Time-Constrained Communication in Multiple Access Networks

James Francis Kurose

CUCS-120-84

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY
1984

ABSTRACT

Time-Constrained Communication in Multiple Access Networks

James F. Kurose

The characteristics of time-constrained communication applications, such as packetized voice, differ significantly from those of standard data communication applications. First, messages not received within a fixed amount of time after their generation at a sending station are considered lost. Secondly, a certain amount of message loss is tolerable. In this thesis we address the problem of supporting time-constrained communication applications in a multiple access network. The principal contributions of this thesis fall into two categories.

The first contribution is the development and analysis of a new class of protocols for supporting multiaccess time-constrained communication. These protocols are based on a generalization of the time window mechanism and provide a family of network-wide message transmission scheduling disciplines based on message generation times. The problem of determining the optimal elements of the protocol's window selection policy is addressed. A semi-markov decision model is formulated for protocol operation and the optimal elements of the windowing policy are found to be both simple and intuitive. The extension of the protocol for transmitting both time-constrained and non-time-constrained messages is considered. In our scheme, time-constrained traffic, when transmitted, receives preemptive priority over other classes of traffic.

Several novel analytic performance models are developed and validated through simulation. The protocol's time-constrained performance is found to critically depend on its imposed scheduling function and is significantly better under the optimal windowing policy elements than under other policy elements. For multiple classes of traffic, our results indicate that trading time-constrained message loss against the average delay of non-time-constrained traffic is not usually a viable option.

The second major contribution of this thesis is the development of a systematic, formal approach towards distributed optimization via a fictitious resource sharing paradigm and a decentralized "microeconomic" solution to the resource sharing problem. This approach, which draws on previous work in mathematical economics, is successfully used to compute the optimum transmission probabilities for both the time window and Slotted Aloha protocols. Interestingly, several network mechanisms, such as flow control and priorities, are found to emerge naturally from this approach.

To Julie
with love

Acknowledgments

I wish to express my sincerest thanks to my research advisors, Mischa Schwartz and Yechiam Yemini, for their constant guidance, support and encouragement throughout my graduate studies at Columbia University; their influence on this work and on me has been tremendous. I consider myself fortunate indeed to have had the privilege and pleasure to have worked with two such distinguished and dedicated individuals.

I wish to also thank my thesis committee: Joseph Traub, Thomas Stern, Gerald Leitner and Aurel Lazar for serving on this committee and for their many helpful comments, suggestions and discussions while this work was in progress. I am also particularly thankful to Professor Traub for his role in building the Department of Computer Science into such a fertile research environment.

The students, staff and faculty in the Department of Computer Science have together managed to make the department more than simply a place to work and go to school. I would like to sincerely thank all my friends at Columbia, especially Kenny Wasserman, Tom Ellman, Cécile Paris, Dayton Clark, Dan Miranker and Tatsuya Suda, for their continuing friendship and support and for sharing the ups and downs of graduate student life.

Finally, I owe a great amount of thanks to my parents for teaching me the joy of learning and so much more. Last, and certainly not least, I owe an immense amount of thanks to my wife Julie; her constant encouragement, support, and love has made this work possible. In the balance, I have probably spent far too many hours working which we could have spent together; I eagerly await overcorrecting this imbalance in the future.

Table of Contents

List of Figures

Chapter 1
Introduction

A little over a decade ago, the first distributed algorithms which allowed a number of geographically separated stations to communicate over a single, shared "broadcast" channel were implemented in the ALOHA network [Abramson 70]. These algorithms, or *multiple access protocols*, permitted remote terminal stations on the Hawaiian islands to communicate with each other and with a single centralized computer system. Since then, the use of these and similar protocols has spread into hundreds of today's communication networks. Recently, we have witnessed the beginning of the evolution of these networks from relatively simply communication tools into distributed intelligent systems providing sophisticated high-level network services and integrated voice, data and image transmission capabilities [Pokress 84]. In order for this evolution to continue, however, numerous theoretical and practical technical challenges must be addressed.

This dissertation is primarily concerned with the development, analysis and optimization of protocols for providing "*time-constrained*" communication capabilities (for applications such as voice transmission) in distributed, packet-switched, multiple access networks. As we will see, however, many of the ideas developed in this thesis also transcend this particular problem domain and have important applications not only for time-constrained problems in other network environments but for problems occurring in other areas of distributed computation and communication, such as distributed resource sharing and decentralized optimization techniques, as well.

In order to make this dissertation as self-contained as possible, this chapter and chapter 2 together provide the background material requisite for reading the remainder of this thesis. In this chapter, we first examine the important

characteristics of time-constrained communication applications. We then define the multiple access problem and consider the problem of communicating in such a network environment. We next discuss the motivation for this work and summarize the major contributions of this dissertation. The organization of the remaining chapters is then presented. Chapter 2 reviews the relevant past literature and provides a closer examination of both time-constrained communication and the multiple access problem. The details of our research are then presented in the subsequent chapters.

1.1. Time-Constrained Communication with Message Loss

The problem setting for our research is the network environment shown in figure 1-1. In this environment, numerous time-constrained network application programs, running at geographically distributed stations, communicate with each other by sending messages (or packets) through a communication network. All stations are typically capable of both sending and receiving such messages, although only one such capability is shown for each of the stations in figure 1-1.

The most important aspect of these time-constrained applications is that *a message generated at the source station by an application program must be received at the destination station within a given amount of time after its generation at the sending station.* If a message's delay (defined as the time between its generation at the sending station and its reception at the destination station) exceeds this time constraint, the message is considered *lost*, regardless of whether or not it is ever received at the destination station.

The message flow for time-constrained communication applications is shown in figure 1-1. The messages generated by a time-constrained application (A) at a sending station are first buffered or stored (B) within the sending station. Once a message has been buffered, the network access mechanism (C) eventually decides whether the message should be transmitted into the communication subsystem (E) or should be explicitly discarded at the sending station (D). If a message is eventually successfully received at the destination station but its delay exceeds its time constraint, the message is lost (G); otherwise, it is passed on to the time-constrained application (H) at the destination station.

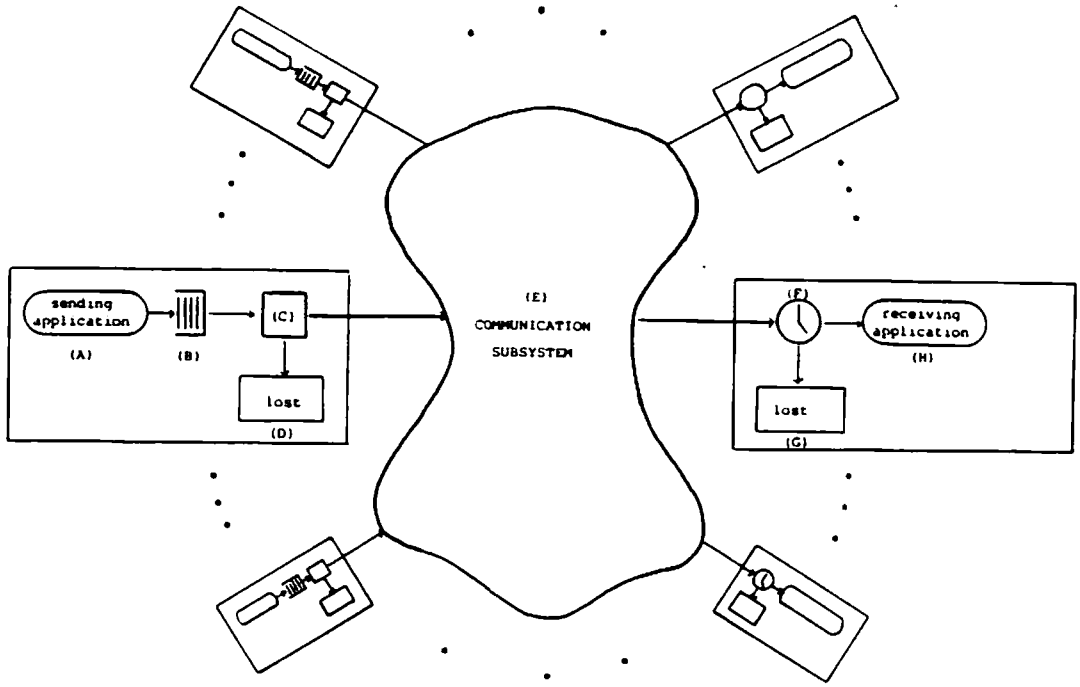


Figure 1-1: Time-constrained communication in a network environment

There are several important communication applications having such time-constrained characteristics. Perhaps the most important of these applications is packetized voice [Coviello 79, Bially et al. 80a], in which human voice is digitized, packetized at the source station, transmitted over the network subsystem, and reconstructed and played out synchronously at the destination station. Since excessive delays can have seriously disruptive effects on human conversation, voice packets are usually constrained to arrive at the destination station within a given amount of time after their generation at the sending station. Those packets which do not arrive within the time bound are considered lost; a small number of lost packets has been shown to have little, if any, effect on human speech intelligibility.

A second application requiring time-constrained communication is distributed sensor networks (DSN) [DSN 82], in which distributed stations attempt to track

a moving object using their local observations and the communicated observations of the other stations. Since the position of the object is continually changing, only a small amount of time is available to fix its current location or trajectory. The distributed observations necessary to determine the current location must thus be communicated within this small amount of time. A small amount of message loss due to excessive delays may be tolerable, but may result in uncertainty in the object's calculated position. A third class of time-constrained applications is real-time control applications in which stations must initiate some action at remote devices within a specified amount of time.

Let us now examine some of the performance considerations for such time-constrained communication applications. Unlike the standard data communication applications to be discussed in chapter 2, *the principal performance consideration for time-constrained traffic is the percentage of messages which are received at destination stations with a message delay below a given time constraint.* We will characterize the percentage of messages with delays exceeding this constraint as the message *loss*. Obviously, message loss is closely related to the *imposed time constraint*, and thus this constraint will be an important performance parameter. Finally, we will see that message loss is also dependent on the *offered load*, or the rate at which time-constrained messages are generated at the sending stations.

The three primary performance considerations for time-constrained communication are thus loss, the imposed time constraint, and the offered load. As might be expected, tradeoffs exist among these three values. For example, if any one value is fixed, a tradeoff exists among the other two values:

- for a fixed offered load, the larger the imposed time constraint, the smaller the message loss.

- for a fixed message loss, the larger the offered traffic load, the larger the time constraint needed to realize this fixed loss.

- for a fixed time constraint, the larger the offered traffic load, the larger the message loss.

1.2. The Multiple Access Problem

Note that we have not yet specified any of the characteristics of the communication subsystem in figure 1-1. Communication networks [Schwartz 77, Tanenbaum 81] divide broadly into two categories: long-haul networks, such as ARPAnet [McQuillan and Walden 77], consisting of a multitude of communication links typically spread over a large geographic area and networks consisting of a single shared communication channel, such as bus and ring local area networks [Clark et al. 78]. In this thesis we will be concerned primarily with networks in this latter category.

Thus, let us consider a situation in which geographically distributed stations wish to communicate over a single shared communication channel. This channel provides the only means of communication among the stations and its properties are such that only a single message can be successfully transmitted over it at any one time. If two or more messages are simultaneously transmitted on the channel, then these messages interfere with each other and none of them will be correctly received by the station(s) for which they were destined. Such an environment is known as a *multiple access environment.* Since all stations can monitor the single communication channel, a message sent by one station can be detected or "heard" by all the other stations; for this reason the multiple access environment is also known as one type of *broadcast environment.*

There are numerous examples of multiple access environments. An everyday example is a group of conversants and the air between them. The air provides the single physical medium through which the people must communicate. As everyone knows, if two or more people talk at once, the usual result is that no one understands what anyone has said. (Actually, the human hearing system can often filter out all but one of the simultaneous conversations, so the analogy here is not exact.) A satellite channel and geographically distributed earth stations (e.g., the ALOHA system [Abramson 73]) also constitute a multiple access environment. In a satellite network, the earth stations transmit messages up to a satellite transponder which then relays the messages down to the earth stations. Simultaneous transmissions by the earth stations or the satellite

transponder result in message interference and the reception of unintelligible messages at the earth stations. Another type of multiple access environment is a ground packet radio network [Kahn et al. 78] in which (possibly mobile) distributed stations communicate over a single radio channel. Radio waves propagate through the media between the stations and interfering radio waves (i.e., simultaneous transmissions by two or more stations) again result in unintelligible message reception at the destination stations. Perhaps the most frequently cited example of multiple access environments are local area networks [Clark et al. 78] such as Ethernet [Metcalfe and Boggs 76], in which distributed stations share a single coaxial cable or optical fiber as the sole communication medium.

Since only a single station can successfully transmit a message at any given time, the distributed stations must somehow coordinate their access to the channel in order to share the channel among themselves. A distributed algorithm by which the stations share the channel is known as a *multiple access protocol.* In the network in figure 1-1, the multiple access protocol is the network access mechanism (C) and the network subsystem itself consists of a single broadcast communication channel.

Several issues complicate the problem of distributed channel sharing. First, since the stations are distributed, they have only local information. That is, a station knows whether or not it has a message to send and whether or not it will attempt to do so, but has no such information about the other stations in the network. Thus, stations must either explicitly or implicitly communicate information to each other if they wish to coordinate channel sharing. However, since there is only a single communication channel, coordination among the users about sharing the channel necessarily requires use of the channel itself! Thus, there is a circular or recursive aspect to the problem. Secondly, since the stations are distributed, they can never instantaneously know the present status of other stations in the environment; information about other stations is always at least as old as the message propagation delay between stations.

When the performance of a multiple access protocol is considered, the metric of

primary concern has traditionally been average time delay, i.e., the average amount of time between the generation of a message at a sending station and its successful reception at a destination station. This delay is typically characterized in terms of the average delay/throughput tradeoff which reflects the effect of increasing message generation rates on the average message delay. Note that this performance metric is quite different from that used for time-constrained applications; this is only one of the many ways in which time-constrained applications differ from the more traditional data communication applications.

1.3. Problem Statement and Motivation for This Dissertation

As we will see in chapter 2, a significant amount of research effort has already been devoted to the design and analysis of multiple access communication protocols [Tobagi 80, Kurose et al. 85]. However, these efforts have focused on protocols which support traditional, non-time-constrained communication applications. These applications differ significantly from time-constrained applications in several respects; these differences are summarized in figure 1-2.

As previously discussed, the primary performance metric for time-constrained applications is the percentage of messages received at destination stations within a specified amount of time after their generation at a sending station; for non-time-constrained applications, the primary performance metric is average delay. In time-constrained applications, a certain amount of message loss is tolerable; in traditional communication applications, 100% reliability is required. Finally, as previously discussed, these two types of applications also differ in terms of their performance tradeoffs. Time-constrained applications have a three-way tradeoff among the imposed time-constraint, the message loss and the offered load. For non-time-constrained applications, however, there is simply a two-way tradeoff between the average delay and the offered load.

The different performance metrics, reliability requirements and performance tradeoffs summarized in figure 1-2 suggest that *multiple access protocols previously developed for traditional data communication applications may not be well-suited for time-constrained applications.* This observation, together

	non-time-constrained applications	time-constrained applications
Primary Performance Metric	Average Delay	% messages with delays less than specified time constraint
Reliability	100%	some loss tolerable
Performance Tradeoffs	offered load versus Average delay	offered load versus message loss versus time constraint

Figure 1-2: Time-constrained communication versus non-time-constrained communication

with the growing interest [Nutt and Bayer 82, Maxemchuk 82, Pokress 84] in supporting time-constrained applications in a multiple access environment provide the primary motivation for the research presented in this dissertation.

Thus, this thesis focuses on the design, analysis and optimization methods of access protocols for supporting time-constrained communication applications in multiple access networks. In addition, however, this specific problem area also serves as a vehicle for developing ideas and techniques which extend beyond this particular problem domain. For example, many of the analytic models developed and validated in the course of this research are also relevant to real-time problems occurring in quite different contexts. Also, the work presented in chapter 6 on the distributed optimization of time-constrained protocols (which itself draws inspiration from the distant field of mathematical economics) has important implications in many other problem areas, such as distributed resource sharing and coordination, in computer communication networks.

1.4. Contributions of this Research and Organization of Subsequent Chapters

In this section we summarize the main results of this dissertation and present an outline of the remaining chapters.

The principal contributions of this thesis fall into two categories:

1. *the development and analysis of a novel class of protocols for supporting time-constrained communication applications in a multiple access environment.* There are several contributions falling under this category:

   a. identification of the critical role of an access protocol as a distributed message transmission scheduling mechanism and the importance of this role in determining the time-constrained performance of a protocol. This scheduling role has been almost completely overlooked in past work on multiaccess protocols. Given the importance of this role in supporting time-constrained communication applications, we have developed a class of multiple access protocols, based on the use of time windows, which can provide any of a family of message transmission scheduling disciplines based on message generation times. Novel exact and approximate performance models are developed for the cases in which the protocol provides FCFS, LCFS and Random scheduling.

   b. derivation of the the optimal elements of the windowing policy of the time window protocol using a semi-Markov decision model. The performance model we develop to examine the time-constrained behavior of the optimal windowing policy is based on a queueing system with impatient customers. This work augments existing analytic modeling techniques by providing a simple, analytically tractable model for determining customer loss in M/G/1 queues in which customers are denied service when their waiting time exceeds a given time bound.

   c. extension of the time window protocol to the multi-class case in which network stations support the transmission of both time-constrained and non-time-constrained classes of traffic.

2. *development of a systematic and formal approach towards distributed optimization via a fictitious resource sharing paradigm and a decentralized microeconomic solution of resource sharing problems.* Much of the past work in the design and optimization of distributed systems has been *ad hoc* in the sense that individual solutions are provided for individual problems without the benefit of a formal or systematic design methodology. This work represents an initial attempt towards developing

such a methodology. Our work draws on models and methods from microeconomic theory to provide blueprints for a systematic approach towards engineering decentralized optimization algorithms and resource sharing mechanisms in distributed systems. Our "microeconomic" approach has been successfully applied to the problem of computing the optimum transmission probabilities for both the time window protocol and the Slotted Aloha protocol. Interestingly, several network mechanisms, such as flow control and priorities have been found to emerge naturally from this approach.

In the following chapter we survey much of the past work on both time-constrained communication and multiple access protocols. A taxonomy for multiple access protocols is first developed in order to characterize common approaches towards resolving the multiple access problem and to provide a framework in which these protocols can be compared and contrasted. Different proposed protocols are then described and discussed and aspects of their performance are examined. The notion of time-constrained communication is then examined in more detail and general approaches towards achieving time-constrained communication in a multiple access environment are then considered.

Since message loss results from message delays exceeding a given bound, the *distribution* of message delay (as opposed to the average message delay, as in standard communication) critically determines the time-constrained performance of a protocol. In chapter 3, the importance of the *network-wide* ordering imposed on message transmissions by the sending stations' access protocol (i.e., the network-wide *scheduling function* performed by the stations) is investigated. It is shown that in addition to the protocol's traditional role as an arbiter of channel sharing, it also serves as a *distributed scheduling mechanism* by imposing an implicit or explicit network-wide transmission order on the messages distributed among the stations in the network. A random access protocol, based on a generalization of time window protocols [Gallager 78, Towsley and Venkatesh 82] is proposed which provides a family of distributed scheduling disciplines for message transmission based on their generation times. Both simulation and novel analytic models are then developed to study the effect of the imposed scheduling discipline on the message waiting time distribution.

Protocol performance is examined for the cases in which the protocol transmits all the messages throughout the network on a FCFS, LCFS and Random basis. None of the examined protocols were found to be uniformly optimal for all values of the network operating parameters. However, the performance results do forcefully demonstrate that a protocol's scheduling function does critically determine its time-constrained performance. A close agreement is found between the analytic and simulation results, thus validating the use of several key independence assumptions introduced in the analytic models.

Significant performance improvements can be realized when sending stations assume a more active role in message transmission. Specifically, since some message loss is tolerable, a sending station can itself decide to explicitly discard a given message (i.e., to not transmit the message). The advantages of losing messages at the sending stations (as opposed to the receiving stations) are twofold. First, resources need never be wasted on transmitting messages which would be lost with certainty at the receiving station. Secondly, in heavy traffic situations, large message delays (and correspondingly large loss) resulting from a temporary overload need not be propagated into the future. In chapter 4, a policy is formulated for the windowing process of the time window protocol when the additional capability of explicitly discarding messages is introduced. The goal of this policy is to maximize the percentage of messages received at destination stations with waiting times less than a given bound. A semi-Markov decision model [Howard 71] is developed for the operation of the sending stations and it is proven that certain temporally local optimal decisions (with respect to message loss) also characterize optimal long term (infinite horizon) behavior. Three of the four optimal elements of the windowing policy are determined within this decision model and are shown to be both intuitive and simple. A heuristic is presented for the final policy element.

Although the semi-Markov decision model can also be used to obtain analytic performance results, the procedure is too computationally expensive to be of practical use. Thus, an alternate performance model based on a general queueing system with impatient customers is developed. For cases in which customer loss is the important performance metric, our model is considerably simpler than

previously developed related queueing models. This model is then used to examine the time-constrained performance of the protocol under the optimal elements of the windowing policy; simulation results are also presented to corroborate the analytic results. As expected, the results demonstrate that significant performance improvements can be realized over the cases in which the sending stations do not use the optimal elements of the windowing policy. Finally, we consider the extension of the ideas developed in this chapter to the case of a non-homogeneous network environment in which the time constraints and relative priorities of each of the stations may be different.

Due to recent interest in the development of integrated services digital networks (ISDN's) [Pokress 84], the development of protocols for the integrated transmission of both time-constrained and non-time-constrained classes of traffic is of considerable interest. In chapter 5, we present the extension of the time window protocol to support both these classes of traffic in a multiple access network. The performance goal of the time-constrained traffic is once again the minimization of message loss; the performance goal of the non-time-constrained traffic is the minimization of average message delay. A windowing policy implementing preemptive-resume priority is presented and the tradeoff between time-constrained message loss and non-time-constrained average message delay is then examined. The mechanism in the multi-class time window protocol for selecting an operating point along this tradeoff curve is identified and an analytic model is then developed to quantitatively study this tradeoff. The performance results indicate that in all but the high traffic cases, a small increase/decrease in the average delay of the non-time-constrained traffic is accompanied by a large decrease/increase in the time-constrained message loss.

In chapter 6, we consider the problem of computing the optimal window sizes for the time-window protocol in a heterogeneous network environment. The relative merits of centralized versus decentralized optimization are first examined. We then demonstrate that the problem of determining the optimal window sizes can be transformed into a *fictitious* resource allocation problem, and that the optimal solution of this fictitious resource allocation problem immediately yields an optimal solution to the window sizing problem. We then examine how

models and methods developed by mathematical economists for decentralized resource allocation problems in perfectly competitive exchange economies [Arrow and Hahn 71] can be used as *blueprints* for engineering decentralized resource allocation algorithms in computer networks; these algorithms, in turn can be used to solve not only the fictitious resource allocation problem posed in this chapter, but real resource allocation problems occurring in networks, as well. The important concepts (including utility, demand, prices, Pareto optimality and optimality results) from the microeconomic model of perfect competition are first presented; earlier work [Yemini and Kleinrock 79, Yemini 81] suggesting a connection between the resource allocation problems in economies and networks is then examined.

We then show that when stations act as "selfish", utility maximizing entities, a simple iterative resource pricing mechanism can often be introduced to permit a decentralized computation of the *optimal* distribution of the fictitious resources and hence be used to determine the optimal window sizes in a heterogeneous network environment. We then experimentally study the behavior of this decentralized optimization algorithm in a small (4 station) simulated multiple access network using the time window protocol. In the perfectly homogeneous case, the optimization results are found to match the centralized optimization results. For the case of heterogeneous stations with differing message generation rates and relative priorities, the results demonstrate how this approach can be used to impose any of a continuum of priority levels on the stations in the network. Another important network mechanism, flow control, is also found to emerge naturally from such an approach.

The application of this decentralized approach towards the optimization of other multiple access protocols is also examined. In the case of Slotted Aloha, the optimality results obtained via the fictitious resource allocation model are found to exactly coincide with those reported in [Abramson 73]. Finally, the feasibility of applying this decentralized "microeconomic" approach to other problems in distributed computation and communication is then considered.

Chapter 7 summarizes and concludes this thesis and discusses topics for related future research.

Chapter 2
A Survey of Related Work

In this chapter, we survey relevant past work in the areas of multiple access protocols and time-constrained communication. We begin by developing a taxonomy for multiple access protocols in order to provide a framework in which the various different approaches can be compared and contrasted. Using this taxonomy as an outline, we then briefly describe and discuss numerous proposed multiple access protocols; due to space limitations, the descriptions are necessarily brief and some refinements to the basic mechanisms have been omitted. Performance-related issues for non-time-constrained applications are then examined and discussed. Finally, we survey recent work on supporting time-constrained communication in a multiple access environment.

2.1. A Taxonomy For Multiple Access Protocols

In the past decade, numerous protocols have been proposed for resolving the multiple access problem [Tobagi 80, Kurose et al. 85]. These protocols may be divided broadly into two classes: *controlled-access* and *contention-based* protocols. These two classes and their further subdivisions are indicated in the multiple access protocol taxonomy shown in figure 2-1.

Controlled-access protocols are characterized by collision-free access to the channel. That is, the transmissions of the distributed stations are coordinated in such a way that two or more stations never attempt to transmit messages simultaneously. This coordination is typically achieved by imposing an ordering on the allocation of channel access (transmission) rights to the stations. Controlled-access protocols can be further characterized by whether the allocation of channel transmission rights varies in response to the changing transmission

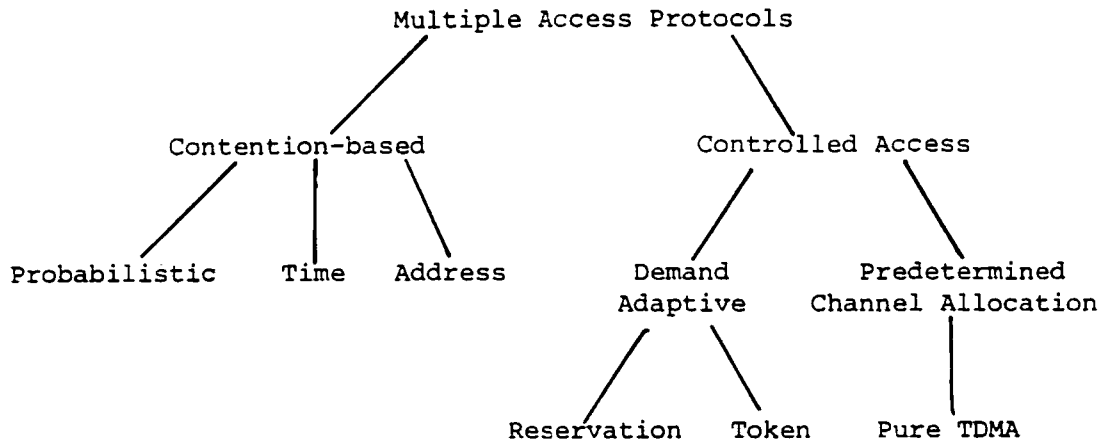


Figure 2-1: A taxonomy of multiple access protocols

requirements of the stations. *Predetermined channel allocation* (PCA) protocols allocate the channel to the stations in a static manner and thus are not responsive to changing transmission requirements of stations. *Demand-adaptive* protocols attempt to allocate the channel in a manner more consistent with the immediate requirements or demands of the stations. In subsequent sections, it will be shown that both PCA protocols and demand-adaptive protocols are inefficient if there are a large number of stations with *bursty* message transmissions. When message transmissions are bursty, each station is usually idle (i.e., has nothing to transmit) but occasionally does have a large amount of data to send. The inefficiency in controlled-access protocols results from granting transmission rights (in PCA protocols) or the opportunity to claim transmission rights (in demand-adaptive protocols), in order, to *all* stations, regardless of whether or not they have messages to transmit. Thus, idle stations are offered (unneeded) transmission rights while stations with messages must wait their turn before transmitting. Note that in demand-adaptive protocols, even if a station is offered, but immediately refuses, transmission rights (e.g., if it does not have a message to send), a certain amount time will be required to pass transmission rights to another station.

The second broad class of multiple access protocols, known as contention-based protocols, attempts to overcome these inefficiencies by simultaneously offering transmission rights to a group of stations in the hope that exactly one of these stations has a message to send. Contention-based protocols thus operate by partitioning the stations in the network into a set of *enabled* stations (those with transmission rights) and a set of *disabled* stations (those without transmission rights); station addresses, the time at which messages are generated at sending stations, and explicit probabilistic mechanisms have been proposed as criteria for determining whether a station is enabled or disabled. If none of the enabled stations are ready (i.e., have a message to send), the channel remains unused and a new partitioning of the stations is eventually determined. If exactly one enabled station is ready, that station transmits its message. Typically, at the end of this transmission, a new partitioning is determined. Finally, if two or more ready stations are in the enabled set, they transmit colliding messages. If stations can detect and abort collided transmissions, the enabled set is often further divided in an attempt to isolate a single ready station in the enabled set. If collided transmissions cannot be detected and aborted, a new enabled set is typically determined at the end of the colliding transmissions.

Before beginning a discussion of the various multiple access protocols, let us conclude this section by suggesting an alternate (and potentially valuable) classification of these protocols. There is a spectrum of information ranging from no information to perfect information about the state of all stations in the network. All protocols operate somewhere along this spectrum and each protocol operates using different information. Thus, multiple access protocols can potentially be characterized in terms of the information they use. But exactly what information is used? Three types can be readily identified. First, there is "hard-wired" information (e.g., a predetermined polling order) known to each station when it begins operation. There is also global information that is obtained from the channel. Finally, there is local information known only to a single station (e.g., the generation time of a message at a particular sending station). Local information can be transformed into global information when it is transmitted over the channel. Note that the use of local information may result in imperfect coordination among the stations. For example, in contention-based

protocols, if two stations use local information (e.g., a message generation time or the value of a local random number) to determine whether or not to access the channel, they may transmit colliding messages. The perfect coordination (i.e., absence of collisions) in controlled-access protocols results from the use of hard-wired information (e.g., a predetermined transmission order) known to all stations, as well as global information. Note, however, that there is a price paid for this global information since idle channel time is used to indicate that a station has no message to send.

Determining the nature and the extent of information used by a protocol is surely a difficult task, but potentially a valuable one. An understanding of exactly *what* information is used could potentially lead to an understanding of its *value*. Ideally, it would then be possible to determine what information is important in determining protocol performance for a given application and what additional information, if any, would be useful. Such an understanding could provide for a qualitative evaluation of the performance of protocols (e.g., ordering their performance for a given application) based on the information they possess without resorting to a quantitative (and potentially difficult) performance analysis.

The first theoretical treatment of the role of information in protocols and the fundamental limits imposed on protocols by the necessity of overhead information was reported in [Gallager 76]. In this work, lower bounds were derived on the amount of information (in the information-theoretic sense) which must be transmitted in a network to indicate the beginning, the end. and the destination of messages. Several researchers have recently developed theoretical upper bounds on the capacity (maximum number of messages successfully transmitted per message transmission time) of various classes of contention-based multiple access protocols. Pippenger [Pippenger 81] used an information theoretic approach to derive an upper bound of .774 for the capacity of synchronous contention-based protocols with a large number of stations and in the case that the time between message generations (network-wide) is exponentially distributed. Molle [Molle 82] and Cruz [Cruz and Hajek 82] later tightened this upper bound to .6731 and .6126, respectively using a "helpful genie" to provide additional

information to the stations at no additional cost. Since these capacity results are realizable only with the aid of a genie, they thus represent an upper bound on the capacity for real, non-genie-aided protocols.

2.2. Controlled Predetermined Channel Allocation Protocols

Predetermined channel allocation (PCA) protocols provide collision-free access to the communication channel and determine the channel transmission rights of stations in a static, predetermined manner. The most prevalent PCA protocols are pure time division multiple access (TDMA) protocols.

Pure TDMA protocols provide collision-free multiple access broadcast communication by permitting each station to periodically utilize the full transmission capacity (or *bandwidth*) of the single communication channel for some fixed amount of time. In this fashion, the channel is shared in time among the stations. Time is divided into fixed length intervals or frames; each frame is further subdivided into slots as shown in figure 2-2. In the simplest version of TDMA, which we refer to as pure TDMA, the number of slots per frame is the same as the number of stations in the network and each station is granted use of the channel for the duration of one time slot per frame.

It is important to note that pure TDMA protocols are potentially very inefficient. One inefficiency arises when the number of stations in the network changes in time, as in ground radio environments with mobile stations. Since pure TDMA performs *a priori* slot assignment, slots are assigned to stations independent of whether or not they are currently in the network. When a station is not in the network, its slot remains idle. A second inefficiency arises even when the number of stations remains constant. When a station has nothing to send, its time slot is unused even though other stations could potentially utilize this time slot. This problem can be particularly acute when message generation is bursty and the number of stations is very large. A variation of pure TDMA, known as slot-switched or variable-frame TDMA, has been developed to overcome these inefficiencies; this scheme will be discussed in section 2.6.1.

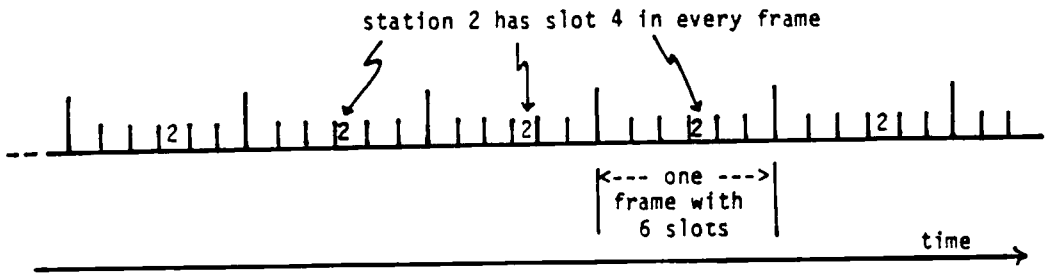


Figure 2-2: Pure time division multiple access

2.3. Controlled Demand-Adaptive Protocols

Controlled demand-adaptive protocols were developed to overcome the previously noted inefficiencies of PCA protocols. They are based on the use of *hub polling* techniques [Schwartz 77] previously developed for communication networks with a centralized control. In demand-adaptive protocols, as in PCA protocols, channel access (transmission) rights are offered to stations according to some access order. This order may be determined *a priori* or can be dynamically determined by the stations. The inefficiencies of PCA protocols are overcome by requiring stations to immediately forfeit their access rights if they have no messages to send when they receive access rights. At the end of a station's transmission, access rights are passed to the next station in the order. Two mechanisms have been developed to maintain the access order: reservation schemes and token passing schemes.

The most straightforward reservation scheme is the basic bit-mapped protocol

[Kleinrock and Scholl 80]. Let us assume there are N stations each having a unique address between 1 and N. The bit-mapped reservation protocol consists of alternating periods of reservation posting and message transmission, as shown in figure 2-3 for the case of 6 stations.

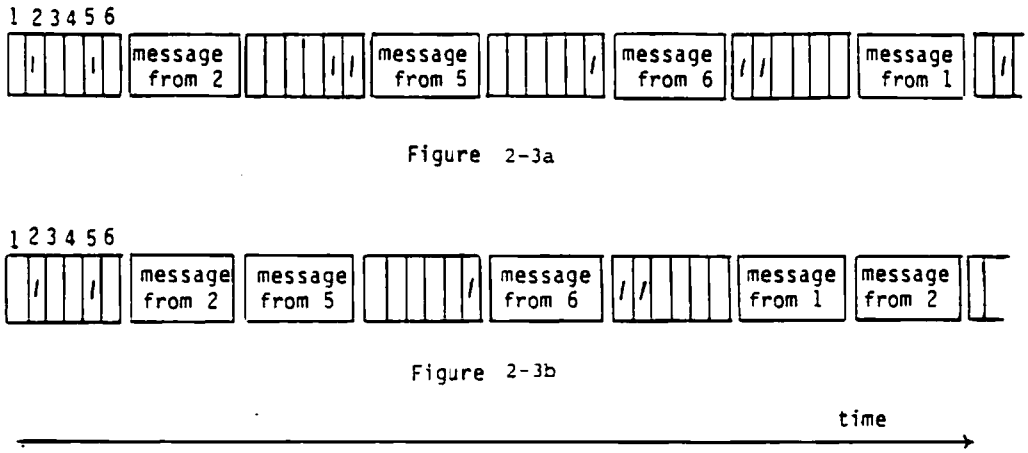


Figure 2-3: Two reservation protocols

Each reservation slot is of length τ, the end-to-end propagation delay of the channel. During the reservation period, a station transmits a burst of noise (shown as a 1 in figure 2-3) during its reservation slot to indicate to the other stations that it is ready to transmit a message during the following message transmission period. All stations monitor the reservation process and thus each station knows the identities of the other stations that are ready to transmit.

In the protocol shown in figure 2-3a, a single message transmission follows the reservation period and then another reservation/transmission cycle begins. The one station to be granted transmission rights is selected from those stations having posted reservations according to some priority rule known to all stations. This protocol was proposed and analyzed in [Kleinrock and Scholl 80] and several

different priority rules were examined. A slight modification of this protocol is shown in figure 2-3b. In this protocol, *all* stations which post reservations are permitted to transmit their messages (once again, according to some known priority rule) before another reservation/transmission cycle begins. Note that while this scheme is more efficient (less channel time is used for reservations per message transmission), the first protocol permits higher priority stations to exercise their priority more often.

The second major class of controlled demand-adaptive protocols, called token passing protocols, circulate a real or imaginary token message among the stations in such a manner that only a single station possesses the token message at any one time. By definition, the station possessing the token message possesses channel access rights.

In explicit token passing schemes, a token message is circulated among the stations in the network and a station cannot transmit until it first receives the token message [Farmer and Newhall 69, Farber at al. 73, Clark et al. 78, Bux et al. 81]. For example, in token rings [Farmer and Newhall 69, Bux et al. 81], the stations are connected to a unidirectional bus, the ends of which have been joined to form a ring. In the case that the stations have nothing to send, a token message consisting of a single zero-valued token bit within a message frame header circulates around the ring. When a station has a message to send, it waits until it detects a message frame header with a zero-valued token bit passing its channel connection, sets the token bit to 1 and appends its message to the message frame header. When this message has made one complete revolution around the ring, the station then sets this token bit back to zero and the token message continues to circulate around the ring.

BRAM (Broadcast Recognition Access Method) [Chlamtac et al. 79] and MSAP (Minislotted Alternating Priorities) [Kleinrock and Scholl 80] can be viewed as using an imaginary token to implement a form of distributed channel sharing. The order in which the stations are granted transmission rights (receive the token) is determined by the numerical order of their station addresses. Station 1 initially has transmission rights. If station 1 has a message to send, it does so;

otherwise it remains silent. The presence or absence of a transmission after time τ indicates to the other stations whether or not station 1 intends to send a message. If station 1 does not begin sending a message, transmission rights are implicitly passed to station 2, which repeats the same procedure as station 1 while the other stations monitor its activity. If station 1 decides to send a message, transmission rights are passed to station 2 as soon as station 1 completes its transmission. This process is shown in figure 2-4 below.

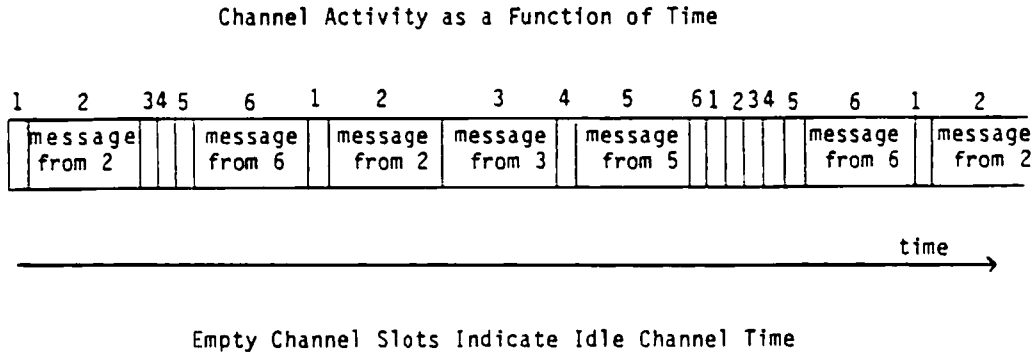


Figure 2-4: MSAP/BRAM: an imaginary token passing scheme

In MSAP/BRAM, an amount of time τ is required to establish the absence of a message transmission. In EXPRESS-NET [Fratta et al. 81] and in FASNET [Limb and Flores 82] the use of a folded *unidirectional* channel and a similar access protocol removes the necessity of waiting for time τ. These schemes work as follows. Each station maintains an inbound connection and an outbound connection to the folded channel as shown in figure 2-5.

The underlying idea is that the station at the head of the channel (i.e., station 1 in figure 2-5) begins a "train" of messages which propagates down the unidirectional channel. When a station detects the end or "caboose" of the train passing its *outbound* connection, the station appends its message (if any) onto the message train. A station reads messages from the train as the train passes its

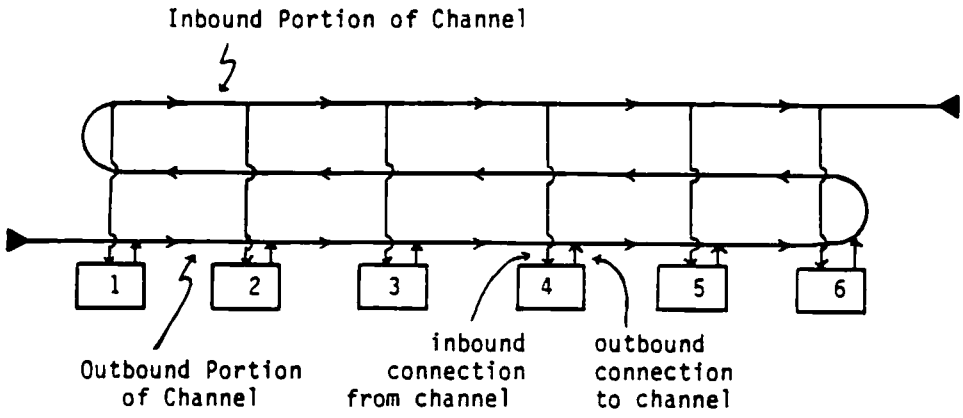


Figure 2-5: EXPRESS-NET channel connections

inbound connection. Note that the spacing between messages on the train is only the amount of time required for a station to detect the end of a passing train and begin transmission of its message. Thus, unlike most other demand-adaptive protocols, the spacing between message transmissions is independent of, and typically less than, τ.

2.4. Contention-Based Protocols

The second major class of multiple access protocols are known as contention-based or random access protocols and are characterized by the possibility that channel *contention* may result from two or more stations attempting to transmit messages simultaneously. We have previously characterized the operation of contention-based protocols as a partitioning process in which the set of all stations in the network is divided into an enabled set of stations (stations with transmission rights) and a disabled set of stations. Three classes of mechanisms have been proposed and developed to perform this partitioning process: probabilistic mechanisms, time-based mechanisms and address-based mechanisms.

2.4.1. Probabilistic Partitioning

The first contention-based solution to the multiple access problem is also the simplest. A station simply transmits a message as soon as it is generated by the application executing at the station. This solution, known as pure ALOHA, was first developed by Abramson [Abramson 70] and involves no coordination among the distributed stations. If two stations happen to transmit messages at the same time, their messages interfere or "collide" and thus require later retransmission. Note that a station must schedule a retransmission to occur after a *random* amount of time. Otherwise, if two or more stations were to interfere and always schedule a retransmission to occur after the same amount of time, these stations would interfere forever. Thus, the probabilistic element in the partitioning process permits interfering stations eventually to become ready at different times in the future.

In pure ALOHA, messages can interfere even if only the first bit of a message beginning transmission overlaps the very last bit of a message ending transmission. Thus, if all messages require t' time units to be transmitted, a station beginning transmission of a message at time t_0 is *vulnerable* to message collisions due to transmissions from other stations that began after t_0-t' or before t_0+t'. The total amount of time the message is vulnerable is thus $2t'$. This situation is shown in figure 2-6 [Tanenbaum 81].

A modification of pure ALOHA, known as slotted ALOHA, was subsequently introduced. In slotted ALOHA, time is divided into intervals or *slots* of the same duration as a single message transmission time. A station can begin sending a message only at the beginning of one of these time slots. A message which is generated at a sending station during a time slot cannot be sent until the beginning of the following time slot. Thus, only those messages which were generated in the previous time slot can interfere with each other. The effect of this synchronization then is to cut the vulnerability period in half, from $2t'$ in pure ALOHA to t' in slotted ALOHA. Note that messages either collide completely for the duration of their transmission or do not collide at all.

In the earliest versions of pure ALOHA and slotted ALOHA, information about

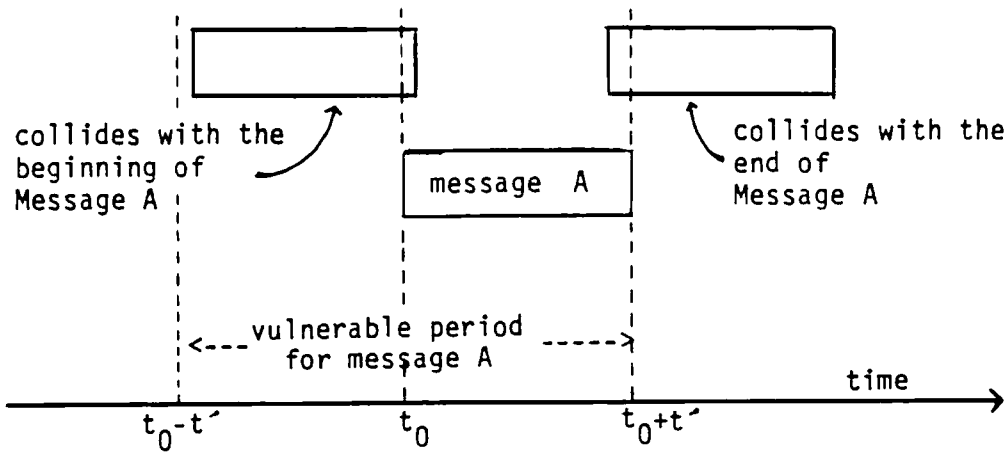


Figure 2-6: Message vulnerability in ALOHA

the state of the channel (e.g., whether or not a message is currently being transmitted) was not used in determining the enabled set of stations. (It was later found [Fayolle et al. 77] that such information was necessary to avoid stability problems.) The class of protocols which have the ability to sense or "listen" to the channel and use this information in determining the enabled set are known as *carrier sense multiple access* (CSMA) protocols. Two of the earliest CSMA protocols are known as nonpersistent and p-persistent CSMA [Kleinrock and Tobagi 75]. In nonpersistent CSMA, when a message is generated at a station, the station senses the channel in order to determine if another station is currently sending a message on the channel. If the channel is sensed empty, its message is sent immediately. If the channel is sensed busy, its message transmission is rescheduled for a later time according to some retransmission time distribution. A rescheduled message can be considered to be "regenerated" at this later time. Both slotted and unslotted versions of nonpersistent CSMA have been investigated.

In p-persistent CSMA, when a message is generated, the station begins sensing the channel. When the channel is sensed to be empty, the station transmits its message with probability p. With probability 1-p, the station waits some fixed amount of time and then senses the channel again. If the channel is again

detected to be empty at this new point in time, the above procedure is repeated. However, if the channel is sensed busy (indicating that another station has begun transmission), the transmission of the message is rescheduled for a later time as in nonpersistent CSMA. Ethernet [Metcalfe and Boggs 76] is a well-known example of a 1-persistent CSMA protocol (p-persistent CSMA with p=1). One proposed probabilistic retransmission mechanism for Ethernet is known as *binary backoff*. In this scheme, stations are assumed to have collision detect (CD) and transmission abort capabilities in addition to the previously mentioned carrier sense (CSMA) capabilities. Once the channel is sensed empty, any station with a message to transmit attempts to do so. If a collision occurs, all stations terminate their transmissions and randomly reschedule their transmissions over some period of time. The time period over which a given station reschedules its message transmission doubles each time the message experiences a collision.

Kleinrock and Yemini [Kleinrock and Yemini 78] have described a slotted multiple access protocol known as the Urn protocol, which uses a probabilistic mechanism in a much different fashion. Suppose there are N stations and it is further known that some n of these stations are ready (i.e., have a message to transmit). The partitioning process which determines the set of enabled stations corresponds to selecting some k stations from an imaginary urn containing the N stations. If the k enabled stations contain exactly one ready station, the partitioning has been successful; if the k stations contain either none or more than one ready station, the partitioning process must be repeated. It can be shown that the value of k which maximizes the probability that *exactly* one of the k stations is ready is given by the integer part of N/n. Under the heavy traffic assumption that every station always has a message to send, k always equals one. Thus the Urn scheme operates in a random TDMA-like fashion in the heavy traffic situation. That is, channel transmission rights are randomly passed among the stations and exactly one station has channel transmission rights at any given time.

