, and especially when T is large.

For maximum initial laxity $T = 10$, analytic results for the three protocols are compared in Figure 4.8. 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 an advantage. However, when $T = 30$, the advantage is lost as illustrated in Figure 4.9. While splitting finely allows somewhat 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 4.10. Furthermore, as seen in Figure 4.11, the

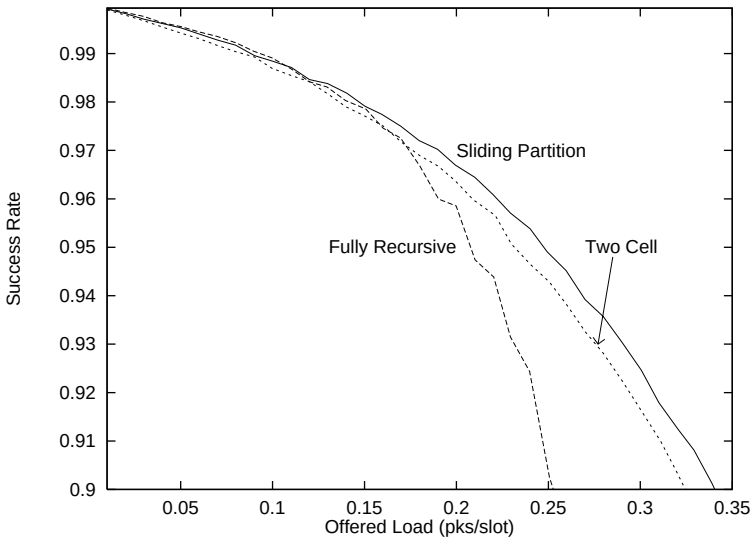


Figure 4.9: Blocked access protocols, $T = 30$.

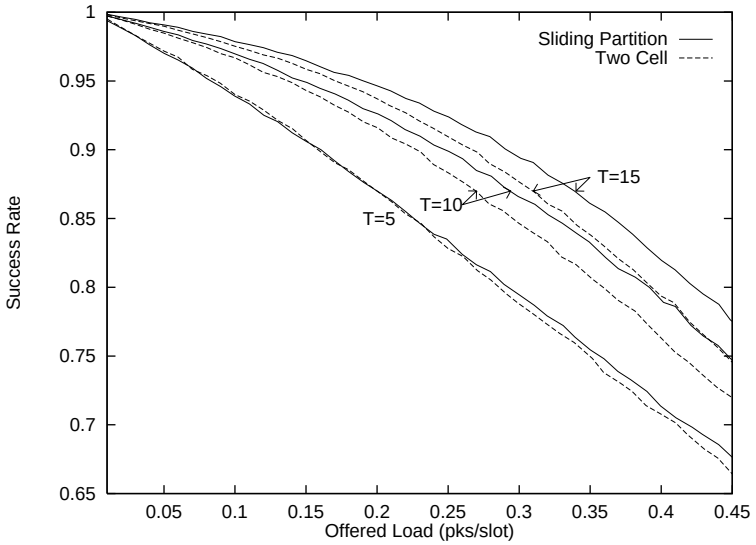


Figure 4.10: Blocked access protocols, $T = 5, 10, 15$.

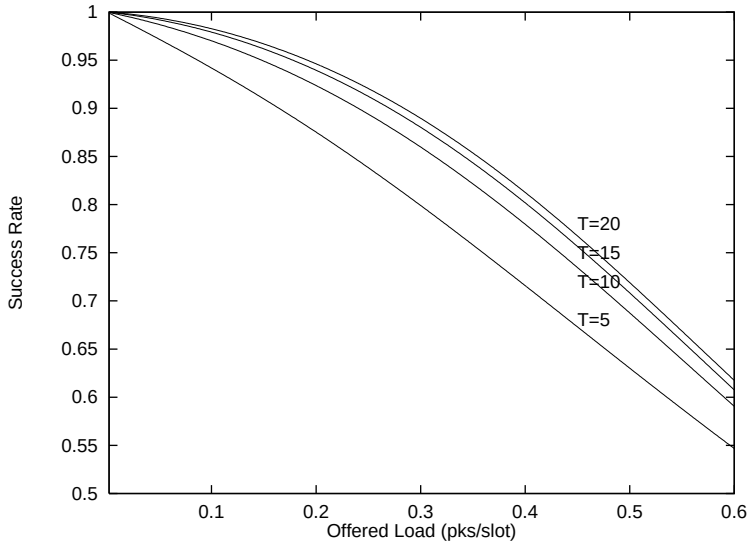


Figure 4.11: Blocked access Sliding Partition, $T = 5, 10, 15, 20$.

CRA is seen to approach an 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. Conversion of LLC deadlines in units of seconds to MAC deadlines in units of slots will be affected by the value of T. This is in turn will affect the design of the LLC scheduling algorithm.

4.6 Free Access CRAs

Because free access CRAs allow nodes with 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 a nonzero probability that the arrival process will add zero or more packets to the system, and a nonzero 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.

4.6.1 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 making the left window edge immovable, regardless of the state of a CRI, lower laxity packets are always admitted. Because splitting still occurs and affects the right window edge, it is not permissible at all times to transmit higher laxity packets. 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 splitting algorithm, the free access variant of the Fully Recursive algorithm is identical to the free access Sliding Partition CRA. (The dashed line in Tables 1.2 and 4.1 indicates that there is no distinction between the two algorithms because they are, in fact, the same.)

4.6.2 Free Access 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. As a result, the free access Two Cell algorithm is more hybrid in nature; partially blocked access, partially free access. But as mentioned in the Sliding Partition algorithm discussion above, real-time protocols, free access or not, have to impose deadline-based limits.

4.7 Free Access Analysis

For free access analysis, the expressions for H, Z, and W must again be derived for use in Equations 4.1 and 4.2. 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. Since a CRI is not guaranteed to be shorter than the largest initial laxity, as in the blocked access case, the notation changes. 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.

4.7.1 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 in 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:

$$\begin{aligned} p_1 &= P(\text{pk drop} \mid T, P, x, w) \qquad (4.13) \\ &= \frac{1}{\min(P, T-1)} \sum_{i=2}^{\min(P,T-1)} \frac{\max(1, x+w-(T-i)) - \max(1, x-(T-i))}{w} \end{aligned}$$

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

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

4.7.2 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 during the past slot (of width 1), while the CRA determines the likelihood of departures. With these probabilities, the expected CRI length is expressed as

$$L_{k_1,k_2,P,x,w_1,w_2} = \begin{cases} 1 + \sum_{a=0}^{\infty} \sum_{d=0}^{k_2} L_{k_2-d+a,0,P+1,w_1+w_2,0} & \\ \quad \cdot P_{drop}(k_2, d \mid T, P, w_1 + w_2) e^{-\lambda} \frac{\lambda^a}{a!}, & k_1 \leq 1 \\ & \\ 1 + \sum_{i=0}^{k_1} \text{bin}(w_1/(2w_1 + 2w_2), k_1, i) \sum_{d_1=0}^{i} \sum_{d_2=0}^{k_2+k_1-i} & \\ \quad \sum_{a_1=0}^{\infty} \sum_{a_2=0}^{\infty} L_{i-d_1+a_1,k_2+k_1-i-d_2+a_2,P+1,w_1/2,w_2+w_1/2} & \\ \quad \cdot P_{drop}(i, d_1 \mid T, P, x, w_1/2) & \\ \quad \cdot P_{drop}(k_2 + k_1 - i, d_2 \mid T, P, x + w_1/2, w_2 + w_1/2) & \\ \quad \cdot e^{-\lambda} \frac{\lambda^{(a_1+a_2)}}{(a_1+a_2)!} \text{bin}(w_1/(w_1 + w_2), a_1 + a_2, a_1) & \text{otherwise} \end{cases} \tag{4.14}$$

The first case represents a noncollision in the active window; either zero or one packets are transmitted. Before the inactive window becomes the active one, however, several things must be considered. During the slot just passed, it is possible that a number of packets arrive. This is represented by the summation over a, where a is the number of new arrivals from a Poisson process in the past slot. Similarly, the summation over d represents the number of waiting packets in the inactive group that expired and were dropped during the previous slot.

The same considerations must be used after a collision, the second case above. Here, though, there is the added consideration of which laxity window half a new packet arrives in, as well as which half a packet might depart from. Hence, the four summations for arrivals and departures from the two halves.

4.7.3 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 in the one slot width, while the CRA determines the likelihood of departures. With these probabilities, the expected CRI length is expressed as

$$Z_{k_1,k_2,P,w_1,w_2} = \begin{cases} k_1 + \sum_{a=0}^{\infty} \sum_{d=0}^{k_2} Z_{k_2-d+a,0,P+1,w_1+w_2,0} & \\ \quad \cdot P_{drop}(k_2, d \mid T, P, w_1+w_2) e^{-\lambda} \frac{\lambda^a}{a!}, & k_1 \leq 1 \\ \\ \sum_{i=0}^{k_1} \text{bin}(w_1/(2w_1+2w_2), k_1, i) \sum_{d_1=0}^{i} \sum_{d_2=0}^{k_2+k_1-i} & \\ \quad \sum_{a_1=0}^{\infty} \sum_{a_2=0}^{\infty} Z_{i-d_1+a_1,k_2+k_1-i-d_2+a_2,P+1,w_1/2,w_2+w_1/2} & \\ \quad \cdot P_{drop}(i, d_1 \mid T, P, w_1/2) & \\ \quad \cdot P_{drop}(k_2+k_1-i, d_2 \mid T, P, w_1/2, w_2+w_1/2) & \\ \quad \cdot e^{-\lambda} \frac{\lambda^{(a_1+a_2)}}{(a_1+a_2)!} \text{bin}(w_1/(w_1+w_2), a_1+a_2, a_1) & \text{otherwise} \end{cases} \tag{4.15}$$

The only difference between this expression and Equation 4.14 is that the value of Z is only incremented after a noncollision due to a successful transmission.

4.7.4 Expected Cumulative Delay

The expected cumulative packet delay during a CRI is slightly more complex. In addition to awaiting transmission during the CRI, an arriving packet waits another half slot on average before receiving its collision/noncollision feedback. Therefore, the expected delay is

$$W_{k_1,k_2,P,w_1,w_2} = \begin{cases} k_1/2 + \sum_{a=0}^{\infty} \sum_{d=0}^{k_2} (W_{k_2-d+a,0,P+1,w_1+w_2,0} \\ \quad Z_{k_2-d+a,0,P+1,w_1+w_2,0}) \\ \quad \cdot P_{drop}(k_2, d \mid T, P, w_1+w_2) e^{-\lambda} \frac{\lambda^a}{a!}, & k_1 \leq 1 \\ \\ \sum_{i=0}^{k_1} \mathrm{bin}(w_1/(2w_1+2w_2), k_1, i) \sum_{d_1=0}^{i} \sum_{d_2=0}^{k_2+k_1-i} \\ \quad \sum_{a_1=0}^{\infty} \sum_{a_2=0}^{\infty} (W_{i-d_1+a_1,k_2+k_1-i-d_2+a_2,P+1,w_1/2,w_2+w_1/2} \\ \quad + Z_{i-d_1+a_1,k_2+k_1-i-d_2+a_2,P+1,w_1/2,w_2+w_1/2}) \\ \quad \cdot P_{drop}(i, d_1 \mid T, P, w_1/2) \\ \quad \cdot P_{drop}(k_2+k_1-i, d_2 \mid T, P, w_1/2, w_2+w_1/2) \\ \quad \cdot e^{-\lambda} \frac{\lambda^{a_1+a_2)}}{(a_1+a_2)!} \mathrm{bin}(w_1/(w_1+w_2), a_1+a_2, a_1) & \text{otherwise} \end{cases} \tag{4.16}$$

Like Equation 3.21 of the last chapter, cumulative delay is determined by augmenting the expression for expected number of packets transmitted during a CRI. In addition to counting packets, the number of slots spent waiting by all packets prior to transmission are also counted. Finally, with values for Z, H, and W, the performance evaluation expressions in Equations 4.1 and 4.2 can be used.

4.8 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,

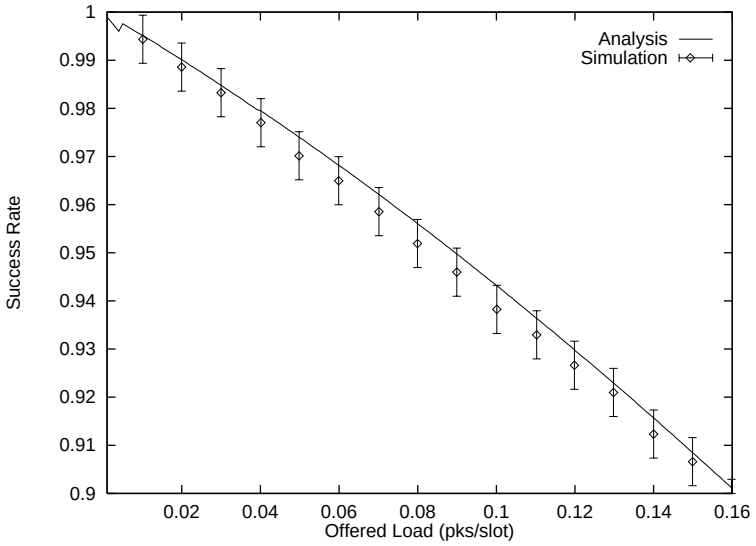


Figure 4.12: Free access Sliding Partition success rate.

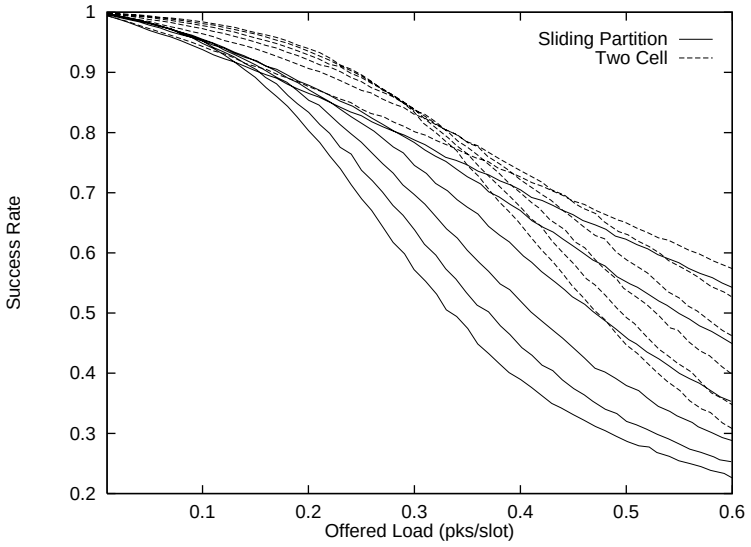


Figure 4.13: Free access Sliding Partition and Two Cell success rates.

