UCLA-ENG-8118
July 1981

# Unifications and Extensions of the Multiple Access Communications Problem

by
Mart Lauri Molle

This research, conducted under the chairmanship of Professor Leonard Kleinrock, was sponsored by the Defense Advanced Research Projects Agency, Department of Defense.

**Computer Science Department**
**School of Engineering and Applied Science**
**University of California**
**Los Angeles**

## ABSTRACT

Multiple access protocols permit a broadcast communications channel to be shared by a large number of stations under distributed control. It is assumed that only one message at a time can be transmitted successfully over the common channel. We derive a local optimality condition for synchronous multiple access protocols, and show that many known protocols are special cases of this condition. We include a survey of much of the recent work on infinite population tree algorithms that use the history of channel activity to carry out short-range dynamic scheduling. A novel approach is presented for deriving upper bounds on the maximum stable throughput with finite average delay for infinite population protocols. Bounds are found for the case of arbitrarily complex algorithms, and for the restricted (but reasonable) class of protocols that obey a "degenerate intersection" property. This latter class is quite interesting, being a slight generalization of first-come first-served that includes all currently proposed protocols. We extend the model to include multiple access protocols aided by a partial reservation channel.

Particular emphasis is placed on multiple access in the context of local networks. We derive a new carrier sense multiple access protocol, *virtual time CSMA*, and prove it to be the *best possible* CSMA protocol under some common assumptions. In virtual time CSMA, messages are assigned transmission times during the idle periods on the channel based on their arrival times through the use of variable speed clocks.

In local networks, the cost of each incorrect scheduling decision is reduced, but it may be the case that the ratio of the cost of scheduling no transmissions to the cost of scheduling two or more transmissions is far from unity. The implications of this effect have not been widely appreciated. We show that these differences in the characteristics of the channel can be large enough to *invalidate* a straightforward extension of the previously described upper bounds on capacity to the case of local networks. In particular, we introduce a new class of hybrid carrier sense-binary search protocols and show that they can achieve surprisingly high stable throughputs when the idle-detect time is much less than the collision-detect time.

## ACKNOWLEDGEMENTS

This research was supported by the Advanced Research Projects Agency of the Department of Defense under Contract MDA 903-77-C-0272, and by the Regents of the University of California through a Chancellor's Intern Fellowship.

I also had the faithful support and guidance of many others. My ever patient wife, Mary, was always willing to hear about some new idea or other, and our parents were always full of encouragement. My chairman, Leonard Kleinrock, and the members of my committee kept me honest. Special thanks also go to my friend and former committee member, James L. Massey, and to Dr. Boris Tsybakov for their help.

I made many friends in the Computer Science Department at UCLA. Michael Molloy, Richard Gail, Randolph Nelson and the rest of the students in Modelling & Analysis were always ready to talk shop; Bruce Walker and Paul Eggert never let me forget the rest of Computer Science. Professor Robert Uzgalis helped teach me teaching. Brenda Ramsey was always resourceful and Cheryle Childress always had a kind word. The staff of our research group, Amanda Daniels, George-Ann Hornor, Leon Lemons, Terry Peters and especially Linda Infeld helped me with numerous letters and reports.

Table of Contents

List of Figures

page

# CHAPTER 1
# Introduction

## 1.1: Some Trends in the Evolution of Computing Systems

Early computers were expensive, slow and unreliable. Every effort was made to transform each problem into a form that was easy for the *computer* to solve, because the cost of the computer itself dominated the cost of computer aided problem solving. Batch jobs were prepared off line using (essentially) manual systems to convert the job into a machine-readable form such as a paper or magnetic tape or a deck of punched cards. After execution, some output in an easily machine-writable form was produced, such as another tape or card deck, or a printed listing.

Over time, the power and reliability of computer systems increased much faster than its cost. However, *programmers* were becoming more expensive and less reliable as many new and ever more complex applications were attempted [Uzga74a]. Because the cost of programmers had begun to dominate the cost of computing, interactive timesharing systems were developed to simplify the programming task. Some efficiency of use of the computer resource was sacrificed in exchange for efficiency of use of the human resource.

This same increase in the power of computer systems also led to greater sharing of computing resources in an attempt to gain efficiency. Multiprogramming allowed several independent computations to progress "simultaneously". Furthermore, such applications as data bases, enquiry-response systems and electronic message systems led implicitly to the notion of shared data and cooperating processes. Thus, *communications* became an integral part of computing.

As integrated circuit technology improves, it is safe to assume that both the size and cost of reasonable "computing engines" will continue its sharp decline. It is expected that the cost of the computer hardware will become negligible compared to the total cost of many applications. Microcomputers are becoming commonplace in office equipment and are even beginning to appear in the home.

As computing power becomes decentralized, we expect that the sharing of data will become even more important. Sharing of data is not possible without some sort of communications network. Since we expect the processing elements that a network is to connect to become smaller, less inexpensive and more numerous, the network itself should be simple, inexpensive and capable of operation without central control.

Loosely speaking, a network consists of any medium that supports the exchange of information between stations together with a set of instructions that describes the way in which the network is used. In point-to-point networks, such as the telephone network or the ARPAnet [Klei76a], the medium consists of a set of links connecting pairs of stations, possibly including some additional switching centers to permit communications between stations having no direct connection. In broadcast networks, the medium consists of a single channel for sending information. If any station transmits a message, it will be received by every other station, thereby providing a direct connection between every pair of stations.

Broadcast networks come in many forms, each with different performance characteristics because of the physical properties of the particular type of multiple access channel that is employed. In satellite networks [Abra73a, Lam74a, Jaco78a], it is assumed that a set of widely separated ground stations wish to communicate by relaying messages through a transponding satellite. In addition, it is usually assumed that neither the ground stations nor the satellite are mobile, so that all ground stations can synchronize their transmissions to arrive at the satellite within *slots* of duration equal to a message transmission time. Since all ground stations can monitor the retransmission of each message as it is transponded by the satellite (including the sender), both positive and negative acknowledgements are provided at no cost. However, because of the long round-trip propagation time from the earth to an orbiting satellite and back to the earth (about 250 msec. for satellites in a geostationary orbit), this acknowledgement will arrive long after the transmission of the message is completed.

In ground radio packet switching [Robe72b, Kahn77a, Kahn78a], it is assumed that the stations are distributed over a small geographic area. In general, the error rate on ground radio channels is not negligible. In addition, it is often assumed that the stations are mobile, so that the topology of the network can change rapidly. Since messages are exchanged directly between nearby stations, the *propagation time* may be small enough for the "leading edge" of a message to arrive at its destination long before the end of the message has left its sender. Thus, the stations can take advantage of the current activity on the channel in making scheduling decisions. However, because of the *capture* effect, it is commonly assumed that a transmitting station will be unable to determine whether its messages were received correctly at their destinations (or "collided" with other messages) simply by monitoring channel activity.

In local "bus" networks [Metc76a, Chri78a, Raws78a], broadcast communications takes place along a coaxial cable or an optical fibre. The bus is used to provide communications in a small area, such as within a building or a complex of buildings, with a very low error rate. When a coaxial cable is used, the end-to-end propagation time over the network is much smaller than a typical message transmission time, providing useful feedback about the state of the channel to all stations in the network. When an optical fibre is used, transmission speeds can be increased dramatically, making the channel feedback information less useful. In addition, because signals propagate in only one dimension (*i.e.*, along the bus), there is no capture effect. Thus, messages can be coded in a manner that permits transmitting stations to detect collisions as they occur.

When the network is to support a very large number of "bursty" stations, *i.e.*, each station has a high peak to average load ratio, the simplicity and potential for sharing of the communications resource of broadcast networks becomes attractive. However, since it is assumed that the channel cannot support several transmissions simultaneously, such networks require some cooperation between the stations in the use of the channel. Multiple access protocols provide such a mechanism for sharing the channel under distributed control. Such protocols rely on relatively simple algorithms to grant stations access to the channel.

### 1.2: The Multiple Access Problem

Let us define the *multiple access problem* to be the efficient scheduling of a broadcast communications channel that is shared by a *distributed* population of stations to transmit *messages*. The channel is a resource that can transmit successfully only one message at a time in any region in space: if two or more nearby stations attempt to use the channel simultaneously, all transmissions are destroyed. However, each transmission has a limited range; if the system is distributed over a much larger region than the transmission range of a typical station, several widely separated stations can successfully transmit messages simultaneously. The complexity of the optimal scheduling problem increases dramatically with the "dispersion" of the stations. We suggest that the following set of problems is representative of this increasing complexity.

0. The *single queue* environment: All stations are located at the same point in space. Control information can be exchanged instantly and at no cost in channel capacity, so stations can form a queue to use the channel. This is just the classical single-server queueing problem.

1. The *one-hop* environment: Stations are not co-located in space, but they are sufficiently close together to share a single, common operating environment. All stations are within range of their intended destinations: each station hears all transmissions on the channel, and all simultaneous messages are lost. No access scheme can exceed the performance of a single server queue, but the stations cannot in general form a queue without using part of the channel to exchange explicit *protocol information*. However, we observe that partial control information is available to each station at no cost in channel capacity by monitoring the activity on the channel.

2. The *hidden station* environment: All stations are within range of their intended destinations, but each station hears only a subset of the transmissions on the channel. We define a *hearing graph* on the network as follows: the stations are the nodes; an arc from station $i$ to station $j$ will exist if either station can hear the transmissions of the other. The environment at the destination is no longer the same as at the source, so stations can no longer reliably determine whether a message will suffer interference at its destination by monitoring the channel at the source. Successful reception of simultaneous messages becomes possible, so the single server queue is no longer an upper bound for the performance of an access scheme. If the stations could form a queue to use the channel, the optimum use of the channel would require solving an (NP-complete) constrained minimal colouring problem on the hearing graph of stations in the queue. (If

have an infinite number of colours corresponding to different time slots. (If we allow buffered stations, a station with $n$ messages queued for transmission must be represented by $n$ nodes in the graph.) The capacity of the channel would then be the average ratio of the number of successful transmissions over the number of colours (*i.e.*, time slots) used to colour the hearing graph. Note that any channel that allows a positive rate of transmission has infinite capacity in the hidden station environment if we allow the ratio of the network diameter over the transmission range of a station to go to infinity. Hence, we shall often be more concerned with the capacity of a *region covered by a single transmission,* than with the total capacity in the hidden station environment.

3. The *multi-hop* environment: Stations need not be within range of their destinations, so messages must be forwarded along a *path* through a series of *repeaters* before reaching their final destination. The environments at the source, destination and each intermediate repeater will be different. The new issues of *congestion, routing, connectivity, locating,* and *hierarchical organization* become important. Even if all stations could form a queue to use the channel, solution of the colouring problem above cannot guarantee the best use of the channel. The flexibility to change the routing of each message requires us to solve the colouring problem for all possible combinations of paths for the messages in the queue.

### 1.3: Previous Work

The behaviour of systems in the single queue environment is well understood in the domain of queueing theory [Cohe69a, Klei75a, Klei76a]. The single-queue environment will not be considered here, except, perhaps, as a comparison for other systems.

There is a vast literature on the one-hop environment. Several authors [Mart70a, Heit76a, Klei76a, Klei77b, Klei79b, Toba80a, Lam79a] have attempted to classify various protocols and explain their advantages, problems and ranges of applicability. Until very recently, there have been many separate results for specific operating environments and analyses of *ad hoc* access schemes, but no unified understanding of the problem. The results for the hidden station environment are typically extensions of one-hop systems into the hidden station environment. Most results for the multi-hop environment are either gross approximations dependent on many strong assumptions, or provide only asymptotic bounds for very large systems [Akav78a, Klei78b].

A key parameter in choosing a particular protocol appears to be the dimensionless product of system performance constraints $\rho T$ [Akav78a, Lam78a], where $\rho$ is the required average throughput rate and $T$ is the allowable average delay. From Little's result [Litt61a], we know that under rather general assumptions, the average number in system equals the product of the average arrival rate and the average time in system. Hence, $\rho T$ is merely the allowable average number of messages in the system.

For a stable system, the channel capacity (measured in successful message transmissions per unit time) must exceed the *average* arrival rate of new messages. Scheduling conflicts (and hence delays) occur only because of statistical fluctuations in message lengths and arrival times. We may thus interpret $\rho T$ as a measure of the "burstiness" of our system. The answers to the multiple access problem in the limits of $\rho T \rightarrow 0$ and $\rho T \rightarrow \infty$ are clear. If the system must deliver messages too quickly to allow any queueing delays, the channel capacity must greatly exceed the *average* load; there will be no conflicts, and no arbitration between requests to use the channel are needed. If unbounded delays are acceptable, no capacity will be wasted if the channel is split to give each conversant pair of stations a dedicated, conflict-free subchannel whose capacity is proportional to their average transmission rates. Unfortunately, the definition of optimal (or even good sub-optimal) protocols for the most important case, namely $\rho T$ non-zero, finite and reasonably small, remains poorly understood.

The first multiple access protocols were *static scheduling algorithms* that allowed small populations of passive stations to communicate with a central controller [Mart70a]. The simplest such schemes involve central control using a *polling algorithm* [Konh74a, Schw77a]. The class of polling algorithms allows control of the channel to be passed from station to station according to a cyclic polling list: all stations are offered a turn to use the channel if they wait long enough. In roll-call polling, the controller has the only copy of the list, and names the station to transmit in each slot. In hub polling, each station must be aware of the polling list. The controller initiates a polling cycle by giving control to the first station in the polling cycle. Thereafter, each station passes control to the next one until the cycle is completed.

Such polling algorithms can be implemented in a distributed fashion. Time Division Multiple Access (TDMA), where the channel is split into a series of *slots* in the time domain that are assigned to stations in a round-robin fashion, may be thought of as an implementation of roll-call polling. MSAP [Scho76a, Klei77a] and BRAM [Chla79a] are implementations of hub polling. Control of the channel is passed between stations according to a cyclic priority list, and the "silence symbol" that delimits the end of a station's transmission signifies that control is passed to the next station in the cycle.

The class of *contention algorithms* requires the stations to actively compete for a turn to use the channel — a station can wait forever without having control of the channel explicitly given to him. In "pure" ALOHA [Abra73a], stations can transmit at any time without regard for the activity of other stations. If they receive no acknowledgement that their transmission was correctly received within a specified time, they retransmit the lost message, taking care to insert a further random delay to prevent two colliding messages from remaining deadlocked forever. Note that pure ALOHA has good delay characteristics under light load, but is wasteful of channel capacity. Furthermore, it can be shown [Klei75b, Lam75a] that ALOHA is inherently unstable because of the positive feedback from the retransmission of previously-collided messages. Fortunately, pure ALOHA is so simple that its performance does not further degrade in the hidden station or multi-hop environments.

Many extensions to the ALOHA protocol have been proposed to increase its performance and improve its stability. Roberts [Robe72a] suggested synchronizing the starting times of each message to increase channel capacity. In a slotted system, messages either destroy each other completely or not at all. If message propagation times across the network are small with respect to the transmission time of a message, the Carrier Sense Multiple Access (CSMA) protocols [Toba74a, Klei75c] can reduce wasted channel capacity in the one-hop environment. Before transmitting, stations sense the channel for activity to determine whether the channel is idle (in which case it can transmit) or is already in use. There is still a danger of collisions between widely separated stations because of race conditions. Whenever a station senses the channel idle and begins transmitting a message, there is a non-zero propagation delay before the "leading edge" of the message reaches the other stations. During this time, the other stations could sense the channel idle and begin transmitting their own messages.

By preventing stations from transmitting when the channel is busy, CSMA introduces the problem of what to do with the messages that arrive when the channel is busy. Several CSMA variants have been proposed to handle this problem, notably *non-persistent,* where stations sensing the channel busy wait a random time before trying again, *1-persistent,* where all stations sensing the channel busy transmit as soon as it is sensed idle, and *p-persistent,* where stations sensing the channel busy transmit with probability $p$ as soon as it is sensed idle. CSMA protocols were first proposed for ground radio packet networks, where a station cannot detect its own collisions. However, CSMA has also been applied to a coaxial cable where collision detection is possible [Metc76a]. Tobagi has also extended the analysis of CSMA to the hidden station environment [Toba75a] with a central destination using a technique of sending a busy tone on the acknowledgement channel.

Static protocols are designed for a particular operating environment; the operating environment includes the channel load, the size of the user population, etc. Adaptive schemes change as a function of their operating environment, progressively reducing contention as load increases. Recently, some good adaptive access schemes for the one-hop environment have been proposed and analyzed. Hayes [Haye78a] has analyzed an adaptive form of roll-call polling that involves *probing* groups of stations in search of ready stations. An entire group with no ready stations can immediately be ignored for the duration of the current polling cycle; groups with ready stations are further probed until single ready stations are isolated and allowed to transmit.

The tree algorithm of Capetanakis [Cape78a] is a distributed analogue of adaptive polling. The algorithm proceeds in a series of service epochs. A group of stations is selected for the first slot of a service epoch; should there be a collision, the stations that were involved in the collision are split up to resolve the collision, and all other stations must wait until that particular collision is completely resolved before continuing. This "split traffic upon collision" idea is applied recursively to form a binary tree whose leaves are idle and successful slots and whose internal nodes are collisions.

The URN scheme [Klei78a, Yemi80a] uses a partial reservation channel to find a good estimate of the number of ready stations. Reservations consist of sending a signal on a ternary channel of small capacity to announce that the station has received a new message and entered the *ready* state. Stations listening to the channel can determine whether zero, one or more than one station received a new message in the current slot. Each station estimates the number of ready stations as the difference between the number of attempted reservations and the number of successful transmissions on the message channel. The estimate is reset at the end of a busy period on the channel. This estimate is then used to choose the optimum number of stations to be allowed to transmit in the current slot to maximize the probability of a successful transmission.

### 1.4: Contributions of This Work

#### 1.4.1: Local Optimality Conditions for Protocols

We have developed a rather general *local* optimality condition for all synchronous multiple access protocols that can be described by an imbedded discrete time Markov chain. This condition provides a rule for selecting transmission rights for each station that maximizes the expected channel utilization over all slots during which the system is in that particular state. Many known protocols are special cases of this general rule, ranging from ALOHA and CSMA to MSAP, TDMA and the URN scheme. In addition, the method can be used to find the conditions under which each of these protocols is (locally) optimal.

This method can also be extended to the hidden station environment. Below, we show how this optimality rule can be used to find the optimum transmission probability (*i.e.*, its "coin bias") for each station in a heterogeneous multihop ALOHA network that maximizes the capacity of the network for a "fixed" (up to a scalar multiple) traffic matrix.

#### 1.4.2: The Asymptotic Behaviour of Multiple Access Networks

In the past, Kleinrock [Klei79a] has examined the asymptotic behaviour of resource sharing models under *centralized* control. He established a "scaling effect", where the perceived performance of a system for each user improves as the size of the system increases. In addition, he observed from the "law of large numbers" that very large probabilistic systems exhibit a deterministic behaviour.

Below, we address the asymptotic behaviour of multiple access systems under *distributed* control. We identify a new phenomenon in these systems. Let us take a fixed traffic intensity (below the nominal channel capacity), distribute it uniformly among $M$ homogeneous stations, and then require those stations to transmit that traffic over the channel. We show that there exists a level of traffic intensity, called the *infinite population channel capacity*, beyond which the average delay grows at least linearly with $M$ for *any* realizable distributed multiple access protocol. Furthermore, for traffic intensities below the infinite population channel capacity, the delay performance for some recently devised protocols is almost completely insensitive to $M$. We have made significant contributions to both the determination of the infinite population

capacity, and to the study of this new class of multiple access protocols.

### 1.4.3: Multiple Access in Local Area Networks

Many local area networks are now being built using variations of the CSMA protocols that were extensively analyzed by Tobagi and Kleinrock. A major drawback to these networks is the long and variable delays inherent in any system that depends on *random retransmission* to resolve collisions. This has limited the applicability (or, at the very least, placed severe restrictions on the allowable channel utilization) of these networks to such "real time" applications as packetized voice.

We have defined a new class of CSMA protocols, *virtual time CSMA*, that offers significantly improved delay characteristics over other CSMA protocols. These virtual time CSMA protocols obey the local optimality conditions that we described above. In addition, we have proven that minislotted virtual time CSMA is optimal over all possible CSMA variants under some commonly applied assumptions.

All CSMA protocols are extensions of the ALOHA protocol, including virtual time CSMA. It is now apparent that the new class of "tree" conflict resolution algorithms offers significant advantages over ALOHA in the case of constant slot sizes and active acknowledgements. We extend these protocols to the case where a *partial reservation* channel is provided. We define a partial reservation request to contain only a few "bits" of information, such as the binary message: "no station began transmitting" or "at least one station began transmitting" that the initial propagation time in a CSMA network provides. Other mechanisms for providing such a partial reservation channel for a local area network have also been suggested, including an auxiliary channel carrying the logical "OR" of a ready bit [Hama80a], or a sequence of ready bits [Haye78a, Mark80a], for each station.

We have developed a class of Hybrid Carrier Sense — Binary Search protocols for such an environment with exceedingly good performance. In fact as $M \rightarrow \infty$, we show that its performance exceeds a naive extension of the infinite population capacity results to the carrier sense environment.

## 1.5: Outline of the Dissertation

In Chapter 2, we give a Markovian interpretation for the operation of synchronous multiple access protocols. We derive a local optimality condition and examine several well known protocols to see how they relate to the local optimality condition. Chapter 3 begins the study of infinite population protocols, and includes a survey of the work on collision resolution tree algorithms. In Chapter 4 we derive some upper bounds on the capacity of the best possible infinite population protocols. We consider both arbitrarily complex algorithms, and the class of "degenerate intersection" protocols (a slight generalization of first-come first-served that includes all currently proposed protocols). Other recent work on capacity bounds is also described.

Chapter 5 introduces the idea of infinite population protocols aided by a partial reservation channel. We examine both binary and ternary reservation channels, and show that the binary channel introduces some interesting new problems that were not present in the original formulation of the infinite population protocols described in Chapters 3 and 4. Chapters 6 and 7 analyze some specific partial reservation-aided infinite population protocols suitable for local networks. In Chapter 6, we discuss the virtual time CSMA protocol, and show that it is the optimal CSMA protocol under some common assumptions. In Chapter 7, we examine a family of hybrid protocols that are specifically designed to take advantage of a common characteristic of local networks, namely that idle slots can be much shorter than collisions. Chapter 8 gives some specific protocol recommendations for local networks and lists some suggestions for future work.

# CHAPTER 2
# Local Optimality in Protocols

## 2.1: A Markovian Model of Protocols

Throughout most of this dissertation, we shall limit our discussions to the one-hop environment. Consider a distributed population of stations using a synchronous multiple access protocol to exchange messages over a noiseless communications channel. In a *synchronous* protocol, all message transmissions fall into constant length *slots* of duration equal to a message transmission time. Roberts [Robe72a] has shown that this will increase the channel capacity significantly: messages destroy each other completely or not at all. The transmissions in a slot can have three outcomes: an *empty slot*, when no station transmits, a *success*, when exactly one station transmits, and a *collision*, when at least two stations transmit simultaneously. We assume that at the end of each slot all stations receive, at no cost, an acknowledgement whenever a message is sent successfully and a non-acknowledgement whenever there is a collision, and that the protocol may depend on the (possibly infinite) history of activity on the channel.

Given a particular number $M$ of stations, an arrival process and a collision resolution algorithm, such a multiple access protocol may conveniently be described as a discrete time Markov process. We note that this results in no loss of generality in our protocols, since an arbitrary amount of information may be encoded into the state description. We say that a protocol is *stable* if a stationary probability distribution exists for the Markov process, *i.e.*, if the backlog of unserviced arrivals to the system remains finite with probability 1, and we define *capacity* to be the supremum over all arrival rates such that the protocol is stable. For example, with a *finite* number of stations, Tsybakov and Mikhailov [Tsyb80a] have shown that $N(t) \triangleq (n_1(t), n_2(t), \ldots, n_M(t))$, where $n_i(t)$ is the queue length at the $i^{th}$ station at time $t$, is a suitable state description for the (memoryless) ALOHA protocol [Abra73a]. Among other results, they proved that the capacity of ALOHA is at least $e^{-1}$ by showing the existence of parameters for which $N(t)$ is ergodic whenever the sum of the station arrival rates is less than $e^{-1}$.

Efficient multiple access protocols should not necessarily prevent all collisions. Instead, they should choose strategies that are likely to yield a high channel utilization when there are waiting messages. Hence distributed algorithms of the following form should be used to schedule the transmission attempts on the channel.

Given the following:

1. a set of $M$ stations,

2. a set of states corresponding to different levels or phases of channel activity,

3. the current state of the system,

4. for each possible subset of stations, the conditional probability that enabling that set of stations to transmit in the current slot will lead to an idle slot, a success, or a collision, given the current state of the system, and

5. a universally agreed upon priority ordering of the stations for the current slot.

Choose:

1. a transmission probability for each member of the population.[1]

2. the next state of the system, given the current state and the outcome of the current slot.

Since the next state of the system is assumed to depend only on the current state and outcome of the current slot, the behaviour of the protocol is Markovian. In general, we may not be able to optimize over all states simultaneously to maximize the efficiency of the protocol. Below, we show how to create good suboptimal protocols by partitioning the problem.

### 2.2: A Local Optimality Condition

Let $\rho$, $\rho \leqslant 1$, be the mean channel utilization, *i.e.*, the fraction of time that successful message transmissions (as opposed to idle slots or collisions) are occurring on the channel. A good scheduling algorithm must be capable of achieving a high channel utilization, and must deliver messages with only a small average delay, $T$. For any value of $\rho$, $T$ is proportional to the average number of messages in the system, $N$, by Little's result. If the message lengths are all drawn independently from the same distribution, and if the scheduling algorithm does not discriminate between messages according to their service requirement (*i.e.*, length), we can apply Kingman's result [King62a] for GI/G/1 queues to show that *mean* delays are invariant with respect to the order in which messages are transmitted, and the *variance* of delays is minimized by the first-come first-served queueing discipline. Thus a sufficient condition for a strategy to be optimal is emptying the system of waiting messages more quickly than any other strategy. Unfortunately, finding an optimal strategy can be a hard combinatorial problem. We thus expect that investigating some reasonable heuristics will be profitable.

An optimal strategy must have the highest channel utilization averaged over *all* states where there are ready stations. Below, we consider a *locally* optimal strategy that independently maximizes the average channel utilization for *each* non-empty system state. This uncouples the various states of the system. The optimization problem is reduced to an assignment of

[1] Yemini [Yemi80a] has shown that it is best to choose these probabilities to be either 0 or 1, so we shall often view this assignment of probabilities as enabling a subset of the population to transmit in the next slot.

transmission probabilities to each station to maximize the conditional throughput over all slots when the system is in that state.

It is easy to see that it is best to choose each of these probabilities to be either 0 or 1, *i.e.*, a "pure" strategy [Yemi80a]. No "fractional" messages are ever sent, so after the probabilistic experiment has been performed we are always left with some (possibly random) pure strategy for each slot. Thus the performance of any probabilistic strategy is given by a convex combination of pure strategies. Since no such *combination* of strategies can attain a higher channel utilization than the *best* strategy, it is clear that a pure stategy must be optimal.

Any pure strategy may be viewed as enabling a subset of the population to transmit in the current slot. Since there is an assumed priority ordering among the stations (which may change at each slot), the optimization problem for pure strategies is reduced to finding the optimal number $k^*$ of stations that should be enabled to transmit in the current slot. Sometimes the selection of the number of transmitters, $k$, is fixed by other considerations. Here we have a "dual" problem: select a set of *transmission probabilities* $\{p'_i\}$ so that some fixed $k$ is locally optimal and the expected channel utilization is maximized for the given $k$.

This local policy may not be globally optimal however, because it does not do long range planning to try to remain in "good" states. Consider a system where messages arrive as a sequence of *arrival points* in discrete time from a single Bernoulli source. Each arrival point independently contains either exactly one message, with probability $p$, or no message, with probability $1-p$. Whenever an arrival point contains a message, that message enters the system and is assigned to one station at random. We assume that the number of stations is so large that it makes more sense to choose enabled sets consisting of arrival points than stations. (We shall examine this system in considerable detail in Chapter 4.) Let $p$ be $1/2+\epsilon$, $0 < \epsilon << 1$. Enabling *single* arrival points, *i.e.*, "TDMA", is locally optimal, giving a channel utilization of $p \gtrsim 1/2$.[1] Alternately, *pairs* of arrival points could be enabled simultaneously. Since *both* arrival points could contain a message, we have introduced the possibility of a collision with probability $p^2$. However, since a collision tells us that *at least two* messages were transmitted, and two arrival points can contain *at most two* messages, it follows immediately that both enabled arrival points must contain messages. These two (colliding) messages can be transmitted without interference in the next two slots. The channel utilization with this second strategy is given by

$$\frac{E[\#\ successes\ per\ service\ epoch]}{E[length\ of\ service\ epoch]} = \frac{2p}{1+2\cdot p^2} \gtrsim \frac{2}{3},$$

which clearly exceeds the performance of the locally optimal policy. Thus, the performance of a locally optimal strategy may be significantly poorer than the globally optimal strategy.

Let us characterize the behaviour of our locally optimal scheduling policy in an arbitrary system state $i$. (Because it is a local policy, all other states can be ignored). Let $\rho_k$ be the expected throughput in state $i$ if the first $k$ stations in priority order are given permission to transmit. We shall permit three sizes of slots: we assume that the length of a successful

[1] We shall use the notation "$a \gtrsim b$" to represent the relation "*a is slightly greater than b*".

transmission probabilities to each station to maximize the conditional throughput over all slots when the system is in that state.

It is easy to see that it is best to choose each of these probabilities to be either 0 or 1, *i.e.*, a "pure" strategy [Yemi80a]. No "fractional" messages are ever sent, so after the probabilistic experiment has been performed we are always left with some (possibly random) pure strategy for each slot. Thus the performance of any probabilistic strategy is given by a convex combination of pure strategies. Since no such *combination* of strategies can attain a higher channel utilization than the *best* strategy, it is clear that a pure stategy must be optimal.

Any pure strategy may be viewed as enabling a subset of the population to transmit in the current slot. Since there is an assumed priority ordering among the stations (which may change at each slot), the optimization problem for pure strategies is reduced to finding the optimal number $k*$ of stations that should be enabled to transmit in the current slot. Sometimes the selection of the number of transmitters, $k$, is fixed by other considerations. Here we have a "dual" problem: select a set of *transmission probabilities* $\{p'_i\}$ so that some fixed $k$ is locally optimal and the expected channel utilization is maximized for the given $k$.

This local policy may not be globally optimal however, because it does not do long range planning to try to remain in "good" states. Consider a system where messages arrive as a sequence of *arrival points* in discrete time from a single Bernoulli source. Each arrival point independently contains either exactly one message, with probability $p$, or no message, with probability $1-p$. Whenever an arrival point contains a message, that message enters the system and is assigned to one station at random. We assume that the number of stations is so large that it makes more sense to choose enabled sets consisting of arrival points than stations. (We shall examine this system in considerable detail in Chapter 4.) Let $p$ be $1/2+\epsilon$, $0<\epsilon<<1$. Enabling *single* arrival points, *i.e.*, "TDMA", is locally optimal, giving a channel utilization of $p \gtrsim 1/2$.[1] Alternately, *pairs* of arrival points could be enabled simultaneously. Since *both* arrival points could contain a message, we have introduced the possibility of a collision with probability $p^2$. However, since a collision tells us that *at least two* messages were transmitted, and two arrival points can contain *at most two* messages, it follows immediately that both enabled arrival points must contain messages. These two (colliding) messages can be transmitted without interference in the next two slots. The channel utilization with this second strategy is given by

$$\frac{E[\#\ successes\ per\ service\ epoch]}{E[length\ of\ service\ epoch]} = \frac{2p}{1+2\cdot p^2} \gtrsim \frac{2}{3},$$

which clearly exceeds the performance of the locally optimal policy. Thus, the performance of a locally optimal strategy may be significantly poorer than the globally optimal strategy.

Let us characterize the behaviour of our locally optimal scheduling policy in an arbitrary system state $i$. (Because it is a local policy, all other states can be ignored). Let $\rho_k$ be the expected throughput in state $i$ if the first $k$ stations in priority order are given permission to transmit. We shall permit three sizes of slots: we assume that the length of a successful

[1] We shall use the notation "$a \gtrsim b$" to represent the relation "*a is slightly greater than b*".

transmission is unity, that the length of an idle slot is $a$, and that the length of a collision is $b$. Because the length of a slot is dependent on the scheduling outcome during that slot, the conditional throughput will not simply be the probability that there is a successful transmission in a slot randomly chosen from among those with the same state information. However, we can make the calculation simply by borrowing a technique from renewal theory; we define a virtual time axis including only those slots during which the protocol has the same state information. The conditional throughput will then be the probability that an observation uniformly distributed over this virtual time axis intercepts a successful transmission. For brevity, let us define

$$\begin{aligned} I_k &\triangleq \Pr[\text{idle slot} \mid \text{permission given to } k, \text{ state } i] \\ S_k &\triangleq \Pr[\text{success} \mid \text{permission given to } k, \text{ state } i] \ . \\ C_k &\triangleq \Pr[\text{collision} \mid \text{permission given to } k, \text{ state } i] \end{aligned}$$

Then $\rho_k$ will be

$$\rho_k \triangleq \frac{1 \cdot S_k}{a \cdot I_k + 1 \cdot S_k + b \cdot C_k} = \frac{S_k}{b - (b-a) I_k + (1-b) S_k} \tag{2.1}$$

It now remains to find the value of $k$ that maximizes the conditional throughput.

**Lemma 2.1:**

Let $f$ be a function defined on the positive integers. If either $f_k \geqslant f_{k+1}$ holds whenever $f_{k-1} \geqslant f_k$, or $f_k \geqslant f_{k-1}$ holds whenever $f_{k+1} \geqslant f_k$, then $f$ is unimodal.

**Proof:**

Let $f_i$, $f_j$ be two distinct strict local maxima, *i.e.*,

$$f_{i-1} < f_i > f_{i+1},$$

and

$$f_{j-1} < f_j > f_{j+1}.$$

Without loss of generality, let $j > i$. Then, by the first condition above, we must have

$$f_i \geqslant f_{i+1} \geqslant \cdots \geqslant f_j,$$

which contradicts that $f_j$ is a strict local maximum. Similarly, by the second condition, we must have

$$f_j \geqslant f_{j-1} \geqslant \cdots \geqslant f_i,$$

which contradicts that $f_i$ is a strict local maximum.

■

**Theorem 2.1:**

The sequence $\{\rho_k\}$ is unimodal in $k$.