2.4.2. Address Partitioning

Hayes and Grami [Hayes 78, Grami et al. 82] and Capetanakis [Capetanakis 79] have proposed contention-based protocols which use station addresses to determine the enabled stations. In the simplest case, address partitioning works as follows. We can think of each of the N stations as a leaf in a binary tree as shown in figure 2-7.

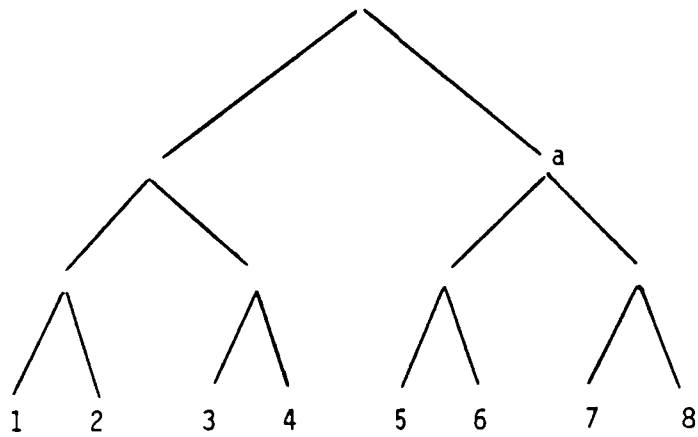


Figure 2-7: Stations as leaves on a binary tree

Time is slotted and after a successful transmission on the channel, all stations with a message to send initially attempt to do so. If a collision occurs, the set of stations is partitioned such that only stations in one of the halves of the binary tree are enabled (e.g., the subtree rooted at a in figure 2-7) in the subsequent time slot. If further collisions occur, the enabled set is continually halved until it eventually has only one ready station. If at some point the tree is halved and there are no ready stations in the enabled half of the tree, an empty slot occurs on the channel and the other half of the tree becomes enabled.

Several improvements exist on this basic approach. If the number of ready stations can be estimated, the tree can be partitioned in such a way as to maximize the probability that the enabled set contains exactly one ready station. In the heavy traffic case, in which all stations have a message to send, the

initially enabled set would be chosen to contain only a single station. In the light traffic case, the entire tree would initially be enabled. Note that this approach has much the same flavor as the Urn protocol. It should also be noted that if trees are deterministically split by the partitioning process, certain ready stations will always transmit their messages before other ready stations. This *de facto* priority can be avoided by introducing randomization into the tree splitting process.

2.4.3. Time Partitioning

The final group of contention-based protocols to be discussed uses the generation times of messages at a station to determine the set of enabled stations. Gallager [Gallager 78] first proposed a a protocol in which the enabled stations are those which have messages to send which were generated during some chosen interval or *window* of time in the past. Thus, when the stations must decide whether to transmit a message, they first select this time window and then transmit a message if, and only if, they have a message to send which was generated during the selected time window. Since the multiple access protocols we develop in chapters 3, 4 and 5 for supporting time-constrained communication applications are based on the use of time-windows, we will defer further discussion of this mechanism until chapter 3.

Molle [Molle 81] has developed a protocol similar to the time window protocol in which each station maintains two clocks: a normal clock and a virtual clock. The virtual clocks can run in either slotted or continuous time, at a possibly variable speed and all stations run their virtual clocks in the same manner. When a station's virtual clock time equals the generation time of a message at the station, that station stops its virtual clock and sends the message. If messages collide, their transmissions are rescheduled for a later time, as in the nonpersistent CSMA protocol. When other stations detect a transmission on the channel, they too stop their virtual clocks and restart them only when channel activity ceases. In this manner the virtual clock sweeps out time such that when it reads t_0, all messages which were generated before t_0 have either been transmitted or have been rescheduled for later transmission.

2.5. The Non-Time-Constrained Performance of Distributed Multiple Access Protocols

2.5.1. Issues in Evaluating Multiple Access Protocols

Traditionally, the "performance" of a multiple access protocol has been characterized by the maximum number of messages that it can deliver per unit time and by the non-time-constrained performance tradeoff of average delay versus throughput, as discussed in chapter 1. Although these performance measures are significantly different from those for time-constrained applications, we can nonetheless gain valuable insight into both the multiple access problem as well as the operation of the previously described protocols by examining these protocol performance measures. This will be done in sections 2.5.2 and 2.5.3. First, however, let us consider other performance issues (which have typically received less attention) that must also be considered in the evaluation of an access protocol.

The *stability* of the protocol, or its ability to operate in spite of varying traffic demands and short term overloading of the network is an important consideration. For example, several of the previously described contention-based protocols have been shown to exhibit *bistable* behavior. Under bistable behavior, the protocol initially operates at a point characterized by high throughput and low average message delay; statistical fluctuations in the message load, however, eventually force the protocol into a second operating point of low throughput and high average delay. Although these same statistical fluctuations will eventually again force the system back to its initial operating point, it has been observed [Kleinrock and Lam 75] that for at least the slotted ALOHA protocol, the average time spend in the state of degraded performance is typically much larger than the amount of time spent in the initial operating state. The bistability of ALOHA protocols was first noted in [Carleial and Hellman 75] and [Kleinrock and Lam 75]; in [Lam and Kleinrock 75] and [Fayolle et al. 77] several control policies were proposed and examined which successfully prevented the protocol from moving away for its initial operating point. In [Tobagi and

Kleinrock 77] and [Mittal and Venetsanopoulos 81], nonpersistent CSMA and the Urn protocols, respectively, were also found to theoretically exhibit such bistable behavior. For practical purposes, however, except in cases of a large number of stations, nonpersistent CSMA was found to exhibit essentially stable behavior since the expected time spent in the initial operating state was found to be quite large.

Another important issue in evaluating an access protocol is its *reliability* and its ability to operate in spite of station failures. A related issue is the *robustness* of the protocol, or its insensitivity to errors, channel noise and misinformation. The ability of the protocol to support different classes of traffic and different priority levels is also currently of great interest due to the possibility of developing integrated services data networks [Gitman et al. 77, Pokress 84] based around multiple access channels; this topic will be further investigated in section 4. Finally, the engineering aspects of protocol implementation [Saltzer and Clark 81], the complexity of the hardware and software and the ease of maintenance are also important considerations if a protocol is ever to be actually implemented.

A complete discussion of these issues is beyond the scope of this thesis; the references cited above provide a more detailed discussion of these issues. In the following two sections, the performance issues of protocol capacity and the time delay/throughput tradeoff will be examined.

2.5.2. The Capacity of Multiple Access Protocols

Since stations in a multiple access environment are geographically distributed, protocols almost always require the exchange of control information in order to achieve some form of coordination. This information can either be explicitly communicated (e.g., reservations) or implicitly communicated (e.g., by a known ordering or by channel activity or silence). Since some protocols require the use of the channel for explicit control information as well as for successful message transmission, the channel will not always be used for "useful" work, i.e., the successful transmission of messages. The fraction of channel time actually used

by successfully transmitted messages, as opposed to control information, is known as the *effective utilization* of the channel. The maximum value of the effective utilization over all possible traffic loads is known as the *capacity* of the protocol. Note that if control information is exchanged over the channel, the capacity of the protocol will be less than unity.

Several aspects of the multiple access environment and the behavior of the protocol itself influence protocol capacity. Capacity is perhaps most greatly affected by the value of the *normalized* end-to-end channel propagation delay of the channel, a, defined as:

$$a = \tau/M$$

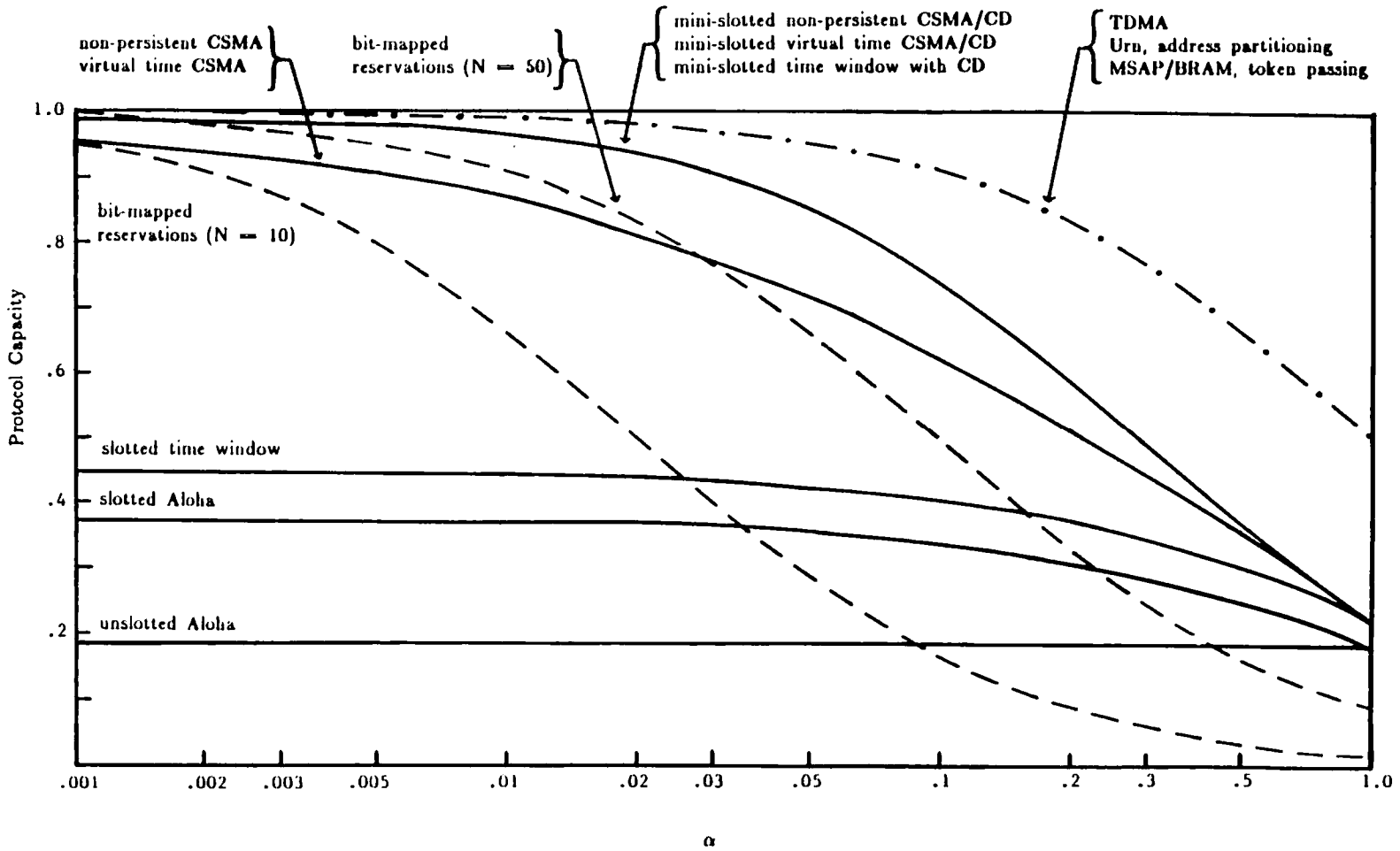
where τ is the end-to-end propagation delay of the channel and M is the average message length (in units of time). The importance of τ is clear: it represents the maximum amount of time into the past for which a station has no information about other stations. In collision-based protocols, a station that senses an empty channel and decides to transmit a message may still interfere with another station that decided to begin transmission within the past τ. Both stations, having sensed an empty channel, will nonetheless transmit colliding messages, resulting in the "non-productive" use of the channel. In reservation schemes, τ represents the minimum reservation slot length needed to insure that all stations know the content of one reservation slot before the next reservation slot begins. The fact that protocol capacity also depends on M reflects a scaling effect. For example, if the reservation or contention period preceding a successful transmission is $n\tau$ (for some value, n) and the message is of length M, the effective channel utilization is the same as if the length of the contention/reservation period was $kn\tau$ and the message length was kM. The equivalence of these two situations is reflected by their identical values of a.

The ability of stations to detect message collisions and subsequently abort transmission of their messages also influences protocol capacity. Collision detection assures that the channel is not wasted by the transmission of the entire length of a colliding message (which will require retransmission in any case). Note that the maximum amount of time needed for a station to determine that no other stations have interfered with its transmission is 2τ, the maximum round trip end-to-end propagation delay.

A third aspect of protocol operation significantly affecting capacity is whether or not a protocol operates in slotted time, i.e., whether or not the protocol operates synchronously. As previously discussed, time can either be "fully-slotted", in which case the length of a time slot is equal to the message transmission time (e.g., as in slotted ALOHA), or can be "mini-slotted", in which case the slot length equals the end-to-end propagation delay of the channel (e.g., as in MSAP). If time is divided into full length slots, stations transmit only when an empty channel is detected and begin their transmissions only at the beginning of a slot, so that messages either interfere completely or do not interfere at all. Fully-slotted time thus helps minimize the waste of channel time for the transmission of colliding messages. Mini-slotted synchronous operation is often associated with contention-based protocols having collision detect and transmission abort capabilities since this permits the stations to detect a collision within time τ and immediately determine a new enabled set of stations at the beginning of the next mini-slot. Even in the absence of collision detect capabilities, if stations detect an idle channel for the duration of a entire mini-slot, this indicates that no currently enabled station wishes to transmit a message and a new enabled set of stations can thus immediately be determined at the beginning of the subsequent mini-slot.

In figure 2-8 we show a comparison of the capacities of some of the previously described protocols as a function of α. These results are from various sources but have been derived under the following common assumptions. The first assumption is that messages (newly generated messages as well as the "re-generated" messages in the CSMA protocols) are generated on a network-wide basis according to a Poisson process or equivalently, that the network-wide inter-generation time of messages is exponentially distributed. Secondly, a message is assumed to occupy the channel for the full length of its transmission time *plus* τ, the time required for the message to propagate across the full length of the channel. The results for contention-based protocols all assume a large (essentially infinite) number of stations. Finally, the capacity results in figure 2-8 assume that a station can transmit only a single message before having to again compete with the other stations for channel access rights.

Figure 2-8: Protocol capacity as a function of α



The protocol with generally the lowest capacity is pure ALOHA. The complete lack of coordination of stations in ALOHA results in a maximum of only 18% of the full communication capacity of the channel ever being utilized for useful communication! Slotted ALOHA provides a twofold increase in capacity over pure ALOHA. This is due to the previously noted twofold decrease in a message's vulnerable period under slotted ALOHA. It should be noted that the slotted ALOHA capacity results shown in figure 2-8 are not those most frequently cited in the literature, (see, for example [Abramson 77]), in which the end-to-end propagation delay of the channel was not considered. In that analysis, once a station terminated a message transmission, the message was no longer considered to occupy the channel. The more realistic model used here reflects the fact that a message must propagate the entire length of the channel before the channel becomes truly free. In this case, a message continues to occupy the channel for τ units of time after transmission by the sending station terminates. The slotted ALOHA capacity results shown in figure 2-8 have also been cited and discussed in [Molle 81].

Figure 2-8 indicates that pure ALOHA does not always have the lowest capacity. For a large number of stations (e.g., 50), the cost of coordination can require so much overhead in reservation slots that the capacity of some bit-mapped protocols is lower than if there was no coordination in the first place [Kleinrock and Scholl 80]! As expected, figure 2-8 shows that the larger the number of stations, the larger the number of reservation slots and the lower the capacity of the bit-mapped protocol. The capacity results for the bit-mapped protocols were computed using the derivation in [Kleinrock and Scholl 80].

As would be expected, the synchronous, mini-slotted, contention-based protocols with collision detect and transmission abort capabilities attain a higher capacity than their unslotted or fully-slotted counterparts. Note that virtual time CSMA and nonpersistent CSMA protocols have identical capacities; this equivalence has been formally established in [Molle 81]. The capacity results for the time window protocol are derived from the analysis presented in the following chapter, where it is assumed that the protocol maintains no past history of the window splitting process. It has been shown that when the protocol does retain a partial

past history of the window splitting process or can obtain an estimate of the number of messages involved in a collision [Georgiadis and Papantoni-Kazakos 81, Eslanadidi and Chu 82, Towsley and Venkatesh 82], the capacity of the time window protocol can exceed that of non-persistent CSMA/CD.

Perhaps the most interesting result in figure 2-8 is the uniformly highest capacity of TDMA, token passing, the Urn protocol and MSAP/BRAM. The significant underlying common characteristics of these protocols are that no explicit control information is exchanged and that, under heavy traffic conditions, each station in turn is given access to the channel according to some predetermined order. Thus, for a finite number of stations and under the heavy traffic assumption that every station always has a message to send, the channel is constantly used for useful message transmission. A station waits for its turn in the transmission sequence and then simply transmits its message. Note that the capacities of these protocols do not equal 1 due to our assumption that the message continues to occupy the channel for time τ after its transmission has terminated.

The apparent superiority of high capacity protocols should not be overestimated. Protocol capacity is only one measure of protocol performance. It is an important performance measure in that it bounds the maximum amount of useful communication that a protocol can possibly provide. Thus, it provides a yes/no answer as to whether a given protocol can support a given traffic load. On the other hand, the theoretical capacity limit may only be achievable under circumstances which, in practice, are unrealistic. For example, the capacity of the TDMA-like protocols is only achievable under heavy traffic loads, in which case the message delays may be intolerably large. Furthermore, the capacity of a protocol represents a only static measure of its performance. It reflects protocol performance for only a *single* network-wide message generation rate (i.e., that rate which maximizes the effective utilization) and provides *no* information about protocol behavior for other message generation rates.

Thus, capacity can at best only partially characterize protocol performance. In the following section we examine a second performance measure of more practical

interest for non-time-constrained communication applications: the relationship between the rate at which messages are generated at the stations and the average message time delay.

2.5.3. Average Time Delay and Average Throughput

Before presenting a quantitative comparison of the time delay versus throughput performance of various multiple access protocols, let us first discuss two system parameters which greatly affect protocol performance and then discuss an upper bound on the non-time-constrained performance of *any* distributed multiple access protocol. The first system parameter of interest is N, the number of stations in the network. As previously indicated, the number of stations in the network influences the performance of both PCA and demand-adaptive protocols. The performance of contention-based protocols is influenced by the traffic generated by the stations rather than the specific number of stations in the network. The parameter α gives the end-to-end propagation delay of the channel in units of message transmission time and reflects the channel propagation time required to communicate from one station to another. Since the station access order is determined *a priori* in PCA protocols, no information is explicitly exchanged to coordinate channel sharing. The performance of PCA protocols is thus independent of the value of α. However, most demand-adaptive and contention-based protocols involve some form of explicit communication to achieve coordination; thus, these protocols will be affected by the value of α. Finally, we note that in a centralized environment in which all messages are generated at a single central location, messages can be selected for transmission without contention or overhead. Such a centralized system, which can be modeled as an M/D/1 queue [Kleinrock 75], thus provides a bound for the optimal time delay versus throughput tradeoff for *any* multiple access distributed message transmission system.

Figures 2-9a through 2-9c are taken from [Kleinrock 77] and provide quantitative delay versus throughput tradeoffs for various protocols representative of the previously identified major classes of multiple access protocols (PCA, demand adaptive and contention-based). We have chosen not to include

Figure 2-9: Time delay versus throughput tradeoffs

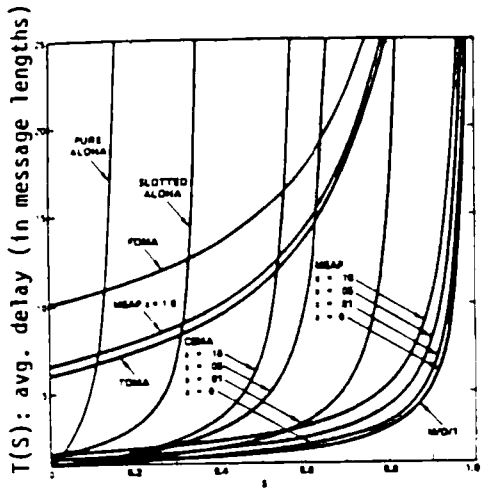


Figure 2-9a: N=10 stations

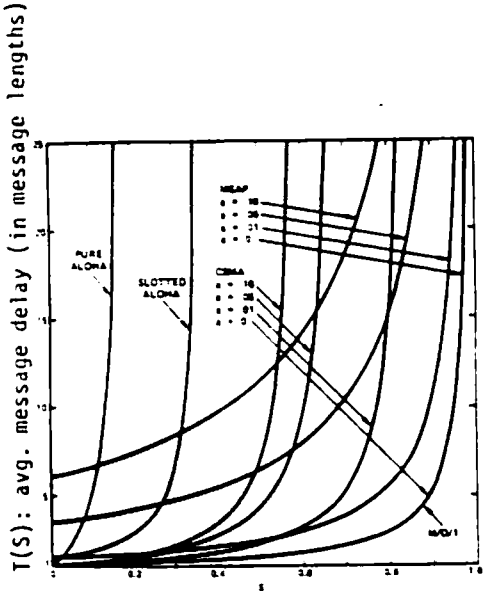


Figure 2-9b: N=100 stations

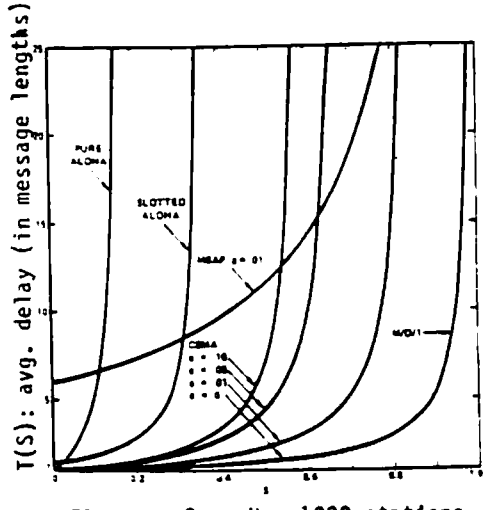


Figure 2-9c: N = 1000 stations

performance curves for all the previously described protocols in an attempt to characterize the relative performance of the *classes* of protocols and to emphasize how the mechanisms underlying each class affects their performance. Such an approach overlooks (often subtle) performance differences that exist between protocols within a given class. A discussion of these differences can be found in the cited references. MSAP is the only demand-adaptive protocol shown in figure 2-9. Since MSAP uses implicit reservations while other demand-adaptive protocols require channel time for explicit reservations or token passing, MSAP offers better throughput and time delay characteristics than other demand adaptive schemes. Two ALOHA protocols and CSMA demonstrate performance characteristics of many of the contention-based protocols.

In figure 2-9, fixed message lengths are assumed and the time scale has been normalized to a message transmission time. S represents the combined message generation rate (or throughput) of messages at all stations in the network. T(S) is the average message delay, i.e., the time between a message's generation at a station and its successful reception at a destination station. The value of **a** corresponds to α, the normalized end-to-end propagation delay of the network. Note that the time delay versus throughput tradeoff shown in figure 2-9 is typical of most shared resource systems: as the number of jobs (messages) contending for the resource (the channel) increases, there is a concommitant increase in the average time needed to acquire the resource (i.e., successfully transmit the message). It should also be noted that the performance curves exhibit asymptotic behavior at some throughput value; this throughput value is exactly the previously examined capacity of the protocol.

Let us first examine figure 2-9a. As expected, the performance of TDMA is independent of **a**. Both ALOHA protocols are also shown to be independent of **a**; this is not strictly valid (since a message is vulnerable to collisions as it propagates down the channel after the sending station has finished transmission) but is a reasonable first approximation. Both CSMA and MSAP are dependent on **a**, as previously discussed. Note that as **a** approaches 0, the performance of CSMA and MSAP converge to the optimal M/D/1 performance. For a relatively small number (10) of stations, MSAP offers the best performance. For small and

intermediate throughput values, CSMA performance is also fairly good, especially for small values of **a**. For high throughput values (i.e., the heavy traffic case), PCA protocols perform as well or better than most of the other protocols; this corroborates our earlier observation that PCA protocols would perform well in heavy traffic situations. However, for small throughput values, their performance is clearly inferior.

In figure 2-9b, the number of stations has been increased to 100. Note that PCA performance curves are not shown in this figure. In TDMA, even as the throughput approaches 0, the average time delay of a generated message is still half the length of a time frame, in this case 50 time units. Thus the performance of TDMA (and MSAP with a=1.0) is literally "off the scale"! The performance of CSMA and ALOHA is the same in figure 2-9b as in figure 2-9a since it is independent of the number of stations in the network. The performance of MSAP has degraded considerably with the increased number of stations and has shifted away from the optimal M/D/1 curve. In figure 2-9c, N has been further increased to 1000 stations and the performance of the contention-based protocols remains constant while that of MSAP deteriorates further.

Figure 2-9 indicates that no protocol or class of protocols performs well for all values of N and S. Thus, if a protocol operates under wide and varying values for these parameters, it should ideally adapt its operation to the current values of N and S. The protocols of Hayes and Capetanakis, the time window protocol and the Urn protocol determine the size of the enabled set as a function of the message generation rate and thus adapt well to changing values of the throughput. In the light traffic case, they operate in a random access fashion. Under heavy traffic loads, they operate in a TDMA-like fashion. These performance characteristics are shown in the time delay versus throughput tradeoff curves for the Urn protocol shown in figure 2-10 [Kleinrock and Yemini 78]. The constant time delay for TDMA results from the assumption of random assignment of TDMA frames and the assumption that a station can buffer at most one message.

most one message.

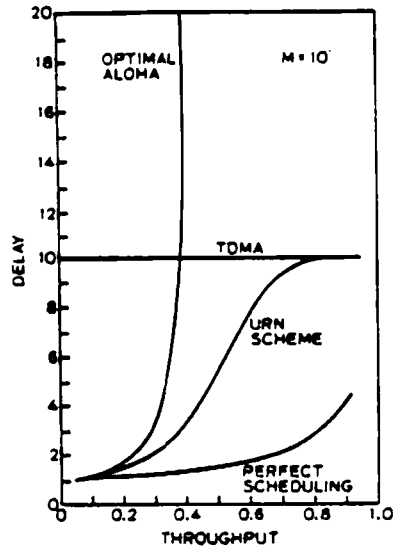


Figure 2-10: Time delay and throughput of the Urn protocol

2.6. Time-Constrained Communication in a Distributed Environment

In the previous sections of this chapter and in chapter 1 we have examined various aspects of the multiple access problem. We have also identified general approaches towards achieving multiple access communication and have examined different protocols employing these approaches. In this section, we examine the issues involved in using these access schemes for supporting time-constrained applications in a multiple access environment. In sections 2.6.1, 2.6.2 and 2.6.3 we discuss the applicability of PCA, demand-adaptive and contention-based protocols in this environment. The relative advantages and disadvantages of these three classes of protocols will be examined and current work in the design and analysis of time-constrained communication protocols will be surveyed.

2.6.1. Predetermined Channel Allocation Protocols

Due to the previously noted inefficiencies of predetermined channel allocation (PCA) protocols such as pure TDMA, this class of protocols has not received much consideration for use in time-constrained communication applications. However, a slight variation on pure TDMA techniques, known as *slot-switched* TDMA, has received considerable attention. This technique has been examined primarily in a centralized environment in which numerous message sources converge at a central location and are then time-multiplexed onto a single outbound channel [Gruber 81, Arthurs and Stuck 79, Mowafi and Kelly 80, Fischer and Harris 76]. Maglaris [Maglaris and Lissack 81] has also investigated the use of slot-switched TDMA in a distributed environment with centralized control.

In slot-switched TDMA (as in pure TDMA), time is divided into fixed length frames and frames are again further divided into slots. However, there are now fewer slots per frame than there are stations. The purpose of reducing the number of slots per frame is to decrease idle channel time and message waiting time due to empty slots being held by stations with nothing to send. Since there are fewer slots than stations, a station must first *claim* a fixed slot number within each frame before using it. Slot allocation is typically determined by beginning each time frame with a slot request period as in demand-adaptive reservation schemes. Allocation is then performed either by a centralized station [Maglaris and Lissack 81] or by the distributed stations according to some policy known to all stations.

Once a station is assigned the ith slot in a frame, it then has sole access to the communication channel during the ith slot of every subsequent frame until it explicitly releases the slot. Typically, a slot is held for numerous frame periods. Note that, as in pure circuit switching, once a station is assigned a slot, it is guaranteed a fixed channel bandwidth (i.e., the full channel bandwidth for one time slot each frame). However, the channel is also time multiplexed among all stations which have been assigned time slots. For this reason, techniques such as slot-switched TDMA are also known as "virtual circuit" approaches and the time during which a station holds the channel is known as a "virtual connection".

An interesting characteristic of PCA protocols is that since a station transmits in a synchronous fashion (i.e., only during its time slot), data is received at the destination station in a synchronous fashion. Thus all messages have a deterministic waiting and transmission time and 100% of the messages are delivered with a fixed time delay. However, a severe price is paid for this 100% reliability and fixed delay. In pure TDMA and slot-switched TDMA, the time delay may be intolerably large. Moreover, since the maximum number of stations which can be multiplexed is fixed by the number of slots in a frame in slot-switched TDMA, once all the frame slots have been assigned, all other stations are blocked from using the channel. This blocking phenomenon thus trades the 100% reliability of messages in a virtual connection with the 0% reliability of messages in a blocked connection (i.e., messages at a station which is denied a frame slot). This problem will be further examined in the following sections on demand-adaptive and contention-based protocols.

As we will see, PCA has both advantages and disadvantages with respect to other strategies for supporting time-constrained communication applications. In addition to the fixed time delay and 100% reliability for assigned connections, the synchronous delivery of data can be an important advantage for applications such as packetized voice, which require synchronous playout of the received messages. Also, since the receiving station knows that data will be received synchronously, (after reception of the first packet), there is no need for control information between the first and last slots used by the connection; this contributes towards better utilization of the channel bandwidth.

As previously discussed, two factors contribute to possible substantial waste of channel bandwidth by PCA protocols. If relatively few slots within a frame are claimed, a station is still only permitted to transmit during its assigned slot, even if it could potentially utilize the remaining empty slots within the frame. Even if most of the slots within a frame are utilized, "silence" periods within a connection can result in wasted channel bandwidth. For example, in packetized voice applications, an average 60% [Bially et al. 80a] of a connection is spent in silence. In the case of numerous connections emanating from a central location, it may be possible to multiplex other connections into the silent periods of a

connection [Bullington and Fraser 59]. However, in a distributed environment, multiplexing connections from geographically distributed stations into a time slot becomes much more difficult. One other limitation of PCA is that while it may be well-suited for message traffic that is relatively synchronous over a long period of time, it may not be as well-suited to bursty traffic such as interactive data. Thus, the integration of both stream-like and bursty sources at a station would probably require separate access mechanisms for the two types of traffic.

In summary, PCA techniques are most suitable for environments requiring long, synchronous, stream-like data transmissions. However, these PCA techniques may preclude guaranteed connections, suffer from large time delays, be potentially wasteful of channel bandwidth and may not be suitable for integration of bursty traffic sources with stream-like sources.

2.6.2. Demand-Adaptive Protocols

Two characteristics of most demand-adaptive protocols make them more attractive than PCA protocols for time-constrained communication applications. First, they provide every station with a *guaranteed* amount of communication capacity. Second, if the time required for token passing or reservation posting is small compared with the message transmission time, unlike PCA protocols, channel bandwidth is not wasted by stations with nothing to transmit.

As in PCA protocols, most demand-adaptive protocols implement some form of round robin channel sharing. This assures that each station will receive a guaranteed bandwidth of one slot per round. However, unlike slot-switched TDMA, *all* stations are guaranteed this bandwidth and thus all connections are guaranteed. The time delay between transmissions by a station is bounded by the maximum length of a polling round or reservation/transmission cycle. Since not all stations may choose to transmit during a round, the length of a round can be less than this maximum value. Even though the time delay is bounded, however, if the time constraint imposed on message delays is less than the maximum time bound, some fraction of the messages may still be lost due to excessive time delays.

If an increasing number of stations begin to transmit messages, the increasing system load is translated into longer delays and thus possible message loss. Recall that in PCA protocols an increasing traffic load results in the blocking of connections and the loss of all messages within these connections. Thus, demand-adaptive protocols might be considered "fairer" than predetermined channel allocation protocols, since they provide all stations with the same delay and loss characteristics rather than providing one set of stations with a high grade of service (i.e., a guaranteed connection with fixed delays) and another set of stations with a lower grade of service (i.e., no connection).

The variable message delay has an important consequence for time-constrained applications such as packetized voice, which require synchronous message playout at the receiving station. In order to "smooth out" the variability of the delays, it is necessary to buffer received messages and introduce an additional initial delay before playout begins. The effect of adding the initial additional delay is shown below in figure 2-11 [Cohen 77].

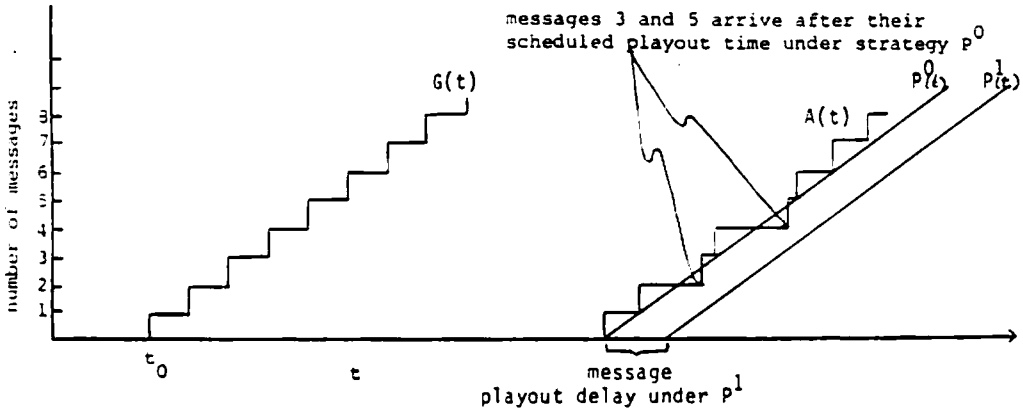


Figure 2-11: The effects of an additional initial waiting time

Figure 2-11 shows a station which begins synchronous generation of messages at time t_0. Let:

- G(t) be the number of packets generated at the source by time t. G(t) increases by unit steps.
- A(t) be the number of packets received at the destination by time t.
- P(t) be defined as follows. If $P(t) > i$, playout of message i should begin at the destination station by time t. Since playout is synchronous, P(t) is a straight line. Let $P^0(t)$ be the playout strategy with no initial additional delay. Let $P^1(t)$ be the playout strategy with an initial additional delay introduced.

Figure 2-11 shows that if synchronous message playout begins immediately upon receipt of the first message (shown by the line P^0), messages 3 and 5 will be lost. However, under playout strategy P^1, in which an additional initial delay has been added, neither message 3 nor message 5 will be lost. Thus, the effect of adding the initial delay is to "smooth out" the differences in message delays.

In figure 2-11 messages are lost because they have not yet arrived at the receiving station when their playout is scheduled to begin. Such lost messages introduce a "glitch" into the synchronous playout. The effects of such delays and glitches on human voice communication has been extensively studied; [Forgie 75] contains an excellent summary of this work. Buffering and delay strategies have also been examined in [Cohen 77] and [Gopal et al. 81].

A critical factor in determining the applicability of demand-adaptive protocols in a time-constrained environment is the number of stations accessing the network. If there are a large number of stations and a large fraction of those stations choose to use their transmission slots in a polling round, the length of a polling round can grow quite large. This may result in a bounded but excessive time delay for time-constrained communication [Kim 83]. Furthermore, if there are a large number of stations using the channel infrequently, a large portion of the channel bandwidth will be wasted on the polling or reservation overhead. This results from the fact that each station must use the channel (if only to

broadcast silence) to inform the other stations that it will not use a time slot during the current polling round.

In summary then, the properties of guaranteed connections, bounded time delay, and the ability to take full advantage of the three-way trade off between time constraint, loss and offered traffic load make demand-adaptive protocols attractive for at least some time-constrained applications. The applicability of demand-adaptive protocols, however, will be strongly dependent on the number of stations in the environment and the rate at which messages are generated at the sending stations.

2.6.3. Contention-Based Protocols

In contention-based protocols, stations access the channel on a message by message basis and thus there is no concept of a "connection". Since there are no connections to be blocked, an increasing traffic load on the channel results in increasingly longer delays and consequently higher message loss. Flow control techniques for limiting the congestion during these periods of excessive traffic have been investigated in the literature; [Bially et al. 80b] and [Forgie and Nemeth 77] discuss general flow control techniques for packetized voice systems.

Contention-based protocols (like demand-adaptive protocols) introduce a variable message delay. Thus, a smoothing buffer and additional delay may also be required for applications requiring synchronous playout at the destination station. A subtle, yet important difference exists between the delays in contention-based and demand-adaptive protocols. In the latter case, the time between successful transmissions by a station is always bounded above by one frame length. This is not true, however of contention-based protocols and thus different stations may have message delay distributions. Maxemchuk [Maxemchuk 82] has noted that while it may be valid to assume that packets are lost at random when aperiodic sources generate traffic, this assumption does not necessarily hold if a majority of the sources are periodic. In this case, the random loss assumption must be carefully examined given the particular random access mechanism employed. Recent experimental measurements by Gonsalves [Gonsalves 83], however, indicate

that for at least one random access protocol (Ethernet), approximately equal message loss is experienced by all the stations in the network.

The advantages of contention-based protocols for time-constrained communication are similar to those for demand adaptive protocols: guaranteed connectivity, the transfer of temporary overloading into packet delay rather than blocked connections and the ability to trade message loss for message delay. In addition, message delays depend only on the traffic load accessing the network and are independent of the *number* of stations accessing the channel. The major disadvantage of contention-based protocols for time-constrained applications is the variable message delay. For stations requiring a synchronous playout of received messages, this necessitates additional buffering capacity and the introduction of an additional time delay before synchronous playout begins.

Recently, some specific contention-based protocols have been proposed and studied in the context of time-constrained communication. Gonsalves [Gonsalves 83] has experimentally investigated the transmission of voice traffic over a 3 Mbits/sec, 0.5 kilometer Ethernet (CSMA/CD with backoff) network. He found that this system can support up to 35 simultaneous 64 Kbits/sec two-way voice conversations by delivering approximately 99% of the messages with a delay less than 10 ms. Nutt and Bayer [Nutt and Bayer 82] have simulated the operation of a 10 Mbits/sec, 1.0 kilometer Ethernet system under a combined voice and data load. Their results indicate that even when the network is supporting data loads such as those observed by [Shoch and Hupp 80], it can additionally support up to 25 simultaneous, two-way 64 Kbits/sec voice calls and deliver approximately 99% of the voice messages with less than a 10 ms delay.

The loss and delay values observed by Nutt and Gonsalves are within the limits for good human speech quality, thus indicating the feasibility of voice communication in a large scale distributed multiple access environment. The performance of voice communication in a contention-based multiple access environment has also been studied by Maxemchuk [Maxemchuk 82]. He observed that the deterministic, synchronous nature of voice communication can be exploited to reduce channel contention and proposed a time-constrained protocol offering performance advantages over the Ethernet protocol.

Chapter 3
The Role of Scheduling in Time-Constrained Communication

3.1. Introduction

In the previous two chapters we have seen that a multiple access protocol has traditionally been viewed as a resource sharing mechanism which permits geographically distributed stations to share access to a single, broadcast communication channel. Note, however, that in addition to this traditional role as an arbiter of channel sharing, any multiple access protocol also serves as a *distributed scheduling mechanism* by (explicitly or implicitly) imposing some *network-wide* ordering on the transmission of messages waiting to be sent at the various stations in the network. (We emphasize "network-wide" here to distinguish the case in which a station determines the order in which its own locally-generated messages are transmitted from the present case in which the transmission order of *all* messages in the network is considered).

This scheduling role of an access protocol has been almost completely overlooked in the past. The reason for this oversight and the importance of this scheduling function for time-constrained communication applications can be readily discerned by considering the primary performance metrics for time-constrained and non-time-constrained applications discussed in chapter 1. For traditional, non-time-constrained applications, the primary performance metric is average delay. Given a network-wide trace or history of the operation of any multiple access protocol, permuting the order of message transmissions has *no effect* on the average message delay. That is, the average message delay is invariant with respect to the *order* in which the messages are actually transmitted [Kleinrock 75] and the primary performance metric for non-time-constrained applications is thus not affected by the message transmission order. However, although the average delay is invariant to the transmission order, the

delay distribution is not. Since message loss in time-constrained applications results from a message's delay exceeding a given time constraint, message loss is dependent on the distribution of message delay and a protocol's scheduling function will thus play a critical role in determining its performance for time-constrained applications.

Given the importance of this scheduling role, in this chapter we present and analyze a new access protocol, based on a generalization of time window protocols [Gallager 78, Towsley and Venkatesh 82], which provides explicit control over the imposed *network-wide* message transmission scheduling discipline. In the following section we describe this random access protocol and discuss its use as a distributed scheduling mechanism. We then study three particular cases in which the protocol provides FCFS, LCFS and Random scheduling and examine the message delay distribution for each of these disciplines. Both analytic and simulation results are presented. Using these results, we then compare the time-constrained performance of the disciplines. Another scheduling discipline which specifically attempts to maximize the percentage of messages sent with a delay less than a specified time constraint is then presented and examined. Finally, the performance results are used to illustrate several important features of the scheduling role of a random access protocol and the impact of this role on its time-constrained performance.

3.2. A Protocol for Time-Constrained Communication Over a CSMA Channel

3.2.1. Tree Random Access Protocols

The protocol presented in this section belongs to the class of CSMA/CD contention-based protocols known as tree protocols [Tanenbaum 81]. The underlying common characteristic of this class of multiple access protocols is the attempt to limit contention by granting channel transmission rights at any given time to only a subset of the stations in the network (known as the enabled stations). A station can transmit a message only when enabled. If no stations in the enabled set are ready (i.e., have a message to send), the channel is idle

and a new enabled set is eventually determined. If two or more ready stations are in the enabled set, they transmit interfering messages and the enabled set is typically reduced in size in an attempt to isolate exactly one ready station in the enabled set. When the enabled set contains exactly one ready station, that station transmits its message without interference. The full operation of one type of tree protocol will be illustrated shortly by example.

In the tree protocols of [Hayes 78] and [Capetanakis 79], the enabled set is determined using station addresses. These protocols maintain a set of station addresses such that any station whose address is in this set is enabled. The tree protocols proposed in [Gallager 78] and [Towsley and Venkatesh 82] use a time window mechanism to determine the enabled set of stations. These protocols maintain an interval of time or *time window* in the past and all stations which have an unsent message which was generated during this time interval are enabled. The protocol presented in this chapter is based on a generalization of the use of time windows. A related protocol using message generation times to determine channel transmission rights has also been proposed by Molle [Molle 81].

In the references cited above, the enabled set of stations is maintained in such a manner so as to minimize the average delay of messages. As previously indicated, the delay *distribution* depends on the imposed message transmission order and critically determines the performance of a protocol for time-constrained communication applications. Thus, the protocol presented in this section provides explicit control over the network-wide order in which messages are transmitted. The goal, of course, will then be to control this scheduling function in order to minimize message loss for a given imposed time constraint.

3.2.2. Description of the Protocol

Let us assume that each station on the multiple access channel possesses a clock which defines the current time, t, and that the clocks at all stations are synchronized. Each station will maintain a value for each unsent message which is generated at the station called the *modified generation time* of the message. The initial value of a message's modified generation time is the actual generation

time of the message at the station; a message's modified generation time may change as described below. In addition, each station will also maintain a value, t_past, such that all messages currently at any station in the network have a modified generation time greater than t_past; all stations initialize t_past to the initial clock value. Finally, each station has a pseudo random number generator and each station initializes the generator with the same seed and therefore produces the same sequence of pseudo-random numbers.

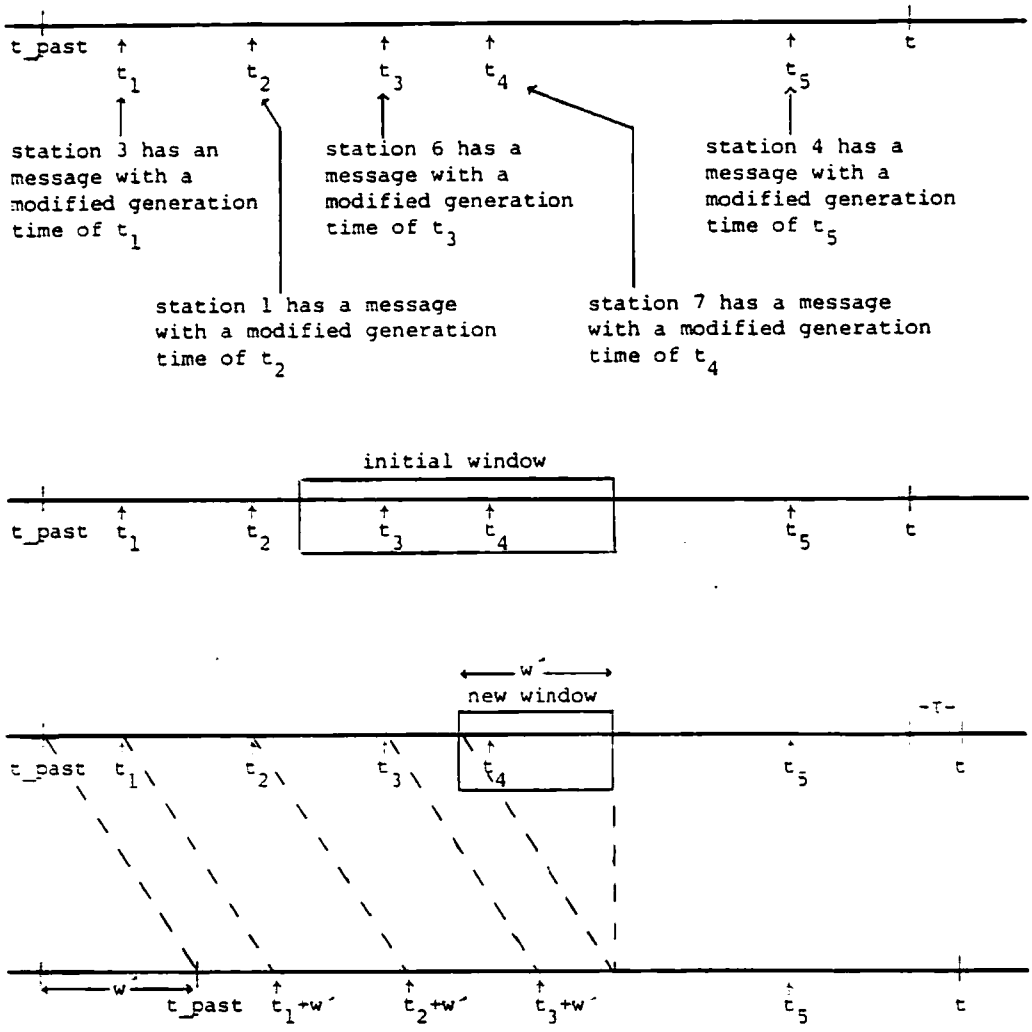


Figure 3-1: The time window protocol

The operation of the protocol is shown in figure 3-1. The modified generation times of all messages at all stations throughout the network which have not yet been successfully transmitted are shown below the time axes in this figure. Note that an individual station knows about messages generated locally but does not know about messages generated at other stations in the network. For example, in figure 3-1, station 6 would know that it has a message with a modified generation time of t_3, but it would not be aware of any of the other messages shown in the figure.

The protocol operates in a synchronous fashion. We will thus consider the time axis to be slotted, with the duration of a slot length equal to τ, the end-to-end propagation delay of the channel (τ thus defines the atomic unit of discrete time in the network). All stations continuously monitor the channel and after an empty channel slot (i.e., an empty slot either following a successful message transmission or resulting from identical values of t_past and t, where t is the current time), each station uses its random number generator and a scheduling policy to be described below to select an interval of time, or *time window*, between t_past and the current time, t. Since all stations have the same value of t_past and generate the same sequence of random numbers, all stations will select the same window of time. As we will see, the assumption that every station selects the same window of time and maintains the same value of t_past can be easily relaxed; for the moment, however, we will keep these assumptions.

The selection of an initial time window is shown on the second time axis in figure 3-1. After this window has been selected, all stations with a message with a modified generation time which falls within this interval of time attempt to transmit the message; one of three possibilities can then occur.