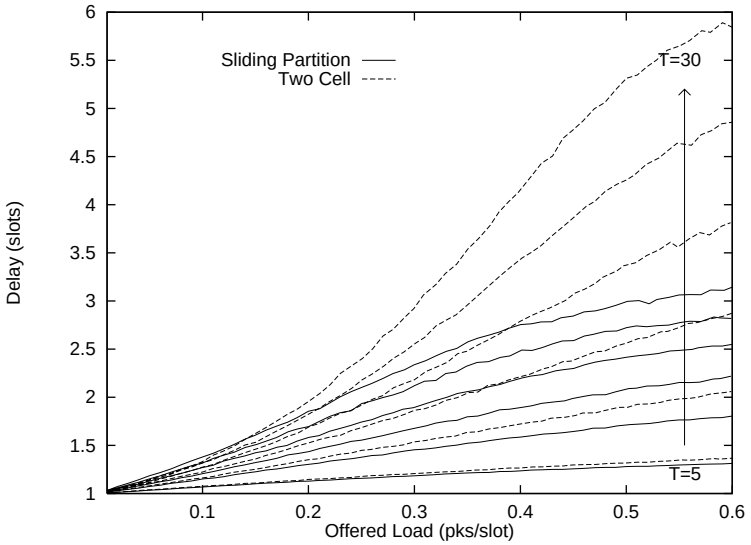


Figure 4.14: Free access Sliding Partition and Two Cell delays.

while Figure 4.12 shows the closeness between the analytic 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 4.13. 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 4.14 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

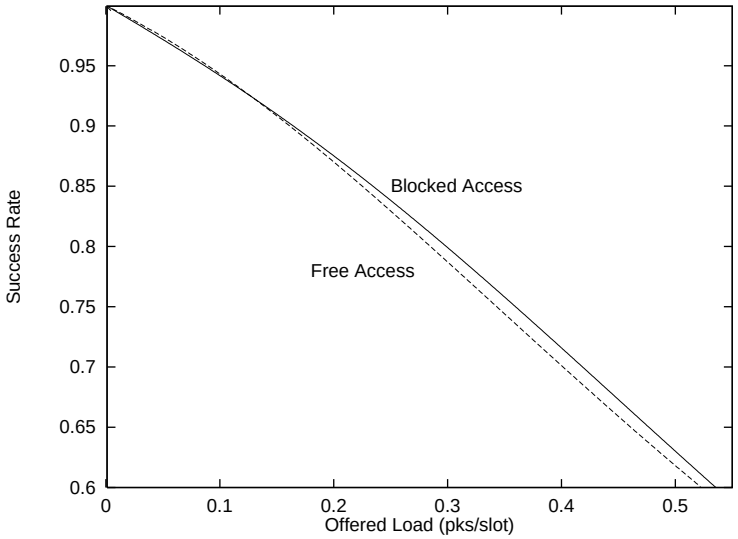


Figure 4.15: Blocked and free access Sliding Partition success rates, $T = 5$.

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, they have lower throughput than blocked access CRAs 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, the two types of algorithms perform similarly. Figure 4.15 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 4.16 graphs success rates for initial laxities in $[2, T]$. We can see that as T becomes larger, the blocked access version of the Sliding Partition algorithm performs increasingly better, whereas the free access version performs increasingly worse. The more practical range of success rates for `SuccFraction`$= 0.9$ is illustrated in Figure 4.17. 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 as T becomes larger, it

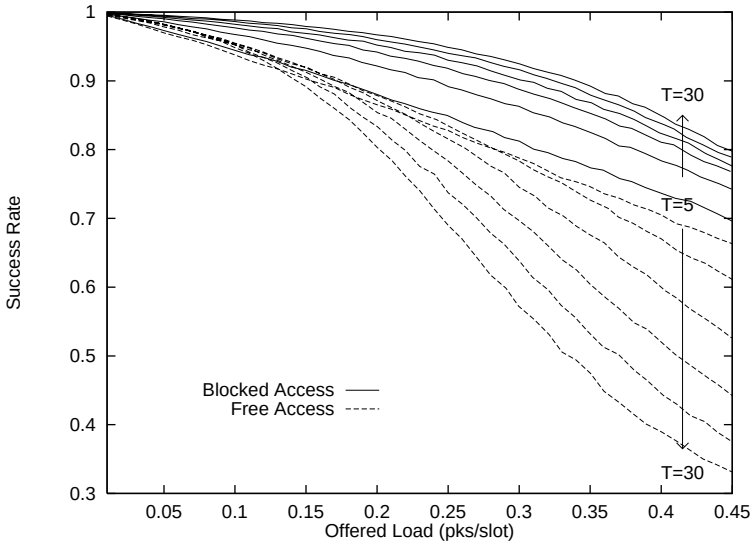


Figure 4.16: Blocked and free access Sliding Partition success rates, $T = 5 \ldots 30$.

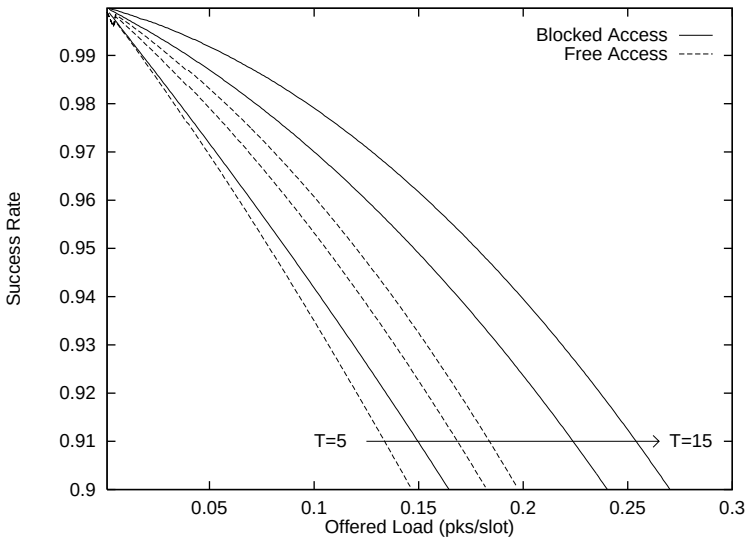


Figure 4.17: Blocked and free access Sliding Partition, `SuccFraction`= .9.

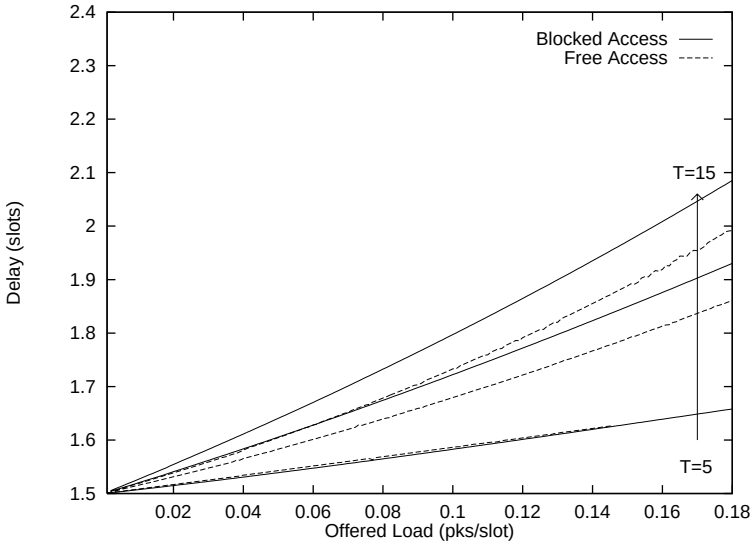


Figure 4.18: Blocked and free access Sliding Partition delay, `SuccFraction`= .9.

is easily seen that even in this restricted range of success rates, the blocked access Sliding Partition CRA performs better. The corresponding delay curve is graphed in Figure 4.18.

4.9 Summary

In this chapter, we have studied the blocked and free access versions of the Fully Recursive, Sliding Partition, and Two Cell CRAs. Recall, though, that the free access Fully Recursive CRA and the free access Sliding Partition CRA reduce to the same algorithm, thus yielding five, rather than six, algorithms. Presenting these five algorithms, we have shown that splitting protocols can be modified to work successfully in soft real-time environments more general than assumed both in the previous chapter as well as the published literature. In addition to development of the algorithms, we derived analytic models of them that can be used to predict performance levels and hence enable us to offer QoS guarantees to the LLC layer. Supporting the analytic techniques, simulations of the algorithmic actions yield

results 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.

It is important to consider not only relative performances of the protocols, but also issues regarding eventual implementation. While the Fully Recursive algorithm performs well in some situations, the CRA cannot be implemented since it requires both an error free channel and that all nodes 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 high bandwidth streaming data, whether continuous or variable bit rate, 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.

Chapter 5

WIRELESS MAC TRANSMISSION OF HARD, SOFT AND NON REAL-TIME DATA

5.1 Introduction

Because wireless links have a much higher error rate than wire or cable links, wireless networks are usually not a first choice for implementation of real-time systems. However, when real-time performance is required by mobile users, wireless networks are the only option. While implementing real-time systems on cellular networks is sometimes an option due to the wide availability of the networks, there are situations when use of a cellular network is not a feasible solution. An extreme case is digital communication on the battlefield. There, not only is long term resource allocation as provided by cellular protocols not required, but more importantly, base stations do not exist. In fact, a centralized system in a military setting is undesirable due to a single point of failure, making the central node an unenviable center of attention.

To honor the deadlines and real-time classes associated with the data, a fully decentralized protocol is required that provides real-time services. In their most general forms, wireless real-time systems will have hard, soft and non real-time data, which must coexist on the same, often noisy, channel. In this chapter, we present a wireless MAC layer protocol that provides a flexible means for the coexistence of these three classes of real-time data along with appropriate quality of service guarantees for hard, soft and non real-time data. We refer to the protocol as

the HSN protocol, where HSN is short for Hard/Soft/Non. The work in this chapter will also appear in [MS98a] and [MS98b].

5.2 System Specification

The sort of wireless network we consider is made up of a small number of nodes, usually no more than a dozen, that are all within transmission range of each other. As before, all nodes use a single frequency and access the channel in a slotted manner. At the end of a data transmission, feedback from each node's physical layer is available, indicating whether the previous slot contained a collision or a noncollision. A noncollision is either an idle channel or a successful transmission during the data slot. In addition to collision feedback, *feed forward* preemption data is also provided. With it, as will be described in detail in the next section, all nodes involved in any ongoing transmission can see if they have been preempted by higher priority data. Hard real-time transmissions always preempt pending soft or non real-time transmissions, and soft always preempts pending non real-time transmissions. Using preemption, guarantees to hard real-time data can always be met, though at the expense of additional delays for soft and non real-time data.

Hard real-time transmissions are accommodated in a straightforward way. Imposed on the system is a minimum interarrival rate for hard real-time packets. On average, at most $\lambda^{(h)}$ packets per slot are permitted. For practical implementation, this translates to allowing at most one packet per cycle, where a cycle is some number of slots to be determined. Since a subset of a known finite population can attempt a hard real-time transmission, calculating the worst case collision resolution time is easy. That worst case value is the minimum possible cycle length.

We can further set a maximum collision resolution length for soft real-time data. If it is known that a deadline will never be more than T slots into the future, the worst case soft collision resolution will be T slots long since the packets will have been transmitted or dropped by then. The sum of the worst case hard and

soft collision resolution intervals (CRIs) is one possible *system cycle length,* though no channel markers exist indicating it as such. That particular cycle length would be fairly conservative by leaving space for all real-time traffic classes. The likely minimum cycle length of interest in most systems would be the largest of worst case hard real-time and worst case soft real-time CRI lengths. The absolute minimum, as mentioned, would be the number of slots to resolve the worst case hard real-time collision with no explicit provision for soft or non real-time data. But on average, the worst case resolution times will not be needed, and left over slots can be used for lower priority real-time transmissions.

Optimal "tuning" of the protocol might additionally impose an average delay on soft real-time transmissions. It has been shown [PGPK89], [MS95] that there is a globally maximum traffic rate for a given range of deadlines that provides a maximum success rate. For non real-time data, we also find that when considering the likeliness of preemption, there is a soft real-time traffic rate minimizing delay for the non real-time stream.

While channel noise disrupting feedback slots will cause stations to lose synchronization, other work [Gon92] has shown that throughput for similar protocols remains nonzero, and so is practical for implementation in non-ideal environments. We will show similar results for our protocol in Chapter 6.

5.3 HSN Protocol Description

As before, we assume that all nodes are operating on the same frequency and that data packets are of a fixed length and transmitted only at slot boundaries. In addition, data slots are always immediately followed by three minislots used by each node to determine what to do for the next data slot, as shown in Figure 5.1. With this simple channel structure, implementing preemption is easy. If either a soft or non real-time CRI is in progress, after the next slot boundary all nodes check the preemption minislots. If the first minislot is set, or a logical 1, then at least one

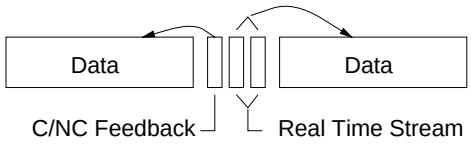


Figure 5.1: Format of distributed feedback.

node will be transmitting a hard real-time packet in the upcoming data slot. All soft and non real-time packets will be preempted until the first preemption minislot is not set, or a logical 0, again, at which time the preempted CRI resumes. Similarly, if the first preemption bit is not set, but the second one is, then all nodes know that the next data slot be a soft real-time transmission. So only when neither of the two preemption minislots are set can a non real-time transmission take place.

To perform the actual splitting in each stream when collisions occur, for hard and soft real-time transmissions, the blocked access Sliding Partition splitting protocol presented in detail in the last chapter is used. Slight modifications are made for hard real-time use. The n hard real-time stations are assumed to be numbered consecutively from 0 to $n-1$ where each node knows its own number. These identifiers are not necessarily a network address, but can be considered resources themselves, an idea put forth by Arvind [Arv91], available for contention at higher protocol levels. Most likely, the LLC layer, in a distributed manner, would be responsible for maintaining a small table of which nodes are assigned which hard real-time identifiers. For a hard real-time system with such a node numbering, the FTTR is simple: if a station has a packet to transmit, it is transmitted at the next slot boundary. If a collision occurs, the CRA is initiated. The CRA uses a one dimensional window encompassing the range $[0, n-1]$. After the initiating collision, the window is split into an active window and an inactive window. At the next slot, nodes whose identifiers are in the active window may again transmit. If another collision occurs, the active window is split, and the new inactive window joins the

previous inactive window. Window splitting continues until feedback indicates a noncollision occurred. Note that the unique node identifier makes "ties" impossible. The worst case CRI for hard real-time nodes would be the amount of time to resolve a collision if all n nodes transmit at the same time. So that guarantees can be offered to hard real-time packets, a constraint is imposed on the system that each hard real-time node may transmit at most one packet during a cycle. A cycle, therefore, is at least as long as the worst case hard CRI.

The Sliding Partition algorithm remains unchanged for use with soft real-time data except that CRIs in progress may be preempted by hard real-time arrivals. For the non real-time data, the Two Cell algorithm presented in [PK92] is used unmodified except that it too is preemptable. Here, the random variable used for splitting is a fair coin toss. On heads, say, a station joins Cell 1, and on tails joins Cell 2. At the next slot, only stations in Cell 1 can transmit. Upon collision, all Cell 1 stations again flip a coin and either remain in that cell or join Cell 2. Cell 2 members never transmit. However, when a data slot is followed by non-collision feedback, all stations rejoin Cell 1.

When a non or soft real-time CRI is in progress, the CRI can potentially be preempted by a hard real-time transmission/CRI. Since only rarely will the worst case collisions occur, many slots will often remain unused by higher priority transmissions. We study the protocol performance resulting from using the largest and smallest likely cycle lengths, that is, the sum of the worst case hard and soft real-time CRIs, or just the maximum of the hard and soft CRI lengths. We do not, however, consider optimization of cycle length here but use the two representative cycles lengths shown in Equation 5.2 in Section 5.5.