**Proof:**

Using Lemma 2.1, to prove the theorem, it is sufficient to prove that

$$\rho_{k+1} \geqslant \rho_k$$

implies that

$$\rho_k \geqslant \rho_{k-1}.$$

Let $\rho_k \geqslant \rho_{k-1}$. Then, substituting Eq. (2.1) into this condition, we obtain

$$\frac{S_k}{b-(b-a)I_k+(1-b)S_k} \geqslant \frac{S_{k-1}}{b-(b-a)I_{k-1}+(1-b)S_{k-1}}$$

or

$$S_k(b-(b-a)I_{k-1}) \geqslant S_{k-1}(b-(b-a)I_k).$$

Since $S_k = (1-p_k)S_{k-1}+p_k I_{k-1}$ and $I_k = (1-p_k)I_{k-1}$, we obtain

$$b(1-p_k)S_{k-1}+(a-b)p_k I_{k-1}^2+bp_k I_{k-1} \geqslant bS_k$$

or

$$\frac{S_{k-1}}{I_{k-1}} \leqslant 1 - \left[1-\frac{a}{b}\right] I_{k-1}. \tag{2.2}$$

Thus, let us assume that $\rho_{K+1} \geqslant \rho_K$. Using Eq. (2.2), it is equivalent to assume

$$\frac{S_k}{I_k} \leqslant 1 - \left[1-\frac{a}{b}\right] I_k.$$

But

$$\frac{S_k}{I_k} = \frac{S_{k-1}}{I_{k-1}} + \frac{p_k}{1-p_k},$$

so that to show that Eq. (2.2) holds, it is sufficient to show that

$$\frac{p_k}{1-p_k} \geqslant p_k\left[1-\frac{a}{b}\right]$$

or

$$1 \geqslant (1-p_k)\left[1-\frac{a}{b}\right],$$

which is clearly true since $a/b$ is positive and $0 \leqslant (1-p_k) \leqslant 1$.

■

Since we have now proven that $\{\rho_k\}$ is unimodal in $k$, it is clear that the throughput is (locally) maximized by enabling the first $k^*$ stations in priority order to transmit, where $k^*$ is the largest value of $k$ for which $\rho_k \geqslant \rho_{k-1}$ holds. Substituting the definitions of $S_{k-1}$ and $I_{k-1}$ into Eq. (2.2), we finally obtain $k^*$ is the largest value of $k$ for which the inequality

$$\sum_{i=1}^{k-1} \frac{p_i}{1-p_i} \leqslant 1 - \left(1 - \frac{a}{b}\right) \prod_{j=1}^{k-1} (1-p_j) \tag{2.3}$$

still holds. We note that in the special case of $a = b$, this condition reduces to a particularly simple form, namely

$$\sum_{i=1}^{k-1} \frac{p_i}{1-p_i} \leqslant 1.$$

We now show the effect of the priority ordering on the conditional throughput.

**Theorem 2.2:**

The conditional throughput is maximized by a priority order that ranks stations in order of *decreasing probability of being ready.*

**Proof:**

We proceed by showing that an arbitrary priority ordering can be sorted into the above order in such a way that the maximum throughput is non-decreasing at each step. Without loss of generality, let us assume that the stations are numbered in order of decreasing probability of being ready, *i.e.* $p_i \geqslant p_j$ iff $i \leqslant j$, but that the initial priority ordering is arbitrary. Let us define $R \triangleq \{r_1, \ldots, r_M\}$ to be the set of station numbers given in the current priority order (*i.e.*, a permutation of $\{1, \ldots, M\}$). We observe that $\rho_{k^*}$ (where $k^*$ is calculated with respect to the initial priority ordering) is invariant with respect to permutations of either of the sets $\{r_1, \ldots, r_{k^*}\}$ or $\{r_{k^*+1}, \ldots, r_M\}$. Hence, the new maximum throughput after either type of permutation cannot decrease. Furthermore, since we need only the set $\{p_{r_1}, \ldots, p_{r_{k^*}}\}$ to show that $\rho_{k^*} \geqslant \rho_{k^*+1}$, the new optimal $k$ with respect to the permuted priority list can only *decrease.* We may therefore sort the set $\{r_1, \ldots, r_{k^*}\}$, recalculate $k^*$, and finally sort the set $\{r_{k^*+1}, \ldots, r_M\}$ at any time without decreasing throughput. To complete the proof, it remains to show that if $p_{r_{k^*+1}} > p_{r_{k^*}}$ (*i.e.*, $R$ is not completely sorted), then exchanging the priorities of these two stations does not decrease the throughput with $k^*$ stations, *i.e.*,

$$\rho'_{k^*} \geqslant \rho_{k^*}$$

or

$$S'_{k^*}[b-(b-a)I_{k^*}] \geqslant S_{k^*}[b-(b-a)I'_{k^*}]$$

but $I'_{k^*} = \frac{1-p_{k^*+1}}{1-p_{k^*}} I_{k^*}$ and $S'_{k^*} = \frac{1-p_{k^*+1}}{1-p_{k^*}} S_{k^*} + \frac{p_{k^*+1}-p_{k^*}}{(1-p_{k^*})^2} I_{k^*}$, so the condition may be

rewritten as

$$\frac{p_{k^*+1}-p_{k^*}}{(1-p_{k^*})^2}[b-(b-a)I_{k^*}]I_{k^*} \geq \frac{p_{k^*+1}-p_{k^*}}{1-p_{k^*}}bS_{k^*}$$

or

$$\left(\frac{a}{b}-1\right)\frac{I_{k^*}^2}{1-p_{k^*}} \geq S_{k^*}-\frac{I_{k^*}}{1-p_{k^*}}$$

but $S=(1-p_{k^*})\,S_{k^*-1}+p_{k^*}\,I_{k^*-1}$ and $I=(1-p_{k^*})\,I_{k^*-1}$, so we may finally rewrite the condition as

$$\left(\frac{a}{b}-1\right)I_{k^*-1}^2 \geq S_{k^*-1}-I_{k^*-1},$$

which follows immediately from Eq. (2.3) and the optimality of $k^*$.

■

To summarize, the locally optimal decision rule, given both the state of the system (for which we have estimated the conditional transmission probabilities $\{p_i\}$) and the priority ordering, is to choose the first $k^*$ stations in priority order, where $k^*$ is the largest integer such that the inequality of Eq. (2.3) is still true. If we are free to choose the priorities, they should be in order of decreasing probability of being busy. We are still left with the task of determining the current state of the system and estimating the set of state-conditional transmission probabilities. We shall now show how certain "classical" multiple access protocols relate to this local optimality condition. Many of them can be derived as special cases of Eq. (2.3) when suitable assumptions about the statistical properties of the population and of the allowable states of the channel are made.

### 2.3: Slotted ALOHA

Let us assume that all slots are of fixed size, *i.e.*, $a=b=1$, and that all messages are of fixed length equal to the slot size. We note that this corresponds to a "central station" model, such as the original ALOHANET [Abra73a] or a satellite channel [Lam74a], since we require all transmissions to be perfectly synchronized into the slots. We do not wish to impose any requirement for coordination between stations, so we shall require all $M$ stations to have equal priority. This can be accomplished by granting permission to transmit to each station whenever it receives a new message without regard for the other stations (*i.e.*, all stations are always enabled). We see from Eq. (2.3) that this policy is locally optimal if

$$\sum_{i=1}^{M-1}\frac{p_i}{1-p_i} \leq 1$$

holds for the current state. This condition is clearly true if the sufficient condition $\max_i\{p_i\} \leq \frac{1}{M}$ holds, or, for statistically identical stations, if the necessary and sufficient condition

$$p_i = p \leqslant \frac{1}{M} \quad \forall i$$

holds. In the Poisson limit as $p \to 0$, holding $\lambda \triangleq Mp$ constant, we obtain $\lambda \leqslant 1$ for local optimality.

### 2.3.1: Stability of ALOHA

It is well known that ALOHA suffers from stability problems. Consider the following example with homogeneous stations. Whenever a collision occurs, we know that there are at least two stations ready to send messages. The probability that any randomly chosen *single* station has a message, given that we know that exactly $m$ out of $M$ stations have a message to send, is simply $\frac{m}{M} \geqslant \frac{2}{M}$. However, the conditional information that a collision has occurred introduces a dependence in the (new) set of probabilities $\{p_i\}$. Thus we cannot use Eq. (2.3) directly. Fortunately, this dependence has a simple form so that the probability of a success when $N$ out of the $M$ stations are enabled can still be calculated exactly. This success probability is a weighted sum of hypergeometric terms. Let $H(1, N, m, M)$ be the term from the hypergeometric distribution representing the probability of selecting exactly one busy station when $N$ stations are enabled, given that exactly $m$ out of $M$ stations were busy originally. Then

$$\begin{aligned} Pr[success|M, N, p] &= \sum_{m=2}^{M} H(1, N, m, M) \cdot Pr[m \text{ messages in } M | collision] \\ &= \sum_{m=2}^{M} \frac{\binom{m}{1}\binom{M-m}{N-1}}{\binom{M}{N}} \cdot \frac{\binom{M}{m} p^m (1-p)^{M-m}}{1-(1-p)^M - Mp(1-p)^{M-1}} \\ &= \frac{Np\left[(1-p)^{N-1} - (1-p)^{M-1}\right]}{1-(1-p)^M - Mp(1-p)^{M-1}}. \end{aligned} \tag{2.4}$$

In the Poisson limit as $p \to 0$, $Mp \to \lambda$ and $Np \to \gamma$, we obtain

$$Pr[success|\lambda, \gamma] = \frac{\gamma\left(e^{-\gamma} - e^{-\lambda}\right)}{1-(1+\lambda)e^{-\lambda}}. \tag{2.5}$$

It is clear why this probability of success following a collision with uncontrolled ALOHA must be zero, since the same set of ready stations will surely collide again. We conclude that ALOHA can only be optimal in a *loss* system.

### 2.3.2: Controlled ALOHA

In the previous section, we showed that uncontrolled ALOHA will deadlock when the first collision occurs unless blocked messages are lost. We now show how the condition of Eq. (2.3) suggests a flow control scheme to stabilize ALOHA that assigns *probabilistic* transmission rights to each station.

Suppose we were told the exact number of ready stations, $m$, at the beginning of each slot. If each ready station were to decide independently whether to actively seek permission to transmit in that slot with probability $1/m$, then the ALOHA properties of equal priorities and independent decision-making among the stations would be preserved. Since there are only $m$ ready stations in the system, it is also a locally optimal policy to grant all active stations permission to transmit in that current slot. If it could be implemented, such a policy must be stable when the arrival rate is strictly less than $1/e$, even as $M \rightarrow \infty$. This follows because the average channel utilization (*i.e.*, service rate) over each non-empty state is $(1-1/m)^{m-1} \geq 1/e$.

In general, the exact number of ready stations is seldom known. For finite $M$, we can still achieve stability (at the cost of long delays under light load) without any knowledge of the number of ready stations. We require that the condition of Eq. (2.3) is met even in the worst case, and let each ready station independently decide to transmit in a slot with probability $1/M$.

### 2.4: CSMA

As was the case with ALOHA, it is again our goal to define a protocol that permits stations to operate in a manner that minimizes the amount of coordination required among the stations. However, we now assume that each station can monitor channel activity through carrier sensing, so that unsuccessful slots can be cut short to improve the efficiency of the protocol.

Unlike the central destination model for slotted ALOHA, it is commonly assumed in the analysis of CSMA protocols that every station may wish to transmit to every other station. Thus, to guarantee that each message will be received within the slot in which it was transmitted, we must adjust the length of each type of slot (*i.e.*, idle, success or collision) to account for the propagation time. More precisely, the length of each slot is now assumed to be just long enough to guarantee that every station is aware of its type and thus able to determine the starting time of the next slot. Thus, to distinguish our notation for the arbitrary destination model from our previous notation for the central destination model, we now assume that the length of an idle slot is $a'$ (a propagation time), that the length of a slot that carries a successful transmission (of unit length) is $1+a'$, and that the length of a collision is $b'+a'$, where $b'$ is taken to be the fraction of each message that gets transmitted during a collision before the sending stations abort their transmissions. It is common to assume that $b'=1$ in local radio networks and $b' \approx 0$ in local coaxial cable networks [Metc76a].

For this carrier sense environment, the throughput equation becomes

$$\rho_k \triangleq \frac{1 \cdot S_k}{a' \cdot I_k + (1+a') \cdot S_k + (b'+a') \cdot C_k} = \frac{S_k}{b'+a' - b' I_k + (1-b') S_k}, \tag{2.6}$$

which is equal to $\frac{1}{1+a'}$ times the solution of Eq. (2.1) evaluated at $a=\frac{a'}{1+a'}$, $b=\frac{a'+b'}{1+a'}$. It follows that the properties of unimodality, local optimality, and optimum priority assignment for inhomogeneous stations that hold for Eq. (2.1) also hold for Eq. (2.6).

For homogeneous stations, it can be shown from Eq. (2.3) that an optimal policy consists of picking $k^* = \min(M, \lfloor x \rfloor)$ stations to transmit in each slot, where $x$ solves

$$x = \frac{1}{p}\left[1 - \left(1 - \frac{a}{b}\right)(1-p)^x\right] = \frac{1}{p}\left[1 - \left(\frac{b'}{b'+a'}\right)(1-p)^x\right]. \tag{2.7}$$

When $b' = 0$, this result reduces to $k^* = \min(M, \lfloor \frac{1}{p} \rfloor)$. In the limit as $M \to \infty$, $\frac{x}{M}$ becomes the optimal Poisson traffic intensity $\gamma$, where $\gamma$ solves

$$\gamma = 1 - \left(\frac{b'}{b'+a'}\right)e^{-\gamma}. \tag{2.8}$$

While this equation does not have a closed form solution in general, an iterative numerical solution can readily be found. For example, Newton iteration could be used with

$$\gamma_N = 1 + \gamma_{N-1} - \frac{\gamma_{N-1}}{1 - \left(\frac{b'}{b'+a'}\right)e^{-\gamma_{N-1}}},$$

giving quadratic convergence. Since $\gamma$ is also the intersection point of the curves $y = 1 - x$ and $y = \left(\frac{b'}{b'+a'}\right)e^{-x}$, we see that $\gamma$ is a unique function of the ratio $\frac{a'}{b'}$: $\gamma = 1$ when $b' = 0$, and, using Taylor series expansions, $\gamma \approx \sqrt{2/b'}$ in the limit as $\frac{a'}{b'} \to 0$.

Figure 2.1 shows maximum throughput as a function of the lengths of idle slots and collisions in the infinite population case. It is clear that reducing the detect times for idle slots and/or collisions increases the capacity of the channel. We observe that the maximum achievable throughput is more sensitive to reductions of the idle detect time than the collision detect time. This is most fortunate for system designers since it is easier to design stations that can distinguish between an idle channel and a busy channel than those that can detect and suppress collisions. This is especially true on a radio channel. It is not difficult for an *inactive* station to detect energy on the channel. However, because of the *capture* phenomenon, an *active* station may be completely unaware of any other channel activity because the energy of its own transmission is (locally) so much greater than any other signal. Thus active simultaneous acknowledgements from an idle observer may be required to detect and suppress collisions.

Because of the same feedback problem described for ALOHA above, CSMA does not meet our optimality condition unless we have a loss system. Moreover, the variable length of slots in CSMA introduces a further difficulty beyond stability problems. We have shown that for CSMA to be locally optimal, the offered load *per slot* must be a *constant*, even though the number of messages available for transmission in any slot depends on the sequence of outcomes in the past slots. In general, 1-persistent and $p$-persistent CSMA cannot provide a con-

For homogeneous stations, it can be shown from Eq. (2.3) that an optimal policy consists of picking $k* = \min(M, \lfloor x \rfloor)$ stations to transmit in each slot, where $x$ solves

$$x = \frac{1}{p}\left[1 - \left(1 - \frac{a}{b}\right)(1-p)^x\right] = \frac{1}{p}\left[1 - \left(\frac{b'}{b'+a'}\right)(1-p)^x\right]. \tag{2.7}$$

When $b' = 0$, this result reduces to $k* = \min(M, \left\lfloor \frac{1}{p} \right\rfloor)$. In the limit as $M \to \infty$, $\frac{x}{M}$ becomes the optimal Poisson traffic intensity $\gamma$, where $\gamma$ solves

$$\gamma = 1 - \left(\frac{b'}{b'+a'}\right)e^{-\gamma}. \tag{2.8}$$

While this equation does not have a closed form solution in general, an iterative numerical solution can readily be found. For example, Newton iteration could be used with

$$\gamma_N = 1 + \gamma_{N-1} - \frac{\gamma_{N-1}}{1 - \left(\frac{b'}{b'+a'}\right)e^{-\gamma_{N-1}}},$$

giving quadratic convergence. Since $\gamma$ is also the intersection point of the curves $y = 1 - x$ and $y = \left(\frac{b'}{b'+a'}\right)e^{-x}$, we see that $\gamma$ is a unique function of the ratio $\frac{a'}{b'}$: $\gamma = 1$ when $b' = 0$, and, using Taylor series expansions, $\gamma \approx \sqrt{2/b'}$ in the limit as $\frac{a'}{b'} \to 0$.

Figure 2.1 shows maximum throughput as a function of the lengths of idle slots and collisions in the infinite population case. It is clear that reducing the detect times for idle slots and/or collisions increases the capacity of the channel. We observe that the maximum achievable throughput is more sensitive to reductions of the idle detect time than the collision detect time. This is most fortunate for system designers since it is easier to design stations that can distinguish between an idle channel and a busy channel than those that can detect and suppress collisions. This is especially true on a radio channel. It is not difficult for an *inactive* station to detect energy on the channel. However, because of the *capture* phenomenon, an *active* station may be completely unaware of any other channel activity because the energy of its own transmission is (locally) so much greater than any other signal. Thus active simultaneous acknowledgements from an idle observer may be required to detect and suppress collisions.

Because of the same feedback problem described for ALOHA above, CSMA does not meet our optimality condition unless we have a loss system. Moreover, the variable length of slots in CSMA introduces a further difficulty beyond stability problems. We have shown that for CSMA to be locally optimal, the offered load *per slot* must be a *constant*, even though the number of messages available for transmission in any slot depends on the sequence of outcomes in the past slots. In general, 1-persistent and $p$-persistent CSMA cannot provide a con-

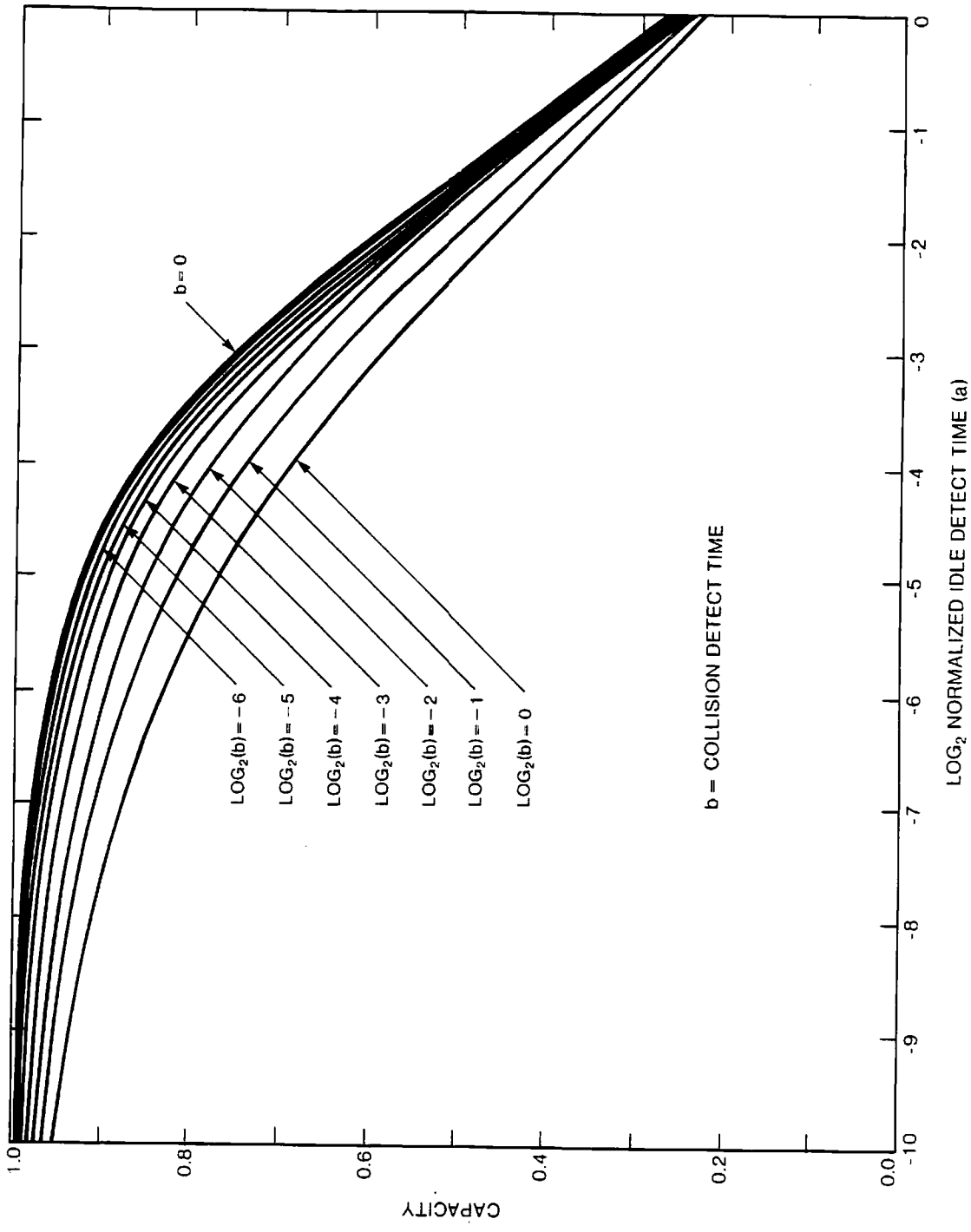


Figure 2.1: Sensitivity of Capacity to Detect Times
in a Locally Optimal CSMA Loss System

stant load per slot and thus cannot be locally optimal.[1] Non-persistent CSMA is locally optimal, but offers poor delay characteristics. In Chapter 6, we will present a new CSMA protocol that satisfies our local optimality conditions and also offers respectable delay characteristics.

### 2.5: TDMA

TDMA is a simple static scheduling procedure that periodically assigns each station exclusive access to the channel in a round-robin fashion, even if it is not ready. It is usually assumed that messages are sent to a common destination, so that all slot lengths are exactly equal to a message transmission time. Thus, since $a=b=1$, we see that $k^*=1$ from Eq. (2.3) (and thus that TDMA is locally optimal) when $\min_i \{p_i\} > \frac{1}{2}$, which is the case in heavy traffic. However, as we shall see in the next chapter, TDMA cannot provide channel access with finite delays at any value of channel utilization with an *infinite* number of stations.

### 2.6: MSAP (and BRAM)

Like TDMA, we wish to guarantee that no collisions will occur by periodically assigning to each station exclusive use of the channel. However, we now assume that each station can monitor channel activity through carrier sensing, so that idle slots can be cut short to improve the efficiency of the protocol. Although it is usually assumed that every station may transmit to every other station, we shall also consider the common destination model.

We note that in MSAP, a station is permitted to transmit *all* messages in its buffer whenever it is given access to the channel, while in BRAM a station cannot transmit more than one message per round-robin cycle. Recall that in our equations, the values of $a$ and $b$ were normalized in such a way that the transmission time for a successful message was unity. Thus, in MSAP, $a$ and $b$ must be re-normalized with respect to the average transmission time for a *train* of messages; in BRAM the normalization continues to be with respect to the transmission time for a *single* message. Thus, for a *fixed* propagation time, the normalized values of $a$ and $b$ will be *smaller* with MSAP than with BRAM. Thus MSAP can continue to be locally optimal when the normalized propagation time grows too large for BRAM to be locally optimal.

It is known that the efficiencies of both MSAP and BRAM decline as the population size increases, and that this sensitivity to $M$ increases as the propagation time increases and as the average load on the channel decreases. However, we see from Eqs. (2.3) and (2.7) that they are locally optimal as long as each of the $\{p_k\}$ is greater than the solution of

$$\frac{p}{1-p} = 1 - K\cdot(1-p), \tag{2.9}$$

where $K = \frac{b'}{b'+a'}$ in the arbitrary destination model and $K = 1-\frac{a}{b}$ in the common destination model. Since Eq. (2.9) is quadratic in $p$, it can easily be solved to show that

---

[1] When $b=1$, $p$-persistent CSMA with $p=1/(1+a)$ is locally optimal. However, when $b \neq 1$, this single parameter does not have enough degrees of freedom to achieve local optimality.

$$\min_{1 \leq i \leq M} p_i > \frac{1 - K - \sqrt{1 - K}}{-K}$$

is a sufficient condition for MSAP and BRAM to be locally optimal. In the arbitrary destination model, this condition becomes

$$\frac{-a' + \sqrt{a'^2 + a'b'}}{b'}, \tag{2.10}$$

which reduces to $\frac{1}{2}$ in the limit as $b' \rightarrow 0$ when the total cost of an idle slot and a collision become equal. In the common destination model, the condition becomes

$$\frac{-a + \sqrt{ab}}{b - a}, \tag{2.11}$$

which again reduces to $\frac{1}{2}$ in the limit as $b \rightarrow a$ when the total cost of an idle slot and a collision become equal. (Recall that the total length of a collision is assumed to be $b'+a'$ in the arbitrary destination model, and merely $b$ in the central destination model.) We shall see in Chapter 4, that this policy is *globally* optimal only when $p \geq \frac{1}{\sqrt{2}} \approx .7071$.

As a numerical example, consider the arbitrary destination model with $a = .01$, $b = 1$, and unbuffered stations (so that MSAP and BRAM are identical). This example is commonly used with CSMA protocols (see [Klei75c] and Chapter 6). In this example, the smallest $p$ for which MSAP and BRAM are locally optimal is $.1\sqrt{1.01} - .01 \approx .0905$, giving a minimum throughput at local optimality of $p/(a+p) \approx .90$ — slightly *higher* than the capacity of the best CSMA protocols.

### 2.7: The URN Scheme

We assume that there is a finite number, $M$, of homogeneous unbuffered stations. We also assume exact knowledge of $m$, the *number* of ready stations, and no information about their identities. Under these assumptions, $p_i = \frac{m}{M}$ holds for any *individual* station, but the probabilities are not independent. If we were to make the approximation that these probabilities are independent, then we see from Eq. (2.3) that $k^*$ would be the largest solution to

$$\sum_{i=1}^{k-1} \frac{\frac{m}{M}}{1 - \frac{m}{M}} \leq 1.$$

Solving for $k$, we obtain

$$k - 1 \leq \frac{M}{m}\left(1 - \frac{m}{M}\right)$$

giving $k^* = \left\lfloor \frac{M}{m} \right\rfloor$, which is the exact result [Klei78a].

The URN scheme proceeds by conducting a series of lotteries that enable some set of $k^*$ stations to transmit in each slot. While this basic URN rule is locally optimal in the *initial* state (*i.e.*, when there is no other information available besides $m$), it is completely memoryless, and thus cannot continue to be locally optimal unless the messages are randomly redistributed among the stations between each slot.

For example, suppose at some time it became known that *every* station had one message to send, at which point the arrival process was stopped. If we were to remember the identities of the stations that have already transmitted their messages, the set $\{p_i\}$ would always be independent so that the local optimality results are exact. Clearly the locally optimal policy would be to select a different station at every slot for $M$ slots until every station had transmitted its message — giving perfect scheduling of the channel. However, using the memoryless URN rule, the expected number of slots required to transmit all the messages would be given by

$$\sum_{m=M}^{1} \frac{1}{p_m},$$

where

$$p_m = \frac{\binom{m}{1}\binom{M-m}{k^*-1}}{\binom{M}{k^*}}$$

and $k^* = \left\lfloor \frac{M}{m} \right\rfloor$. Using Stirling's approximation for $n! \approx \sqrt{2\pi n}\,(n/e)^n$, it can be shown that

$$p_m \approx e \cdot (1 - 1/m)^{m-1} \cdot (1 - m/M)^{M/m-1},$$

which is approximately $1/e$ when $1 << M/m$ and $1 << m$. Thus as $M$ grows large in this example, the memoryless URN policy would require almost $e$ times as many slots to empty the system as the locally optimal scheme with perfect memory (*i.e.*, round-robin).

### 2.8: Application to the Hidden Station Environment

While our *local* optimality condition cannot be used to find a *globally* optimal protocol for the hidden station environment, it can be used with some protocols (such as slotted ALOHA) to find optimal values of system parameters. In particular, since the slotted ALOHA protocol is memoryless, local optimality can be used to calculate the set of probabilistic transmission rights for busy stations, $\{p_i\}$, that allows maximum stable throughput to be achieved for a fixed (up to a scalar multiple) traffic matrix for the network.

Figure 2.2 shows the hearing graph of a simple four station network in the hidden station environment. We assume that each station generates messages at the same rate, and that the destination of each message is equally likely to be either of the two nearest neighbours of the originating station. We also assume that all transmissions are "omni-directional" so that, for example, $A$'s transmissions to $B$ interfere with $C$'s transmissions to $D$. By symmetry, it is clear that we need examine only a single source — destination pair, say the attempts by $A$ to

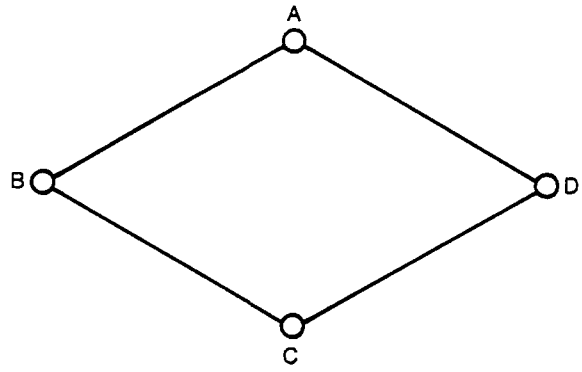


Figure 2.2: Hearing Graph of an Example Network

send to $B$.

Since we have already chosen to use the controlled slotted ALOHA protocol, and have specified the relative throughput by each station, the problem can be formulated as follows. We apply the local optimality conditions at the intended destination, namely $B$, giving $A$ (locally) lowest priority, and look for the largest $p$ such that it would still be locally optimal at $B$ not to completely disable $A$ from transmitting. Assuming constant size slots, $A$ will be granted permission according to the local optimality rule as long as

$$2 \cdot \frac{p}{1-p} \leqslant 1$$

holds, from which we obtain $p \leqslant 1/3$, giving $p^* = 1/3$. Thus, the "capacity" of this network for this symmetric nearest neighbour traffic matrix using controlled slotted ALOHA is

$$4 \cdot p^*(1-p^*)^2 = \frac{16}{27} \approx .5926$$

message-hops/slot.

Using controlled slotted ALOHA is anything but globally optimal for this particular network, however. Careful examination of the hearing graph shows that the true capacity is 2 message-hops/slot. This level of throughput is clearly attainable by simultaneously enabling any neighbouring pair of stations to transmit to their opposite neighbours (e.g., enabling both $A$ and $D$ to transmit to $B$ and $C$, respectively). No higher throughput could possibly be achieved because each message (including multi-destination "broadcast" messages) is heard by exactly two potential receivers. Since a station can receive at most 1 message per slot and no station can transmit and receive simultaneously, achieving a throughput of 3 message-hops/slot would require the network to contain 3 receivers and at least 2 transmitters — an obvious impossibility in a 4 station network.

## 2.9: An Evaluation of Local Optimality

Above, we have given three examples comparing the performance of the protocol we obtain by applying the local optimality conditions to a particular operating environment. In two cases, namely the introductory example in §2.2 and the hidden station example immediately above, the performance of the locally optimal protocol is much poorer than the globally optimal protocol. In the third case (in §2.7), local optimality (with memory) gave much better performance than the memoryless URN scheme. While we are encouraged that many known protocols appear to obey local optimality, the above performance comparisons clearly do not provide enough information to judge the merits of this heuristic.

In the sequel, we will return to our local optimality condition several times. In Chapter 3, we will present the locally optimal infinite population tree algorithm. We show that its capacity (and, indeed, its behaviour) is very close to the best known infinite population protocol for Poisson arrivals. Using the results of Chapter 4, where we examine Bernoulli arrival processes in detail, we see that our choice of the Bernoulli arrival probability in the example of §2.2, namely $p = \frac{1}{2} + \epsilon$, is really the *worst* case for local optimality in such a system. On the other hand, local optimality gives the exact globally optimal protocol for all $p \geq \frac{1}{\sqrt{2}} \approx .7071$. In Chapter 6, we will present the virtual time CSMA protocol. This protocol satisfies our local optimality conditions in a loss system, and we prove it to be the globally optimal protocol for Poisson arrivals under that assumption. Thus, we feel that, on the whole, local optimality is probably a reasonable heuristic. In addition, since the state of the system in real networks can change dynamically in the middle of the collision resolution process as new messages enter the system or because of the mobility of the stations, the value of any planning that local optimality fails to perform should not be overestimated.

# CHAPTER 3
## Synchronous Protocols without Reservations

### 3.1: The Inherent Difficulty of the Multiple Access Problem

In a normal queueing system, all the work collects in a queue from which the server selects customers. It is implicitly assumed that the server can identify any customers waiting in the queue. In addition, most queueing systems are assumed to be *work conserving, i.e.*, the server is busy granting useful service to some customer(s) whenever there are customers in the system. Variability in the inter-arrival and service times causes queueing delays which grow as the utilization of the system increases, but (in the absence of any overhead) the useful utilization of the system can be made arbitrarily close to unity if we are willing to tolerate a sufficiently-large (but finite) average delay.

When a distributed population of stations shares a channel, however, there is no observable queue of messages waiting to be transmitted since these distributed messages cannot "see" each other. Scheduling the use of the channel becomes difficult, since waiting messages must first be found before they can be transmitted. Thus part of the channel capacity will be lost as overhead either directly in the exchange of protocol information, or indirectly by an imperfect scheduling algorithm that wastes the channel in idle periods or collisions when there are waiting messages. Thus the price of having a distributed system will be increased delays for a given level of system utilization. In fact, as first shown by Pippenger [Pipp81a], there are conditions under which this overhead is so severe that no realizable protocol can be made to fully utilize the channel.

Let us therefore define the *capacity*, $C$, of a channel, given a particular number $M$ of stations, the arrival processes for all stations, and a multiple access protocol, to be the supremum over all channel utilizations such that *the average delay is finite.* This (new) definition is consistent with the definition of capacity for the single queue environment.

Care must be taken in defining capacity of systems in the limit of an *infinite* number of stations.. In this case, the evaluation of capacity requires that two limits ($M \rightarrow \infty$, and the supremum over all utilizations having finite delay) be taken simultaneously. Consequently, we require a more precise definition of capacity that clearly prevents any possible ambiguities.

Consider the problem of determining the capacity of a channel using round-robin TDMA for $M$ identical stations as $M \rightarrow \infty$. The capacity is unity for any *finite* $M$, since each of a finite number of stations has a finite queue (giving a finite average number of messages in the system) whenever the channel utilization is strictly less than unity. Thus, if we could first find capacity for $M$ stations and then let $M \rightarrow \infty$, the capacity with an infinite population would

appear to be unity. However, if we first let $M \to \infty$, it is clear that the average backlog of messages will be finite only if the average queue length at each station is zero. Since the expected channel utilization with TDMA would be zero if the expected number of waiting messages at each station was zero, TDMA appears to have zero capacity in the infinite population limit.