One possibility is that no station has a message with a modified generation time in the selected initial time window. In this case, no stations are enabled, no messages are transmitted and the channel remains silent. Once the channel remains silent for time τ, all stations know that no other stations have a message with a modified generation time in this time interval. All stations then select a new initial time window and repeat the channel access procedure after

updating their value of t_past and the modified generation times of unsent messages as discussed below.

The second possibility is that exactly one station has a message with a modified generation time in the initial window. In this case, this station begins transmitting its message without interference.

Finally, if more than one station has a message with a modified generation time which falls within the selected time window (the second time axis in figure 3-1), two or more stations attempt to simultaneously transmit a message and a collision occurs. In the second time axis in figure 3-1, for example, stations 6 and 7 transmit interfering messages. If there is a collision, it is detected by all stations within time τ. *All* stations continue to monitor the channel and attempt to resolve this collision by splitting the initial time window in half. The stations then use their random number generator and the scheduling policy described below to select one of the two halves of the initial window, as shown on the third time axis in figure 3-1. Since all stations use the same sequence of pseudo random numbers, all stations will select the same half of the split window. The random access procedure is then repeated using the selected half of the split window as the new time window. If no message has a modified generation time which falls in the selected half of a split window, an empty slot will occur on the channel and all stations will then select the remaining half of the split window, immediately split this new window (since it is known to contain two or more messages), choose one of the halves of the newly split window, and repeat the above access procedure using that half of the newly split window. This splitting process continues until a single message is finally transmitted.

All stations perform the windowing process and thus each station knows the width, w', and the starting time, t', of every time window which contains either no message generations or a single message generation. Once a window with exactly zero or one message generations has been selected and the message (if any) has been transmitted, all stations can effectively *remove this window of time from consideration*, as if the interval of time had never occurred. The effect of removing this window of time is twofold. First, all stations with a

message with a modified generation time before t' must update the modified generation time of the message by the width, w', of the window. Secondly, since the modified generation time of each message which was generated before t' has been increased by w', there is an interval of time from t_past to t_past+w' for which it is known that there are *no* messages network-wide which have a modified generation time within this interval. Thus, *all* stations can update t_past accordingly. The effect of updating the value of t_past and the modified generation times is shown on the fourth time axis in figure 3-1.

The selection of the position of the initial window and the method by which one of the two halves of a split window is selected determine the scheduling policy implemented by the protocol. Suppose that the time between t_past and t divides into n equal length windows; we note three special cases:

1. The first (oldest) of the n windows is always selected as the initial window and the first half of a split window is always chosen before the second.

2. The last (newest) of the n windows is always selected as the initial window and the second half of a split window is always chosen before the first.

3. Each of the n windows is equally likely to be chosen as the initial window and both halves of a split window are equally likely to be chosen first.

Case (1) above implements FCFS scheduling, case (2) implements LCFS scheduling and case (3) implements Random scheduling. In the most general case:

4. Each of the n windows is selected with some probability according to some discrete probability distribution δ and the first half of a split window is selected with probability q and the second half is selected with probability 1-q.

Depending on the distribution δ chosen in (4) and the value of q, any one of a large class of scheduling disciplines can be selected. As we will see, the selection of δ and q should be dependent on current system demands, including the offered load and loss tolerances.

3.2.3. Generalizing the Basic Time Window Mechanism

In the protocol description of the previous section, we have assumed:

1. All stations choose the same position (in time) and size for the initial time window and always choose the same half of a split window (and thus as a result, all stations maintain the same value of t_past).

2. All stations simultaneously select a time window and thus begin their message transmissions (if any) simultaneously.

The protocol itself, however, is robust in the sense that it will continue to operate even if the first assumption is violated. In fact, in certain circumstances it may even be desirable to explicitly relax this condition.

To see that this assumption can be relaxed, we first note that the selection of a time window simply serves as a probabilistic transmission mechanism. When a station chooses an initial time window, it essentially observes a *local* random process (its own message generation process) over an interval of time, with the interval of time itself being chosen independently of the observed process. The result of this observation is that if a locally generated message has a modified generation time falling within the observed interval, the station transmits that message; otherwise it remains silent. In the protocol description above, we have required every station to observe its local message generation process over the same interval of time. Note, however, that if different stations observe this process over different intervals of time (i.e., they choose different initial time windows), the only possible difference from the case in which all stations behave identically is that the set of resulting transmission probabilities may be different. If collisions occur, each station simply splits its locally determined window as before and the window splitting process again serves simply to decrease the size of the enabled set.

In the worst case, the resulting transmission probabilities may be such that the stations transmit too frequently (in which case the windows must be split an excessive number of times) or too infrequently (in which case an excessive number of empty windows are chosen). In any case, however, the protocol continues to operate, although possibly (but not necessarily) at some degraded

performance level. As we will see in chapter 6, permitting stations to explicitly choose different window sizes provides a natural mechanism for introducing priorities into the network. The basic idea in such a priority scheme is to permit higher priority stations to choose larger initial windows than the lower priority stations. The larger the initial time window, the larger the probability that a message was generated locally within the window, hence the larger the probability of transmission for the higher priority stations. Conversely, the smaller the window, the smaller the probability that a message was generated within the chosen window and hence the smaller the probability of transmission.

The second assumption that all stations simultaneously determine their initial window (and thus begin their message transmissions, if any, simultaneously) is equivalent to stating that the stations operate in a synchronous manner. This assumption cannot be relaxed given the current description of protocol operation. Recently, Molle [Molle 83] has examined the effect of asynchronous operation of tree protocols on their capacity. However, the effect of asynchronous operation on the delay distribution and hence the time-constrained performance of tree protocols has not been addressed in the literature and remains an open problem for future work.

3.2.4. Analysis of the Average Performance of the Window Mechanism

In this section we derive an approximate expression for the average message scheduling time of the protocol, defined as the average time between the selection an initial time window and the beginning of a successful message transmission. In order to obtain this expression we will first follow Towsley [Towsley and Venkatesh 82], Molle [Molle 81] and other references therein and analytically determine the exact value of the message scheduling delay under saturation conditions, which occur when the average message generation rate exceeds the average time needed to schedule and transmit a message. Specifically, we will make use of the property that at saturation, the average waiting time is unbounded and thus *the difference between the current time and t_past is always greater than any window width chosen.*

Once the average *scheduling* delay under saturation conditions has been obtained, the actual message generation rate at which saturation occurs can then be determined. This value for the saturation message generation rate and the average message scheduling time at saturation can then be used to provide an approximate expression for the message scheduling delay for message generation rates less than saturation.

Recall that time is slotted in units of τ, where τ is the end-to-end propagation delay of the channel. We will assume that all stations can detect message collisions (CSMA-CD) and can abort transmission of a collided message in a negligible amount of time. The *overall* generation of messages at *all* stations in the network is assumed to constitute a Poisson process with rate λ (message generations/slot).

The final assumption is that since the overall message generation process is Poisson, at *any* point in time, the time between successive *modified* generation times of messages (on a network-wide basis) is exponentially distributed with mean $1/\lambda$. This is equivalent to assuming that the removal of a time window and shifting of the modified generation times preserves the exponential inter-generation times of messages. Note that this property is exactly preserved when an entire initial window is removed since the position of the initial window is selected independently of the modified message generation times. However, if one half of a split window is removed, it can be shown that the inter-generation times of messages in the remaining half of the window are no longer exponentially distributed. Nonetheless, the agreement between our simulation and analytic results indicate that the following analysis is based on reasonable assumptions and approximations.

One important consequence of the (approximated) exponential nature of the modified inter-generation times is that any two windows of time between the current time and t_past which are of equal length are statistically identical with respect to the modified generation times. Thus the position (along the time axis) of the initial window and the policy for selecting one half of a split window have no effect on the statistics of the average message scheduling delay. In particular,

the average message scheduling time is independent of the scheduling discipline implemented by the protocol.

Let us now proceed to determine the average message scheduling delay which, in general, will be a function of the network-wide message generation rate, λ. Define:

$\bar{s}(\lambda)$ - average time (in slots) to schedule a message, assuming messages are being generated (network-wide) at rate λ.

As discussed earlier, in order to obtain an approximate expression for $\bar{s}(\lambda)$, we will first determine the exact value of $\bar{s}(\lambda)$, under saturation conditions. Define:

$\bar{s}_{sat}$ - the average number of slots needed to schedule a message at saturation. The last slot in the windowing process (in which exactly 1 message is transmitted) is *not* considered part of the scheduling time.

s_k - the average number of slots needed to schedule a message given a window is known to contain k messages, with $k \geq 2$.

$q_{k,i}$ - the probability that i messages are in one half of a window given there are k messages in the window. Due to the memoryless property of the inter-generation process, $q_{k,i}$ is given by:

$$q_{k,i} = \binom{k}{i} 2^{-k}$$

p_i - the probability of i message generations in a window. If the length of a window is given by the parameter w (in units of time slots), then:

$$p_i = (\lambda w)^i e^{-\lambda w} / i!$$

Now, $\bar{s}_{sat}$ can be expressed in terms of s_k by conditioning on the number of message generations in a window:

$$\bar{s}_{sat} = p_0(1+\bar{s}_{sat}) + p_1 \cdot 0 + \sum_{k=2}^{\infty} p_k(1+s_k)$$

or

$$\bar{s}_{sat} = \frac{p_0 + \sum_{k=2}^{\infty} p_k(1+s_k)}{1 - p_0} \qquad (3.1)$$

The added 1's in the terms $(1+s_k)$ and $(1+\bar{s}_{sat})$ in equation 3.1 are due to the fact that 1 time slot is first required to learn that zero or two or more collisions have occurred. We can now condition s_k on the events of zero, one, or more than one message generations in the selected half of a split window. Note that if

no message occurs in the selected half of a split window (this occurs with probability $q_{k,0}$), no slot is needed to determine that two or more messages occur in the remaining half of the split window and that half can be split immediately. Conditioning s_k thus gives:

$$s_k = q_{k,0}(1+(s_k-1)) + q_{k,1}\cdot 0 + \sum_{i=2}^{k} q_{k,i}(1+s_i)$$

or

$$s_k = \frac{(1 - q_{k,1} - q_{k,0}) + \sum_{i=2}^{k-1} q_{k,i}s_i}{(1 - q_{k,0} - q_{k,k})} \qquad (3 \leq k) \tag{3.2}$$

with the initial condition $s_2 = 0.5$. Values for s_k can be iteratively obtained from equation 3.2 above and then substituted into equation 3.1 to determine the value for $\bar{s}_{sat}$. Note, however, that $\bar{s}_{sat}$ is a function of the yet unspecified value of λw. This parameter will be chosen to minimize the value of $\bar{s}_{sat}$. Equation 3.1 can be evaluated numerically to show that $\bar{s}_{sat}$ achieves a minimum value of 1.24 slots when λw has a value of 1.2. Thus, regardless of the actual message generation rate at which saturation occurs, the initial window size, w, should be chosen such that $\lambda w = 1.2$ and thus $\bar{s}_{sat} = 1.24$ slots, independent of the message generation rate at which saturation occurs.

Equation 3.1 thus gives the average message scheduling delay under saturation conditions to be 1.24 slots; let us now determine the actual message generation rate at which saturation occurs. Following Lam [Lam 80], we note that the overall channel utilization ρ (the message generation rate times the average time required to select (schedule) and transmit a message) must be bounded above by unity. If M is the average actual transmission time of a message (measured in units of τ), this channel utilization bound is expressed:

$$\lambda(M + \bar{s}(\lambda)) \leq 1 \tag{3.3}$$

The saturation value of the generation rate, λ_{sat}, is that value of λ for which the equality in equation 3.3 holds and thus:

$$\lambda_{sat} = 1 / (M + \bar{s}_{sat})$$

or (3.4)

$$\lambda_{sat} = 1 / (M + 1.24)$$

If we define the effective channel throughput, ρ', to be the fraction of the channel which is utilized by successfully transmitted messages, (i.e., $\rho' = \lambda M$) the effective channel throughput at saturation or the maximum effective channel utilization is given by:

$$\rho'_{sat} = \frac{M}{M + \bar{s}_{sat}} = \frac{1}{1 + \bar{s}_{sat}/M}$$

The analysis thus far has provided the average message scheduling delay under saturation conditions as well as the actual message generation rate (and thus effective throughput) at which saturation occurs. We also know that as λ and thus ρ' approaches zero, the average message scheduling delay also approaches zero since a message would always be sent without contention as soon as it is generated. Given these two endpoint values, we will approximate the intermediate points of the average message scheduling time, $\bar{s}(\rho')$, by fitting a function of the form $\rho' / (b - \rho')$ to these endpoint values, where b is a suitably chosen constant. The results of this approximation are compared with the average message scheduling times obtained through simulation for various message sizes in figure 3-2. In the simulations, the initial window size, w, was chosen such that $\lambda w = 1.2$ for all values of λ.

In the following section, the average message scheduling times will be used to study the message delay distribution under different scheduling disciplines.

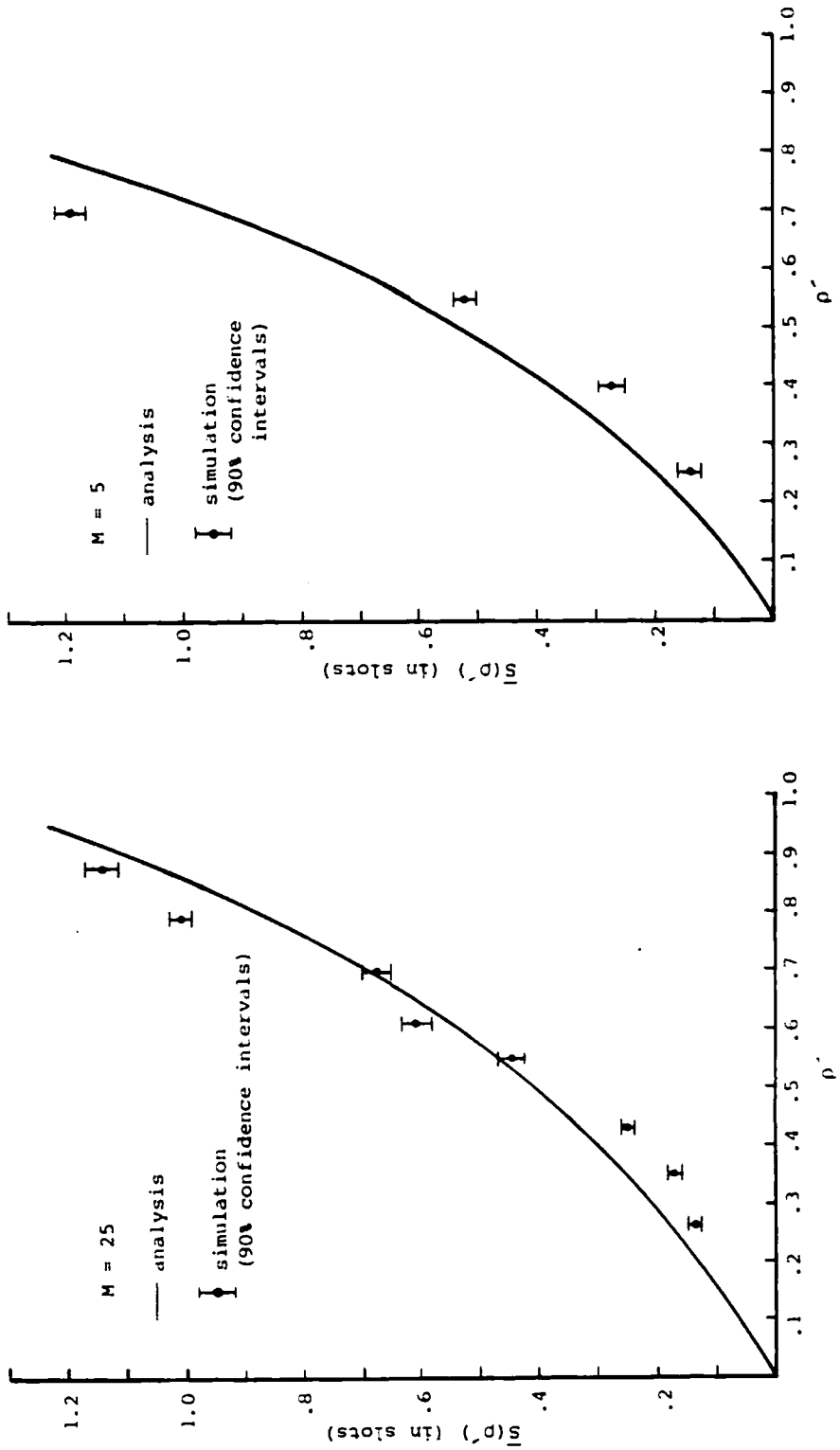


Figure 3-2: Average message scheduling times as a function of ρ'

3.3. Waiting Time Distributions for FCFS, LCFS and Random Scheduling

In this section, we present analytic and simulation results for the percentage of messages lost (i.e., having delays exceeding a given time constraint) as a function of the imposed time constraint for fixed length packets and for cases in which the window random access protocol provides FCFS, LCFS and Random scheduling. *The delay of a message will be defined as the time between a message's generation at a station and the beginning of the contention (windowing) period (if any) immediately preceding, and resulting in, its successful transmission*

3.3.1. Service Time Distribution of Messages

The analysis developed in this section is based on viewing the messages distributed among the stations throughout the network as customers in a *distributed* queue. If we view channel activity as a function of time, the window protocol results in alternate use of the channel for the windowing process (to determine the message to be transmitted) and for the successful transmission of that message as shown in figure 3-3.

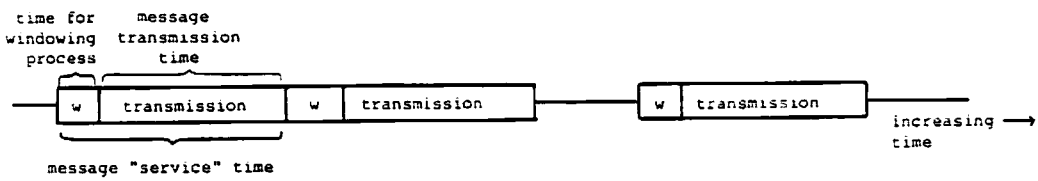


Figure 3-3: The "service time" of messages

In our distributed queueing model we thus consider the time required by the windowing processes immediately preceding, and resulting in, a message's transmission (i.e., its scheduling time) to be part of its *service* time. The scheduling time component of a message's service time is thus the time between either the end of the most recent message transmission or its own generation

time (whichever is more recent) and the start of its own successful transmission. In this case the distributed queue is equivalent to a centralized queue in which a message's service time consists of two components: the scheduling time component and the actual transmission time component.

Let us now determine the service time distribution for fixed length messages. First, we note that since the statistics of the message scheduling time have been shown to be independent of the scheduling discipline imposed by the protocol, this service time distribution will also be independent of the scheduling discipline. Let b(t) be the distribution (probability density function) of the message service time, s(t) be the distribution of the scheduling time (i.e., the distribution of the length of time required by the windowing process to determine the single message to be transmitted), and x(t) be the distribution of the message transmission time. Since the service time is the sum of two independent random variables, the service time distribution is given by:

$$b(t) = s(t) \odot x(t) \tag{3.5}$$

where $\odot$ is the convolution operator. For the case of fixed length messages, x(t) is given by:

$$x(t) = \mu_0(t - M) \tag{3.6}$$

where μ_0 is the unit impulse function and M is the fixed message length (in units of τ).

The distribution of the message scheduling time is more difficult to obtain. However, our simulation studies have shown that the geometric distribution is a good approximation for the message scheduling time distribution, where the mean of the geometric distribution is taken to be the mean scheduling time as determined in the previous section. Let $\bar{s}(\lambda)$ be the average message scheduling delay and define c_i to be the probability that the message scheduling delay is i slots. Then:

$$c_i = c(1-c)^i$$

where:

$$c = 1 / (1 + \bar{s}(\lambda))$$

and the probability distribution for the message scheduling time is thus given by:

$$s(t) = \sum_{i=0}^{\infty} c_i \mu_0(t-i) \tag{3.7}$$

Using the value for x(t) and s(t) from equations 3.6 and 3.7 respectively, equation 3.5 gives the service time distribution for a message as:

$$b(t) = \sum_{i=0}^{\infty} c_i \mu_0(t - (M + i)) \tag{3.8}$$

3.3.2. Distribution for FCFS Scheduling

Since the message scheduling or collision resolution time has been modeled as part of the service time of the messages, our model of the distributed queue reduces to the case of an M/G/1 queue, where the service distribution of the customers is given by equation 3.8.

The waiting time density function for customers in an FCFS M/G/1 queue is given by [Kleinrock 75]:

$$w_{fcfs}(t) = \sum_{k=0}^{\infty} (1-\rho)\rho^k \beta^k(t) \tag{3.9}$$

where

$\beta(t)$ = the density function of the residual service time that an arriving customer finds for the customer (if any) in service.

$\beta^k(t)$ = the k-fold convolution of $\beta(t)$

ρ = the server utilization, previously defined by $\rho = \lambda(M + \bar{s}(\lambda))$

For values of ρ not especially close to unity, the infinite sum in equation 3.9 can be truncated at some finite value of k to produce an excellent approximation to $w_{fcfs}(t)$. The residual service time density for a given message generation rate (λ) and a service time distribution given by equation 3.8 can be calculated [Kleinrock 75] to be:

$$\beta(t) = 1 - \frac{\sum_{i=0}^{\infty} c_i \mu_{-1}(t - (M + i))}{(M + \bar{s}(\lambda))} \tag{3.10}$$

where μ_{-1} is the unit step function.

Since messages are lost when their waiting time at the sending station exceeds a given time constraint, K (measured in units of τ), the fraction of messages lost under FCFS scheduling (equivalently, the steady state probability that a message is lost) is given by:

$$\text{loss}(K) = 1.0 - \int_0^K w_{fcfs}(t)dt \qquad (3.11)$$

Note that the second term on the right hand side of equation 3.11 is the fraction of messages with a delay less than K. Hence, loss(K) is simply 1.0 minus this value.

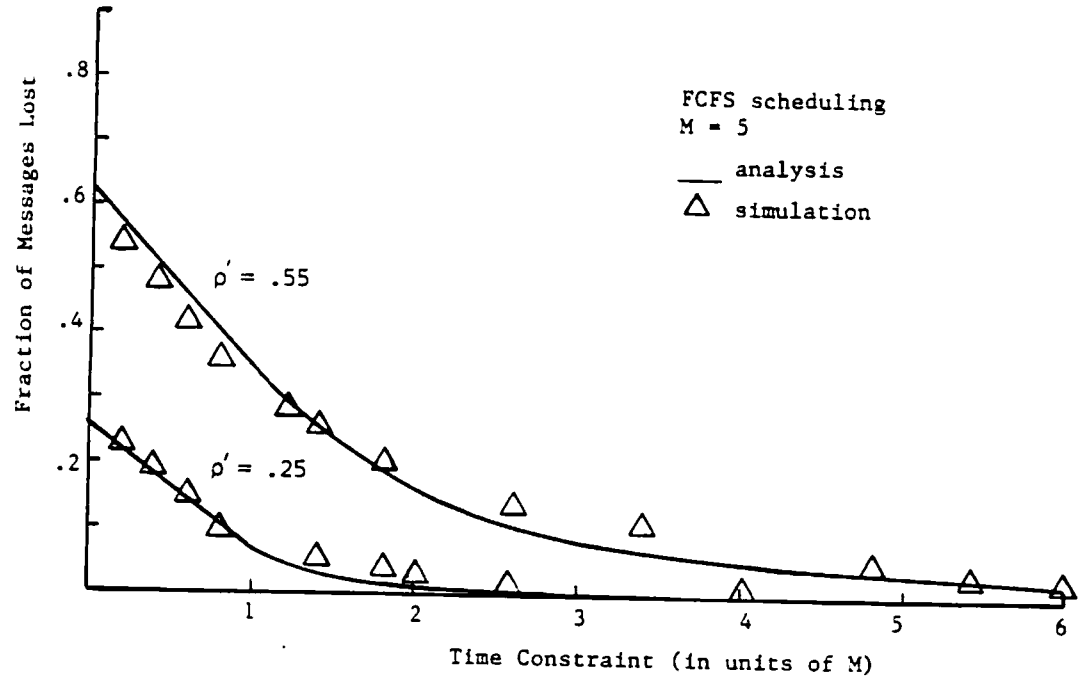
Figure 3-4 shows the message loss as a function of the imposed time constraint for the case in which the time window protocol provides FCFS network-wide scheduling; the sum in equation 3.9 was terminated at k=9 to obtain these values. Loss percentages are shown for 3 different message generation rates and for message lengths 5 and 25 times the end-to-end channel propagation delay. The degree to which the analytic results in figure 3-4 agree with the simulation values indicate that good approximations were introduced to obtain the analytic form of $w_{fcfs}(t)$. The results in figure 3-4 will be further discussed in section 3.4.

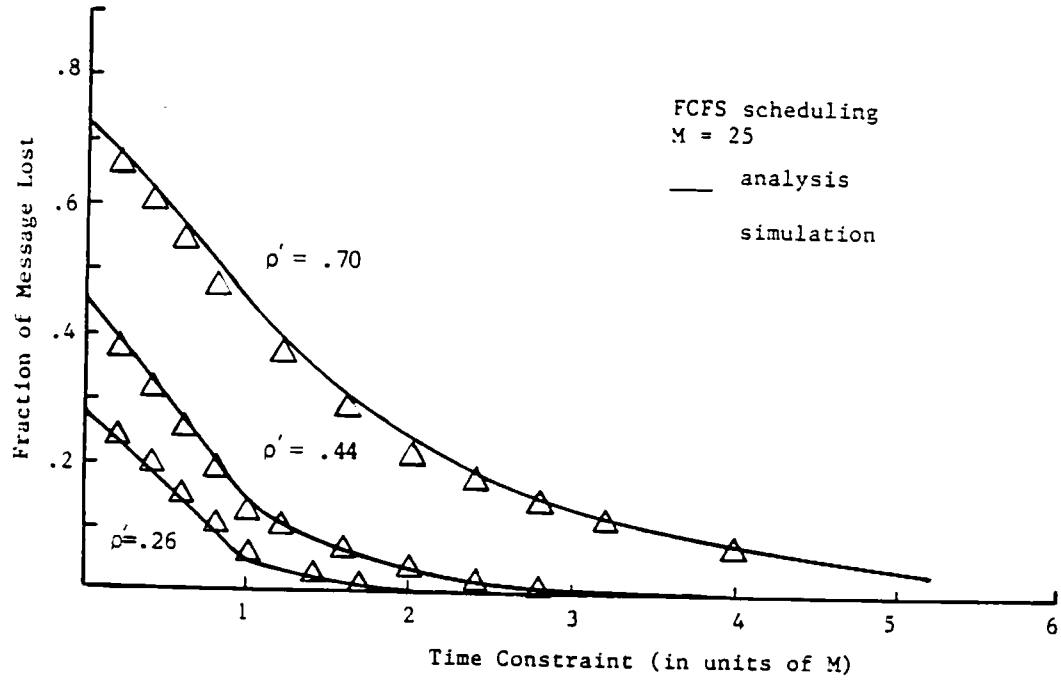
3.3.3. Distribution for LCFS Scheduling

In order to compute the waiting time distribution under LCFS scheduling, the *entire waiting time* of a message can be modeled as a series of waiting time components as shown in figure 3-5. (These *waiting* time components should not be confused with *service* time components typically used in the "method of stages" approximation [Kobayashi 78] for *service* time distributions.)

A message has no waiting time (i.e., it is transmitted immediately after its generation at the sending station) with probability γ_1. With probability $1-\gamma_1$ a message has a first waiting time component with a distribution (probability density function) given by $d_1(t)$. A message which finishes the first waiting time component begins service (i.e., begins transmission) with probability γ_2 and requires a second waiting time component (with a distribution $d_2(t)$) with probability $1-\gamma_2$. In general, γ_i represents the probability, given that a message

Figure 3-4: Message loss as a function of the imposed time constraint for FCFS scheduling





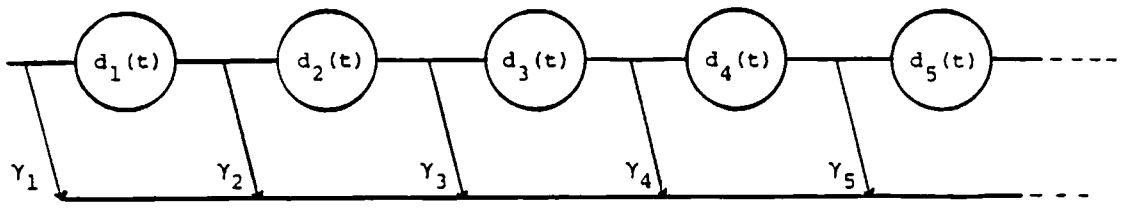


Figure 3-5: Waiting time components for LCFS scheduling

has completed i-1 waiting time components, that it will begin service after component i-1. Note that in order for a message to have exactly i waiting time components, it must *not* enter service after completing each of the first i-1 waiting time components and must enter service after the ith component. Thus the probability that a message has exactly i waiting time components is given by: $(1-\gamma_1)(1-\gamma_2)\cdots(1-\gamma_i)\gamma_{i+1}$. The distribution of the length of the ith component is given by $d_i(t)$.

If a message experiences any waiting time at all, the first component of its waiting time, $d_1(t)$, results from the residual service of another message already in service when the message is generated; $d_1(t)$ is thus given by $\beta(t)$, the previously determined residual service time density function. Since the scheduling discipline is LCFS, the remaining components of a message's waiting time result from other messages which are generated after, but transmitted before the message; thus $d_i(t)$ is given by $b(t)$ for $i>1$.

Using the above model of the waiting time, the distribution of the waiting time is given by:

$$w_{lcfs}(t) = \gamma_1\mu_0(t) + (1-\gamma_1)\gamma_2\beta(t) + \sum_{i=2}^{\infty} (1-\gamma_1)\cdots(1-\gamma_i)\gamma_{i+1}\beta(t)\odot b^{i-2}(t) \qquad (3.12)$$

where μ_0 and $\beta(t)$ are as previously defined and $b^{i-2}(t)$ is the (i-2)fold convolution of the service time density function. We must still compute the unknown values, $\{\gamma_i\}$. To do this, we define:

q_0 - the probability that no messages are waiting to be sent (queue is empty) at equilibrium.

$\bar{p}_i$ - the probability of i message generations during a residual service time.

p_i' - the probability of i message generations during a single message's service time.

P_j^k - the probability that j messages were generated during the first k components of the message waiting time given that the kth component of the message waiting time has just ended.

Q_j^k - the probability that j messages were generated during the first k-1 components of the message waiting time given that the kth component of the message waiting time has just begun.

Clearly, $\gamma_1 = q_0$ and $\gamma_2 = \bar{p}_0$. The value of q_0 can be determined from an analysis for the number in the queue as in [Lam 80]. Since the ith waiting time component is the last component if and only if exactly i-1 messages were generated during the first i waiting time components, the remaining values for $\{\gamma_i\}$ are given by $\gamma_{i+1} = P_{i-1}^i$

In order to compute the values of $\{\gamma_i\}$, we must thus compute the sets of probabilities $\{P_j^k\}$ and $\{Q_j^{k+1}\}$ for all $j,k \geq 1$; the relationship between these two sets of probabilities is shown in figure 3-6. These probabilities can be iteratively computed as follows, using the initial condition $P_j^1 = \bar{p}_j$.

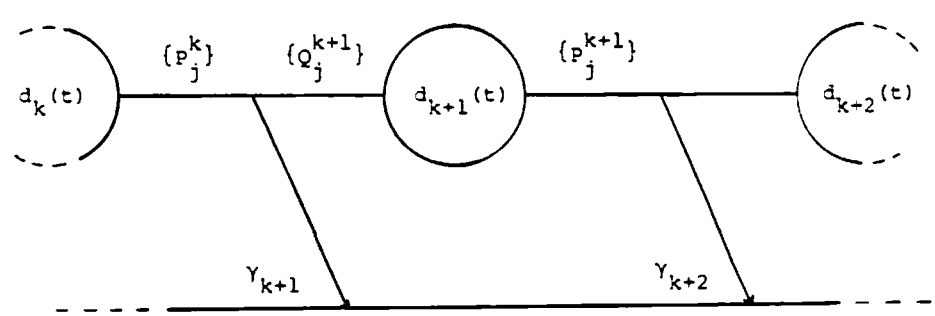


Figure 3-6: Probability sets for LCFS waiting time distribution calculation

Let us first consider the values of $\{Q_j^{k+1}\}$, the probabilities that j message generations occurred during the first k waiting time components given that the k+1st waiting time component has begun, for the cases in which $j<k$. Note that if a message begins the k+1st waiting time component, there must have been at least k messages generated during the previous waiting time components, since otherwise the message would have previously begun service. Thus, we have $Q_j^{k+1}=0$ for all $j<k$. (Note that by the same argument we also have $P_j^k=0$ for all $j<k-1$.)

Let us now consider the values of $\{Q_j^{k+1}\}$ for $j\geq k$. The value of Q_j^{k+1} (the probability of j message generations during the first k waiting time components given that the k+1st waiting time component has begun, and thus given that the kth waiting time component was completed) can be related to P_j^k (the probability of j message generations during the first k waiting time components given that the kth waiting time component was completed) by a simple conditioning of probabilities:

$$Q_j^{k+1} = \frac{P_j^k}{\text{probability}\{\text{the k+1st waiting time component is begun given that the kth waiting time component is completed}\}}$$

The probability that the k+1st waiting time component is *not* begun given that the kth waiting time component is completed is given by P_{k-1}^k. Thus, the probability in the denominator above is simply $1 - P_{k-1}^k$ and we have:

$$Q_j^{k+1} = \begin{cases} 0 & 0<j<k \\ P_j^k/(1 - P_{k-1}^k) & j\geq k \end{cases} \qquad (3.13)$$

Finally, given the values for Q_j^{k+1} for a fixed k and all j, the values of P_j^{k+1} can now be computed by conditioning on the number of messages generated during the k+1st waiting time component. If there are n messages generated during the k+1st waiting time component and j messages generated during *all* of the first k+1 components, j-n messages must have been generated during the first k waiting time components. Thus:

$$P_j^{k+1} = \sum_{n=0}^{j} Q_{j-n}^{k+1} p_n' \qquad (3.14)$$

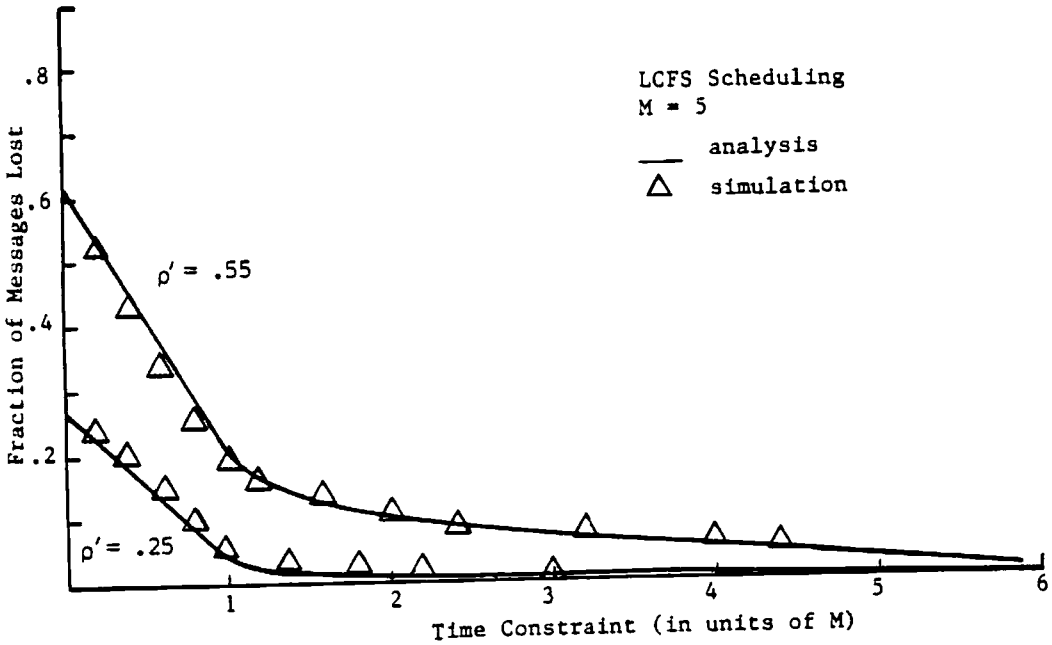
Equations 3.13 and 3.14 together provide for the iterative calculation of $\{P_j^k\}$ and $\{Q_j^{k+1}\}$ for all j,k. These values, in turn, are then used in the calculation of $\{\gamma_i\}$ and equation 3.12 can then be used to compute the message loss as a function of the imposed time constraint. Figure 3-7 shows this message loss. Performance results are shown for 3 different message generation rates and for message lengths equal to 5 and 25 times the end-to-end propagation delay of the channel. Note that all the loss curves in figure 3-7 fall rapidly with an increasing time constraint for time constraints less than the message length. This results from the fact that under LCFS scheduling, the message with the smallest delay is transmitted first and hence a large percentage of messages are transmitted immediately after termination of the message transmission (if any) that was in service when the message was generated. Note, however, that if an additional message is generated before a waiting message can begin service, this newly generated message (and possibly subsequently generated messages) will receive service first under the LCFS policy.

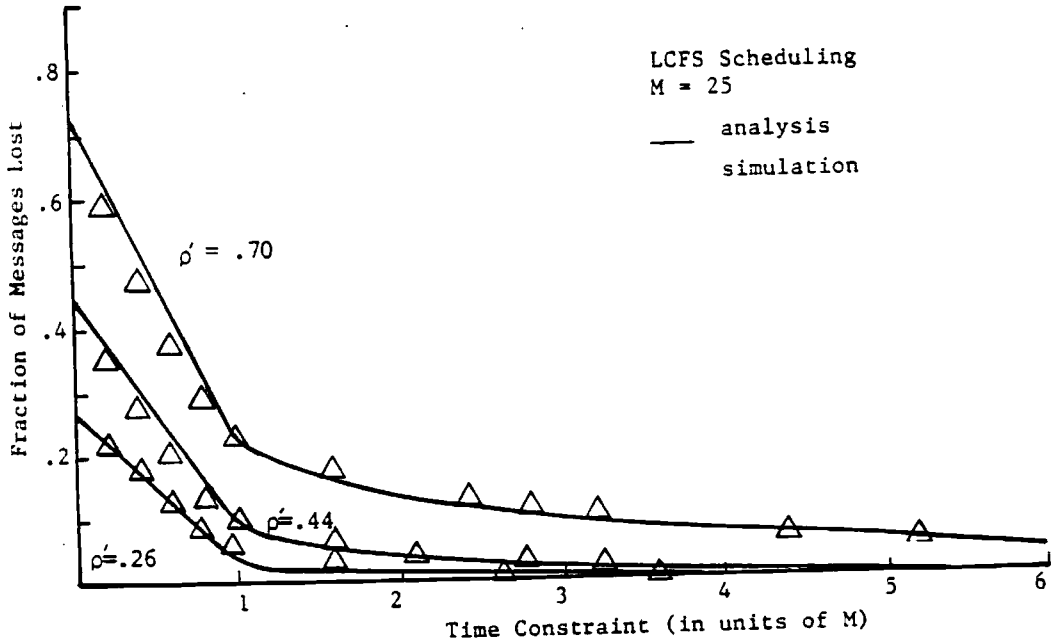
3.3.4. Distribution for Random Scheduling

The waiting time distribution under Random scheduling can be approximated using waiting time components. Once again, the first component of the waiting time results from the residual service time for another message (if any) in service when the message is first generated at a station. Also, the remaining components of the waiting time again result from the service times of other messages. However, since the message scheduling discipline is Random, these other messages may have been generated at any time.

The value for γ_0 will be exactly the same as under LCFS since the probability that a generated message finds the queue empty is independent of the order in which messages are selected for service [Kleinrock 76]. In order to determine the remaining values for $\{\gamma_i\}$, we can make use of the average number of messages in the distributed queue given there is at least one message in the queue (i.e.,

Figure 3-7: Message loss as a function of the imposed time constraint for LCFS scheduling





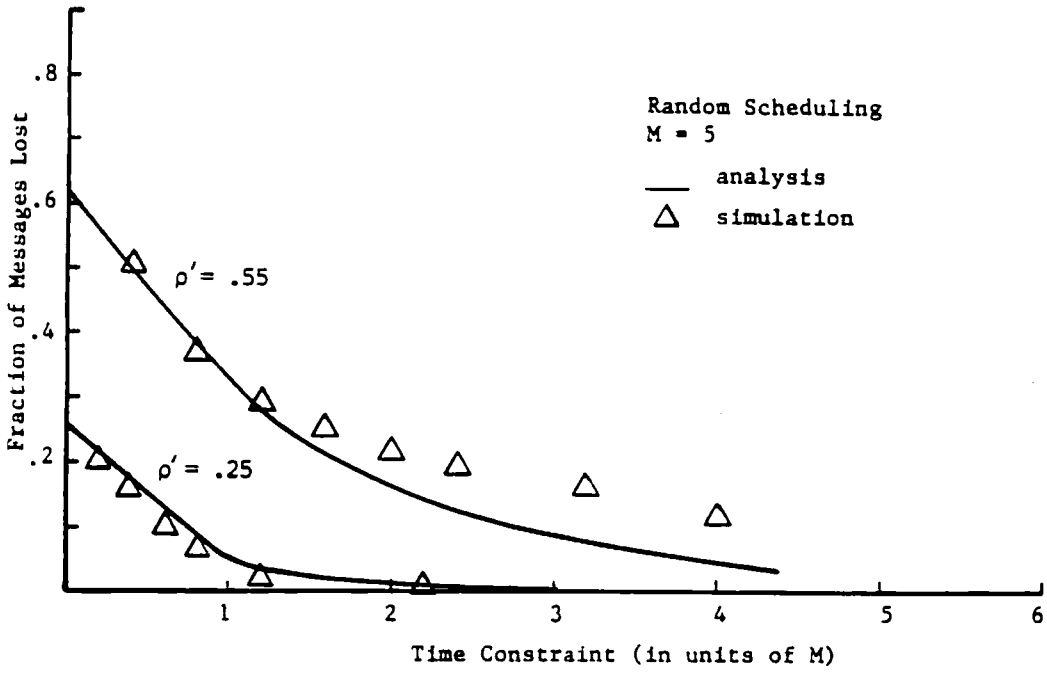
the message for which the waiting time distribution is being computed). This conditional value is given by $\bar{q}/(1-q_0)$, where $\bar{q}$ is the unconditional average number of customers in the queue. Since Random scheduling implies that all messages in the queue are equally likely to begin service next, if there are on the average, $\bar{q}/(1-q_0)$ messages in the queue, one way to *model* the probability that a particular message begins service immediately after waiting time component i is:

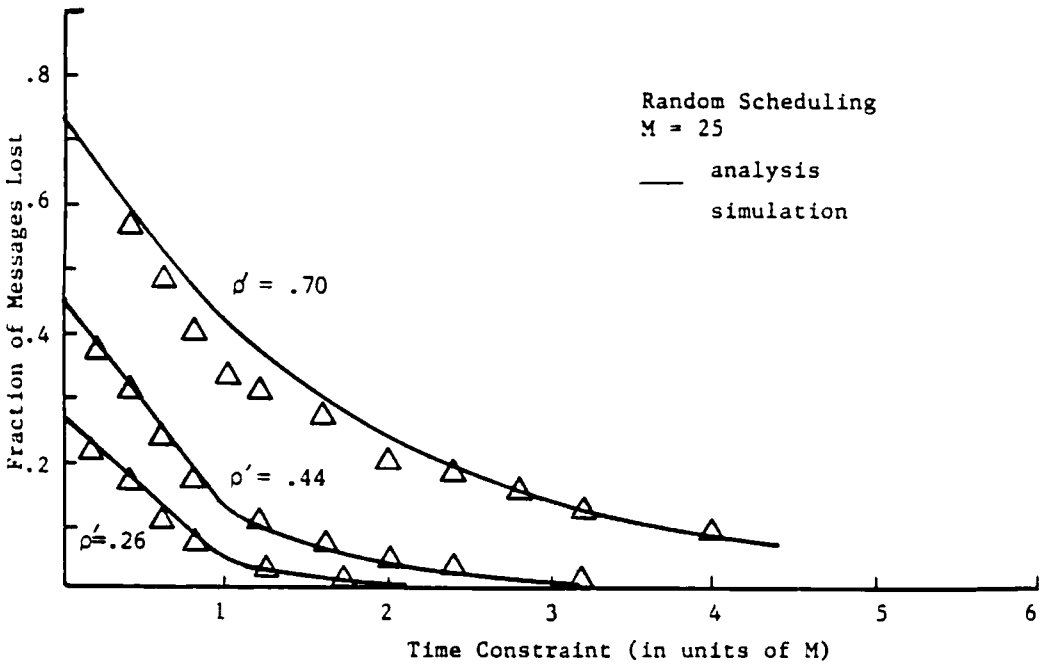
$$\gamma_i = \frac{1}{\frac{\bar{q}}{1 - q_0}} \quad \text{or} \quad \gamma_i = \frac{1 - q_0}{\bar{q}} \tag{3.15}$$

$w_{Random}(t)$, the waiting time density function under Random scheduling can be computed using these values of $\{\gamma_i\}$ in equation 3.12. The message loss as a function of the imposed time constraint can then be computed as in equation 3.11. Figure 3-8 shows computed message loss curves for Random scheduling for three different message generation rates and message sizes equal to 5 and 25 times the end-to-end propagation delay of the channel.

As shown in figure 3-8, the simulation results for large values of ρ' are not in as close agreement with the analytic results as the results for FCFS and LCFS scheduling were. Recall that in these latter two cases, the analytic results were computed exactly; the waiting time distribution for Random scheduling was only *approximated*. In equation 3.15 we used only two values, $\bar{q}$ and q_0, to approximate all of the γ_i. Also, we chose the values of γ_i such that the probability that a message begins transmission after the ith waiting time component is the same for all $i>1$. For small values of ρ', a message's wait will typically consist of only a few waiting time components and this appears to be a reasonable approximation. For large values of ρ', however, a message's waiting time will typically consist of many components and our approximation for $\{\gamma_i\}$ does not adequately capture the effect of these waiting time components.

Figure 3-8: Message loss as a function of the imposed time constraint for RANDOM scheduling





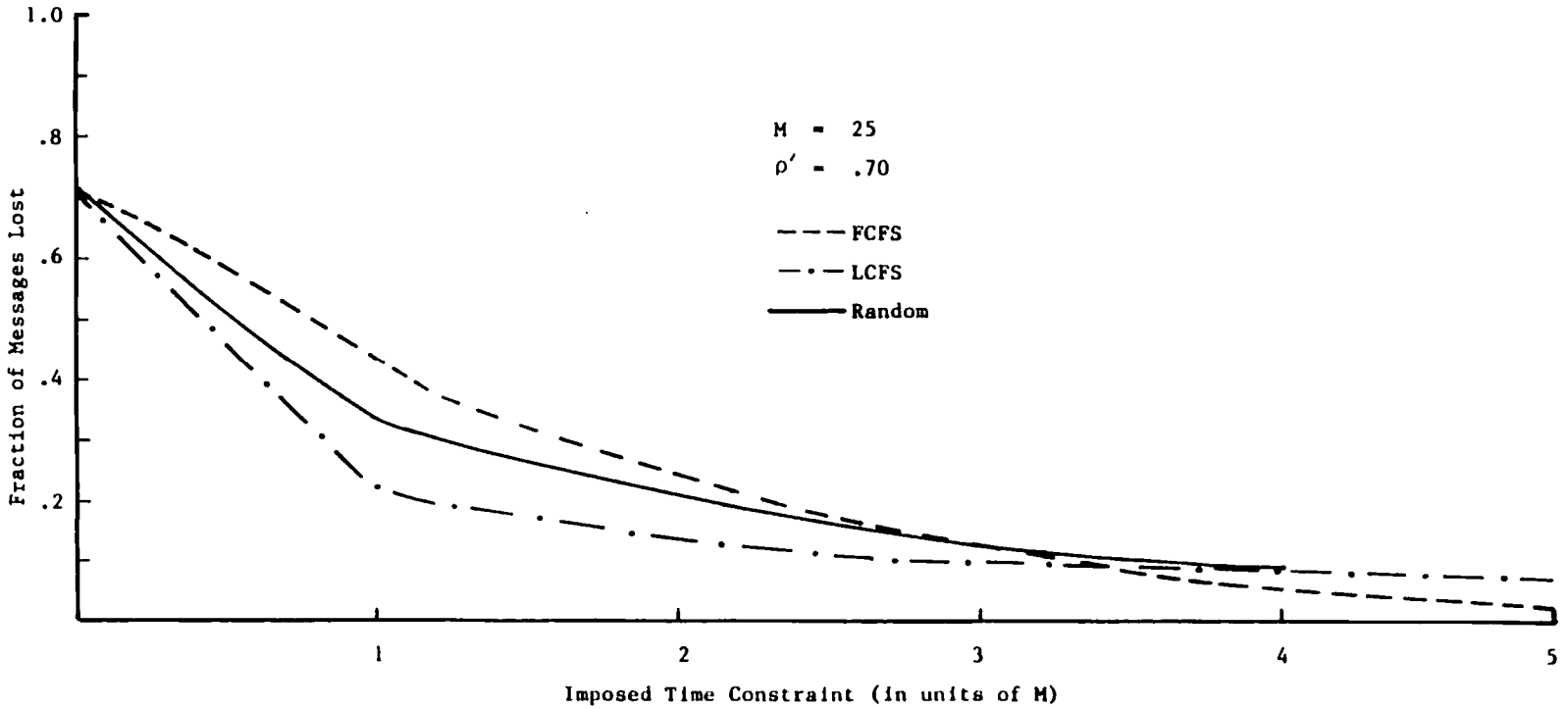
3.4. The Impact of Scheduling Disciplines on the Time-Constrained Performance of Random Access Protocols

In figure 3-9 we compare the loss curves for a fixed message size and message generation rate for FCFS, LCFS and Random scheduling in order to provide a quantitative example of the impact that a scheduling discipline can have on the time-constrained performance of the time window protocol.