Figure 5.2 illustrates how each of the three algorithms operates. Recall that two consecutive non-collisions always mark the end of a CRI since there are never more than two windows, the active and the inactive. Once each is resolved, the CRI

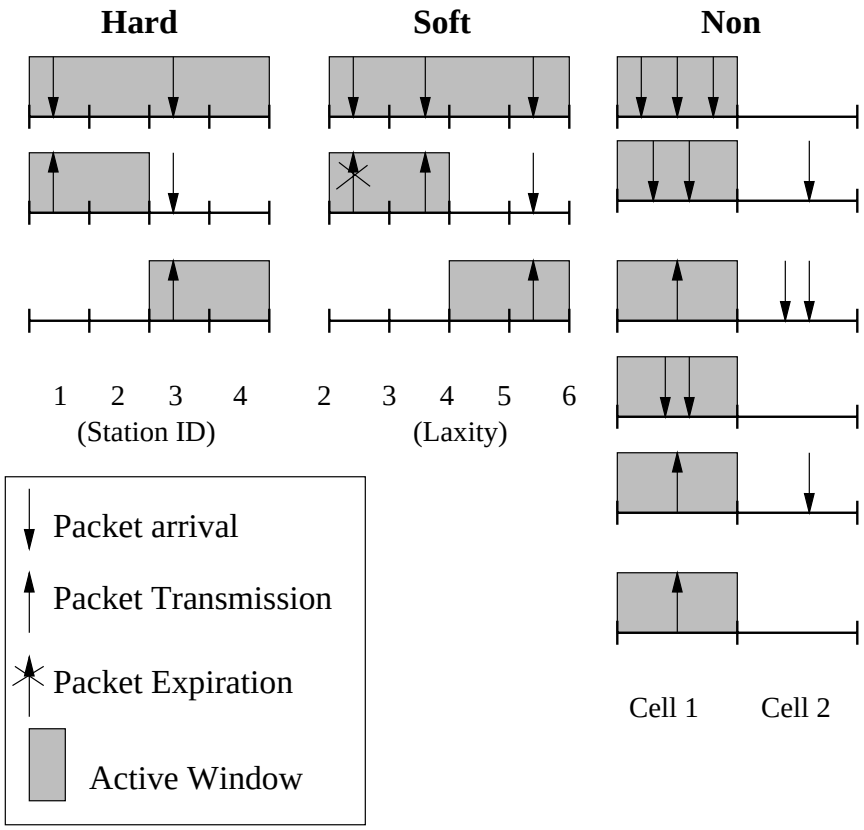


Figure 5.2: Hard, soft and non real-time splitting techniques.

ends and the FTTR is reenabled. The left and middle portions of the figure illustrate how the blocked access Sliding Partition CRA operates in the hard and soft real-time streams. The hard real-time axis is labeled with station identifiers representing the number of nodes in the network. The axis labels for the soft real-time illustration are the full initial laxity range, arbitrarily chosen to be $[2, 6)$ for the example. The non real-time portion of Figure 5.2 shows a Two Cell CRI. Sections 4.3.1 through sec:c4tc provide more detail and additional illustrations regarding these three algorithms.

5.4 Distributed Feedback Algorithm

A unique requirement to the real-time systems considered here is that they are fully decentralized. Traditionally, splitting protocols receive their feedback data from a centralized node such as a base station. Since less predictable environments make this impossible, all feedback, both the initial preemption minislots as well as the trailing collision/noncollision minislot, must be generated by the member stations of the network.

Since an antenna cannot be used to simultaneously transmit and receive, a transmitting station must rely on feedback generated by one or more other stations. Of the stations that did not transmit data, one or more of them will have generated feedback. Indeed, knowing that more than one station agrees on the observation is desirable. The feedback, therefore, cannot itself be a bit pattern that needs decoding because that would be destroyed by a collision. "Collisions" in feedback are actually a good sign that much of the system is synchronized. So rather than holding bit patterns, the minislots are used as *something-or-nothing* slots. That is, the presence of any transmission, not necessarily a decodable bit pattern, is a *something,* while the absence of transmission energy is a *nothing.*

The simplest algorithm would require that all nodes not involved in a transmission generate feedback. For military systems, periodic generation of feedback is not desirable. In a military wireless network, a simple solution would be for each of the nodes not transmitting, to generate feedback with probability $p = 1/(n-1)$, where n is the number of nodes in the network. If not known in advance, n can be estimated based on recent channel utilization.

As part of the physical layer, whether to use the existence or absence of energy to indicate collision is beyond the scope of this work but interesting to consider briefly. On an error free channel, loss of synchronization occurs only when all nodes remain quiet when feedback with a *something* level should have been generated but

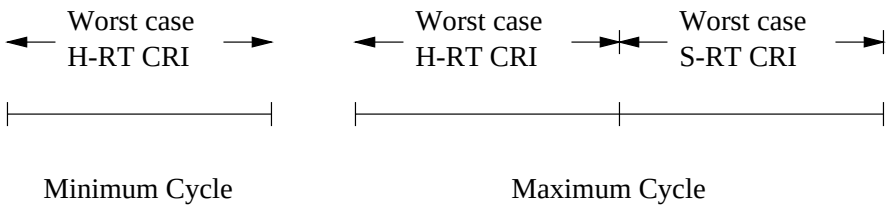


Figure 5.3: Hard real-time cycles.

was not. No feedback is generated only when $p < 1$ and all nodes happen to draw a random number forcing them to remain silent. On non-ideal channels this can also happen because of fading or because a noise burst cancels the transmitted signal. In these cases, making *nothing* equivalent to collision status is more desirable. That way, the nodes split their windows when in fact no splitting was necessary. However, the net remains fully synchronized.

On a non-ideal channel, though, occasional bursts of background noise will generate *something* feedback when it should have been *nothing.* Here, *something* should be equivalent to collision for the same reason stated in the preceding paragraph. Since channel noise is obviously beyond the control of a protocol, a wise choice for p—which is application dependent—offers the best tradeoff. The analysis presented in this chapter assumes feedback is generated with $p = 1$ and that there are zero channel errors. However, later in the chapter we present results for non-idealized channels.

5.5 Hard Real-Time Analysis

Note that the system cycle length, used to determine the minimum interrival rate of hard real-time traffic, is different from one epoch of the regenerative stochastic process characterizing the protocol. A regeneration point is reached when there are no packets requiring service. To determine the system cycle length, the length of a worst case hard real-time collision resolution interval (CRI) must be determined.

To resolve the high multiplicity collision when all hard real-time stations collide, a simple repeating pattern occurs; the window is successively split followed by a collision in each until finally the active window encompasses only a single station's identifier. After making the formerly inactive window active, the pattern repeats until the final station transmits. Summing the collisions based on splitting the entire active window and adding one more slot for the final noncollision feedback, yields

$$L_{\max}^{(h)} = 1 + \sum_{i=2}^{n} (\lceil \log_2 i \rceil + 1) \tag{5.1}$$

The maximum and minimum system cycle lengths, respectively, are then

$$c = L_{\max}^{(h)} + T, \qquad c = \max(L_{\max}^{(h)}, T) \tag{5.2}$$

and are illustrated more conceptually in Figure 5.3. Once can imagine systems with the majority of traffic being non real-time, so that additional slots might be added to the cycle length to guarantee a minimum fraction of non real-time traffic. However, we do not consider such systems since doing so requires only a simple extension of the analysis.

5.5.1 Hard Real-Time Probable CRI Length

A collision that is not worst case offers more possible outcomes and hence requires more elaborate analysis. Given an interval encompassing station identifiers with left edge x and left (active) and right (inactive) windows of lengths w_1 and w_2, respectively, the probability that after splitting the new active window will contain i packets is:

$$P_{split}^{(h)}(k_1, k_2, x, w_1, w_2, i) \tag{5.3}$$
$$= \begin{cases} 0, & \text{if } i < \max(0, k_1 - w_1'), \\ & \quad \text{or } i > \min(k_1, w_1') \\ \binom{w_1'}{i}\binom{w_2'}{k_2+k_1-i} \Big/ \sum_{n=\max(0,k_1-w_1')}^{\min(k_1,w_1')} \binom{w_1'}{n}\binom{w_2'}{k_2+k_1-n}, & \text{otherwise} \end{cases}$$

where $w_1' = \lceil x + \frac{w_1}{2} \rceil - \lceil x \rceil$, and $w_2' = \lceil x + w_1 + w_2 \rceil - \lceil x + \frac{w_1}{2} \rceil$. The limits of summation and conditions on the probability being zero take into account the fact stations have consecutively numbered integers. For instance, a window six slots wide containing four packets must contain, after splitting, at least one and at most three packets in a window half. Equation 5.4 is used by most of the hard real-time analysis to consider the probability of various system outcomes.

5.5.2 Hard Real-Time Expected CRI Length

To calculate the expected length of a hard CRI involving k packets and a window encompassing the interval $[x, x+w)$, all possible splits need to be considered. Similar to Equation 5.2, the notation $L^{(h)}_{k_1,k_2,x,w_1,w_2}$ indicates the expected length of a hard CRI given that the active window $[x, x + w_1)$ contains k_1 packets and the inactive window $[x+w_1, x+w_1+w_2)$ contains k_2 packets. Then the expected length of a hard CRI for a given situation is

$$L^{(h)}_{k_1,k_2,x,w_1,w_2} = \begin{cases} L^{(h)}_{k_2,0,x+w_1,w_2,0}, & \text{if } \lceil x + w_1 \rceil - \lceil x \rceil < 2, \text{ or } k_1 < 2 \\ \sum_{i=0}^{k_1} P^{(h)}_{split}(k_1, k_2, x, w_1, w_2, i) & \\ \quad \cdot(1 + L^{(h)}_{i,k_2+k_1-i,x,\frac{w_1}{2},w_2+\frac{w_1}{2}}), & \text{otherwise} \end{cases} \tag{5.4}$$

with initial conditions: $L^{(h)}_{0,0,\ldots} = 0; L^{(h)}_{1,0,\ldots} = 1; L^{(h)}_{0,1,\ldots} = 2; L^{(h)}_{1,1,\ldots} = 2$.

The summation considers the expected length of the remaining CRI after a split resulted in i packets in the active window. For all possible values of i, the expected length is subsequently multiplied by the probability that a split leaving i packets would actually occur.

Since our goal, though, is to determine the expected length of a hard CRI in general, not just in particular situations as modeled in Equation 5.4, an additional probability is needed to consider how many packets might arrive at a given slot within a cycle. This probability can be obtained by designing a counter that monitors how often a slot might contain i packets. Initially zero, the counter is

incremented by one each time another slot under consideration contains i new hard real-time arrivals. An expression accomplishing this is shown below, where k is the total number of packets that will be transmitted during this cycle, n is the number transmitted in the current slot, s is the number of slots remaining in the cycle, and c is the counter value.

$$m_i(k,n,s,c) = \begin{cases} 0, & \text{if } s = 0 \text{ and } k \neq 0 \\ c, & \text{if } s = 0 \text{ and } k = 0 \text{ and } n \neq i \\ c+1, & \text{if } s = 0 \text{ and } k = 0 \text{ and } n = i \\ \sum_{j=0}^{k} m_i(k-j, j, s-1, c), & \text{if } n = i \\ \sum_{j=0}^{k} m_i(k-j, j, s-1, c+1), & \text{if } n \neq i \end{cases} \quad (5.5)$$

The function m_i would be initially called with $n = -1$ so that with certainty $n \neq i$, and with counter $c = 0$. Ultimately the value of the counter c is returned and is the number of combinations where at least one slot in the cycle contains i packets. As such, c is only incremented when a slot has i packets as shown in the third case of Equation 5.5. To get the fraction of all combinations that have one or more slots with i packets, the above result is divided by the number of possible $k-$transmission combinations during a cycle.

$$t(k,s,c) = \begin{cases} 0, & \text{if } s = 0 \text{ and } k \neq 0 \\ c, & \text{if } s = 0 \text{ and } k = 0 \\ \sum_{j=0}^{k} t(k-j, s-1, c+1), & \text{otherwise} \end{cases} \quad (5.6)$$

In this expression, the counter is incremented by one for every valid combination. Invalid combinations occur when, for instance, k packets will be transmitted in a cycle, but when reaching the end of the cycle, less than k have been transmitted.

With these equations, one can express the expected length of a hard CRI where each of the n hard real-time stations will transmit with probability p one packet during a cycle. The first summation below considers the cases of zero through all stations transmitting sometime in the cycle, the second summation considers the

number of transmissions at each slot in the CRI, and the final factor generates the expected CRI length for particular situations:

$$l^{(h)} = E\{\text{CRI len}\} = \underbrace{\sum_{k=0}^{n} \binom{n}{k} p^k (1-p)^{n-k}}_{P\{k \text{ tx'd in cycle}\}} \cdot \underbrace{\sum_{j=0}^{k} \frac{m_j(k,-1,c,0)}{t(k,c,0)}}_{P\{j \text{ tx'd this slot}\}} \cdot \underbrace{L^{(h)}_{j,0,1,n,0,}}_{E\{\text{ CRI len}\}} \quad (5.7)$$

where c is one of the system cycle lengths of Equation 5.2.

Recalling that hard real-time transmissions always preempt both soft and non real-time transmissions, the analysis of those systems must consider on a slot by slot basis the probable lengths of preempting hard real-time CRIs. That is, an expression must be derived to calculate what is the probability that a hard CRI is of a length l. Such an expression can be easily specified in a recursive manner similar to Equation 5.7:

$$P^{(h)}(l \mid n,p) = \sum_{k=0}^{n} \binom{n}{k} p^k (1-p)^{n-k} \sum_{j=0}^{k} \frac{m_j(k,-1,c,0)}{t(k,c,0)} P^{(h)}(l \mid j,0,1,n,0). \quad (5.8)$$

where the probability of a CRI in particular cases being of length l is

$$P^{(h)}_{len}(l \mid k_1,k_2,x,w_1,w_2) = \begin{cases} P^{(h)}_{len}(l-1 \mid k_2,0,x+w_1,w_2,0). & k_1 \leq 1 \\ \sum_{i=0}^{k} P^{(h)}_{split}(k_1,k_2,x,w_1,w_2,i) & \\ \quad \cdot P^{(h)}_{len}(l-1 \mid i,k_2+k_1-i,x,\frac{w_1}{2},w_2+\frac{w_1}{2}), & k_1 > 1 \end{cases} \quad (5.9)$$

with initial conditions: $P(l = 0 \mid k_1 = 0, k_2 = 0, \ldots) = 1; P(l = 0 \mid \ldots)$ and $k_1 + k_2 \neq 0$ is 0; $P(l \neq 0 \mid \ldots)$ and $k_1 + k_2 = 0$ is 0; $P(l > c \mid \ldots) = 0; P(l = 1 \mid k_1 = 1, k_2 = 0, \ldots) = 1; P(l = 1 \mid \ldots)$ and $k_1 + k_2 > 1$ is 0; $P(l = 2 \mid k_1 = 1, k_2 = 1, \ldots) = 1; P(l \neq 2 \mid k_1 = 1, k_2 = 1, \ldots) = 0$.

5.5.3 Hard Real-Time Expected Packet Delay

Like the previous analyses, per packet delay is determined by dividing the number of packets transmitted by the cumulative delay. In the case of the hard real-time subsystem, clearly the number of packets transmitted during a system cycle of

length c slots is

$$N^{(h)} = np \tag{5.10}$$

where n is the number of hard real-time nodes, and p is the probability that a given node will transmit a packet during a system cycle.

An expression for cumulative delay is more involved. Like determining CRI length, the cumulative delay expression must consider the delay regarding every possible per cycle transmission combination, multiplying a given value by the probability of the occurrence of such a pattern.

$$Z^{(h)}_{k_1,k_2,x,w_1,w_2} = \begin{cases} 0, & \text{if } k_1 = k_2 = 0 \\ 1, & \text{if } k_1 = 1, k_2 = 0 \\ 2, & \text{if } k_1 = 0, k_2 = 1 \\ 3, & \text{if } k_1 = 1, k_2 = 1 \\ Z^{(h)}_{k_2,0,x+w_1,w_2,0}, & \text{if } \lceil x + w_1 \rceil - \lceil x \rceil < 2, \\ & \quad \text{or } k_1 < 2 \\ \sum_{i=0}^{k_1} P^{(h)}_{split}(k_1, k_2, x, w_1, w_2, i) & \text{otherwise} \\ \quad \cdot (k_1 + k_2 + Z^{(h)}_{i,k_2+k_1-i,x,\frac{w_1}{2},w_2+\frac{w_1}{2}}) & \end{cases} \tag{5.11}$$

Given expected delay from Equation 5.11 for particular situations, it is next necessary to determine the likelihood of those particular situations. Paralleling the derivation of the CRI length analysis, the expected delay considering each of the finite possible initial collisions is

$$a^{(h)} = \underbrace{\sum_{k=0}^{n} \binom{n}{k} p^k (1-p)^{n-k}}_{P\{k \text{ tx'd in cycle}\}} \cdot \underbrace{\sum_{j=0}^{k} \frac{m_j(k,-1,c,0)}{t(k,c,0)}}_{P\{j \text{ tx'd this slot}\}} \cdot \underbrace{W^{(h)}_{j,0,1,n,0},}_{E\{\text{ packet delay}\}} \tag{5.12}$$

where c is again a system cycle length from Equation 5.2.