Maximum throughput is not the only performance statistic of interest for real systems. Mean delay and the probability of blocking are also important. There has been much recent interest in new measures of system performance that combine these measures into a single performance statistic. For example, *power* [Yosh77a, Gies78a, Klei78b, Klei79a] increases with throughput, and decreases with delay and blocking. The resolution of this apparent ambiguity in the capacity calculation should be done in a way that favours reasonable delay.

We therefore propose the following method for calculating capacity. Choose any constraint $\tau < \infty$ on the maximum mean delay $T(\rho)$ (expressed in units of a message transmission time). We assume that $T(\rho)$ is an non-decreasing function of the channel utilization $\rho$. Define $C_\tau$ to be the supremum over all $\rho$ such that $T(\rho) < \tau$. Then the capacity is simply

$$C \triangleq \sup_{\tau < \infty} C_\tau.$$

Returning to the previous example, we know [Mart70a] that for TDMA

$$T(\rho, M) = 1 + \frac{M}{2(1-\rho)},$$

which is an increasing function of both $\rho$ and $M$. It follows that for any $\tau < \infty$ and any $\rho$, we can find a large enough (but finite) $M$ such that

$$T(\rho, M) > \tau.$$

Indeed, this inequality is true for all $M$ exceeding

$$2(\tau - 1)(1 - \rho).$$

Thus $\lim_{M \to \infty} C_\tau \to 0$ for all $\tau < \infty$, and hence $C \triangleq \sup_{\tau < \infty} C_\tau = 0$, and the infinite population capacity of TDMA is clearly zero.

This apparent discontinuity in capacity as $M \to \infty$ is really not a discontinuity in a practical sense. In real systems, there is always some finite upper bound to the tolerable mean delay. Consequently, it is not $C$ but $C_\tau$, for some $\tau < \infty$, that is important. We have already shown that the mean delay grows *linearly* with $M$ when TDMA is used, so there will always be a *finite* value of $M$ beyond which TDMA becomes impractical.

### 3.2: Infinite Population Multiple Access Protocols

Results like the TDMA example in the previous section have stimulated considerable theoretical interest in infinite population multiple access protocols. Protocols suitable for an infinite population of stations are also of considerable practical interest for systems with a very large (but finite) number of stations. Their performance in the finite population case is at least as good as it was in the infinite population case. They are robust in the sense that each station need not be aware of the addresses of every other station in the system, or even of the exact number of stations in the system. Thus stations can enter or leave a working system, as would be required, for example, in a mobile or a dynamic environment — a key property for practical systems. Furthermore, when such protocols are used, the average message delay for a given channel utilization is insensitive to even large changes in the number of stations sharing the channel.

As we have already seen in the TDMA example, any protocol that operates in a manner that creates a positive average queue length at each station will be infeasible in the infinite population case. The average number of messages in the system will grow without bound as $M \to \infty$, and hence the mean delay must also grow to infinity by Little's result [Litt61a]. Consequently, it makes no sense to represent the message queue at each *station* explicitly when describing an infinite population multiple access protocol. Instead, the "queue" should be viewed as a finite set of "real" *messages* (*i.e.*, messages *actually* waiting to be transmitted) uniformly distributed over an infinite set of "potential" messages (*i.e.*, messages *that could be* waiting to be transmitted), and service becomes a probabilistic event. In many infinite population protocols, the elements of these sets are either points on the real time line, each possibly representing the arrival time for some message, or infinite sequences of binary digits, each possibly representing the address of some station having a message for transmission (these addresses may be pre-assigned deterministically or randomly constructed through, say, tossing coins).

At the start of each slot, the protocol *enables* (*i.e.*, grants permission to transmit to) a subset of the population: each station transmits an enabled message if it has one; all other stations remain silent for the duration of the slot. The choice of which subset to enable may depend on the outcome of previous attempts. Each attempt to offer service may result in a successful message transmission if exactly one "real" message was enabled, or merely provide some information about the distribution of "real" messages. The object of a protocol is to efficiently partition the set of potential messages into subsets each containing one real message.

The *slotted ALOHA* protocol [Robe72a, Abra73a] was one of the first infinite population protocols studied and implemented. It is also one of the simplest from both a conceptual and analytic viewpoint. This protocol allows messages to be transmitted without regard for the actions of other stations or the activity on the channel. Newly generated messages are transmitted in the first slot after their arrival; any message lost in a collision is repeated (*i.e.*, retransmitted) after a random delay. This protocol gathers no information from the channel history, so its performance does not degrade in the hidden station environment. In fact, it effectively *destroys* information by randomizing the retransmission of lost messages. In

exchange for simplicity and small delays under very light load, ALOHA offers poor channel utilization, with maximum throughput not exceeding $1/e \approx .37$ in the infinite population limit, and lacks stability (except in a loss system) because the retransmission of lost messages causes a positive feedback on the input rate that will eventually overload the system [Lam75a, Ferg75a]. Even with a finite population, where the stability issue can be solved, ALOHA suffers from large, unpredictable message delays with high variance.

### 3.3: Tree Resolution Algorithms

Recently, a new class of stable "self-organizing" synchronous multiple access protocols has been discovered. These protocols employ scheduling algorithms that dynamically adapt to the channel history. Such dynamic scheduling algorithms are commonly called *tree algorithms* because their state transition diagrams may be drawn as a (possibly infinite) ternary decision tree.

The performance analysis of tree algorithms has proven to be a difficult task because of the complexity of the state space. For example, in order to calculate delays, each state must encode both the relevant information from the past history of channel activity and size of the current backlog of unexamined "potential" messages. Fortunately, the capacity calculation is much more manageable. No reasonable algorithm will ever enable so large a set of potential messages that the expected number of actual messages is infinite. It is a certainty that a collision will occur when such a large set is enabled — giving no new information about the distribution of messages at the cost of one slot. Thus, for the capacity calculation, we may assume that the current backlog of unexamined potential messages is so large (but still finite) that the algorithm is always free to enable as many potential messages as it wishes. The exact size of the backlog becomes unimportant to the operation of the protocol, so that for the capacity calculation it is sufficient for the states to encode only the relevant past history of channel activity.

Capetanakis [Cape78a] defined the algorithm shown in Figure 3.1. Stations can only join the collision resolution algorithm (*i.e.*, become active) at the start of what we shall call a *service epoch*. All messages generated during one service epoch are transmitted successfully at some time during the *following* epoch. Access to the channel within a service epoch is controlled by having the stations toss fair binary coins. Only those active stations that have generated a particular sequence of coin tosses are allowed to transmit in each slot.

The algorithm proceeds as follows. In the first slot of each service epoch, the protocol enables all active stations to transmit their message (*i.e.*, to transmit, each station must have trivially generated the same zero-length sequence of coin tosses). One can deduce from an idle slot or a success that *at most* one station generated that sequence, thus terminating some branch of the collision resolution tree. The service epoch ends when all branches of the tree have been terminated. Whenever there is a collision, however, stations learn that *more than one* station had a common sequence of coin tosses. Each of the stations involved in that collision must continue tossing to generate a unique sequence, thereby extending the collision resolution tree. Those stations tossing "1" retransmit immediately; the rest wait until the new subtree that is created by all the stations that tossed "1" terminates. Thus the active stations organize

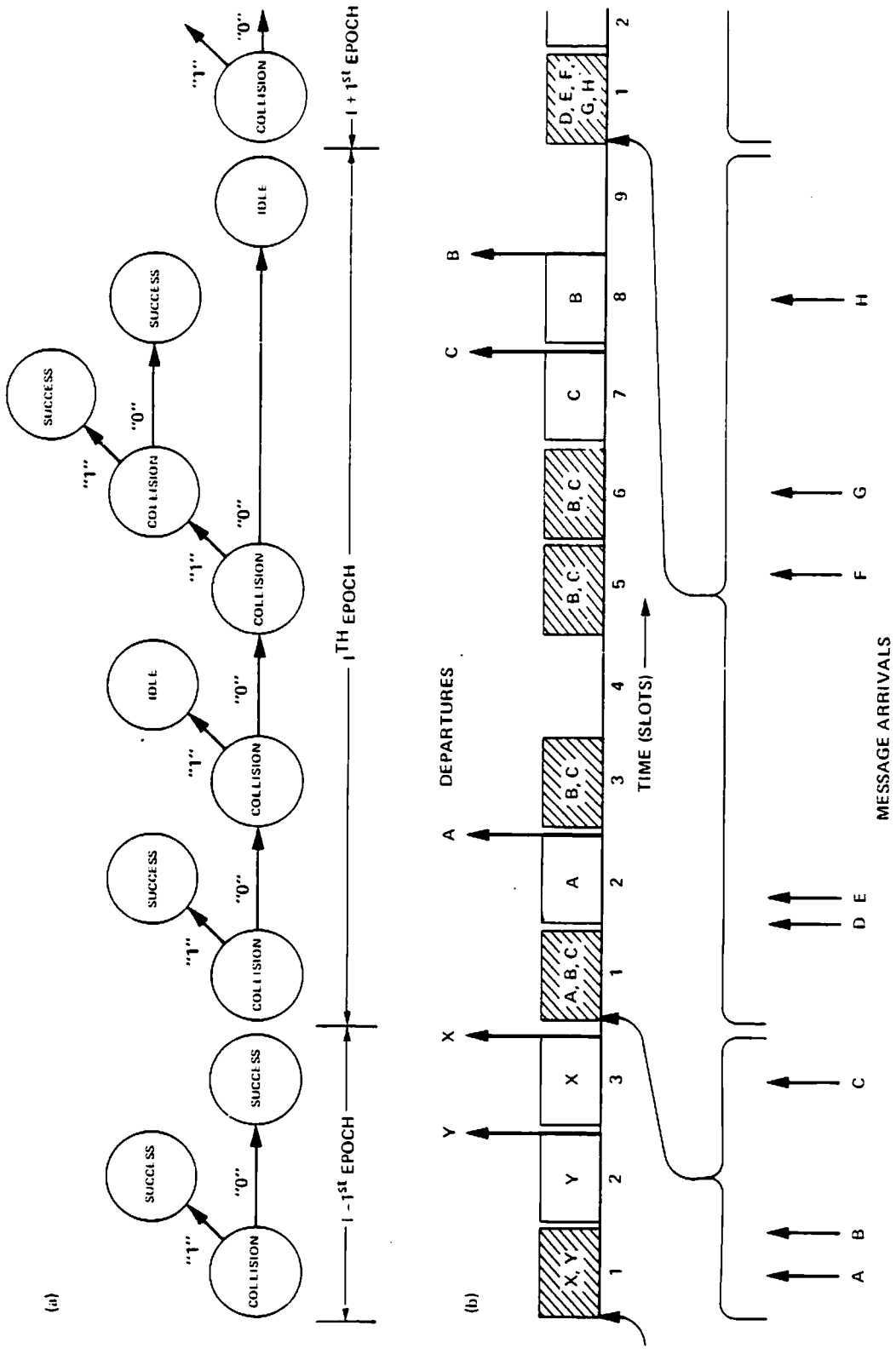


Figure 3.1: The Capetanakis Algorithm:
(a) A Sequence of Collision Resolution Trees (b) A Diagram Corresponding to the Channel Activity

themselves by tossing binary coins until each active station has generated a unique sequence of coin tosses (and thus succeeded in transmitting its message successfully).

In Capetanakis' protocol, the length (and efficiency) of a service epoch depends on the number of messages that are transmitted in the epoch. It is characteristic of this protocol for the average length of a service epoch to become very long under heavy load. The original tree algorithm [Cape78a] granted *all* stations having a message that arrived during the previous service epoch permission to transmit in the first slot of the next service epoch. The first few slots of such a service epoch are unlikely to contain successful transmissions if the expected number of arrivals during the previous service epoch, $\triangleq N$, is much larger than unity. Thus, it is clear that the original tree algorithm suffers from a strong positive feedback on $N$, which limits its capacity to only $\approx .346$ — less than the (unstable, but stabilizable) slotted ALOHA protocol.

Capetanakis also extended the basic tree algorithm to reduce this dependence and its attendant feedback. Beginning with $N$, a function of $\rho$ and the length of the previous busy period, the first step of the algorithm immediately splits the traffic into groups, each of which is to contain $\approx 1.1$ messages on the average, if possible. This significantly improved the efficiency of the basic Capetanakis algorithm from $\approx .346$ to $\approx .429$.

An improvement to the collision resolution strategy (due to Massey [Mass80a]) is to skip the certain *repeat* collision that would follow a given collision if the algorithm were to enable all stations that tossed "0" in the coin toss following the given collision after finding that *none* of the stations tossed "1" in this toss (e.g., slot 5 of the $1^{th}$ service epoch in Figure 3.1 (a)). A further coin toss is performed immediately, as if the repeat collision had occurred (after all, this collision is a certainty). This improvement increases the maximum stable throughput of the basic Capetanakis algorithm from $\approx .346$ to $\approx .375$ — a result also discovered independently by Tsybakov and Mikhailov [Tsyb80b] — and of the improved algorithm from $\approx .429$ to $\approx .462$. However, Massey [Mass80a] has shown that this improvement leads to a deadlock if there are *errors* in the feedback channel. If a channel error ever makes an idle slot appear to be a collision, the algorithm will continue splitting and skipping apparent collisions indefinitely. In this same report, Massey investigates the Capetanakis algorithm and its extensions in considerable depth, and gives a thorough analysis of their performance, stability and sensitivity to errors.

### 3.4: The Gallager-Tsybakov Algorithm

Gallager [Gall78a], Tsybakov [Tsyb79a] and Ruget [Berg80a] independently discovered an improved FCFS protocol. Access to the channel is controlled by a window based on the current age of messages. Figure 3.2 shows the operation of the algorithm by plotting the window currently being explored against real time. Messages are shown as lines of unit slope since they age at a rate of 1 second per second from their arrival time until they have been successfully transmitted. Each window is shown as a parallelogram "sweeping out" the range of ages of messages to be transmitted in that slot. All messages whose current age falls within the window are enabled in that slot.

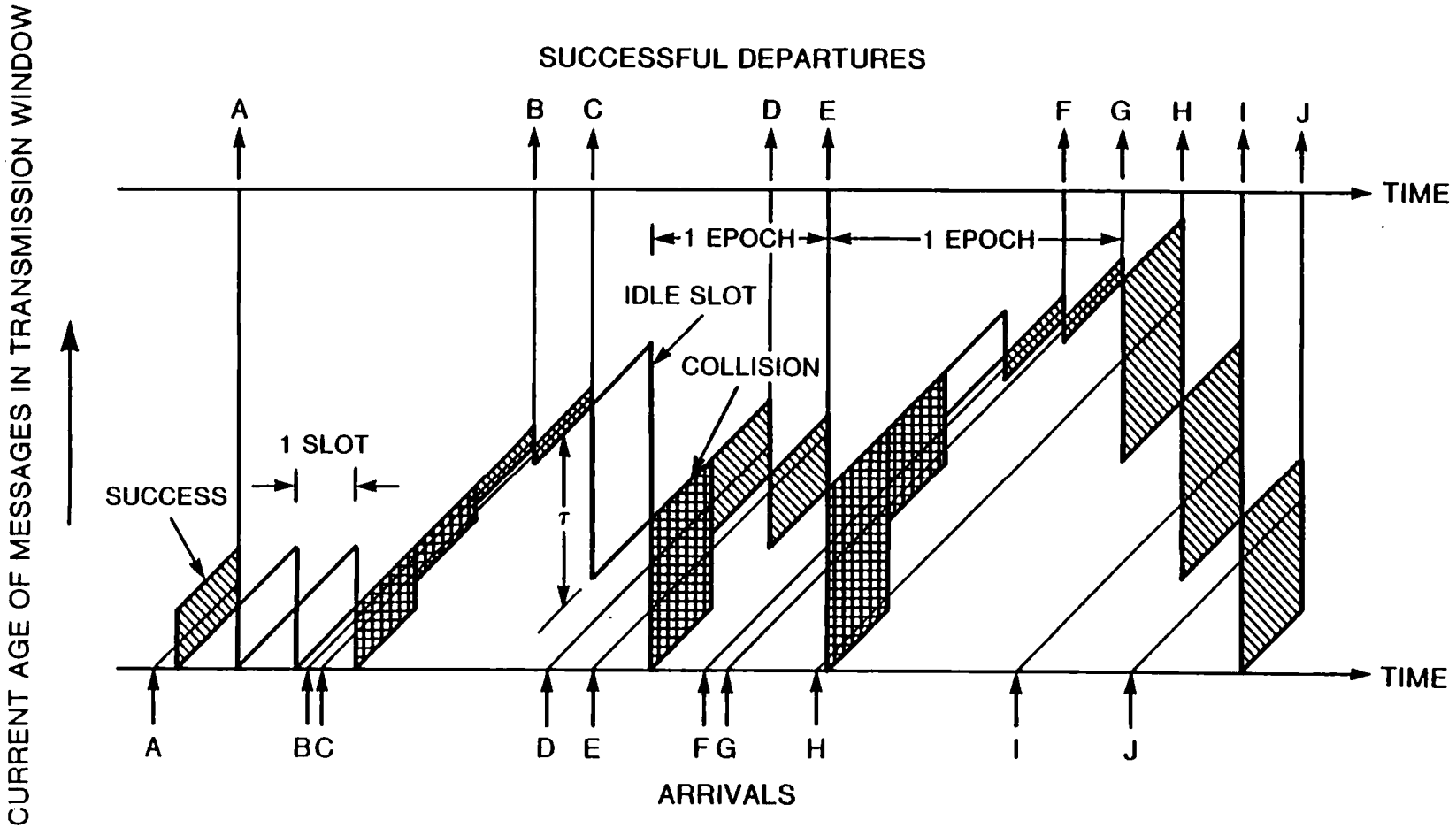


Figure 3.2: Operation of the Gallager-Tsybakov Algorithm

At the start of an epoch, all that is known is the maximum age in slots, $\alpha$, that any message could have and still be in the system. An initial window is chosen to enable all messages whose current age is between $\alpha$ and $\max(\alpha-\tau, 0)$ , where $\tau$ is a constant chosen to maximize the processing efficiency when $\alpha$ is very large. If the initial window is found to have at most one message, the epoch ends immediately and $\alpha$ is decreased to $\max(\alpha-\tau,0)+1$. Should this cause a collision, the window is split into two halves for continued processing. There are now three cases to consider. If enabling the first half causes a further *collision*, all knowledge about the second half is erased: it is effectively unexamined with a Poisson distribution of messages at the original intensity. Thus the first half is immediately split and the second half is ignored for the rest of the epoch. If enabling the first half causes an *idle* slot, the second half is immediately split — it must surely contain at least two messages. If the first half gives a *success*, the entire second half is enabled. An epoch that begins with a collision continues until enabling the (undiscarded) second half of some pair gives a success.

The performance of the Gallager-Tsybakov algorithm can easily be found from the following renewal theoretic argument. Let us partition the time axis into the series of processing epochs used by the algorithm, and distinguish between epochs according to the number of messages they contain initially. Since the arrival process is Poisson, we assume that the initial distribution of messages within a window is uniform, so that the probability that $j$ messages are found in the first half of an window containing $i$ messages is given by

$$p_{i,j} \triangleq \binom{i}{j} 2^{-i}.$$

Thus the expected number of slots needed to process an epoch with $k$ messages using the Gallager-Tsybakov algorithm, $\triangleq w_k$, is given by

$$w_k = p_{k,0}(1+w_k) + p_{k,k}(1+w_k) + p_{k,1}(2+w_{k-1}) + \sum_{i=2}^{k-1} p_{k,i}(1+w_i)$$

$$= \frac{1 + p_{k,1}(1+w_{k-1}) + \sum_{i=2}^{k-1} p_{k,i}\, w_i}{1 - p_{k,0} - p_{k,k}}. \tag{3.1}$$

with boundary conditions $w_0 = w_1 = 1$. Similarly, the expected number of messages actually transmitted in an epoch that initially contained $k$ messages is given by

$$n_k = p_{k,0}\, n_k + p_{k,1}(1+n_{k-1}) + \sum_{i=2}^{k} p_{k,i}\, n_i = \frac{p_{k,1}(1+n_{k-1}) + \sum_{i=2}^{k-1} p_{k,i}\, n_i}{1 - p_{k,0} - p_{k,k}} \tag{3.2}$$

with boundary conditions $n_i = i$ for $i \leqslant 2$. Thus the throughput is equal to the probability that a random observation over the time axis intercepts a success, *i.e.*,

$$\rho \triangleq \frac{E[w]}{E[n]} = \frac{\sum_{k=0}^{\infty} q_k\, n_k}{\sum_{k=0}^{\infty} q_k\, w_k} \tag{3.3}$$

where $q_k$ is the probability that an epoch initially contains $k$ messages. As long as the system

does not empty (*i.e.*, we assume that the system is heavily loaded), the number of messages in an initial window has a Poisson distribution with parameter $\lambda$ independent of the past history of activity on the channel. In this case

$$q_k = \frac{\lambda^k}{k!} e^{-\lambda}. \tag{3.4}$$

Equations (3.1) – (3.4) define a simple numerical procedure for finding the efficiency of the Gallager-Tsybakov algorithm for any $\lambda$. Computation reveals that the maximum (*i.e.*, the capacity of the algorithm) is $\approx .4871$, which is achieved at the optimum value when $\lambda \approx 1.266$. If the average arrival rate is less than .4871 messages per slot, the system must eventually empty and thus be stable.

### 3.5: Tuning the Collision Resolution Procedure

We can construct a slightly more efficient tree algorithm by selecting the enabled set at each step of the collision resolution procedure to satisfy the local optimality conditions defined in the previous chapter. We assume that at the start of each service epoch, an enabled set $E_0$ is selected so that the distribution of messages in $E_0$ is Poisson with parameter $\lambda_0$. A service epoch that begins with an idle slot or a success requires no further service. Thus, each multislot service epoch must begin with a collision. We will permit optimization of $\lambda_0$ to find maximum capacity, but require that all other enabled sets within a service epoch be selected to satisfy the local optimality conditions.

Consider an arbitrary state, say $i$, within a service epoch. We assume that being in state $i$ implies that we know from the channel history that some set $E_i$ must contain at least $m$ messages, $m \in \{1, 2\}$, and that the distribution of messages in $E_i$ is conditionally Poisson with some parameter $\lambda_i$. Recall that the locally optimal strategy for state $i$ is to maximize the expected channel utilization over all time that the protocol spends in state $i$. Since we have assumed that all slots are of constant length, it is sufficient to maximize the probability of success in each slot.

If the set $E_i$ is known to contain *one* or more messages, then enabling a superset of $E_i$ can only increase the probability of a collision and clearly cannot be locally optimal. The probability of success when a fraction $p$ of $E_i$ is enabled is given by

$$\sum_{k=1}^{\infty} \binom{k}{1} p(1-p)^{k-1} \frac{\frac{\lambda_i^k}{k!} e^{-\lambda_i}}{1 - e^{-\lambda_i}} = \frac{\lambda_i p\, e^{-\lambda_i p}}{1 - e^{-\lambda_i}}. \tag{3.5}$$

Since it is well known that the maximum of the function $xe^{-x}$ occurs for $x = 1$, it is clear that the maximum (w.r.t. $p$) of Eq. (3.5) occurs for $p = \min\{1, 1/\lambda_i\}$. Thus, since we expect $\lambda_i$ to be less than unity in the region of interest, enabling *all* of $E_i$ will be locally optimal.

If the set $E_i$ is known to contain *two* or more messages, then the probability of success when a fraction $p$ of $E_i$ is enabled is given by

$$\sum_{k=2}^{\infty}\binom{k}{1}p(1-p)^{k-1}\frac{\frac{\lambda_i^k}{k!}e^{-\lambda_i}}{1-e^{-\lambda_i}-\lambda_i e^{-\lambda_i}}=\frac{\lambda_i p\,[e^{-\lambda_i p}-e^{-\lambda_i}]}{1-e^{-\lambda_i}-\lambda_i e^{-\lambda_i}}. \tag{3.6}$$

Differentiating Eq. (3.6) with respect to $p$, we find that

$$\lambda_i p = 1-e^{-\lambda_i(1-p)}$$

must hold at optimality. While there is no closed form solution for the (locally) optimal $p$, an iterative numerical solution corresponding to any $\lambda_i$ can readily be found. For example, Newton iteration could be used with

$$p_N=\frac{1}{\lambda_i}\left(\frac{1+(\lambda_i p_{N-1})\,e^{-\lambda_i(1-p_{N-1})}}{1+e^{-\lambda_i p_{N-1}}}\right),$$

giving quadratic convergence.

Given the locally optimal strategy for selecting the next enabled set from $E_i$ as a function of $\lambda_i$, it remains to define a numerical procedure for finding the efficiency of the algorithm for a particular choice of $\lambda_0$. Since we have now permitted the splitting procedure to depend on the measure of the set $E_i$ (*i.e.*, $\lambda_i$), it is no longer possible to derive a set of difference equations of the form of Eqs. (3.1) and (3.2) that are based solely on the number of messages in $E_i$.

Any multiple access protocol that uses ternary feedback (*i.e.*, idle, success, or collision) to choose its enabled sets can be described by a ternary decision tree of infinite depth, with each node in the tree representing a state of knowledge for the protocol. Thus, to numerically calculate the performance of a specific protocol for a specific choice of $\lambda_0$, we may solve the difference equations for all nodes in the tree up to some finite depth to obtain the probability of occurrence of each state and the probability of success in that state. We may account for the infinite "tail" of each path down the tree by approximating the remaining states by a geometric sum.

To make the programming of this technique manageable, the tree may be represented by a ternary "heap" stored in a one-dimensional array. The root node is in location 0; the three sons of node $i$ are in locations $3i+1$, $3i+2$ and $3i+3$. By convention, if $i \bmod 3 = 2$, then the set $E_i$ is known to contain one or more messages, otherwise $E_i$ is known to contain two or more messages.

Let the protocol choose $E'_i \subseteq E_i$ as the enabled set at node $i$. If $E_i$ had been known to contain one or more messages, then node $3i+1$ is unused, node $3i+2$ contains $E_i - E'_i$ (corresponding to the case of $E'_i$ being empty), and node $3i+3$ contains $E'_i$ (corresponding to two or more messages in $E'_i$). If $E_i$ had been known to contain two or more messages, then

node $3i+1$ contains $E_i - E'_i$ ($E'_i$ is empty), node $3i+2$ contains $E_i - E'_i$ ($E'_i$ contained exactly one message), and node $3i+3$ contains $E'_i$ ($E'_i$ contained two or more messages). This technique permits rapid, accurate numerical calculation of both the equilibrium probabilities for each state and the conditional probability of a successful transmission given that the system is in that state.

Using this technique for direct solution of the equilibrium probabilities, we have calculated the capacity of the locally optimal tree algorithm to be $\approx .48714$ when $\lambda_0 \approx 1.266$. We have also investigated several other splitting algorithms, including various combinations of local optimality, even splitting (*i.e.*, the Gallager-Tsybakov algorithm), and choosing a fraction equal to the reciprocal of the expected number of messages in the interval. The splitting algorithm that attained the highest capacity was the arithmetic average of local optimality and even splitting (the optimal weights in the average were .493 for local optimality, and .507 for even splitting), which achieved a maximum capacity of $\approx .48785$ with $\lambda_0 \approx 1.2768$. Humblet [Mose79a] has performed a similar numerical study, and was able to achieve the capacity of $\approx .48775$.

Mosely [Mose79a] used a different approach to optimizing the parameters of the splitting procedure. By treating the splitting procedure as a continuous function of the size of the interval, the technique of value iteration [Howa60a] can be applied to iteratively converge on the optimal splitting algorithm. It is believed that her optimal policy represents the best possible first-come first-served protocol.

Mosely's optimum policy is indistinguishable from the best policy that we found in our study, namely the arithmetic average of local optimality and even splitting. This optimal splitting algorithm is to enable the *entire* interval when it is known to contain one or more messages, or to enable a fraction $B(\lambda)$, given approximately by $B(\lambda) \approx .50109 \cdot \lambda - .03466 \cdot \lambda^2$, when the interval is known to contain two or more messages. While it is tantalizing to try to find some physical interpretation to explain this coincidence, no such interpretation has yet been found.

# CHAPTER 4
## Capacity Bounds for Infinite Population Protocols without Reservations

In this chapter, we focus on upper bounds on the capacity of arbitrary protocols, not capacity results for given protocols. We shall continue to examine systems having an infinite number of stations in which the arrival process is either Poisson or consists of a series of independent *arrival points*, each having probability $p$ of containing exactly one message and $1-p$ of containing no messages. As the protocol operates, we may be able to deduce information about the current state of an arrival point from the history of channel activity. Initially, we say that each arrival point is *unexamined*. We will say that an arrival point is *busy* if it is known to contain a message that has not yet been successfully transmitted, and *idle* if it is known not to contain a message or if its message has already been transmitted successfully. We note that if there are $\eta$ arrival points per slot then in the limit as $p \to 0$, $\eta \to \infty$ and preserving the product $\lambda \triangleq \eta p$, the arrivals form a continuous Poisson process with parameter $\lambda$.

### 4.1: Information Theoretic Bounds

Pippenger [Pipp81a] used an entropy argument to develop an upper bound, $\xi_p$, on the capacity of such systems. A message can only be transmitted successfully when the protocol enables a set of arrival points that contains exactly one busy point. Thus, to bound the efficiency of multiple access protocols, it is equivalent to consider the problem of partitioning a set of arrival points so that each element of the partition contains exactly one busy point. By upper bounding the probability that any particular partition could correctly separate the busy points in the set, a lower bound on the entropy of valid partitions can be constructed. Any multiple access protocol must construct a valid partition by enabling subsets of the arrival points according to a ternary decision tree. A protocol that attains a throughput of $\xi_p$ must take the "success" branch in the decision tree a fraction $\xi_p$ of the time. An upper bound on the entropy of the identity of the terminal node can be constructed as a function of $\xi_p$. Since each terminal node in the tree determines a unique partition of the set, these two entropy results may be combined to form an upper bound on $\xi_p$. In the Poisson limit as $p \to 0$, Pippenger's upper bound on capacity is $\approx .744$.

Pippenger [Pipp81a] also investigated the case where the protocol can correctly distinguish between the events that *exactly* $0, 1, \ldots, D-1$ stations or *at least* $D$ stations transmitted in a slot, thereby being able to gather more information about the distribution of busy points from each collision. As $D \to \infty$, Pippenger was able to show the existence of protocols with capacities arbitrarily close to unity.

This same general information theoretic approach was extended by Hajek [Haje81a] to give a tighter bound of $\approx .711$ in the Poisson case. Humblet [Humb80a] further tightened the Poisson bound to $\approx .704$, and gave a bound on the rate at which capacity approaches unity as $D$ increases.

In the next section, we give an even tighter bound, also described in [Moll80a], that relies on a very different and far simpler method of proof. This result gives both an *optimum protocol* and the *exact* capacity for all $p \gtrsim .568$, and further tightens Pippenger's upper bound for Poisson arrivals to $\approx .6731$. For small values of the Bernoulli probability, $p$, we also describe tighter upper bounds. At this date, the tightest upper bound on capacity for Poisson arrivals is $\approx .587$. However, it is commonly believed that the true capacity is near .5.

### 4.2: Genie-Aided Bounds

In general it is not obvious how an optimal (in the sense that it achieves maximum capacity) multiple access protocol should operate. However, it is possible to describe optimal protocols for systems where a helpful "genie" provides certain extra information at no cost to the protocol. No protocol that explicitly requires the genie's information to operate is feasible without the genie. Thus the performance of optimal *genie-aided* protocols will be an upper bound on the performance of optimal *unaided* protocols. As a trivial example, perfect utilization of the channel would be possible if we could convince the genie to publicly label each busy point. The key to this approach is the selection of some particular information that does not make the contention resolution problem "too easy" but still allows one to make meaningful statements about the performance of an optimal protocol.

#### 4.2.1: An Optimal Genie-Aided Protocol

Suppose the "genie" were to label, at no cost, two busy points (and possibly some idle points) from each collision. Since a collision implies only that at least two busy points were enabled, we can get no further information from a collision with genie labelling as long as the labelling is done in such a way that all unlabelled points remain effectively unexamined with probability $p$ of being busy.

Let us assume that the genie uses the following algorithm to label points from each collision. If no previously known busy points were enabled, the genie labels the *first two* previously unknown enabled busy points (thus possibly creating some known idle points). If one known busy point was enabled, the genie labels that busy point plus the *first* unlabelled (*i.e.*, previously unknown) busy point that was enabled (also possibly creating some known idle points), giving less new information. If at least two known busy points were enabled, the genie trivially labels any two known busy points, giving no new information. Simultaneously enabling several known busy points gives a certain collision and clearly cannot be optimal.

We note that it is convenient but unnecessary for the genie to examine the enabled set in a first-come first-served order. The genie may examine the arrival points arbitrarily if each point is labelled idle or busy as it is examined until two busy points are found. Since the Bernoulli trials were initially independent, all unlabelled arrival points remain effectively unexamined with probability $p$ of being busy even with perfect information about the labelled points. Thus, any genie-aided collision resolution algorithm faces only known busy points, known idle points and unexamined arrival points. It remains to find an optimal genie-aided collision resolution algorithm and to determine the capacity of a protocol that uses that algorithm.

We now show that without loss of generality the search for an optimal algorithm can be restricted to those algorithms that separately enable each known busy point. Let $A$ be any genie-aided protocol that sometimes enables both a single known busy point and some unexamined arrival points. Define a new protocol, $A'$, that simulates the behaviour of $A$ but makes the following modification. Whenever $A$ would enable both one known busy point and $k>0$ unexamined arrival points — giving a success with probability $(1-p)^k$ and a collision (from which the genie labels *one* new busy point) with probability $1-(1-p)^k$ — $A'$ enables either one known busy point (and no unexamined arrival points) or all remaining unexamined arrival points (and no known busy points) with probabilities $(1-p)^k$ and $1-(1-p)^k$, respectively. Should it choose the former, $A'$ resumes its simulation of $A$ as if a success had occurred. Should it choose the latter, there is certain to be a collision from which the genie labels *two* new busy points; $A'$ interrupts its simulation for one slot to transmit successfully one such point and thereafter resumes its simulation of $A$ as if a collision had occurred. There is thus *perfect* utilization of the channel over all slots for which $A'$ interrupts its simulation of $A$. In addition, the simulation is a faithful probabilistic replica of $A$: it achieves a successful message transmission with probability $(1-p)^k$ (maintaining the same throughput as $A$), and it either increases the number of known busy points by one with probability $1-(1-p)^k$, or decreases the number of known busy points by one with probability $(1-p)^k$. Thus $A'$ must have at least as high a throughput as $A$. It follows that no genie-aided protocol can have a higher capacity than the best genie-aided protocol that chooses each enabled set to be either a single known busy point or a set of unexamined arrival points. Without loss of efficiency, this may clearly be done FCFS.

For any such genie-aided protocol, a new period of activity begins whenever the protocol enables some (possibly random) number, $N$, of unexamined arrival points. Each idle period takes one slot to process no messages, each success takes one slot to process one message, and each collision takes three slots to successfully transmit the two genie-labelled messages. Over all periods where a particular value of $N$ is chosen, the conditional genie-aided throughput, $\tilde{\rho}_N$, is found from a renewal argument to be

$$\tilde{\rho}_N \triangleq \frac{S_N+2C_N}{I_N+S_N+3C_N} = \frac{2-2I_N-S_N}{3-2I_N-2S_N} = 1-\frac{1-S_N}{3-2I_N-2S_N} \tag{4.1}$$

where $I_N=(1-p)^N$, $S_N=Np(1-p)^{N-1}$ and $C_N=1-I_N-S_N$ are the probabilities that enabling $N$ Bernoulli arrival points gives an idle slot, a success or a collision, respectively. Since the unconditional throughput is a convex combination of $\{\tilde{\rho}_N\}$, it cannot exceed $\tilde{\rho}_{N^*}$, where $N^*$ achieves maximum conditional throughput.