These results indicate that for the given message generation rate and message size, none of the three scheduling disciplines is uniformly the best in the sense of minimizing loss for all possible time constraints. For small time constraints (large loss), LCFS is better than FCFS and Random, while for large time constraints (small loss), FCFS is better than LCFS and Random. Similar results can be found by comparing other message loss curves from figures 3-4, 3-7 and 3-8. The results in figure 3-9 also indicate that there can be significant performance differences due to the imposed scheduling discipline. For example, for the same fixed time bound, the message loss for FCFS and LCFS can differ by as much as 100%; for the same message loss, the time bounds required by FCFS and LCFS to realize this loss can also differ by as much as 100%. Clearly, the manner in which the initial windows and halves of split windows are chosen can have a great affect on the time-constrained performance of the time window protocol.

Although FCFS and LCFS each perform better than the other (and Random) for certain values of the imposed time constraint or tolerable message loss, the question arises whether other scheduling disciplines exist which perform better than both FCFS and LCFS in such regions. Since we are interested in maximizing the probability that a message has a waiting time below some given bound, a scheduling discipline similar to minimum slack time scheduling in deterministic scheduling [Coffman 76] would seem promising. Under minimum slack time scheduling, that message with a current waiting time closest to, but not exceeding, the time constraint is transmitted next. This message can be selected by choosing the beginning of the initial time window to be either the current time minus the time constraint or the current value of t_past

Figure 3-9: Comparison of loss as a function of the imposed time constraint for FCFS, LCFS and RANDOM scheduling

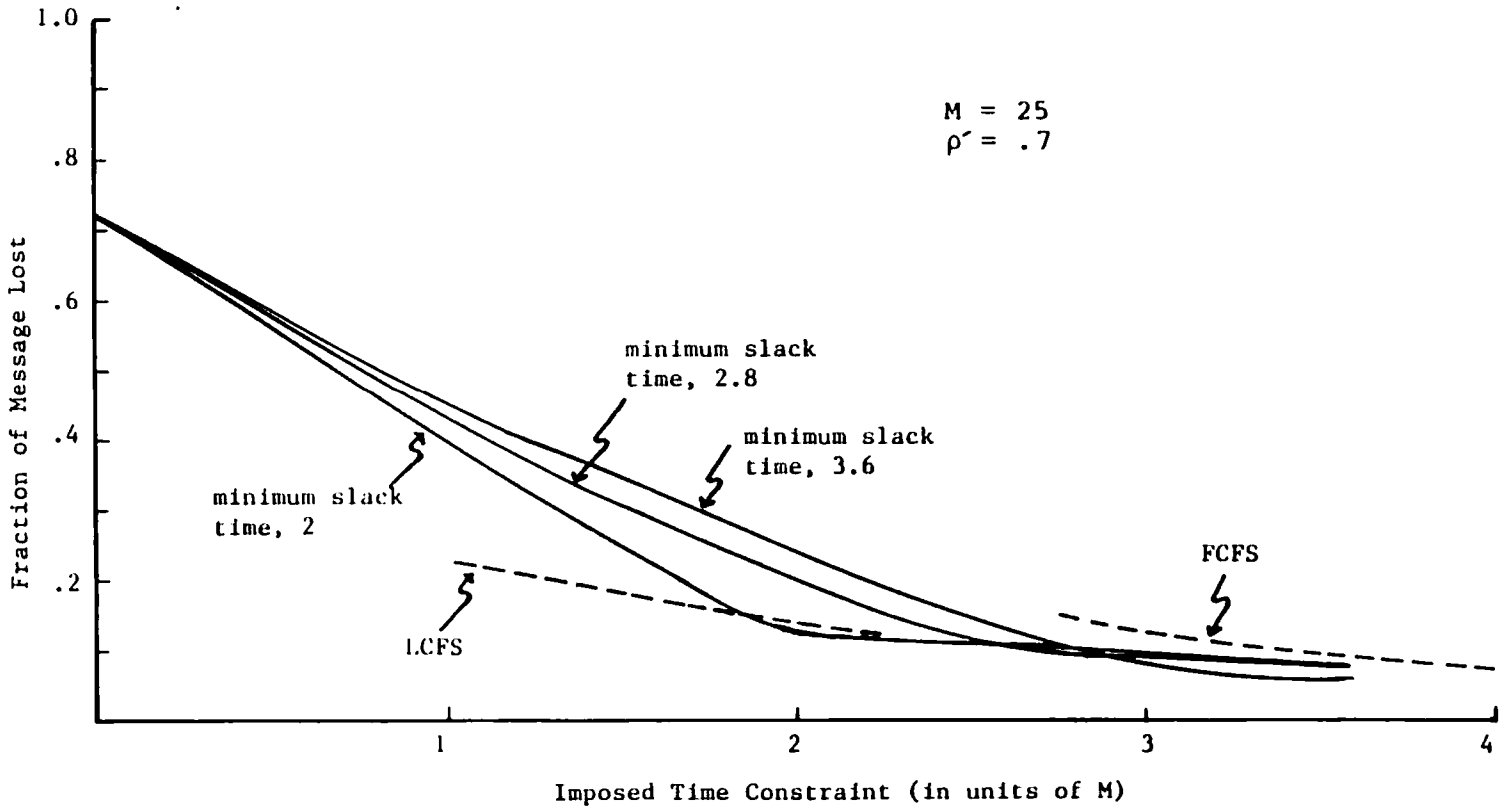


(whichever is more recent) and resolving collisions within a window on a FCFS basis. Note that since the window protocol only maintains modified message generation times, an exact realization of minimum slack time scheduling is not possible. However, since a message's actual delay (the amount of time between a message's actual generation time and the current time) is always greater than its modified delay (measured using its modified generation time), the minimum slack time scheduling algorithm does insure that messages with a modified delay (and hence actual delay) known to exceed the imposed time constraint are not transmitted before messages with a modified delay below the time constraint.

Figure 3-10 shows simulation values for minimum slack time scheduling for time constraints of 2.0, 2.8 and 3.6 times the message transmission time. From our simulation studies and as evidenced in figure 3-10, we have noted that minimum slack time scheduling for a specific time bound performs equally as well or better than both FCFS and LCFS in the region of the imposed time constraint. It should be noted that although the absolute decrease in message loss under minimum slack time scheduling is relatively small, the percentage decrease in the loss may actually be quite significant.

Finally, it should be noted that all the various scheduling disciplines discussed so far are implemented by the same general window mechanism. In practice, system characteristics such as the message generation rate, loss tolerances and the imposed time constraint may vary over time. Since the relative time-constrained performance of the different scheduling disciplines depends strongly on these variable system characteristics, a feature of the general window mechanism which makes it particularly attractive for time-constrained applications is the possibility that the single window mechanism could be used to dynamically impose different scheduling disciplines in response to the changing system characteristics. The performance of such an adaptive scheme, however, would crucially depend on the ability of all stations to accurately estimate the values for changing system parameters and the scheme itself might require additional synchronization among the stations to coordinate adaption. The development of an adaptive scheduling scheme is thus a challenging problem for further research.

Figure 3-10: Comparison of loss as a function of the imposed time constraint for FCFS, LCFS and minimum slack time scheduling



3.5. Summary

In this chapter, we have seen that although a multiple access protocol has traditionally been viewed simply as a distributed resource sharing mechanism, it also serves a very important additional role as a distributed message transmission scheduling mechanism. Given the importance of this role, we developed a channel access protocol, based on a generalization of the time window mechanism, which is suitable for supporting time-constrained communication applications in a multiple access environment. This protocol can provide any of a large class of distributed network-wide message transmission scheduling disciplines based on message generation times.

Both simulation and novel analytic and numerical models were developed to study the effects of the imposed scheduling discipline on the time-constrained performance of the time window protocol. We examined cases in which the protocol provides FCFS, LCFS and Random scheduling. In addition, a protocol which specifically attempts to maximize the percentage of messages with a delay less than a given time constraint was also introduced and examined. The performance results quantitatively demonstrated that the scheduling role of the protocol does indeed have a dramatic impact on its performance for time-constrained applications.

Chapter 4
Controlling Time Window Protocols for Time-Constrained Communication

4.1. Introduction

In the previous chapter, the importance of the scheduling discipline imposed on message transmissions by an access protocol was examined and found to have a critical impact on the protocol's performance for time-constrained communication applications. A protocol which provides explicit control over this scheduling function was developed and its performance examined for cases in which several different message scheduling disciplines were imposed. We found that among FCFS, LCFS and Random scheduling disciplines, none was uniformly the best for all values of an imposed time constraint. An additional scheduling discipline was thus proposed which implemented a minimum slack time scheduling discipline based on the *modified* generation times of messages. The performance of this scheduling discipline was studied through simulation.

The research presented in this chapter is motivated by the observation that the time window protocol, as previously described, is *passive* in the sense that sending stations eventually transmit *every* message, regardless of the amount of time a message may have spent waiting for transmission. In this chapter we introduce the additional capability of explicitly discarding (losing) messages at the sending stations. The advantages of losing messages at the sending stations (as opposed to the receiving stations) are twofold. First, resources need never be wasted on transmitting a message which would be lost with certainty at the receiving station. Secondly, in heavy traffic situations, large message delays (and correspondingly large message loss) resulting from a temporary overload need not be propagated into the future.

In this chapter we also formally address the problem of determining the optimal elements of the scheduling (windowing) policy. In the following section, we first present several extensions to the operation of the time window protocol. We then formulate a policy for controlling the operation of the time window protocol when the additional capability of explicitly discarding messages at the sending station has been introduced. A semi-Markov decision model [Howard 71] is developed for the operation of the sending stations and it is proven that certain temporally local optimum decisions with respect to message loss also characterize optimal long term (infinite horizon) behavior. Three of the four optimal elements of the windowing policy are determined within this decision model and are shown to be both intuitive and simple. A heuristic is presented for the final policy element.

Although the semi-Markov decision model can also be used to obtain analytic performance results, the procedure is too computationally expensive to be of practical use. Thus, an alternate performance model based on a general queueing system with impatient customers is developed in section 4.4. For cases in which message loss is the primary performance metric, our model is considerably simpler than previously developed related queueing models. This model is then used to examine the time-constrained performance of the time window protocol for the case in which the optimal elements of the windowing policy are used; simulation results are also presented to corroborate the analytic results. The results quantitatively demonstrate that significant performance improvements can be realized over the cases in which the sending stations provide FCFS, LCFS, Random or minimum slack time scheduling.

4.2. A Modification to the Time Window Protocol

4.2.1. Modified Message Generation Times versus Actual Message Generation Times

In the time window protocol described in section 3.1, an interval of time known to contain either zero or a single message generation is removed from future consideration by the protocol as soon as the message (if any) with a modified

generation time falling within this time interval has been transmitted. This is accomplished by updating the appropriate modified message generation times and the value of t_past as previously shown in figure 3-1. The notion of modified message generation times was introduced in the time window protocol presented in chapter 3 in order to reduce the amount of state information maintained by the protocol. An alternative to using modified message generation times and maintaining the single state variable, t_past, is to use the actual message generation times and to maintain state information which records every interval of time in the past known to contain no unsent message generations. We will refer to such an interval of time as a *fully probed* interval of time. That is, an interval of time is considered to be *fully probed* if, and only if, it is known that every message that was generated during the interval has already been transmitted.

Note that the simple state information used by the time window protocol presented in chapter 3 requires scheduling decisions to be made on the basis of the modified generation time of messages rather than their actual generation time. A message is lost, however, when its actual waiting time, determined by its *actual* generation time, exceeds the imposed time constraint. Thus, although the time window protocol described in chapter 3 requires only a minimal amount of state information (i.e., the stations need only maintain a single value, t_past, and the modified generation times of locally generated messages), the cost of limiting the amount of state information is that only *partial* information about the actual generation times of messages is used by the protocol. Modified generation times can be used to distinguish messages according to the temporal ordering of their message generation times but *not* according to the actual times at which the messages were generated.

In the 'the time window protocol described below, the use of modified message generation times has thus been discarded in favor of the use of the actual generation times of messages. In this case, every fully probed interval of time must now be included in the protocol's state information. As we will see, however, in the case that the *optimal* elements of the windowing policy are employed, only a single piece of state information (similar to t_past) need be maintained.

The operation of the time window protocol using actual message generation times is shown in figure 4-1 and is essentially the same as that of the protocol presented in chapter 3. Once again, stations synchronously select initial windows of time according to some policy (to be described below) and a station transmits a message if, and only if, it has a message to send which was actually generated during this interval of time.

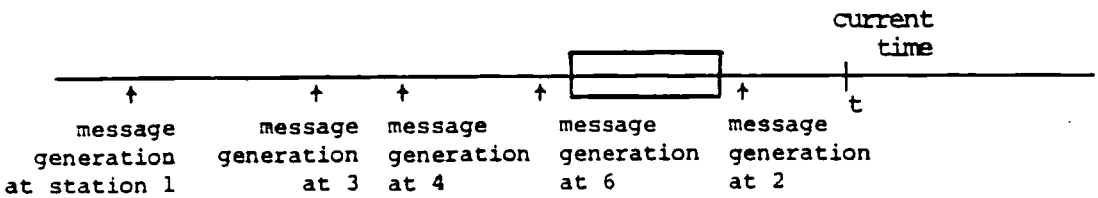


fig. 4-1a: the initial window contains no message generations

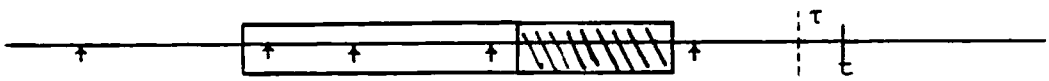


fig 4-1b: a new window is chosen and collisions occur

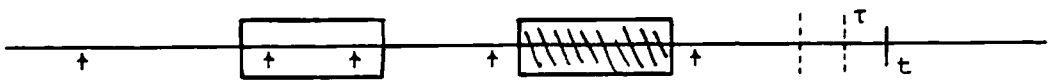


fig. 4-1c: window is split in half and another collision occurs

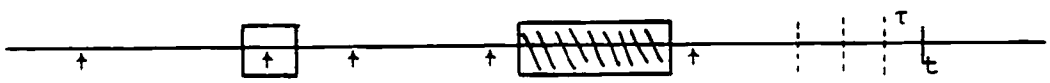


fig. 4-1d: window is split again and station 3 begins successful transmission of its message

Figure 4-1: Operation of the time window protocol

Since the notion of modified generation times has been discarded, the protocol must now record every fully probed interval of time. For example, if an initial

time window is chosen as in figure 4-1a and no messages were generated during this time window, this fact must be incorporated into the protocol's state information. In figure 4-1, such fully probed intervals of time are indicated by shaded regions. Thus, the stations might view the time axis as shown below in figure 4-2.

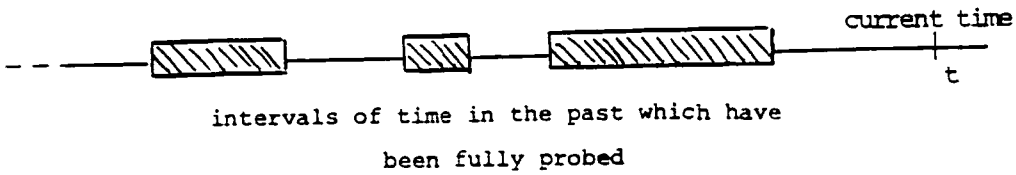


Figure 4-2: A station's view of the time axis

4.2.2. A Policy for Controlling the Windowing Process

Note that the opportunity for controlling protocol operation arises only at those times when an initial window must be selected. Assuming the protocol maintains some state information (such as a record of its past history as in the time axis in figure 4-2), then a selection of

1. the position of the initial window
2. the length of the initial window
3. a procedure for splitting the window should collisions occur

must be made, based on the current state information, from some set of alternatives. Once an alternative for operation has been selected, the probabilistic evolution of the protocol is determined until the next time an initial window must be chosen. The selection of an alternative for action (i.e., specifying (1) through (3) above) given the current state is known as a *decision* and the set of all decisions is known as a *policy.*

The problem we now want to address is how to select those alternatives for

operation that will maximize the percentage of messages with delays below the given time constraint. A message's delay (or waiting time) will once again be defined as the amount of time between its generation at the sending station and the beginning of the windowing process immediately preceding, and resulting in, its own successful transmission. Thus, a message's delay does not (by definition) include the time required for the single windowing process resulting in its transmission. However, the delay *does* include the windowing time for all other messages transmitted since it was generated; this point will be further discussed in section 4.4.

4.3. Controlling the Time Window Protocol

In this section we address the problem of selecting policy elements (1) - (3) in order to maximize the percentage of messages with a delay below a given bound. First, we develop a state space representation suitable to encode the necessary past history of the protocol. This state space description is then used as the basis for a semi-Markov decision model [Howard 71] of protocol operation. The usual approach for finding the optimal elements of a policy is to choose some initial policy elements and then to iteratively obtain better and better policy elements. Unfortunately this iterative process can be computationally quite expensive and moreover may provide little insight into the operation of the protocol itself. Our approach here will be to use our understanding of the protocol to infer elements (1) and (3) of the optimal policy and then to prove that no policy iteration would yield a better policy. A simple closed form characterization of (2) does not appear possible; this problem will be further discussed in section 4.4.

4.3.1. A State Space Description and Pseudo Time

Let us assume that time is discrete in units of Δ (where Δ can be arbitrarily small but finite) and is small enough so that the probability of more than one message generation (anywhere in the network) in time Δ can be assumed to be zero. The most straightforward state space approach is simply to encode for each Δ unit of time in the past, whether or not it has been fully probed. This

approach leads to a complicated state space which grows exponentially in size with each Δ.

An alternative approach is to introduce the notion of *pseudo time*, determine the optimal policy elements (1) and (3) within a state space based on pseudo time and then to relate these results back to actual time. The relationship between actual time and pseudo time is shown below in figure 4-3. Pseudo time is defined such that each unit of pseudo time in the past is associated with a unit of actual time in the past which has not yet been fully probed. With no loss of generality, we can require that if t_1 precedes t_2 in actual time then the pseudo time associated with t_1 precedes the pseudo time associated with t_2. In this case, the introduction of pseudo time essentially compresses the actual time axis by removing fully probed intervals of time. Thus, the notion of pseudo time is closely related to the notion of modified message generation times introduced in the previous chapter.

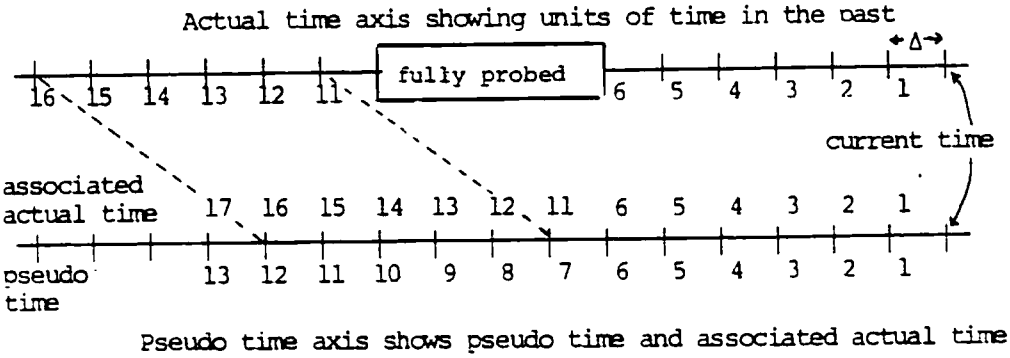


Figure 4-3: Actual time and pseudo time

Each state in the pseudo time state space will simply correspond to the total amount of actual time which has not yet been fully probed. One possible pseudo time state space description is thus:

$$S = \{ 0, 1, 2, 3 \ldots \} \tag{4.1}$$

where a particular state, i, indicates that i units of time have not yet been fully probed.

This state space description can be further refined as follows. Let K (in units of Δ) represent the imposed time constraint. Clearly, transmitting a message with a delay greater than K (in actual time) is useless work since the message will be lost at the receiver with probability 1. Thus the protocol should never send a message with a delay greater than K. When an initial window is chosen, all messages with a delay greater than K can therefore be discarded *by the sending station*. Thus, in addition to policy elements (1), (2) and (3) of section 4.2, the optimal policy will also:

4. discard any messages with a delay greater than K. These messages can be effectively discarded by marking the actual time intervals containing message generations which would have a delay greater than K as if these intervals of time had already been fully probed.

As a result of policy element (4), there are never more than K units of actual time which are not marked as fully probed. This bounds the size of the pseudo time state space and thus the pseudo time state space can be further refined to:

$$S = \{ 0, 1, 2, ... K\text{-}1, K\} \tag{4.2}$$

4.3.2. Optimal Elements of the Window Control Policy

In this section we establish the following theorem which states that the window control policy elements (1) and (3) which maximize the percentage of messages with delays less than K, result in successfully transmitted messages being sent on a global (network-wide) FCFS basis:

Theorem 4.1: In the case that all message lengths are identically distributed and given policy element (4), the optimal selection of policy elements (1) and (3) is independent of policy element (2) and can be characterized as follows:

1. The beginning of an initial window should be placed at the point in time closest to, but not exceeding, K units of time in the past (where K is the imposed time constraint) which is not marked as fully probed.

3. the older half of a split window is always selected first.

Note that a message is lost unless the windowing process which results in its successful transmission begins within K units of time after its generation. Intuitively, since all the message lengths are identically distributed, it would seem reasonable to transmit that message with a delay closest to, but not exceeding, the imposed time constraint; this is exactly what (1) and (3) above specify. This policy is similar to the minimum slack time scheduling policy examined in the previous chapter except that in the present case, messages are being selected for transmission based on their actual generation time rather than on the basis of their modified message generation time. Minimum slack time scheduling [Coffman 76], can be easily proven optimal in the deterministic case that all message generation times are known in advance. The present case is somewhat more complicated, however, since only probabilistic information concerning message generation times is known and since any decision influences the future evolution of the system.

In establishing Theorem 4.1, the following definitions will be useful:

- *actual delay* of a message. The amount of time between the current time and the time at which the message was generated at the sending station.

- *pseudo delay* of a message. The amount of time between the current time and the pseudo time associated with the time at which the message was generated. Note that while a message's actual delay always increases with time, the pseudo delay may both increase and decrease in time, depending on the behavior of the window mechanism.

- *actual loss*: fraction of messages not successfully transmitted with an actual delay less than K. Since the loss will be a function of the policy, we will write actual loss(P) to indicate the actual loss under policy P.

- *pseudo loss*: fraction of messages not successfully transmitted with a *pseudo delay* less than K. Pseudo loss(P) indicates the pseudo loss under policy P.

- *one-step pseudo loss*: the expected number of messages which have a pseudo delay less than K when a decision is made, but have a pseudo delay greater than K (and hence are lost under policy element 4) when the next decision is made.

In order to establish Theorem 4.1, we will first establish the following lemmas, which will require the following assumption:

Assumption 4.1 : The distribution of message generation times measured in actual time and in pseudo time is the same, i.e., the removal of intervals of time and the concommitant shifting of actual generation times (to get the pseudo generation times) preserves the distribution of inter-generation times. For the case that messages are generated according to a Poisson process, the removal of an initial window containing 0 or 1 message generations preserves this property exactly. In general, however, if one half of a split window is removed, an exponential inter-generation time distribution would not be exactly preserved.

Lemma 4.1 : For any policy P: actual loss(P) $\geq$ pseudo loss(P)

Proof: By the definition of pseudo time, the pseudo delay of a message is always less than or equal to the actual delay of a message. Thus if a message's pseudo delay exceeds K, its actual delay also exceeds K.

Lemma 4.2 : Given policy element (4), any policy with policy elements (1) and (3) as in Theorem 1 preserves the following property of all messages which are not lost:

pseudo delay of a message = actual delay of a message

Proof: Lemma 4.2 can be inductively established. Suppose the above property holds when a decision is made. If this property does not hold when the next decision is made, the selection of the first window must have been such that there was a message which was generated between K units of time in the past and the start of the first initial window. However, by hypothesis, the policy contains elements (1) and (3) as in Theorem 1 and thus no such message can exist. Thus the above property is preserved by each decision.

Lemma 4.3 : Let $\{P^w\}$ be the set of all policies which choose the same *size* window when in the same state (i.e., the set of all policies with the same second element). Let $P^w_{ms} \in \{P^w\}$ be the single policy with elements (1) and (3) as in Theorem 4.1. Then P^w_{ms} minimizes the one-step pseudo loss over all policies in $\{P^w\}$.

Proof: Let us define the *critical messages* associated with a decision as follows. When a decision is made at time t' in state i, a probabilistic amount of time, σ, is required for the windowing process and message transmission (if any). The critical messages associated with the decision made at t' are those messages with a pseudo generation time after $t'-i$ but before $t'+\sigma-K$. That is, a critical message is one with a pseudo delay less than i at t' which would have a pseudo delay greater than K at $t'+\sigma$ (when the next decision is made) if it is not transmitted. The one-step pseudo loss can thus be expressed by the following expected value:

$$\text{one-step pseudo loss} = E[\text{number of critical messages} - \text{prob. that a critical message is transmitted}] \quad (4.3)$$

By assumption 1, all units of pseudo time are statistically identical with respect to message generation times. Thus, for a given window size, the time between two consecutive decisions is independent of both the *position* of the first initial window and the procedure for choosing halves of a split window. Thus, for a given window size, the number of critical messages associated with a decision is independent of policy elements (1) and (3). Now, since P^{w}_{ms} always selects the message with a pseudo delay closest to, but not exceeding K (i.e., the message that will be critical if *any* messages are critical), P^{w}_{ms} maximizes the second term in equation 4.3. Thus P^{w}_{ms} minimizes the one-step pseudo loss.

Lemma 4.4 : Given policy element 4 in section 4.3.1, a policy which minimizes the one-step pseudo loss also minimizes the pseudo loss.

Lemma 4 relates the short term pseudo loss to the long term average pseudo loss and relies on results from decision theory. The proof of lemma 4.4 can be found in the appendix at the end of this chapter. Given lemmas 4.1 through 4.4 we can now establish theorem 4.1:

Proof of Theorem 4.1: Let $\{P^{w}\}$ be the set of all policies which have the same second policy element and let $P^{w}_{ms} \in \{P^{w}\}$ be the policy with elements (1) and (3) as in theorem 4.1. We want to show that no policy in $\{P^{w}\}$ has an actual loss less than P^{w}_{ms} . Suppose there exists some policy, P^{w}_{α} which has an actual loss less than that of policy P^{w}_{ms} :

$$\text{actual loss}(P^{w}_{\alpha}) < \text{actual loss}(P^{w}_{ms})$$

By lemma 4.2, we then have

$$\text{actual loss}(P^{w}_{\alpha}) < \text{actual loss}(P^{w}_{ms}) = \text{pseudo loss}(P^{w}_{ms})$$

and then by lemmas 4.3 and 4.4:

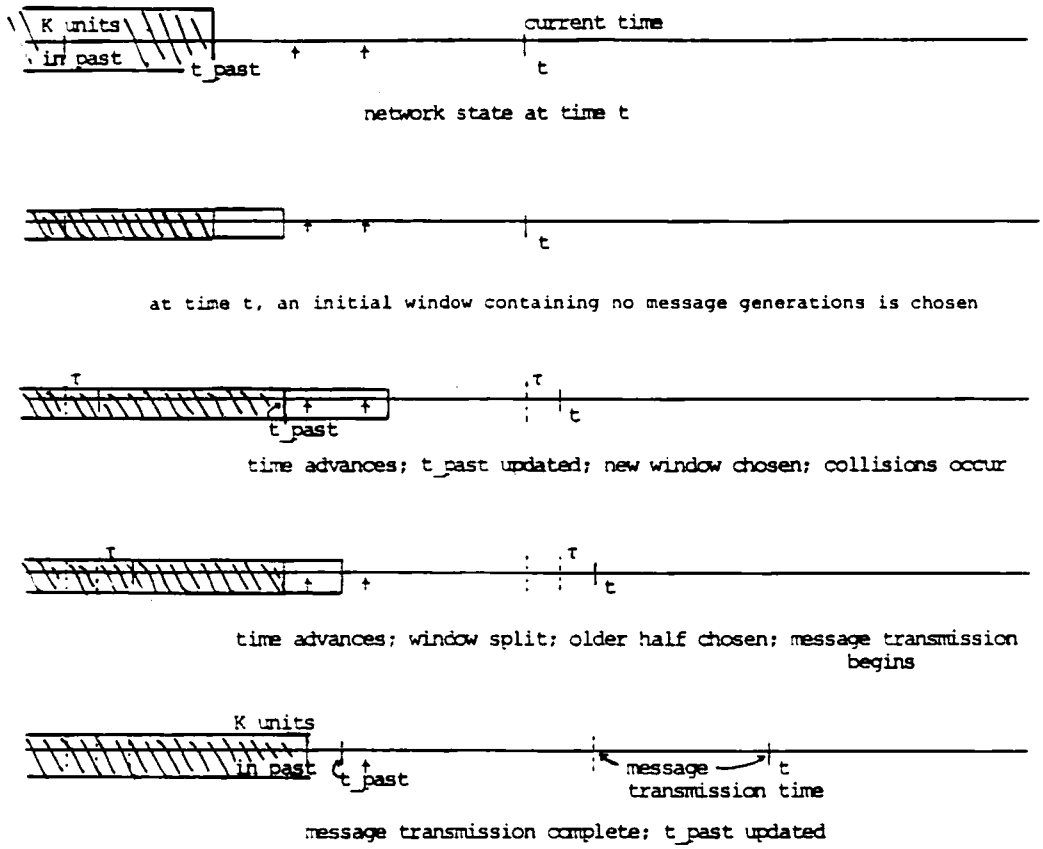
$$\text{actual loss}(P^{w}_{\alpha}) < \text{pseudo loss}(P^{w}_{ms}) = \min_{P' \epsilon (P^{w})} \text{pseudo loss}(P')$$

and thus specifically since $P^{w}_{\alpha} \epsilon \{P^{w}\}$:

$$\text{actual loss}(P^{w}_{\alpha}) < \text{pseudo loss}(P^{w}_{\alpha})$$

which contradicts lemma 4.1. Thus P^{w}_{α} cannot exist and thus P^{w}_{ms} is the optimal policy.

An important consequence of theorem 4.1 (stated in lemma 4.2) is that under optimal policy elements (1), (3) and (4), there is no difference between pseudo time and actual time. Thus, there are no "gaps" in time (the shaded regions in figure 4-2) between fully probed intervals of actual time. Thus the state space need not be a large and complicated encoding of numerous fully probed intervals of time. Rather, only a single piece of information need be maintained by the protocol - that point in time closest to, but not exceeding K units of time in the past which has not yet been fully probed and thus may contain untransmitted message generations. Once again, we will refer to this value as *t_past*. Under optimal policy elements (1), (3) and (4), it is further known that there may be unsent messages in the network which were generated at *any* point in time between t_past and the current time. Figure 4-4 provides an example of the operation of the protocol under optimal policy elements (1), (3) and (4) and the manner in which t_past is maintained.



4.4. A Queueing Model of Protocol Performance

The analysis developed in this section is again based on viewing the messages distributed among the stations throughout the network as customers in a *distributed* queue. Recall that in this model, the service time consists of two components: the scheduling time component and the actual transmission time component, where the scheduling time component of a message's service time is the time between either the end of the most recent message transmission or its own generation time (whichever is more recent) and the start of its own successful transmission.

4.4.1. An M/G/1 Queue With Customer Loss

A consequence of policy elements (1), (3) and (4) is that all successfully transmitted messages are sent on a FCFS basis. Furthermore, as a result of policy element (4), messages are lost at the sender (i.e., never sent) only when their waiting time (as defined in section 4.2) exceeds the given time constraint. Thus, the operation of the optimal policy can be modeled as a FCFS queue in which messages at the front of the distributed queue are lost (denied service) if their waiting time in the queue has exceeded the time constraint; this model is shown in figure 4-5a.

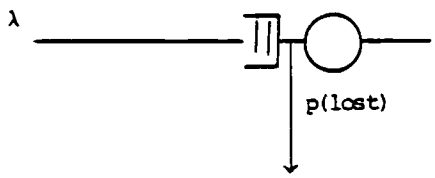


Fig. 4-5a: Customers are denied service if wait in queue > K.

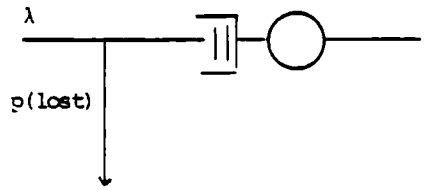


Fig. 4-5b: Customers determine waiting time and do not join the queue if wait time > K.

Figure 4-5: Two models of a queue with customer loss

In terms of server utilization, it makes no difference whether customers are lost as soon as they are generated and arrive at the front of the queue or whether they somehow determine their waiting time and join the queue if, and only if, their waiting time is less than the specified time constraint. Thus, the probability that the server is busy is the same in the model in figure 4-5a as in the model in figure 4-5b. We will model the distributed queue using the M/G/1 queueing system shown in figure 4-5b. It should be noted that this model is only approximate since the "service" times are not truly independent. This second queueing model was recently studied in [Baccelli and Hebuterne 81] for the M/G/1 case for the waiting time distribution of customers entering service. If we are only interested in the probability of message loss, an alternative approach can considerably simplify the analysis for the M/G/1 case.

Let us define F(w,t) as the probability distribution function (PDF) of the unfinished work in the queue at time t:

$$F(w,t) \triangleq P(\text{ unfinished work } < w)$$

Note that the unfinished work in the queue corresponds to the waiting time an arriving (newly generated) customer would experience under FCFS scheduling. By elementary continuity arguments, the PDF of the unfinished work at time $t+\Delta t$ can be characterized in terms of its value at time t as follows:

$$\begin{aligned} F(w,t+\Delta t) \;=\; & (1-\lambda\Delta t)F(w+\Delta t,t) \;+ \\ & \lambda\Delta t(1-\mu_1(w-K))\int_0^w B(w-x)\frac{\partial F(x,t)}{\partial x}dx \;+ \\ & \lambda\Delta t(\mu_1(w-K))\{\; c_1\int_0^w B(w-x)\frac{\partial F(x,t)}{\partial x}dx \;+\; (1-c_1)F(w+\Delta t,t)\;\} \end{aligned} \tag{4.4}$$

where K is the time constraint, B(x) is the service time PDF of a message (customer), $\mu_1(w-K)$ is the unit step function at K and $c_1 = F(K,t)$.

The first term on the right hand side of equation 4.4 relates the value of $F(w,t+\Delta t)$ to the value of the PDF at time t in the case that no messages are generated in Δt (i.e., no message arrivals to the queue). The second term is for the case that one message is generated and there is unfinished work at time $t+\Delta t$ less than or equal to K. The final term is for the case that one message is generated and the unfinished work in the queue at time $t+\Delta t$ is greater than K; the values of c_1 and $1-c_1$ represent, respectively, the probability that a generated message found its waiting time to be less than or greater than K. Rearranging the terms in equation 4.4, taking the limit as Δt approaches 0 and then (assuming equilibrium exists) taking the limit as t approaches infinity, results in the following pair of integro-differential equations for the distribution of unfinished work in the queue:

$$0 = \frac{dF(w)}{dw} - \lambda F(w) + \lambda \int_0^w B(w-x) d_x F(x) \qquad (0<w\leq K) \quad (4.5a)$$

$$0 = \frac{dF(w)}{dw} - \lambda c_1 F(w) + \lambda c_1 \int_0^w B(w-x) d_x F(x) \qquad (K < w) \quad (4.5b)$$

Let $f_{w<K}(w)$ be the derivative of the solution to equation 4.5 (i.e., the waiting time density function (pdf)) in the region $0<w\leq K$ and let $f_{K<w}(w)$ be the waiting time density function in the region $K<w$. Then by conservation of probability, we have:

$$\int_0^K f_{w<K}(w)dw \quad + \quad \int_K^\infty f_{K<w}(w)dw \quad = \quad 1 \qquad (4.6)$$

At this point, we could solve equation 4.5 for $f_{w<K}(w)$ and $f_{K<w}(w)$ using equation 4.6 to determine the unknown constants. The solution to equation 4.5a can be shown [Kleinrock 75] to be:

$$f_{w<K}(w) \quad = \quad P(0) \sum_{i=0}^{\infty} \rho^i \beta^i(w) \qquad (4.7)$$

where:

- P(0) is the probability that the server (transmission channel) is idle
- $\rho = \lambda\bar{x}$, where λ and $\bar{x}$ are, respectively, the message generation rate and average service times of messages
- $\beta(w)$ has the form of the residual service time distribution of an M/G/1 queue with no loss, i.e., the distribution of the remaining work that a newly generated message would find for a message in service.
- $\beta^i(w)$ is the i-fold convolution of $\beta(w)$.

The solution of equation 4.5b is considerably more complex and requires the numerical inversion of a difficult Laplace transform; fortunately, it need not be solved. Note that the second term on the left of equation 4.6 is simply the probability that a newly generated message finds a waiting time greater than K and thus does *not* join the queue and is lost:

$$\int_K^\infty f_{K<w}(w)dw \quad = \quad p(\text{loss}) \quad = \quad 1 - p(\text{accepted}) \tag{4.8}$$

The loss probability can be related to P(0), the probability that the server is idle, using the simple conservation of flow argument shown in figure 4-6. The average rate at which messages actually join the distributed queue is given by $\lambda p(\text{accept})$ and the average rate at which messages leave the queue after service is given by the probability that the server is busy times the rate at which the server services these customers. By conservation of flow, the average rate at which messages join and depart from the queue must be the same and thus:

$$p(\text{accept})\rho \quad = \quad 1 - P(0) \tag{4.9}$$

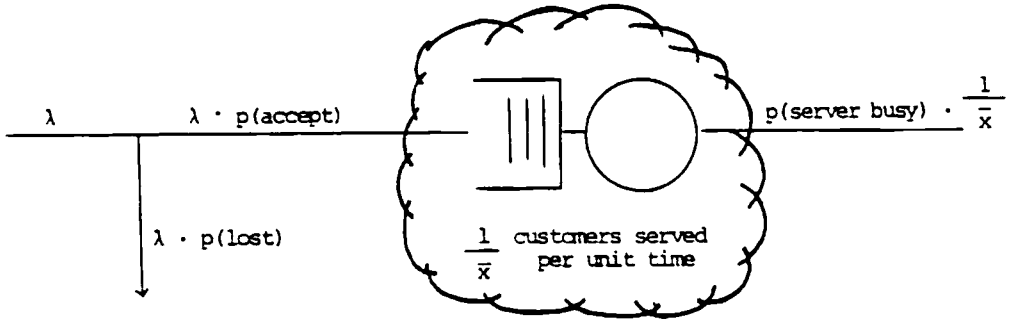


Figure 4-6: Flow conservation

Using equations 4.6 - 4.9 we can now determine the probability of message loss:

$$p(\text{loss}) \quad = \quad 1 \quad - \quad \rho^{-1} \quad + \quad \frac{1}{\rho + \rho^2 z(K,\rho)} \tag{4.10}$$

where $z(K,\rho) \quad = \quad \sum_{i=0}^{\infty} \rho^i \int_0^K \beta^i(w)dw$

As a check of equation 4.10, note that in the limit as K approaches infinity, p(loss) approaches 0. As K approaches 0, p(loss) approaches 1-P(0). That is, the probability that a message is lost approaches the probability that the server is busy, as would be expected since as K approaches 0, a message would only enter service if, and only if, the queue was empty and the server was idle.

Equation 4.10 thus gives the probability of message loss under optimal policy elements (1), (3) and (4). Note that we have not yet specified policy element (2) which determines the initial window length. The selection of policy element (2) will affect both $\overline{x}$ (and thus ρ) and $\beta(w)$ in equation 4.10. Unfortunately, computing the optimal value of policy element (2) would require solving the set of equations (A1) in the appendix and thus is computationally too expensive to be of practical use; similar problems have been encountered in other protocol models based on semi-Markov decision analysis [Lam and Kleinrock 75]. Thus, rather than compute the optimal values for (2), let us examine the performance of the protocol using a heuristic rule for (2).

Specifically, let us assume that (2) is chosen to minimize the *average* amount of time required by the windowing process to schedule a message under saturation conditions, as discussed in the previous chapter. Even if the window sizes are selected in this manner, determining the first moment and distribution of the message scheduling time is not an easy task. Recall that in the previous chapter, these values were approximated by exactly determining the average scheduling time for two message generation rates and fitting a function to these endpoints to approximate the average scheduling time for intermediate message generation rates. This average scheduling time was then used as the mean of a geometrically distributed collision resolution (windowing) time. The performance results obtained using these analytic approximations were shown to coincide closely with simulation results.

A final complication in evaluating equation 4.10 is that the average scheduling time (and thus message service time) depends on the fraction of messages actually entering the queue and eventually receiving service, i.e., the scheduling time components of $\overline{x}$ and $\beta(w)$ are dependent on p(loss). However, since the scheduling delay is known to be exactly 0 for the case that K=0, $\overline{x}$, $\beta(w)$ and p(loss) can be computed exactly at K=0. The values of $\overline{x}$ and $\beta(w)$ at K=ϵ (ϵ close to 0) can then be closely approximated using the exact fraction of messages entering service for K=0; these values can then be used to compute the loss at K=ϵ. In this fashion the loss at the nth value of K can be iteratively computed using the loss at the (n-1)st value of K to compute $\overline{x}$ and $\beta(w)$.

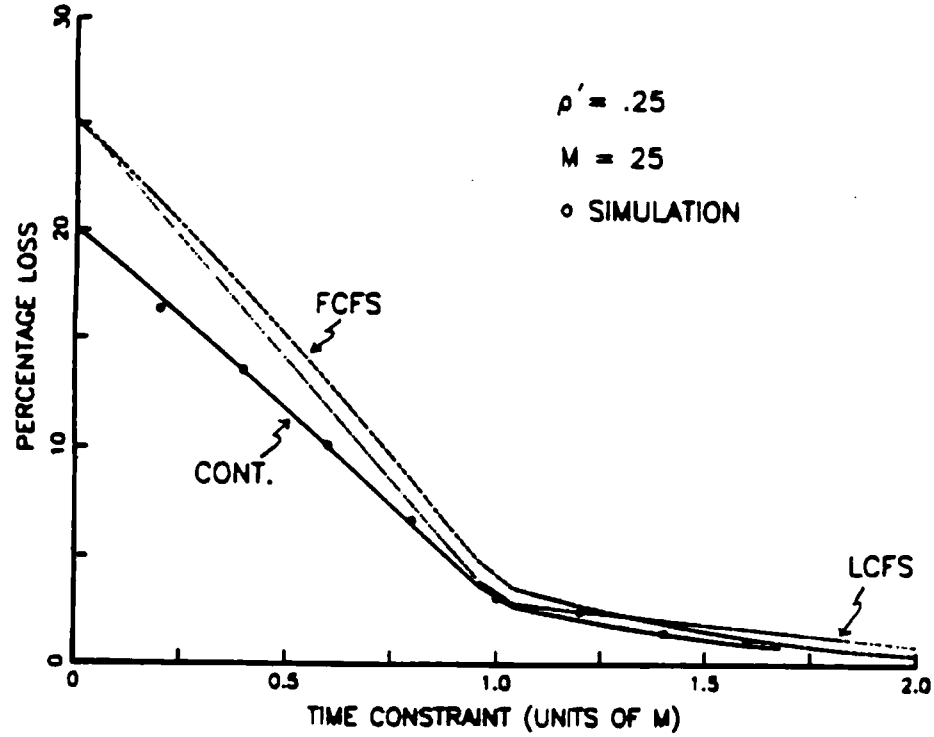
4.4.2. Some Numerical Results

In figures 4-7 through 4-9 we present some numerical results for the performance of the time window protocol in the case that the windowing policy elements (1), (3) and (4) are chosen optimally and the heuristic discussed above is adopted for policy element (2). These results are compared with time-constrained performance results in the case that the protocol provides FCFS and LCFS service and messages are lost only at the receiving stations. Performance results are given for various values of M and ρ', where M is the fixed message length in units of the end-to-end propagation delay of the channel, τ, and ρ' is the network-wide message generation rate (lost and transmitted) times M.

As expected, these results show significant performance improvements over the FCFS and LCFS results. Two factors contribute to this increase in performance. First, the optimal policy elements (1) and (3) have been used; the critical role of these two policy elements was demonstrated in the previous chapter. However, the increase in performance probably results primarily from the inclusion of policy element (4). Note that as a result of this policy element, the channel is used only for "useful" work. That is, if the protocol sends a message, due to element (4) and our definition of waiting time, that message will be accepted at the receiving station with probability 1. Thus, unlike the time window protocol described in the previous chapter, the channel is *never* used for the transmission of messages which are lost at the receiving station.