5.6 Soft Real-Time Analysis

The soft real-time algorithm is the most involved of the three streams because of probabilistic guarantees are needed. For both hard and non real-time data, all submitted packets are always, assuming good channel state, transmitted. The difference, of course, is that hard traffic further guaranteed a worst case delay, while non real-time traffic is not. Soft real-time traffic, on the other hand, is guaranteed only that on average a given fraction of the offered load will be transmitted before expiring. Because not all packets can always be transmitted successfully in a CRI, there is a nonzero probability that zero or more packets will expire in each slot. To make a probabilistic guarantee on success rate, the protocol necessarily needs to be aware of the possible range of initial laxities associated with the traffic arrival process. As mentioned in previous chapters, the minimum laxity possible is two slots, while the maximum, also in units of slots, is a system parameter denoted by T. We use the blocked access Sliding Partition algorithm presented and analyzed in [MS98c] but modify the algorithm to accommodate preemption by the hard real-time stream.

5.6.1 Soft Real-Time Probable CRI Length

Each recursion of the analytic equations takes place at slot boundaries. As such, at each boundary it is possible that preemption by the hard real-time traffic stream will occur. The first summation in Equation 5.13 considers just this. Given that the preemption is s slots long, where $s \geq 0$, the next recursion, rather than decrementing just one from the slots remaining in the CRI, one plus the number of preemption slots is decremented.

$$P_{len}^{(s)}(l \mid k_1, k_2, B, x, w_1, w_2) = \begin{cases} \overbrace{\sum_{d=0}^{k_2} \sum_{s=0}^{L_{\max}^{(h)}} P_{len}^{(h)}(s \mid n, p)}^{\text{Hard CRI}} \\ \quad \cdot P_{len}^{(s)}(l-1 \mid k_2 - d, 0, B-1-s, \\ \qquad x + w_1, w_2, 0) \\ \quad \cdot P_{drop}^{(s)}(k_2, d \mid T, B-s, x+w_1, w_2), \quad k_1 = 0, 1 \\ \\ \sum_{i=0}^{k_1} \text{bin}(w_1/(2w_1+2w_2), k_1, i) \\ \quad \cdot \sum_{d=0}^{i} \sum_{s=0}^{L_{\max}^{(h)}} P_{len}^{(h)}(s \mid n, p) \\ \quad \cdot P_{len}^{(s)}(l-1 \mid i-d, k_2+k_1-i, \\ \qquad B-1-s, x, w_1/2, w_2+w_1/2) \\ \quad \cdot P_{drop}^{(s)}(i, d \mid T, B-s, x, w_1/2), \qquad \text{otherwise} \end{cases} \tag{5.13}$$

Preemption complicates the initial conditions as well. For cases where all packets have not been either transmitted or dropped, and yet the CRI is nearly ended, additional conditions now exist for a nonzero probability that the CRI will be of length l. For instance, in a multiplicity three collision of packets all having laxities of 10 slots, the CRI length might still be only one slot. After the initial collision, the soft CRI could be preempted by a hard CRI that takes 9 or more slots. Therefore, the initial conditions taking into account preemption become

$$P_{len}^{(s)}(l=1 \mid \ldots) = \begin{cases} 1, & \text{if } B = 1 \\ 1, & \text{if } k_1 = 1, k_2 = 0 \\ \sum_{s=0}^{L_{\max}^{(h)}} P^{(h)}(s \mid n,p) & \text{if } k_1 \leq 1, k_2 > 0 \\ \quad \cdot P_{drop}^{(s)}(k_2, k_2 \mid T, B-s, x+w_1, w_2), & B - s > 1 \\ \sum_{s=0}^{L_{\max}^{(h)}} P^{(h)}(s \mid n,p) & \text{if } k_1 > 1, k_2 = 0 \\ \quad \cdot P_{drop}^{(s)}(k_1, k_1 \mid T, B-s, x+w_1, w_2), & B - s > 1 \\ \sum_{s=0}^{L_{\max}^{(h)}} P^{(h)}(s \mid n,p) & \\ \quad \cdot P_{drop}^{(s)}(k_1, k_1 \mid T, B-s, x, w_1) & \text{if } k_1 > 1, k_2 > 0 \\ \quad \cdot P_{drop}^{(s)}(k_2, k_2 \mid T, B-s, x+w_1, w_2), & B - s > 1 \end{cases} \tag{5.14}$$

with further initial conditions $P_{len}^{(s)}(l \mid \ldots) = 0$ if $k_1 = k_2 = 0$, or if $l > B$.

5.6.2 Soft Real-Time Expected CRI Length

Similarly, expressions are modified for expected CRI length, expected number of packets transmitted during a CRI, and expected delay of those packets. The expression is shown below for the Sliding Partition expected length of a CRI. The only differences involve preemption by newly arriving hard real-time packets. Prior to each recursion, each possible number s of hard arrivals is considered. The variable B counting down slots remaining before the soft packets are discarded is decremented by the length of any hard real-time preemption.

$$L^{(s)}_{k_1,k_2,B,x,w_1,w_2} = \begin{cases} 1 + \sum_{d=0}^{k_2} \sum_{s=0}^{L^{(h)}_{\max}} P^{(h)}_{len}(s \mid n,p) & \\ \quad \cdot P^{(s)}_{drop}(k_2, d \mid T, B-s, x+w_1, w_2) & \\ \quad \cdot (s + L^{(s)}_{k_2-d,0,B-1-s,x+w_1,w_2,0}), & k_1 \leq 1 \\ & \\ \sum_{i=0}^{k_1} \mathrm{bin}(w_1/(2w_1+2w_2), k_1, i) & \\ \quad \cdot \sum_{d=0}^{i} \sum_{s=0}^{L^{(h)}_{\max}} P^{(h)}_{len}(s \mid n,p) & \\ \quad \cdot P^{(s)}_{drop}(i, d \mid T, B-s, x, w_1/2) & \\ \quad \cdot (s + L^{(s)}_{i-d,k_2+k_1-i,B-1-s,x,w_1/2,w_2+w_1/2}), & \text{otherwise} \end{cases} \tag{5.15}$$

with initial conditions $L^{(s)} = 1$ if $B = 1$, $L^{(s)} = 1$ if $k_1 \leq 1, k_2 = 0$.

5.6.3 Soft Real-Time Expected Number of Transmitted Packets

In the same manner, Equation 4.8 is modified to account for hard real-time preemption:

$$N^{(s)}_{k_1,k_2,B,x,w_1,w_2} = \begin{cases} k_1 + \sum_{d=0}^{k_2} \sum_{s=0}^{L^{(h)}_{\max}} P^{(h)}_{len}(s \mid n,p) & \\ \quad \cdot P^{(s)}_{drop}(k_2, d \mid T, B-s, x+w_1, w_2) & \\ \quad \cdot N^{(s)}_{k_2-d,0,B-1-s,x+w_1,w_2,0}, & k_1 \leq 1 \\ & \\ \sum_{i=0}^{k_1} \mathrm{bin}(w_1/(2w_1+2w_2), k_1, i) & \\ \quad \cdot \sum_{d=0}^{i} \sum_{s=0}^{L^{(h)}_{\max}} P^{(h)}_{len}(s \mid n,p) & \\ \quad \cdot P^{(s)}_{drop}(i, d \mid T, B-s, x, w_1/2) & \\ \quad \cdot N^{(s)}_{i-d,k_2+k_1-i,B-1-s,x,w_1/2,w_2+w_1/2}, & \text{otherwise} \end{cases} \tag{5.16}$$

with initial conditions $N^{(s)} = 1$ if $k_1 = 1, k_2 = 0$, and $N^{(s)} = 0$ if $k_1 = k_2 = 0$.

5.6.4 Soft Real-Time Expected Cumulative Delay

And finally, Equation 4.10 is modified as well:

$$Z^{(s)}_{k_1,k_2,B,x,w_1,w_2} = \begin{cases} k_1 + \sum_{d=0}^{k_2} \sum_{s=0}^{L^{(h)}_{\max}} P^{(h)}_{len}(s \mid n,p) \\ \quad \cdot P^{(s)}_{drop}(k_2, d \mid T, B-s, x+w_1, w_2) \\ \quad \cdot Z^{(s)}_{k_2-d,0,B-1-s,x+w_1,w_2,0}, & k_1 \le 1 \\ \\ \sum_{i=0}^{k_1} \mathrm{bin}(w_1/(2w_1+2w_2), k_1, i) \\ \quad \cdot \sum_{d=0}^{i} \sum_{s=0}^{L^{(h)}_{\max}} P^{(h)}_{len}(s \mid n,p) \\ \quad \cdot P^{(s)}_{drop}(i, d \mid T, B-s, x, w_1/2) \\ \quad \cdot (N^{(s)}_{i-d,k_2+k_1-i,B-1-s,x,w_1/2,w_2+w_1/2} \\ \quad s + Z^{(s)}_{i-d,k_2+k_1-i,B-1-s,x,w_1/2,w_2+w_1/2}), & \text{otherwise} \end{cases} \tag{5.17}$$

with initial conditions $Z^{(s)} = 0$ if $k_1 = k_2 = 0$, or if $B = 1$; $Z^{(s)} = 1$ if $k_1 = 1, k_2 = 0$, or $k_1 = 1, B = 1$.

5.7 Non Real-Time Analysis

The Two Cell algorithm presented by Paterakis *et al.* [PPK89] is used unmodified for non real-time transmission. However, analysis of the algorithm must allow preemption by both hard and soft real-time streams. The Two Cell CRA is a window splitting protocol that splits based on a coin toss, and maintains the channel format of data slots followed by binary feedback and preemption minislots. A preemption of non real-time servicing can occur if there is one or more hard real-time arrivals or if there are zero hard and one or more soft real-time arrivals. The probability of these events occurring is:

$$p^{(n)} = \overbrace{\sum_{l=1}^{L^{(h)}_{\max}} l \cdot P^{(h)}_{len}(l \mid n,p)}^{\text{hard CRI}} + \overbrace{P^{(h)}_{len}(0 \mid n,p) \sum_{l=0}^{T} l \cdot P^{(s)}(l \mid \ldots)}^{\text{soft CRI}} \tag{5.18}$$

Accordingly, the expected length of a non real-time CRI is modeled as

$$L^{(n)}_{k_1,k_2} = \begin{cases} p^{(n)} + 1 + L^{(n)}_{k_2,0}, & k_1 \le 1 \\ p^{(n)} + 1 + 2^{-k_1} \sum_{i=0}^{k_1} \binom{k_1}{i} L^{(n)}_{i,k_2+k_1-i}, & \text{otherwise} \end{cases} \tag{5.19}$$

However, this generates a system of linear equations infinite in dimensionality. So we can instead use a simpler approach. From the analysis of the Two Cell algorithm, we know that its maximum throughput is 0.43 packets per slot. Therefore, the algorithm's throughput in the preemptive real-time setting is

$$s^{(n)} = 0.43\left[1 - l^{(h)} - l^{(s)}\right] \tag{5.20}$$

where $l^{(h)}$ is the expected CRI length of a hard real-time CRI, and $l^{(s)}$ is the expected CRI length of a soft real-time CRI but ignores preempting hard real-time slots. The two individual expected lengths are needed, rather than simply using Equation 5.15 as is, to account for times when there is hard real-time traffic but no soft real-time traffic and vice-versa. Ignoring the hard preemption slots is easily accomplished by modifying Equation 5.15 so that the "$s + \ldots$" in the expressions are replaced by "$1 + \ldots$" where the terms represented by the ellipsis are unchanged. The idea is to count just the number of slots preempted. Since soft CRIs can be preempted by hard CRIs, the counter in the soft preemption expressions must not count for a second time the slots already counted by s in the hard real-time expressions. Packet delay is similarly calculated by augmenting the isolated, or stand-alone, Two Cell delay results from [PPK89] with preempting delays.

5.8 Results

Because of the preemptive nature of the HSN protocol described in this chapter, not only are the general performance results interesting, but also interesting are results showing what effect preemption has on the various real-time traffic streams. The hard real-time stream, since it always has highest priority, is of course unaffected by any preemption. However, depending on how system cycle length is chosen, the maximum arrival rate of hard traffic is, in fact, affected. Since one packet per node per cycle may be transmitted, the shorter the cycle the higher the maximum transmission rate.

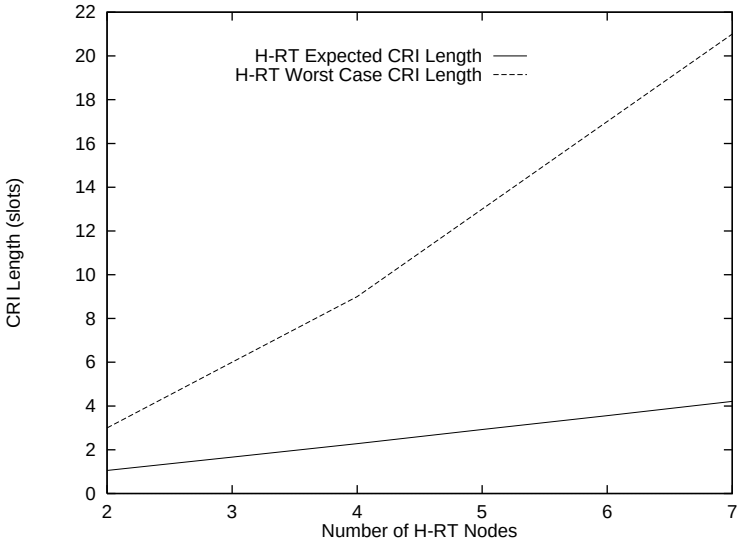


Figure 5.4: H-RT worst case and expected length CRIs.

Results were obtained for both the minimum likely and the maximum likely system cycle lengths shown in Equation 5.2. For various numbers of hard real-time nodes, Figure 5.4 graphs both the worst case and the expected CRI lengths when a maximum cycle length is used. It is immediately clear that the worst case CRI length grows much more quickly than the expected length. As a result, the minimum likely cycle length, $\max(L_{\max}^{(h)}, T)$, is often the better choice since it allows more frequent transmissions by hard real-time nodes, at the expense of less throughput in the soft and non real-time portions of the system. If still higher frequency of hard real-time transmissions is needed, the cycle length could be made as short as the worst case hard CRI length, $L^{(h)}$. A cycle length this short would be desirable when the majority of the system traffic is hard real-time. If the majority is non real-time, one can envision the need for an even longer cycle length than our "maximum likely" length used in this analysis. The cycle length could be further increased to accommodate more non real-time traffic, for instance.

In Figure 5.5, the effect of varying transmission rates of the hard real-time

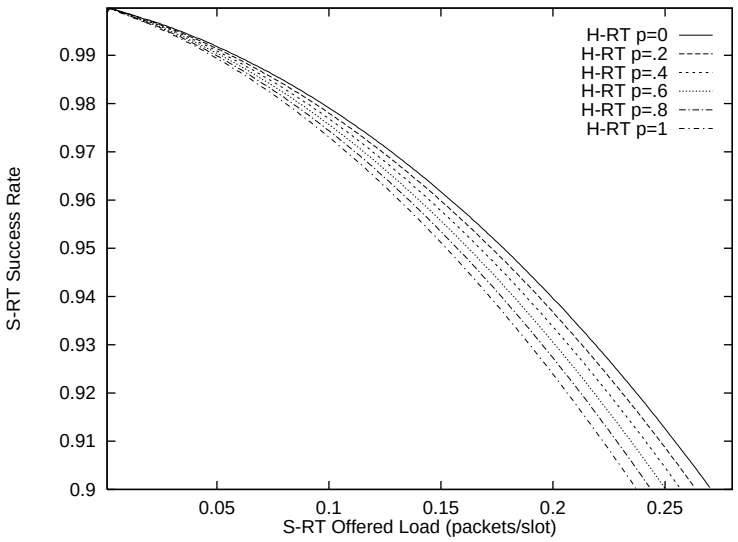


Figure 5.5: Effect of H-RT traffic rate on S-RT using max cycle, T=5.