We have thus established that an optimal genie-aided strategy is to enable the first known busy point if there is one, or to enable some fixed number $N^*$ of unexamined arrival points otherwise. This optimal strategy also transmits all messages in a first-come first-served order. It remains to determine $N^*$ for all $p$.

**Theorem 4.1:**

For any fixed Bernoulli probability $p$, the sequence $\{\tilde{\rho}_N\}$ is unimodal in $N$.

**Proof:**

Let $\tilde{\rho}_N \geqslant \tilde{\rho}_{N+1}$. Then

$$1 - \frac{1-S_N}{3-2I_N-2S_N} \geqslant 1 - \frac{1-S_{N+1}}{3-2I_{N+1}-2S_{N+1}}$$

or

$$S_N - 2S_N I_{N+1} - 2I_N \geqslant S_{N+1} - 2S_{N+1} I_N - 2I_{N+1}. \tag{4.2}$$

But $I_{N+1} = (1-p) I_N$, and $S_{N+1} = pI_N + (1-p) S_N$, so that Eq. (4.2) gives

$$S_N \geqslant I_N(3-2I_N)$$

or

$$\frac{Np}{1-p} \geqslant 3-2I_N. \tag{4.3}$$

Using Lemma 2.1, to show unimodality, it is sufficient to show that $\tilde{\rho}_N \geqslant \tilde{\rho}_{N+1}$ implies $\tilde{\rho}_{N+1} \geqslant \tilde{\rho}_{N+2}$. Hence, to show that $\tilde{\rho}_{N+1} \geqslant \tilde{\rho}_{N+2}$, it suffices to show

$$\frac{(N+1)p}{1-p} \geqslant 3-2I_{N+1} = 3-2I_N+2pI_N.$$

Because of Eq. (4.3), this inequality holds if

$$\frac{p}{1-p} \geqslant 2pI_N$$

or

$$\frac{1}{2} \geqslant (1-p)^{N+1}, \tag{4.4}$$

which is clearly true for $p \geqslant \frac{1}{2}$. We thus assume $p < \frac{1}{2}$. Since $I_N \leqslant 1$, it must be the case from Eq. (4.3) that $N \geqslant \frac{1}{p} - 1$. Thus to show Eq. (4.4), it suffices to show that

$$\frac{1}{2} \geqslant (1-p)\cdot(1-p)^{(1/p)-1}. \tag{4.5}$$

Let $\nu \triangleq 1/p > 2$. Since it is well known that $(1-1/\nu)^{\nu-1}$ decreases monotonically from 1 to $1/e$ as $\nu$ increases from 1 to $\infty$, Eq. (4.5) follows for all $\nu$ if it holds for the smallest value, namely $\nu = 2$, where it is clearly true.

■

**Theorem 4.2:**

For fixed $N$, there is a unique solution, $p_N$, to $\tilde{\rho}_N(p)=\tilde{\rho}_{N+1}(p)$ for $p$ in the range $0<p<1$. If $p<p_N$, then $\tilde{\rho}_N(p)<\tilde{\rho}_{N+1}(p)$; if $p>p_N$, then $\tilde{\rho}_N(p)>\tilde{\rho}_{N+1}(p)$.

**Proof:**

If $\tilde{\rho}_N(p)=\tilde{\rho}_{N+1}(p)$, then Eq. (4.3) must be true as an equality. Let $f(p) \triangleq \frac{NP}{1-p}$ and $g(p) \triangleq 3-2(1-p)^N$ be the left and right sides of Eq. (4.3), respectively. We now show that assuming the existence of two solutions, $f(p_1)=g(p_1)$ and $f(p_2)=g(p_2)$ for $0<p_1<p_2<1$, leads to a contradiction. Since $f''(p)=N/(1-p)^3$ and $g''(p)=-2N(N-1)(1-p)^{N-2}$, $f(p)$ is strictly convex while $g(p)$ is strictly concave for $0<p<1$. For any $p_0$ such that $0<p_0<p_1$, choose $\alpha$ to satisfy $p_1=\alpha p_0+(1-\alpha)p_2$. Then, by convexity,

$$f(p_1) < \alpha f(p_0)+(1-\alpha)f(p_2).$$

Similarly, by concavity,

$$g(p_1) > \alpha g(p_0)+(1-\alpha)g(p_2).$$

Thus

$$f(p_0)>g(p_0)$$

for all $p_0$. Since $f$ and $g$ are continuous and differentiable for $0\leqslant p<1$ and $f(p)\rightarrow 0$ and $g(p)\rightarrow 1$ in the limit as $p\rightarrow 0$, we have an obvious contradiction. There can thus be at most one solution, $p_N$. Such a solution must exist, however, since $f(p)\rightarrow\infty$ and $g(p)\rightarrow 3$ as $p\rightarrow 1$.

■

**Corollary 4.1:**

$N^*$, the optimal number of unexamined arrival points to enable simultaneously, is a non-increasing function of $p$.

**Proof:**

Choose any $p$ and determine $N^*(p)$. By Theorem 4.1, $\tilde{\rho}_{N^*}\geqslant\tilde{\rho}_{N^*+1}$. By Theorem 4.2, since equality can occur for only one value of $p$, $\tilde{\rho}_{N^*}>\tilde{\rho}_{N^*+1}$ must hold for all $p'>p$. But $\tilde{\rho}_N$ is unimodal in $N$, so $N^*(p')\leqslant N^*(p)$.

■

**Corollary 4.2:**

For $p > 0$, $N^*$ decreases in unit steps as $p$ increases.

**Proof:**

It is sufficient to show that $p_N \leqslant p_{N-1}$. We thus assume $p_N > p_{N-1}$ and consider $\hat{p} \in (p_{N-1}, p_N)$. Then by Theorem 4.2 we must have $\tilde{\rho}_{N+1} > \tilde{\rho}_N$ since $\hat{p} < p_N$, and $\tilde{\rho}_{N-1} \geqslant \tilde{\rho}_N$ since $\hat{p} > p_{N-1}$, which contradicts that $\tilde{\rho}$ is unimodal in $N$.

■

We have thus established that $\tilde{\rho}_N$ is an upper bound to capacity for all $p$ in the interval $[p_N, p_{N-1})$.

### 4.2.2: Calculating the Bounds

Before computing some specific upper and lower bounds on the capacity of optimal protocols, we wish to establish that capacity (of optimal protocols) must be a *non-decreasing* function of $p$. This is true because we can always simulate an arrival sequence with Bernoulli probability $p_1$ given an arrival sequence with $p_2 > p_1$. The next arrival point in this simulated arrival sequence is independently defined either to be the next arrival point from the real arrival sequence (with probability $p_1/p_2$) or to be empty (with probability $1 - p_1/p_2$). Thus the performance of any protocol for $p_1$ can be achieved given $p_2$ by applying the protocol to the simulated arrival sequence.

We note in particular that if $p_1 > 0$ then the *randomized binomial* strategy for $p_2$ that selects $k$ arrival points with probability $\binom{N}{k}(p_1/p_2)^k(1-p_1/p_2)^{N-k}$, $0 \leqslant k \leqslant N$, enables exactly $n$ busy points with probability

$$\sum_{k=n}^{N}\binom{N}{k}(p_1/p_2)^k(1-p_1/p_2)^{N-k}\binom{k}{n}p_2^n(1-p_2)^{k-n} = \binom{N}{n}p_1^n(1-p_1)^{N-n}.$$

This distribution is identical to the probability of enabling exactly $n$ busy points by the *fixed* strategy for $p_1$ that selects $N$ arrival points. Similarly in the Poisson case where $p_1 = 0$, the randomized Poisson strategy for $p_2$ that selects $k \geqslant 0$ arrival points with probability

$$\frac{(\lambda/p_2)^k}{k!}e^{-\lambda/p_2}$$

achieves the same distribution for the number of enabled busy points as the fixed strategy for Poisson arrivals that enables a set with parameter $\lambda$. It follows that the performance of the previously described optimal genie-aided protocol for $p_1$ can be attained given $p_2 > p_1$ by defining a randomized genie-aided protocol. However, since the throughput will be a convex combination of the throughputs of several fixed strategies, there can be no advantage in randomization.

We now consider a Bernoulli arrival process with $p > \frac{1}{\sqrt{2}} \approx .7071$. We note from Eq.(4.1) that $\tilde{\rho}_2 = \frac{2p}{1+2p^2} < p$ for all $p > \frac{1}{\sqrt{2}}$. But since $\tilde{\rho}_1 = p$ and $\tilde{\rho}_N$ is unimodal in $N$, no genie-aided protocol (and hence no unaided protocol) can achieve a throughput exceeding $p$ for any $p > \frac{1}{\sqrt{2}}$. However, enabling individual arrival points ($N=1$, *i.e.*, "TDMA") achieves a capacity of $p$, does not require the genie's help, and is thus *feasible* and optimal. We note that $\frac{1}{\sqrt{2}}$ must also be an upper bound for $p \leqslant \frac{1}{\sqrt{2}}$ since we have shown that the capacity is a non-decreasing function of $p$.

Similarly *pairwise* enabling (*i.e.*, $N$=2) is optimal for $p$ between $\frac{1}{\sqrt{2}}$ and the solution of

$$\tilde{\rho}_3 = \frac{3p-p^3}{1+6p^2-4p^3} = \frac{2p}{1+2p^2} = \tilde{\rho}_2,$$

namely $p \approx .568$, where $\tilde{\rho}_2 = \tilde{\rho}_3 \approx .6904$. We can clearly label two busy points from a collision if only two arrival points were enabled, so pairwise enabling is feasible without the genie's help, and thus this is an optimal protocol in the range $.568 \lesssim p \leqslant \frac{1}{\sqrt{2}}$. We may continue to numerically evaluate the boundary where $N \geqslant 3$, but the optimal protocols now do require the genie's information. In the Poisson limit where $p \to 0$, $N$ becomes infinite and thus the maximum genie-aided throughput is

$$\tilde{\rho}_\infty = \frac{2-(2+\lambda)e^{-\lambda}}{3-2(1+\lambda)e^{-\lambda}}$$

This maximum occurs at $\lambda \approx 2.89$, the solution of $3-\lambda = 2e^{-\lambda}$. Hence for the case of Poisson arrivals, the throughput cannot exceed $\tilde{\rho}_\infty \approx .6731$. This completes the calculation of an upper bound on capacity for all $p$.

We now find a lower bound on capacity in the region $.568 \gtrsim p > 0$, where optimal genie-aided protocols simultaneously enable more than two arrival points. Since we cannot label two busy points without the genie's help if a collision occurs when $N \geqslant 3$, the performance of an optimal genie-aided protocol need not necessarily be attainable without the genie's help. However, the performance of any feasible protocol does form a lower bound on capacity. We have also shown that capacity is a non-decreasing function of $p$, so the capacity of Mosely's algorithm for Poisson arrivals [Mose79a], $\approx .488$, can be used as a lower bound for all $p$. We thus obtain a lower bound by taking the maximum of the capacity of Mosely's algorithm and the capacities as a function of $p$ of TDMA and pairwise enabling (the $N$=1 and $N$=2 cases, respectively).

The feasible region can be expanded upwards slightly by considering contention among more than two Bernoulli trials. We consider three obvious FCFS collision resolution algorithms for $N$=3. For the first case, assume that when a collision occurs, individual arrival points from the collision are enabled until two messages are transmitted. (There can be no advantage in separately enabling the third arrival point if the first two points were busy because it is effectively unexamined.) Two extra slots resolve the collision if the first two arrival points are

busy. Otherwise three extra slots are required. Hence, for this first algorithm, the throughput is given by

$$\frac{3p-p^3}{1+2[p^3+p^2(1-p)]+3[2p^2(1-p)]} = \frac{3p-p^3}{1+8p^2-6p^3}. \tag{4.6}$$

In the second case, we immediately enable a *pair* of arrival points whenever a collision occurs. There will either be a success if exactly one of the first two arrival points is idle (and we are done after enabling the third arrival point, now known to be busy), or a further collision if both are busy. Here again, the collision is resolved after exactly two messages have been transmitted, giving a throughput of

$$\frac{3p-p^3}{1+3[p^3+p^2(1-p)]+2[2p^2(1-p)]} = \frac{3p-p^3}{1+7p^2-4p^3}. \tag{4.7}$$

Finally, we may first enable a single arrival point from a collision. If that arrival point is idle, the remaining two points (now known to be busy) are separately enabled. If it is busy, both remaining arrival points are enabled, possibly resulting in a further collision. In this case, collision resolution takes two extra slots if either of the last two arrival points is empty, three extra if the first point is empty, and four extra if all arrival points contained messages. All messages in the enabled set are always transmitted, and the throughput is

$$\frac{3p}{1+2[2p^2(1-p)]+3[p^2(1-p)]+4[p^3]} = \frac{3p}{1+7p^2-3p^3}. \tag{4.8}$$

As one can easily verify, Eq. (4.8) exceeds both Eq. (4.6) and Eq. (4.7) in the range $.206 \lesssim p \lesssim .430$ where Eq. (4.8) exceeds both the throughput of pairwise enabling and the lower bound obtained from Mosely's algorithm.

Figure 4.1 plots throughput $\rho$ against the probability $p$ of a message arriving at an arrival point. Pippenger's upper bound is shown. In addition the new upper bound presented above is shown delimiting the unattainable region. Mosely's FCFS algorithm, the above FCFS algorithm for $N=3$, pairwise enabling (the $N=2$ case described above), and TDMA on arrival points (the $N=1$ case described above) delimit the attainable region.

### 4.2.3: Extension to Non-Constant Slot Lengths

Let us now consider a more general model of synchronous protocols in which the ratios of the average length of an idle slot to a success ($\triangleq a$), and of the average length of a collision to a success ($\triangleq b$) are arbitrary. When $a=b=1$, this reduces to the case of the previous section.

Unlike the previous section, we must now require that each station attempts to transmit only *complete* messages. When both the idle- and collision-detect times are unity, it is obvious that this restriction does not reduce capacity — there is no reason to attempt to transmit anything that is not a complete message if every slot is long enough to have transmitted a complete message. However, when the ratio of idle detect time to collision detect time is far from unity, it is conceivable that some form of "reservation" strategy, in which some slots are required to

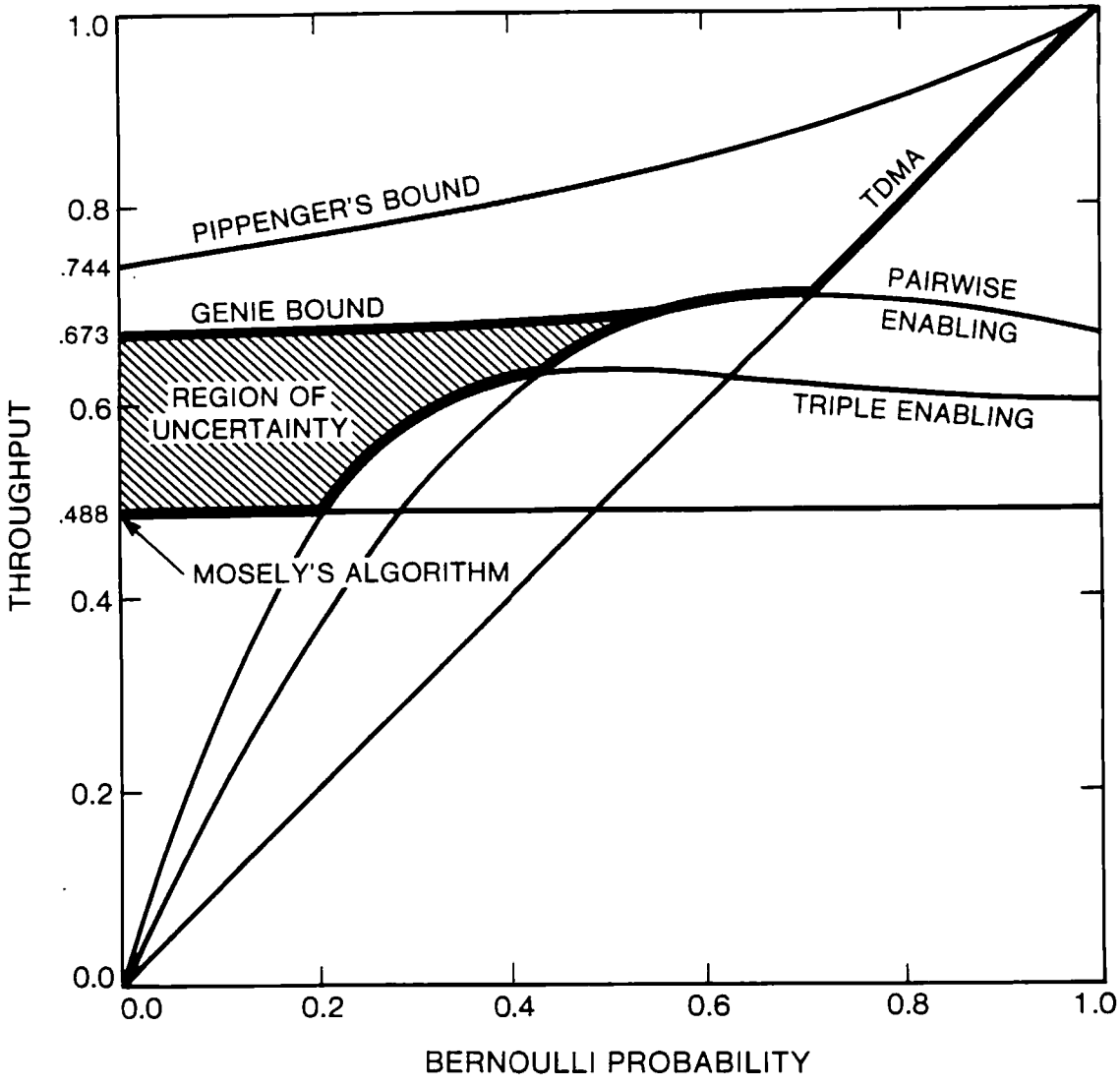


Figure 4.1: Throughput vs. Bernoulli Probability

carry only short reservation requests rather than complete messages, may be more efficient. (We will, in fact, describe a family of protocols in Chapter 7 that take advantage of a short propagation time by using short reservation bursts on the channel to improve scheduling. When $a/b << 1$, we show that the capacities of these protocols can exceed this "naive" extension of the genie-aided upper bound.)

Since Eq. (4.1) is based on a renewal argument, it can easily be extended to the case of non-constant slot lengths. In that case, the average channel utilization becomes

$$\tilde{\rho}_N \triangleq \frac{S_N + 2C_N}{aI_N + S_N + (b+2)C_N} = 1 - \frac{aI_N + bC_N}{aI_N + S_N + (b+2)C_N} \tag{4.9}$$

and a similar, albeit more tedious, argument can be used to prove its unimodality, as follows:

**Theorem 4.3:**

The sequence $\{\tilde{\rho}_N\}$ is unimodal in $N$.

**Proof:** Let $\tilde{\rho}_N \geq \tilde{\rho}_{N+1}$. Then

$$\frac{aI_{N+1} + bC_{N+1}}{aI_{N+1} + S_{N+1} + (b+2)C_{N+1}} \geq \frac{aI_N + bC_N}{aI_N + S_N + (b+2)C_N}$$

or

$$bS_N - (a+b)S_N I_{N+1} - 2aI_N \geq bS_{N+1} - (a+b)S_{N+1} I_N - 2aI_{N+1}. \tag{4.10}$$

But $I_{N+1} = (1-p)I_N$, and $S_{N+1} = pI_N + (1-p)S_N$, so that Eq. (4.10) gives

$$bS_N \geq I_N(2a + b - (a+b)I_N)$$

or

$$\frac{Nbp}{1-p} \geq 2a + b - (a+b)I_N. \tag{4.11}$$

Hence, to show that $\tilde{\rho}_{N+1} \geq \tilde{\rho}_{N+2}$, it suffices to show

$$\frac{(N+1)bp}{1-p} \geq 2a + b - (a+b)I_{N+1} = 2a + b \Lambda (a+b)I_N + (a+b)pI_N.$$

The inequality holds if

$$\frac{bp}{1-p} \geq p(a+b)I_N$$

or

$$\frac{b}{a+b} \geq (1-p)^{N+1}, \tag{4.12}$$

which is clearly true for $p \geq \frac{a}{a+b}$. Therefore assume that $p < \frac{a}{a+b}$. Since $I_N \leq 1$, it must be the case from Eq. (4.12) that $N \geq \frac{a}{b}\left(\frac{1}{p} - 1\right)$. Thus to show Eq. (4.12), it suffices to show that

$$\frac{b}{a+b} \geqslant (1-p)^{(a/b)((1/p)-1)}$$

or

$$\left(\frac{b}{a+b}\right)^{b/a} \geqslant (1-p)^{(1/p)-1}. \tag{4.13}$$

Let $\nu \triangleq 1/p > 1 + b/a$. Since it is well known that $(1-1/\nu)^{\nu-1}$ decreases monotonically from 1 to $1/e$ as $\nu$ increases from 1 to $\infty$, Eq. (4.13) follows for all $\nu$ if it holds for the smallest value (where it is clearly true).

■

Figure 4.2 shows a family of contours based on Eq. (4.9) that upper bound the capacity of "reservation-free" protocols for various combinations of idle detect time and collision detect time. We observe that if idle detect times exceed collision detect times, an unlikely event from physical constraints, the bound is almost completely insensitive to the idle detect time. We further note that as the idle detect time decreases significantly below the collision detect time, a common ocurrence in a radio environment, the sensitivity of the bound to the collision detect time is reduced. This is, of course, encouraging for system designers, since idle detection is much simpler to implement than collision detection.

### 4.2.4: Further Genie-Aided Bounds

In the previous section, we assumed that the genie identifies the exact location of enough messages to completely explain each collision. More recently, other authors [Cruz80a, Berg81a] have obtained tighter bounds on capacity using more complex, "less helpful" genies.

The information provided by these new genies is less specific. Consequently, more complicated genie-aided protocols must be considered. It has not been possible in these new systems to find an optimal genie-aided protocol explicitly and calculate its capacity. Instead, the proofs have relied on indirect arguments.

As a protocol operates, it gathers information about the set of arrival points. This information may be represented as a set of "states of knowledge". The action of a genie will be to provide additional information (besides the ternary feedback from enabling subsets to transmit) to limit the complexity of the state space. For example, the states for our original genie-aided system are simply the number of known busy points.

In the initial state, no messages have been transmitted and the protocol has no conditional information about the distribution of messages. In the final state, all messages have been successfully transmitted and there is again no conditional information. Each enabled set may result in a successful transmission with some probability. In addition, the state of knowledge will change. In general, those enabled sets that have a high probability of success will "use up" conditional information. Such sets cannot be enabled more frequently than the prerequisite conditional information becomes available to the protocol. Thus, bounds on capacity are

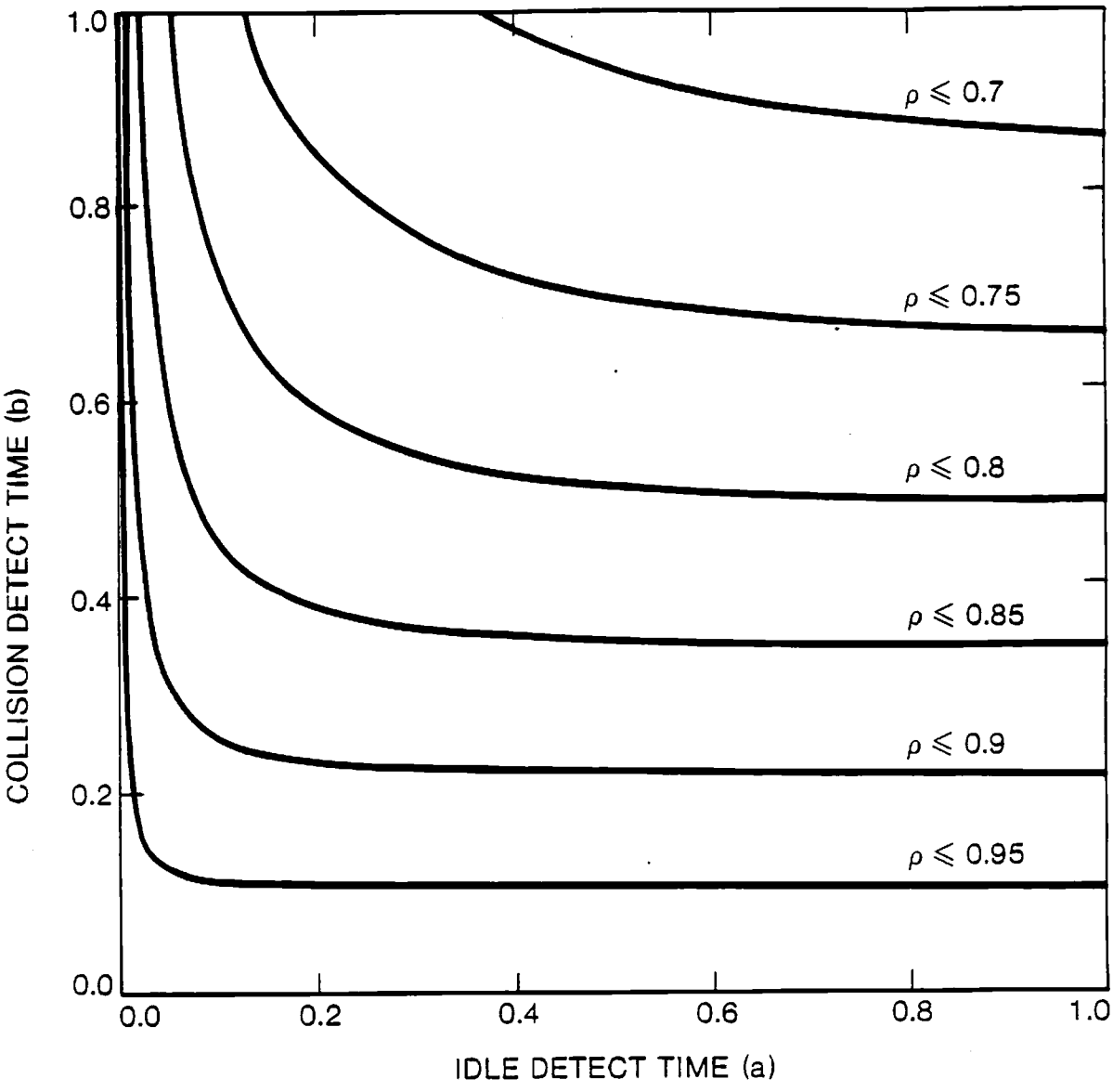


Figure 4.2: Capacity Bounds for Poisson Arrivals and Variable Slot Sizes

obtained by solving a form of max-flow min-cut problem.

Cruz and Hajek [Cruz80a] assumed that the genie would identify exactly one busy point from each set known to contain one or more messages, and enough messages from *overlapping* sets, each known to contain two or more messages, to explain away the overlap. With the aid of this genie, a collision resolution algorithm faces only known busy points, disjoint sets known to contain *two* or more messages, and unexamined arrival points. For this system, they were able to show that the capacity of any genie-aided protocol cannot exceed $\approx .6126$. This work was later extended by Berger and Mehravari [Berg81a] by relaxing the assumption that the genie always identifies one message from any set known to contain one or more messages. Thus, a collision resolution algorithm aided by Berger's genie must also handle sets known to contain *one* or more messages. Using a similar argument, they were able to bound the capacity to $\approx .587$.

### 4.3: The Tsybakov-Mikhailov Bound Extended to Bernoulli Arrivals

Very recently, Tsybakov and Mikhailov [Tsyb81a] introduced a new combinatorial method to prove that $\approx .5875$ is an upper bound on the capacity of infinite population multiple access protocols for Poisson arrivals. Since it is easy to extend their method to bound the capacity of a number of different systems that we will introduce later, we will now present their results in some detail.

Unlike the extensions of the "genie" argument described in the previous section [Cruz80a, Berg81a], this combinatorial method does not depend on the fact that Poisson arrivals can occur arbitrarily closely in time. Consequently, the method can be extended to the case of Bernoulli arrivals in a straightforward manner, even though Tsybakov and Mikhailov examined only Poisson arrivals. Below we present an extension of their result to the case of Bernoulli arrivals.

Recall that in the Bernoulli multiple access problem, *arrival points* occur in discrete time, each independently *busy* with probability $p$, $p > 0$, and *idle* with probability $1-p$. Without loss of generality, we may assume that exactly one arrival point occurs at each of the positive integers, $I^+$, and that there are no other arrival points. Define the set of busy points to be $X \triangleq \{x_1, x_2, \cdots\}$, where $x_i \in I^+$, $x_1 < x_2 < \cdots$, and the interarrival distribution is geometric with parameter $p$.

A protocol, $A$, is a synchronous algorithm for processing a finite interval, which we can assume to be $[1, M]$ without loss of generality by the independence of the arrival points. At each time $t = 1, 2, \cdots$, the protocol enables $E_t$, $E_t \subseteq [1, M]$. We define the function $\Theta(t) \triangleq (\theta_1(E_1), \ldots, \theta_t(E_t))$ to give the sequence of outcomes from enabling $E_1, \ldots, E_t$. Thus $\theta_t(E_t) = 0$ if enabling $E_t$ gives an idle slot (*i.e.*, $E_t \bigcap X = \varnothing$), $\theta_t(E_t) = 1$ if enabling $E_t$ give a success (*i.e.*, $E_t \bigcap X$ contains exactly one busy point), and $\theta_t(E_t) = 2$ if enabling $E_t$ gives a collision (*i.e.*, $E_t \bigcap X$ contains more than one busy point).

Whenever $\theta_t(E_t)=1$, we will delete the point $X \bigcap E_t$ from $X$. The algorithm will have completed processing $[1, M]$ when $X \bigcap [1, M]$ is known to be empty. We define $\tau = \tau(A, M)$ be the first time this is the case. Then $E[\tau(A, M)]$ is inversely proportional to the efficiency of $A$. An alternate characterization of $\tau$ can be made as follows. Each time $\theta_t(E_t) \in \{0, 1\}$, it becomes known to all stations that $E_t \bigcap X = \varnothing$ for all future time. Define

$$D_t \triangleq \bigcup_{\substack{s=1 \\ \theta_s(E_s)\neq 2}}^{t} E_s$$

to be the *accepted set* at time $t$. Thus $\tau$ may also be found from the relation

$$\tau = \min\{t : [1, M] \subset D_t\}.$$

Since the expected number of busy points in $[1, M]$ is clearly $Mp$, we define the efficiency of $A$ to be

$$\rho_A \triangleq \lim_{M\to\infty} \frac{Mp}{E[\tau(A, M)]}.$$

It follows that the capacity of the channel for the given $p$ must be

$$C_p \triangleq \sup_A \rho_A.$$

To complete the proof, it remains to find an upper bound on $\rho_A$ for all $p$.

**Lemma 4.1:**

For all $t \leqslant \tau$, given any protocol $A$, any $M$, and any sequence of past outcomes $\Theta(t-1)$, the probabilities that choosing any enabled set $E_t$ gives an idle slot or a success obey

$$Pr[\theta_t(E_t)=0 | \Theta(t-1)] \leqslant (1-p)^N,$$

$$Pr[\theta_t(E_t)=1 | \Theta(t-1)] \leqslant \left[1 - Pr[\theta_t(E_t)=0 | \Theta(t-1)]\right] \cdot \frac{Np(1-p)^{N-1}}{1-(1-p)^N},$$

where $N \triangleq \text{mes}(E_t - D_{t-1})$ is the measure of (*i.e.*, the number of arrival points in) that subset of the enabled set disjoint from the accepted set.

**Proof:**

Since $D_{t-1}$ is known not to contain any busy points, either including or excluding points from $D_{t-1}$ can have no effect on $\theta_t(E_t)$. Thus, without loss of generality, we can assume that $E_t \bigcap D_{t-1} = \varnothing$ for all $t$.

We now form the set $\{B_1, \ldots, B_R\}$ by enumerating all minimal intersections of any members of $\{E_1, \ldots, E_t\}$, *i.e.*,

1. $\bigcup_{i=1}^{R} B_i = \bigcup_{s=1}^{t} E_s$,

2. $\forall i \in \{1, \ldots, R\}$, $\forall s \in \{1, \ldots, t\}$, it must be the case that either $B_i \bigcap E_s = B_i$ or $B_i \bigcap E_s = \varnothing$,

3. $\forall i \neq j \in \{1, \ldots, R\}$, there exists an $s \in \{1, \ldots, t\}$ such that either both $B_i \bigcap E_s = B_i$ and $B_j \bigcap E_s = \varnothing$ are true, or both $B_i \bigcap E_s = \varnothing$ and $B_j \bigcap E_s = B_j$ are true.

Let $I = \{i_1, \ldots, i_K\}$ be the index set that specifies all members of $\{B_i\}$ that are contained in $E_t$, *i.e.*,

$$E_t = \bigcup_{j=1}^{K} B_{i_j}.$$

Define $\theta_0(B_j)$ to be 0, 1, or 2 if the outcome of enabling $B_j$ at $t = 0$ would have been idle, success, or collision, respectively. Then

$$\eta \triangleq \{\theta_0(B_i) : i \in \bar{I}\}$$

is an ordered set whose elements depend on the initial distribution of busy points in $X$. Since each element in $\eta$ can take on at most three values, there can be at most $3^{R-K}$ such sets, exactly one of which corresponds to any particular initial distribution.

Recall from the law of total probability that, for every protocol $A$ and any outcome $y \in \{0, 1, 2\}$, it must be the case that

$$Pr[\theta_t(E_t) = y | \Theta(t-1)] = \sum_k Pr[\theta_t(E_t) = y | \eta = \eta_k, \Theta(t-1)] \cdot Pr[\eta = \eta_k | \Theta(t-1)].$$

Thus, to prove the lemma, it is sufficient to prove that both

$$Pr[\theta_t(E_t) = 0 | \eta = \eta_k, \Theta(t-1)] \leqslant (1-p)^N$$

and

$$Pr[\theta_t(E_t) = 1 | \eta = \eta_k k\ \Theta(t-1)] \leqslant \left[1 - Pr[\theta_t(E_t) = 0 | \eta = \eta_k, \Theta(t-1)]\right] \cdot \frac{Np(1-p)^{N-1}}{1-(1-p)^N}$$

hold whenever $Pr[\eta = \eta_k | \Theta(t-1)] > 0$, where $N \triangleq \text{mes}(E_t - D_{t-1})$.