Recall that our definition of waiting time in section 4.2 does not include a message's *own* scheduling time as part of its waiting time. This definition only approximates the more traditional definition of waiting time: the time between a message's generation at a sending station and the start of its successful transmission. The waiting time approximation was introduced to avoid considerable complication to the analyses of sections 4.3 and 4.4. In the *simulation* results shown in figures 4-7 through 4-9, messages were considered lost when the amount of time between their generation and the beginning of their successful transmission (i.e, their waiting time as traditionally defined) exceeded the imposed time constraint. (Note that in this case it is possible for

Figure 4-7: Message loss as a function of time imposed time constraint for the controlled time window protocol with $\rho' = .25$



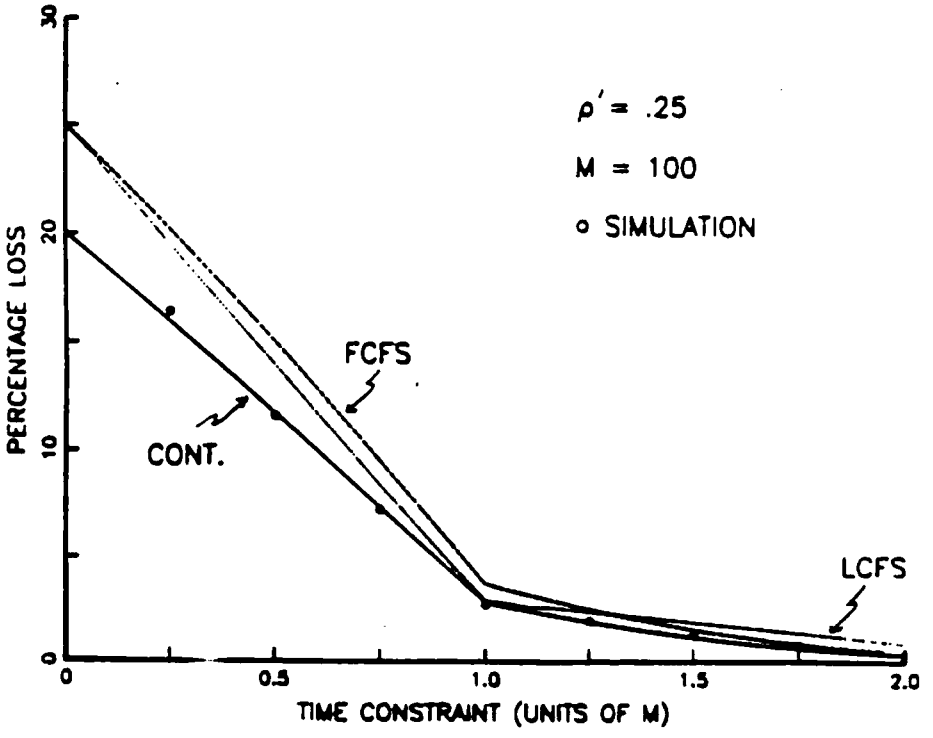
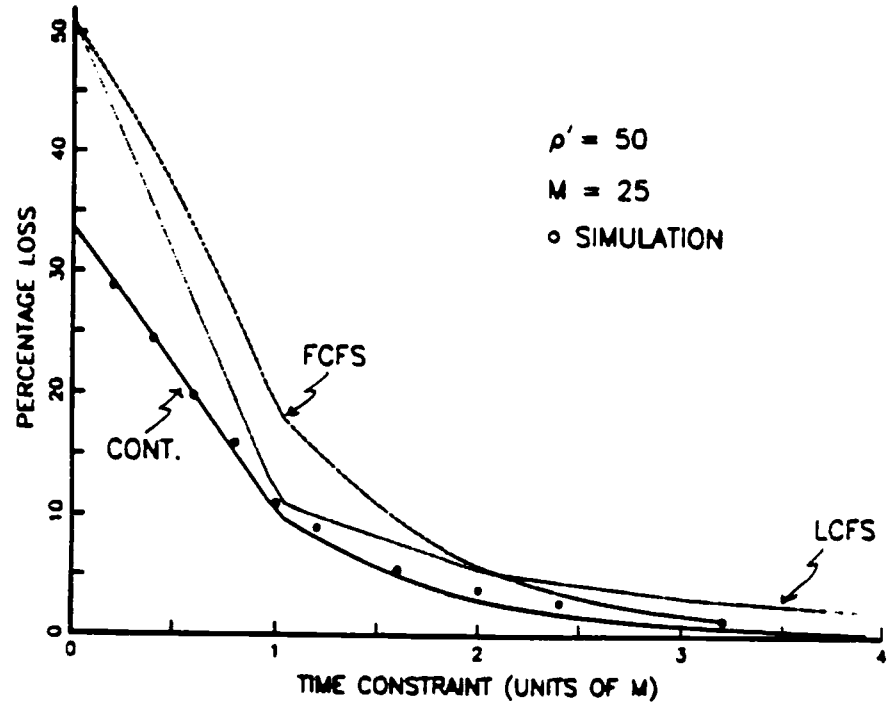


Figure 4-8: Message loss as a function of time imposed time constraint for the controlled time window protocol with $\rho' = .50$



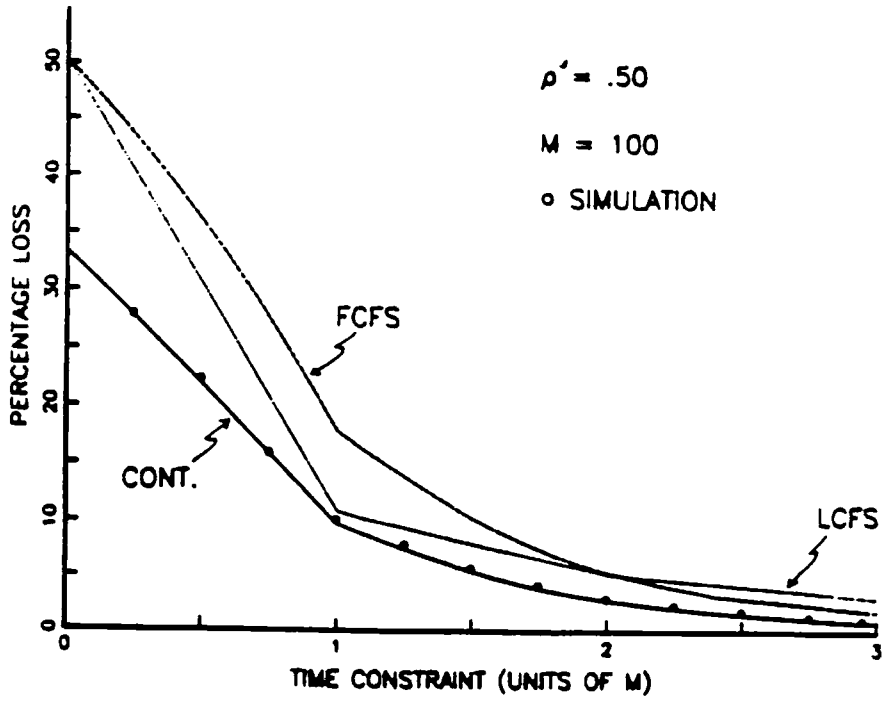
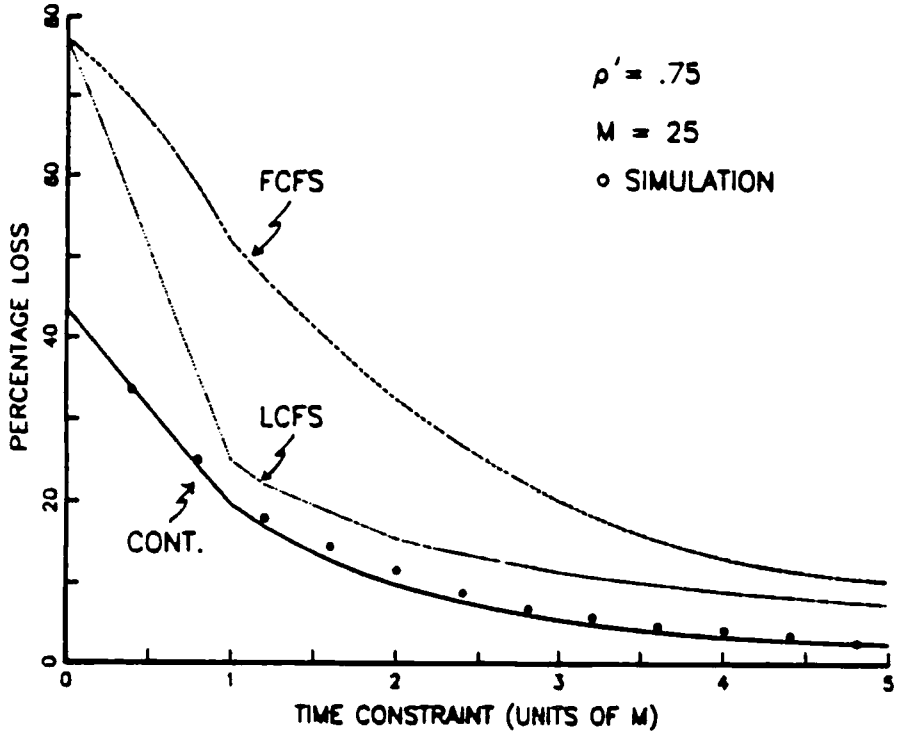
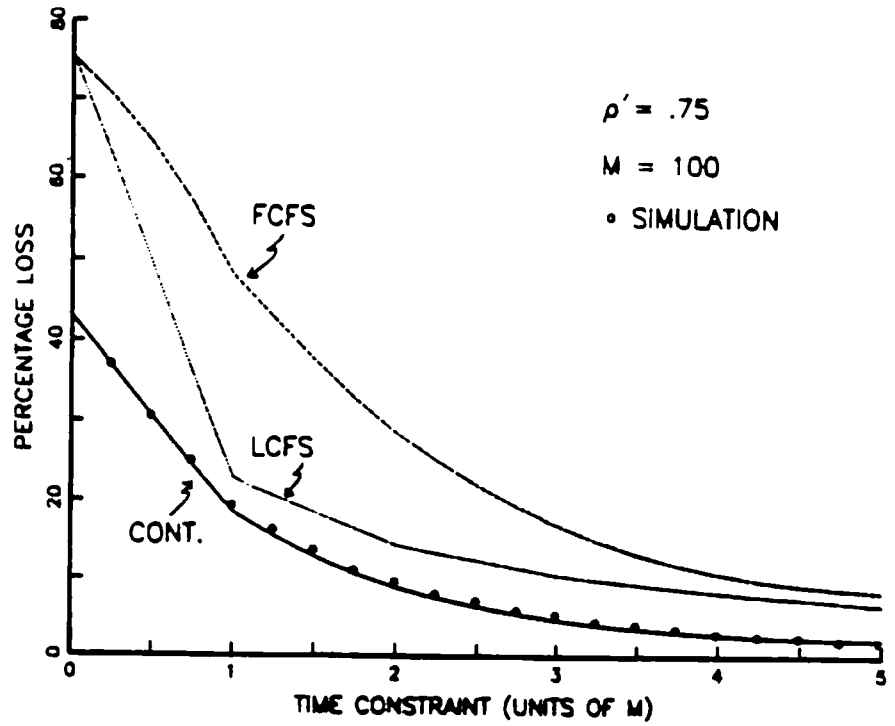


Figure 4-9: Message loss as a function of time imposed time constraint for the controlled time window protocol with $\rho' = .75$





messages to be lost at both the sending and receiving stations.) The close agreement between the analytic results and the simulation results indicate that the waiting time approximation as well as the other approximations and assumptions underlying the analysis are indeed reasonable.

4.5. Summary

In this chapter we have examined the problem of controlling the time window protocol in order to maximize the percentage of messages with delays below a given bound. First, we identified four policy elements which comprised a policy for controlling protocol operation. A semi-Markov decision model was then developed and three of the four optimal policy elements were determined within this model; these elements were found to be both intuitive and simple. A simple closed form characterization of the final optimal policy element was not found.

Although the decision model was shown to be sufficient to provide performance results, it was found to be too computationally expensive to be of practical use. Thus an alternative performance model, based on a queueing system with impatient customers, was developed. A heuristic was then adopted for the final policy element. Protocol performance was then examined and found to be superior to cases in which it was not controlled using optimal policy elements (1), (2) and (4).

4.6. Appendix to Chapter 4: Proof of Lemma 4.4

Lemma 4.4 : Given policy element 4 in section 4.3.1, a policy which minimizes the one-step pseudo loss also minimizes the pseudo loss.

Proof: Let $P' \in \{P^w\}$ be a policy with a decision, k', which for some state $s_i \epsilon \mathbf{S}$ does *not* minimize the one step pseudo loss. Also define:

- p_{ij}^k to be the probability that state s_j immediately succeeds state s_i given that decision k is made upon entry to state s_i.

- τ_i^k to be the average time from when a decision is made upon entry to state s_i to the time when the next decision is made, given decision k is made upon entry to state s_i.

- r_i^k to be the one step pseudo loss in state s_i, given decision k is made upon entry to state s_i.

Recall that $\{P^w\}$ is a set of policies which choose the same action for policy element (2), i.e., choose the same size initial window when in the same state. Since any two equal length intervals of time are statistically identical with respect to the message generation process, if policies P and P′ are in $\{P^w\}$, and choose actions k and k′ respectively when in state s_i, then

$$p_{ij}^k = p_{ij}^{k'} \quad \text{and} \quad \tau_i^k = \tau_i^{k'}$$

We now want to determine if P′ minimizes the average pseudo loss even though it does not (by hypothesis) minimize the one step pseudo loss for state s_i. If P′ does not minimize the average pseudo loss then there must exist a policy iteration [Howard 71] starting from P′ which yields a policy with a smaller pseudo loss. To determine if such a policy iteration exists, we would normally have to solve the set of simultaneous equations:

$$v_n + g\tau_n^{k'} = -r_n^{k'} + \sum_{j=1}^{K} p_{nj}^{k'} v_j \qquad n=1,2 \ldots K \qquad \text{(A1)}$$

where $v_K=0$ and $\tau_n^{k'}$, $r_n^{k'}$ and $P_{nj}^{k'}$ are computed using the decisions of policy P′ to determine the values of g and $\{v_j\}$. g is known as the *gain* under policy P′ and can be related to the average pseudo loss; $\{v_j\}$ is known as the set of relative values. To determine if a policy iteration exists, we must examine the value of γ_i^k, defined below, for all possible decisions, k, in state s_i:

$$\gamma_i^k = -(r_i^k/\tau_i^k) + (1/\tau_i^k) \sum_{j=1}^{K} p_{ij}^k v_j \qquad \text{(A2)}$$

Fortunately, for the purpose of proving that P′ does not minimize the average pseudo loss, we need not actually solve A1 (note, however, that a numerical value for the gain (and hence the message loss) can be obtained from solving the set of equations, A1). Now, if we can show that there is some alternate decision, *k*, such that

$$\gamma_i^{k'} < \gamma_i^k$$

then P′ does not maximize the gain and hence does not minimize the average pseudo loss [Howard 71]. By our above arguments, p_{ij}^k, and τ_i^k are independent of k for all $P \epsilon \{P^w\}$. Thus, A2 can be expressed:

$$\gamma_i^k = -c_1 r_i^k + c_2$$

where c_1 and c_2 are constant with respect to k and c_1 is positive. Since $r_i^k \geq 0$, the maximum value of γ_i^k occurs when r_i^k is minimized. By hypothesis, P′ chooses k′ such that r_i^k is *not* minimized and thus there exists some *k* such that r_i^k is less than $r_i^{k'}$. Thus there exists a policy iteration on P′ and hence P′ does not minimize the average pseudo loss. Thus, any policy which does not minimize the one step pseudo loss in every state does not minimize the average pseudo loss.

Chapter 5
Integrating Multiple Classes of Traffic

In the previous two chapters we have studied protocol mechanisms for efficiently transmitting time-constrained traffic in a multiple access network. In this chapter, we examine the extension of these basic mechanisms to support both time-constrained and non-time-constrained traffic in such a network environment.

5.1. Introduction

The advent of integrated services digital networks (ISDN's) [Pokress 84] provides a new problem setting for time-constrained communication applications and introduces new problems of both practical and theoretical interest. In an ISDN environment, the communication network must support the transmission requirements of not only time-constrained applications (e.g., packetized voice and video) but additional non-time-constrained applications (e.g., interactive, bulk and facsimile traffic) as well; the differing characteristics, performance requirements and performance tradeoffs for these two types of traffic have been previously described in chapter 1. Network applications requiring such multi-media transmission capabilities include multi-media mail, teleconferencing and computer-aided instruction.

A principal challenge in the development of ISDN's is the design and analysis of efficient and robust transmission protocols which can support both time-constrained and non-time-constrained types of traffic. Furthermore, in order to reduce the complexity of the network hardware and software, a single protocol mechanism should ideally accommodate the differing characteristics of the two traffic types.

Previous research on ISDN's has primarily focused on the *centralized* problem of multiplexing the two types of traffic onto a single outgoing transmission channel [Chang 77, Maglaris 79, Weinstein et al. 80, Schwartz and Kraimeche 83]. In these approaches, time is divided into frames and the frames further divided into slots (as in TDMA). A fraction of these frame slots are then allocated for the transmission of the time-constrained traffic (on either a static or dynamic basis) and the remaining frame slots are then allocated to the non-time-constrained traffic. The matching of the incoming message traffic to the frame slots is performed by the centralized multiplexing mechanism.

In a multiple access network, these two types of traffic can be generated at each of the geographically distributed stations and no such centralized multiplexing point exists. In this case, the stations' channel access protocol determines the manner in which the messages are multiplexed and transmitted over the multiple access channel. One protocol for transmitting both voice (time-constrained) and data (non-time-constrained) traffic in a multiple access network was recently proposed by Maxemchuk [Maxemchuk 82]. In this scheme, voice messages take priority over the data messages and the periodic nature of the voice messages within a single conversation is used to predict the generation times and transmission times of future voice messages within the conversation and thus reduce the number of message collisions. One drawback of this scheme, however, is the necessity of transmitting both voice activity as well as voice silent periods; the ability to detect voice silent periods and transmit other messages during these silent periods was shown to be of considerable importance in [Bially et al. 80a]. A second drawback of this scheme is the need for each station to maintain information about *every* voice conversation in the network.

In this chapter, we demonstrate how the ideas developed in previous chapters naturally extend to the case of ISDN applications. Although we will focus on the case of a single class of time-constrained traffic and a single class of non-time-constrained traffic, the ideas and results presented in this chapter can be extended for the case of additional classes of these two basic traffic types as well. The basic extensions to the time window mechanism necessary to support these two traffic classes are first presented in the following section. We then

consider the tradeoffs between the performance levels of the time-constrained and non-time-constrained traffic and demonstrate how the windowing mechanism can be used to select an operating point along such a tradeoff curve. An approximate performance model is then developed for the multi-class time window protocol and these performance tradeoffs are then quantitatively examined. The numerical results are then discussed and compared with simulation results.

5.2. Extending the Time Window Protocol for Multiple Classes of Traffic

The windowing mechanism described in the previous two chapters can be easily generalized for the case of multiple classes of traffic as follows. The same process of choosing intervals (or windows) of time in the past can again be used to select messages (on a network-wide basis) for transmission. However, rather than maintaining a single time axis history as shown in figure 4-2 of the previous chapter for the single class case, each station now maintains a separate time axis history for each class of traffic. Thus, as shown in figure 5-1, each station now maintains two time axis histories, one for the time-constrained traffic and one for the non-time-constrained traffic. Each time axis history is again used to record those intervals of time in the past which have been *fully probed* for messages of the associated traffic type. In the present multi-class case, a window of time is considered to be fully probed with respect to a given traffic type if, and only if, a window of time has been placed over the time interval, either zero or one messages with the associated traffic type were found to have been generated within this interval, and the message (if any) which was generated during this interval has been successfully transmitted.

The stations again operate synchronously. When the opportunity arises to attempt message transmissions, each station selects both a window of time in the past and a traffic type and then attempts to transmit a message if it has a message which was generated within the chosen time window and a message type which matches the selected message type. Recall that these windows are chosen according to some *policy*. For the present case of two classes of traffic, in addition to determining the windowing policy elements described in chapter 4:

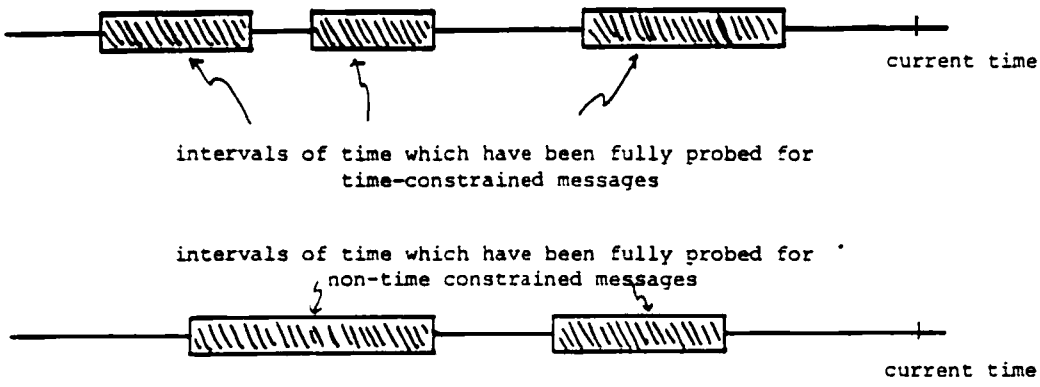


Figure 5-1: A station's view of the time axis for two class of traffic

1. **position of the initial window**
2. **length of the initial window**
3. **window splitting policy**
4. **intervals of time to discard** (i.e., intervals of time to be marked as fully probed even though a window has never been placed over the time interval)

each station must also decide:

5. **selection of either time-constrained or non-time-constrained traffic**, i.e., the class of traffic to check for within the chosen time window

Once the initial window of time and message type have been chosen, the windowing process can then proceed as described in the previous two chapters. Note that we have not yet specified the stations' policy elements (1) through (5) above, nor have we automatically adopted the optimal windowing policy elements of the previous chapter for the time-constrained traffic. First, we wish to examine the effects of policy elements (1) through (5) on the performance tradeoffs between the time-constrained and non-time-constrained classes of traffic.

5.3. Performance Tradeoffs and a Multi-Class Time Window Protocol

In this section we present a multi-class windowing scheme for supporting both time-constrained and non-time-constrained classes of traffic in a multiple access network. In this scheme, time-constrained messages are given preemptive priority over the non-time-constrained messages. It should be noted that the protocol described below, however, represents but one way in which the time window protocol can be modified to support multiple classes of traffic. Our goal here is simply to demonstrate that the windowing mechanism can be easily extended to such an environment and to examine some of the possible tradeoffs between the performance levels of the two classes of traffic. Alternate windowing schemes can be easily imagined and a comparative performance study of such schemes remains a problem for future research.

As discussed in chapter 1, a primary performance metric for time-constrained traffic is message loss, i.e., the fraction of messages which are *not* transmitted by the sending station within a given amount of time after their generation at a sending station; average message delay is the primary performance measure for the non-time-constrained traffic.

The tradeoff which exists between these two performance measures in our multi-class preemptive priority scheme has the qualitative characterization that the smaller the message loss of the time-constrained traffic, the larger the average delay of the non-time-constrained traffic. This can be easily seen by noting that, provided messages are lost at the sending stations, a smaller message loss implies that the channel is *less* frequently available for transmission of non-time-constrained traffic and hence the average delay of this traffic class will be greater. Conversely, a larger message loss for the time-constrained traffic implies a smaller average time delay for the non-time-constrained traffic. Ideally, the multi-class time window protocol should easily permit the selection of a particular operating point along such a tradeoff curve; the scheme described below provides exactly this flexibility.

In our multi-class version of the time window protocol, time-constrained traffic

is given preemptive (resume) priority over the non-time-constrained traffic. That is, if any unsent time-constrained message presently in the network has a delay less than the specified time constraint, it will be transmitted before any present non-time-constrained message is sent. Furthermore, if a non-time-constrained message is being sent and a time-constrained message is generated at any station in the network, that station jams the channel by transmitting a burst of noise, and the station transmitting the non-time-constrained message is preempted and ceases message transmission; the preempting station then immediately begins transmission of its time-constrained messages. At some later point, when it is known that there are no further time-constrained messages in the network (i.e., when every interval of time in the past has been fully probed for time-constrained messages), the preempted station can then resume the transmission of its message. In the discussion below, we will assume that two stations never attempt to simultaneously preempt a message transmission, i.e., preempting stations never transmit interfering messages. This assumption, however, could be easily relaxed by introducing a suitable recovery mechanism into the preemption process.

Given the above preemptive-resume scheme, the existence of lower priority non-time-constrained traffic is essentially invisible to the time-constrained traffic. This is not strictly true, since time is required to preempt a lower priority message and since, in order for a time-constrained message to be transmitted without interference, preemption cannot occur during the collision resolution (window splitting) process of lower priority traffic. However, since these times are typically on the order of a single end-to-end propagation delay of the channel, these effects can be assumed to be negligible when the ratio of the end-to-end channel propagation delay to the message transmission time is small. Thus, *the optimal window policy elements (1) through (4) from the previous chapter remain optimal for the time-constrained traffic in the preemptive-resume priority multi-class case.* Specifically:

1. **Position of initial window, time-constrained traffic** Always choose the beginning of the time window at that point in time closest to, but more recent than, t-K (where t is the current time and K is the imposed time constraint) which has not yet been fully probed for time-constrained messages.

2. **Window length heuristic, time-constrained traffic** Given the network-wide rate at which time-constrained traffic is generated, $\lambda_{t\text{-}c}$, choose the window length to minimize the average windowing time under the model of chapter 3.

3. **Window splitting policy, time-constrained traffic** Always choose the older half of a split window.

4. **Discard policy, time-constrained traffic** Always discard intervals of time before t-K.

Furthermore, given the preemptive-resume priority scheme discussed above, we have

5. **selection of time-constrained or non-time-constrained traffic** If any interval of time in [t-K,t] has not yet been fully probed for time-constrained messages, check for time-constrained traffic using windowing policy elements (1) through (4) above. If no such interval exists, check for non-time-constrained traffic using window policy elements (1) through (4) below.

with the additional condition:

6. **Preemption.** If a non-time-constrained message is being transmitted and a time-constrained message is generated at a station in the network, the non-time-constrained message is preempted as discussed above. Furthermore, at the next point in time when it is known there are no time-constrained messages in the network, transmission of the preempted message can be resumed immediately.

All that remains is the specification of policy elements (1) through (4) for non-time-constrained traffic. Recall that under the assumptions of chapter 4, the average delay is invariant to policy elements (1) and (3). Thus, let us assume that (1) and (3) are chosen to implement FCFS scheduling. Since no non-time-constrained message loss is allowable, no non-time-constrained messages can be discarded at the sending stations via policy element (4). Finally, we will again choose the window length in order to minimize the average contention time under the model of chapter 3. In summary:

1. **Position of initial window, non-time-constrained traffic** Always choose the beginning of the time window at the oldest point in time which has not yet been fully probed for non-time-constrained messages.

2. **Window length heuristic, non-time-constrained traffic** Given the network-wide rate at which non-time-constrained traffic is generated, $\lambda_{n\text{-}t\text{-}c}$, choose the window size to minimize the average windowing time under the model of chapter 3.

3. **Window splitting policy, non-time-constrained traffic** Always choose the older half of a split window.

4. **Discard policy, non-time-constrained traffic** Never discard any intervals of time.

The operation of the multi-class time window protocol under policy elements (1) through (6) above is shown in figure 5-2; t represents the current time, K is the time constraint and τ is the end-to-end propagation delay of the channel. Since *all* messages which are sent are transmitted on a FCFS basis, we can again use single values, $t\_past_{t\text{-}c}$ and $t\_past_{n\text{-}t\text{-}c}$, to represent those points in time such that all untransmitted time-constrained and non-time-constrained messages are known to have been generated in the intervals $[t\text{-}t\_past_{t\text{-}c},t]$ and $[t\text{-}t\_past_{n\text{-}t\text{-}c}]$, respectively. The network-wide time-constrained and non-time-constrained message generations are indicated by arrows below the respective time axes (note that although there are no unsent time-constrained messages in the network, this fact is not known by the stations until the state of the network is as shown in figure 5-2d). Figure 5-2a shows the system state just before the beginning of a windowing process. Since the interval $[t\text{-}t\_past_{t\text{-}c},t]$ has not been fully probed for time-constrained messages, the initial time window is placed at the beginning of this interval as shown in figure 5-2b. No time-constrained messages were generated during this interval and thus after the channel remains idle for time τ, the stations again choose a new time window (figure 5-2c). Again, no time-constrained messages were generated during the selected time interval. At this point, all stations now know that there are no unsent time-constrained messages with a delay less than K (the imposed time constraint) anywhere in the network (unless such a message was generated in the most recent τ units of time, i.e., in the interval $[t\text{-}\tau,t]$ in figure 5-2d). Thus, the stations can begin searching for a non-time-constrained message to send by placing an initial time window over the non-time-constrained axis (figure 5-2d). Since two non-time-constrained messages were generated during this initial window, a collision occurs, the window is split in half, and the older half of the split window is selected as the new time window. Only a single message was generated during this new time window and thus the successful transmission of this message begins (figure 5-2e).

Figure 5-2: Operation of the multi-class time window protocol

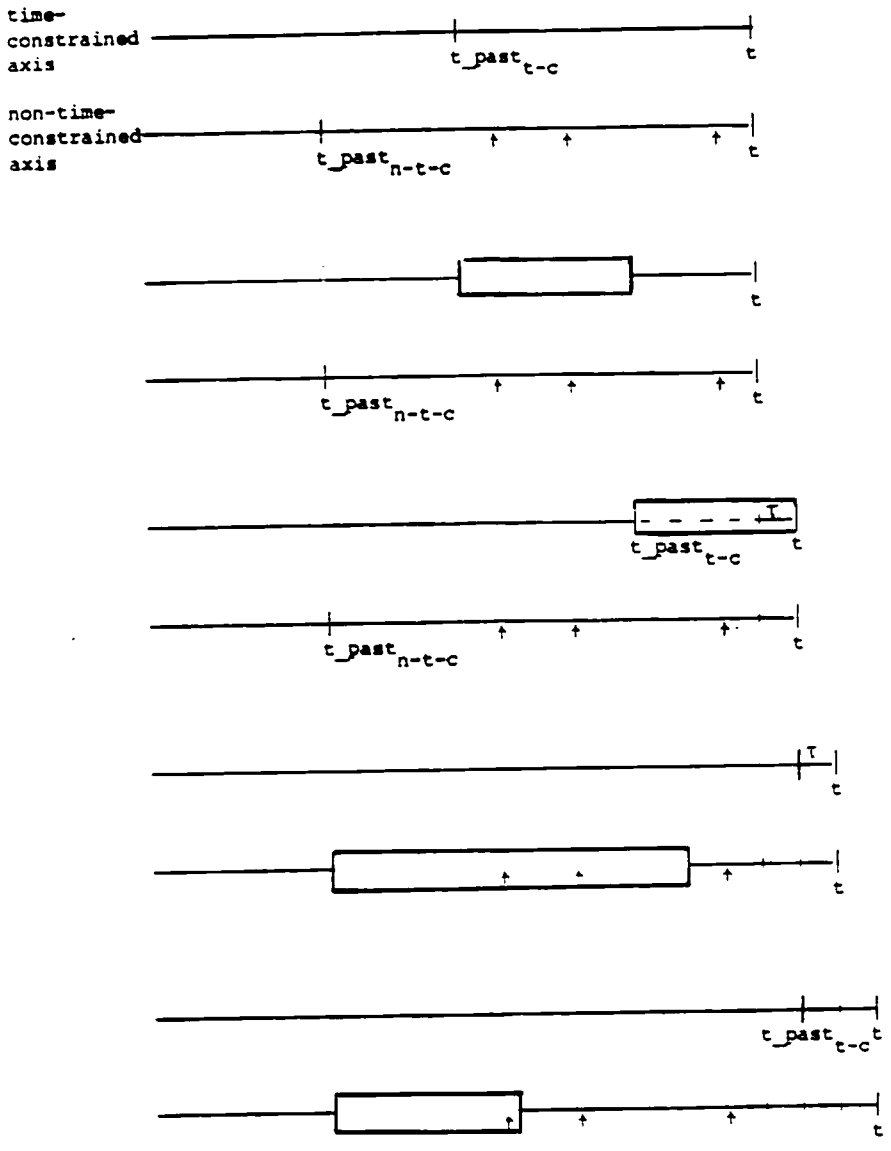


figure 5-2a

figure 5-2b

figure 5-2c

figure 5-2d

figure 5-2e

Note that once the transmission of this non-time-constrained message has begun, any time-constrained message which was generated during the windowing period for the non-time-constrained traffic (i.e., in the interval [t-2τ,t] in figure 5-2e) or is generated during the transmission time of the non-time-constrained message can preempt the transmission of this message. Also, once the transmission of a non-time-constrained message begins, the value of t_past_{t-c} can be continuously updated, since the absence of channel jamming indicates that a time-constrained message has not yet been generated anywhere in the network.

In terms of the tradeoff between the performance levels of the two classes of traffic, it is clear that the time constraint, K, plays a critical role. The larger the value of K, the smaller the loss and the larger the average delay. Conversely, the smaller the value of K, the larger the loss and the smaller the average time delay. In the following section we construct an approximate analytic performance model in order to quantitatively study this tradeoff.

5.4. An Analytic Model of the Multi-class Time Window Protocol

5.4.1. A Multi-Class Queueing Model with Time-Constrained and Non-Time-Constrained Traffic Classes

The analytic performance model presented in this section is again based on considering the messages distributed at stations throughout the network as customers in a distributed M/G/1 queue. In this model, the service time distribution of these customers is again given by equation 3.8. Policy elements (1) through (6) of the preceding section determine the order in which these messages are transmitted. Thus, time-constrained messages are again transmitted on a FCFS basis and are lost (and denied service) if their wait in the queue exceeds K, the time constraint. Non-time-constrained messages begin service only when there are no time-constrained messages in the queue and can be preempted (as discussed above) when a time-constrained message is generated at a network station.

Our multi-class queueing model is shown in figure 5-3 and differs from standard multi-class queueing models [Kleinrock 76] in one important respect: a time-constrained message is denied service (and hence leaves the queue before service) if its waiting time exceeds the imposed time constraint and thus the arrival process of messages actually entering service is no longer Markovian. As we will see, the average delay of the non-time-constrained messages depends on the average delay of the time-constrained messages and due to the above non-Markovian property, an exact computation of this latter value is not possible. Thus, our performance analysis will require the introduction of several modeling assumptions; these assumptions will be identified in the derivation below.

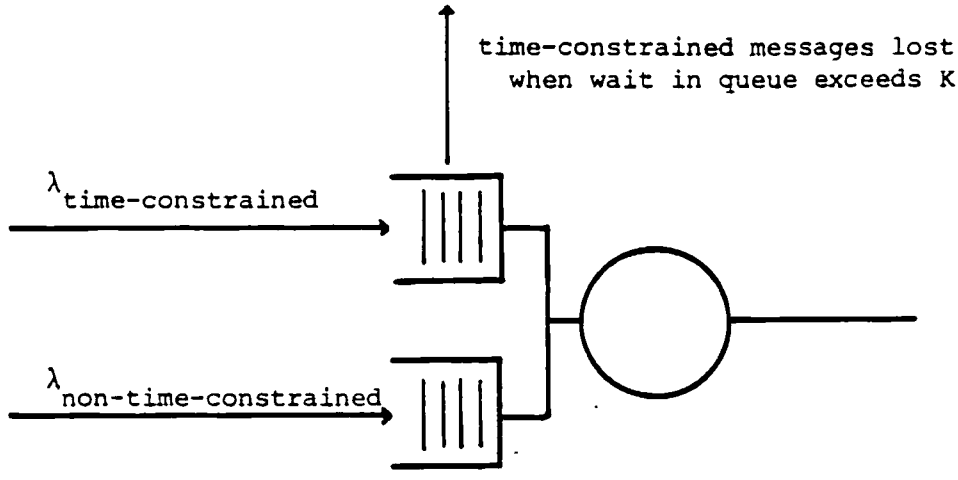


Figure 5-3: A queuing model with two classes of customers

The performance metrics of interest here are the message loss of the time-constrained traffic and the average delay of the non-time-constrained traffic. Recall that due to the preemptive priority transmission discipline, the existence of non-time-constrained traffic is invisible to the time-constrained messages and thus the average message loss of the time-constrained traffic can be computed directly using equation 4.10. In order to determine the average wait of a non-time-constrained customer, (exclusive of its transmission time) we note that this wait consists of several distinct components. Thus, let us define:

$W_{n\text{-}t\text{-}c}$ the average time a non-time-constrained message spends in the system, exclusive of transmission time.

$W^{Q}_{n\text{-}t\text{-}c}$ the average time a non-time-constrained message spends in the system up to the beginning of its transmission.

$W^{P}_{n\text{-}t\text{-}c}$ the average time a non-time-constrained message spends in the queue (due to preemption) after its transmission has begun, exclusive of transmission time.

$\bar{x}_{t\text{-}c}, \bar{x}_{n\text{-}t\text{-}c}$ the average service times (contention time plus transmission time) of time-constrained and non-time-constrained messages, respectively.

$\lambda_{t\text{-}c}, \lambda_{n\text{-}t\text{-}c}$ the average message generation rates of time-constrained and non-time-constrained messages, respectively. The distribution of the inter-generation times is assumed to be exponential.

$loss_{t\text{-}c}$ the fraction of time-constrained messages which are lost. The value of $loss_{t\text{-}c}$ is computed via equation 4.10 and is a function only of K and $\lambda_{t\text{-}c}$.

In order to determine the value of $W_{n\text{-}t\text{-}c}$, we can use the standard technique of following a "tagged" customer through the queue. The total time a customer spends in the system (exclusive of transmission time) is composed of the time spent waiting for transmission to begin plus the time spent in the queue as a result of preemption. Thus:

$$W_{n\text{-}t\text{-}c} = W^{Q}_{n\text{-}t\text{-}c} + W^{P}_{n\text{-}t\text{-}c} \tag{5.1}$$

Note that $W^{Q}_{n\text{-}t\text{-}c}$ itself is composed of two components:

$$W^{Q}_{n\text{-}t\text{-}c} = U^{initial}_{n\text{-}t\text{-}c} + U^{generated}_{n\text{-}t\text{-}c} \tag{5.2}$$

where:

$U^{initial}_{n\text{-}t\text{-}c}$ is the average unfinished work (due to both time-constrained and non-time-constrained messages) found in the queue by a newly generated tagged non-time-constrained message. If no additional time-constrained messages were to be generated during the tagged message's wait, $U^{initial}_{n\text{-}t\text{-}c}$ would then be the total wait (exclusive of preemption) for the tagged customer.

$U^{generated}_{n\text{-}t\text{-}c}$ is the average wait experienced by a lower priority, tagged non-time-constrained message due to higher priority time-constrained messages which are generated *after* it was generated but are transmitted *before* it is transmitted.

$U^{generated}_{n\text{-}t\text{-}c}$ is simply $\lambda_{t\text{-}c}(1\text{-}loss_{t\text{-}c})W^{Q}_{n\text{-}t\text{-}c}\bar{x}_{t\text{-}c}$, the average number of time-constrained messages which are generated during the amount of time, $W^{Q}_{n\text{-}t\text{-}c}$, times the average service time for these messages. In order to compute $U^{initial}_{n\text{-}t\text{-}c}$ we must first compute the average wait of the time-constrained messages which enter service. Unfortunately, an exact computation of this average waiting time requires the solution of a functional Laplace transform equation and a difficult numerical inversion of the resulting solution [Baccelli and Hebuterne 81]. Thus, let us attempt to *approximate* $U^{initial}_{n\text{-}t\text{-}c}$ by assuming that the average waiting time for *time-constrained* messages equals the average waiting time for customers in an M/G/1 queue with no loss and a Poisson message generation rate with mean $\lambda_{t\text{-}c}(1\text{-}loss_{t\text{-}c})$. In this case, $U^{initial}_{n\text{-}t\text{-}c}$ is given by [Kleinrock 76]:

$$U^{initial}_{n\text{-}t\text{-}c} = \frac{[\lambda_{t\text{-}c}\bar{x}^2_{t\text{-}c}(1\text{-}loss_{t\text{-}c})]/2 + [\lambda_{n\text{-}t\text{-}c}\bar{x}^2_{n\text{-}t\text{-}c}]/2}{1 - \lambda_{n\text{-}t\text{-}c}\bar{x}_{n\text{-}t\text{-}c} - \lambda_{t\text{-}c}(1\text{-}loss_{t\text{-}c})\bar{x}_{t\text{-}c}} \tag{5.3}$$

Substituting the above values for $U^{initial}_{n\text{-}t\text{-}c}$ and $U^{generated}_{n\text{-}t\text{-}c}$ into equation 5.2 and rearranging, we get

$$W^{Q}_{n\text{-}t\text{-}c} = \frac{[\lambda_{t\text{-}c}\bar{x}^2_{t\text{-}c}(1\text{-}loss_{t\text{-}c})]/2 + [\lambda_{n\text{-}t\text{-}c}\bar{x}^2_{n\text{-}t\text{-}c}]/2}{(1 - \lambda_{n\text{-}t\text{-}c}\bar{x}_{n\text{-}t\text{-}c} - \lambda_{t\text{-}c}(1\text{-}loss_{t\text{-}c})\bar{x}_{t\text{-}c})\,(1 - \lambda_{t\text{-}c}(1\text{-}loss_{t\text{-}c})\bar{x}_{t\text{-}c})} \tag{5.4}$$

In order to compute $W^{P}_{n\text{-}t\text{-}c}$, the waiting time component of $W_{n\text{-}t\text{-}c}$ due to preemption in equation 5.1, we note that whenever a non-time-constrained message is preempted, a busy period of the time-constrained traffic is begun and the preempted message resumes transmission again at the end of this busy period. Thus, if a non-time-constrained message is preempted k times, the component of its waiting time due to preemption is given by k times the average length of a busy period for time-constrained traffic. The transmission of a non-time-constrained message is preempted k times if, and only if, exactly k time-constrained messages are generated during its service time. Thus we have:

$$W^{P}_{n\text{-}t\text{-}c} = \sum_{i=0}^{\infty} \text{prob(i time-constrained messages generated in } \bar{x}_{n\text{-}t\text{-}c})\cdot i\cdot \bar{B}_{t\text{-}c}$$

where $\bar{B}_{t\text{-}c}$ is the average length of the busy period of the time-constrained traffic.

$\overline{B}_{t\text{-}c}$ can be taken outside the above sum and the remaining sum can be readily identified as the average number of time-constrained messages generated during the service time of a non-time-constrained message. Thus we have:

$$W^P_{n\text{-}t\text{-}c} = \overline{B}_{t\text{-}c}\lambda_{t\text{-}c}\overline{x}_{n\text{-}t\text{-}c} \tag{5.5}$$

Note that in equation 5.5, the message generation rate, $\lambda_{t\text{-}c}$, is not multiplied by the factor $(1\text{-loss}_{t\text{-}c})$. This is due to the fact that the time between the beginning or resumption of a non-time-constrained message transmission and the generation of a time-constrained message is exponentially distributed with mean $1/\lambda_{t\text{-}c}$, not with mean $1/(\lambda_{t\text{-}c}(1\text{-loss}_{t\text{-}c}))$.

If $\overline{S}_{t\text{-}c}$ is the average length of the silent period for time-constrained traffic, the channel utilization for the time-constrained traffic can be expressed:

$$\frac{\overline{B}_{t\text{-}c}}{\overline{B}_{t\text{-}c} + \overline{S}_{t\text{-}c}} = \lambda_{t\text{-}c}(1\text{-loss}_{t\text{-}c})\overline{x}_{t\text{-}c}$$

$\overline{S}_{t\text{-}c}$ is simply $1/\lambda_{t\text{-}c}$, so rearranging and solving for $\overline{B}_{t\text{-}c}$ gives:

$$\overline{B}_{t\text{-}c} = \frac{\overline{x}_{t\text{-}c}(1\text{-loss}_{t\text{-}c})}{1 - \lambda_{t\text{-}c}\overline{x}_{t\text{-}c}(1\text{-loss}_{t\text{-}c})} \tag{5.6}$$

Finally, combining equations 5.1 through 5.6 above we get:

$$W_{n\text{-}t\text{-}c} = \frac{[\lambda_{t\text{-}c}\overline{x^2}_{t\text{-}c}](1\text{-loss}_{t\text{-}c})]/2 + [\lambda_{n\text{-}t\text{-}c}\overline{x^2}_{n\text{-}t\text{-}c}]/2}{(1 - \lambda_{n\text{-}t\text{-}c}\overline{x}_{n\text{-}t\text{-}c} - \lambda_{t\text{-}c}(1\text{-loss}_{t\text{-}c})\overline{x}_{t\text{-}c})(1\text{-}\lambda_{t\text{-}c}\overline{x}_{t\text{-}c}(1 - \text{loss}_{t\text{-}c}))} + \frac{\lambda_{t\text{-}c}\overline{x}_{n\text{-}t\text{-}c}\overline{x}_{t\text{-}c}(1\text{-loss}_{t\text{-}c})}{(1\text{-}\lambda_{t\text{-}c}\overline{x}_{t\text{-}c}(1\text{-loss}_{t\text{-}c}))} \tag{5.7}$$

5.4.2. Some Numerical Results

Figures 5-4, 5-5 and 5-6 show the tradeoff between the message loss of the time-constrained traffic and the average delay (measured in units of message transmission time) of the non-time-constrained traffic for various message

generation rates; the loss and average delay were computed using equations 4.10 and 5.7. In each of these figures, the message length, M, is assumed to be fixed and equal to 100 times the end-to-end propagation delay of the channel. $\rho_T = \rho_{t\text{-}c} + \rho_{n\text{-}t\text{-}c}$, where $\rho_{t\text{-}c} = M\lambda_{t\text{-}c}$ and $\rho_{n\text{-}t\text{-}c} = M\lambda_{n\text{-}t\text{-}c}$. Analytic results are shown in solid lines and simulation results are shown as point values.

Recall that the message loss (and hence the average waiting time) is dependent on the imposed time constraint, K. The value of K associated with each of the simulation points is also shown in figures 5-4 through 5-6. Where discrepancies exist between the analytic and simulation results, the simulation result for a given K is paired with its corresponding analytic value. Note that the upper and lower endpoints of the tradeoff curves correspond to the cases K=0 and K=∞, respectively.

As shown in figures 5-4 through 5-6, the analytic results agree closely with the simulation results in all but some of the heavy traffic cases; even in these cases, the difference in the average delay is always within 25% of the computed value and the simulated message loss is within a few percent of the computed loss. We believe that the discrepancy in the calculated average delay results primarily from the approximation used to compute $U_{n\text{-}t\text{-}c}^{initial}$.

Note that in all but the heavy traffic cases, the tradeoff curves are almost vertical. This reveals an important aspect of the tradeoff between loss and average delay: *in all but the high traffic cases, a small increase/decrease in the average delay of the non-time-constrained traffic is accompanied by a large decrease/increase in the time-constrained message loss.* For example, in figure 5-4 with $\rho_{t\text{-}c} = \rho_{n\text{-}t\text{-}c} = .3$, a time constraint of 1 results in a 4% loss and an average delay of 1.4. If the time constraint is tightened to 0.25, the average delay is reduced approximately 20% to 1.1 but message loss increases over 450% to 18.5%. Thus, except for high traffic situations, trading off time-constrained message loss against the average delay of non-time-constrained traffic does not appear to be a reasonable option. Rather, the operating point should simply be chosen as that point with the largest tolerable time constraint possible.