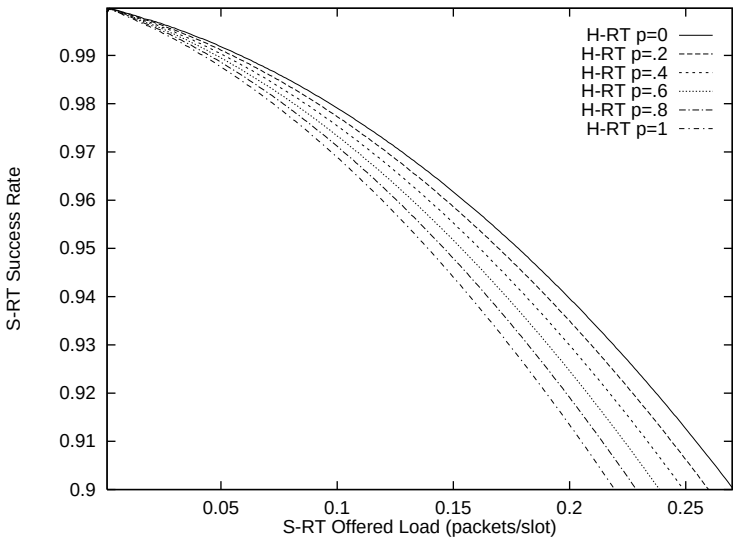


Figure 5.6: Effect of H-RT traffic rate on S-RT using min cycle, T=5.

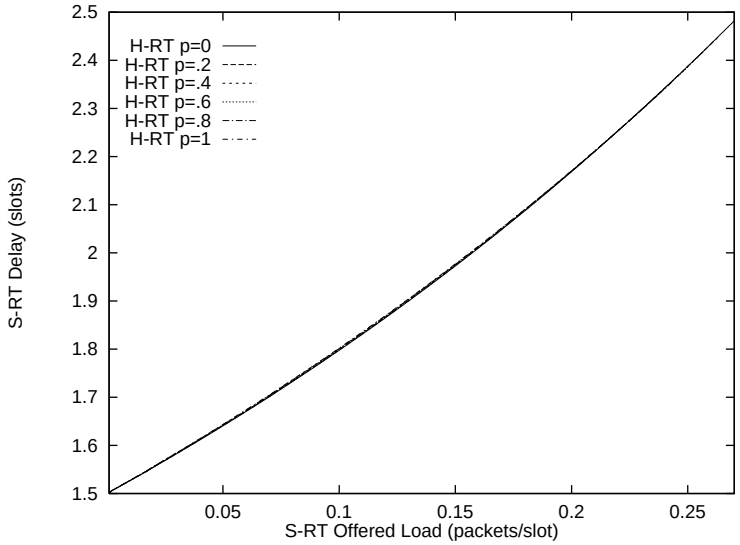


Figure 5.7: Effect of H-RT traffic rate on S-RT delay, max cycle, T=5.

nodes on the soft real-time traffic is shown. As the probability p, of a single node transmitting a hard real-time packet in a cycle length, here 14 slots with 4 hard real-time nodes, is increased from zero to one, the success rate for the soft real-time traffic steadily drops. In this case, the maximum likely cycle length is used, and so the drop in success rate is about 2% at worst. In Figure 5.6, where the minimum likely cycle length is used, the performance is affected more significantly with an almost 4% drop in success rate due to more frequent preemption while maintaining the same 90% success rate. For the maximum cycle length, Figure 5.7 graphs the delay curves for the same success rate curves of Figure 5.5. Interestingly, delays for all values of p are nearly identical because we are graphing performance for success rates of 90% or above. Since a soft CRI can last at most T slots, where $T = 5$ slots for these results, preempting the CRI does not "stop time" for the soft packets. Preemption only prevents soft packets from being transmitted as they age resulting in less successful transmission but the same delay for those that do not expire. As a result, the curves show that the maximum average delay is reached

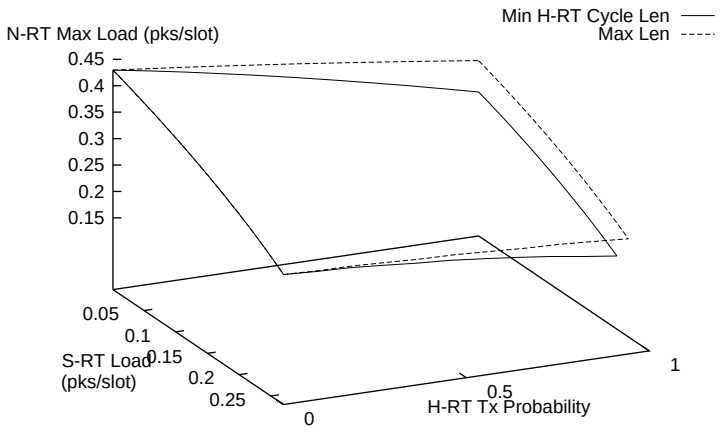


Figure 5.8: N-RT maximum loads for min/max cycle lengths, 4 H-RT nodes, T=30.

in all cases, but due to preemption, the offered load is decreased correspondingly. That is, since we are analytically finding the maximum λ that meets the desired success rate, this is equivalent to finding the success rate (as preemption frequency increases) whose delay is as high as tolerable. So while the success rates differ, the delay characteristics are the same.

Studying non real-time traffic is still more involved since non real-time traffic can be preempted by both hard and soft traffic. We know from [PPK89] that the maximum throughput of the Two Cell algorithm is 0.43 of the available bandwidth. In general, the sum of hard and soft traffic is nonzero and so the total throughput of the non real-time system is not only reduced, also the per packet delay is proportionally increased.

Figure 5.8 compares the non real-time maximum throughput when the maximum and minimum cycle lengths are used. The uppermost surface is the set of results when the maximum cycle length is used. The x axis is the probability that

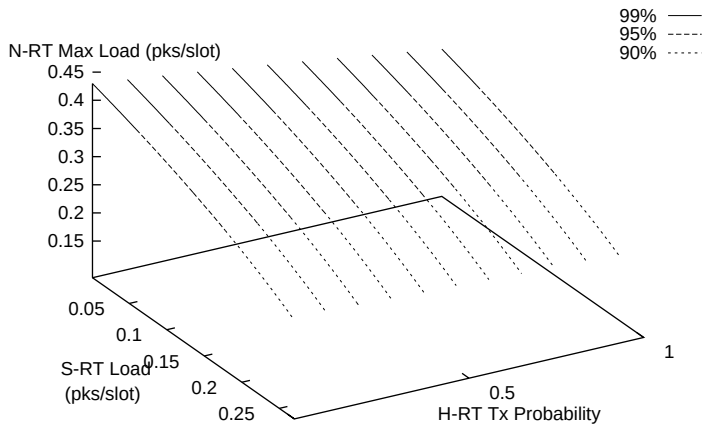


Figure 5.9: Maximum sustainable N-RT load, 6 H-RT nodes, T=30.

each of hard real-time nodes will transmit during the current cycle, the y axis is the range of corresponding soft real-time traffic rates that will be offered a 90% or better success rate, and the z axis is the non real-time throughput. There are six hard real-time nodes.

As one would expect, Figure 5.8 shows that with zero hard or soft real-time traffic, the non real-time system has its theoretical maximum throughput of 0.43 packets/slot. But when either or both of the hard and soft systems approach their maximum transmissions rates, the non real-time system has its lowest throughput. When the cycle length is shorter, however, we see that throughput for the non real-time system suffers more than in the corresponding system with a long cycle length, though the difference is only approximately 5%.

Next, we look more closely at the results when the maximum cycle length is used. In Figure 5.9, the surface is additionally divided into three regions labeled in the legend, where the soft real-time success rate is 90%, 95% and 99% or better for the offered load. For given hard and soft traffic rates, Figure 5.10 shows the

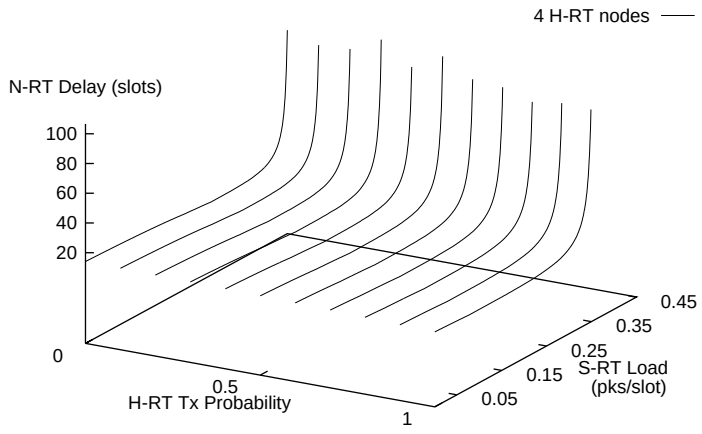


Figure 5.10: N-RT delay for max sustainable loads, 6 H-RT nodes, T=30.

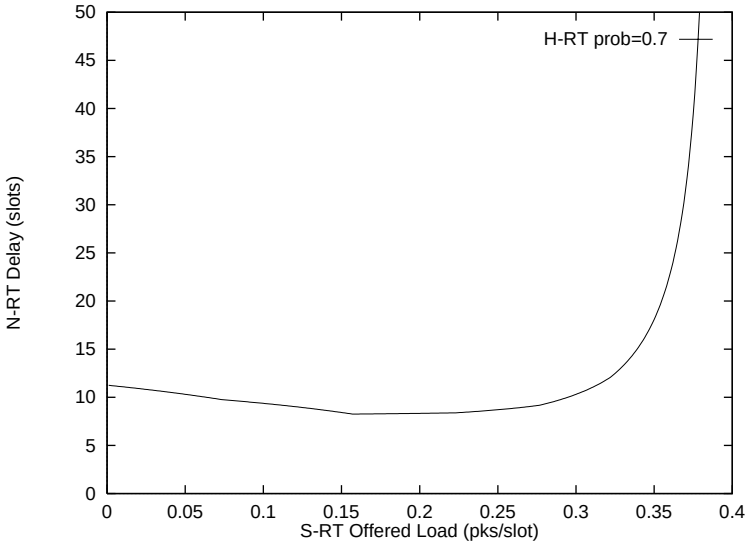


Figure 5.11: S-RT traffic rate vs. N-RT delay, 6 H-RT nodes, T=30.

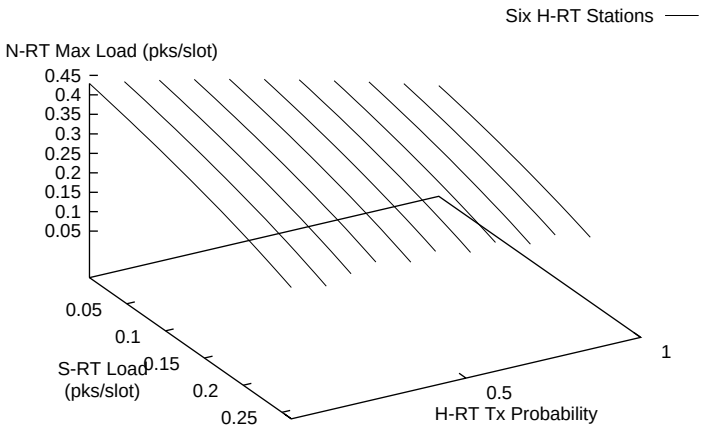


Figure 5.12: Maximum sustainable N-RT load, 6 H-RT nodes, T=30.

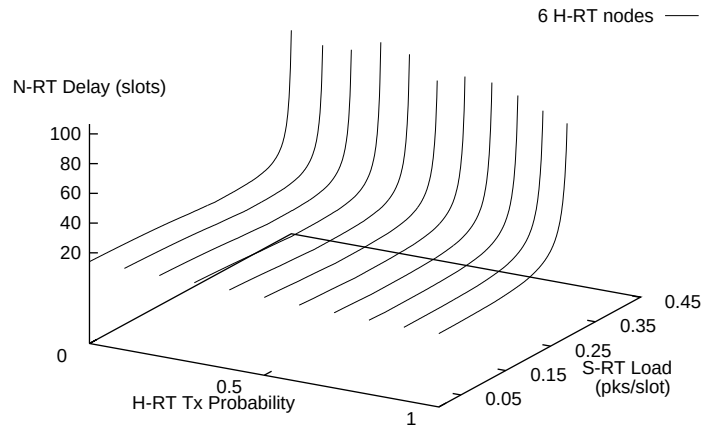


Figure 5.13: N-RT delay for max sustainable loads, 6 H-RT nodes, T=30.

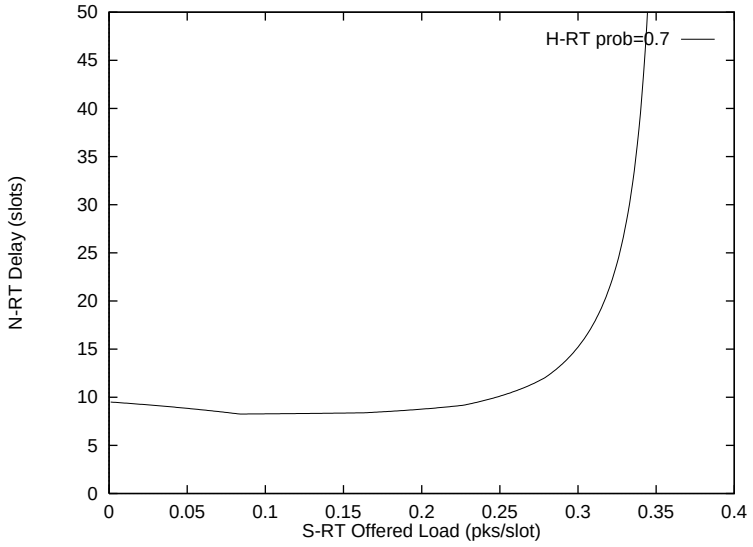


Figure 5.14: S-RT traffic rate vs. N-RT delay, 6 H-RT nodes, T=30.

delay experienced by the non real-time traffic stream. Figure 5.11, redrawn with the independent axis in units of soft real-time packets/slot, shows a typical "slice" of the surface. What is noteworthy is that the minima create a slight valley through the surface. Figures 5.12, 5.13 and 5.14 show performance results under the same conditions but for six hard real-time stations with a minimum system cycle length. Similar trends are seen, though of course the delay is higher because of increased preemption.

The initial decrease in non real-time delay as non real-time traffic rate increases can be explained by preemption. The preemption permits less non real-time traffic to enter, which in turn offers less non real-time delay since there is less contention in that stream. However, as preemption increases due to increased soft real-time traffic, the non real-time delay begins to rise again. Optimal use of the protocol would, when possible, tune parameters to operate in the lowest delay "valley."

It should be emphasized that the analytic results obtained in this section

show the system performance at the extremes of operation. For instance, using the Bernoulli probability p for the hard real-time system implies a traffic rate of

$$\lambda^{(h)} = pn/c \iff p = c\lambda^{(h)}/n \tag{5.21}$$

where n are the number of nodes and c the cycle length. For a given traffic rate $\lambda^{(h)}$, it is possible to generate a non-ideal p, or one not as large as it could be. Hence, the system will have more bandwidth available for soft and non real-time traffic. Similarly, non real-time traffic would usually not be transmitted at the maximum sustainable rate, but at a lesser rate with a correspondingly smaller delay.

Which cycle length to use is of course determined more by the needs of the system than by protocol performance. The results obtained show clear tradeoffs of using a short versus a long one. Whether the differences in performance between the two are viewed as good or bad depend solely on the real-time system being implemented on the network.

5.9 Summary

Relatively little research has been conducted in the area of real-time wireless networks, and what does exist mostly deals with one or two types of coexisting real-time traffic streams. We have presented the HSN protocol, the first fully distributed MAC protocol that, assuming good channel state, can offer guarantees to hard real-time traffic, probabilistic guarantees to soft real-time traffic, and provide best effort service to non real-time traffic in *ad hoc* wireless networks. All three subsystems use similar splitting protocols that rely on binary and preemption feedback. Because feedback has traditionally been assumed to be provided by a central source, something unavailable in the environment we consider, we have also presented a simple algorithm providing feedback in a fully distributed manner. The real-time MAC and the distributed feedback algorithms are, however, fully independent of each other.