Define

$$\sigma_s \triangleq \sum_{i \in \bar{I},\, B_i \subset E_s} \theta_0(B_i)$$

to be the number of messages known to be in $E_s$, $s = 1, \ldots, t-1$, by examining $\{B_i\}$ given the event that $\eta = \eta_k$. Given $\sigma_s$, $s = 1, \ldots, t-1$, we can separate the previously enabled sets $E_s$ into five classes, $W_0, \ldots, W_4$, as follows:

$$W_0 = \{s : \theta_s(E_s) = 0\}$$

$$W_1 = \{s : \theta_s(E_s) = 1\}$$

$$W_2 = \{s : \theta_s(E_s) = 2, \sigma_s \geq 2\}$$

$$W_3 = \{s : \theta_s(E_s) = 2, \sigma_s = 1\}$$

$$W_4 = \{s : \theta_s(E_s) = 2, \sigma_s = 0\}$$

Since we have assumed that $E_t \bigcap D_{t-1} = \varnothing$ for all $t$, all busy points from $E_s$ that were in $X$ at $t=0$ must still be in $E_s$ at $s$. Thus, the event $\eta = \eta_k$ implies $\theta_s(E_s) = \sigma_s$ for all $s \in W_0 \bigcup W_1 \bigcup W_2$. However, $\{\sigma_s\}$ depends only on arrival points *disjoint* from $E_t$. The arrival points were initially *independent* Bernoulli trials, so the distribution of busy points on disjoint sets must remain independent. It follows that the events $\{\eta = \eta_k, \Theta(t-1)\}$ and $\{\eta = \eta_k, [\theta_s(E_s), s \in W_3 \bigcup W_4]\}$ are equivalent. To complete the proof, it remains to show that the desired result holds in each of the following three cases.

1. $W_4 \neq \varnothing$:
   It must be the case that $\theta_t(E_t) = 2$, since there exists a previously enabled set, $E_s$ say, with $\theta_s(E_s) = 2$ and $\sigma_s = 0$, *i.e.*, $E_s$ contains no busy points from $X$ not also in $E_t$.

2. $W_3 \bigcup W_4 = \varnothing$:
   No previous step gives any extra information about $E_t$. Thus

$$Pr[\theta_t(E_t) = 0 | \eta = \eta_k, \Theta(t-1)] = Pr[\theta_t(E_t) = 0] = (1-p)^N$$

   and

$$Pr[\theta_t(E_t) = 1 | \eta = \eta_k, \Theta(t-1)] = Pr[\theta_t(E_t) = 1] = Np(1-p)^{N-1},$$

   where $N \triangleq \text{mes}(E_t - D_{t-1})$.

3. $W_3 \neq \varnothing$, $W_4 = \varnothing$:
   Clearly $\theta_t(E_t) \neq 0$, since $E_t$ is known to contain at least one busy point from some $E_s$ with $\theta_s(E_s) = 2$ and $\sigma_s = 1$. Define

$$\hat{B} \triangleq E_t \bigcap \left( \bigcap_{s \in W_3} E_s \right).$$

   Then it is clear that $\theta_t(E_t) = 1$ can occur *only* if the set $\hat{B}$ contains *exactly one* busy point from $X$ and the set $E_t - \hat{B}$ contains *no* busy points from $X$. Let $\text{mes}(\hat{B}) = n$. Since

$$Pr[\theta_t(E_t) = 1, \eta = \eta_k, \Theta(t-1)] = Pr[\eta = \eta_k] \cdot np(1-p)^{n-1} \cdot (1-p)^{N-n}$$

   and

$$Pr[\eta = \eta_k, \Theta(t-1)] \geq Pr[\eta = \eta_k, \Theta(t-1), \theta_t(\hat{B}) \neq 0]$$

$$= Pr[\eta = \eta_k] \cdot (1 - (1-p)^n),$$

it follows from the law of conditional probability that

$$Pr[\theta_t(E_t) = 1 | \eta = \eta_k, \Theta(t-1)] = \frac{Pr[\theta_t(E_t) = 1, \eta = \eta_k, \Theta(t-1)]}{Pr[\eta = \eta_k, \Theta(t-1)]}$$

$$\leqslant \frac{np(1-p)^{N-1}}{1 - (1-p)^n}$$

$$\leqslant \frac{Np(1-p)^{N-1}}{1 - (1-p)^N},$$

where the last inequality follows from the fact that $\frac{n}{1-(1-p)^n}$ is a non-decreasing function in $n$.

■

Lemma 4.1 has established an extremely useful result. Intuitively, this result states that the conditional information obtained from the history of past attempts by the protocol can only *increase* the density of busy points. The following theorem applies this result to produce the desired upper bound.

**Theorem 4.4:**

Define $h(q)$ as

$$h(q) \triangleq \sup_{N>0} \left[ \frac{(N+q) \cdot Np(1-p)^{N-1}}{1 - (1-p)^N} + N(1-p)^N \cdot \max\left\{0, 1 - \frac{(N+q)p(1-p)^N}{1-(1-p)^N}\right\}\right].$$

Then for any given $p$,

$$C_p \leqslant \min_q \frac{h(q)}{1/p + q}.$$

**Proof:**

When $A$ completes the processing of $[1, M]$ at $\tau$, all arrival points from $[1, M]$ (and possibly some additional points) must be in the accepted set $D_\tau$. It must also be the case that all busy points in $X \bigcap [1, M]$ must have been deleted. Since the same strategy need not maximize the marginal progress towards each of these two goals at every step, we shall combine both requirements in the following objective function. Define

$$\gamma_t \triangleq \text{mes}\, D_t + q \cdot s_t$$

where $s_t$ gives the number of successes up to $t$ and $q$ is any real number. The parameter $q$ serves to weight the relative importance of accepting points into $D_t$ and of successfully transmitting messages. We note that if $M < \infty$, then $Pr[\tau < \infty] = 1$ for at least a few protocols. Since $E[s_\tau] = Mp$, it follows that

$$E[\gamma_\tau] \geqslant M(1+qp). \tag{4.14}$$

Having thus lower bounded $E[\gamma_\tau]$ for any given $q$, we continue by finding an upper bound on the incremental increase in the objective, $\gamma_t - \gamma_{t-1}$, at each step.

It is clear that if $E_t$ is enabled at $t$, mes $(E_t - D_t) = N$, then $\gamma_t - \gamma_{t-1} = N$ if $\theta_t(E_t) = 0$, $\gamma_t - \gamma_{t-1} = N+q$ if $\theta_t(E_t) = 1$, and $\gamma_t - \gamma_{t-1} = 0$ if $\theta_t(E_t) = 2$. Let $\gamma(t) \triangleq (\gamma_1, \ldots, \gamma_t)$. Then for every $t \leqslant \tau$,

$$E[\gamma_t - \gamma_{t-1} | \gamma(t-1)] = \sum_{\Theta(t-1)} E[\gamma_t - \gamma_{t-1} | \gamma(t-1), \Theta(t-1)]\, Pr[\Theta(t-1) | \gamma(t-1)].$$

Since $\Theta(t-1)$ and $E_1, \ldots, E_{t-1}$ together determine $\gamma(t-1)$, $E[\gamma_t - \gamma_{t-1} | \gamma(t-1)]$ may be rewritten as

$$\sum_{\Theta(t-1)} \Big[ N \cdot Pr[\theta_t(E_t) = 0 | \Theta(t-1)] + (N+q) \cdot Pr[\theta_t(E_t) = 1 | \Theta(t-1)] \Big] Pr[\Theta(t-1) | \gamma(t-1)].$$

If we can now upper bound the bracketed expression for all $\Theta(t-1)$, we will have upper bounded the expected incremental increase at any step. Thus, applying Lemma 4.1, we obtain

$$\begin{aligned}
&E[\gamma_t - \gamma_{t-1} | \gamma(t-1)] \\
&\leqslant N \cdot Pr[\theta_t(E_t) = 0 | \Theta(t-1)] + \frac{(N+q)\, Np(1-p)^{N-1}}{1-(1-p)^N} \cdot \Big[ 1 - Pr[\theta_t(E_t) = 0 | \Theta(t-1)] \Big] \\
&= \frac{(N+q)\, Np(1-p)^{N-1}}{1-(1-p)^N} + Pr[\theta_t(E_t) = 0 | \Theta(t-1)] \cdot \left[ N - \frac{(N+q)\, Np(1-p)^{N-1}}{1-(1-p)^N} \right] \\
&\leqslant H(N, q),
\end{aligned}$$

where

$$H(N, q) \triangleq \frac{(N+q) \cdot Np(1-p)^{N-1}}{1-(1-p)^N} + N(1-p)^N \cdot \max\left\{ 0,\ 1 - \frac{(N+q)p(1-p)^N}{1-(1-p)^N} \right\}.$$

It follows that for every $t \leqslant \tau$,

$$E[\gamma_t - \gamma_{t-1} | \gamma(t-1)] \leqslant h(q) \triangleq \sup_{N>0} H(N, q). \tag{4.15}$$

We shall now establish that

$$E[\tau] \geqslant \frac{E[\gamma_\tau]}{h(q)}. \tag{4.16}$$

Define $\delta_t \triangleq \gamma_t - t \cdot h(q)$. Hence, from Eq. (4.15), it follows that

$$E[\delta_t - \delta_{t-1} | \delta_{t-1}, \ldots, \delta_0, t \leqslant \tau] \leqslant 0.$$

If we could now show that $E[\min\{\delta_t, 0\}] > -\infty$ for all $t$, we will have proven that $\{\delta_t\}$ is a supermartingale [Karl75a]. This condition can be assured by restricting $t$ so that $t \leqslant \tau_T \triangleq \min\{T, \tau\}$, where $\tau_T$ is a Markov time for $\delta_t$. Thus, from the properties of supermartingales, it follows that $E[\delta_{\tau_T}] \leqslant \delta_0 \equiv 0$. Hence

$$E[\delta_{\tau_T}] = E[\gamma_{\tau_T}] - h(q)\cdot E[\tau_T] \leqslant 0.$$

and

$$E[\tau] \geqslant E[\tau_T] \geqslant \frac{E[\gamma_{\tau_T}]}{h(q)}.$$

Since $\lim_{T\to\infty} E[\gamma_{\tau_T}] = E[\gamma_\tau]$, we have shown that Eq. (4.16) holds. Combining Eqs. (4.14) and (4.16), we obtain

$$M(1+q\cdot p) \leqslant E[\gamma_\tau] < h(q)\cdot E[\tau],$$

and hence that

$$p_A \triangleq \lim_{M\to\infty} \frac{Mp}{E[\tau(A,M)]} \leqslant \frac{h(q)}{1/p+q}$$

must hold for every protocol $A$ and every $q$. Thus the channel capacity for the given $p$, $C_p$, must satisfy

$$C_p \leqslant \min_q \frac{h(q)}{1/p+q},$$

giving the desired upper bound.

■

Figure 4.3 shows the numerical value of the upper bound implied by Theorem 4.4 for all $p$. For $p \gtrsim .42$, this new bound is weaker than the genie aided argument of Section 4.2. For $p \lesssim .42$, the bound is stronger than the previous result. As shown by Tsybakov and Mikhailov, this bound converges to $\approx .5875$ in the Poisson limit.

### 4.4: Tighter Bounds for the Class of Degenerate Intersection Protocols

#### 4.4.1: Degenerate Intersection Protocols

In this section, we prove a stronger bound on the performance of a restricted class of protocols, which has been described in [Moll81a]. These protocols must choose each enabled set to satisfy the following *degenerate intersection* property. Following the notation of the previous section, define $\theta_t(S)$ to be the *smallest* number of messages that could be contained in $S$, given the complete channel history up to the start of the $t^{th}$ slot. We assume that at the start of the $t^{th}$ slot, there are exactly $n_t$ distinct minimal *busy subsets* (not necessarily entirely disjoint), $B_1^t, B_2^t, \ldots, B_{n_t}^t$, for which $\theta_t(B_i^t) > 0$. We now require that the enabled set at the $t^{th}$ slot, $E_t$, must satisfy

$$E_t \bigcap B_i^t \in \{E_t, B_i^t, \varnothing\}$$

for all $i = 1, 2, \ldots, n_t$. It is easy to see that if $n_1 = 0$, then for all $t > 0$ the partition of the set of arrival points induced by $E_t$ and $B^t \triangleq \{B_i^t\}$ forms a forest. Thus the intersection operation never generates any new sets.

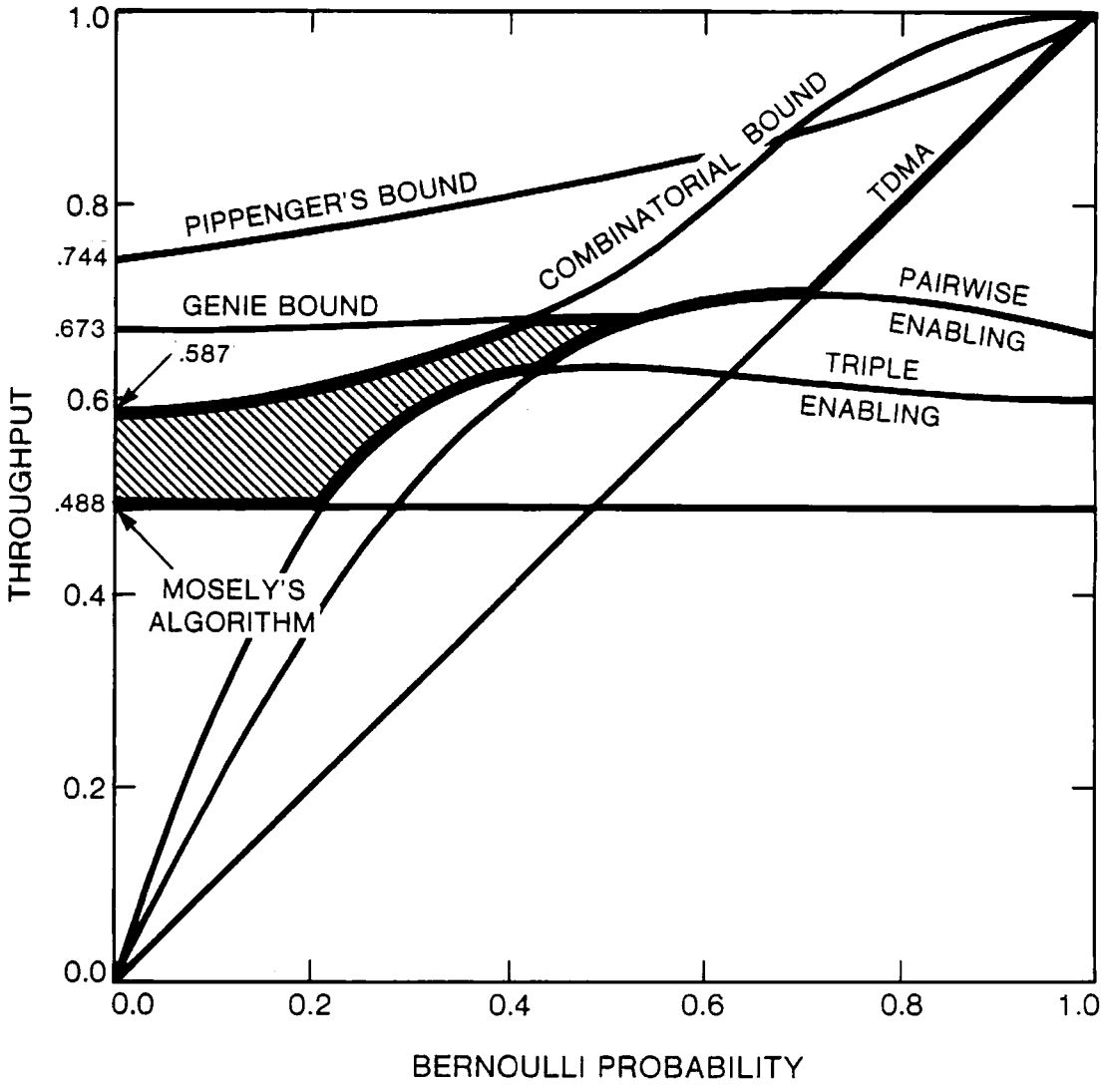


Figure 4.3: Improved Bounds Using the Approach of Tsybakov

All currently proposed protocols satisfy this degenerate intersection property. For example, Hayes [Haye78a] examines ambiguous binary addresses of increasing length in an attempt to isolate busy stations. Each new probe appends a further digit to the end of the last previous ambiguous address that may still specify some waiting ready stations. Thus each set of enabled stations will always be a subset of the last previously enabled set of stations that is not yet known to be completely served. Capetanakis [Cape78a] uses sequences of coin tosses to resolve collisions at random. The set of arrival points consists of all possible infinite sequences of coin tosses, and the enabled sets specify the outcome for a finite subsequence. Like Hayes, Capetanakis' algorithm always enables a subset of the last previously enabled set of stations that still could have waiting messages. Gallager [Gall78a], Humblet, and Mosely [Mose79a] use the real time axis to represent the set of arrival points, and each enabled subset obeys the FCFS property.

**Lemma 4.2**

For all $t > 0$, if $B_i^t \bigcap B_j^t \neq \varnothing$ then $B_i^t \bigcap B_k^t = \varnothing$ and $B_j^t \bigcap B_k^t = \varnothing$ for any distinct sets in $B^t$. Furthermore, when $B_i^t \bigcap B_j^t \neq \varnothing$, it must be the case that $\theta_t(B_i^t \bigcap B_j^t) = 1$, and that $\theta_t(B_i^t \bigcup B_j^t) = 2$.

**Proof:**

We show by induction on $t$ that $B^t$ consists of disjoint sets for which $\theta_t(\cdot) \in \{1, 2\}$, and of nested pairs of sets as described above. Since the claim is trivially true at $t = 1$, assume it true at $t = \tau$ and consider $B^{\tau+1}$. We consider four cases for $E_\tau$.

1. $E_\tau \bigcap B_i^\tau = \varnothing$ for all $B_i^\tau \in B^\tau$:
   If enabling $E_\tau$ gives an *idle slot* or a *success*, then $B^{\tau+1} = B^\tau$. If enabling $E_\tau$ gives a *collision*, then $B^{\tau+1} = B^\tau \bigcup \{E_\tau\}$, and $\theta_{\tau+1}(E_\tau) = 2$.

2. $E_\tau \subseteq B_i^\tau$ and $E_\tau \bigcap B_j^\tau = \varnothing$ for all $B_j^\tau \in B^\tau - \{B_i^\tau\}$:
   If enabling $E_\tau$ gives an *idle slot*, then $B^{\tau+1} = (B^\tau - \{B_i^\tau\}) \bigcup \{B_i^\tau - E_\tau\}$, and $\theta_{\tau+1}(B_i^\tau - E_\tau) = \theta_\tau(B_i^\tau)$. If enabling $E_\tau$ gives an *success*, then $B^{\tau+1} = B^\tau - \{B_i^\tau\}$ if $\theta_\tau(B_i^\tau) = 1$, or if $\theta_\tau(B_i^\tau) = 2$ and there exists $B_j^\tau \in B^\tau - \{B_i^\tau\}$ such that $B_j^\tau \subset B_i^\tau$ (for which $\theta_\tau(B_j) = 1$ by the inductive hypothesis); otherwise, $B^{\tau+1} = B^\tau$ and $\theta_{\tau+1}(B_i) = 1$. If enabling $E_\tau$ gives a *collision*, then $B^{\tau+1} = (B^\tau - \{B_i^\tau\}) \bigcup \{E_\tau\}$, and $\theta_{\tau+1}(E_\tau) = 2$.

3. There exists one busy set $B_i^\tau \subset E_\tau$ but $B_j^\tau \not\subset E_\tau$ for all $B_j^\tau \in B^\tau - \{B_i^\tau\}$:
   Clearly enabling $E_\tau$ cannot give an idle slot because $\theta_\tau(B_i^\tau) > 0$ by assumption. If enabling $E_\tau$ gives a *success*, then $B^{\tau+1} = B^\tau - \{B_i^\tau\}$ if $E_\tau \bigcap B_j^\tau = \varnothing$ for all $B_j^\tau \in B^\tau - \{B_i^\tau\}$, and $B^{\tau+1} = (B^\tau - \{B_i^\tau, B_j^\tau\}) \bigcup \{B_j^\tau - E_\tau\}$ if $E_\tau \subset B_j^\tau$. If enabling $E_\tau$ gives a *collision*, then $B^{\tau+1} = B^\tau$ if $\theta_\tau(B_i^\tau) = 2$, $B^{\tau+1} = B^\tau \bigcup \{E_\tau\}$ if $\theta_\tau(B_i^\tau) = 1$ and $E_\tau \bigcap B_j^\tau = \varnothing$ for all $B_j^\tau \in B^\tau - \{B_i^\tau\}$, and $B^{\tau+1} = (B^\tau - \{B_j^\tau\}) \bigcup \{B_j^\tau - E_\tau\}$ if $E_\tau \subset B_j^\tau$.

4. $B_i^\tau \subset E_\tau$ and $B_j^\tau \subset E_\tau$ for $B_i^\tau \neq B_j^\tau \in B^\tau$:
Enabling $E^\tau$ must result in a collision, so $B^{\tau+1} = B^\tau$.

■

This degenerate intersection property defines a strict superset of the class of FCFS protocols. This is true even if we ignore trivial extensions obtained by transforming both the arrival points and the enabled sets (such as implementing the Gallager-Tsybakov algorithm by tossing fair coins rather than by splitting time intervals) or permuting the order in which the independent sets are enabled. In particular, a FCFS algorithm may reach a state in which a nested pair of sets, $B_1^t \subset B_2^t$, are known to contain messages, and $\theta_t(B_1^t) = 1$ while $\theta_t(B_2^t) = 2$ [Mose79a]. By the FCFS property, this can happen only if $B_1^t$ contains the "oldest" message, so a FCFS protocol is required to enable a subset or superset of $B_1^t$ in the next slot. The degenerate intersection property, however, permits us to consider protocols that could choose to enable a subset of $B_2^t - B_1^t$, which would have be forbidden under FCFS.

We now show that if $A^*$ is optimal over the class of degenerate intersection protocols for some $p$, and if the capacity of $A^*$ exceeds .5, then $A^*$ must never enable a strict superset of any known busy set. To simplify the proof, we must temporarily consider degenerate intersection protocols for the following *genie-aided* problem. Whenever a genie-aided protocol enables a strict superset of exactly one known busy set, not necessarily disjoint from all other known busy sets (*i.e.*, case 3. in Lemma 4.2, above), we assume that this (new) genie will examine $E_t - B_i^t$ in some prearranged order and publicly label the *first* busy point in $E_t - B_i^t$, if any. We note that this may also create some known idle points. It is clear from Lemma 4.2, above, that $B^t$ for any genie-aided protocol must contain only mutually disjoint sets, some of which may be single points that have been previously labelled by the genie.

We now show by contradiction that a genie-aided degenerate intersection protocol $A^*$ that enables supersets of known busy sets cannot be optimal if its capacity exceeds .5. Consider the behaviour of $A^*$ over a randomly chosen sample execution. Each time the genie labels a new busy point from $E_t - B_i^t$, let us "tag" the $t^{th}$ slot, the newly-labelled busy point, and the (future) slot during which this tagged point is transmitted successfully. We note that the above rule for tagging slots is such that we can choose to tag the $t^{th}$ slot *only* if enabling $E_t$ gave no new information about $B_i^t$. In addition, let us also tag all idle points that are labelled by the genie, even if the slot in which they are labelled is not a tagged slot, or if there was a success so the genie's labelling information is redundant.

We now choose to determine the performance of $A^*$ over that sample execution by considering its performance over the *tagged* slots and points separately from its performance over the *untagged* slots and points. Clearly the efficiency with which the tagged busy points are transmitted over the tagged slots is exactly .5. If the genie labels only idle points, we tag the idle points but not the slot. If the genie labels also one busy point, we tag *two* slots to transmit *one* one busy point (*i.e.*, the slot in which it is tagged and the slot in which it is transmitted successfully). We now examine the efficiency with which the untagged points are transmitted over the untagged slots. The untagged slots and points may be viewed as a sample execution of

another protocol, $A'$ say, which never enables supersets of known busy sets. We note that $A'$ must be a valid protocol, *even without the genie,* because the tagged slots, which we have ignored, gave no information about *any* untagged points, and because even complete information about the tagged points, which we have also ignored, gives no information about the untagged points by the independence of the arrival points.

It is clear by construction that the efficiency of $A^*$ over any sample execution cannot exceed the maximum of .5 and the efficiency with which $A'$ transmits the untagged points. Since we have already assumed that the capacity of $A^*$ exceeds .5, and since a convex combination of two unequal numbers is strictly less than their maximum, it follows that $A^*$ must have a lower capacity than $A'$ — which contradicts the optimality of $A^*$.

Thus, for any Bernoulli $p$ for which there exists a degenerate intersection protocol with a capacity of at least .5, we have shown that there must be an *optimal* degenerate intersection protocol that never enables supersets of known busy sets. This result also allows us to form an upper bound for all $p$ on the capacity of arbitrary degenerate intersection protocols. This upper bound is simply the maximum of .5 and the capacity of the best degenerate intersection protocol that never enables supersets. It now remains to find the best degenerate intersection protocol for all $p$ that never enables supersets. Such protocols process each known busy set independently, so without loss of generality, we need examine only FCFS protocols.

Recall the FCFS protocols from Section 4.2 that were used to lower bound capacity as a function of $p$. Since the capacity of the "triple enabling" protocols (*i.e.*, $N=3$) is above .5 over the relevant values of $p$, we know from this last result that the optimal degenerate intersection protocol for $N=3$ cannot enable supersets of known messages. Thus the optimal degenerate intersection protocol for $N=3$ can clearly be found by enumerating all possible FCFS protocols for $N=3$ that do not enable supersets.

Whenever a collision among three arrival points occurs, any such protocol must enable either a single point or a pair of points. Enabling a *single* point from the original collision must give either an idle slot or a success. If it is idle, the two remaining points must be enabled separately to avoid a certain collision. If it is a success, the two remaining points, now known to contain at least one busy point, may be enabled separately or together. If they are enabled separately, there is no point in separately enabling the third point if the first two points were busy — it is effectively unexamined with probability $p$ of being busy. Enabling it corresponds to mixing the $N=3$ protocol with an *independent* $N=1$ protocol, which cannot be as efficient as the best of the two strategies. Enabling a *pair* of points from the original collision must give either a success or a collision. If it is a success, we have no choice but to enable the last remaining (busy) point. If there is a collision, we should again ignore the third point, now effectively unexamined, and separately enable each of the first two (busy) points.

Having thus enumerated all candidates for the optimal degenerate intersection protocol for $N=3$, we see that all the candidates have already been examined in Section 4.2. Thus the best of these three "triple enabling" protocols from Section 4.2 really is the optimal degenerate intersection protocol over the relevant values of $p$. However, our goal was to construct the *complete* capacity curve for degenerate intersection protocols. Consequently, we must also find optimal protocols for all $N \geq 4$. It is clear that the combinatorial explosion required for exhaustive enumeration rapidly makes this approach impractical as $N$ increases. We are thus led to consider yet another genie-aided problem in order to upper bound the capacity of degenerate intersection protocols.

### 4.4.2: Optimal Genie-Aided Degenerate Intersection Protocols

We now assume that a genie examines the points in each busy set $B_i^t$ for which $\theta_t(B_i^t)=1$, in some prearranged order, and publicly labels the *first* busy point in $B_i^t$. This information may also create some known idle points. It is clear from Lemma 4.2 that any genie-aided degenerate intersection protocol faces only known busy points, disjoint busy sets for which $\theta_t(\cdot)=2$, and unexamined arrival points.

The information provided by this genie is a strict superset of the information provided by the other genie described earlier in this section. Consequently, the same argument can again be used to prove that degenerate intersection protocols aided by this new genie cannot be optimal if they enable supersets of known busy sets and their capacity exceeds .5. Similarly, we can form an upper bound on the capacity of genie-aided (and, thus, also *unaided*) degenerate intersection protocols for all $p$ from the maximum of .5 and the capacity of the best genie-aided degenerate intersection protocol that never enables supersets of known busy sets. Without loss of generality, only FCFS protocols need be considered.

Any genie-aided FCFS protocol that never enables supersets begins a new period of activity whenever it enables some (possibly random) number $N$ of unexamined arrival points. Each idle period takes one slot to process no messages, and each success takes one slot to process one message. However, unlike Section 4.2, we now require the protocol to search for the *first* busy point involved in each collision before the genie identifies a *second* busy point. Consequently, each collision takes *at least three* slots to process two messages (one for the collision, at least one to find the first busy point, and one more to transmit the busy point that is labelled by the genie). Thus, to find an upper bound on throughput for genie-aided protocols, we must find a lower bound on the expected number of slots in a collision busy period.

Suppose a FCFS protocol is faced with a set $B^t$, for which $\theta_t(B^t)=2$. We assume that $B^t$ contains $N$ arrival points, each initially having probability $p$ of an arrival. By the FCFS property, the protocol must enable some subset $E^t \subset B^t$ containing $M<N$ arrival points. The probability of an idle slot when $E_t$ is enabled can be found as a weighted sum of hypergeometric terms, *i.e.*,

$$Pr[idle|N, M, p] = \sum_{i=2}^{N} H(0, M, i, N) \cdot Pr[i \text{ messages in } N | i \geq 2]$$

$$= \sum_{i=2}^{N} \frac{\binom{N-i}{M}}{\binom{N}{M}} \cdot \frac{\binom{N}{i} p^i (1-p)^{N-i}}{1-(1-p)^N - Np(1-p)^{N-1}}$$

$$= \frac{(1-p)^M \left[1-(1-p)^{N-M} - (N-M)p(1-p)^{N-M-1}\right]}{1-(1-p)^N - Np(1-p)^{N-1}}.$$

Similarly, the probability of a success when $E_t$ is enabled is

$$Pr[success|N, M, p] = \sum_{i=2}^{N} H(1, M, i, N) \cdot Pr[i \text{ messages in } N | i \geq 2]$$

$$= \sum_{i=2}^{N} \frac{\binom{i}{1}\binom{N-i}{M-1}}{\binom{N}{M}} \cdot \frac{\binom{N}{i} p^i (1-p)^{N-i}}{1-(1-p)^N - Np(1-p)^{N-1}}$$

$$= \frac{Mp\left[(1-p)^{M-1} - (1-p)^{N-1}\right]}{1-(1-p)^N - Np(1-p)^{N-1}},$$

and the probability of a collision when $E_t$ is enabled is

$$Pr[collision|N, M, p] = 1 - Pr[idle|N, M, p] - Pr[success|N, M, p].$$

In the Poisson limit as $p \rightarrow 0$, $Np \rightarrow \lambda$ and $Mp \rightarrow \gamma$, we obtain

$$Pr[idle|\lambda, \gamma] = \frac{e^{-\gamma}\left[1 - e^{-(\lambda-\gamma)} - (\lambda-\gamma)e^{-(\lambda-\gamma)}\right]}{1-(1+\lambda)e^{-\lambda}},$$

$$Pr[success|\lambda, \gamma] = \frac{\gamma\left[e^{-\gamma} - e^{-\lambda}\right]}{1-(1+\lambda)e^{-\lambda}},$$

and

$$Pr[collsion|\lambda, \gamma] = 1 - Pr[idle|\lambda, \gamma] - Pr[success|\lambda, \gamma].$$

Finding the minimum expected number of slots until the first message in $B^t$ is successfully transmitted, $\sigma^*_N$, is a multistage decision process that can be solved using dynamic programming [Bell57a]. It is well known that optimal solutions of multistage decision problems must obey the following *principle of optimality*. No matter what the initial state is, and what initial decision is made, a policy cannot be optimal unless the remaining decisions represent an optimal policy for the new state of the process that results from the first decision.

Thus, for any $N$ and any Bernoulli probability $p$, $\sigma^*_N$ can be found numerically from the following recursive equation:

$$\sigma^*_N \triangleq 1 + \min_{0<M<N}\left\{Pr[idle|N, M, p]\cdot\sigma^*_{N-M} + Pr[collision|N, M, p]\cdot\sigma^*_M\right\}$$

under the initial condition $\sigma^*_2 \triangleq 1$. In the Poisson limit, we have

$$\sigma^*_\lambda \triangleq 1 + \min_{0<\gamma<\lambda}\left\{Pr[idle|\lambda, \gamma]\cdot\sigma^*_{\lambda-\gamma} + Pr[collision|\lambda, \gamma]\cdot\sigma^*_\gamma\right\}$$

with the initial condition $\lim_{\lambda\to 0}\sigma^*_\lambda \to 2$.

Over all periods where a particular value of $N$ is chosen, the conditional genie-aided throughput, $\tilde{\rho}_N$, is found from a renewal argument to be

$$\tilde{\rho}_N \triangleq \frac{S_N + 2C_N}{I_N + S_N + (2+\sigma^*_N)\cdot C_N} \tag{4.17}$$

where $I_N = (1-p)^N$, $S_N = Np(1-p)^{N-1}$ and $C_N = 1 - I_N - S_N$ are the probabilities that enabling $N$ independent Bernoulli arrival points gives an idle slot, a success or begins a collision busy period, respectively. Since the unconditional throughput is a convex combination of $\{\tilde{\rho}_N\}$, it cannot exceed $\tilde{\rho}_{N^*}$, where $N^*$ achieves maximum conditional throughput.

We have thus established that an optimal genie-aided strategy is to enable the first known busy point if there is one, or to enable some fixed number $N^*$ of unexamined arrival points otherwise. Figure 4.4 shows the upper bound for all $p$ in the range $0 \leqslant p \lesssim .375$, where the performance of genie-aided degenerate intersection protocols with $N \geqslant 4$ exceeds the performance of TDMA, "pairwise enabling", and "triple enabling". Computation reveals that $\approx .508$ is an upper bound on the capacity of degenerate intersection protocols for Poisson arrivals. Since protocols with capacity $\approx .488$ are known to exist [Mose79a], only a small range of uncertainty in the capacity of degenerate intersection protocols remains.

### 4.5: Infinite Population Results as the Asymptotic Behaviour of Large Finite Systems

Recently, a number of adaptive multiple access protocols have been proposed in which the rules for selecting the enabled set varies with the (short-term) average load on the channel. For example, the URN scheme [Klei78a] adapts from slotted ALOHA under light traffic to TDMA under heavy traffic. Finite population adaptive tree protocols [Haye78a, Cape79a, Mark80a] have a similar behaviour.