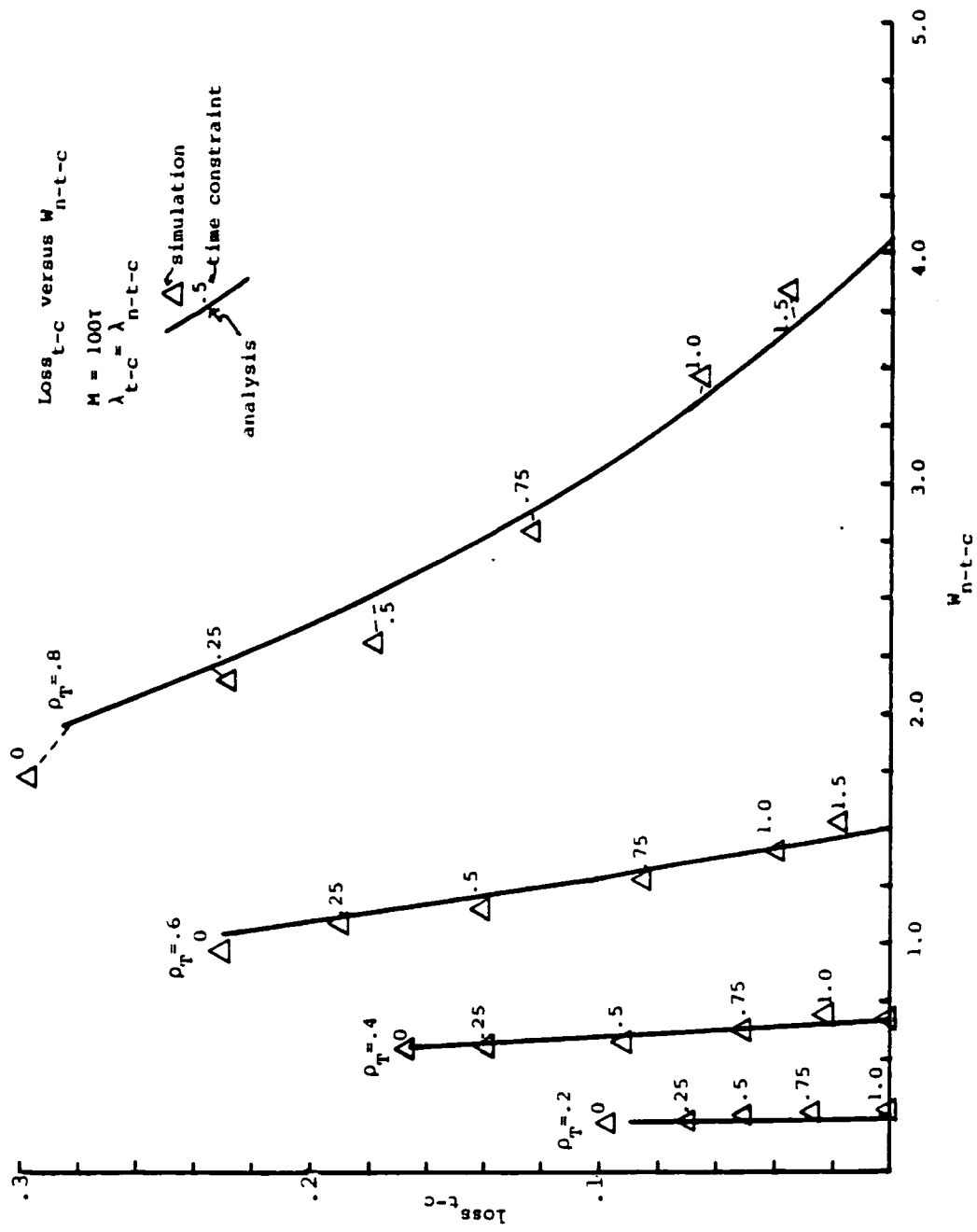


Figure 5-4: Loss versus W_{n-t-c} tradeoff, $\lambda_{t-c}=\lambda_{n-t-c}$

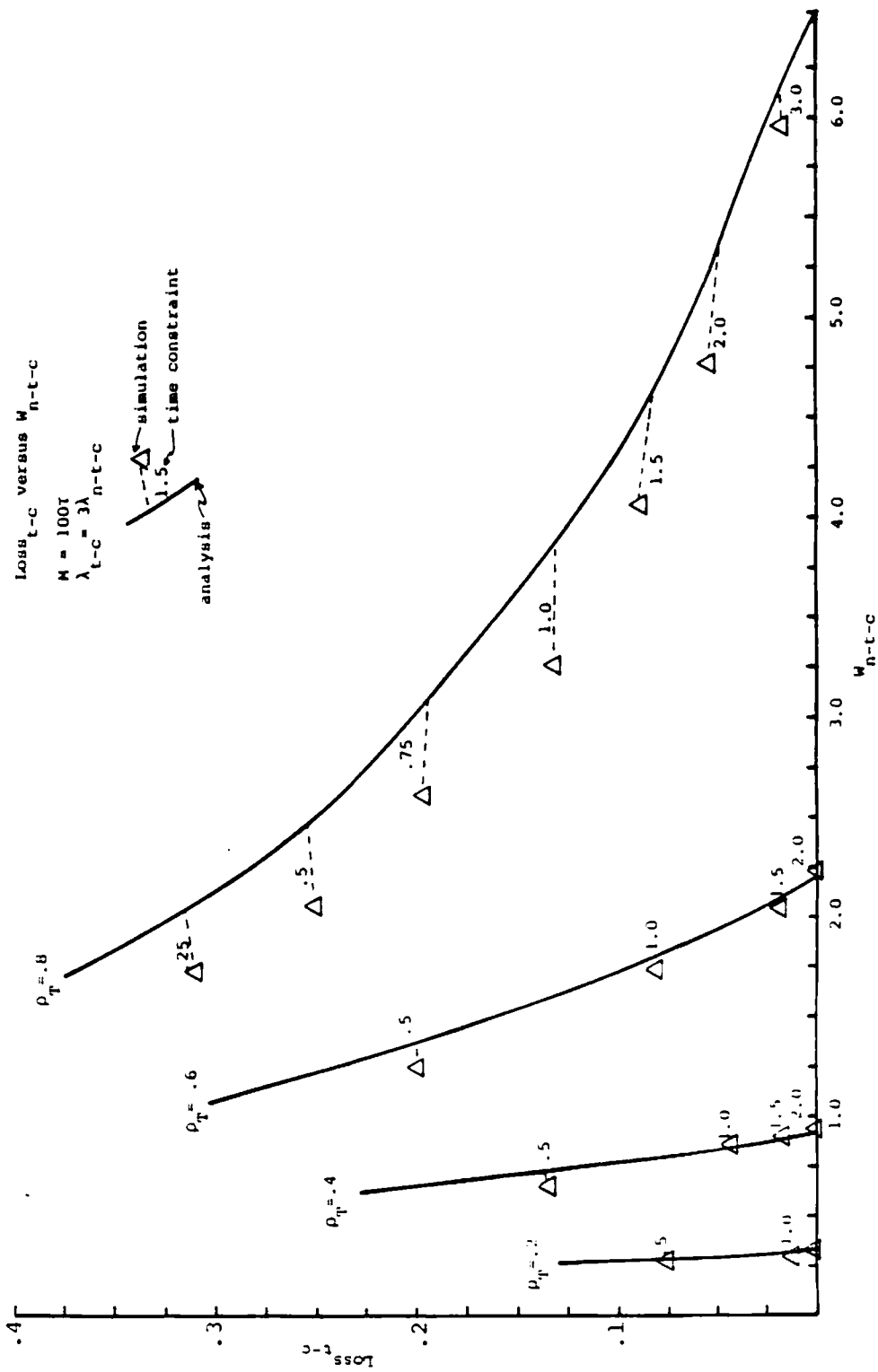


Figure 5-5: Loss versus W_{n-t-c} tradeoff, $\lambda_{t-c}=3\lambda_{n-t-c}$

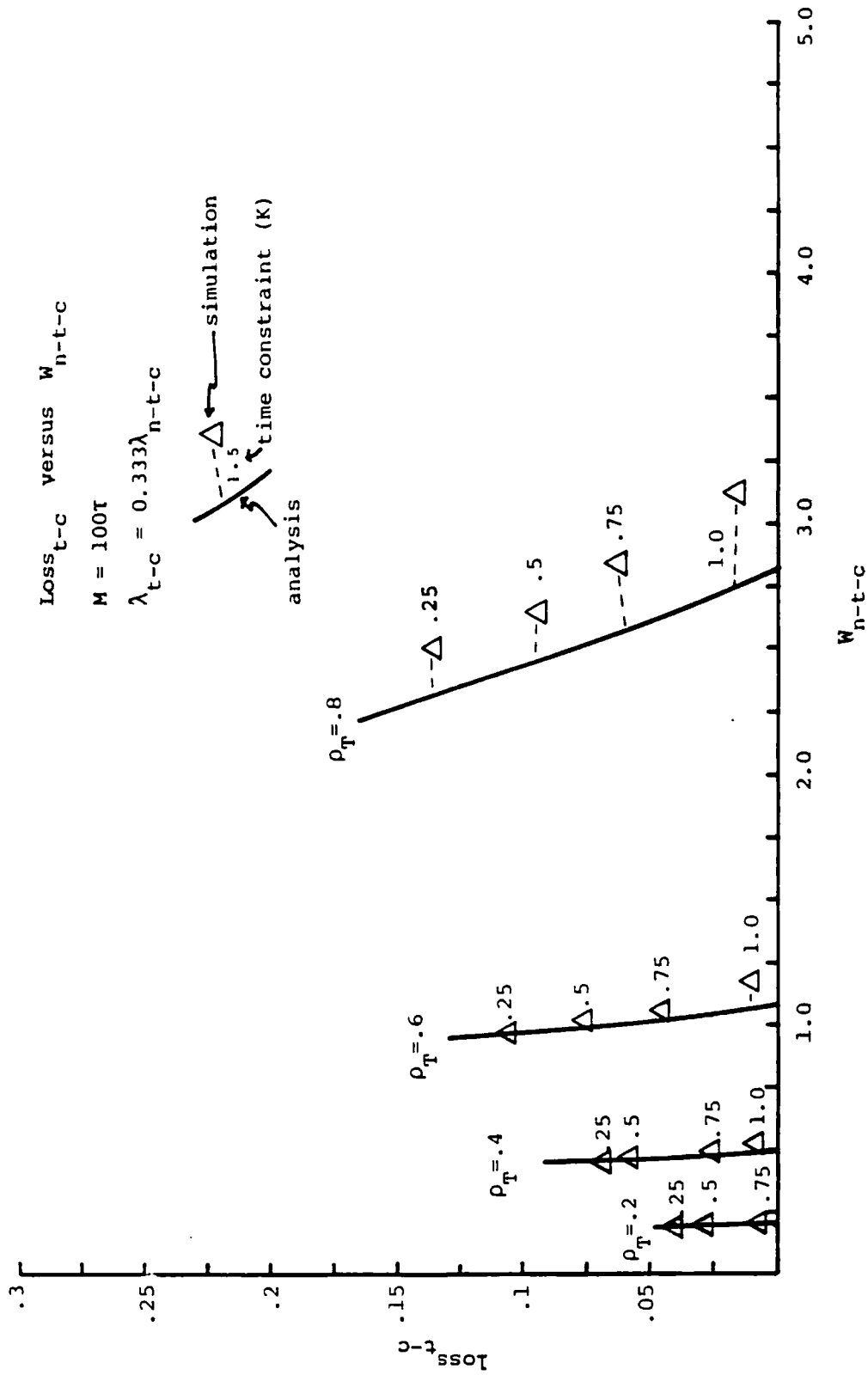


Figure 5-8: Loss versus W_{n-t-c} tradeoff, $3\lambda_{t-c}=\lambda_{n-t-c}$

5.5. Summary

In this chapter we have examined the problem of extending the time window protocol to support both time-constrained and non-time-constrained traffic in a multiple access network. One possible windowing policy, which gives time-constrained messages preemptive resume priority over the non-time-constrained messages was proposed and the tradeoff between time-constrained message loss and non-time-constrained average message delay was then examined. The mechanism in the multi-class time window protocol for selecting an operating point along this tradeoff curve was identified and an analytic model was then developed to quantitatively study this tradeoff. The performance results indicated that in all but the high traffic cases, a small increase/decrease in the average delay of the non-time-constrained traffic was accompanied by a large decrease/increase in the time-constrained message loss. These results suggest that, except for high traffic situations, trading off time-constrained message loss against the average delay of non-time-constrained traffic does not appear to be a reasonable option. In such cases, the time constraint should simply be set to the largest tolerable value. In the high traffic situations, reasonable tradeoffs do exist between the performance levels of the time-constrained and non-time-constrained classes of traffic; the selection of a particular operating point along these tradeoff curves will depend on the performance requirements of the given time-constrained and non-time-constrained applications.

Chapter 6
A Microeconomic Approach Towards Decentralized Optimization of the Time Window Protocol

6.1. Introduction

In the previous chapters, we have presented and examined several protocols based on the use of time windows for supporting time-constrained communication applications in a distributed multiple access network. Recall that the window sizes used by these protocols (i.e., windowing policy element 2 in chapters 3, 4, and 5), were those window sizes which minimized the average amount of time required by the windowing process to select a single message for transmission or, equivalently, maximized the capacity of the protocol. In an operational network, the optimum window size must be computed by the stations themselves. This optimization problem would typically be first solved when the network is initialized (i.e., before any generated messages are transmitted). Then, if the network operating parameters (e.g., message generation rates and the relative priorities of the stations) change over time, the stations would periodically cease message transmission and recompute the optimal window size given the new values of these parameters.

The approach taken in this chapter towards solving such an optimization problem in an operational network is based on the observation that related optimization problems are routinely and efficiently solved in another decentralized environment, which is in many ways similar to our network setting. As in a network setting, the distributed, intelligent, information-processing entities in this environment must share and process information in order to solve certain resource allocation problems. The environment is the perfectly competitive economic marketplace and people are the intelligent information-processing agents within this environment.

In this chapter, we will show that in many cases, *the problem of optimizing transmissions in a multiple access network (specifically, optimizing the steady state probability with which a station attempts a message transmission when permitted to do so) can be formulated in terms of a fictitious optimal resource allocation problem* and that an optimal solution to this fictitious problem immediately yields an optimal solution to the message transmission problem. We will then use a pricing system and the perfectly competitive market mechanism similar to that used in microeconomic models of perfect competition [Karlin 59, Arrow and Hahn 71] to provide a *distributed* solution to this fictitious resource allocation problem.

The approach taken towards distributed optimization in this chapter is thus to consider the network stations as forming a loosely coupled *artificial society* of processors and to use *models* of human economic and social organization as *blueprints* for engineering optimization algorithms within this society of network agents. In this approach, agents act as "selfish", utility-maximizing entities, and their interaction through the pricing mechanism serves as a decentralized computational device for computing an optimal allocation of resources. Quite interestingly, when the optimization results are related back to the original problem of optimizing transmissions in multiple access networks, several network mechanisms, such as flow control and priorities, are seen to emerge naturally from this approach.

In the following section, we discuss the problem of centralized versus decentralized optimization and discuss related work in the fields of computer networks and microeconomics. In section 6.3, we define the fictitious network resources and relate the problem of determining an optimal allocation of these fictitious resources to the problem of determining optimal transmission probabilities in a multiple access network. In order to first test these ideas in a fairly simple network setting, the microeconomic approach is first used to determine the optimal transmission probabilities in a Slotted Aloha [Abramson 70, Abramson 73] network. In section 6.5, we then examine the use of this approach in determining the optimal window sizes (windowing policy element 2 in previous chapters) for the time window protocol. The use of the decentralized,

microeconomic approach towards optimizing the Slotted Aloha and time window protocols is experimentally investigated in a small (4 node) multiple access network. In the case of a homogeneous network environment, this approach towards optimizing the channel access policy (specifically, optimizing the probability of a successful message transmission subject to a throughput constraint) is shown to reproduce known optimality results for both the slotted ALOHA and time window protocols. The extension of these optimality results to a heterogeneous network environment and the introduction of station priorities is also studied. A summary and final discussion of this work is then presented in section 6.6.

6.2. Centralized versus Decentralized Optimization

How should an optimization problem such as computing the optimal window sizes for the time window protocol or determining the optimum allocation of resources among network agents (node, processes, etc..) be solved in a network environment? The classical approach to this problem views the network as a single entity for which some global performance measure has been defined. Such an approach leads to a centralized optimization problem (typically solved by a single agent or network manager), the results of which are then made known to, and imposed on, the network agents. There are several drawbacks to such an approach. First, an inherent drawback in any such centralized scheme is that of reliability. If a critical network function is performed by any single agent, the operation of the entire network is vulnerable to a failure in that agent. A second drawback is the cost and performance of such a centralized computation. A centralized optimization requires information about the operational parameters (traffic statistics, constraints, priority information) and the functional form of the relevant performance metrics for *each* network agent; transmitting this information to the network manager can be a long and costly process. Furthermore, the optimization problem itself may be an extremely computationally complex task, particularly in a heterogeneous environment. A centralized approach towards optimization ignores the distributed computational power inherent in the network itself and instead utilizes only the computing power of the single agent performing the optimization.

An alternative approach is to view the network as a loose collection of interacting agents, each attempting to maximize its own selfish performance metric. Of course, the conflicting optimization goals of the agents must be reconciled and thus some negotiating process is required to determine an adequate compromise to these conflicting goals. In such an approach, the agents themselves become the elemental computing units and the solution to the optimization problem is obtained through the agents' interaction via the negotiating process. Interestingly, Pareto [Pareto 27] compared the economic marketplace, in which such negotiations take place, to a computing machine as early as 1927!

In a decentralized approach towards optimization, the absence of a single, centralized performance objective to be optimized requires that some alternate notion of optimality be adopted. In this case, the performance objective usually adopted is that of *Pareto optimality.* In the context of our fictitious resource allocation problem, a distribution of resources is said to be Pareto optimal if, and only if, there is no alternate distribution of resources which improves the performance of one set of agents without a concommitant performance degradation for some other set of agents. There is thus a *set* of Pareto optimal allocations. As we will see, the collective choice of a particular allocation of resources from within this set will be determined by the relative priorities of the agents in the network.

Several decentralized resource allocation problems have been recently examined in [Yemini and Kleinrock 79, Yemini 81, Brooks 83]. In this work, it was shown that when network resources are properly defined, the optimum allocation of these resources and the resulting optimal solution for several network control problems can be characterized in terms of simple and intuitive balance relations among these resources. In this present work, we build on these ideas by showing how a resource pricing mechanism can be introduced as a decentralized negotiating mechanism to actually compute such optimal allocations of resources in heterogeneous network environments.

Although the decentralized resource allocation problem has received attention in

the computer networks literature, the major results thus far have come from the field of mathematical economics. In the past three decades, mathematical economists have developed elegant models describing how goods are produced and distributed among agents in an economy. The work of Arrow, Hurwicz and Debreu [Arrow and Debreu 54, Arrow et al. 59, Debreu 59, Arrow and Hahn 71], has been of considerable importance in the development of these models. In particular, it has been shown that under certain assumptions, if agents act as selfish, utility-maximizing entities and negotiate their resource demand conflicts through an iterative supply and demand pricing algorithm, an equilibrium distribution of resources exists, can be computed and is optimal. It is from this work that we have drawn many of the ideas and techniques presented in subsequent sections.

6.3. The Multiple Access Environment as a Perfectly Competitive Economic Marketplace

In this section we describe the decentralized algorithm used to solve the fictitious resource allocation problem. First, the fictitious resources are defined and their relationships to the transmission behavior of the stations in the multiple access network are discussed. The important concepts from the microeconomic model of perfect competition (including utility, demand, prices and price adjustment) are then presented and related to the problem of channel sharing in a multiple access environment. The decentralized resource sharing algorithm itself is then presented and finally, several aspects of the algorithm, including its optimality properties and decentralized nature are described and discussed. Since this fictitious resource allocation problem can be used to solve for the optimal transmission behavior of several multiple access protocols (including Slotted Aloha and the time window protocol), the algorithmic description is purposefully presented in its general form, without reference to any specific multiple access protocol. In sections 6.4 and 6.5 this algorithm will be specialized to the cases of the Slotted Aloha and time window protocols.

6.3.1. Network Resources

In the network economy, each agent is endowed with some initial amount of network "resources" and the total amount of these "resources" in the network is assumed to be fixed. The production of additional resources is not possible and thus the only economic activity available is the exchange of these resources. As we will see, agents are motivated to enter into the network economy by the possibility of exchanging their initial distribution of resources for some alternate, more "useful" resource distribution.

In the most general case, the network resources themselves may be tangible resources such as buffers, communication links, and computation time or, as previously discussed, may be *fictitious* resources such as the probability of transmitting over a channel or permission to access a particular database file. In the multiple access networks we will examine, the fictitious network resources will be related to the stations' probability of transmission. If such a notion is adopted, previous work on selfish optimization in multiple access networks [Yemini and Kleinrock 79, Brooks 83] can be interpreted as stating that a station should trade enough of its access rights (i.e., not transmit and remain silent) so as to balance its loss in throughput (by remaining silent) with an equal amount of observed throughput achieved by the other network stations with which it could have interfered. The computation of the distribution of resources which actually achieves such a network-wide balance will be addressed in the following sections.

Our present definition of resources within a multiple access network economy will be related to the above suggested definition in that channel access privileges or, equivalently, the probability of transmitting (and thus potentially interfering with other network agents) are the fictitious resources to be exchanged. However, in order to devise a distributed algorithm for actually computing the optimal exchange of these resources, we must adopt a notion of a resource which reflects the individual interactions between the individual network agents. The reason for this is clear. The previous characterizations of selfish optimization

specify a balance between an individual agent and the "remainder of the network". However, no such entity as the "remainder of the network" really exists; an agent can only interact with other single agents and it is from these individual interactions that an agent's effect on the remainder of the network is determined.

Thus, for each *ordered* pair of distinct network agents, i and j, let us define a fictitious network resource known as the $i \rightarrow j$ *transmission potential* between these two agents, as shown in figure 6-1. If there are N agents, there will then be N^2-N such transmission potentials in the network economy. The total amount of each $i \rightarrow j$ transmission potential in the network is 1.0 and stations i and j are the only stations which can own any of an $i \rightarrow j$ transmission potential.

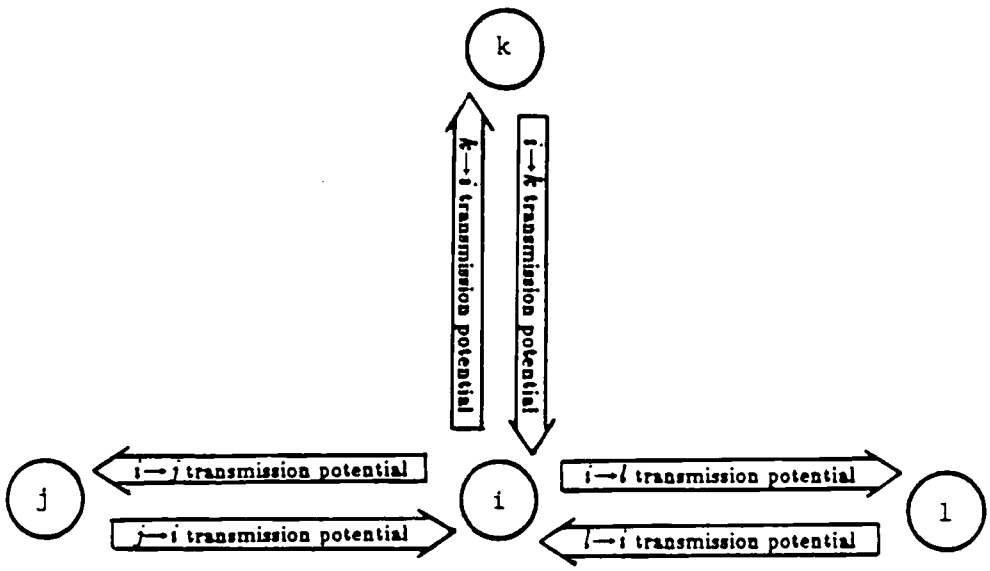


Figure 6-1: Transmission potentials

In the Slotted Aloha and time window protocols examined in sections 6.4 and 6.5, time is divided into slots (fixed length slots in Slotted Aloha and variable length slots in the time window protocol). In these protocols, each station must determine the fraction of slots in which it will attempt a message transmission or equivalently (under the assumptions of sections 6.4 and 6.5) determine its steady state probability of transmission during a time slot. Recall that these fictitious

transmission potentials have been introduced as an artifice to permit the stations to compute the optimal values for these transmission probabilities. In order to understand the role of the transmission potentials in this process, it is necessary to examine their significance to the stations which can own these resources.

Each $i \rightarrow j$ transmission potential is used by stations i and j to determine the steady state transmission probability of station i. During each step in the resource allocation procedure, stations i and j will be required to formulate a *demand* for this fictitious resource. Station i's demand for the $i \rightarrow j$ transmission potential, which can be any value in [0,1], is denoted $X^{i}_{i \rightarrow j}$ and represents i's demand to j that it (station i) be permitted to transmit with a steady state probability of $X^{i}_{i \rightarrow j}$. The demand for this same $i \rightarrow j$ transmission potential formulated by station j has the converse interpretation. Station j's demand is denoted $X^{j}_{i \rightarrow j}$ and represents j's demand to i that station i *not* transmit with a steady state probability of $X^{j}_{i \rightarrow j}$. $X^{j}_{i \rightarrow j}$ can thus be thought of as j's demand for the "silence" of i and can also be any value in [0,1]. (In our notation, the demand superscript thus identifies the station demanding the transmission potential specified by the demand subscript. Also, the station number preceding the arrow in the subscript indicates that station whose probability of transmission is affected by the given transmission potential).

The above definition focuses on the significance of a single transmission potential to the two stations which can own this resource. The transmission potential demands formulated by a single station, i, are summarized below:

6.3.2. Decentralized Computation of the Optimal Distribution of Resources

In the process of demand formulation, stations i and j may formulate inconsistent demands in the sense that $X^{i}_{i \rightarrow j} + X^{j}_{i \rightarrow j} \neq 1$. In this section, we examine a decentralized algorithm for resolving such conflicting demands in such a way that the final allocation of transmission potentials is known to be Pareto optimal.

The algorithm itself is an iterative process of decentralized demand formulation

Demands by station i	Interpretation
$X_{i\rightarrow 1}^{i}, \ldots, X_{i\rightarrow N}^{i}$	Each $X_{i\rightarrow j}^{i}$ represents station i's demand to station j that it (station i) transmit with a steady state probability of $X_{i\rightarrow j}$.
$X_{1\rightarrow i}^{i}, \ldots, X_{N\rightarrow i}^{i}$	Each $X_{j\rightarrow i}^{i}$ represents station i's demand to station j that station j *not* transmit with a steady state probability of $X_{j\rightarrow i}$

Figure 6-2: Station i's demand for transmission potentials

and price modification for computing a set of resource "prices" at which the network-wide total demand for resources exactly equals the total amount of resources in the network. In this algorithm, no transmission potentials are exchanged nor are any normal message transmissions attempted until this set of equilibrium prices has been determined. The individual aspects of the pricing algorithm will be examined in sections 6.3.2.1 through 6.3.2.4.; the algorithm itself will then be summarized in section 6.3.2.5.

6.3.2.1 The Initial Distribution of Goods and the Motivation for Change

Each station enters the network economy with an initial steady state probability of transmission and some initial amount of the "silence" of the other stations in the network. This initial allocation of transmission potentials can either be imposed by a higher level network manager or a station may enter the network with a default value for its initial allocation of resources. With no loss of generality, we will assume that station i only has an initial allocation of the $i\rightarrow 1,\ldots,i\rightarrow N$ and $1\rightarrow i,\ldots,N\rightarrow i$ transmission potentials which directly affect its performance; the notation $\overline{X}_{i\rightarrow j}^{i}$ will be used to indicate the amount of the $i\rightarrow j$ transmission potential *initially* owned by i. We will also assume that the initial allocation of transmission potentials is consistent in the sense that $\overline{X}_{i\rightarrow j}^{i} + \overline{X}_{i\rightarrow j}^{j} = 1$ for all i,j.

As discussed earlier, a station is motivated to enter into the network economy by the possibility of exchanging its initial allocation of resources for some alternate, more "useful" distribution of resources. Consider, for example, a heavily loaded Slotted Aloha network in which each station always has a message to send and suppose the initial allocation of transmission potentials is such that each station would transmit at the beginning of every slot with probability 1 (i.e., station i begins with each $\overline{X}^{i}_{i \rightarrow j}$ equal to 1 and each $\overline{X}^{i}_{j \rightarrow i}$ equal to 0). If no redistribution of transmission potentials takes place, each station would attempt to transmit in every slot, collisions would always occur and the throughput per station would be zero. However, if stations agree to "trade" some of their own transmission probability in return for the silence of other stations (i.e., demand an alternate allocation of resources in which their own probability of transmission is smaller but the silence demanded of other stations is larger), a non-zero throughput can be realized. Moreover, all stations could benefit from such a redistribution of resources. Thus, the selfish motivation for the possible exchange of resources is clear.

The above example also illustrates an important aspect of the resource allocation process: conservation of resources versus conservation of utility. Clearly, one station's decrease in a resource is necessarily another station's increase in that resource. Thus it might seem that a conservation law would imply that *every* resource distribution is Pareto optimal since no trade could make some stations better off without making other stations worse off. The important observation is that such a conservation law applies to the resources themselves but *not* to the utility or performance levels of the stations. Resources are always conserved; as demonstrated in the above example, however, the overall utility can be either decreased or (hopefully) increased through the redistribution of resources.

6.3.2.2. Prices and a Station's Worth

Let us define the price, $P_{i \rightarrow j}$, associated with the $i \rightarrow j$ transmission potential simply as some positive, real-valued number and the N^2-N dimensional vector of these prices as the *price vector* for the resources in the network economy. For

a given price vector, the initial distribution of transmission potentials held by station i is defined to have a "value" or "worth" defined by:

$$worth_i = \sum_{j=1}^{N} (P_{i \rightarrow j} \ \overline{X}^{i}_{i \rightarrow j} + P_{j \rightarrow i} \ \overline{X}^{i}_{j \rightarrow i})$$

Note that as prices change in the network economy, a station's "worth" may also change. As we will see, the price vector will play an important role in restricting the demands that a station can make by imposing a *budget constraint* on these demands. Informally, this budget constraint will require that a station not "spend" more (in demanding an alternate distribution of resources) than it is currently worth, i.e., a station will only be able to demand a distribution of resources which, given the current prices, "costs" no more than its current worth, as defined above.

6.3.2.3. Selfish Utility Maximization

At each step in the iterative pricing procedure, each station i is required to formulate a *demand* $X^{i}_{i \rightarrow 1}, \ldots, X^{i}_{i \rightarrow N}, X^{i}_{1 \rightarrow i}, \ldots, X^{i}_{N \rightarrow i}$ for those transmission potentials which affect its performance. In this subsection, we examine the process by which a station formulates these demands.

Since $X^{i}_{1 \rightarrow i}, \ldots, X^{i}_{N \rightarrow i}$ are station i's demands for the other stations' silence, *if i actually receives its demanded allocation of transmission potentials*, then the $X^{i}_{1 \rightarrow i}, \ldots, X^{i}_{N \rightarrow i}$ will be the steady state probabilities with which other stations do *not* attempt message transmissions. The actual probability, $P\text{-}succ_i$, that i successfully transmits a message in a slot (when it attempts to do so) will depend on the transmission probabilities of the other stations, the manner in which collisions are resolved and the definition of a slot itself, i.e., the specific form of $P\text{-}succ_i$ will vary from one multiple access protocol to another. Thus, for now we simply note that $P\text{-}succ_i$ will be *functionally* dependent only on the probabilities that the other N-1 stations are not attempting a message transmission in the same slot as i. Thus:

$$P\text{-}succ_i = P\text{-}succ_i(X^{i}_{1 \rightarrow i}, \ldots, X^{i}_{N \rightarrow i}) \qquad (6.1)$$

This probability of a successful transmission (when attempted) together with i's actual probability of transmission determine i's throughput of successfully transmitted messages. Recall that the $X^{i}_{i\rightarrow 1}, \ldots, X^{i}_{i\rightarrow N}$ are i's demands to each of the other N-1 stations for those transmission potentials which determine its own probability of transmission. Thus, if i actually receives its demanded allocation of transmission potentials, its throughput of successfully transmitted messages, $thruput_i$, will have the functional dependence:

$$thruput_i = thruput_i(X^{i}_{1\rightarrow i}, \ldots, X^{i}_{N\rightarrow i}, X^{i}_{i\rightarrow 1}, \ldots, X^{i}_{i\rightarrow N}) \qquad (6.2)$$

Given equations 6.1 and 6.2, each station formulates its demand for resources as follows. If λ_i is the rate at which messages are generated at station i, i's demand for resources, given the set of current prices, is that distribution of resources which:

maximizes:

$$thruput_i(X^{i}_{1\rightarrow i}, \ldots, X^{i}_{N\rightarrow i}, X^{i}_{i\rightarrow 1}, \ldots, X^{i}_{i\rightarrow N}) \qquad (6.3)$$

subject to the constraints:

$$thruput_i(X^{i}_{1\rightarrow i}, \ldots, X^{i}_{N\rightarrow i}, X^{i}_{i\rightarrow 1}, \ldots, X^{i}_{i\rightarrow N}) \leq \lambda_i \qquad (6.4)$$

$$\sum_{j=1}^{N} (P_{i\rightarrow j}\, X^{i}_{i\rightarrow j} + P_{j\rightarrow i}\, X^{i}_{j\rightarrow i}) \leq \sum_{j=1}^{N} (P_{i\rightarrow j}\, \bar{X}^{i}_{i\rightarrow j} + P_{j\rightarrow i}\, \bar{X}^{i}_{j\rightarrow i}) \qquad (6.5)$$

In the case that more than one resource distribution satisfies 6.3 through 6.5, i selects that distribution which also:

maximizes: $\quad P\text{-}succ_i(X^{i}_{1\rightarrow i}, \ldots, X^{i}_{N\rightarrow i}) \qquad (6.6)$

Equations 6.3, 6.4 and 6.6 define the *utility* of a particular allocation of transmission potentials to station i. They indicate that a station should first attempt to secure enough of the transmission potentials to insure that all its messages will be transmitted (i.e., that its message throughput equals the rate at which messages are being generated locally). Among those allocations which insure this condition, a station should then choose that distribution which maximizes its probability of a successful message transmission (when attempted) or equivalently, minimizes the number of times a message must be retransmitted.

Equation 6.5 is simply the budget constraint that the cost of a demanded allocation can not exceed the current worth of the station's initial allocation of transmission potentials.

For the Slotted Aloha and time window protocols, we will see that equations 6.3 through 6.6 are sufficient to uniquely determine the resource demands of station i. Note that these equations also formally define the "selfish" behavior of an agent. Demand is formulated based only on a station's own performance considerations; the effects of its demands on the performance of other stations in the network are not considered in the demand formulation.

6.3.2.4. Price Adjustment

As previously discussed, the resource demands of the network agents (formulated via equations 6.3 - 6.6) may be unbalanced in the sense that the total demand by stations i and j for the $i \rightarrow j$ transmission potential does not equal 1. A total demand for the $i \rightarrow j$ transmission potential exceeding 1 indicates that j is demanding more silence from i than i is willing to provide; if the demand is less than 1, i is willing to provide more silence than j is willing to acquire. The balancing of such demands is iteratively accomplished through the familiar mechanism of price adjustment.

Specifically, if the total demand of stations i and j for $i \rightarrow j$ transmission potential exceeds 1, the price, $P_{i \rightarrow j}$, of this resource is increased. Conversely, if the total demand is less than 1, $P_{i \rightarrow j}$ is decreased. The rationale behind the adjustment process is simple: if the price of a resource is increased/decreased, stations will demand less/more of that resource (due to their budget constraints) and thus the total demand for the resource will decrease/increase. In summary:

$$\Delta P_{i \rightarrow j} = c_{i \rightarrow j} * \{ 1.0 - (X^{i}_{i \rightarrow j} + X^{j}_{i \rightarrow j}) \} \qquad (6.7)$$

where $c_{i \rightarrow j}$ is a suitably chosen, problem-specific, constant. The bracketed quantity on the right hand side of equation 6.7 is typically known as the *excess demand* for the $i \rightarrow j$ transmission potential.

Once the new prices have been computed, the agents again formulate their resource demands (equations 6.3 - 6.6) at the new set of prices. Once the new demands have been computed, the prices are again changed as specified by equation 6.7. This iterative process of demand formulation and subsequent price adjustment, known in the microeconomics literature as *tatonnement* [Arrow and Hahn 71], continues until the total demand for each transmission potential exactly equals 1 or, equivalently, until $\Delta P_{i \rightarrow j}$ equals 0 for all i, j. At this point, the agents' demands are known to be mutually compatible and the agents are thus then allocated their demanded amount of resources. The convergence of this process will be discussed in section 6.3.3; the computation of $\Delta P_{i \rightarrow j}$ in a network environment will be discussed in section 6.3.4.

6.3.2.5. Summary

The major elements of the decentralized resource allocation process have been individually presented in the preceding subsections. A summary of this process is given below:

1. Each station i begins with an initial amount, $\bar{X}^{i}_{i \rightarrow 1}, \ldots, \bar{X}^{i}_{i \rightarrow N}, \bar{X}^{i}_{1 \rightarrow i}, \ldots, \bar{X}_{N \rightarrow i}$ of the $i \rightarrow 1, \ldots, i \rightarrow N$ $1 \rightarrow i, \ldots, N \rightarrow i$ transmission potentials and an initial set of prices for these resources.

2. <u>do</u>

   - Each station, i computes its demands $X^{i}_{i \rightarrow 1}, \ldots, X^{i}_{i \rightarrow N}, X^{i}_{1 \rightarrow i}, \ldots, X_{N \rightarrow i}$ for transmission potentials as that set of resources which maximizes its utility (performance) (equations 6.3, 6.4, 6.6) subject to its budget constraint (equation 6.5).

   - The price adjustment, $\Delta P_{i \rightarrow j}$, for the $i \rightarrow j$ transmission potential is computed as a function of the excess demand for this resource, as specified in equation 6.7.

   <u>until</u> $\Delta P_{i \rightarrow j} = 0$ for all i, j.

Algorithm 6.1

Once $\Delta P_{i \rightarrow j}$ equals zero for all i, j, the total demand for each transmission potential exactly equals 1.0 and the agents are then allocated their demanded amount of resources. Thus, *once algorithm 6.1 converges, the demands* $\{ X^{i}_{i \rightarrow 1},$

$\ldots, X^{i}_{i\rightarrow N}, X^{i}_{1\rightarrow i}, \ldots, X^{i}_{N\rightarrow i}\}$ *indicate the amount of transmission potentials actually allocated to agent* i. In subsequent sections, we will thus refer to $X^{i}_{i\rightarrow j}$ as an amount of the $i\rightarrow j$ transmission potential actually allocated to agent i once algorithm 6.1 has converged.

6.3.3. Existence and Computability of the Optimal Distribution

The pricing algorithm and selfish optimization procedure discussed in the previous section defines the individual behavior of the stations in the multiple access network. In this section we examine the collective results of this individual behavior. The most important result is a well-known theorem from microeconomics; it states that when a demanded distribution and its associated price vector are in equilibrium (as specified by the termination criteria for algorithm 6.1), then that distribution of resources is Pareto optimal. Recall that at this equilibrium point, known as a *competitive equilibrium*, each agent has demanded a distribution of transmission potentials which, given the current prices, maximizes its utility (as defined by equations 6.3, 6.4, and 6.6) subject to its budget constraint (6.5).

Theorem: If the network-wide demand, $\mathbf{X}$, for transmission potentials and a non-negative price vector $\mathbf{P}$ constitute a competitive equilibrium, i.e., if algorithm 6.1 terminates, then $\mathbf{X}$ is a Pareto optimal distribution of transmission potentials.

Proof: Suppose $\mathbf{X}$ is not Pareto optimal. Then there exists another distribution $\mathbf{X}'$ which is Pareto superior to $\mathbf{X}$, i.e.:

$$\begin{aligned} & \mathbf{X}' \succeq_j \mathbf{X} && \text{for all } j \\ & \mathbf{X}' \succ_k \mathbf{X} && \text{for at least one } k \\ \text{and for all } i,j: \quad & X'^{i}_{i\rightarrow j} + X'^{j}_{i\rightarrow j} = 1 && (6.8) \end{aligned}$$

where $\mathbf{X}' \succeq_k \mathbf{X}$ indicates that the transmission potentials in the network-wide distribution $\mathbf{X}'$ are strictly preferred (in the sense of equations 6.3, 6.4 and 6.6) by station k to in $\mathbf{X}$. Equations 6.3, 6.4 and 6.6 thus serve to define an ordinal utility order, $\succeq$, over the elements in the set of possible resource allocations.

Following our previous conventions, $X'^{i}_{i\rightarrow j}$ represents the amount of the $i\rightarrow j$ transmission potential received by i in the network-wide distribution $\mathbf{X}'$.

Let us now focus on agent k. Since, by hypothesis, $\mathbf{X}$ is a competitive equilibrium, the distribution of resources k receives in $\mathbf{X}'$ must fall outside of k's budget constraint (given the price vector $\mathbf{P}$) for otherwise k itself would have demanded that distribution as a result of its maximizing behavior. Thus, for agent k:

$$\sum_{j=1}^{N} (P_{k\rightarrow j}\ X'^{k}_{k\rightarrow j} + P_{j\rightarrow k}\ X'^{k}_{j\rightarrow k}\ ) > \sum_{j=1}^{N} (P_{k\rightarrow j}\ \bar{X}^{k}_{k\rightarrow j} + P_{j\rightarrow k}\ \bar{X}^{k}_{j\rightarrow k}\ )$$

or rearranging:

$$\text{for } k\text{:}\quad \sum_{j=1}^{N} (P_{k\rightarrow j}\ (X'^{k}_{k\rightarrow j} - \bar{X}^{k}_{k\rightarrow j}\ ) + P_{j\rightarrow k}\ (X'^{k}_{j\rightarrow k} - \bar{X}^{k}_{j\rightarrow k}\ )\ ) > 0$$

and for the remaining agents, i:

$$\text{for all } i \neq k\text{:}\quad \sum_{j=1}^{N} (P_{i\rightarrow j}\ (X'^{i}_{i\rightarrow j} - \bar{X}^{i}_{i\rightarrow j}\ ) + P_{j\rightarrow i}\ (X'^{i}_{j\rightarrow i} - \bar{X}^{i}_{j\rightarrow i}\ )\ ) \geq 0$$

Summing the above *set* of equations with the single equation for agent k and rearranging gives:

$$\sum_{j=1}^{N} \left\{ \sum_{i=1}^{N} (P_{i\rightarrow j}\ (X'^{i}_{i\rightarrow j} + X'^{j}_{i\rightarrow j} - 1) + P_{j\rightarrow i}\ (X'^{i}_{j\rightarrow i} + X'^{j}_{j\rightarrow i} - 1))\right\} > 0$$

Since the prices are non-negative, the above inequality implies that at least one set of i,j satisfies $X'^{i}_{i\rightarrow j} + X'^{j}_{i\rightarrow j} > 1.0$, which contradicts equation 6.8. Thus, a Pareto superior $\mathbf{X}'$ cannot exist and hence $\mathbf{X}$ is Pareto optimal. **Q.E.D.**

The above theorem states a static property of the iterative pricing algorithm: if algorithm 6.1 converges, the individual demands of the agents are compatible and the resulting distribution of transmission potentials is known to be Pareto optimal. Unfortunately, determining the necessary conditions (the initial price vector, the functional forms of $thruput_i$ and $p\text{-}succ_i$) under which algorithm 6.1 converges is an extremely difficult problem. As we will see, the few available results [Negishi 62, Arrow and Hahn 71] for sufficient conditions for convergence

are violated at each station i in a multiple access network by the dependence of the demand on a term, min $\{ X_{i\_1}^{i}, \ldots, X_{i\_N}^{i} \}$.

The primary complicating factor in examining the convergence properties of algorithm 6.1, given the specific forms of $P\text{-}succ_i$ and $thruput_i$, is that the agent's demands at each iteration (and hence the price changes) are computed as the result of a maximization process. In practice, the functional forms of $thruput_i$ and $p\text{-}succ_i$ are non-trivial and a closed-form solution to the maximization problem at each iteration is usually not available. Rather, the individual demands must be numerically determined, typically (as we will see) as the solution of a set of simultaneous non-linear equations which arise from the maximization process. An additional complication in examining the convergence of algorithm 6.1 results from properties of $\succeq$, the preference relation induced over the set of feasible distributions by equation 6.3, 6.4 and 6.6. The type of preference relation defined by these equations is known as a lexicographic order and it can be shown [Hildenbrand and Kirman 76] that such relations are non-continuous in the sense that the sets $\{ X \mid X \succeq X' \}$ $\{ X \mid X' \succeq X \}$ are not closed for all allocations, X'. Given the lack of such elementary continuity properties, examining the convergence properties of algorithm 6.1 is beyond the scope of this thesis. However, in the following sections, we will discuss the results of our experience with the convergence of algorithm 6.1 for the Slotted Aloha and time window protocols. These results, however, are purely experimental; a theoretical examination of the convergence problem remains a topic for future research.

6.3.4. How Decentralized is Decentralized?

Before applying algorithm 6.1 to the Slotted Aloha and time window protocols, let us examine the decentralized and non-decentralized aspects of the algorithm itself. Clearly, information about the utility functions at the various stations is completely decentralized; a station knows its own local utility function but has no information concerning the utility functions of the other stations in the network. Since each station also locally performs the maximization step, the computation of its demand at each iteration is also decentralized (and thus

during each iteration of algorithm 6.1, the N local maximizations can be computed in parallel).

As in most distributed systems, however, some information must be shared among the network stations. Several pieces of shared information are involved in the initialization of algorithm 6.1. The initial price of the $i \rightarrow j$ transmission potential and the value of $c_{i \rightarrow j}$ (in equation 6.7) must be known to all agents which demand this resource. In addition, although individual stations need only know their own initial distribution of resources, shared information is also required to insure the consistency of the initial distribution itself (i.e., that $\overline{X}^{i}_{i \rightarrow j} + \overline{X}^{j}_{i \rightarrow j} = 1$ for all i,j). As we will see, determining the initial distribution of resources is equivalent to determining the relative priorities of the stations. These priority levels can either be set by a higher level network mechanism or can be determined among the stations themselves according to some set of predefined rules (e.g., i and j determine their initial allocations of the $i \rightarrow j$ and $j \rightarrow i$ transmission potentials according to the ratio of their message generation rates).

Finally, during the execution of the algorithm itself, each station must also determine the network-wide demand for those transmission potentials which it itself is demanding, in order to compute the relevant price changes. Thus, although each station locally computes price changes, its demand for a resource must be communicated to all other stations which also demand this resource. In a single-hop multiple access network, a simple way to accomplish this is for i to broadcast its N-1 demands $\{ X^{i}_{1 \rightarrow i}, \ldots, X^{i}_{N \rightarrow i} \}$ for the silence of other stations as well as its single demand (as discussed in section 6.4.1) for its own probability of transmission. In a single-hop environment, this information is necessarily shared globally.

Interestingly, in a multi-hop environment, however, this information need not be shared globally. To see this, consider the 4-hop network shown in figure 6-3. The four broadcast groups in this figure indicate those stations which can directly communicate with each other. For example, station 1's transmissions are heard directly by stations 2 and 3 (since they are in 1's broadcast group), but

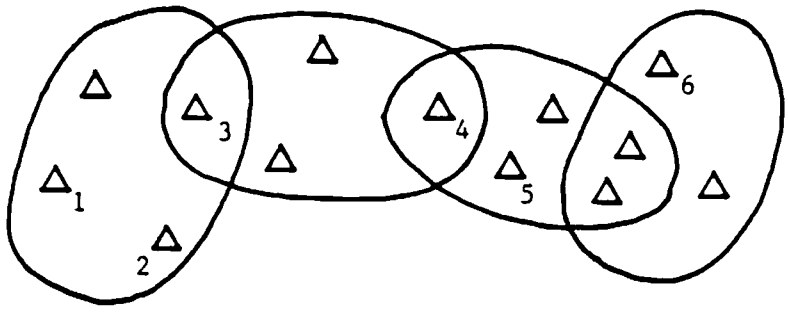


Figure 6-3: A 4-hop multiple access network

not by stations 4, 5 or 6 and only those stations which are in a broadcast group containing a station which is also in station 1's broadcast group can interfere with a transmission by station 1. Thus, although station 4 can interfere with a transmission by station 1 to station 3, station 5 can not do so. Since a station's performance is thus only affected by those stations which can directly transmit to a station in its broadcast group, a transmission potential need only be defined between it and those stations with which it can directly interfere. For example, in the network in figure 6-3 a transmission potential would be defined between stations 1 and 2, and 1 and 4 (since 1 and 4 can interfere at station 3), but no such resource need be defined between 1 and 5 or 1 and 6 since these stations have no direct effect on the message transmissions of station 1. This example illustrates an important aspect of the resource allocation process: *even if the resources are mutually coupled, as the resources themselves become localized, so too does the interaction involving their distribution.* That is, as the resources themselves obtain a degree of spatial locality, the interaction required for the allocation of these resources also naturally decomposes along these spatial lines.

6.4. An Application: Slotted Aloha

In this section we examine the use of algorithm 6.1 to determine the Pareto optimal steady state transmission probabilities in a Slotted Aloha [Abramson 73] multiple access network.

As previously discussed in chapter 2, in the Slotted Aloha protocol, time is divided into fixed length slots, with the length of each slot equal to a single message transmission time.

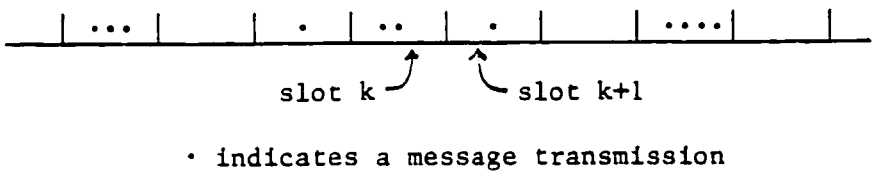


Figure 6-4: Time slots in the Slotted Aloha protocol

All message transmissions must start at the beginning of a time slot and thus terminate at the end of the slot. If two or more stations attempt to transmit messages in the same slot (e.g., slot k in figure 6-4), their messages interfere (collide) and must be scheduled for retransmission at some later time. A message is thus sent successfully if it is the only message transmitted during a time slot (e.g., slot k+1 in figure 6-4).

In order to apply algorithm 6.1 to the Slotted Aloha or any other multiple access protocol, we need only specify the stations' local utility functions, $thruput_i$ and $P\text{-}succ_i$ in equations 6.2 and 6.4; these functions will be presented in section 6.4.1. In the remaining subsections, we present the performance results for a small (4 station) Slotted Aloha network in which the protocol's optimum transmission probabilities have been computed using algorithm 6.1. These results are shown to both coincide with and extend the previously reported results of Abramson [Abramson 73].

6.4.1. The Utility Functions

Should station i receive the transmission potentials it demands of the other stations, then each $X^{i}_{j \rightarrow i}$ in $\{ X^{i}_{1 \rightarrow i}, \ldots, X^{i}_{N \rightarrow i} \}$ represents the steady state probability that station j will *not* attempt a message transmission at the beginning of a time slot. If we make the standard assumption regarding the independence of the times at which stations actually attempt their transmissions, the probability that agent i is successful when it attempts to send a message is simply the probability that the remaining N-1 stations are silent during that slot. Thus, for th· Slotted Aloha protocol:

$$P\text{-}succ_i = \prod_{j=1}^{N} X^{i}_{j \rightarrow i} \tag{6.9}$$

Station i's throughput (messages successfully transmitted/slot) is simply the probability that i attempts a transmission during a slot multiplied by $P\text{-}succ_i$. Recall that $\{ X^{i}_{i \rightarrow 1}, \ldots, X^{i}_{i \rightarrow N} \}$ are i's demands to stations 1,..,N for the $i \rightarrow 1, \ldots, i \rightarrow N$ transmission potentials which dictate its own transmission probability. If i is required to actually transmit with a probability given by the minimum of $\{X^{i}_{i \rightarrow j}\}$ (equivalently, to honor the maximum amount of its silence which it is willing to let other stations own), the message throughput at station i is given by:

$$thruput_i = \min_{j=1,N} \{ X^{i}_{i \rightarrow j} \} \cdot \prod_{j=1}^{N} X^{i}_{j \rightarrow i} \tag{6.10}$$

Note that the min{} term in equation 6.10 will cause i to formulate its demands such that the $\{X^{i}_{i \rightarrow j}\}$ are all equal, since otherwise i would be wasting part of its budget in demanding extra (unusable) amounts of transmission potentials.