Both the MAC protocol and the distributed feedback protocol appear to be the first of their kind.

The assumption of good channel state in a wireless system clearly does not hold up for long in practical implementations. Any system built upon this or any other wireless protocol will necessarily require some sort of fault tolerance. Likewise, the hard real-time data flowing in these systems will not be of the same critical nature as data in wire-based hard real-time systems in, say, power plants or weapon systems. But in situations where wireless nets are the only option, the best possible service should still be made available to applications. Furthermore, the three subprotocols should all make identical use of the feedback and preemption minislots. Since one can assume with certainty that synchronization will occasionally be lost in actual implementations, the protocol must be able to resynchronize without an explicit awareness of doing so. That is, even in a worst case preemption error where several stations think they are resolving hard, several think they resolving soft, and still others think they are resolving non real-time collisions, the protocol should allow each station to return to a renewal point. In all cases, two consecutive noncollisions allow this. Similarly, correct preemption minislots further speed resolutions and returns to renewal points.

While probably not ideal for such applications as digitized voice transmission, where long term resource allocation is needed, the HSN protocol is singularly suited to networks where each node is a part of a larger, cooperating real-time system and transmits relatively short messages. The protocol's performance is readily predicted for reasonable ranges of operation, and there is no added channel complexity beyond the preemption feedback. That is, the cycle length is an implicit property of the algorithm; there are no channel markers indicating when various real-time streams may contend for the channel. Not requiring physical channel markers not only reduces complexity of implementation, but reduces average packet delay per stream

as well. Similarly, because all three streams, hard, soft, and non, may contend in a fully random access manner, the gains in channel utilization due to statistical multiplexing are available that are not present in TDMA or polling type systems.

The HSN protocol offers easy implementation, predictability, and most importantly, provides the foundation of the protocol stack, the MAC layer, with the ability to offer general real-time services to higher layers.

Chapter 6

A REAL-TIME DATA LINK LAYER ON A NON-IDEAL CHANNEL

6.1 Introduction

In this chapter, we focus on design issues and protocol performance when moving from the purely theoretical realms of previous chapters into environments more typical of those found in actual implementations. Studies involving ideal models were needed because there has been little work in the literature regarding such protocols. A basic understanding was needed answering questions of what worked and what did not. Recall, for instance, that the blocked access Fully Recursive MAC layer protocol performed worse than a non real-time protocol. Additionally, a theoretical understanding was necessary so that performance could be predicted. Without that ability, designing a system to make the fullest use of available resources is impossible. After building a simple mathematical model and then extending it several times, we were ultimately able to design the HSN MAC protocol supporting hard, soft, and non real-time traffic. That protocol is used in the following studies as a base for a simple LLC layer. Together, they yield a full data link layer. Services provided are simple because the goal is to learn about core functionality. When that is incorporated in the data link layer, more elaborate services are easily added.

6.2 Model

Augmenting the ideal theoretical predictions with simulations of realistic, non-ideal environments is vital to understand how the algorithms perform when

things go wrong. Simulations described in the previous chapters modeled the arrival process in an ideal manner, where typically an infinite number of users on an ideal channel were simulated. Additionally, traffic was made up of only new arrivals; there were no retransmissions of failed packets. In real wireless systems, though, errors are quite common. Errors decrease throughput both by causing retransmission of destroyed data and by causing loss of synchronization between stations when feedback information is lost. This chapter addresses two topics: studying MAC layer performance on non-ideal channels, and adding a basic LLC layer to study how a complete data link layer performs.

6.3 Modeling of the Physical and MAC Layers

The physical wireless link modeled is typical of many systems in most respects. An 800 MHz carrier is used with a bandwidth of 30 kHz. For most modulation methods, this yields about 10 kb/s. To study the effects of channel degradation, we additionally assign a bit error rate (BER) probability. Doing this effectively means that the modulation model is impervious to noise. Following this ideal stage, the data, bit by bit, is run through individual BER Bernoulli trials.

The channel structure is composed of slots that are 10 kb long, including the three minislots. A forward error correction code (ECC) is used that can correct up to any 10 bit errors. Such an ECC is reasonable considering that wireless channels often experience bit error rates in the area of 10^{-4}. For a noisier than average channel with a BER of about 10^{-3}, it is easy to calculate that that is about the limit of what this ECC can correct. Therefore, the data is generated on channels with three different BERs: 0, 10^{-4}, and 10^{-3}.

All radios have 1 W transmitters and are randomly placed, though always positioned within range of every other radio. While the capture effect takes place, it is ignored. That is, the distributed feedback protocol indicates whether or not a collision was sensed in the data slot. Even if the destination node correctly received

the packet due to the capture effect, but another node observed a collision, then the packet is discarded. While not making use of a correctly received packet leaves room for improvements to the protocol, basing MAC algorithmic decisions only on the feedback offers an advantage. Using just the feedback more fully uncouples the splitting algorithm from physical properties of the channel and makes it easier for the stations to remain synchronized. A better alternative would be for the station to pass the captured packet up the protocol stack but still take part in the splitting to maintain synchronization. The LLC would then ignore duplicates.

The MAC layer then only needs the ability to receive the physical layer's channel sensing results to implement the splitting. At the physical layer, a signal level below a threshold is considered a non-collision, and a level above the threshold is considered a collision. Within the MAC layer, the resource requirements are minimal though a bit more than in traditional MACs. Usually, the MAC layer has one queue of length one. However, to implement this protocol, three queues of length one are needed, one each for hard, soft, and non real-time packets, so that preempted packets are not lost. MAC/LLC signaling is also straightforward. The MAC layer is informed by the LLC that it has a packet of a given class to transmit. The MAC responds by either accepting, rejecting, or, in the case of hard real-time, possibly offering to accept the packet but without a guarantee. The latter response is possible when the MAC is free yet the packet's H-RT deadline cannot be met should a worst case collision occur, *i.e.,* the deadline is less than one cycle length away.

Figure 6.1 shows how the hard real-time performance degrades as channel noise increases. In the worst scenario, the graph shows that the hard real-time data reaches only a 96% success rate rather than the ideal of 100% indicating that a fault tolerance subsystem will be a necessary and important part of any wireless real-time system. The graph is also somewhat conservative in that a minimum hard

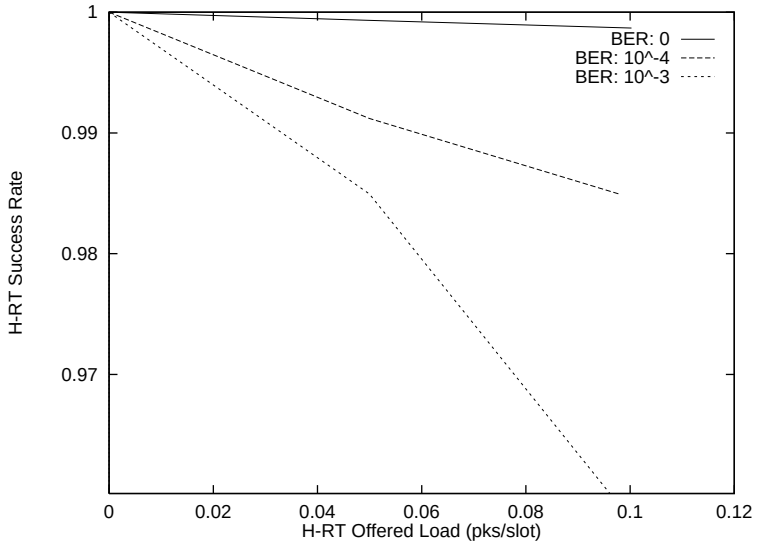


Figure 6.1: MAC H-RT performance as channel degrades.

real-time cycle length, as described in Chapter 5, is used. If it is known that the target environment will be noisy, a longer cycle length can be used to accommodate the expected increase in retransmissions.

Figure 6.2 shows, with no hard real-time traffic, how increasing bit error rates degrades soft real-time success rates. Figures 6.3 through 6.5 show similar results, except that for a given bit error rate in each graph, results show the added effect of increasing hard real-time traffic loads. As expected, soft real-time traffic experiences lower success rates as it is preempted by an increasing amount of hard real-time traffic. The graphs show the effect of increasing hard real-time traffic rates from none, to 0.05 packets/slot, and finally to a load of 0.1 packets/slot when there are six real-time nodes.

Also, as BER increases, the results appear to show the MAC protocol withstanding channel degradation surprisingly well. In fact, though, the results are optimistic. At the MAC layer, success and failure are determined completely by the collision/noncollision feedback minislot. Even with a BER of 10^{-3}, we can expect

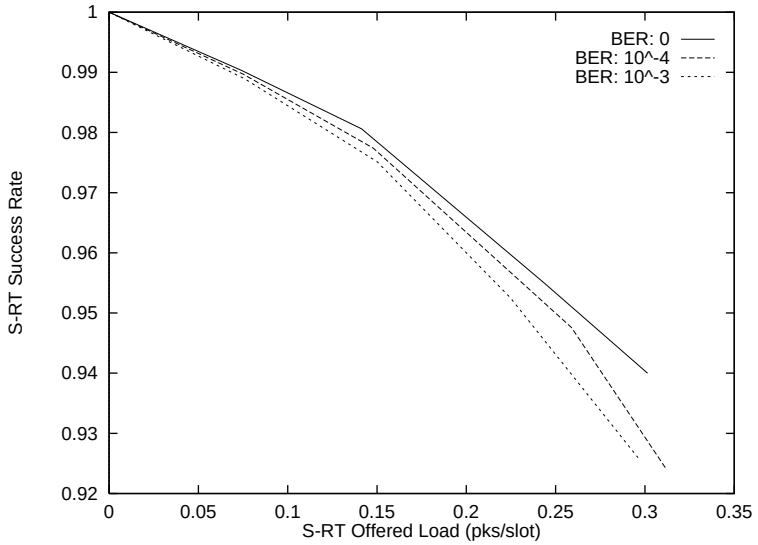


Figure 6.2: MAC S-RT performance with no H-RT load as channel degrades.

incorrect feedback only once every 1,000 slots. As a result, the MAC layer sees very nearly the same success rate in these examples, though the higher layers will not. Most errors will be in the data itself. So while the nodes remain synchronized, the MAC layer is unaware of the increased rate of retransmitted higher layer data.

6.4 LLC Layer Modeled

Most real-time LLC schedulers in the literature are studied with the simplifying assumption that once a packet is deemed best for transmission, the packet is submitted to the MAC layer and transmitted in one time unit. For certain wire- or fiber-based networks this is a valid assumption. But for the finite node wireless networks we study, incorporating the wide variation of transmission times into the model is desirable. Due to the nature of the splitting algorithm, there is a large range in the number of slots needed to transmit a packet. Additionally, by studying an LLC on a fully detailed MAC layer, realistic results are obtained showing more concretely what can be achieved.

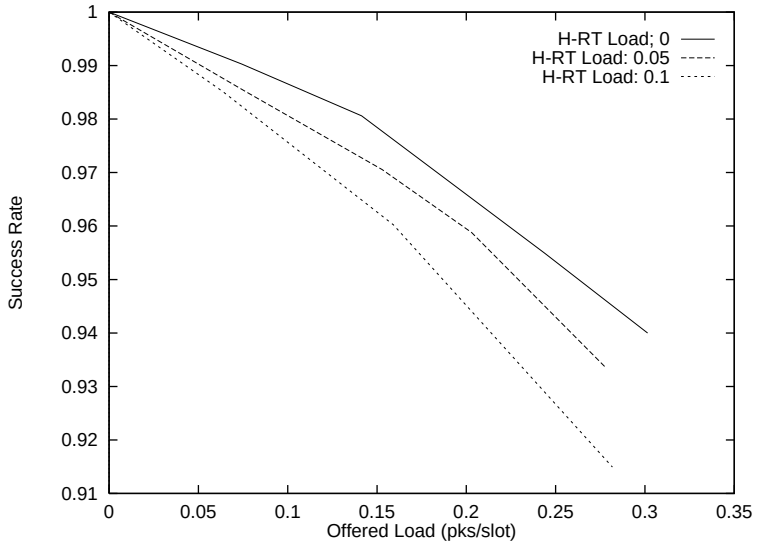


Figure 6.3: MAC S-RT performance as H-RT load increases, ideal channel.

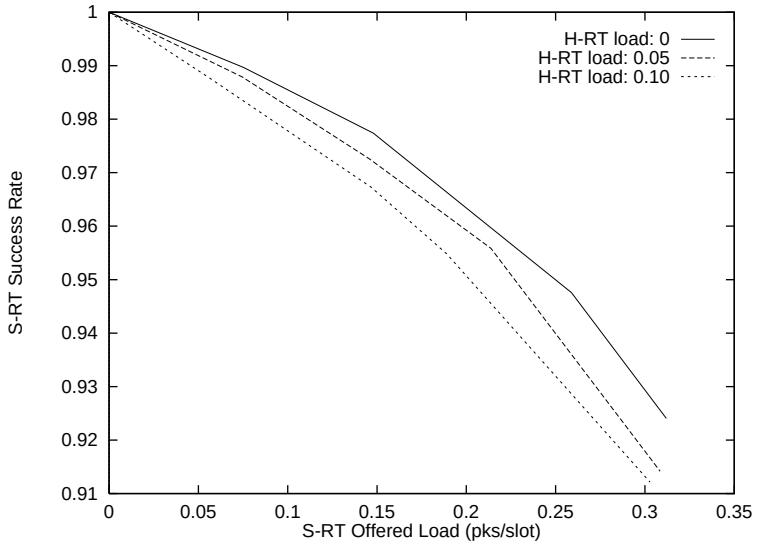


Figure 6.4: MAC S-RT performance as H-RT load increases, $BER = 10^{-4}$.

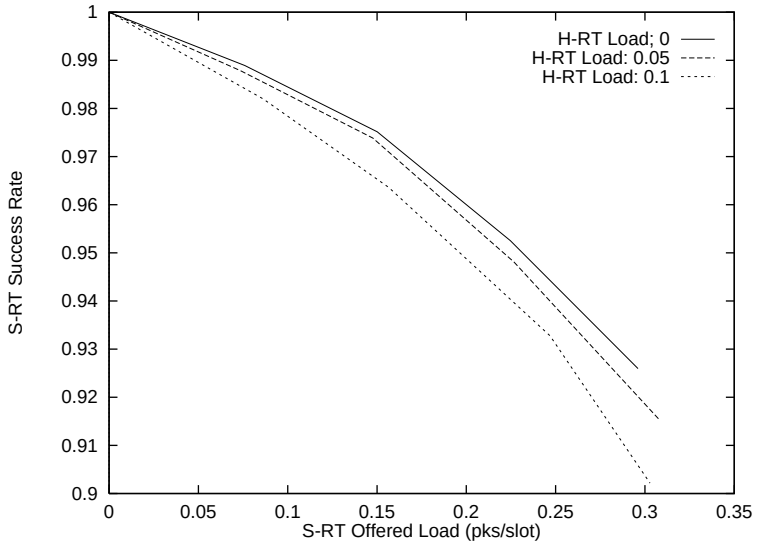


Figure 6.5: MAC S-RT performance as H-RT load increases, $BER = 10^{-3}$.

While the MAC layer is distributed in nature with each node cooperatively taking part in the algorithm, the LLC layer is concerned only with the traffic in a given node. The LLC contains three queues for the hard, soft and non real-time traffic arriving from the network layer, as well as queues for reassembling arriving MAC packets addressed to the node. For every packet submitted to the MAC layer, the LLC receives a response indicating that the packet was transmitted before its deadline or else was discarded because its deadline expired.

Recall, too, that the MAC layer has a queue of length one for each of the three real-time streams. The LLC, since it receives success/failure information for each packet submitted, is aware of which of its node's MAC queues are empty or full. The general rule is that a lower priority packet is never submitted when a higher priority MAC queue is full. However, since preemption occurs within a node as well as between nodes, higher priority packets can always be submitted. In an extreme case, a non real-time CRI can be preempted by a soft real-time CRI, which in turn is preempted by a hard real-time CRI resulting in three packets at one or

more nodes' MAC layers.