It is instructive to compare the throughput-delay curves for such adaptive systems as the number of stations, $M$, grows larger. In spite of vast differences in the details of the protocols, these curves all exhibit an unmistakable "knee" in the middle range of channel utilizations. Furthermore, as $M$ increases, it is characteristic of these curves for the knee to become sharper and move towards lower values of utilization and higher values of delay simultaneously. (See, for example, Figure 3.2 in [Cape79a] or Figure 5 in [Klei78a]. Figure 3 in [Mitt81a] is

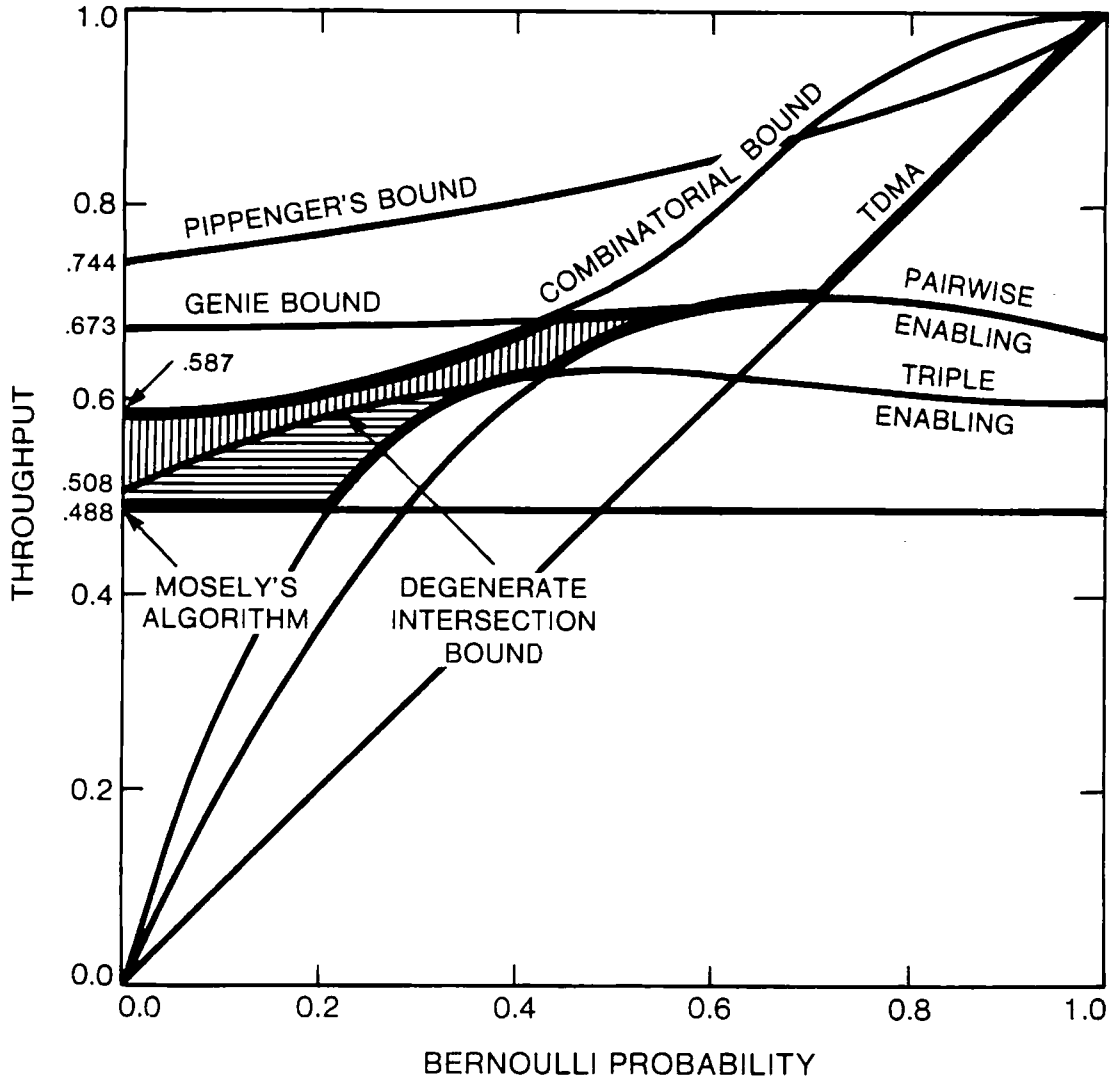


Figure 4.4: Performance Bounds for Degenerate Intersection Protocols

particularly interesting, since it shows a family of delay curves for $M = 10, 20, 50, 100$.)

The explanation for this phenomenon lies in the results that we have presented above for infinite population bounds on capacity. When the channel utilization is well below the infinite population capacity of the protocol for Poisson arrivals, the delay will be essentially independent of $M$. This result forms the steep face of the knee.

However, we have proven above that TDMA is globally optimal with Bernoulli arrivals when we can select a single busy point with high enough probability. Since the probabilities of *individual stations* being busy is an increasing function of the channel utilization in the finite population case, TDMA must also be a globally optimal protocol for high enough values of the channel utilization. Thus the delay curve for any adaptive protocol must eventually be lower bounded by the corresponding delay curve for TDMA.

To lower bound the delay curve for intermediate channel utilizations, we note that the adaptive mechanism in all of the above protocols limits the number of contending messages in any single slot to some value that is a *decreasing* function of the channel utilization. Thus, for $M$ sufficiently large, a first order approximation of the behaviour of any such protocol shows that an individual station is dormant for a long (random) time between each opportunity to transmit a message. The time between periods of activity dominates the time to resolve any conflicts for transmission rights with other stations having overlapping periods of activity. Thus, by the law of large numbers, as $M \rightarrow \infty$ the coefficient of variation of the time between opportunities to transmit decays to zero. The Bernoulli bounds presented above show by how much we can reduce the *apparent* number of stations in the cycle through "local contention", and hence reduce the delay to some fraction of the value of TDMA.

For example, when the perceived utilization of an individual station as observed by the protocol (*i.e.*, $p$) is 1/2, we see from §4.2.2 that pairwise enabling can attain a channel utilization of $\frac{2\cdot(1/2)}{1+2\cdot(1/2)^2} = 2/3$. Thus, for $M$ sufficiently large, were we to use pairwise enabling rather than TDMA, the average time to complete a round-robin cycle would decrease to $3M/4$ slots compared to $M$ slots for TDMA. This permits a channel of lower capacity to support the same number of stations with the same mean delay.

In general, if each station is ready to transmit with probability $p$, and for the given $p$, the protocol can achieve a channel utilization of $\rho \geq p$, then the number of stations in the equivalent TDMA network, $M'$, is given by

$$\rho \cdot M' = p \cdot M,$$

or

$$M' = \frac{p}{\rho} \cdot M. \tag{4.18}$$

Thus for large $M$, the delay equation is approaching a fraction of the corresponding result for TDMA:

$$T_{LB}(p, M) = 1 + \frac{M'}{2(1-p)} \approx \frac{Mp}{2p(1-p)}.$$

Figure 4.5 shows the limiting behaviour of $T_{LB}/T_{TDMA}$ as $M \to \infty$.

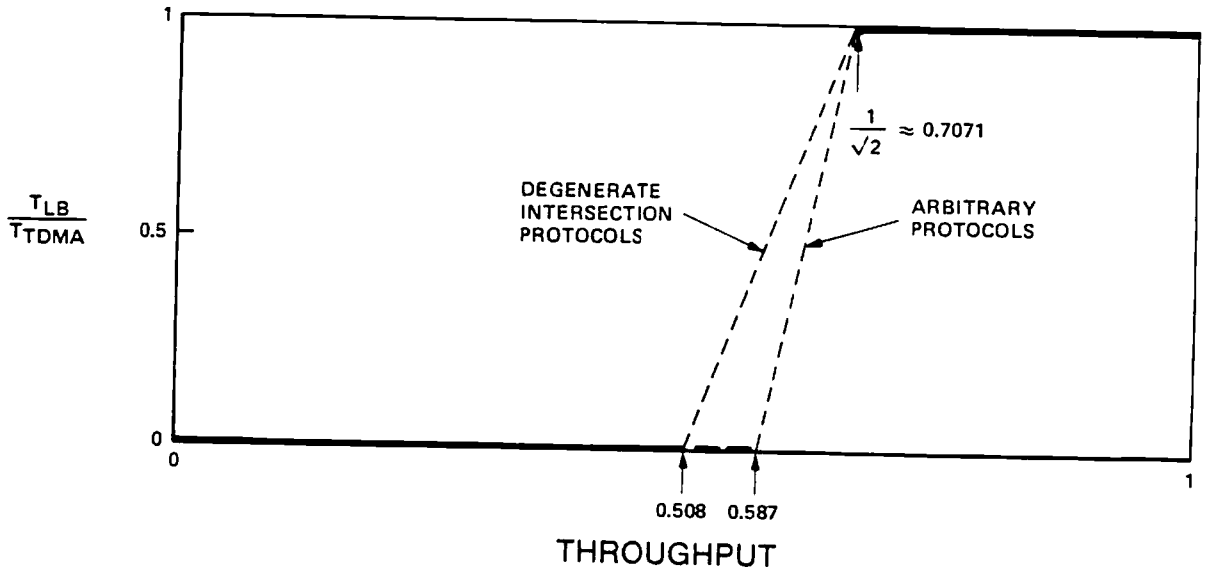


Figure 4.5: A Lower Bound on Delay for Arbitrarily Large Systems

# CHAPTER 5
## Protocols with a Partial Reservation Channel

### 5.1: Reservation-Based Protocols

In previous chapters, we examined protocols that operate without the exchange of any explicit protocol messages. However, many practical protocols rely on more information than just the passive observation of channel history. In this chapter we examine reservation-based protocols.

We identify two classes of reservation-based protocols, pure reservation and partial reservation strategies. *Pure reservation* strategies restrict the contention problem in multiple access to a *reservation sub-channel.* Each reservation request is large enough to identify the station that attempted to make the reservation. It may also provide some information about the message. Such a reservation request requires exclusive access to the reservation channel. Although one could conceive of a multi-level reservation protocol, at some point a non-reservation multiple access protocol must be used to control access to the reservation sub-channel. Some, like reservation-ALOHA [Lam80a], are not applicable for the infinite population case, for no station will have a backlog of waiting messages. Like [Tsyb80c], most pure reservation protocols may be thought of as a device for increasing channel utilization by, in effect, decreasing the idle and collision detect times. As first observed by Capetanakis [Cape78a], such pure reservation protocols cannot solve the multiple access problem without appealing to another, non-reservation protocol operating on the reservation channel. Pure reservation protocols are also sensitive to errors in the receipt of *reservations.* Inconsistencies in the distributed "reservation queue" will lead to catastrophic results! We shall therefore not attempt to design or analyze yet another pure reservation protocol, bearing in mind that our other protocols could easily be used to control access to the reservation channel in any pure reservation scheme.

*Partial reservation* strategies are based on the exchange of imprecise reservation requests conveying only a few "bits" of information. Such partial reservation requests require little bandwidth on the channel. Their size can be made independent of the number of stations in the network. It may also be possible to encode them in a manner that does not depend on granting each such request exclusive access to the reservation channel. However, a partial reservation request does not necessarily contain enough information to identify the requesting station uniquely. Thus protocols that use a partial reservation channel may require both the reservation sub-channel and the message channel to completely solve the multiple access contention problem.

An alternate approach is to construct pure reservations for each message from a sequence of partial reservations. Such protocols can work well with small populations, but are inherently inefficient with very large populations. The tree algorithms of Hayes [Haye78a] and Mark [Mark80a] require a sequence of $\log_2(M)$ binary reservations to exactly determine the identity of a ready station, thus exhibiting a logarithmic degradation in efficiency as $M$ increases. Various forms of round-robin scheduling under distributed control also fall into this category. MSAP [Klei77a] and BRAM [Chla79a] pass a "silence token" of length equal to a propagation time across the network between the stations, requiring a total of $M$ binary reservations per round-robin cycle. Hamacher [Hama80a] proposed an auxiliary channel for the binary reservations, on which signals are actively propagated in only one direction. This reduces the overhead per round-robin cycle since the "token" need only propagate to the next active station. However, to prevent race conditions, a small constant delay must be inserted for each station. Thus, in the limit, round-robin scheduling protocols exhibit a linear degradation with $M$ in the channel overhead.

## 5.2: Ternary Reservations

### 5.2.1: The Inefficiency of the URN Scheme

The URN scheme [Klei78a] is perhaps the most well known protocol that uses a partial reservation channel. Each station announces the arrival of new messages on a slotted ternary channel that is monitored by all stations. These announcements are received as "0", "1", or "e" — corresponding to *no* request, a *single* request, or a *multiple* request, respectively. This information provides an estimate of the number of stations that are ready to send messages. The URN policy then enables just enough stations to maximize the probability of selecting exactly one ready station. We note that in the limit of very large systems, the URN policy is, in some sense, the "dual" of optimally-controlled ALOHA: where optimal ALOHA chooses the *probability* with which each active station should (independently) choose to transmit given the number of active stations, the URN policy selects the *number* of stations (not necessarily known to be active) that should be granted permission to transmit given the probability that a station would transmit if it were granted permission.

A serious limitation to the URN scheme is its dependence on a ternary reservation channel. Stations in a radio environment suffer from a *saturation* effect. Whenever a station transmits, its own signal overpowers all others in its immediate neighbourhood. Consequently stations are unable to determine whether their *own* arrival announcements are received as single or multiple requests by the other stations. Thus only *inactive* stations can reliably estimate the number of ready stations in the network — and, of course, they have the least use for the information!

More importantly, if you *are* willing to provide a ternary reservation channel, the URN protocol discards almost all the information that it provides. All stations can (trivially) completely order the slots on the reservation channel by *time*. However, some reservation channel slots carry *multiple* reservation requests. Thus, the ordering of the reservation channel slots defines only a *partial* ordering on the messages. Thus, for each multiple reservation, two or

more messages remain mutually incomparable. The minimal amount of further work required of a multiple access protocol is just to order these incomparable messages. The URN scheme, on the other hand, discards all of this ordering information. Only an estimate of the number of messages remains. Consequently, the URN scheme is efficient only when the system is either almost full — which is not allowed with an infinite population — or almost empty. Mittal and Venetsanopoulos [Mitt81a] have studied the transient behaviour of URN using a fluid approximation. To improve its efficiency, they describe a dynamic entrance control policy to force the system to remain lightly loaded. Below we show that this is not necessary, and that it is in fact undesirable.

### 5.2.2: Multiple Access with a Partially Ordered Queue of Messages

The URN scheme operates with limited storage. Only a single integer $N$, the estimate of the number of ready stations, is required. However, significantly improved performance is possible by using the ordering information from the reservation channel to form a *partially ordered queue* of messages. Such a queue is illustrated in Figure 5.1. All reservation slots that contain no requests require no further service, and are ignored; all reservation slots that contain one or more reservation requests join the queue for access to the message channel. With a ternary reservation channel, it is also necessary to save a one bit "tag" for each entry in the queue to distinguish between single (tagged "0") and multiple (tagged "1") reservations.

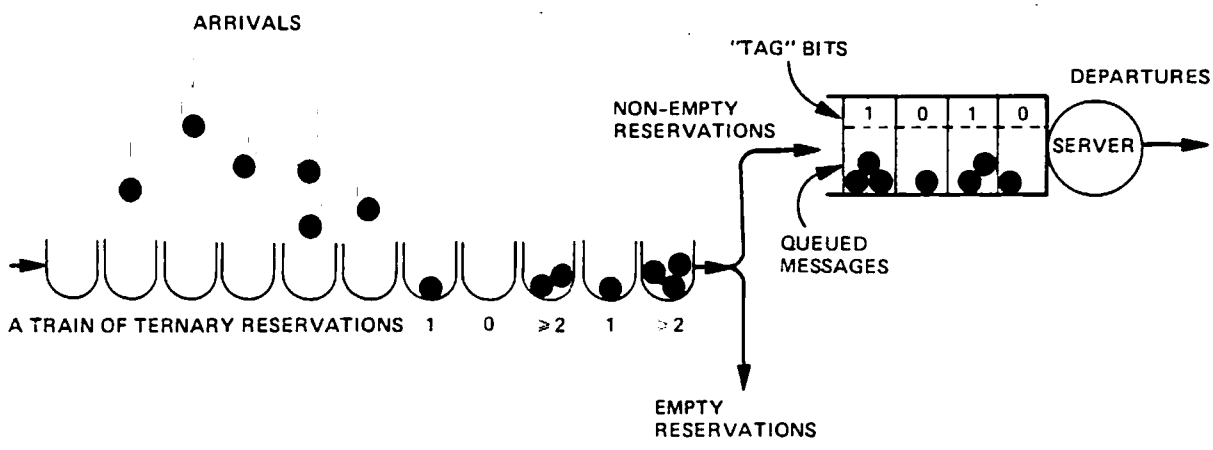


Figure 5.1: Using Reservation Requests to Form a Partially Ordered Queue

To implement such a queue in a distributed manner, it is sufficient for each station to maintain a count $Q_t$ of the current backlog of unserviced reservation requests. For each reservation slot containing one or more reservation requests, $Q_t$ is incremented by one; at the completion of service for each queued reservation (using some collision resolution algorithm) $Q_t$ is decremented by one. To gain access to the channel, a station announces the arrival of its new message on the reservation channel and joins the queue for the message channel. Its position in the message channel queue is given by the value of $Q_t$ just before this announcement.

When its reservation request reaches the head of the queue, the station is allowed access to the channel according to the rules of the collision resolution algorithm.

Maintaining a partially ordered queue of messages has two advantages over the input controlled URN scheme [Mitt81a]. First, the input control procedure rejects some newly-arriving messages to limit the number of messages in the system. If a performance analysis based on equilibrium conditions is to make any sense, the regeneration times for these rejected messages must be large enough to minimize the positive feedback on the arrival rate. A realistic analysis must include the regeneration delays for rejected messages in the delay calculation. Maintaining a partially ordered queue of messages never rejects any newly-arriving messages, so we expect real delays to be greatly reduced. Second, the efficiency of known collision resolution algorithms decreases monotonically as we increase the number of messages that must be handled simultaneously. Mittal's input control procedure tries to take advantage of this fact by limiting the number of messages in the system. However, since this control procedure works by *rejecting* messages rather than *queueing* them, the control procedure must tolerate a larger average number of messages in the system if it is to limit the number of times that the system empties and remains idle for a (possibly long) inter-message generation time. Maintaining a queue of reservations allows individual reservations (each representing a only a few messages) to be resolved separately, and thus more efficiently, without emptying the system until the entire queue of reservations is exhausted.

The composition of a queue of reservations can be found as follows. We assume that the input to the reservation channel is a sequence of independent Poisson samples with parameter $\lambda$. Each non-empty sample will be queued for the message channel. Thus, each queued reservation is an independent Poisson sample (conditioned on being non-empty) with parameter $\lambda$, where samples of exactly one message are distinguishable from samples with more than one message by the one-bit tag mentioned above. Thus, for $k > 0$, the probability that a non-idle reservation represents $k$ messages is given by

$$q_k = \frac{\frac{\lambda^k}{k!} e^{-\lambda}}{1 - e^{-\lambda}}. \tag{5.1}$$

The curve in Figure 5.2 labelled 'Ternary Reservations, Tagged Queue' shows the average utilization of the message channel as a function of $\lambda$ under heavy traffic (*i.e.*, another non-empty reservation is always available at the end of each service epoch) when each multiple reservation request is tagged, and the Gallager-Tsybakov algorithm[1] is used on the message

[1] This algorithm may discard some messages from one epoch and process them in a later epoch. So were we to just process the queue of reservations directly, we would have a loss system. Like ALOHA, this would require randomized retransmissions to preserve the Poisson statistics of the arrivals. However, we can extend the model to avoid this loss, and its attendant problems of instability and long delays, without affecting the capacity of our system. In the spirit of the original algorithm, let a window control access to the reservation channel. Thus if the algorithm subsequently discards any part of this window, the discarded portion can be included in the next reservation channel window.

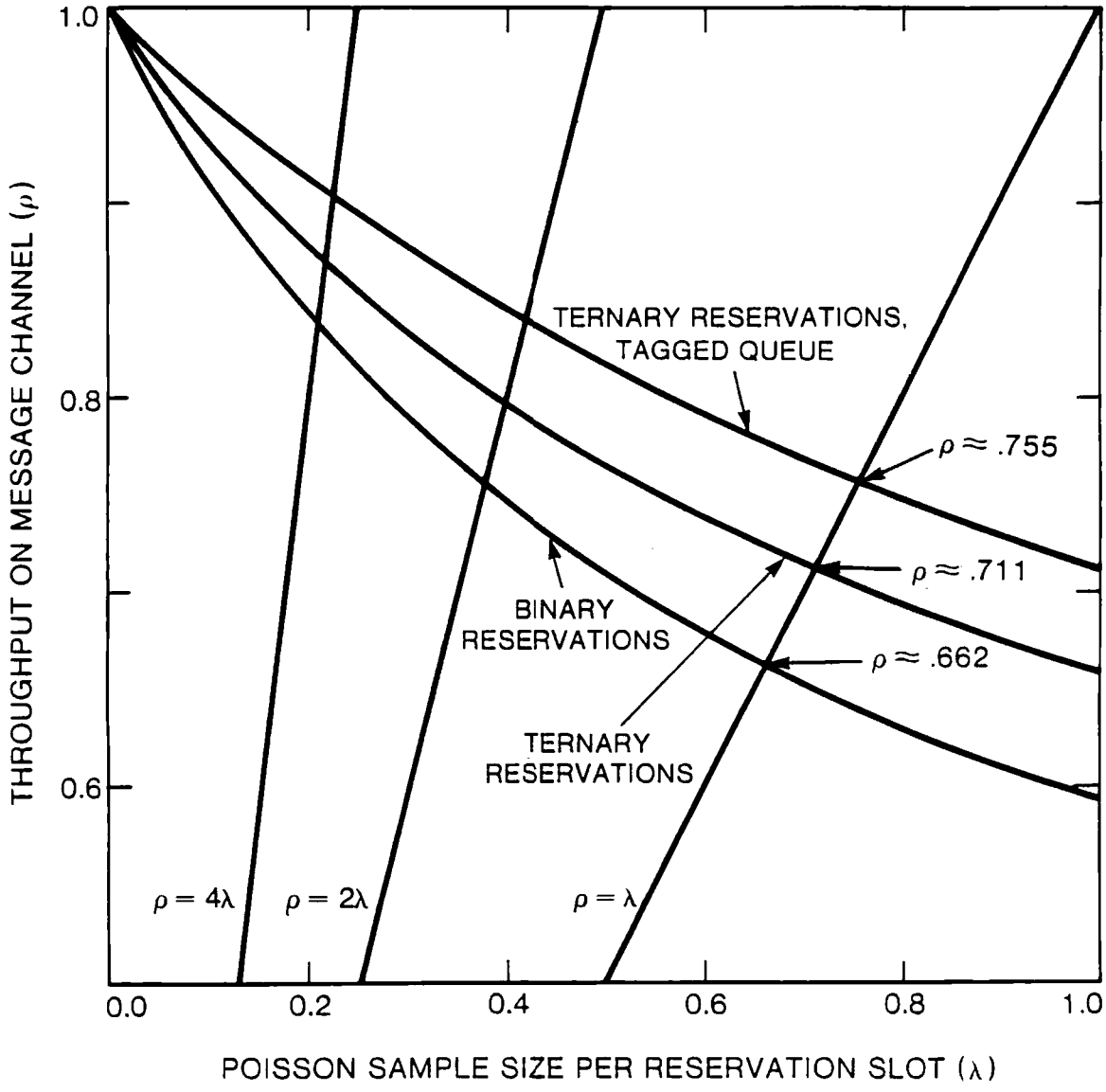


Figure 5.2: The Efficiency of the Gallager-Tsybakov Algorithm with Reservations

channel to resolve collisions. Here the reservation channel eliminates the epochs with no messages and the knowledge of multiple requests eliminates the initial collision in processing epochs where $k > 1$. The sets $\{n_k\}$ and $\{w_k\}$ are given by Eqs. (3.1) and (3.2), and the throughput for any $\lambda$ is given by

$$\rho = \frac{\sum_{k=1}^{\infty} q_k \, n_k}{\sum_{k=1}^{\infty} q_k \, w_k - (1 - q_1)}. \tag{5.2}$$

The capacity of this partial reservation system when exactly one (free) reservation request per message slot is provided is the value at which the average input rate, $\lambda$, equals the average utilization of the message channel, *i.e.*, $\approx .755$. We note that this is considerably more than the capacity of either the Gallager-Tsybakov algorithm ($\approx .487$), which requires no reservation channel, or the URN scheme ($1/e$ with an infinite population), which also requires a (free) ternary reservation channel. However, it may be necessary to store arbitrarily many "tag" bits.

We can eliminate the need to store tag bits by modifying the above protocol. For each multiple reservation request, let the reservation window be split immediately and $Q$ be increased by two. Then only the single integer $Q$ is required. The algorithm then proceeds as if each queued reservation initially contains one message. This strategy still avoids the certain initial collision when $k > 1$, but an empty reservation is queued with probability $p_{k,0} + p_{k,k}$. We have provided no way to remember the dependence between the two half-windows resulting from an multiple request, so a collision that would have been avoided by the previous protocol is introduced with probability $p_{k,0}$. Thus storage requirements are reduced at the expense of some loss of efficiency. The throughput as a function of $\lambda$ is now reduced to

$$\rho = \frac{q_1 \, n_1 + \sum_{k=2}^{\infty} \sum_{i=0}^{k} q_k \, p_{k,i} \, (n_i + n_{k-i})}{q_1 \, w_1 + \sum_{k=2}^{\infty} \sum_{i=0}^{k} q_k \, p_{k,i} \, (w_i + w_{k-i})} \tag{5.3}$$

The capacity of this modified system, assuming one (free) reservation request per message slot, has been reduced from $\approx .755$ with the 'tag' bits to $\approx .711$ without them.

The performance of the above protocols improves as the number of reservation requests per message slot increases. The Poisson sample size at the reservation channel can be decreased, reducing the likelihood of (ambiguous) multiple requests, while still producing enough non-empty reservations to keep the message channel occupied. The capacity with $\eta$ reservations per message slot can be seen as the intersection in Figure 5.2 of the throughput function with the line $\rho = \eta\lambda$. One can easily see that the capacity asymptotically approaches 1 as $\eta \to \infty$. However, the Gallager-Tsybakov algorithm can also be run directly on the ternary reservation channel to generate conflict-free reservations. If an independent use of the Gallager-Tsybakov algorithm were to control the unreserved slots of the message channel, the system would have a capacity of $.487\,\eta + (1 - .487\,\eta)\,.487$. When $\eta = 1$, the capacity is $\approx .737$, which is intermediate between our two previous protocols. Here, however, the improvement is

linear in $\eta$; full utilization of the message channel becomes possible as soon as the rate of reservations exceeds $1/.487 \approx 2.05$ per message slot.

### 5.3: Binary Reservations

We shall now consider the more realistic case of *binary* reservation channels, *i.e.*, only "no requests" and "at least one request" can be distinguished on this channel. It avoids the difficulty of implementing a ternary reservation channel in a radio environment. It also reduces the amount of information that must be stored. A single integer $Q$, the current backlog of non-empty reservations, preserves all the information provided by the reservation channel.

The curve in Figure 5.2 labelled 'Binary Reservations' shows the average utilization of the message channel as a function of $\lambda$ under heavy traffic (so that the supply of binary reservations does not run out) when collisions on the message channel are resolved by the Gallager-Tsybakov algorithm. No information besides the count $Q$ is required. Here $\{n_k\}$ and $\{w_k\}$ are given by Eqs. (3.1) and (3.2), $\{q_k\}$ is given by Eq. (5.1), and the throughput as a function of $\lambda$ is given by Eq. (3.4). Again, the capacity of the system, assuming one (free) reservation slot per message slot can be found graphically from Figure 5.2 to be $\approx .662$. The capacity of this system with $\eta$ reservation slots per message slot is shown in Figure 5.3. While it is clear that the capacity asymptotically approaches 1 as $\eta \to \infty$, one can see that the marginal improvement from increasing $\eta$ decays rapidly. Two other curves are shown for comparison.

Instead of resolving collisions with the Gallager-Tsybakov algorithm, we could also choose the "ALOHA" approach and just discard collisions. Here each non-empty reservation is given exactly one slot on the message channel: any message that suffers a collision is immediately destroyed and must be regenerated after a long, random delay. Thus, the probability of a successful transmission given a non-empty reservation is simply

$$\frac{\lambda e^{-\lambda}}{1-e^{-\lambda}} = \frac{\lambda}{e^{\lambda}-1}. \tag{5.4}$$

The probability of having a non-empty reservation available must be $\min(\eta(1-e^{-\lambda}), 1)$ by conservation of flow, which we can represent by the constraint:

$$\eta \leqslant \frac{1}{1-e^{-\lambda}}$$

or

$$\lambda \leqslant -\ln(1-1/\eta).$$

Recalling that there can be no useful throughput if all the non-empty reservations have been serviced, the throughput as a function of $\eta$ and $\lambda$ must be

$$\rho(\eta, \lambda) = \eta \lambda e^{-\lambda}.$$

For any positive $\eta$, the capacity of this system can be found as a constrained optimization over the feasible values of $\lambda$ using the method of Lagrange multipliers [Luen73a]. For any $\eta$, we see that the optimum value of $\lambda$ must be either $\lambda = 1$ or $\lambda = -\ln(1-1/\eta)$. However, since it is

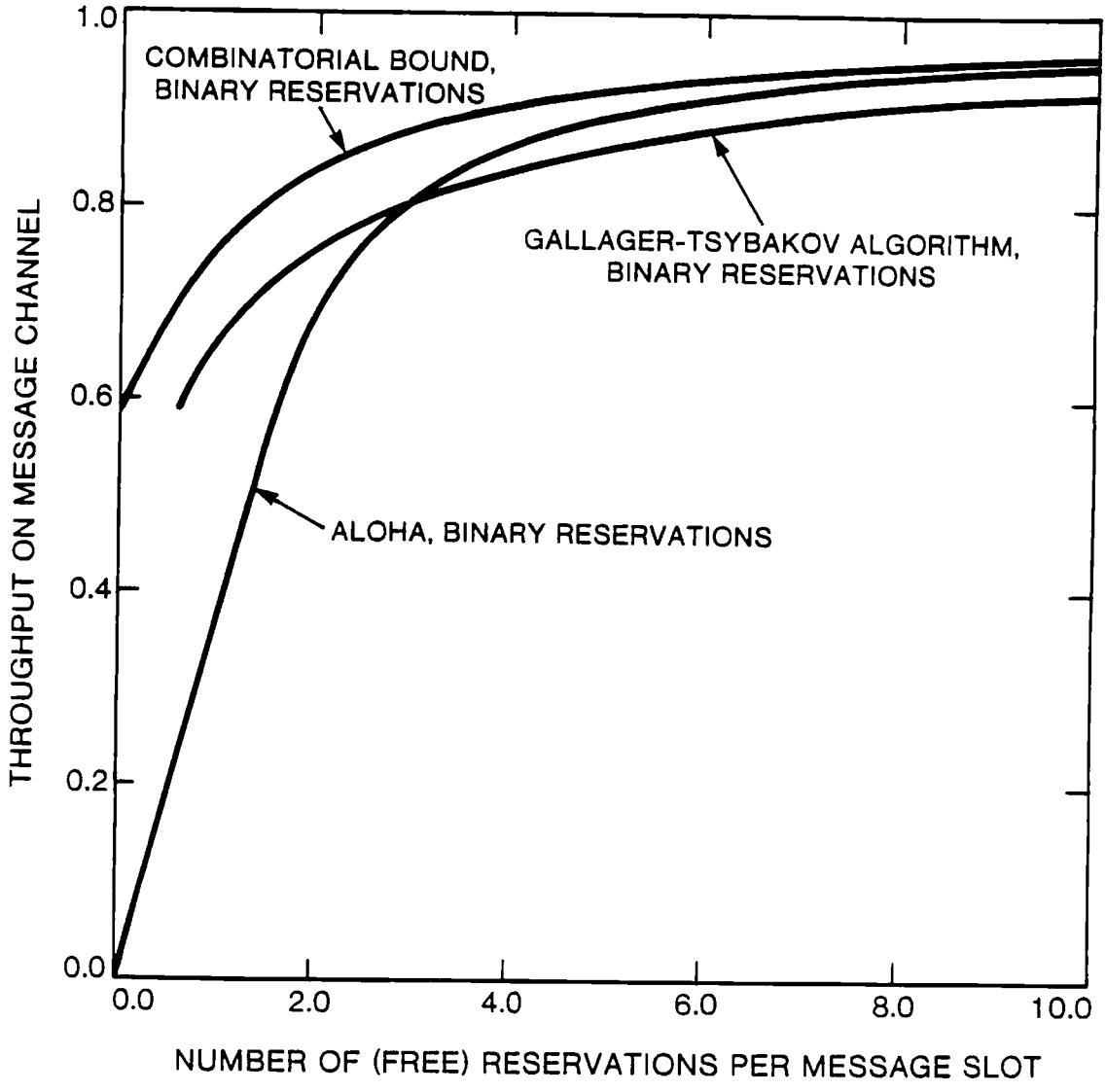


Figure 5.3: Sensitivity of Capacity to Reservation Rate

well known that $\lambda = 1$ maximizes the function $\lambda e^{-\lambda}$, it is clear that $\lambda = 1$ is optimal whenever it is feasible. Thus the capacity of this ALOHA-based system is

$$C_\eta = \begin{cases} \dfrac{\eta}{e} & \eta \leqslant \dfrac{1}{1-e^{-1}} \\ -(\eta-1)\ln(1-1/\eta) & \eta > \dfrac{1}{1-e^{-1}} \end{cases}. \tag{5.5}$$

It is not hard to see that a minislotted CSMA protocol is one implementation of an ALOHA protocol fed by a binary reservation channel. Each binary reservation costs one propagation time, $\triangleq a$, on the channel so that the performance curves in Figure 5.3 must be scaled down by a factor of $\frac{1}{1+\eta a}$. Since there is considerable interest CSMA protocols for local networks, we will devote the following chapter to exploring the performance of a new CSMA protocol based on this binary reservation channel model.

### 5.4: Bounding Capacity if a Binary Channel Announces New Arrivals

Figure 5.3 allows us to compare the capacity as a function of $\eta$ when collisions on the message channel are resolved using the Gallager-Tsybakov algorithm and ALOHA. It is clear that ALOHA rapidly improves and gives better performance than the Gallager-Tsybakov algorithm when $\eta \gtrsim 3$. While this may seem counter-intuitive at first, the third curve on Figure 5.3 demonstrates an even stronger result. Below we extend the Tysbakov-Mikhailov bound on capacity for Poisson arrivals to the case of protocols that receive their input from a binary reservation channel at rate $\eta$.

Let us assume that we have bulk arrivals to the system, such that the number of messages in each bulk is drawn independently from a Poisson distribution with parameter $\lambda$, and that bulks of size zero may be distinguished immediately from bulks of size greater than zero at no cost to the protocol. We assume that the messages in each bulk arrive uniformly distributed over a set of measure $\lambda$, and that these sets are disjoint. Such a system is exactly equivalent to a genie-aided Poisson multiple access problem in which the genie partitions the time axis into disjoint sets of measure $\lambda$ and announces at no cost whether each such set is empty or contains some messages. Below we apply the proof technique introduced by Tsybakov and Mikhailov to find an upper bound, $C_\lambda$, for any infinite population protocol operating in such a system.

**Theorem 5.1:**

Define $h(q)$ as

$$h(q) = \sup_{a>0}\left\{ \frac{a\,(a+q)}{e^a - 1} + a\,e^{-a} \max\left\{0,\ 1 - \frac{a+q}{e^a-1}\right\}\right\}.$$

Then for any given $\lambda$,

$$C_\lambda \leqslant \min_q \frac{h(q)}{1-e^{-\lambda}+q}.$$

**Proof:**

Since we have already presented the proof technique of Tsybakov and Mikhailov in complete detail in Theorem 4.4 (in which we extended their proof to the case of Bernoulli arrivals) we will only sketch the changes required here to prove the new result. Recall that $D_t$ is the accepted set up to time $t$ (*i.e.*, the set of points known from the past history, $\Theta(t-1)$, not to contain any messages). The objective function is to be

$$\gamma_t \triangleq \text{mes}\, D_t + q \cdot s_t,$$

where $s_t$ is the number of successful transmissions up to time $t$ and $q$ is any real number. In this new model, Tsybakov's upper bound on the expected incremental increase in the objective at any step, $h(q)$, still holds. To complete the proof, it is necessary to find a lower bound on $E[\gamma_\tau]$, the value of the objective function when the protocol has completed processing all the arrival points, for this new system.