During each iteration of algorithm 6.1, each station thus performs the maximization step (equation 6.3 through 6.6) using equations 6.9 and 6.10. This step can be reformulated as the following constrained optimization problem:

maximize: $P\text{-}succ_i$ (6.11)

subject to: (1) the budget constraint, equation 3.5 (6.12)

(2) $thruput_i \leq^{*} \lambda_i$ (6.13)

where λ_i is the message generation rate at station i and the relational operator $\leq^*$ indicates that *P-succ*$_i$ should be maximized subject to the constraint that the associated throughput be as close to, but not greater than, λ_i. Thus, a distribution of transmission potentials for which the throughput of station i is equal to λ_i is to be preferred by station i to any other distribution which has a different associated throughput, regardless of the value of *P-succ*$_i$.

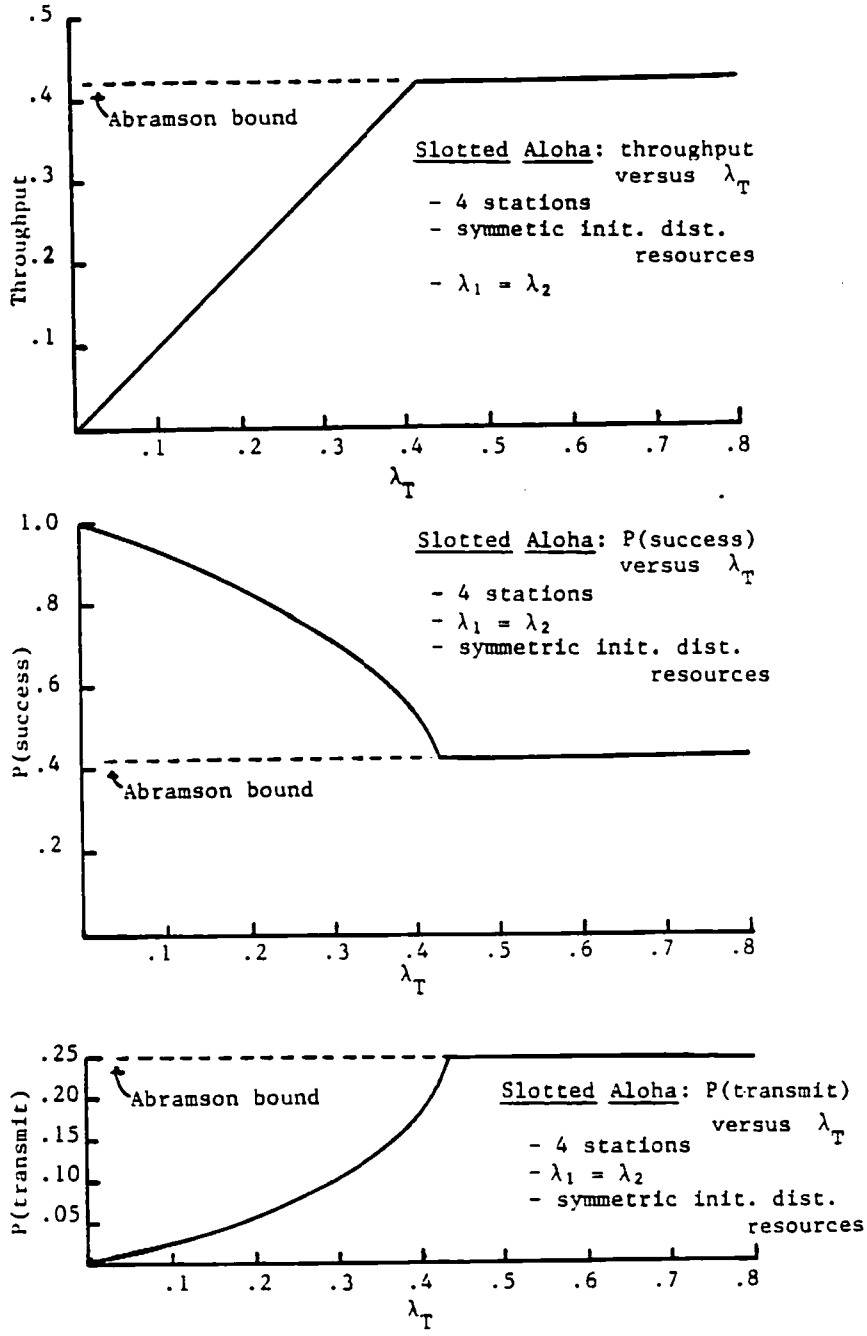
As previously discussed, the solution to the above constrained optimization problem determines a station's demand at the current set of prices. The optimization problem itself can be solved using standard techniques, such as the Lagrange method of undetermined multipliers. Unfortunately, a simple closed-form solution to the above maximization problem is not possible and numerical techniques must be employed to solve the set of simultaneous equations resulting from the Lagrangian technique.

6.4.2. Perfectly Symmetric Stations

Figure 6-5 shows the results of using algorithm 6.1 to determine the Pareto optimal set of transmission probabilities in a completely symmetric four station Slotted Aloha network for various message generation rates. In this example, the rates at which stations generate messages for transmission (messages generated/slot) are identical ($\lambda_i = \lambda_T/4$, where λ_T is the total system-wide message generation rate) and each station begins with a symmetric allocation of 0.5 of each transmission potential which it demands. λ_T is plotted along the horizontal axis and the total system throughput, P(success) (the probability of a successful message transmission, when attempted), and P(transmit) (the probability of message transmission) are plotted along the vertical axes.

The results indicate that for any message generation rate, even though each station can initially transmit with a probability of 0.5 in each slot (due to the initial allocation of transmission potentials), a station will always demand (and eventually receive) an alternate allocation of transmission potentials for which its own transmission probability is smaller but the silence of the other stations is larger. In a lightly loaded network, this is an obvious result. Informally, if λ_i

Figure 6-5: Slotted Aloha: the completely symmetric case



is small enough so that a station need not transmit in half of the slots, it should obviously "trade away" some of its own unneeded probability of transmission in return for the silence of other stations. In a heavily loaded network, however, a station will still trade away some of its probability of transmission, even in the limiting case that it always had a message to send! This is because it is in a station's own selfish best interest to flow control itself (trade away some of its initial probability of transmission) in exchange for similar flow control by the other stations in the network. Thus, flow control arises naturally from a station's pursuit of its own selfish interests.

The performance results themselves indicate that for a system-wide message generation rate less than 0.42, the stations set their transmission probabilities such that the total system throughput equals the rate at which messages are being generated, i.e., every message which is generated at a station is eventually transmitted. As the message generation rate increases, each station increases P(transmit) such that, although P(success) is decreasing at the same time, it is still possible to transmit every message. However, once λ_T exceeds 0.42, P(transmit) remains constant, the system throughput thus remains constant and an increasing message generation rate results in increasing message loss. For $\lambda_T > 0.42$, increasing P(transmit) past 0.23 would result in a lower throughput due to an increased probability of a message collision. Since such a throughput loss is clearly not in a station's best interest, the final value of P(transmit) will never be greater than 0.23 for $\lambda_T > 0.42$.

As shown in figure 6-5, the system throughput, P(success) and P(transmit) resulting from the decentralized "microeconomic" approach to channel sharing all approach the heavy traffic bounds predicted by Abramson's [Abramson 73] centralized optimization of the finite population Slotted Aloha model. As N (the number of stations) grows large, we expect the point at which the stations exhibit such asymptotic behavior to decrease from $\lambda_T = 0.42$ to $\lambda_T = 0.368$, Abramson's bound for the infinite population Slotted Aloha system. We would also expect the stations' throughput to again match the message generation rate until it reaches its asymptotic value; our reason for this conjecture will be discussed in the following section.

Finally, we note that the ability of the present approach to reproduce previous, known results provides an important check of the equations and numerical methods used to determine a station's demand. It also provides a firm foundation from which other aspects of the "microeconomic" approach can now be explored.

6.4.3. Multiple Priority Levels and Heterogeneous Stations

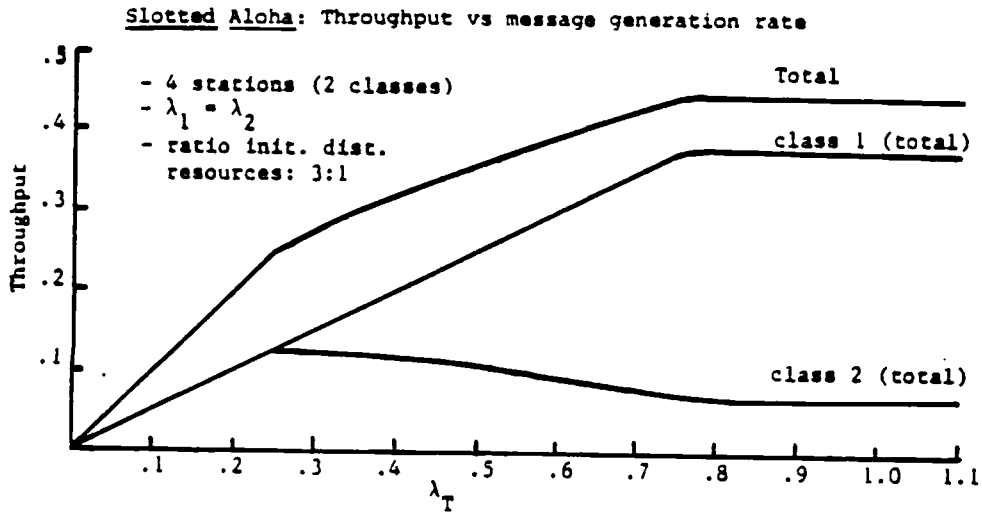
6.4.3.1 Identical Message Generation Rates; Differing Initial Allocations of Resources

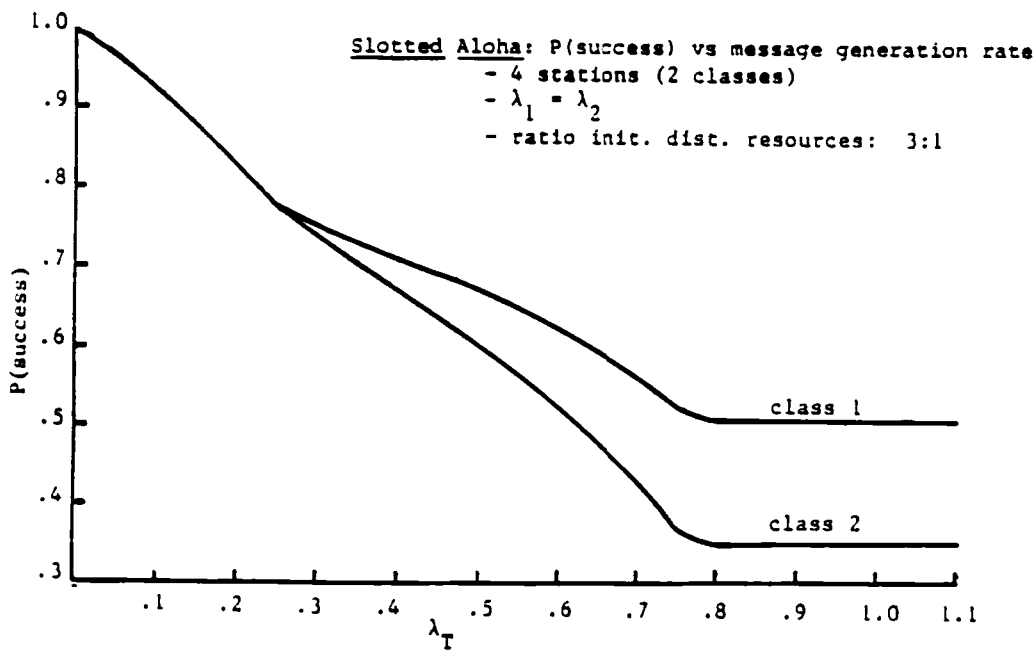
In this section, we examine the effects of asymmetric initial allocations of transmission potentials on the stations' performance. The four stations will be divided into two classes, with 2 stations in each class. The rate at which the stations generate messages will again be identical ($\lambda_i = \lambda_T/4$, for each station i) but the ratio of their initial allocation of transmission potentials will be 3:1. That is, if stations i and j are in different classes, $\overline{X}_{i\rightarrow j}^{\,i} = 0.75$, $\overline{X}_{i\rightarrow j}^{\,j} = 0.25$ and $\overline{X}_{j\rightarrow i}^{\,i} = 0.75$, $\overline{X}_{j\rightarrow i}^{\,j} = 0.25$. If i and j are in the same class, $\overline{X}_{i\rightarrow j}^{\,i} = \overline{X}_{i\rightarrow j}^{\,j} = \overline{X}_{j\rightarrow i}^{\,i} = \overline{X}_{j\rightarrow i}^{\,j} = 0.5$.

Figure 6-6 shows the performance results for a 3:1 initial allocation of transmission potentials. For $\lambda_T < 0.8$, the throughput of class 1 equals the rate at which messages are being generated; for $\lambda_T > 0.8$, the throughput and P(success) remain constant. Thus the general behavior of class 1 stations is similar to the perfectly symmetric case, except that they flow control themselves at a larger value of λ_T. (Recall that in this case, flow control is indicated by the fact that a station's throughput is less than its message generation rate.)

A more dramatic change occurs in the performance of the class 2 stations. For $\lambda_T < 0.25$, the throughput of class 2 equals the rate at which messages are being generated. However, for $\lambda_T > 0.25$, class 2's throughput actually *decreases* with an increasing message generation rate. This is because as λ_T increases, the message generation rate of stations in class 1 increases, and their demands for the transmission potentials which dictate their transmission probability thus also

Figure 6-6: Slotted Aloha: identical message generation rates; 3:1 initial allocation of transmission potentials





increase. Since class 1 stations start with a larger initial allocation of transmission potentials, these increasing demands will be satisfied, and, for $\lambda_T > 0.25$, result in a final distribution of transmission potentials in which class 2 stations are not allocated enough resources to transmit all their messages. Unfortunately, we have not yet been able to characterize (in terms of parameters such as the message generation rates, number of stations or initial distribution of transmission potentials) the point at which the performance of class 1 and class 2 stations begin to visibly diverge.

Of course, the increasing demands of the class 1 stations can not be indefinitely satisfied, since the transmission potentials initially held by class 2 stations guarantee some minimum performance level, regardless of the message generation rate. Note that the class 2 stations approach this minimum performance level as the class 1 stations approach their maximum performance level and *both* classes start flow-controlling themselves at that point where their throughput begins to exhibit its asymptotic behavior.

Several of the above observations can be generalized for any number of classes, ratio of message generation rates and initial allocation of transmission potentials by the following claim. Suppose that at the network-wide message generation rate, λ_T^*, all classes of stations have begun flow-controlling themselves. Our claim is that at λ_T^*, all classes of stations have reached their asymptotic throughput values for all $\lambda_T' > \lambda_T^*$. That is, once *all* stations have begun flow controlling themselves, their throughput remains constant for all larger values of the network-wide message generation rate. Note that one implication of the above claim is that there is always at least one class of stations (e.g., class 1 in figure 6-6) whose throughput matches its message generation rate up to the point at which the *entire* system begins to exhibit asymptotic behavior.

To establish this claim, we first note that the throughput of *all* classes cannot be smaller at some λ_T' ($\lambda_T' > \lambda_T^*$) than at λ_T^*, since all stations could simply use the optimum transmission probabilities used at λ_T^* to achieve the same throughput at λ_T'. Thus if the above claim is *not* to hold, there must be at least one class, i, of stations which has a flow-controlled, but larger, throughput

at λ_T' than at λ_T^*. It can be easily shown that no such class, i, of stations can exist, thus establishing the above claim.

Informally, let $S_i(\lambda_T')$ be the throughput of class i when the network-wide message generation rate is λ_T' and let $S_i(\lambda_T^*)$ be the throughput at λ_T' ($\lambda_T' > \lambda_T^*$, $S_i(\lambda_T') > S_i(\lambda_T^*)$ and all classes flow-controlled at λ_T^*, by hypothesis). Since $S_i(\lambda_T') > S_i(\lambda_T^*)$, class i stations are receiving a larger amount of transmission potentials (to support their larger throughput) at λ_T' than at λ_T^*. However, if class i stations can receive enough transmission potentials to support $S_i(\lambda_T')$ at λ_T', they can also receive enough transmission potentials to achieve up to this same level of throughput at λ_T^*, since the resource needs of *all* stations are less at λ_T^* than at λ_T'. The only case in which the class i stations would not actually demand and receive enough transmission potentials to support throughput $S_i(\lambda_T')$ at λ_T^* is if the message generation rate at $\lambda_T^*$is less than $S_i(\lambda_T')$, in which case they could still receive enough transmission potentials to support their message generation rate at λ_T^*. Thus, either $S_i(\lambda_T^*) = S_i(\lambda_T')$ or $S_i(\lambda_T^*) = \lambda_T^*$. Thus there is no class, i, of stations such that $S_i(\lambda_T') > S_i(\lambda_T^*)$, with *all* stations flow controlled at both λ_T^* and λ_T'.

The effects of increasing the disparity in the initial allocation of transmission potentials is shown in figure 6-7. This figure plots the throughput (by class) versus the probability of message collision (which determines the average number of times a message must be retransmitted and hence, influences the message delay) for a ratio of initial allocations of 1:1, 3:1 and 9:1. The dashed lines in figure 6-7 indicate levels of constant system-wide message generation rate. As previously discussed, after a certain generation rate, a station's throughput, P(success) and P(transmit) remain constant and thus the endpoints of the throughput/P(collision) curves provide the performance results for *all* message generation rates greater than this value.

The effects of changing the ratio of the initial allocation of transmission potentials are best understood by examining system performance along a curve of constant message generation rate. Consider, for example, the dotted line for $\lambda_T = 0.4$. For a 1:1 initial allocation of transmission potentials, both classes

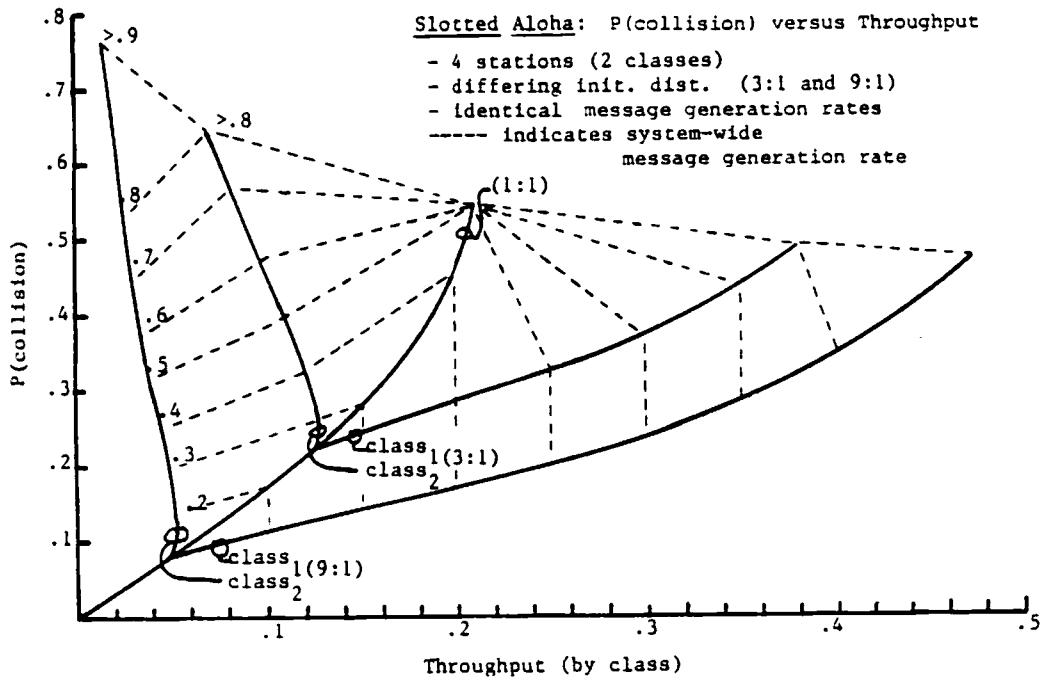


Figure 6-7: Slotted Aloha: identical message generation rates; differing initial allocation of transmission potentials

operate with a non-flow-controlled throughput of 0.2 and a 0.45 probability of collision. For a 3:1 initial allocation of transmission potentials, class 1 operates at the same non-flow-controlled throughput of 0.2 but has a lower probability of collisions (and hence better performance than in the 1:1 case). This performance increase for the class 1 stations is accompanied by a performance degradation for the class 2 stations: although the probability of collisions has decreased for the class 2 stations, their throughput has also decreased (which, according to equations 6.3, 6.4 and 6.6 is a performance degradation). When the ratio of the initial allocations is increased to 9:1, the disparity of the performance levels widens even more.

The role of the initial allocation of transmission potentials as a prioritization

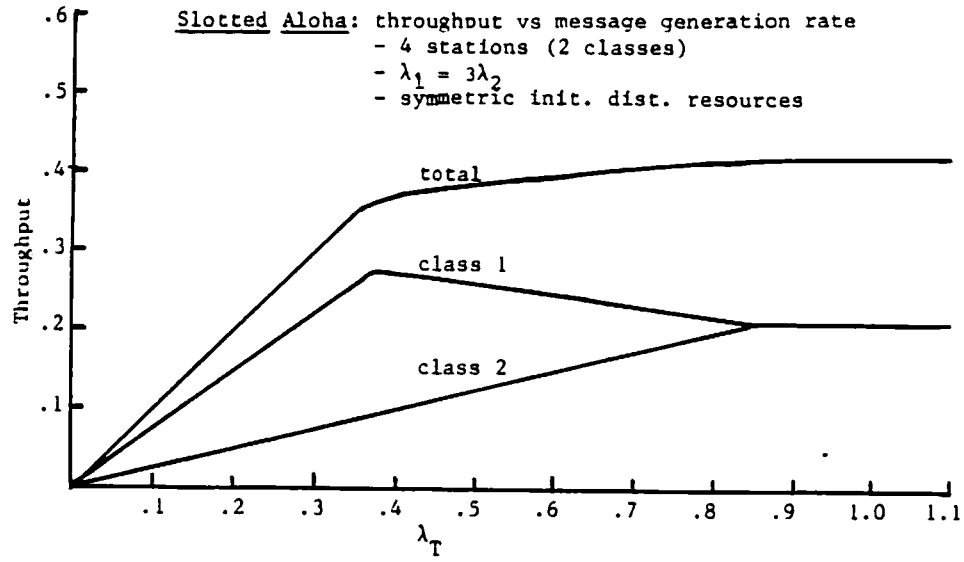
mechanism should be clear from the results of figures 6-6 and 6-7. Stations with a larger initial allocation of resources can essentially "buy" a better performance level due to the larger value of their initial "worth" (as defined in section 6.3.2.). For a constant network-wide message generation rate, an increased initial allocation of transmission potentials to the higher priority stations means these stations can begin flow controlling the lower priority stations at an even lower message generation rate and begin flow controlling themselves at a larger message generation rate. As shown in figure 6-7, the point at which the stations flow control themselves critically depends on the initial allocation of transmission potentials. This demonstrates an important aspect of the priority mechanism itself: the ability to select an initial allocation of transmission potentials from a continuum of such allocations, provides the capability of imposing any of a continuum of priority levels over the stations in the network.

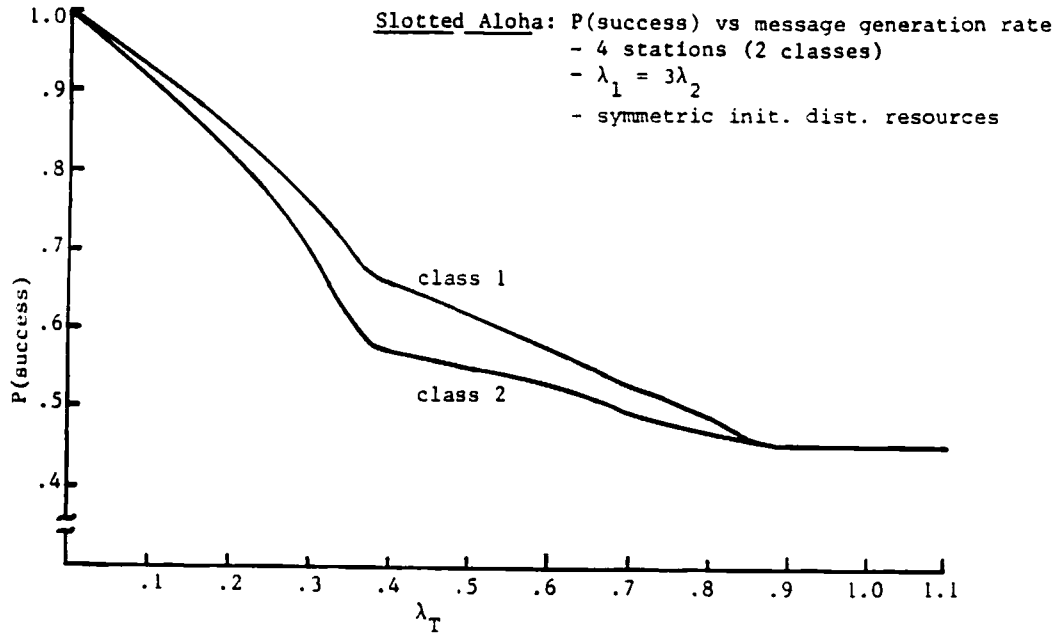
6.4.3.2 Differing Message Generation Rates; Symmetric Initial Allocation of Resources

In this section we consider the case in which the stations are partitioned into two classes (with 2 stations in each class) with asymmetric message generation rates and a symmetric initial allocation of transmission potentials. Given the role of the initial allocation of transmission potentials as a priority mechanism, one might suspect that the symmetric initial allocation in this case results in equivalent performance levels for each class of stations. Alternatively, since the performance levels themselves are dependent on λ_i (equation 6.4), one might also expect the differing message generation rates to result in different performance levels for each class of stations. Figure 6-8 indicates that, depending on the value of λ_T, either or neither of the above suspicions is correct.

In figure 6-8, the throughput (by class) and P(success) are plotted versus λ_T for a 3:1 ratio of message generation rates ($\lambda_{class\ 1} = 3\lambda_{class\ 2}$) and a symmetric initial allocation of transmission potentials. As shown in figure 6-8, the performance characteristics of the 2 classes differ in each of 3 intervals of λ_T. For $\lambda_T < 0.37$, both classes can secure enough transmission potentials to insure

Figure 6-8: Slotted Aloha: $\lambda_{\text{class 1}} = 3\lambda_{\text{class 2}}$; symmetric initial allocation of transmission potentials





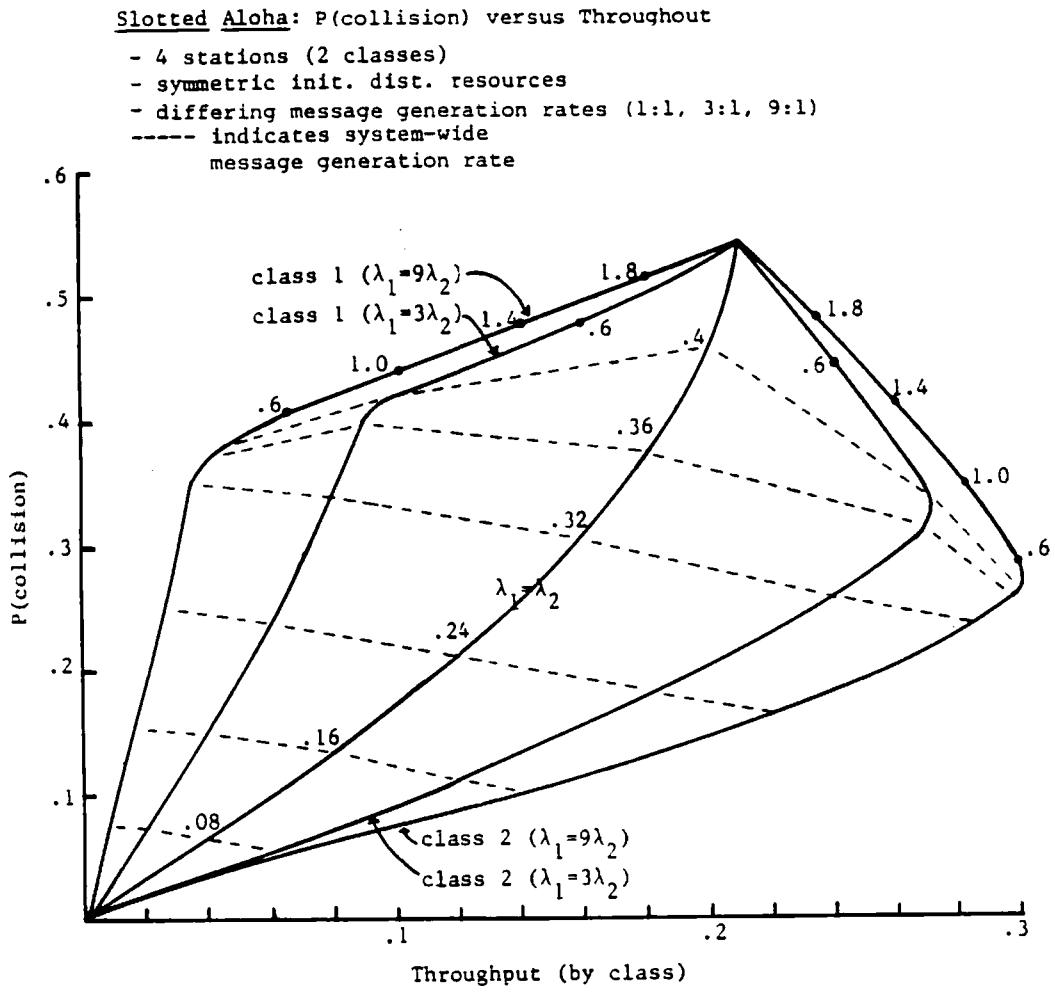
that their throughput matches their message generation rate. The differences in P(success) in this region results from the small number of stations in the network. Note that P(success) for a single class 2 station depends on the transmission probabilities of one other class 2 station and *two* class 1 stations, while P(success) for a class 1 station, depends on the transmission probabilities of only one other class 1 station and two class 2 stations. Since all messages (from all classes) are transmitted when $\lambda_T < 0.37$ and since $\lambda_{class1} > \lambda_{class2}$, a class 2 station thus sees more traffic being transmitted by the other three stations in the network than a class 1 station sees. Thus, a class 2 station has a smaller value of P(success).

In the region $0.37 < \lambda_T < 0.87$, the throughput of class 2 increases and the throughput of class 1 decreases, with an increasing λ_T. In this region, although the class 2 stations still require less of the transmission potentials than the class 1 stations (due to their smaller message generation rate), they do require enough of these resources that the class 1 stations can no longer obtain enough resources to match their throughput with their message generation rate. Thus, as $\lambda_{class\ 2}$ increases, the class 2 stations require more resources, and are allocated resources which for smaller λ_T, were allocated to the class 1 stations. Thus, in this region, the performance of the class 1 stations degrades with an increasing value of λ_T.

Of course, as $\lambda_{class\ 2}$ increases, class 2's increasing demand for transmission potentials will be satisfied only up to a certain point. Since both classes have the same initial allocation of resources, once the final allocation of resources is identical, an increasing value of λ_T will no longer affect the final distribution of resources. This occurs at $\lambda_T = 0.87$ and thus for $\lambda_T > 0.87$, both classes of stations realize the same throughput and P(success). Note that, as predicted by our claim in the previous section, once both classes begin flow controlling themselves, both their throughputs remain constant for all larger values of λ_T.

Figure 6-9 plots the throughput versus P(collision) for various ratios of λ_{class1}:λ_{class2} and a symmetric initial allocation of transmission potentials. Once again, the dashed lines indicate levels of constant system-wide message generation

Figure 6-9: Slotted Aloha: differing message generation rates; symmetric initial allocation of transmission potentials



rate; the single points indicate the value of λ_T at the indicated performance point. Note that as the disparity of message generation rates increases, the maximum throughput achievable by the class 1 stations also increases. This results from the fact that as the ratio of the message generation rates increases, the system behavior at these points of maximum throughput becomes increasingly more like that of a system consisting solely of (a smaller number of) class 1 stations. As λ_T increases, however, the increasing demands of the class 2 stations always eventually cause the class 1 stations to flow control themselves until ultimately, the throughput and P(success) of the two classes converge to identical values. Note that although these are exactly the same limiting throughput and P(success) values as in the completely symmetric case, as the ratio of the message generation rates increases, the message generation rate at which the performance values of the classes converge also increases.

6.4.4. Comments

The results in figures 6-5 through 6-9 were obtained by simulating the process of resource demand formulation and price change described by algorithm 6.1 using equations 6.9 and 6.10 to define the utility of an allocation of transmission potentials. In simulating this process, numerous initial price vectors and values for the constant $c_{i \rightarrow j}$ were chosen. It was found that, as long as $c_{i \rightarrow j}$ was chosen small enough to insure that the price changes always resulted in a new non-negative price value, algorithm 6.1 terminated with the results shown in figures 6-5 through 6-9. The algorithm was considered to have converged when the total demand for every resource was within half a percent of the total amount available. As would be expected, the value of $c_{i \rightarrow j}$ was found to greatly influence the algorithmic convergence time. In general, for a given set of initial prices, the smaller the initial value chosen for $c_{i \rightarrow j}$, the larger the convergence time of algorithm 6.1.

6.5. An Application: The Time Window Protocol

In this section, we examine the use of algorithm 6.1 in determining the optimal window sizes for the time window protocol in a heterogeneous multiple access network. Our previous work on optimizing the window size has focused on optimizing the window size with respect to a single system-wide performance objective, with the optimization itself being performed in a centralized manner. As previously discussed, there are several drawbacks to such a centralized scheme. Furthermore, due to the nature of the centralized performance objective, each station is also locked into a *de facto* priority level based on its message generation rate and the single performance objective itself provides no rational basis for introducing different priority levels among the stations in the network. In the distributed, "microeconomic" approach explored in this section, decentralized performance objectives are again formulated, the optimization is performed in a distributed manner (using algorithm 6.1) and, once again, any of a continuum of priority levels can be imposed over the stations in the network. Of course, since the single optimal window size computed by the centralized approach is also a Pareto optimal window size, the "microeconomic" approach will also be able to reproduce this result given some appropriate initial allocation of transmission potentials.

Recall that in the time window protocol, the probability that a station initially attempts to transmit a message when permitted to do so (i.e., whenever an initial window is chosen) is determined by the probability that it has a message which was generated during the selected interval of time. Thus, the lengths or sizes of the initial time windows will determine the steady state probabilities that the stations attempt message transmissions (when initially permitted to do so, i.e., whenever an initial window is chosen) and vice versa. Throughout the remainder of this section, we will focus on these steady state transmission probabilities rather than directly on the window sizes. Once the stations have computed these steady state transmission probabilities, each station can then determine the initial window size realizing its steady state transmission probability. This can be be accomplished, for example, by first estimating the initial window size and then quasi-statically adjusting the estimate until the

optimal transmission probability is realized. The details of such an adjustment process will not be considered here, however, and remain a topic for future research.

Our model of the time window protocol is thus as follows. Time is divided into mini-slots of length τ, the end-to-end propagation delay of the multiple access channel. A message's transmission time is typically many times larger than τ and thus a message will be transmitted over a period of many such mini-slots. Each station can attempt to begin a message transmission only in a mini-slot which is not part of an on-going message transmission (e.g., mini-slot 1 in figure 6-10) and also not part of the collision resolution time for previously colliding messages (e.g., mini-slot n in figure 6-10). In figure 6-10, the number of stations initially attempting to begin a message transmission in a mini-slot is given by the number of dots in the mini-slot.

The mini-slots themselves form variable length frames consisting of one or more mini-slots. Each frame begins with a mini-slot in which the stations have the opportunity to attempt a message transmission. In the case that no stations attempt a message transmission in this first mini-slot (e.g., mini-slots 1,2,3 and 1 in figure 6-10), the frame consists of this mini-slot alone. If exactly one station successfully attempts to begin a transmission in the first mini-slot (e.g., mini-slot m in figure 6-10), the frame consists of this and all subsequent mini-slots until the end of the message transmission. In figure 6-10, a successful message transmission (shown by a darkly outlined rectangle) consists of numerous mini-slots and begins with a mini-slot containing a single attempted message transmission. The remaining mini-slots in a successful transmission are shaded in this figure.

Finally, if two or more stations attempt a message transmission in the first mini-slot (e.g., mini-slots 4 and n in figure 6-10), the collision resolution time as well as time required for the eventual successful transmission of a message is included in the frame length. For example, in figure 6-10, three stations have chosen their initial time windows such that they initially attempt to begin a message transmission in mini-slot 4. In this case, the initial time windows would

be split and halves of the split windows are then selected as the new time windows, as previously discussed. In our example, suppose the three stations still have messages which were generated during their new time windows and thus all three again attempt to initiate a message transmission during mini-slot 5. At mini-slot 6, the windows have been further split and only two stations now attempt to begin a message transmission. The windowing process continues until mini-slot 8, at which point a single station initiates successful transmission of its message; this message is then successfully transmitted during the remaining mini-slots of the frame.

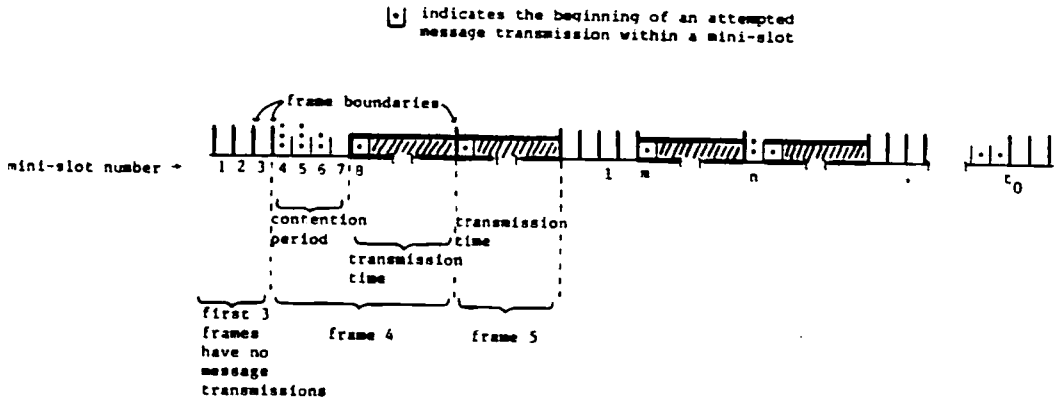


Figure 6-10: Mini-slots and frames in the time window protocol

In order to apply algorithm 6.1 to the time window protocol, we need only specify the stations' local utility functions, $thruput_i$ and $P\text{-}succ_i$, in equations 6.2 and 6.4. These functions will be presented in section 6.5.1. In the remaining subsections we present the performance results for cases in which the steady state probabilities of transmission have been computed using the "microeconomic" approach of algorithm 6.1.

6.5.1. The Utility Functions

Given the above definition of a frame, the opportunity for a station to attempt a message transmission arises only during the first mini-slot of each time frame. Thus, should station i receive the transmission potentials it demands from the other N-1 stations, each $X^{i}_{j \rightarrow i}$ in $\{X^{i}_{1 \rightarrow i}, \ldots, X^{i}_{N \rightarrow i}\}$ represents the steady state probability that station j will *not* attempt a transmission at the beginning of a frame. Once again, we will assume that each station's decision of whether or not to transmit in a frame is independent of its past attempts. Let us now derive the $P\text{-}succ_i$ and $thruput_i$ functions which determine the utility of a particular allocation of transmission potentials to station i.

The probability that station i is actually successful in transmitting a message in a frame in which it attempts a transmission can be determined by conditioning on the transmission probabilities of the other stations. Given that windows are chosen independently of the message generation process, a characteristic of the time window protocol's collision resolution process is that each station which attempts a message transmission at the beginning of a frame is equally likely to be the station which eventually successfully transmits a message in that frame. Thus we have:

$$P\text{-}succ_i = \sum_{j=0}^{N-1} \{ \text{P(j other stations attempt to transmit)} / (j+1) \} \qquad (6.14)$$

Since station i knows the probabilities $\{X^{i}_{1 \rightarrow i}, \ldots, X^{i}_{N \rightarrow i}\}$ of *non*-transmission (by the other N-1 stations) associated with a particular allocation of transmission potentials, the transmission probabilities associated with this allocation are also known and are simply $1- X^{i}_{1 \rightarrow i}$, ..., $1- X^{i}_{N \rightarrow i}$. Given these probabilities, i can easily compute the probabilities inside the sum of equation 6.14 by a straightforward combinatorial analysis.

Deriving the $thruput_i$ utility function is somewhat more complicated. We begin by deriving an expression for the system-wide throughput associated with a given allocation of transmission potentials. Consider the operation of the time window protocol over a finite number of mini-slots, as shown in figure 6-10, and suppose the initial allocation of transmission potentials is such that $n(t_0)$ messages are

successfully transmitted in the time interval $[0,t_0]$. In this case, the system-wide throughput (in units of the fraction of each mini-slot used for a successful message transmission) is given by:

$$\textit{total-thruput} = n(t_0)M \;/\; t_0 \tag{6.15}$$

where M is the transmission time (in mini-slots) of a message.

As shown in figure 6-10, the mini-slots in $[0,t_0]$ occur in alternating groups of empty frames followed by a frame containing a (possibly zero length) contention period and a message transmission. Note that the number of empty frames preceding a message transmission may itself be zero. If we define:

$\epsilon_{[0,t_0],k}$ as the number of occurrences of a run of k empty slots immediately preceding a message transmission in $[0,t_0]$.

$\sigma_{[0,t_0],k}$ as the number of contention periods of exactly k mini-slots in $[0,t_0]$.

then we have:

$$t_0 = n(t_0)M + \sum_{k=0}^{\infty} \epsilon_{[0,t_0],k}k + \sum_{k=0}^{\infty} \sigma_{[0,t_0],k}k \tag{6.16}$$

and from equation 6.15 and 6.16, we thus have:

$$\textit{total-thruput}_{[0,t_0]} = \frac{M}{M + \sum_{k=0}^{\infty} \{\epsilon_{[0,t_0],k}/n(t_0)\}k + \sum_{k=0}^{\infty} \{\sigma_{[0,t_0],k}/n(t_0)\}k} \tag{6.17}$$

Taking the limit of equation 6.17 as $t_0 \rightarrow \infty$, we get:

$$\lim_{t_0 \rightarrow \infty} \textit{total-thruput}_{[0,t_0]} = \frac{M}{M + \sum_{k=0}^{\infty} e_k k + \sum_{k=0}^{\infty} s_k k} \tag{6.18}$$

where

e_k is the probability of k empty frames immediately preceding each frame containing a message transmission.

s_k is the probability that a contention period is k mini-slots long.

Note that the first sum in the denominator of equation 6.18 is simply the

average number of empty frames associated with each frame containing a message transmission. The value of this sum is thus simply the ratio of the number of empty frames to the number of non-empty frames, or:

$$\sum_{k=0}^{\infty} e_k k = \text{P(empty frame)/P(non-empty frame)} \tag{6.19}$$

Since station i knows the value of its own transmission probability ($\min\{X_{i\rightarrow 1}^{i}, \ldots, X_{i\rightarrow N}^{i}\}$) associated with a given allocation of transmission potentials, as well as those of the other N-1 stations, it can easily compute the two probabilities in equation 6.19. The second sum in the denominator of equation 6.18 can be readily identified as the average length of a contention period. The value of this sum can also be computed in a straightforward fashion by each station using a combinatorial analysis involving the stations' transmission probabilities, as previously discussed in chapter 3.

Each station i can thus compute the system-wide throughput associated with a given allocation of transmission potentials using equation 6.18. Given equation 6.18, the $thruput_i$ function can now be easily determined. Since i itself transmits with a steady state probability of $\min_j\{X_{i\rightarrow j}^{i}\}$ or, equivalently, transmits in a fraction, $\min_j\{X_{i\rightarrow j}^{i}\}$, of the frames, it is easily seen that i's fraction of the total throughput is given by:

$$\frac{\min_j\{X_{i\rightarrow j}^{i}\}}{\min_j\{X_{i\rightarrow j}^{i}\} + \sum_{j=1}^{N} X_{j\rightarrow i}^{i}}$$

and thus

$$thruput_i = \frac{\min_j\{X_{i\rightarrow j}^{i}\}}{\min_j\{X_{i\rightarrow j}^{i}\} + \sum_{j=1}^{N} X_{j\rightarrow i}^{i}} \cdot \frac{M}{M + \sum_{k=0}^{\infty} e_k k + \sum_{k=0}^{\infty} s_k k} \tag{6.20}$$

Equations 6.14 and 6.20 thus specify the local $P\text{-}succ_i$ and $thruput_i$ functions for each of the stations. In the following subsections, we examine the use of

algorithm 6.1 in computing the optimal transmission probabilities for the time window protocol. In each of these examples, the maximization step will be performed using the formulation of equations 6.11, 6.12 and 6.13, using equations 6.14 and 6.20 to define the utility functions.

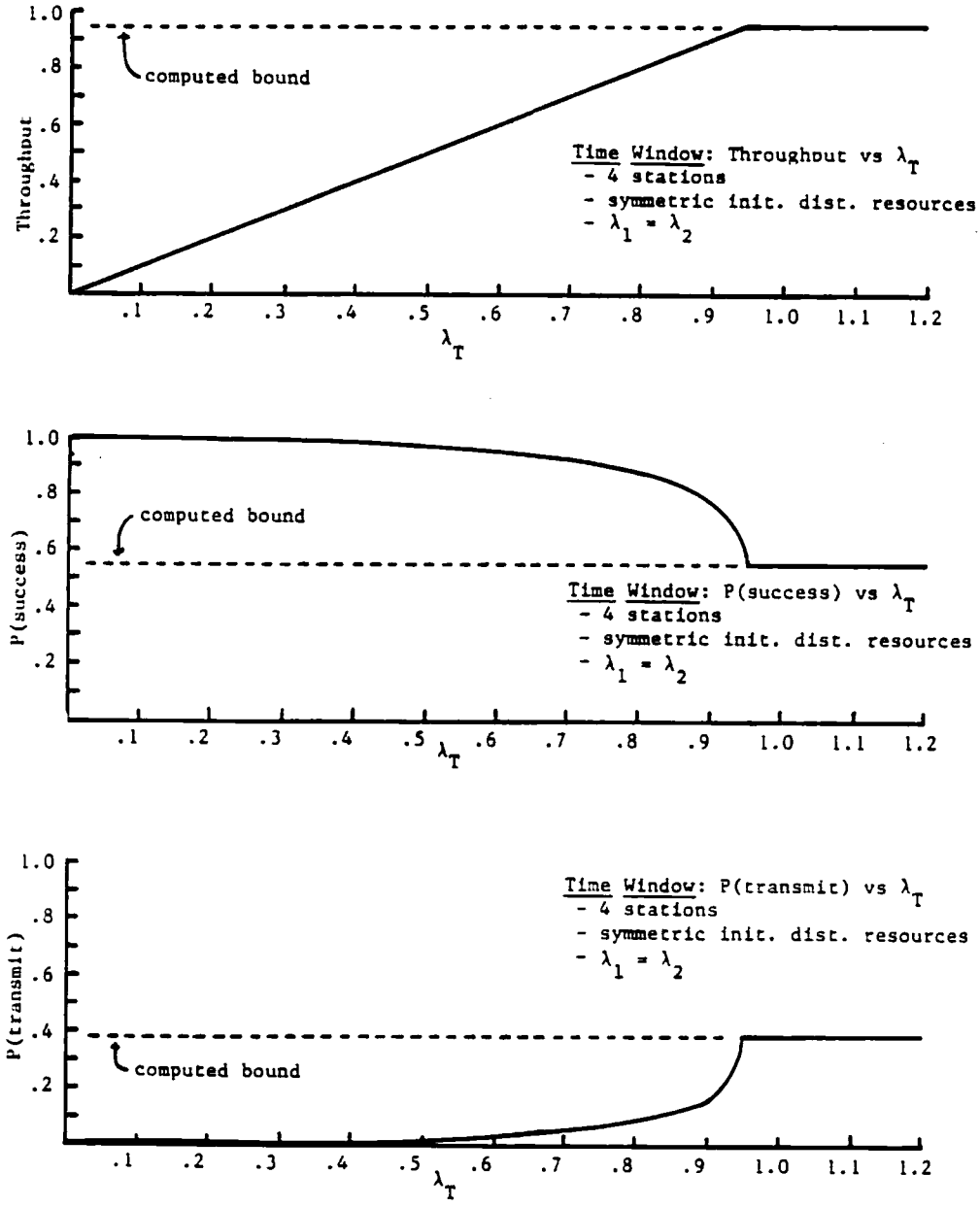
6.5.2. Perfectly Symmetric Stations

Figure 6-11 shows the results of using algorithm 6.1 to determine the Pareto optimal set of transmission probabilities in a completely symmetric four station multiple access network using the time window protocol. In this example, the rates at which stations generate messages (messages generated/message transmission time) are identical and each station begins with a symmetric allocation of 0.5 of each transmission potential which it demands. The message size, M, (in mini-slots) for this and all subsequent examples is taken to be M=10. The sum of the message generation rates at all four stations is plotted along the horizontal axis and the total system throughput, P(success) (the probability of successfully transmitting a message in a frame in which a transmission is initially attempted), and P(transmit) (the probability of message transmission) are plotted along the vertical axes.