6.4.1 Mapping Deadlines between LLC and MAC Layers

Based on packet deadlines and queue lengths, as well as the MAC layer success/failure responses, the LLC layer uses a scheduling algorithm to decide which of the layer's potentially many packets is the best one to submit to the MAC for transmission. Heuristics must be used since scheduling under resource constraints has been shown to be an NP-complete problem [GJ75] and thus computationally intractable. (The popular *rate-monotonic scheduling* method [LL73] is suitable only when all deadlines are known before-hand, and arrivals occur with known periods. We study a more general system and cannot use this method.)

Also, the MAC and LLC operate using different units of time. At the MAC layer everything is done based on slot lengths since that is how long a transmission takes and is also the interval at which feedback arrives. But the LLC receives packets whose deadlines are in units of seconds. We previously showed that real-time splitting algorithms reach a performance plateau when the range of initial laxities is about $[2, 30)$. However, the LLC can potentially deal with packets whose deadlines are much further in the future than that. Before submission to the MAC layer, the LLC maps the time units of seconds into those of slots and must decide how to handle packets whose deadlines are greater than 30 slots.

6.4.2 Opportunities Presented by Deadline Mapping

The conversion of deadline time units allows for possibilities beyond the scope of this research. For instance, a message to be transmitted that has a distant hard deadline, might be initially assigned by the LLC to the non or soft real-time queue with an artificially short deadline. If the message expires in the queue, the LLC scheduler can assign a new deadline and now put the packet in the hard real-time queue. In a similar manner, firm real-time can be implemented. If a certain type of

Table 6.1: LLC algorithms studied.

LLC	FCFS
	MLF
	Priority

message must be transmitted successfully at least once every n times, for the first $n-1$ times the message can be put in the soft or non real-time queues. If all $n-1$ attempts are unsuccessful, then the message can be put in the hard real-time queue. These sorts of scheduling techniques allow QoS guarantees to be met using the least amount possible of system resources.

In addition to making the LLC itself more capable, the LLC/Network layer interface can further expand on the basic real-time services available at the MAC/LLC layer interface. The ability to accept or reject connection-oriented real-time data can be added, a choice of schedulers to accommodate peculiar traffic patterns at particular nodes, detailed reasons for packet failure, ability to permit or deny late arrivals, acknowledgements, and so on.

At this early stage in the research, we are less interested in the more elaborate services that can be built with a strong real-time data link layer—though that is exactly what the end user is ultimately concerned with—than we are with studying the relative performances of different schedulers. The core of the real-time LLC is the scheduling algorithm because the heuristics used have a huge impact on the maximum supportable loads for given QoS guarantees. Additional details added to the LLC make it more useful and appealing to the user, but the scheduler used is what most affects maximum throughput.

6.4.3 Scheduling Algorithms

The three scheduling algorithms studied for use in a real-time LLC layer are shown in Table 6.1, which is the LLC portion of Table 1.2. All three schedulers treat hard real-time packets similarly. If a hard real-time packet's deadline will arrive within the upcoming system cycle, the packet will always be submitted to the MAC layer. When a hard deadline is not imminent, however, the scheduling algorithms act differently.

The *FCFS* scheduler is the first algorithm considered. It is not, however, purely first-come-first-served. The difference between it and pure FCFS is that hard real-time deadlines are always met as described in the previous paragraph. Beyond that, the algorithm makes no distinction regarding real-time stream and behaves in a pure FCFS manner. The algorithm is simple to implement and interesting to study in light of the fact that the MAC already transmits data based on real-time properties.

The *MLF,* or Minimum Laxity First, scheduler is studied next. Each of the data queues is looked at, and among them all, the packet with the most pressing deadline is chosen for submission to the MAC layer.

Finally, the *Priority* scheduler is considered. It places a very high priority on hard real-time data. In most systems, hard real-time data makes up a very small fraction of the transmitted data. Accordingly, considering an algorithm that puts a high priority on transmitting hard real-time data seems reasonable. Each node simply cycles through its LLC layer queues. If anything exists in the hard real-time queue and the MAC is able to accept a hard real-time packet, then the most pressing hard real-time packet is submitted. If the hard real-time queue is empty, then the soft real-time queue is serviced in the same manner as the hard. Similarly, if the soft queue is empty, then the non real-time queue is used.

None of the three algorithms are overly difficult to implement. Details of

implementation are important to consider because the LLC, near the base of the protocol stack, must operate quickly, efficiently, and with little overhead.

6.4.4 Scheduling Results

All scheduling results were obtained through simulation using eight nodes where six could transmit hard real-time data and all eight could transmit soft and non real-time data. Each simulation was run with 95% confidence intervals that the fraction of successful transmissions was ± 0.005 of the steady state value. While the protocol supports only connectionless communication, coupled acknowledgements (acks) were implemented. A single bit, when acknowledged service was requested, following the data packet is used as the ack. In these simulations, 50% of transmitted packets used acks. While not presented, results from simulations using 0% or 100% acks did not differ significantly because it is rare for the single ack bit to be destroyed by channel noise. Also, while we implemented segmentation and reassembly at the LLC layer, we ultimately decided to use LLC packets that are the same length as MAC packets. Since packet length in a real-time system is closely tied to the data being transmitted, we wanted to make as few assumptions as possible. Therefore, results based on the MAC packet unit length seemed most useful.

Comparing the LLC performance results, hard real-time performance is not graphed since hard real-time packets are treated the same by all schedulers: regardless of what is waiting in the queues, a hard real-time packet will always be submitted to the MAC when the deadline is imminent. As a result, there are no significant relative differences. Performances of each scheduler for soft real-time data are compared in Figures 6.6 through 6.8, where the same trend can be seen among the graphs. MLF always performs better than FCFS, which performs better than Priority. What is more interesting is not the ranking but the relative differences as the hard real-time loads increase.

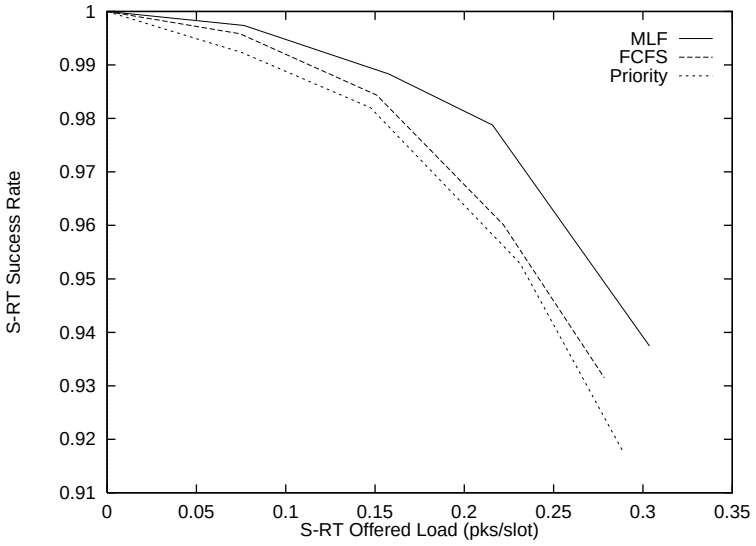


Figure 6.6: LLC schedulers for S-RT with no H-RT traffic.

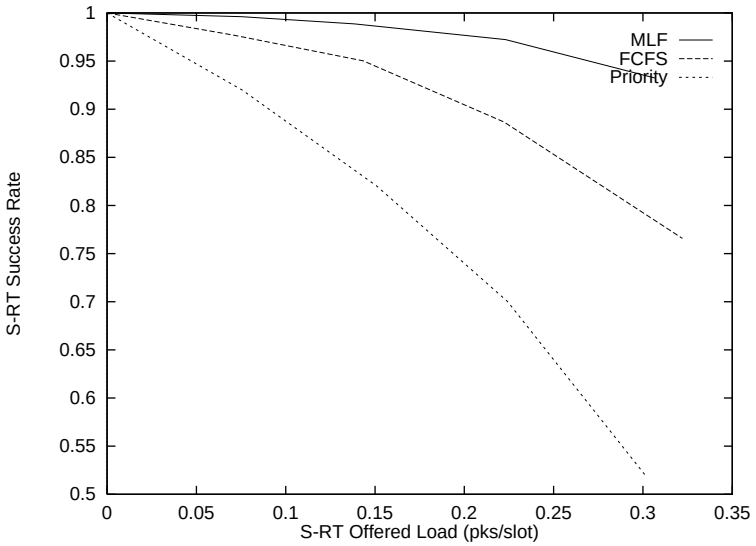


Figure 6.7: LLC schedulers for S-RT with 0.05 pks/slot H-RT load.

Even with no real-time traffic load, the MLF technique performs noticeably better than the others. As seen in Figure 6.6, FCFS and the Priority algorithms have a similar performance. However, in Figure 6.7, the differences become apparent when the hard real-time load increases to a moderate level. While the MLF success rate at a load of 0.3 packets/slot drops from about 0.95 to 0.92, the FCFS scheme drops from about 0.92 to 0.80. Suffering even more drastically, the Priority scheduler success rate goes from about 0.91 when there is no hard real-time traffic, down to only slightly more than 0.50 when the hard real-time load increases. In all cases, this is somewhat worse performance than typical because, as mentioned previously, the MAC layer for these results uses a minimum length hard real-time system cycle. But in general, Priority scheduled soft real-time traffic will never perform well in the presence of hard real-time traffic. Whenever any node has a hard real-time packet, that node will set the hard preemption minislot. As a result, as soon as there is more than minimal hard real-time traffic, soft and non real-time streams have little opportunity to contend for the channel. Hard real-time delay is decreased, but at the expense of greatly decreased throughput for the other real-time streams.

When the hard real-time load is increased to its maximum in Figure 6.8 to handle the worst case high multiplicity collision, while FCFS and especially Priority scheduling suffer markedly, MLF still performs quite well. We can see, for example, that in Figure 6.9 where only FCFS results are graphed, the success rate drop is severe as the hard real-time load becomes heavier. But Figure 6.10 shows that MLF, while of course suffering some success rate decrease, is much more resilient in the face of high preemption rates.

Figure 6.11 graphs non real-time LLC results for MLF when there is no hard real-time traffic and when the soft real-time load increases. Since non real-time success rate is ideally 100%, the data is plotted with the more traditional delay versus throughput curves. The throughput for finite delay is what one would

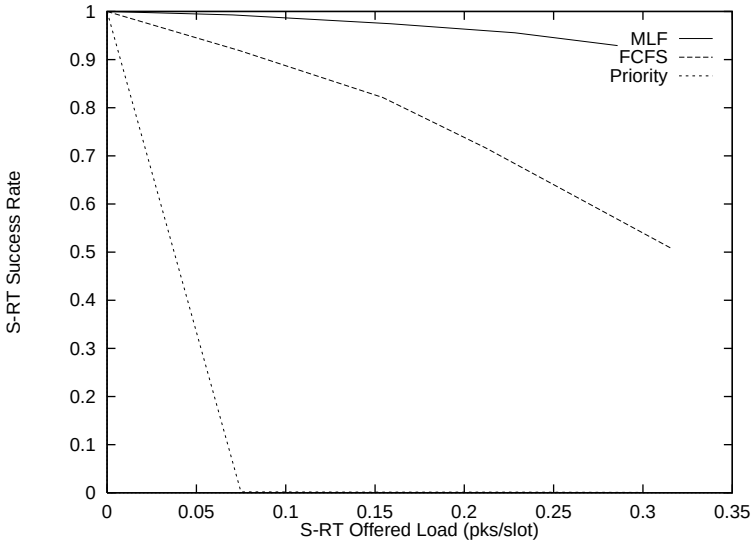


Figure 6.8: LLC schedulers for S-RT with 0.1 pks/slot H-RT load.

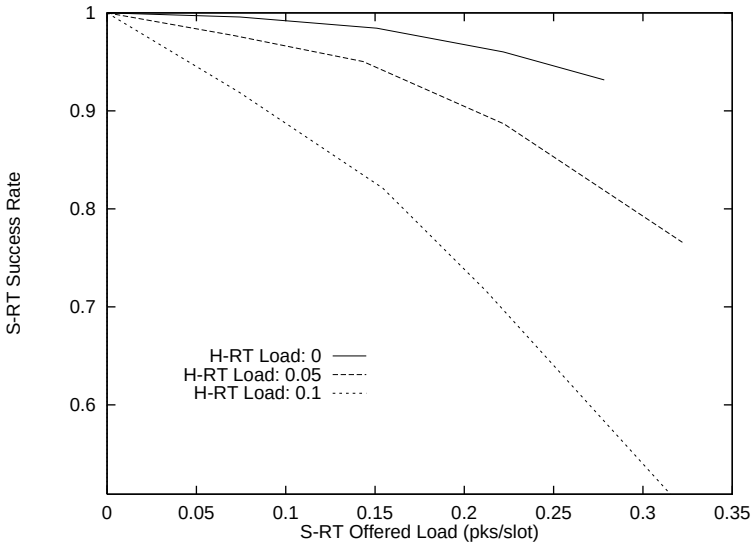


Figure 6.9: FCFS S-RT performance for increasing H-RT load.

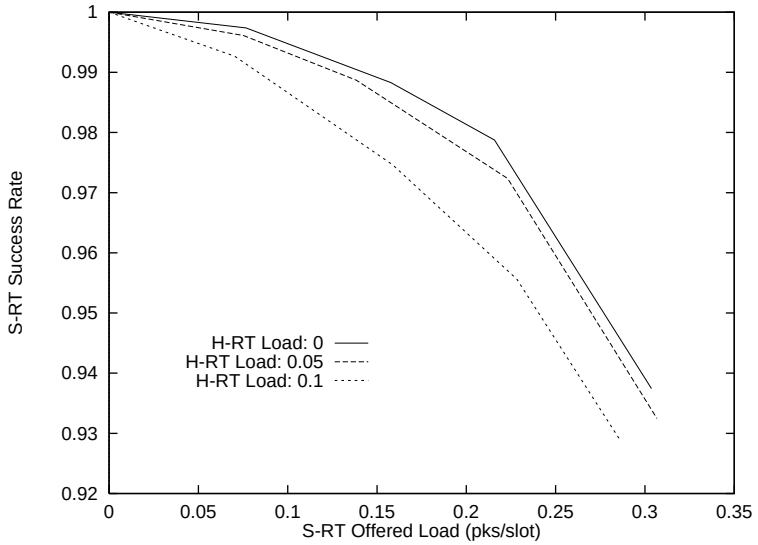


Figure 6.10: MLF S-RT performance for increasing H-RT load.

expect, considering that the theoretical throughput maximum for the non real-time protocol is 0.43 packets/slot when there is no other traffic. For all three streams combined, simulations show a total maximum throughput of approximately 0.52 packets/slot; this is higher than the theoretical 0.5 packet/slot cutoff for splitting protocols because the protocol is able to drop packets [Gon92].