Let the measure of the set of arrival points to be processed by the protocol be $M$. Since this set is composed of $M/\lambda$ disjoint subsets, each known to contain one or more messages according to a conditional Poisson distribution with parameter $\lambda$, the expected number of messages in $M$ is given by

$$E[s_\tau] = \frac{M}{\lambda} \sum_{i=1}^{\infty} i \cdot \frac{\frac{\lambda^i}{i!} e^{-\lambda}}{1-e^{-\lambda}} = \frac{M}{1-e^{-\lambda}}. \tag{5.6}$$

Thus

$$E[\gamma_\tau] \geqslant M + q \cdot \frac{M}{1-e^{-\lambda}} = M\left(\frac{1+q-e^{-\lambda}}{1-e^{-\lambda}}\right). \tag{5.7}$$

Combining Eqs. (5.7) and (4.16), we obtain

$$M\left(\frac{1+q-e^{-\lambda}}{1-e^{-\lambda}}\right) \leqslant E[\gamma_\tau] \leqslant h(q) \cdot E[\tau]$$

and hence that

$$\frac{M}{h(q)}\left(\frac{1+q-e^{-\lambda}}{1-e^{-\lambda}}\right) \leqslant E[\tau]. \tag{5.8}$$

Combining Eqs. (5.6) and (5.8), it follows that for any protocol, $A$, and any $\lambda > 0$

$$\rho_{A,\lambda} \triangleq \lim_{M \to \infty} \frac{E[s_\tau]}{E[\tau]} \leqslant \frac{h(q)}{1-e^{-\lambda}+q}$$

must hold. Thus the channel capacity for the given $\lambda$, $C_\lambda$, must satisfy

$$C_\lambda \leqslant \min_q \frac{h(q)}{1-e^{-\lambda}+q},$$

giving the desired upper bound.

■

We have now derived an upper bound, $C_\lambda$, on the efficiency of arbitrary multiple access protocols (which operate entirely on the message channel) given a sequence of binary reservations from a Poisson source with parameter $\lambda$ as input. To plot this bound on Figure 5.3, it remains to find $\eta$, the required reservation rate, as a function of $\lambda$. But, from conservation of flow, the rate at which messages arrive on the reservation channel must be at least as large as the rate at which they are transmitted successfully on the message channel. Thus

$$C_\lambda \leqslant \eta \cdot (1-e^{-\lambda}) \cdot \frac{\lambda}{1-e^{-\lambda}}$$

or

$$\eta \geqslant \frac{C_\lambda}{\lambda}. \tag{5.9}$$

A plot of this bound in Figure 5.3 shows that for $\eta >> 1$, the capacity of ALOHA with binary reservations approaches this upper bound on capacity more quickly than the bound approaches unity.

**5.5: Summary**

We have introduced multiple access protocols that use partial reservation channels. We have shown that any infinite population multiple access protocol can operate entirely on a *ternary* reservation channel, yielding perfect utilization of the message channel when the reservation rate exceeds about two reservation requests per message transmission time. Thus a ternary partial reservation channel provides about as much useful information as a pure reservation channel in the infinite population case, so that studying ternary reservation channels provides little new insight into the multiple access problem.

A *binary* reservation channel would be much simpler to provide, particularly in a radio environment. Unfortunately, binary reservations do not provide enough feedback information for any infinite population protocol to operate: one can never be sure that a non-empty reservation slot corresponds to a *single* request without allowing the actual message transmission to take place. Thus some part of the collision resolution algorithm must take place on the message channel.

One feasible approach is to resolve collisions entirely on the message channel using some multiple access protocol. However, for high reservation rates, we found it to be more efficient to discard colliding messages than to resolve collisions with the best known multiple access protocol (*i.e.*, the Gallager-Tsybakov protocol). Indeed, the capacity of the simple loss system rapidly approaches our extension of the Tsybakov-Mikhailov upper bound results as the reservation rate grows large.

# CHAPTER 6
# Virtual Time CSMA

## 6.1: Introduction

In this chapter, we apply some of our results on partial reservation protocols from the previous chapter to develop a new class of CSMA protocols for local networks. Recall that in local networks, it is assumed that the propagation time across the network is much less than a message transmission time. Thus, multiple access protocols suitable for local networks can take advantage of the rapid feedback of the result of each scheduling decision. Depending on the communications medium, it may be possible to distinguish between an idle or busy channel (*i.e.*, the *carrier sense* environment), or even to discriminate between successful and interfering transmissions while they are in progress (*i.e.*, carrier sense with *collision detect*).

After devoting several previous chapters to the study of tree algorithms, which are known to have better performance characteristics than ALOHA, the reader may be wondering why we now intend to study a CSMA protocol in some detail. Many operational local networks employ CSMA protocols, including the Department of Defense PRnet packet radio network [Kahn78a], and the Ethernet coaxial cable local network [Metc76a]. Indeed, the IEEE is currently involved in an effort to standardize a protocol for local networks, and, among others, is considering a scheme based on the Ethernet implementation of 1-persistent CSMA. In addition, CSMA protocols can operate asynchronously on an *unslotted* channel, whereas all tree algorithms can only operate on a *slotted* channel. Since maintaining accurate slot synchronization in a distributed local network has proven to be a difficult problem, asynchronous protocols are often used in real networks. Thus, it seems clear that an improved CSMA protocol should be of some practical interest.

In addition, the characteristics of known tree algorithms make them unsuitable for many types of networks, especially those subject to random noise or whose stations may dynamically enter or leave the system. Massey has shown that many of the more efficient tree algorithms, including the Gallager-Tsybakov algorithm, can deadlock in the face of errors in the feedback information (e.g., when an idle slot is mistakenly interpreted as a collision). More significantly, this same deadlock can also occur if stations are ever allowed to discard messages that are waiting for retransmission after a previous collision. This situation could arise when a station left the network in the midst of a service epoch (possibly due to failure of the station or because the stations are mobile). A more serious problem is that of higher level protocols that permit stations to "give up" trying to transmit a message after some timeout has elapsed. The random retransmission strategy of CSMA protocols is insensitive to both errors in the feedback information and dropped messages. Finally, the current state of the system in many tree algorithms can depend on the channel history over an unbounded number of previous slots. Thus

for many tree algorithms, it is difficult for stations to join (or rejoin after an interruption) a working network. Consequently, we feel that tree algorithms are most suited for relatively noise-free channels (such as a coaxial cable or optical fibre). At the current state of the art, CSMA protocols seem better suited for packet radio networks. Thus, even from an academic point of view, we find many strong reasons for studying CSMA protocols *because of* their random times of retransmission.

While the ability to monitor channel activity can clearly be used to force stations to *refrain from transmitting* when another transmission is already in progress (and hence to reduce the number of collisions), it has not been clear how this information should best be used to decide when a ready station *should* begin transmitting a message. Several families of CSMA protocols were first analyzed by Tobagi and Kleinrock [Toba74a, Klei75c]. The analysis was later extended to include collision detection by several authors [Toba79a, Lam79a]. The *non-persistent CSMA* protocol permits ready stations to access the channel only if it is idle when the message arrives. If the channel is busy, the station acts as if a collision occurred and reexamines the channel only after a random delay. The *1-persistent CSMA* protocol permits ready stations to access the channel immediately if it is idle when the message arrives, or as soon as the channel becomes idle if it is busy. The *p-persistent CSMA* protocol also grants stations immediate access to the channel if it is idle when their messages arrive. However, if the channel is busy when a message arrives, the station waits until the channel first becomes idle and then is permitted to transmit only with probability $p$. If the channel remains idle for a further end-to-end propagation time, the ready stations again choose whether to begin transmitting with probability $p$, or to delay for another propagation time. If the channel subsequently becomes busy before the station begins transmitting, the station acts as if a collision occurred and repeats the algorithm after a random delay. Below we present a new algorithm for selecting transmission times using this same information in a way that is both efficient and fair. A fair algorithm prevents a subset of the stations from monopolizing the channel to achieve a better grade of service than the remaining stations. Our algorithm achieves fairness by granting stations permission to access the channel in the order in which their messages became available. The algorithm can also be easily extended to provide priorities when it is required.

In general, a multiple access protocol is an algorithm for partitioning the set of messages in a network until each message is sufficiently isolated to be transmitted without interference [Pipp81a, Moll80a]. Current CSMA protocols depend heavily on randomization to achieve this partitioning. When randomization is used, message delays are inherently highly variable and can be very long. Consequently, CSMA networks are unsuitable for such applications as packetized voice that are sensitive to both the mean and variance of delay. The intent of the virtual time CSMA protocol is to improve the delay characteristics by reducing the need for randomization. We achieve this goal by observing that the differences in *generation times* between messages are usually large enough for a protocol to distinguish between the messages. Our protocol operates in a way that preserves these differences, allowing messages to be transmitted FCFS after some queueing delay, so random rescheduling is only required in the rare event that several messages have arrived "too closely" in time.

## 6.2: The Virtual Time CSMA Contention Resolution Algorithm

Let each station be equipped with two "clocks", $C$ and $C'$, giving real time $\tau$ and virtual time $\tau'$, respectively. Initially, we assume both clocks begin at 0. The real time clock runs continuously at real time while the virtual time clock behaves as follows. Whenever the channel is sensed *busy*, the virtual time clock is disabled; whenever the channel is sensed *idle*, the virtual time clock is enabled (to run at some rate—see below). We consider both synchronized and unsynchronized systems. In the *unslotted* case, the virtual time clock runs continuously, whenever it is enabled, at a rate $\eta$ ($\eta > 1$) times real time if $\tau' < \tau$, or at real time otherwise. In the *minislotted* case.[1] the virtual time clock advances in discontinuous jumps. Stations must defer to channel activity that began in the previous minislot, so the minislot size (*i.e.*, the time between jumps) is chosen to be the worst-case propagation time, $a$, across the network. Each time it accumulates another $a$ units of *enabled* real time, the virtual clock advances by the minimum of a fixed quantum $\omega \triangleq a \cdot \eta$ and the instantaneous backlog $Q \triangleq \tau - \tau'$.

When each new message, $m$, is generated, its real generation time $\tau_m$ is recorded. We shall permit this message to be transmitted as soon as the virtual time clock first exceeds $\tau_m$. Thus $Q$ is proportional to the expected number of messages queued for access to the channel at any time, and the virtual clock provides a method for spreading out the transmissions of these messages. Each message will be delivered successfully unless some other transmission begins during the vulnerable period ($a$) before its signal has propagated to all parts of the network. Should a collision occur, all the affected messages are retransmitted after some random delays.

We note that the behaviour of the clock in the minislotted case is similar to the window mechanism defined in [Gall78a] as a FCFS way of granting permission to transmit to messages. Such mechanisms have the desirable property that the set of messages selected for transmission in any slot depends only on the current instantaneous backlog and can be freely chosen to optimize the efficiency of the protocol. Furthermore, since the operation of the protocol is otherwise independent of the details of the past activity on the channel, both the analysis and implementation of the protocol are simplified.

Figure 6.1 shows an example of the clock readings and channel activity during the operation of the minislotted virtual time CSMA protocol. It is assumed that the propagation time $a$ is 1/7 of the transmission time, that there is no collision detection, and that the virtual clock runs at twice real time. The real clock reading is shown as a line of unit slope since it advances continuously at a rate of 1 second per second. Activity on the channel is shown as a series of blocks extending above the real clock reading. Each block represents a transmission attempt; those that result in a collision are shaded. The virtual clock reading is shown as an irregular "staircase" line at a vertical distance $Q$ below the real clock reading. While the channel is idle, the virtual clock waits for time $a$, then advances by $\min\{Q, 2a\}$. While the channel

[1] We shall follow [Klei77a] in calling these short time intervals *minislots* rather than *slots* as defined in [Toba74a, Klei75c]. This avoids some confusion with the meaning of *slots* in the slotted ALOHA protocol [Robe72a], where it is assumed that all slots are large enough to accommodate the transmission of a complete message.

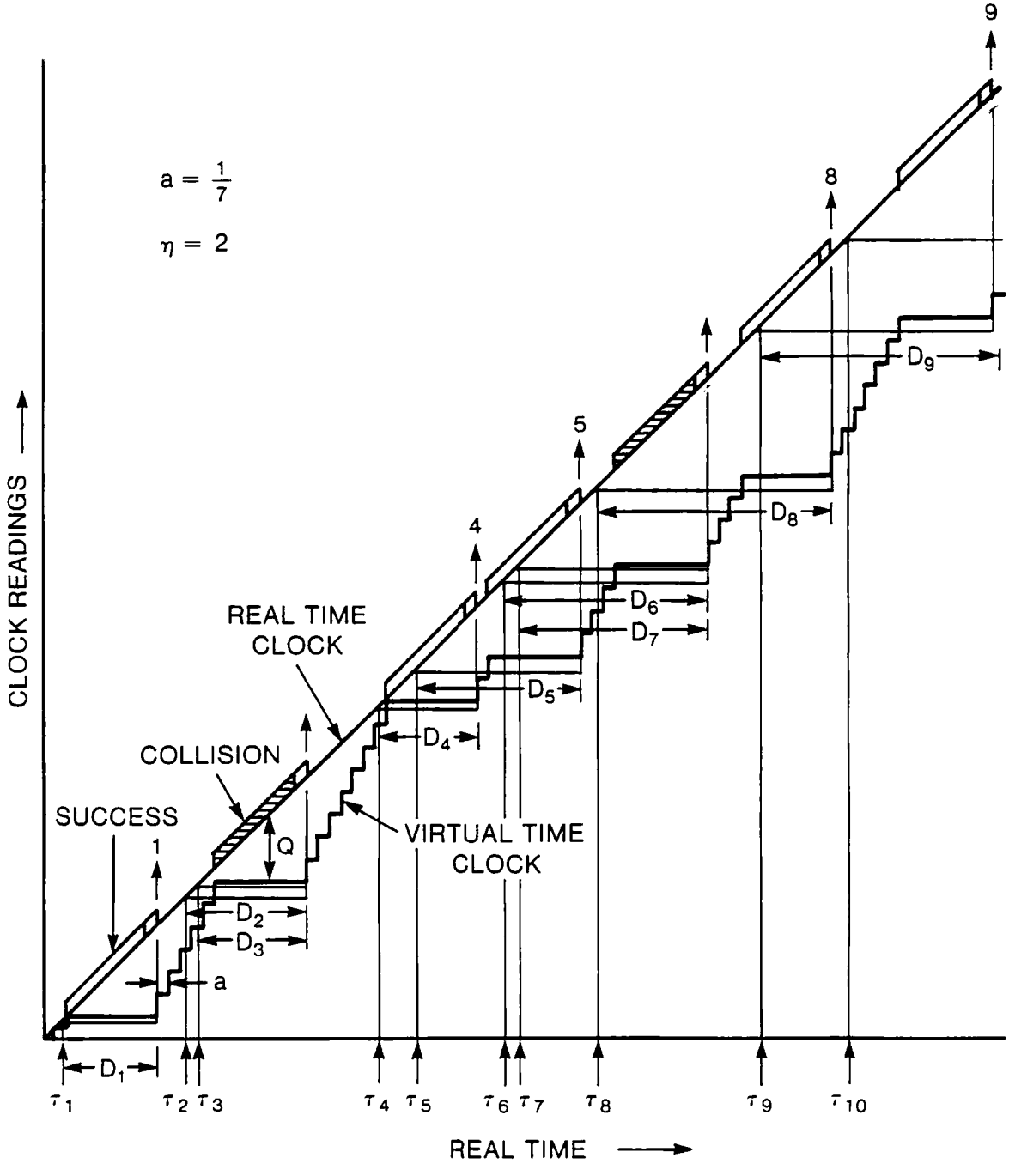


Figure 6.1: The Operation of Minislotted Virtual Time CSMA

is busy, the virtual clock remains stopped. Messages arrive to the system at some time $\tau_i$, $i=1,2,\cdots$, begin transmission when the virtual clock passes $\tau_i$, and finally leave the network (after a total time in system of $D_i$) when the transmission has been completely received at their destination after a further time of $1+a$. When a collision occurs, we assume that the messages are either lost or retransmitted after some random delay.

Given this brief description of the protocol, we must now calculate its performance to permit meaningful comparisons with other CSMA variants. Let us assume that the number of stations in the network is very large. We may approximate such a network as an infinite number of stations generating new messages according to a Poisson process with intensity $S$, and look at the "pseudoequilibrium" throughput and delay curves. (Recall that we assume the transmission time for messages to be unity.) We note that this requires the "bold" Poisson assumption [Mass80a] that the total offered load on the network is a Poisson process with intensity $G$, $G \geqslant S$. This is certainly true in a *loss system*, where messages are destroyed if they collide. Here the arrivals consist entirely of new messages (not all of which will be transmitted successfully), so that $G=S>\rho$. When we resolve collisions with the aid of some randomized retransmission algorithm, we assume that all messages will be transmitted successfully (possibly after many unsuccessful attempts) so that $G>S=\rho$. In this latter case, the Poisson traffic assumption is not completely correct. Nevertheless, the Poisson assumption is common in estimating the performance of protocols, including this one. However, it will become clear that our protocol distorts the arrival process much less than was the case with previous protocols where such an analysis was performed [Klei75c, Robe72a, Abra70a, Klei75b]. We note that if the arrivals on the real time axis were Poisson with intensity $G$, then the actual message transmission times recorded by the real time clock would still form a Poisson process that generates arrivals *only* when the channel is sensed idle. It would have intensity $G$ if the virtual clock had caught up to the real time clock, and $\eta G$ if it had fallen behind.

We begin by finding the capacity of both the minislotted and unslotted protocols. Let propagation across the network require a fraction $a<1$ of a message transmission time. Whenever a collision occurs, let the sending stations stop transmitting after a fraction $b \leqslant 1$ of each message has been transmitted. We assume the 'arbitrary destination' model from Chapter 2, so that collisions occupy the channel for a total time of $a+b$.[1] As we approach capacity, the system will almost always have a backlog. Therefore for this capacity calculation we may assume that the arrival rate is always $G' \triangleq \eta G$. For any value of $G'$, we can find the pseudoequilibrium throughput from renewal theory as the probability that a random observation along the time line intersects a successful transmission. The probability that we observe a particular type of period is proportional to the product of its relative length and relative frequency of occurrence.

In the *unslotted* case, we view the time axis as a renewal process with idle periods alternating with busy periods. By definition a busy period begins when some station first begins transmitting a message over an idle channel. In this capacity calculation, we may assume that each idle period has an average length of $1/G'$. (Recall that the virtual clock speed is always $\eta$

[1] Since we will not consider the 'central destination' model from Chapter 2 in this chapter or the following chapter, we shall drop the '' for notational simplicity.

for systems operating at maximum capacity.) A busy period will be a success if and only if no other message transmission begins within the vulnerable period ($a$) before the other stations have detected the start of the first transmission. Thus, each busy period will be a success with probability $e^{-aG'}$ or a collision with probability $1-e^{-aG'}$. In calculating the length of a busy period, we shall assume the worst case [Klei75c], namely that each transmission occupies the channel for a duration of $1+a$. Thus the length of each busy period having a successful transmission is assumed to be $1+a$. For each busy period having a collision, we must also account for the *lag* between the starting times of the first and last messages involved in that collision to begin transmission (see [Klei75c]). Let $\bar{Y}$ be the expected value of this lag time (in end-to-end propagation times). Then the expected length of a busy period having a collision is assumed to be $a(1+\bar{Y})+b$. It remains to find $\bar{Y}$.

Let $Y$ be the random variable representing this lag time. Since stations are forbidden to begin any new transmissions once the channel is sensed busy, it must be the case in every collision that $0 \leqslant Y \leqslant 1$. If we again make the worst-case assumption [Klei75c] that all other stations in the network will remain free to begin a new transmission for a full end-to-end propagation time after some station first begins transmitting a message (thereby ending an idle period) then the distribution of starting times over this vulnerable period will be conditionally Poisson with intensity $G'$. Thus

$$Pr[Y \geqslant y] = 1 - \frac{e^{aG'\cdot y} - 1}{e^{aG'} - 1} = \frac{e^{aG'} - e^{aG'\cdot y}}{e^{aG'} - 1}.$$

Since $\bar{Y} \triangleq E[Y] = \int_0^1 Pr[Y \geqslant y]\,dy$, we finally obtain

$$\bar{Y} = \frac{1-(1-e^{-aG'})/aG'}{1-e^{-aG'}}.$$

When $b=1$, this result for $\bar{Y}$ clearly reduces to Eq. (7) in [Klei75c] if we are careful to note that in [Klei75c], $\bar{Y}$ was averaged over *all* busy periods (including successes where its value must be zero), and that message slots rather than propagation times were used as the time unit. Thus, in the unslotted case, the throughput equation becomes

$$\rho = \frac{G'e^{-aG'}}{G'[2a+b+(1-b)e^{-aG'}]+e^{-aG'}}. \tag{6.1}$$

In the *minislotted* case, time is divided into a series of intervals of varying lengths, each of which is independently idle, a success, or a collision with probability $e^{-aG'}$, $aG'e^{-aG'}$ and $1-(1+aG')e^{-aG'}$, respectively. Since each idle interval has duration $a$, each success has duration $1+a$ and each collision has duration $b+a$, we may immediately write down the throughput in the minislotted case as

$$\rho = \frac{aG'e^{-aG'}}{b(1-e^{-aG'})+a+aG'e^{-aG'}(1-b)}. \tag{6.2}$$

Setting $b$=1 in Eqs. (6.1-6.2), these throughput equations become identical to the throughput equations for non-persistent CSMA without collision detect [Klei75c]. Thus virtual time CSMA must have the same theoretical capacity as non-persistent CSMA. This is not unexpected, since both analyses assume that there is a Poisson arrival process that is disabled whenever the channel is sensed busy. In fact, since the only difference between non-persistent CSMA and virtual time CSMA is the method by which the arrivals are disabled when the channel is busy, it should be clear that both protocols could be used on the same network. Below we show that a message is much more likely to be successfully transmitted on any attempt with virtual time CSMA than with non-persistent CSMA. It follows that we can only improve the stability and delay characteristics of a network by replacing some stations using non-persistent CSMA with an equal number of stations using virtual time CSMA.

Consider the example from [Klei75c] of a system with $a$=.01, $b$=1. Optimizing Eqs. (6.1-6.2) over $G'$, we find a capacity of $\approx$.815 in the unslotted case when $G' \approx 9.45$, and capacity of $\approx$.8655 in the minislotted case[1] when $G' \approx 13.45$. Thus, for the minislotted virtual time CSMA protocol operating at capacity, the probability that a message is successfully transmitted at any attempt (which we assume to be independent of the number of attempts) is $e^{-aG'} \approx .874$, giving $\approx 1.144$ as the expected number of attempts per message. Slotted non-persistent CSMA enjoys the same probability of success whenever a message is transmitted. However, should the channel be sensed busy when a station wishes to transmit a message, the message is not transmitted and immediately suffers a further random delay before another attempt is allowed. The probability that a message is transmitted at a random attempt for slotted non-persistent CSMA is

$$\frac{a}{1 - e^{-aG'} + a},$$

which declines to only $\approx$.0736 at capacity, giving $\approx 15.5$ as the expected number of attempts per message. In this example virtual time CSMA reduced the expected number of attempts per message by a factor of $\approx 13.6$ at capacity (compared to slotted non-persistent CSMA) by allowing queueing. Consequently, if the new arrivals were to form a Poisson process, the total offered load with virtual time CSMA would still be approximately Poisson, since each message need "arrive" only about 1.144 times. This is probably not the case with slotted non-persistent CSMA, however, since at capacity each message circulates an average of 15.5 times before being successfully transmitted.

### 6.3: Delay Characteristics

Let us now examine two models for estimating the mean queueing delay. The first model is easy to solve but does not follow the detailed system behaviour exactly. The second, more detailed model is used to validate the first model. We shall restrict our analysis to the minislotted case for mathematical convenience. We shall also fix $\eta$ to a value chosen to achieve maximum capacity. Since we calculate maximum capacity by optimizing over $G' \triangleq \eta G$,

[1] Under these same conditions, the capacity of slotted non-persistent CSMA was stated as .857 [Toba74a, Klei75c]. However, a more precise optimization of the same throughput equation shows that its capacity is really .8655.

$\eta$ cannot be uniquely determined without invoking some additional boundary conditions. Recall that to attain maximum capacity, the virtual clock must be run *only* at a rate $\eta$. However, the virtual clock runs only during those periods in which the channel is sensed idle by the stations. Thus, we choose $\eta$ so that on the average, the virtual clock can just keep up to real time.

We view the "initial propagation time" at the start of each slot as a binary reservation process, through which the collision resolution algorithm learns that either no station or at least one station has requested to transmit a message. Such reservations share the channel in the time domain with successful transmissions and collisions: each non-empty "reservation" is followed immediately by the transmission of the corresponding messages before the reservation process is allowed to continue.

For our first model, let us make the approximation that there is a separate subchannel in the frequency domain for reservations. Reservations and message transmissions are uncoupled. The reservation channel merely determines the arrival process to the message channel. This arrival process consists of a sequence of equally spaced independent Bernoulli trials, each producing a reservation with probability $p \triangleq 1 - e^{-aG'}$. We account for the reservation overhead by assuming that this subchannel utilizes the same time-frequency product as was required by the original time division reservations at capacity. Thus, if the original model achieved capacity with an arrival rate $aG^*$, this approximation requires that the reservation rate be $1/p^* = (1 - e^{-aG^*})^{-1}$ Bernoulli trials per message slot, and that the reservation overhead reduces message channel throughput by a factor of $\frac{1}{1+a/p^*}$.

This first model may be described as the following queueing system. Non-empty reservations arrive independently and join a queue to gain access to the message channel. The transmission time for the message(s) represented by each reservation contributes to the busy time of the message channel (*i.e.*, the utilization $p/p^*$ of the queueing system). However, each reservation independently represents a success or a collision with probability $aG'e^{-aG'}/p$ and $1 - aG'e^{-aG'}/p$, respectively. Should it represent a success, the transmission time for that single message will contribute to the channel throughput. Should it represent a collision, all messages will still be transmitted, but there will be no contribution to the channel throughput. Thus the throughput on the message channel is given by

$$\rho \triangleq \min\{p/p^*, 1\} \cdot aG'e^{-aG'}/p = \min\{1/p^*, 1/p\} \cdot aG'e^{-aG'},$$

which will in general be less than the busy time on the message channel. We note, however, that message delays depend directly on the busy time on the channel and not the throughput.

Since we have assumed that the probability of success on any transmission attempt is identical, let us focus on an individual transmit attempt. If we require the transmission times on the message channel for both successful transmissions and collisions to be evenly divisible by the transmission time for a reservation on the reservation subchannel, then this first model corresponds exactly to a *discrete time* M/G/1 queue. This model gives an *exact* expression for the delay in a variation of minislotted virtual time CSMA in which we require virtual clock to

advance *only* in steps of size $\omega$, thereby delaying each "tick" of the virtual time clock for a period of $\max\{0, \omega - Q\}$. In addition, this model gives a lower bound on the mean delay for the more conventional model of virtual time CSMA described above.

In the Appendix, we show that the average time in system in such a queue, normalized by the mean service time, is given by

$$\frac{T}{\bar{x}} = 1 + \frac{\rho(C_b^2 + 1) - p}{2(1-\rho)} \tag{6.3}$$

where $\rho$ is the utilization of the queueing model (*i.e.*, the fraction of time that the channel is not idle), and $C_b^2$ is the coefficient of variation of the service time. Eq. (6.3) has a particularly simple form when there is no early collision detection. In that case $C_b^2 = 0$ since all reservations require the same service time — be they successful transmissions or collisions. If early collision detection is possible, then let $n_s$ and $n_c$ to be the number of minislots to service a success or collision, and let $b_s$ and $b_c$ be the probabilities of a reservation representing a success or a collision, respectively. Then letting $\bar{x} = b_c \cdot n_c + b_s \cdot n_s$, we obtain

$$C_b^2 = p \frac{b_c b_s (n_s - n_c)^2}{\bar{x}},$$

and the relevant time in system becomes $T/n_s$.

Figure 6.2 shows the delay for a single transmission attempt as a function of throughput for two models when $a$=.01 and $b$=1. Since we have already shown that $G^* \approx 13.45$ for this choice of parameters, we find that $1/p^* \approx 7.95$. We are thus exceedingly fortunate in that rounding $1/p^*$ to an integral number (*i.e.*, 8) of reservation slots per message slot introduces almost no error in our model. (It follows that the protocol should use $\omega \approx 1/8$, and thus $\eta = \omega/a \approx 12$ — see model two, below.)

These same values for $a$ and $b$ have previously appeared in numerical examples by Tobagi and Kleinrock [Toba74a, Klei75c]. It is assumed that all messages will eventually be transmitted successfully, so that $S = \rho$ and $G/S$ is the average number of transmission attempts per message. While it would be helpful to compare the total delay for this model with the total delays presented in those previous analyses, their chosen values for the retransmission delay parameters are not available for the comparison. Since $D \approx (G/S - 1)R + 1$, we were able to use Figures 11 and 12 of [Klei75c] to obtain $R \lesssim G/S$ as a first order approximation to $D$. With this admittedly crude approximation, our total delay curve should be $D \approx T \cdot G/S$. (Recall that with these parameters, $G/S \lesssim 1.144$ even at capacity. Since the backlog must exceed one transmission time after a collision, we may absorb the randomization time into the queueing time.) We thus expect that our total delay curve differs only slightly from our per attempt delay curve when we account for retransmissions. Our Figure 6.3 reproduces Figure 12 from [Klei75c] with this approximate analytic curve added for comparison. The apparent improvement in delay performance over all previously considered CSMA protocols is significant.

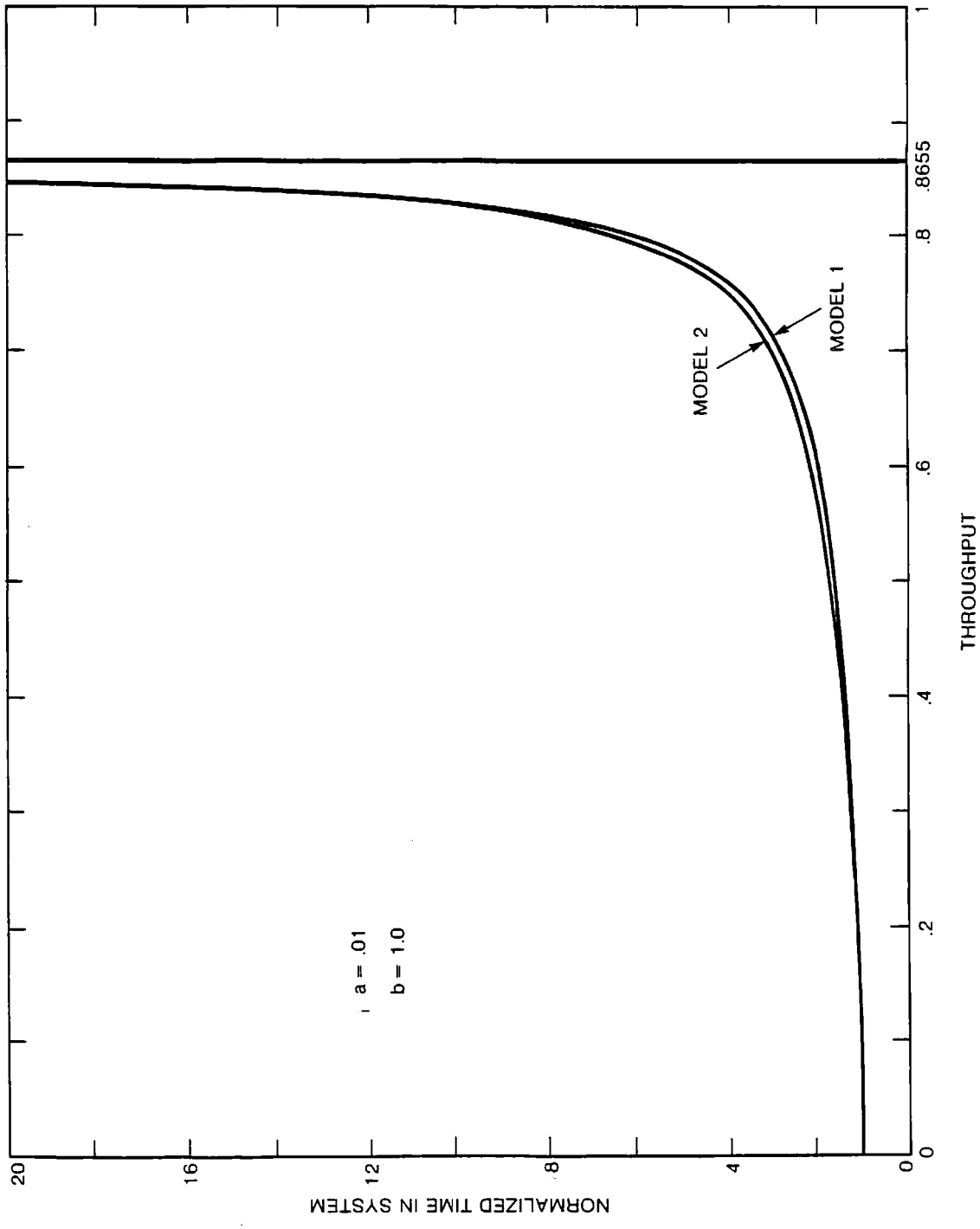


Figure 6.2: Time In System per Transmission Attempt

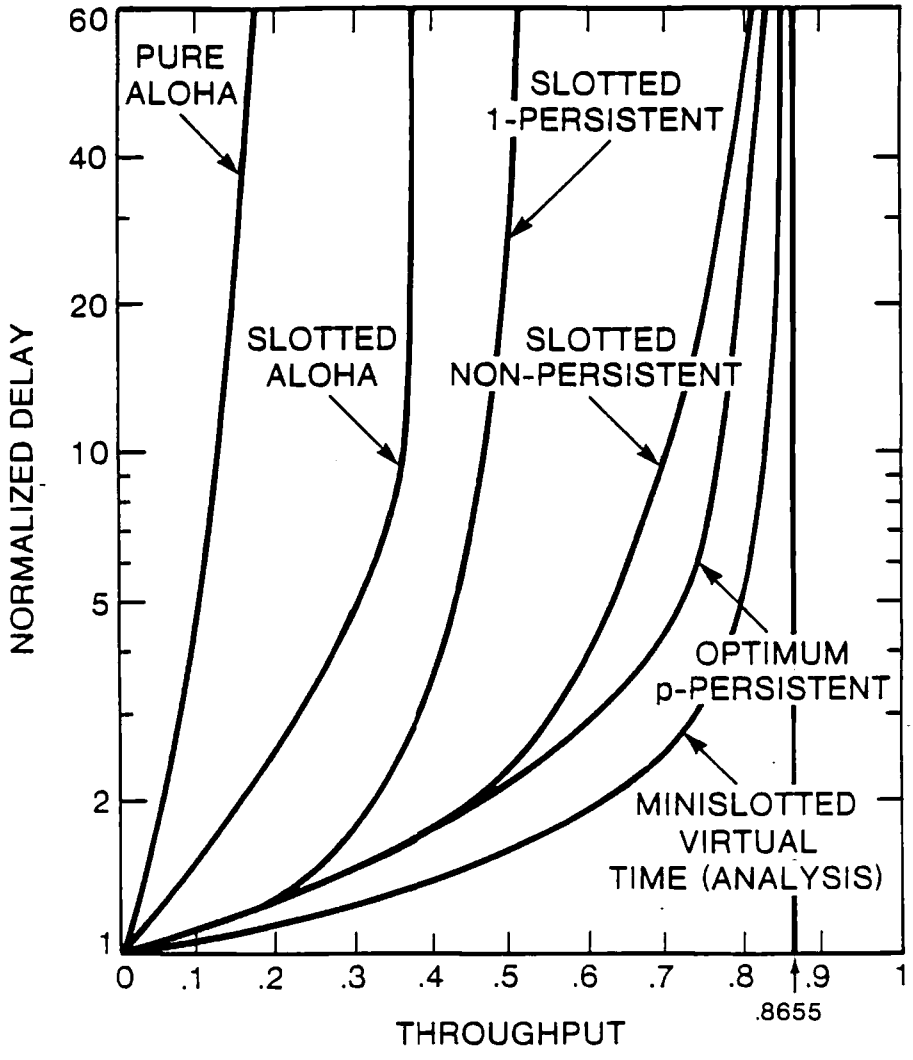


Figure 6.3: Comparison of Throughput-Delay Tradeoffs
with Existing CSMA Protocols

For our second model, we attempted a direct solution of the discrete time random walk over the instantaneous backlog $Q_t$ (measured in units of one transmission slot) at the *end* of the $t^{th}$ slot. We found this approach manageable for a carefully chosen set of system parameters. In particular, let $a = .01$, let each transmission occupy the channel for one unit of time (of which .01 is the propagation time overhead), and let $\omega = .12 \approx 1/8$. Then not only do we have a discrete state random walk (since all allowable values of $Q$ are multiples of .01), but the set of reachable states is actually very small. Let the initial state be $Q_0 = .01$. If an interval of length .01 is idle (with probability $p_{0,0} \triangleq e^{-aG}$), the next state will again be .01 since one interval of length .01 was "served" over a period of .01. If it is busy (with probability $p_{0,9} = 1 - p_{0,0}$), its service lasts a period of 1.00, increasing $Q$ by .99. If an interval of length $\omega$ is idle (with probability $p_- \triangleq e^{-\omega G}$), $Q$ must decrease by .11 since an interval of length .12 was served over a period of .01. If it is busy (with probability $p_+ = 1 - p_-$), $Q$ increases by $1.00 - .12 = .88$. It should be clear that the backlog can only take on values equal to $.11k + .01$ for some $k = 0, 1, \cdots$. We may thus reduce our state space by a factor of 11 and change variables to obtain the following equations of motion:

$$\pi_k = \begin{cases} p_{0,0}\pi_0 + p_-\pi_1 & k = 0 \\ p_-\pi_{k+1} & 1 \leqslant k \leqslant 8 \\ p_-\pi_{10} + p_{0,9}\pi_0 + p_+\pi_1 & k = 9 \\ p_-\pi_{k+1} + p_+\pi_{k-8} & k \geqslant 10 \end{cases}$$

where $\pi_k$ is the probability that the backlog is $.11k + .01$. Taking the $z$-transform of these balance equations, we find the transform of the equilibrium distribution of $\pi$ to be

$$\Pi(z) \triangleq \sum_{i=0}^{\infty} \pi_i z^i = \pi_0 \left[ 1 + \frac{p_{0,9} z^2}{p_+} \sum_{i=1}^{\infty} \left[ \frac{1 - z^9}{1 - z} p_+ \right]^i \right],$$

where

$$\pi_0 = \frac{1 - 9p_+}{1 - 9p_+ + 9p_{0,9}}.$$

Unfortunately, we were unable to invert this transform and thus could only find $\{\pi_k\}$ numerically. This can easily be done, however, since $\pi_0$ has a closed form solution, and the remaining $\{\pi_k\}$ may be solved as a triangular set from the balance equations.