The results of figure 6-11 indicate that the general behavior of the time window protocol is similar to that of the Slotted Aloha protocol in a completely symmetric 4 station network. Once again, the throughput matches the message generation rate up to the point at which stations begin flow controlling themselves. As λ_T increases beyond this point, the throughput again remains constant. Note, however, that the throughput at which the stations flow control themselves (and hence the maximum achievable throughput) in the time window protocol is significantly greater than in the Slotted Aloha protocol. This results primarily from the stations' ability to detect and abort colliding transmissions after only a single mini-slot.

As expected, P(success) decreases with, and P(transmit) increases with, an increasing message generation rate. Note, however, that P(transmit) remains quite small for most values of λ_T. Since each station has the opportunity to

Figure 6-11: Time Window Protocol: the completely symmetric case



transmit at the beginning of every frame, as the number of empty frames increases, so too does the opportunity to transmit. For small values of λ_T, most frames are empty and thus, not only do stations need to transmit infrequently, but the opportunity to do so arises more frequently than in the heavily loaded case. Thus, P(transmit) remains small for all but relatively large values of λ_T.

In this completely symmetric case, we can easily check the results of the distributed computation of algorithm 6.1. Let us play the role of network manager and compute the optimal transmission probabilities using a centralized optimization scheme. Since all stations have equal priority and equal message generation rates, their optimal transmission probabilities should also be identical. Thus, we can simply maximize $thruput_i$ over all possible values of this transmission probability. A straightforward numerical maximization of $thruput_i$ reveals that the unique throughput maximum occurs at P(transmit) = 0.38. This computed bound, as well as the resulting bounds for the throughput and P(success) are shown by the dashed line in figure 6-11. Note that these bounds exactly match the bounds computed via the decentralized demand formulation and price adjustment mechanism of algorithm 6.1. Once again, the ability of algorithm 6.1 to reproduce the centralized results provides an important check of the equations and numerical methods used in the computation of the stations' demands.

6.5.3. Multiple Priority Levels and Heterogeneous Stations

6.5.3.1. Identical Message Generation Rates; Differing Initial Allocations of Resources

In this section, we examine the effects of asymmetric initial allocations of transmission potentials on the stations' performance. The four stations will again be divided into two classes, with 2 stations in each class. The message generation rates for each class of stations will again be identical ($\lambda_i = \lambda_T/4$, for each station i) but the ratio of their initial allocations of transmission potentials will be 3:1.

Figure 6-12: Time Window Protocol: identical message generation rates; 3:1 initial allocation of transmission potentials

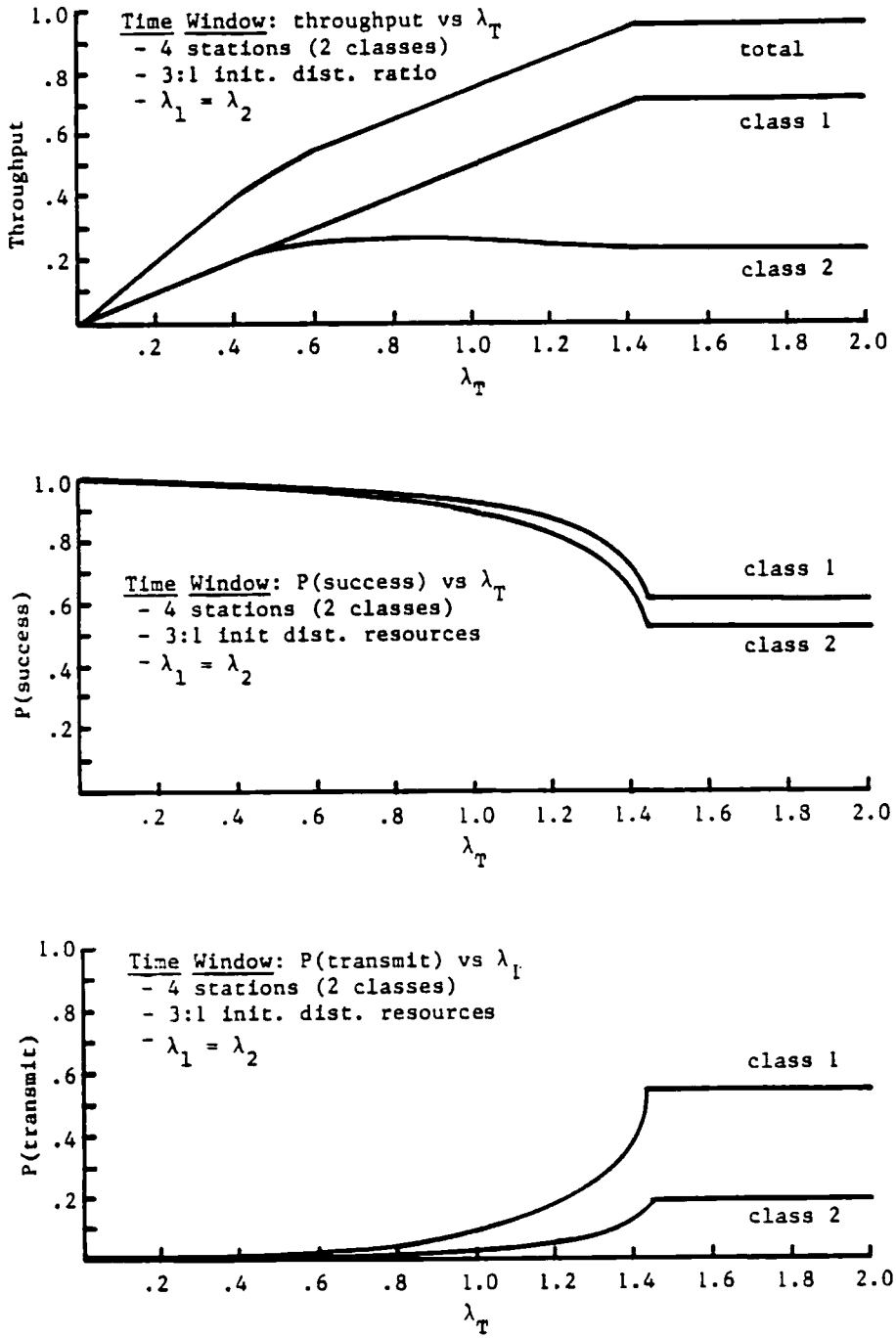


Figure 6-12 shows the performance results for a 3:1 initial allocation of transmission potentials. As in the perfectly symmetric case, the performance results exhibit the same general behavior as in the Slotted Aloha protocol. Also, as established in section 6.4.3, once both classes begin flow controlling themselves, their throughput remains constant with increasing λ_T.

Once again, since the class 1 stations possess a larger initial distribution of transmission potentials, they can essentially "buy" themselves a better level of performance. For $\lambda_T < 0.45$, there is no noticeable difference in the performance levels of the two classes; this is because at such low values of λ_T, there is little contention for the channel and each class of stations almost always transmits without interference. For values $0.45 < \lambda_T < 1.45$, however, the increase in performance for the class 1 stations (over the completely symmetric case) results in a performance degradation for the class 2 stations. Moreover, as the value of λ_T increases in this interval, the performance of the class 2 stations degrades further, although not as severely as in the Slotted Aloha example. The performance of the class 2 stations can not degrade indefinitely, however, since their initial allocation of transmission potentials will always guarantee some minimum performance level. As shown in figure 6-12, class 2 stations achieve this minimum level of performance for all values of $\lambda_T > 1.45$. Note that in any such regions in which λ_T exceeds the throughput, there must be some mechanism for losing messages at the sending stations (e.g., policy element 4 in the windowing policy) in order to prevent infinite length message queues from building up at the sending stations.

The effects of increasing the disparity in the initial allocations of transmission potentials is shown in figure 6-13, which plots the throughput (by class) versus the probability that a station in *not* successful in eventually transmitting a message in a frame in which it initially attempts a message transmission. Performance results are shown for ratios of initial allocations of 1:1, 3:1 and 9:1; once again the dashed lines indicate constant levels of λ_T. Figure 6-13 again demonstrates the role of the initial allocation of transmission potentials as a prioritization mechanism. For example, with a network-wide message generation rate of 0.8 (messages generated/message transmission time) and a completely

symmetric initial allocation, both classes realize the same throughput and P(unsuccessful). At the same value of λ_T, with a 3:1 ratio of initial allocations, the throughput of class 1 stations remains constant, but P(unsuccessful) decreases - a performance increase. The performance increase for the class 1 stations, however, is balanced by a throughput loss (performance degradation) for the class 2 stations. When the ratio of the initial allocations is increased to 9:1 the disparity in the performance levels becomes even greater.

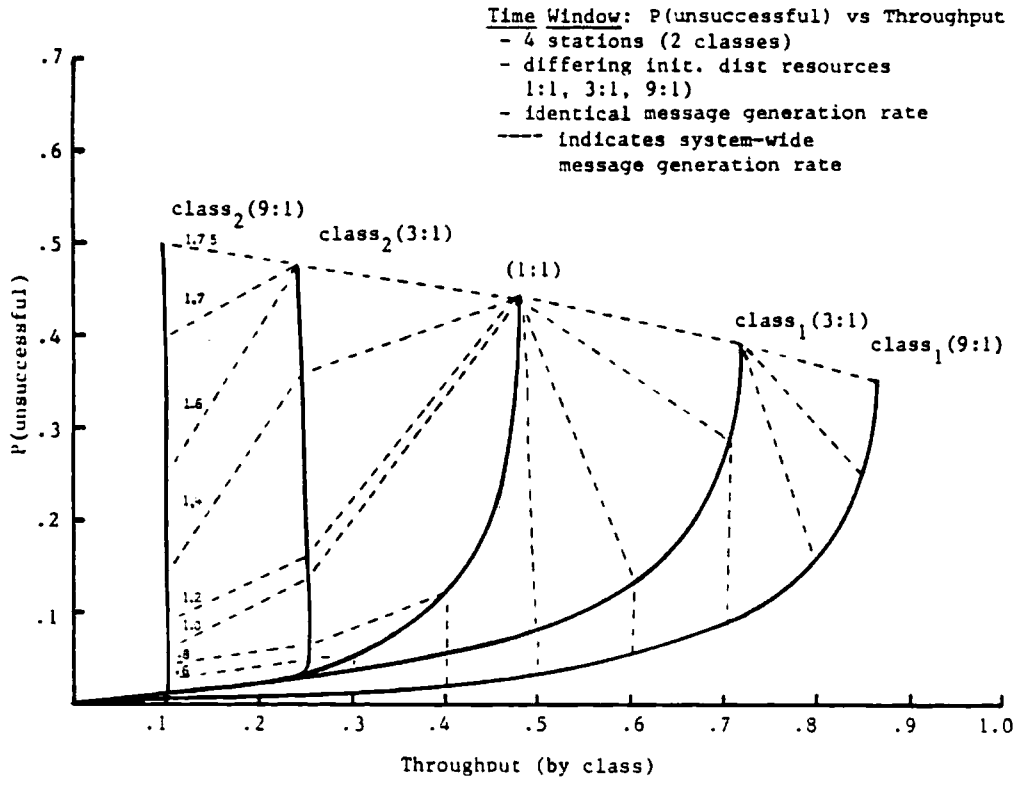


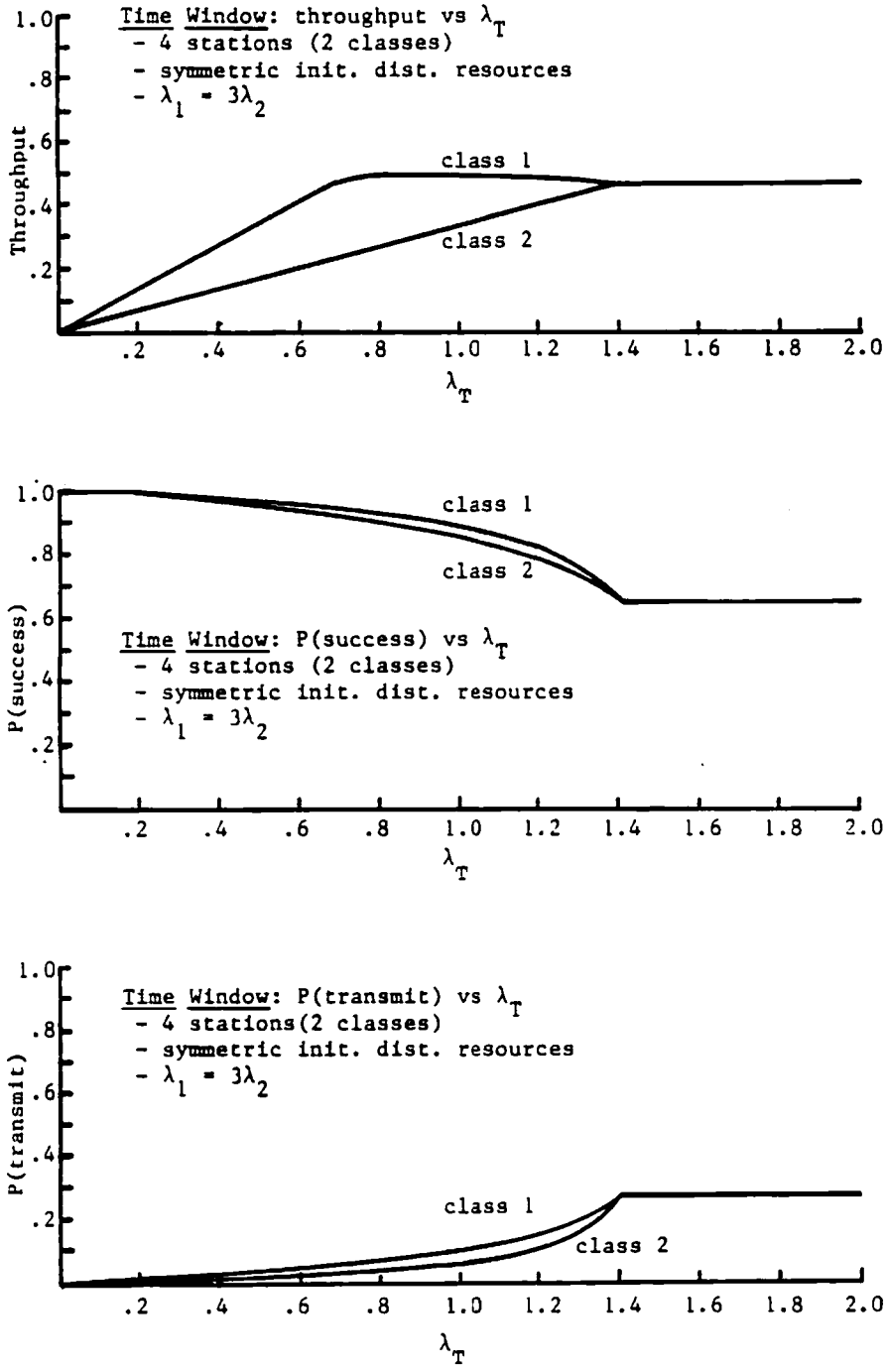
Figure 6-13: Time Window Protocol: identical message generation rates; differing initial allocation of transmission potentials

6.5.3.2. Differing Message Generation Rates;
Symmetric Initial Allocation of Resources

Figure 6-14 plots the throughput (by class), P(transmit) and P(success) versus λ_T for a 3:1 ratio of message generation rates ($\lambda_{class\ 1} = 3\lambda_{class\ 2}$) and a symmetric initial allocation of transmission potentials. As in the Slotted Aloha case, the performance characteristics of the 2 classes differ in each of 3 intervals of λ_T. In the region $\lambda_T < 0.5$, both stations can secure enough transmission potentials to match their throughput and message generation rates. Once again, as previously discussed in the Slotted Aloha case, the differences in the values of P(success) in this region result from the small number of stations in the network. In the region, $0.5 < \lambda_T < 1.4$, the increasing resource demands of class 2 stations no longer permit the class 1 stations to secure enough transmission potentials to match their throughput with their (3:1 larger) message generation rate. Thus, as λ_T increases, the throughput of the class 1 stations *decreases* due to the increasing demands of the class 2 stations. The increasing demands of the class 2 stations can, of course, be satisfied only up to a point. Since each class begins with an identical initial allocation of transmission potentials, as λ_T increases, each class should also eventually obtain an identical final allocation of transmission potentials. As shown in figure 6-14, for a 3:1 ratio of message generation rates, this occurs at $\lambda_T = 1.4$ and for $\lambda_T > 1.4$, the throughput, P(success) and P(transmit) of class 1 and class 2 stations remain identical and constant.

Figure 6-15 plots the throughput versus P(unsuccessful) for various ratios of $\lambda_{class1} : \lambda_{class2}$ and a symmetric initial allocation of transmission potentials. The dashed lines indicate levels of constant λ_T and the single points indicate the value of the system-wide message generation rate at the indicated performance point. Once again, as in Slotted Aloha, as the disparity of the message generation rates increases, the maximum throughput achievable by the class 1 stations also increases, although not nearly as dramatically as in Slotted Aloha. Also, as in Slotted Aloha, the performance results for each ratio of message generation rates converge to (nearly) identical values as the rate at which messages are generated increases. Note, however, that the value of λ_T at which these results converge itself increases as the disparity in the message generation rates increases.

Figure 6-14: Time Window Protocol: $\lambda_{class\ 1} = 3\lambda_{class\ 2}$; symmetric initial allocation of transmission potentials



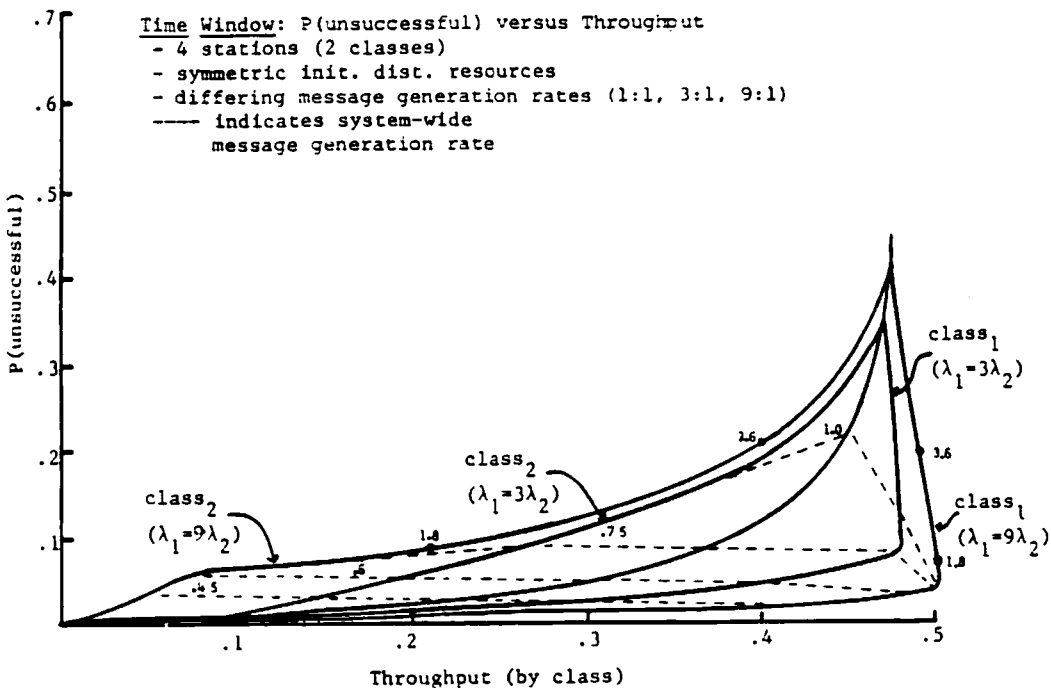


Figure 6-15: Time Window Protocol: differing message generation rates; symmetric initial allocation of transmission potentials

6.5.4. Comments

Our experiences with the convergence of algorithm 6.1 using the time window protocol utility functions differed significantly from those with Slotted Aloha utility functions. As previously discussed, for Slotted Aloha, it was found that as long as $c_{i \to j}$ was small enough to insure that the prices remained positive, algorithm 6.1 converged for all values of $c_{i \to j}$ and initial price vectors which were examined. For the time window protocol, this condition was found to be necessary but not sufficient to insure convergence. That is, while algorithm 6.1

was found to converge for most initial price vectors examined, several initial price vectors were found such that algorithm 6.1 did not converge, regardless of the values chosen for c_{i-j} . In these cases, the initial price vectors typically differed significantly from the equilibrium price vector, Furthermore, for the case of asymmetric message generation rates or initial allocation of transmission potentials, it was necessary to additionally specify, for each station, those stations which were in the same class (and thus would have the same performance level) in order to reduce the numerical complexity of the problem. We are currently uncertain whether the observed non-convergence is a result of numerical difficulties or is an inherent property of the utility functions themselves. While we suspect the former possibility is more likely (primarily due to our success with the Slotted Aloha protocol and in most cases with the time window protocol), we cannot rule out the latter possibility, since the existence of utility functions for which the algorithm 6.1 does not converge has been previously demonstrated [Scarf 60, Negishi 62] in the literature.

6.6. Summary

In this chapter we have examined a decentralized approach towards optimizing one of the operational parameters of a multiple access protocol. Our approach was to first formulate this optimization problem in terms of a fictitious resource allocation problem and then develop a distributed algorithm for computing the optimal distribution of these fictitious resources to solve this second problem. This algorithm was based on *models* previously developed in the field of microeconomics which attempt to explain how people optimally share resources among themselves in an informationally decentralized manner. In this scheme, agents act as selfish, utility-maximizing entities and their interaction through a resource pricing mechanism serves as a decentralized computational device for determining the optimal distribution of resources.

The motivation for developing such a distributed approach towards optimization in a network environment, especially in heterogeneous network environments, was presented in section 6.2 and related previous work in the fields of computer networks and microeconomics was then discussed. The fictitious network

resources, known as transmission potentials, and the notion of the multiple access network as a perfectly competitive marketplace were then developed in section 6.3. The iterative process of demand formulation and subsequent price modification was then detailed and several of its aspects, including its decentralized nature, convergence properties, and the provable optimality of its results were then considered. In sections 6.4 and 6.5, the microeconomic approach was then used to determine the optimal transmission probabilities for the Slotted Aloha and time window protocols in heterogeneous multiple access environments. The results were shown to both coincide with, and extend, results obtained using centralized optimization techniques. Flow control and priorities were also shown to emerge naturally from this approach.

In the following chapter, the conclusion to this thesis, we will discuss several additional observations about this microeconomic approach, including its possible use in solving *real* resource allocation problems in networks. We will also identify directions for possible future research within this area.

Chapter 7
Summary and Directions for Future Research

In this thesis we have addressed the problem of the design and analysis of communication protocols for supporting time-constrained communication applications in multiple access networks. The principal contributions of this thesis fall into two categories:

1. *the development and analysis of a novel class of protocols for supporting time-constrained communication applications in a multiple access environment.* There are several contributions falling under this category:

   a. identification of the critical role of an access protocol as a distributed message transmission scheduling mechanism and the importance of this role in determining the time-constrained performance of a protocol. Given the importance of this role, we have developed a class of multiple access protocols, based on the use of time windows, which can provide any of a family of message transmission scheduling disciplines based on message generation times. Novel exact and approximate performance models were developed for the cases in which the protocol provides FCFS, LCFS and Random scheduling.

   b. derivation of the optimal elements of the windowing policy of the time window protocol using a semi-Markov decision model. The performance model we developed to examine the time-constrained behavior of the optimal windowing policy is based on a queueing system with impatient customers. This work augments existing analytic modeling techniques by providing a simple, analytically tractable model for determining customer loss in M/G/1 queues in which customers are denied service when their waiting time exceeds a given time bound.

   c. extension of the time window protocol to the multi-class case in which network stations must support the transmission of both time-constrained and non-time-constrained classes of traffic.

2. *development of a systematic and formal approach towards distributed optimization via a fictitious resource sharing paradigm and a decentralized microeconomic approach towards the solution of resource sharing problems.* This work draws on models and methods from microeconomic theory to provide blueprints for a more systematic approach towards engineering decentralized optimization algorithms and distributed resource sharing mechanisms in distributed systems. Our "microeconomic" approach was successfully applied to the problem of computing the optimum transmission probabilities for both the time window protocol and the Slotted Aloha protocol. Interestingly, several network mechanisms, such as flow control and priorities were found to emerge naturally from this approach.

We believe that many of the above contributions transcend the particular problem domain of time-constrained communication in multiple access networks and have applications to problems occurring in other areas of distributed computation and communication. For example, many of our performance models, such as our work on queues with impatient customers and our semi-Markov decision model of protocol operation, have applications to other network problems such as calculation of buffer overflow probabilities. More generally, these models will be applicable to most other problems in which network entities have a time-critical need to access a shared network resource. In our work on the microeconomic approach towards decentralized optimization of the transmission probabilities, we cast the optimization problem in terms of a fictitious resource allocation problem and used the decentralized, microeconomic approach to solve this problem. We believe such an approach is likely to be applicable to other optimization problems as well. Also, the application of these ideas to the optimal allocation of *real* resources in a distributed network environment seems an obvious, yet currently unexplored, extension of this work.

There are many additional possible directions in which the research presented in this thesis can be extended. Several extensions of our work on the time window protocol seem promising to pursue. First, note that our *definition* of a policy (i.e., elements (1) through (6) in section 5.3) is only one policy by which the protocol can be controlled. Introducing additional policy elements (e.g., not necessarily splitting a window in half), may result in further performance improvements. Secondly, the protocols presented in this thesis all operate in a

synchronous manner. Since the synchronization of distributed stations is often a difficult task, it would be desirable for the protocol to operate in an asynchronous manner. It would be interesting to explore different approaches for achieving this asynchronous operation and the effect of these approaches on the time-constrained performance of the protocol. Molle [Molle 83] has recently investigated several aspects of this problem. Finally, in chapter 5 we presented a scheme for generalizing the time window mechanism for the case in which both time-constrained and non-time-constrained messages must be transmitted by the network stations. Our aim here was simply to demonstrate that the windowing mechanism can be naturally extended to such an environment and to examine the performance tradeoffs available when both classes of traffic are supported. The extensions to the time window protocol presented in chapter 5 are but one way in which the protocol can be modified to support multiple classes of message traffic; alternate windowing schemes can be easily imagined and a comparative performance study of such schemes remains another problem for future research.

We believe that many of the most important and promising directions for related future research lie in the areas of decentralized resource sharing and distributed optimization. The most important, and probably the most difficult, immediate problem involves the convergence properties of the resource pricing mechanism. Few results are currently available in this area [Arrow and Hahn 71] and significant additional theoretical results are likely to be required before such a mechanism can be considered for implementation in actual systems. The application of the selfish, microeconomic approach to other network problems, such as flow control in virtual circuits [Yemini 81] and *real* resource sharing problems, such as distributed loadsharing and distributed directory placement, seems quite promising and should also be pursued. Finally, several alternate pricing mechanisms have been proposed, though not as thoroughly explored, in the economics literature. The investigation of these approaches, especially those which would not require resource allocation to be performed prior to system startup, is also an important area for future research.

Finally, we believe that the notion of a network of distributed agents as an artificial society of interacting entities is a particularly rich and powerful

metaphor and evocative of the many similarities which exist between naturally occurring and man-made (engineered) distributed systems. Recently, we have seen ideas from seemingly distant fields such as organizational management, team decision theory and the organization of natural systems [Hluchyj and Gallager 81, Fox 81, Lesser and Corkill 81, Lesser and Corkill 83] appearing in the area of distributed problem solving. We believe this cross-fertilization of ideas may lead to particularly fruitful research directions in the future. For just as humans, as individuals, process information efficiently and researchers in artificial intelligence look to the human mind for efficient methods of storing, retrieving and processing information, we believe that humans, as societal agents, also organize and process information efficiently as a group and that designers of distributed systems can thus look to models of human economic and social organization for insight into efficient methods in distributed problem solving.

Bibliography

[Abramson 70] N. Abramson.
The ALOHA System - Another Alternative for Computer Communications.
In *Proc. AFIPS Fall Joint Computer Conference*, (Houston, Tx., Nov. 17-19), AFIPS Press, Montvale, N.J., pp. 281-285.

[Abramson 73] N. Abramson.
Packet Switching with Satellites.
In *Proc. AFIPS National Computer Conference*, (N.Y., N.Y., June 4-8), AFIPS Press, Montvale, N.J., pp. 695-703.

[Abramson 77] N. Abramson.
The Throughput of Packet Broadcasting Channels.
IEEE Transactions on Communications **COM-25**, 1 (January), pp. 117-128.

[Arrow and Debreu 54]
K. Arrow, G. Debreu.
Existence of Equilibrium for a Competitive Economy.
Econometrica **22**, 3 (July), pp. 265-290.

[Arrow and Hahn 71]
K. Arrow and F. Hahn.
General Competetive Analysis.
Holden Day Publishers, San Francisco, Ca., 1971.

[Arrow et al. 59]
K. Arrow, H. Block, L. Hurwicz.
On the Stability of the Competetive Equilibrium II.
Econometrica **27**, 1 (January), pp. 82-109.

[Arthurs and Stuck 79]
E. Arthurs and B.W. Stuck.
A Theoretical Traffic Performance Analysis of an Integrated Voice-Data Virtual Circuit Packet Switch.
IEEE Transactions on Communications **COM-27**, 7 (July), pp. 1104-1111.

[Baccelli and Hebuterne 81]
Francois Baccelli and Gerard Hebuterne.
On Queues With Impatient Customers.
In *Performance '81*, ( ), North-Holland Publishing Company, Amsterdam, pp. 159-179.

[Bially et al. 80a]
T. Bially, A.J. McLaughlin and C.J. Weinstein.
Voice Communications in Integrated Digital Voice and Data Networks.
IEEE Transactions on Communications **COM-28**, 9 (September), pp. 1478-1490.

[Bially et al. 80b]
T. Bially, B. Gold and S. Seneff.
A Technique for Adaptive Voice Control in Integrated Packet Networks.
IEEE Transactions on Communications **COM-28**, 3 (March), pp. 325-333.

[Brooks 83] R. Brooks.
Experiments in Distributed Problem Solving with Iterative Refinement.
PhD thesis, University of Massachusetts, Department of Computer and Information Sciences, February, 1983.

[Bullington and Fraser 59]
K. Bullington and J. Fraser.
Engineering Aspects of TASI.
Bell System Technical Journal **38**, 2 (March), pp. 353-364.

[Bux et al. 81] W. Bux, F. Closs, P.A. Janson, K. Kummerle and H.R. Muller.
A Reliable Token Ring System for Local Area Communication.
In *Proc. National Telecommunications Conference*, (New Orleans, La., Nov. 29 - Dec. 3), IEEE, Piscataway, N.J., pp. A2.2.1 to A2.2.6.

[Capetanakis 79] J. I. Capetanakis.
Generalized TDMA: The Multi-Accessing Tree Protocol.
IEEE Transactions on Communications **COM-27**, 10 (October), pp. 1476-1484.

[Carleial and Hellman 75]
A. Carleial and M. Hellman.
Bistable Behavior of ALOHA-Type Systems.
IEEE Transactions on Communications **COM-23**, 4 (April), pp. 401-409.

[Chang 77] L. Chang.
Analysis of Integrated Voice and Data Communication Network.
PhD thesis, Carnegie-Mellon University, Department of Computer Science, November, 1977.

[Chlamtac et al. 79]
I. Chlamtac, W.R. Franta and K.D. Levin.
BRAM: The Broadcast Recognizing Access Method.
IEEE Transactions on Communications **COM-27**, 8 (August), pp. 1183-1190.

[Clark et al. 78] D. Clark, K. Progran and D. Reed.
An Introduction to Local Area Networks.
Proc. IEEE **66**, 11 (November), pp. 1497-1516.

[Coffman 76] E.G. Coffman.
Computer and Job Shop Scheduling.
J. Wiley and Sons, N.Y. N.Y., 1976.

[Cohen 77] D. Cohen.
Issues in Transnet Packetized Voice Communication.
In *Proc. Fifth Data Communications Symposium*, (Snowbird, Utah, Sept. 27-29), ACM, N.Y., N.Y., pp. 6-60 to 6-13.

[Coviello 79] G.J. Coviello.
Comparative Discussion of Circuit vs. Packet Switched Voice.
IEEE Transactions on Communications **COM-27**, 8 (August), pp. 1153-1160.

[Cruz and Hajek 82]
R. Cruz and B. Hajek.
A New Upper Bound to the Throughput of a Multi-Access Broadcast Channel.
IEEE Transactions on Information Theory **IT-28**, 3 (May), pp. 402-405.

[Debreu 59] G. Debreu.
Theory of Value.
Wiley, N.Y., N.Y., 1959.

(DSN 82) MIT Lincoln Laboratories, Lexington, Mass..
Workshop on Distributed Sensor Networks, 1982.
(Lexington, Mass., Jan.).

[Eslanadidi and Chu 82]
M. Eslanadidi and W. Chu.
An Analysis of a Time Window Multiaccess Protocol with Collision Size Feedback.
In *Proc. ACM Computer Networks Symp.*, (College Park, Md., April 13-14), ACM, N.Y., N.Y., pp. 112-117.

[Farber at al. 73]
D.J. Farber, J. Feldman, F. Heinrich, M. Hopwood, K. Larson, D. Loomis and L. Rowe.
The Distributed Computing System.
In *Proc. 7th Annual IEEE Comp. Soc. Int. Conference (COMPCON)*, (San Francisco, Ca., Feb. 27 - Mar. 1), IEEE, Piscataway, N.J., pp. 31-34.

[Farmer and Newhall 69]
W.D. Farmer and E.E. Newhall.
An Experimental Distributed Switching System to Handle Bursty Computer Traffic.
In *Proc. ACM Symposium on Problems in the Optimization of Data Communications Systems*, (Pine Mountain Ga., Oct. 13-16), ACM, N.Y., N.Y., pp. 1-33.

[Fayolle et al. 77]
G. Fayolle, E. Gelenbe and J. Labetoulle.
Stability and Optimal Control of the Packet Switching Broadcast Channel.
Journal of the Association for Computing Machinery **24**, 3 (July), pp. 375-386.

[Fischer and Harris 76]
M.J. Fischer and T.C. Harris.
A Model for Evaluating the Performance of an Integrated Circuit and Packet Switched Multiplex Structure.
IEEE Transactions on Communications **COM-24**, 2 (February), pp. 95-202.

[Forgie 75] J.W. Forgie.
Speech Transmission in Packet Switched Store and Forward Networks.
In *Proc. AFIPS National Computer Conference*, (Anaheim, Ca., May 19-22), AFIPS Press, Montvale, N.J., pp. 137-142.

[Forgie and Nemeth 77]
J. Forgie and A. Nemeth.
An Efficient Packetized Voice/Data Network Using Statistical Flow Control.
In *Proc. International Communications Conference*, (Chicago, Ill., June 12-15), IEEE, Piscataway, N.J., pp. 38.2-44 to 38.2-48.

[Fox 81] M. Fox.
An Organizational View of Distributed Systems.
IEEE Transactions on Systems, Man, and Cybernetics **SMC-11**, 1 (January), pp. 70-80.

[Fratta et al. 81]
L. Fratta, F. Borgonovo and F. Tobagi.
The EXPRESS-Net: A Local Area Communication Network Integrating Voice and Data.
In G. Pujolle (editor), *Performance of Data Communication Systems*, pages 77-88. North-Holland, 1981.

[Gallager 76] R.G. Gallager.
Basic Limits on Protocol Information in Data Communication Networks.
IEEE Transactions on Information Theory **IT-22**, 4 (July), pp. 385-398.

[Gallager 78] R. G. Gallager.
Conflict Resolution in Random Access Broadcast Networks.
In *Proc. AFOSR Workshop in Communication Theory and Applications*, (Provincetown, Mass., Sept. 17-20), IEEE, Piscataway, N.J., pp. 74-76.

[Georgiadis and Papantoni-Kazakos 81]
L. Georgiadis and P. Papantoni-Kazakos.
A Collision Resolution Process Using Energy Detectors.
In *Proc. National Telecommunications Conference*, (New Orleans, La., Nov. 29 - Dec. 3), IEEE, Piscataway, N.J., pp. E3.6.1 - E3.6.5.

[Gitman et al. 77]
I. Gitman, H. Frank, B. Occhiogrosso and W. Hsieh.
Issues in Integrated Network Design.
In *Proc. International Communications Conference*, (Chicago, Ill., June 12-15), IEEE, Piscataway, N.J., pp. 38.1-36 to 38.1-43.

[Gonsalves 83] T. Gonsalves.
Packet-Voice Communication on an Ethernet Local Computer Network: an Experimental Study.
In *Proc. SIGCOMM Communications Architectures and Protocols Symposium*, (Austin, Tx., March 8-9), ACM, N.Y., N.Y., pp. 178-185.

[Gopal et al. 81]
P.M. Gopal, J.W. Wong and J.C. Majithia.
An Evaluation of Playout Strategies for Voice Transmission in Packet Networks.
In *Proc. Computer Networking Symposium*, (Gaithersburg, Md., Dec. 8), IEEE, Piscataway, N.J., pp. 33-38.

[Grami et al. 82]
A. Grami, K. Sohraby, and J. Hayes.
Further Results on Probing.
In *Proc. International Communications Conference*, (Philadelphia, Pa., June 13-17), IEEE, Piscataway, N.J., pp. 1C.3.1 - 1C.3.3.

[Gruber 81] J.G. Gruber.
Delay Related Issues in Integrated Voice and Data Networks.
IEEE Transactions on Communications **COM-29**, 6 (June), pp. 786-800.

[Hayes 78] J. F. Hayes.
An Adaptive Technique for Local Distribution.
IEEE Transactions on Communications **COM-26**, 8 (August), pp. 1178-1186.

[Hildenbrand and Kirman 76]
W. Hildenbrand and A. Kirman.
Advanced Textbooks in Economics: Introduction to Equilibrium Analysis.
North-Holland Publishing Company, Amsterdam, 1976.

[Hluchyj and Gallager 81]
M. Hluchyj and R. Gallager.
Multiaccess of a Slotted Channel by Finitely Many Users.
In *Proc. National Telecommunications Conference*, (New Orleans, La., Nov. 29 - Dec. 3), IEEE, Piscataway, N.J., pp. D4.2.1-D4.2.6.

[Howard 71] Ronald A. Howard.
Dynamic Probabilistic Systems, Volume 2: Semi-Markov and Decision Processes.
J. Wiley and Sons, N.Y. N.Y., 1971.

[Kahn et al. 78] R. Kahn, S.A. Gronemeyer, J. Burchfiel and R.C. Kunzelman.
Advances in Packet Radio Technology.
Proc. IEEE **66**, 11 (November), pp. 1468-1496.

[Karlin 59] S. Karlin.
Addison-Wesley Series in Statistics: Mathematical Methods and Theory in Games, Prorgamming and Economics.
Addison-Wesley, Reading, Mass., 1959.

[Kim 83] B. Kim.
Two Adaptive Token Ring Strategies for Real-Time Traffic.
In *Proc. Computer Networking Symposium*, (Silver Spring, Md., Dec. 13), IEEE, Piscataway, N.J., pp. 119-121.

[Kleinrock 75] L. Kleinrock.
Queueing Systems Volume I: Theory.
J. Wiley and Sons, N.Y., N.Y., 1975.

[Kleinrock 76] L. Kleinrock.
Queueing Systems Volume II: Computer Applications.
J. Wiley and Sons, N.Y., N.Y., 1976.

[Kleinrock 77] L. Kleinrock.
Performance of Distributed Multi-Access Computer Communication Systems.
In *1977 IFIP Congress Proceedings*, (Toronto, Canada, Aug. 8-12), North Holland Publishing, Amsterdam, pp. 547-552.

[Kleinrock and Lam 75]
L. Kleinrock and S.S. Lam.
Packet Switching in a Multiaccess Broadcast Channel: Performance Evaluation.
IEEE Transactions on Communications **COM-23**, 4 (April), pp. 410-423.

[Kleinrock and Scholl 80]
L. Kleinrock and M.O. Scholl.
Packet Switching in Radio Channels: New Conflict-Free Multiple Access Schemes.
IEEE Transactions on Communications **COM-28**, 7 (July), pp. 1015-1029.

[Kleinrock and Tobagi 75]
L. Kleinrock and F. A. Tobagi.
Packet Switching in Radio Channels: Part I - Carrier Sense Multiple Access Modes and Their Throughput-Delay Characteristics.
IEEE Transactions on Communications **COM-23**, 12 (December), pp. 1400-1416.

[Kleinrock and Yemini 78]
L. Kleinrock and Y. Yemini.
An Optimal Adaptive Scheme for Multiple Access Broadcast Communication.
In *Proc. International Communications Conference*, (Toronto, Canada, June 4-7), IEEE, Piscataway, N.J., pp. 7.2.1-7.2.5.

[Kobayashi 78] H. Kobayashi.
Modeling and Analysis.
Addison Wesley, Reading, Massachusetts, 1978.

[Kurose et al. 85]
J. F. Kurose, M. Schwartz and Y. Yemini.
Distributed Multiple Access Protocols and Real-Time Communication.
to appear in Computing Surveys , 1985.

[Lam 80] S. Lam.
A Carrier Sense Multiple Access Protocol for Local Networks.
Computer Networks **4**, 1 (January), pp. 21-32.

[Lam and Kleinrock 75]
S.S. Lam and L. Kleinrock.
Packet Switching in a Multiaccess Broadcast Channel: Dynamic Control Procedures.
IEEE Transactions on Communications **COM-23**, 9 (September), pp. 891-904.

[Lesser and Corkill 81]
V. Lesser and D. Corkill.
Functionally Accurate, Cooperative Distributed Systems.
IEEE Transactions on Systems, Man, and Cybernetics **SMC-11**, 1 (January), pp. 81-96.

[Lesser and Corkill 83]
V. Lesser and D. Corkill.
The Distributed Vehicle Monitoring Testbed.
AI Magazine **4**, 3 (Fall), pp. 15-33.

[Limb and Flores 82]
J. Limb and C. Flores.
Description of FASNET - A Unidirectional Local-area Communications Network.
Bell System Technical Journal **61**, 7 (September), pp. 1413-1440.

[Maglaris 79] B. Maglaris.
Studies in Integrated Line and Packet-switched Computer Communication Systems.
PhD thesis, Columbia University, Department of Electrical Engineering, May, 1979.

[Maglaris and Lissack 81]
B. Maglaris and T. Lissack.
A Priority TDMA Protocol for Satellite Data Communications.
In *Proc. International Communications Conference*, (Denver, Co., June 14-18), IEEE, Piscataway, N.J., pp. 73.3.1 to 73.3.5.

[Maxemchuk 82] N. Maxemchuk.
A Variation on CSMA/CD That Yields Movable TDM Slots in Integrated Voice/Data Local Networks.
Bell System Technical Journal **61**, 7 (September), pp. 1527-1550.

[McQuillan and Walden 77]
J.M. McQuillan and D.C. Walden.
The ARPA Network Design Decisions.
Computer Networks **1**, (August), pp. 243-289.

[Metcalfe and Boggs 76]
R.M. Metcalfe and D.R. Boggs.
Ethernet: Distributed Packet Switching for Local Computer Networks.
Communications of the ACM **19**, 7 (July), pp. 395-403.

[Mittal and Venetsanopoulos 81]
K.K. Mittal and A. N. Venetsanopoulos.
On the Dynamic Control of the Urn Scheme for Multiple Access Broadcast Communication Systems.
IEEE Transactions on Communications **COM-29**, 7 (July), pp. 962-970.

[Molle 81] M. L. Molle.
Extensions and Unifications of the Multiple Access Communication Problem.
Technical Report CSD-810730, UCLA Computer Science Dept., July, 1981.

[Molle 82] M. Molle.
On the Capacity of Infinite Population Multiple Access Protocols.
IEEE Transactions on Information Theory **IT-28**, 3 (May), pp. 396-401.

[Molle 83] M. Molle.
Asynchronous Multiple Access Tree Algorithms.
In *Proc. SIGCOMM Communications Architectures and Protocols Symposium*, (Austin, Tx., March 8-9), ACM, N.Y., N.Y., pp. 214-218.

[Mowafi and Kelly 80]
O.A. Mowafi and W.J. Kelly.
Integrated Voice/Data Packet Switching Techniques for Future Military Networks.
IEEE Transactions on Communications COM-28, 9 (September), pp. 1655-1662.

[Negishi 62] T. Negishi.
The Stability of a Competitive Economy: A Survey Article.
Econometrica **30**, 4 (October), pp. 635-669.

[Nutt and Bayer 82]
G.J. Nutt and D.L. Bayer.
Performance of CSMA/CD Networks Under Combined Voice and Data Loads.
IEEE Transactions on Communications **COM-30**, 1 (January), pp. 6-11.

[Pareto 27] V. Pareto.
Manuel d'Economie Politique.
Paris, 1927.

[Pippenger 81] N. Pippenger.
Bounds on the Performance of Protocols for a Multiple Access Broadcast Channel.
IEEE Transactions on Information Theory **IT-27**, 2 (March), pp. 145-151.

[Pokress 84] R. Pokress, editor.
Special Issue on Integrated Services Digital Networks.
IEEE Communications Magazine **22**, 1 (January).

[Saltzer and Clark 81]
J. Saltzer and D Clark.
Why a Ring?
In *Proc. 7th Data Communication Sym.*, (Mexico City, Mexico, Oct. 27-29), IEEE, Piscataway, N.J., pp. 211-223.

[Scarf 60] H. Scarf.
Some Examples of Global Instability of the Competetive Equilibrium.
International Economic Review **1**, 3 (September), pp. 157-172.

[Schwartz 77] M. Schwartz.
Computer Communication Network Design and Analysis.
Prentice Hall, 1977.

[Schwartz and Kraimeche 83]
M. Schwartz and B. Kraimeche.
An Analytic Control Model for an Integrated Node.
In *Proc. IEEE INFOCOM 1983*, (San Diego, Ca., Apr. 18-21), IEEE, Piscataway, N.J., pp. 540-546.

[Shoch and Hupp 80]
J.F. Shoch and J. Hupp.
Measured Performance of an Ethernet Local Network.
Communications of the ACM **23**, 12 (December), pp. 711-721.

[Tanenbaum 81] A. S. Tanenbaum.
Computer Networks.
Prentice Hall, Englewood Cliffs, N.J., 1981.

[Tobagi 80] F. Tobagi.
Multiaccess Protocols in Packet Communication Systems.
IEEE Transactions on Communications **COM-28**, 4 (April), pp. 468-488.

[Tobagi and Kleinrock 77]
F. Tobagi and L. Kleinrock.
Packet Switching in Radio Channels: Part IV - Stability Considerations and Dynamic Control in Carrier Sense Multiple Access.
IEEE Transactions on Communications **COM-25**, 10 (October), pp. 1103-1119.

[Towsley and Venkatesh 82]
D. Towsley and G. Venkatesh.
Window Random Access Protocols for Local Computer Networks.
IEEE Transactions on Computers **C-31**, 8 (August), pp. 715-722.

[Weinstein et al. 80]
C.J. Weinstein, M.L. Malpass and M.J. Fisher.
Data Traffic Performance of an Integrated Circuit- and Packet-Switched Multiplex Structure.
IEEE Transactions on Communications **COM-28**, 6 (June), pp. 873-878.

[Yemini 81] Y. Yemini.
Selfish Optimization in Computer Networks.
In *Proc. 20th IEEE Conference on Decision and Control*, (San Diego, December), IEEE, Piscataway, N.J., pp. 281-285.

[Yemini and Kleinrock 79]
Y. Yemini and L. Kleinrock.
On a General Rule for Access Control or, Silence is Golden...,
In *Proc. International Symposium on Flow Control in Computer Networks*, (Versailles, Feb. 12-15), North Holland Press, Amsterdam, pp. 335-347.