6.5 Data link Layer Performance Evaluation

An increase in channel noise will clearly result in a decrease in real-time success rates. However, the MAC results show that even on a much noisier than typical channel, hard real-time data still reaches 96% success at heaviest load and with a minimum hard real-time cycle. For all but the most critical of data, a fault tolerance system should be able to easily handle this amount of loss. For soft and non real-time data, managing the success rate decrease is easier to manage. In these cases, the success rate on non-ideal channels is simply used as part of calculations in QoS negotiations. Similarly, reduced non real-time throughputs (or the corresponding

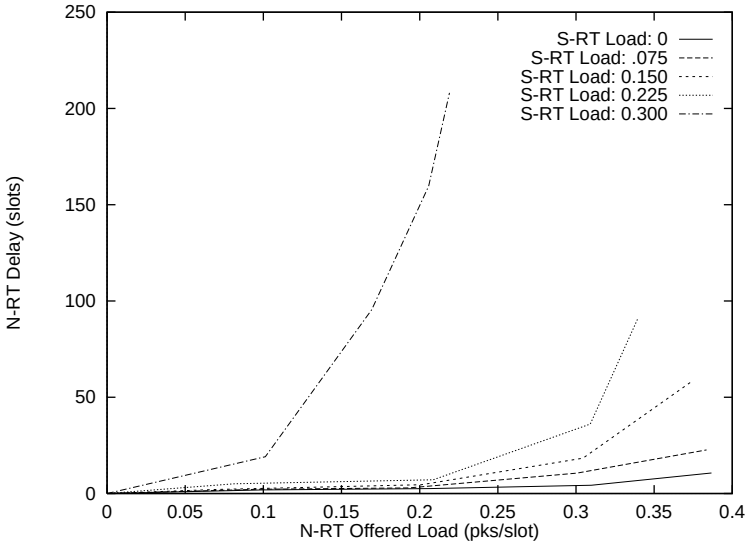


Figure 6.11: MLF N-RT performance for no H-RT load, increasing S-RT load.

increased delay) would be used in QoS negotiations for that stream.

The schedulers show dramatically different performance results especially at high hard real-time loads. However, each can be appropriate depending on system requirements. The Priority heuristic quickly prevents all lower Priority real-time traffic from even contending for the channel. But for a system whose traffic is mainly non real-time with occasional hard or soft real-time traffic, this very simple scheduler can be sufficient. FCFS does much better for mixed traffic and is only slightly more involved than Priority to implement. Since the MAC layer is already dealing with deadlines, this scheduler can be used especially when hard real-time traffic is infrequent. MLF, though, is unquestionably the best of the schedulers studied. Though it might be the most difficult to implement since each queue entry has to be searched prior to every MAC submission, MLF is the best candidate for a fully general real-time data link layer.

6.6 Conclusions

The model and results presented in this chapter represent moving the protocols from a purely theoretical realm to an environment simulating more closely what would be experienced in a real implementation: non-ideal channel, non-ideal arrival process with retransmissions, and loss of station synchronization. For real-time *ad hoc* wireless networks, this is the first published study of schedulers used in a simple LLC coupled with a realistic MAC layer. The usefulness of these results lies in the fact that they go beyond showing proof of concept. They can be used as limits in the application layer design of real-time systems. If system requirements can be met, then a real implementation can be developed.

Chapter 7

CONCLUSIONS

A real-time system meets its functional requirements not only by generating correct results, but by also generating timely results. Deadlines can be most fully supported if an awareness of time exists in all layers of the protocol stack. Incorporating the concept of time into random access protocols, however, has been traditionally difficult to do, because 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. We have shown that at the MAC layer, splitting protocols can be modified to work successfully in soft real-time environments, and that at the LLC layer, schedulers can be designed to make use of MAC layer results in real-time decision making.

7.1 MAC Layer Contributions

Initially, we designed straightforward real-time protocols that were conceptually similar to non real-time splitting algorithms. After analyzing several, learning what works well and what does not in such a setting, we then tested the algorithms within the two major classes of splitting algorithms: blocked access and free access. In non real-time systems, free access algorithms have been shown to offer less throughput than blocked access. Due to the more involved departure process of a real-time system, though, we wanted to see if the same holds in the real-time environment. But even with departures from deadline expiration decreasing the

load somewhat, real-time free access algorithms still offer less throughput than their blocked access counterparts.

Following this, we expanded on our initial results by developing a fully distributed MAC protocol, the HSN protocol, that allows for the coexistence of hard, soft, and non real-time data while still offering quality of service guarantees. The protocol supports preemption both within a node and in a distributed fashion between all nodes. To support the protocol and avoid the need for a centralized station providing collision/non-collision feedback, we developed a simple protocol where all stations are involved with the generation of the feedback minislot data. Parameter values can be adjusted for use in military environments where minimizing the number of nodes transmitting feedback is desirable, or for use in general settings where frequent transmissions are not a concern. While splitting algorithms require such feedback to operate, the splitting algorithms and feedback algorithm are completely independent of each other; other feedback algorithms can be used with the HSN protocol and vice-versa if desired.

In addition to the algorithms, we have presented 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, a new algorithm designed by us, offers the best performance.

Addressing the practicality of the protocols studied, we 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. Stations can use this property to easily join the net by

simply waiting for two non-collisions before their first transmissions.

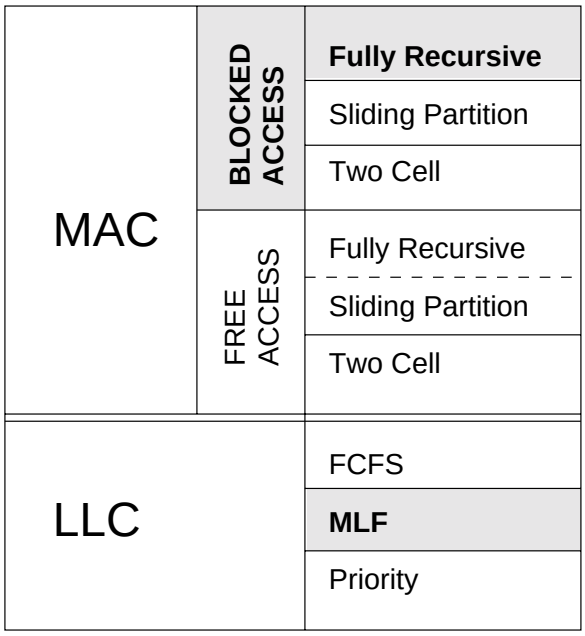
7.2 LLC Layer Contributions

In most published work involving the analysis of real-time schedulers, the MAC layer is ignored and assumed to offer a normalized unit transmission time. While this assumption is fine for straightforward comparisons of scheduling algorithms, this did not meet our requirements. We not only wished to determine a suitable scheduling heuristic but to use it in the design of a complete real-time data link layer. The important differences between the standard idealized assumption and an initial protocol design are that in an implemented system, the arrival process at the LLC layer includes retransmissions and the mapping of deadlines from units of seconds to units of slots. By simulating schedulers that have these additional functionalities, performance results representative of those of a real system can be obtained.

We compared the success rates of three scheduling techniques: Priority, where the existence of a hard real-time packet anywhere in the network preempts all other packets, and existence of a soft real-time packet anywhere in the system preempts any non real-time packets; FCFS (first-come-first-served), operating as its name implies; and MLF (minimum laxity first), also operating as one would expect. Each of the schedulers was further modified, though, so that hard real-time packets always met their deadlines. MLF performed best by far, FCFS behind that, and the Priority scheduler performing worst.

Each scheduler mapped deadlines into slot units. When a MAC layer packet was not transmitted by its deadline, the LLC scheduler directly made use of the per packet MAC layer results so that, if the real world deadline allowed, a new MAC layer deadline could be assigned and the packet resubmitted for transmission. While only connectionless service was provided for, acknowledgements were supported. The MAC layer per-packet success/failure indications were expanded upon so that more

Table 7.1: Best performing data link layer algorithms studied.



information was available at the LLC/network layer interface. Failures could be ascribed to MAC layer expiration, LLC queue timeout, missing acknowledgement, etc. An LLC response with this level of detail allows a real-time network layer to make better decisions for resource allocation to meet peer to peer quality of service guarantees when at least some links are on *ad hoc* wireless networks.

7.3 Evaluation of Data Link Layer Algorithms

Table 1.2 was used to indicate which algorithms would be studied and within which layer of the protocol stack. After studying each algorithm in turn, the information is updated slightly in Table 7.1. The newly shaded boxes in the table indicate the best performing MAC and LLC algorithms.

To support wireless hard and soft real-time data at the MAC layer, there appears to be no reason to use anything other than the blocked access Sliding

Partition CRA. It outperforms others in nearly all situations and is no more difficult than the other MAC algorithms to implement. Deciding which LLC scheduling algorithm to use requires a little more thought. As discussed in Section 6.5 of Chapter 6, each can be useful is different situations. When both hard and soft real-time traffic rates are low, the Priority scheduler is easiest to implement and would work well. When hard real-time traffic is infrequent, FCFS can also perform well. But for the general case, the best scheduler is MLF. The only reasons to consider the other schedulers are if it is too time consuming for MLF to study all queues before submitting a packet to the MAC layer (unlikely when the speed of computers is compared to the speed of low bandwidth wireless links), or if it is known with certainty that the traffic will be made up mostly of non real-time data. But even then, performance will not be decreased by using MLF. Accordingly, the recommendation to handle general real-time situations is to use the HSN protocol, made up of two Sliding Partition CRAs and one Two Cell CRA, at the MAC layer, and the MLF scheduler at the LLC layer.

An HSN/MLF wireless real-time data link layer would be able to offer appropriate quality of service guarantees to hard, soft and non real-time data. The battlefield, forestry service, and search and rescue examples described in Chapter 1, as well as any similar real-time systems, could be fully supported, something not possible with current data link layer protocols.

7.4 Future Work

While splitting protocols can operate well in wireless networks where short messages are passed between nodes that are cooperating as parts of a larger soft real-time system, splitting protocols are not well suited for the transmission of high bandwidth streaming data. Data of this sort is best handled by connection-oriented service. That is not to say, however, that the two connection types cannot coexist. In our MAC protocol, note that only three bit patterns are used of the four available

with two minislots. At the MAC layer, an easy extension is to use the fourth bit combination to indicate the highest level of preemption, even higher than hard real-time. That pattern would indicate arriving connection-oriented traffic. The extension is easy here because heavy burden is added to the LLC layer. It would be expected to submit a packet only when safe to do so. And the LLC would now have the duty of determining if, on the local single hop network, enough resources exist to admit an additional connection while still maintaining already agreed upon QoS levels.

Alternatives could be as simple as adding more slots to the system cycle length to provide bandwidth for a given number of connections, or as complex as determining dynamically if new connections can be admitted. Similarly, recall that in our MAC protocols, performance is based on λ, the aggregate traffic rate. A decision needs to made regarding where this λ is controlled. The most likely place is again at the LLC layer. Only it is fully aware of the extent of current QoS guarantees.

But perhaps more importantly, protocol characteristics desirable to end users must be added. While the MAC layer might guarantee, say, a 96% success rate for the current aggregate traffic rate, a given LLC stream might need only an 80% success rate. To better utilize resources, the LLC should modify its scheduling behavior to meet this lower rate. To combat the relatively high incidence of errors on a wireless channel, another option is to have an error correction code on both the header and data portions of the MAC packets. That way, when an error occurs, it is more likely to damage the long data packet and leave the header information intact. Even without the data, the intact header can still be passed to the LLC to aid in its decision making. For instance, if communication with a particular node is observed to suffer for a length of time, transmissions to it could be temporarily suspended, reduced, or addressed in some other way.

While these are important additions for a real implementation, the work described in this dissertation represents the core requirements of a real-time data link layer. The addition of new services will make the protocol more useful and flexible, but are not expected to change the relative performances of the MAC layer algorithms or the LLC scheduling algorithms. Providing a strong foundation for true real-time support with these algorithms now makes possible further developments and enhancements for actual implementation. But most importantly, this dissertation research takes the first step of offering theoretical and simulation support for what is possible.

BIBLIOGRAPHY

[Abr70] N. Abramson. Another alternative for computer communications. In *Proceedings of the Fall Joint Computer Conference, AFIPS Conference*, volume 37, pages 281–285, 1970.

[Abr85] N. Abramson. Development of ALOHANET. *IEEE Transactions on Information Theory*, 5(2):28–42, March 1985.

[Arv91] K. Arvind. *Protocols for Distributed Real-Time Systems*. Ph.D. dissertation, University of Massachusetts, Amherst, MA, 1991.

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

[Cap79a] J. Capetanakis. Generalized TDMA: The multiple access tree protocol. *IEEE Transactions on Communications*, 27(10):1476–1484, October 1979.

[Cap79b] J. Capetanakis. Tree algorithms for packet broadcast channels. *IEEE Transactions on Information Theory*, 25(5):505–515, September 1979.

[CKT89] R. Chipalkatti, J. Kurose, and D. Towsley. Scheduling policies for real-time and non-real-time traffic. In *IEEE Proceedings Infocom*, pages 774–783, 1989.

[CS97] S. H. Choi and K. G. Shin. A cellular wireless local area network with QoS guarantees for heterogeneous traffic. In *IEEE Proceedings Infocom*, April 1997.

[DPK93] H. Delic and P. Papantoni-Kazakos. A class of scheduling policies for mixed data with renewal arrival processes. *IEEE Transactions on Automatic Control*, 38(3):455–459, 1993.

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

[Gal96] R. G. Gallager. *Discrete Stochastic Processes*. Kluwer Academic Publishers, 1996.

[GJ75] M. R. Garey and D. S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *Society for Industrial and Applied Mathematics Journal of Computing*, 4:397–411, 1975.

[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.

[Gon92] Y. Gong. *MAC Protocols in Communication Networks: Design and Performance Analysis*. Ph.D. dissertation, University of Delaware, 1992.

[KT75] L. Kleinrock and F. A. Tobagi. Packet switching in radio channels: Part 1: CSMA modes and their throughput-delay characteristics. *IEEE Transactions on Communications*, 23:1400–1416, 1975.

[LL73] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the Association for Computing Machinery*, 20(1):46–61, 1973.

[Mas81] J. Massey. Collision resolution algorithms and random access communications. In I. G. Longo, editor, *Multi-User Communication Systems, CISM Courses and Lectures*, pages 73–137. Springer–Verlag, New York, 1981.

[Mil96] Mil 3, Inc. Opnet modeler, v3.0, 1996. Washington, DC.

[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.

[MS95] 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.

[MS96] 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.

[MS97a] M. J. Markowski and A. S. Sethi. Real-time wireless communication using splitting protocols. In *IEEE Proceedings Global Telecommunications Conference*, pages 1621–1624, Phoenix, AZ, November 1997.

[MS97b] M. J. Markowski and A. S. Sethi. Wireless MAC protocols for real-time battlefield communications. In *IEEE Proceedings Military Communications Conference*, Monterey, CA, November 1997.

[MS98a] M. J. Markowski and A. S. Sethi. Fully distributed wireless MAC transmission of real-time data. In *IEEE Proceedings Real-Time Technology and Applications Symposium*, Denver, CO, to appear June 1998.

[MS98b] M. J. Markowski and A. S. Sethi. Fully distributed wireless transmission of heterogeneous real-time data. In *IEEE Proceedings Vehicular Technology Conference*, Ottawa, Canada, to appear May 1998.

[MS98c] M. J. Markowski and A. S. Sethi. Blocked and free access real-time splitting protocols. *Integrated Computer-Aided Engineering*, 5(3), to appear July, 1998. John Wiley & Sons, Inc.

[PGPK87] 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.

[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.

[PK92] 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.

[PKLT95] T. Papantoni-Kazakos, N. B. Likhanov, and B. S. Tsybakov. A protocol for random multiple access of packets with mixed priorities in wireless networks. *IEEE Journal on Selected Areas in Communications*, 13(7):1324–1331, September 1995.

[PPK89] M. Paterakis and P. Papatoni-Kazakos. A simple window random access algorithm with advantageous properties. *IEEE Transactions on Information Theory*, 39:1124–1130, September 1989.

[PTA93] 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.

[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.

[TM78] B. Tsybakov and V. Mikhailov. Free synchronous packet access in a broadcast channel with feedback. *Problemy Peredachi Informassi*, 14(4):259–280, October–December 1978.

[TM80] B. Tsybakov and V. Mikhailov. Random multiple access of packets: Part-and-try algorithm. *Problemy Peredachi Informassi*, 16(4):65–79, October–December 1980.

[VPK92] S. Vassilopoulos and P. Papantoni-Kazakos. A transmission scheduling algorithm for mixed traffic: High and low priority. In *IEEE Proceedings Infocom*, volume 3, pages 2251–2259, 1992.

[Wol85] J. K. Wolf. Born again group testing: Multiaccess communications. *IEEE Transactions on Information Theory*, 31(2):185–191, 1985.

[Won64] E. Wong. A linear search problem. *SIAM Review*, 6(2):168–174, April 1964.

[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.