Given $\{\pi_k\}$, it remains to find the system throughput, $\rho$, and mean time in system, $T$, using renewal theoretic arguments. Define the normalization constant $K$ to be

$$K \triangleq .01 + .99[\pi_0 p_{0,9} + (1 - \pi_0) p_+].$$

Then the throughput is given by

$$\rho = \frac{.99}{K} \left[ \pi_0 \frac{aGe^{-aG}}{1 - e^{-aG}} + (1 - \pi_0) \frac{\omega G e^{-\omega G}}{1 - e^{-\omega G}} \right].$$

The expected delay may also be found from $\{\pi_k\}$, although we were unable to obtain a closed-form solution. We now find it convenient to return to our original measure of backlog. Recall that $\pi_k$ gives the relative frequencies of finding a backlog of $.11k+.01$ at the end of a slot. Consider the conditional expected queueing delay for a message that arrived in a slot ending with a backlog of $.11k+.01$. If that slot was idle, it must have had a duration of .01 so our message must have arrived during the last .01 of the backlog. Since $\omega=.12$, our message suffers a queueing delay of $\lfloor .11k/.12 \rfloor \cdot \bar{x}$, where $\bar{x} \triangleq .01+.99p_+$ is the average service time. If the slot was busy, our message could have arrived anywhere over the last 1.00 units of the backlog. Its queueing delay will thus be somewhere in the range

$$\left\lfloor \frac{.11k+.03}{.12} - 8 \right\rfloor \cdot \bar{x} \leqslant w_k \Big|_{busy} \leqslant \left\lfloor \frac{.11k}{.12} \right\rfloor \cdot \bar{x},$$

which we will approximate as $\approx (11k/12-4)\bar{x}$. Thus, the normalized time in system (measured in transmission times) is given by

$$\frac{T}{\bar{x}} = 1 + \frac{1}{K} \sum_{k=0}^{\infty} w_k \pi_k,$$

where:

$$w_0 = .005 \cdot (p_{0,0}\pi_0 + p_-\pi_1)$$

$$w_k = p_-\pi_{k+1} \cdot \left( .005 + \left\lfloor \frac{.11k}{.12} \right\rfloor \bar{x} \right) \qquad 1 \leqslant k \leqslant 8$$

$$w_9 = (p_{0,9}\pi_0 + p_+\pi_1) \cdot \left( .5 + \left\lfloor \frac{.11k}{.12} - 4 \right\rfloor \bar{x} \right) + p_-\pi_{10} \cdot \left( .005 + \left\lfloor \frac{11}{.12} \right\rfloor \bar{x} \right)$$

$$w_k = (p_+\pi_{k-8}) \cdot \left( .5 + \left\lfloor \frac{.11k}{.12} - 4 \right\rfloor \bar{x} \right) + p_-\pi_{k+1} \cdot \left( .005 + \left\lfloor \frac{.11k}{.12} \right\rfloor \bar{x} \right) \qquad 10 \leqslant k$$

The mean delay for a single transmission attempt as a function of throughput for this second model is also plotted on Figure 6.2. We note a surprisingly good agreement with the previous, much simpler delay model. We thus feel confident that the "separate reservation channel" model gives us a meaningful picture of the performance of the minislotted virtual time CSMA protocol.

### 6.4: Simulation Results

A simple finite population simulation model with 50 stations was used to test the validity of the above analysis. The time parameters and retransmission strategy were chosen to be consistent with Tobagi's numerical calculation of the equilibrium throughput-delay curve for slotted non-persistent CSMA presented in Figure 6-2 of [Toba74a]. In both models, it is assumed that when collisions occur, each message is scheduled for retransmission after a further time of $K \cdot a$ according to a geometric distribution with parameter $\nu$. In Tobagi's model, each station either has no message and is in a "thinking" state or has one message and is in an "active" state. The arrival of a message activates a thinking station: the station transmits a new message immediately, and thereafter attempts to retransmit its message in each successive

minislot with probability $\nu$ until it is transmitted successfully. In our model, the minislotted virtual time CSMA protocol controls access to the channel at each station. We say that a message is *activated* when the virtual clock passes its time of arrival, and *re-activated* when the virtual clock passes the time at which it is scheduled for retransmission following a collision. Each station transmits a newly-activated message immediately; and thereafter attempts to retransmit the message each time it is re-activated. In this model, we assume that the virtual time clocks shall continue to run at each station during the period in which a message is waiting to be re-activated. However, we shall permit each station to have only one message at a time that is either activated or waiting to be re-activated, so that a few newly activated messages may be blocked. (It follows that at most one message per station can be scheduled for transmission in a single minislot.)

The results for both protocols are shown in Figure 6.4. However, they are not directly comparable. Tobagi's model assumed that the stations were unbuffered, so that all new arrivals to stations already having an undelivered message are blocked. Since any message that would suffer queueing delays is lost, the delay performance of his model is somewhat optimistic. Our model permits any number of messages to be queued within a station, but still exhibits some blocking. In our model, any newly activated message must be blocked if the station already has a message waiting to be re-activated. We thus expect our delay predictions to be more realistic than Tobagi's but still optimistic.

Tobagi's results for slotted non-persistent CSMA considered a range of values for the retransmission parameter, $\nu$. Our model used a constant value of $\nu$ just small enough for the system to achieve maximum throughput with stability when infinite queues are permitted at each station [Tsyb80a]. This value is obtained by observing that 50 blocked stations is effectively an "infinite" number as far as convergence to the limiting value of stable capacity is concerned. When there is a backlog (which is very likely at capacity), our protocol grants permission to all messages whose scheduled transmission times fall within a window of length $\omega$ (recall that $\omega \triangleq a \cdot \eta$). For stability, the expected number of arrivals in that window should be close to the optimum Poisson load in the infinite population case when all stations have a message waiting for retransmission. Thus, for $a = .01$ and $\omega = .12$, and recalling from §6.2 that for maximum throughput we choose $aG' \approx .1345$, we require $50 \cdot 12 \cdot \nu \approx .1345$ for stability. Consequently, we chose $\nu = .0003$ for our simulation. We observe that our simulation points fall closely in line with Tobagi's $\nu = 1/100$ curve. While this is good evidence that our protocol does not *hurt* performance, we must point out that Tobagi's chosen values of $\nu$ are too small for stability. Recall that the series $(1 - 1/N)^{N-1}$, $N = 1, 2, \cdots$, converges rapidly to $1/e \approx .368 \cdots$. Thus, when $\nu = 1/50$ and all stations are attempting to retransmit a message, the probability of any minislot being idle is $(1 - 1/50)^{50} \approx 1/e$, and the probability of any minislot initiating a successful transmission is $(1 - 1/50)^{49} \approx 1/e$. Similarly, when $\nu = 1/100$, the probabilities of an idle minislot or a success are $(1 - 1/100)^{50} \approx 1/\sqrt{e}$ and $(50/100) \cdot (1 - 1/100)^{49} \approx 1/(2 \cdot \sqrt{e})$, respectively. Thus, even if we were to ignore the time spent in idle minislots, the stable throughput (when we permit queueing) is still limited to only

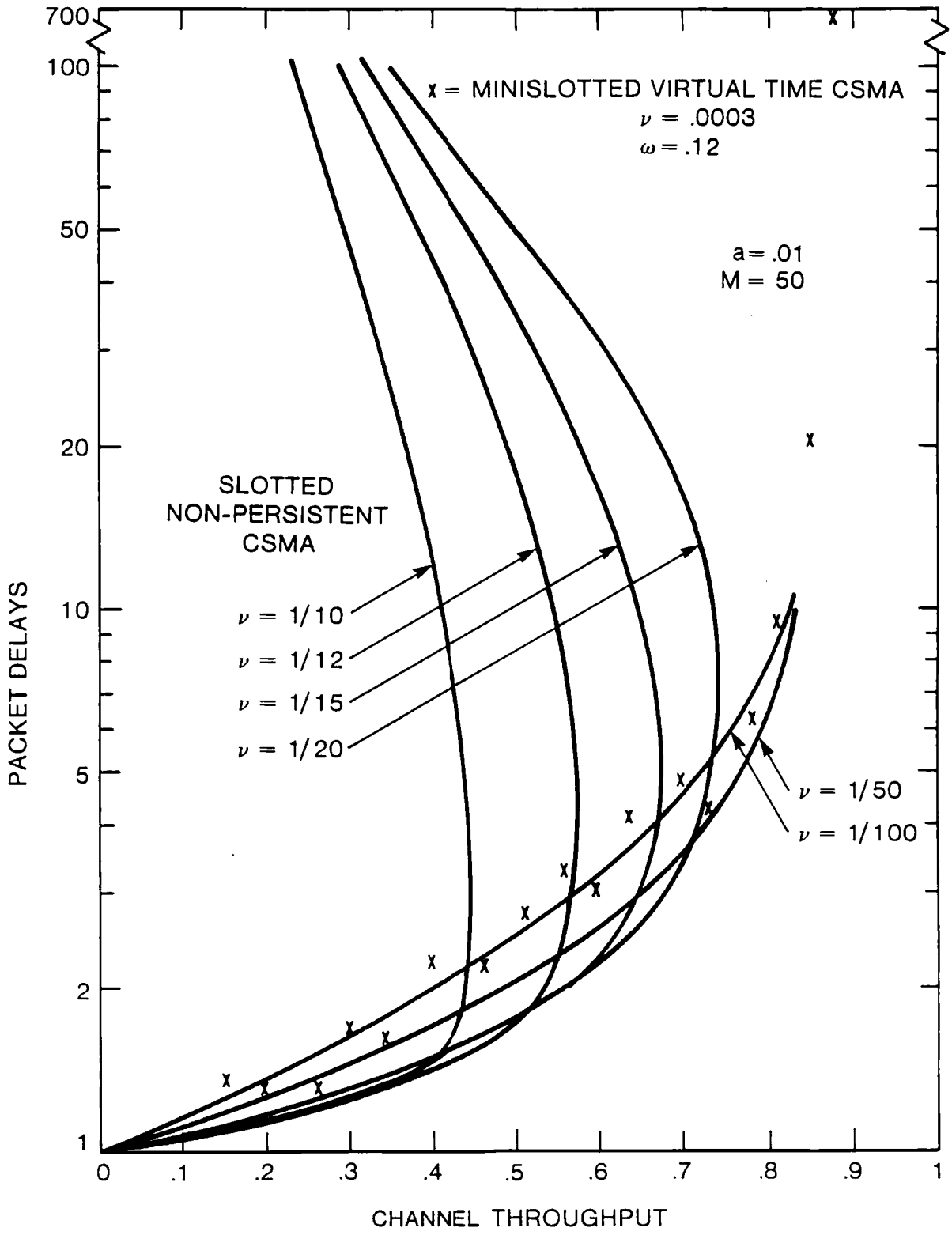


Figure 6.4: Finite Population Model Performance

$$\frac{1/e}{1-1/e} \approx .58$$

with $\nu = 1/50$, and only

$$\frac{1/(2\cdot\sqrt{e})}{1-1/\sqrt{e}} \approx .77$$

with $\nu = 1/100$. Maximum stable throughput is achieved only for $\nu \approx .1345/50 \approx 1/372$, which would probably give somewhat higher delays than $\nu = 1/100$. To summarize, our simulation of minislotted virtual time CSMA achieved a similar delay performance to Tobagi's $\nu = 1/100$ curve while doubly handicapped because it used a smaller value of $\nu$ and it allowed queueing of messages within a station (with some probability of blocking) which was prohibited in Tobagi's model. We therefore conclude that there is again some evidence that minislotted virtual time CSMA gives better delay performance than slotted non-persistent CSMA.

Some recent results by Marathe [Mara80a] provide some indirect evidence of performance gains with virtual time CSMA protocols. As part of a simulation study of Ethernet-like networks, he compared several different "backoff" algorithms for randomly rescheduling messages after a collision. In all cases, access to the channel at each attempt was controlled by the standard 1-persistent CSMA protocol used on the Ethernet [Metc76a], so his results are not directly applicable to virtual time CSMA. However, we can get some useful information by comparing the performance of the standard Ethernet algorithm with a proposed "stop backoff" algorithm. Both of these are variations on the binary exponential backoff algorithm in which retransmissions are uniformly distributed over an interval whose length increases exponentially as a function of the number of previous (unsuccessful) attempts to transmit the same message. However, the latter algorithm requires each waiting station to stop its backoff timer should some other station begin a conflict-free transmission immediately. The remaining waiting time is to be spent at the end of the successful transmission. This strategy approximates using a virtual clock to control the transmission times of retransmissions but not of new messages.

Marathe's comparison of backoff algorithms examined short messages (*i.e.*, a "timesharing" workload) and long messages under both light and heavy loads ($\rho = .4$ and $\rho = .7$, respectively). With the timesharing workload, the performance of the two algorithms was indistinguishable under light load. Under heavy load, the stop backoff algorithm reduced the average service time by 9%, reduced the 90 percentile of service time by 10%, reduced the 90 percentile of the number of attempts from 4 to 3, and reduced the percentage of packets aborted after 16 unsuccessful attempts from 0.2 to 0.0. With long messages, the performance of the two algorithms was almost the same except for the percentage of packets aborted. This percentage dropped from 0.9 to 0.0 under light load, and from 5.3 to 0.0 under heavy load.

Marathe's results for an approximation to virtual time CSMA are encouraging for two reasons. First, unlike the standard algorithm, the stop backoff algorithm always delivered the messages successfully within 16 tries, while the standard algorithm aborted a significant number of "problem" messages. No penalty was associated with aborting messages [Mara81a]; it was assumed that each message would be retransmitted instantaneously if it was aborted. Thus, we expect the performance statistics for the standard algorithm to be optimistic. Since the standard

algorithm still did not provide better service, an unbiased comparison would show that the stop backoff algorithm was even better. Second, our analysis of virtual time CSMA ignored the specific details of the retransmission backoff algorithm. As long as the backoff algorithm adequately spreads out retransmission attempts, we showed that our virtual clock mechanism can significantly improve the performance of the network. In particular, we showed it that works extremely well if messages are aborted after one unsuccessful attempt. These simulation results showed that adding a virtual clock mechanism to control a backoff algorithm did not degrade any of the performance measures. We therefore believe that the virtual time CSMA protocols will continue to show excellent performance characteristics when we account for the retransmission backoff algorithm in detail.

### 6.5: Optimality Conditions for Virtual Time CSMA

Actual operating conditions are exceedingly difficult to model accurately. Consequently, most analysts make a number of strong assumptions and consider the behaviour of protocols only under these somewhat idealized conditions. While it may be argued that some common assumptions are unrealistic, such analyses have nevertheless led to many interesting results. Below we prove the optimality of the class of minislotted virtual time CSMA protocols, given some common (but perhaps questionable) assumptions. It follows that no "better" protocol can be found unless a more realistic set of assumptions is considered.

**Theorem 6.1**

*Assume:*

A1. The protocol supports an infinite population of homogeneous stations.

A2. Each message is immediately and irretrievably lost after its first transmission attempt on the channel, whether or not it was successful. (This strong assumption makes the total traffic a stationary Poisson process, and is equivalent to selecting a uniform retransmission density over the interval $(t, \infty)$.)

A3. No explicit protocol messages may be exchanged over the channel: stations may only announce that they have a message to send by actually transmitting it.

A4. All transmissions are synchronized to start at the beginning of a minislot, *i.e.*, each transmission starts at a time $K \cdot a$, $K = 0,1, \cdots$, where $a$ is the normalized propagation time between stations in the network.

*Then* for any given values of $a > 0$ and $1 \geqslant b > 0$:

1. No protocol can achieve a higher capacity than the best virtual time CSMA protocol, which we obtain by optimizing $\eta$ as a function of $a$ and $b$ (*i.e.*, virtual time CSMA is both locally and globally optimal under these assumptions).

2. For any feasible value of throughput, there exists a virtual time CSMA protocol that

achieves the lowest probability of loss of any protocol.

3. Given any constraint on the maximum probability of loss for messages in each slot not exceeding the loss probability at maximum capacity, there exists a virtual time CSMA protocol that achieves the lowest average delay for unblocked messages at all feasible values of throughput.

4. The optimal virtual time CSMA protocol described immediately above also achieves the lowest *variance* of delay for unblocked messages over all protocols that achieve the lowest *average* delay.

**Proof of 1:**

By A1, the protocol can ignore the identities of the stations (the expected number of messages at a station must be zero for the average message delay to remain finite) and merely concentrate on the messages themselves. The only available discrimination between messages is their times of arrival, so the problem reduces to partitioning a Poisson arrival process. By A2, the history of past activity on the channel provides no useful information about the arrival times of the remaining messages in the system other than an estimate of $G$, which we assume is given. Thus, using A3, the set of arrival points selected by any feasible algorithm for transmission in, say, the $i^{th}$ minislot must be independent of the actual distribution of messages over the enabled set so that the number of messages transmitted in any minislot must have a Poisson distribution with some parameter $\lambda$. Using a renewal argument, the conditional average throughput over all slots where the protocol chooses a particular value $\lambda = \lambda_0$ is given by

$$\rho_{\lambda_0} \triangleq \frac{\lambda_0 e^{-\lambda_0}}{b(1-e^{-\lambda_0}) + a + \lambda_0 e^{-\lambda_0}(1-b)}. \tag{6.4}$$

The probability of loss for a message chosen at random from those transmitted in a slot where $\lambda = \lambda_0$ is given by

$$B = B(\lambda_0) = 1 - e^{-\lambda_0}.$$

Hence $e^{-\lambda_0} = 1-B$ and $\lambda_0 = -\ln(1-B)$. We may thus rewrite Eq. (6.4) to give $\rho$ as a function of $B$:

$$\rho = \frac{-\ln(1-B)\cdot(1-B)}{b\cdot B + a - \ln(1-B)\cdot(1-B)\cdot(1-b)}. \tag{6.5}$$

A real valued function $f$ over a convex set $X \subset R^n$ is said to be *strongly quasiconvex* if [Avri76a]

$$f(q_1 x^1 + q_2 x^2) < \max[f(x^1), f(x^2)]$$

for any two points $x^1 \in X$, $x^2 \in X$, $x^1 \neq x^2$ and for any positive weights $q_1 > 0$, $q_2 > 0$, $q_1 + q_2 = 1$. A function $f$ is *strongly quasiconcave* if $-f$ is strongly quasiconvex. It is also easy to show that a non-negative function $f$ is strongly quasiconvex if and only if $1/f$ is strongly quasiconcave. A strongly quasiconcave function $f$ defined on a convex set $X \subset R^n$ attains its

maximum on $X$ at no more than one point. If we can show that $\rho$ is a strongly quasiconcave function of $B$ over the closed convex set $0 \leqslant B \leqslant 1$ then since $\rho$ is continuous and bounded to at most 1, $\rho$ must attain that maximum for some value $B^*$. Thus to show $\rho$ is strongly quasiconcave, is suffices to show

$$1 - b + \frac{b \cdot B + a}{-\ln(1-B)\cdot(1-B)}$$

is strongly quasiconvex. Since adding a constant has no effect on the strong quasiconvexity of a function, and a positive linear combination of strictly convex functions is strictly convex and thus strongly quasiconvex, it suffices to show that both

$$\frac{1}{-\ln(1-B)\cdot(1-B)}$$

and

$$\frac{B}{-\ln(1-B)\cdot(1-B)}$$

are strictly convex. Taking the second derivative of each with respect to $B$, we obtain

$$\frac{-\ln(1-B) + 2[1+\ln(1-B)]^2}{[-\ln(1-B)\cdot(1-B)]^3}$$

and

$$\frac{-B\ln(1-B) - 2(1+\ln(1-B))[-\ln(1-B)\cdot(1-B) - B(1+ln(1-B))]}{[-\ln(1-B)\cdot(1-B)]^3}$$

$$= \frac{\ln(1-B)\cdot(B-2) + 2(B-1) + 2[1+\ln(1-B)]^2}{[-\ln(1-B)\cdot(1-B)]^3},$$

which are clearly positive for $0 < B < 1$ since $\ln(1-B) < 0$.

We have thus established that there is a unique value $B^*$ for which $\rho$ achieves its global maximum $\rho^*$. Consequently, no protocol can achieve a higher throughput than any protocol that always chooses $\lambda^*$. Whenever there is sufficient backlog (as would almost always be the case as we approach capacity), minislotted virtual time CSMA always chooses some fixed quantum $\omega$. If the actual arrival rate is $G$ per unit time, $G \geqslant \lambda^*$, then the virtual time CSMA protocol with $\omega = \frac{\lambda^*}{G}$ achieves maximum capacity, possibly with infinite delays if $G$ is too large.

■

**Proof of 2:**

If we can now show that $\rho$ is a strictly concave function of $B$ over the convex set $0 \leqslant B \leqslant B^*$, we will have proven that for any average value of $\rho \leqslant \rho_{MAX}$, the average value of $B$ is minimized by any protocol that chooses the *same* blocking probability (and hence the same probability of success) at *every* slot.

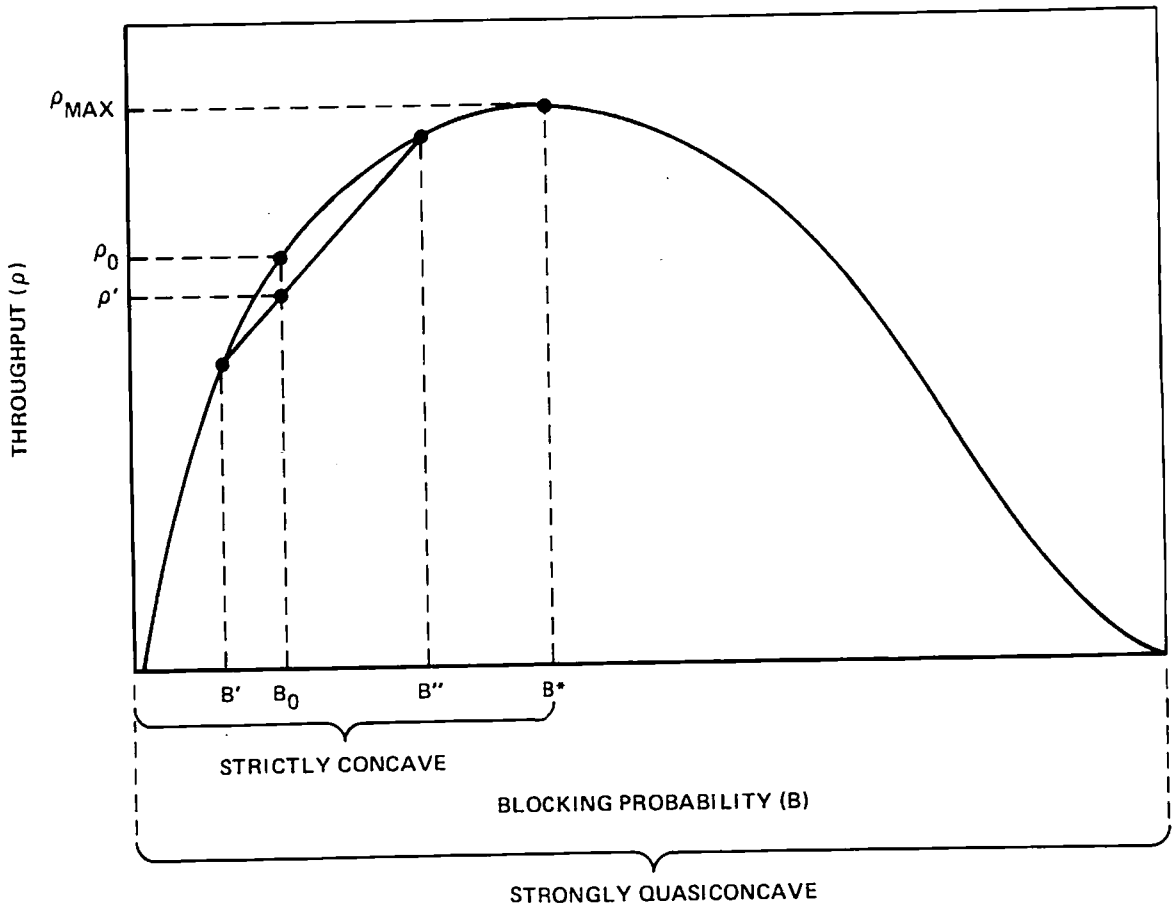


Figure 6.5: Throughput vs. Blocking Probability in Virtual Time CSMA

If we write Eq. (6.5) as $\rho = \frac{\alpha}{\beta}$, where $\alpha \triangleq -\ln(1-B)\cdot(1-B)$ and $\beta \triangleq b\cdot B + a - \ln(1-B)\cdot(1-B)$ are functions of $B$, then

$$\rho' \triangleq \frac{\alpha'\beta - \alpha\beta'}{\beta^2} \geqslant 0$$

for $0 \leqslant B \leqslant B^*$, and

$$\rho'' \triangleq \frac{[\alpha''\beta - \alpha\beta'']\beta^2 - [\alpha'\beta - \alpha\beta']2\beta\beta'}{\beta^4}.$$

Since $\alpha'' \leqslant \beta'' \leqslant 0$ and $\beta \geqslant \alpha \geqslant 0$, $[\alpha''\beta - \alpha\beta'']\beta^2$ is non-positive. Since $\rho' \geqslant 0$ and since $\beta' < 0$ implies $\alpha' \leqslant \beta' \leqslant 0$, it must be the case that $\beta' \geqslant 0$. Thus $[\alpha'\beta - \alpha\beta']2\beta\beta'$ is non-negative, so

that $\rho'' \leqslant 0$ and hence $\rho$ is strictly concave for all $B \leqslant B^*$. Consequently, for any feasible $\rho$, $B$ is minimized by a virtual time CSMA protocol whose *capacity* is $\rho$.

■

**Proof of 3:**

Applying Little's result [Litt61a] to busy periods (*i.e.*, those intervals of time where the backlog $Q$ remains positive) we observe that no other feasible protocol can have a higher throughput (since $\rho$ is an *increasing* function of $B$ for $B \leqslant B^*$) or a lesser number customers in the system that will be served successfully (since it is a *decreasing* function of $B$), and hence that no other protocol can achieve a lower average delay.

■

**Proof of 4:**

This result follows directly from proposition 3 above, since it can be shown [King62a] that FCFS minimizes the variance of delays for G/G/1 queues.

■

We may apply these results to the various known slotted CSMA protocols to predict their relative capacities. We have already shown that both slotted non-persistent and minislotted virtual time CSMA allow the same Poisson sample size to be selected in every slot at capacity, and hence achieve maximum capacity by optimizing $G$. Since slotted 1-persistent CSMA selects a sample of size $aG$ if the previous slot was idle and a sample of size $(1+a)G$ if it was busy, it is clear that slotted 1-persistent CSMA must have a lower capacity than either of the previous two protocols for any finite value of $a$.

The behaviour of $p$-persistent CSMA with arbitrary $p$ is rather complex. In the first minislot following a busy slot, the Poisson sample size is $p \cdot G(1+a)$. If that is idle, the next sample size is $p(1-p) \cdot G(1+a) + p \cdot aG$, and so on. This relation allows us to easily calculate the optimum $p$ that maximizes the capacity of $p$-persistent CSMA by equating the Poisson sample sizes in two consecutive minislots (the first of which must be idle), giving

$$p \cdot G(1+a) = p(1-p) \cdot G(1+a) + p \cdot aG$$

or

$$p = \frac{a}{1+a}.$$

Substituting this value for $p$ into the derivation for the throughput of $p$-persistent CSMA from [Klei75c] gives us Eq. (6.2). Thus optimum $p$-persistent CSMA protocol can achieve optimum capacity, but only when $b = 1$, *i.e.*, when there is no collision detection. We note, however, that its delay performance is inferior even to slotted non-persistent CSMA. Where slotted non-persistent CSMA immediately discards the set of messages that arrives during a busy slot (which is Poissonly distributed with parameter $G$), $p$-persistent CSMA discards the set of messages still "tossing coins" at the end of an idle period — which, for the optimum $p$, is again

Poisson with parameter $G$. Consequently, these lost messages suffer an additional fixed delay of an idle period compared to slotted non-persistent CSMA. Furthermore, the set of unblocked messages is transmitted in random order, rather than FCFS, increasing the variance of the delay.

Figure 6.6 shows the sensitivity of capacity to the propagation time, $a$, when $b=1$ for the various CSMA protocols. Similar results were shown in Figure 10 of [Klei75c], but the curve for slotted non-persistent CSMA was shown incorrectly. As predicted in Theorem 1, it forms an upper bound on performance at all values of $a$. In addition, we show the corresponding curves for both slotted and unslotted ALOHA. Since stations are allowed to transmit to any other station, we note that the capacity of slotted ALOHA should be $1/((1+a)e)$, which decreases to $1/2e$ at $a=1$. Thus the capacity of minislotted virtual time CSMA clearly exceeds the capacity of slotted ALOHA for all $a$ if $b \leqslant 1$ (which is clearly possible by avoiding collision detection if necessary) since the denominator of Eq. (6.2) is less than $1+a$. We note that since minislotted virtual time CSMA is locally optimal in a loss system, Figure 2.1 shows its sensitivity of capacity to both $a$ and $b$.

It is worth noting that, unlike minislotted virtual time CSMA, unslotted virtual time CSMA is by no means an optimal protocol. For example, unslotted 1-persistent CSMA enjoys a higher capacity when $a \gtrsim .2$. The reason for this is evident from examining the behaviour of 1-persistent CSMA. Under heavy load, a significant fraction of the messages arrive when the channel is busy, and are thus transmitted as soon as the channel goes idle. Since this event occurs all at once according to the model [Klei75c], there will be a momentary "spike" in the traffic intensity at the end of each transmission. These traffic spikes will tend to produce trains of partially synchronized packets with little intervening idle time, thereby increasing the efficiency of the protocol. Thus a good "unslotted" CSMA protocol will gather the messages to produce a periodicity in the arrival process, with the minislotted case occurring in the limit.

### 6.6: Synchronism, Variable Rate Clocks and Priorities

Throughout our analysis, we have assumed that the virtual time clocks at every station remain completely synchronized (or, in the unslotted case, that they at least reach the same virtual time reading within a propagation time). This global synchronism guarantees the fairness property of the protocol, *i.e.*, the FCFS transmission of messages. Since the expected queueing delay for a message depends only on the *difference* between the readings of the real and virtual time clocks when it arrives, it is easy to see that *absolute* synchronism is not required. Fairness still holds if for each pair of stations there is a *constant* difference in both their real and virtual time readings. Thus new stations can easily join a working network since accurate synchronization of the clocks at every station is not required. Indeed each "clock" can be implemented as a modulo $k$ counter.

The virtual clock speed-up rate, $\eta$, has been chosen to be a constant whenever the virtual clock is running behind real time. We have shown in Section 6.5, above, that this strategy minimizes the mean delay for unblocked messages in a loss system subject to a constraint on the *maximum* blocking probability in any slot. It is conceivable that the performance of the

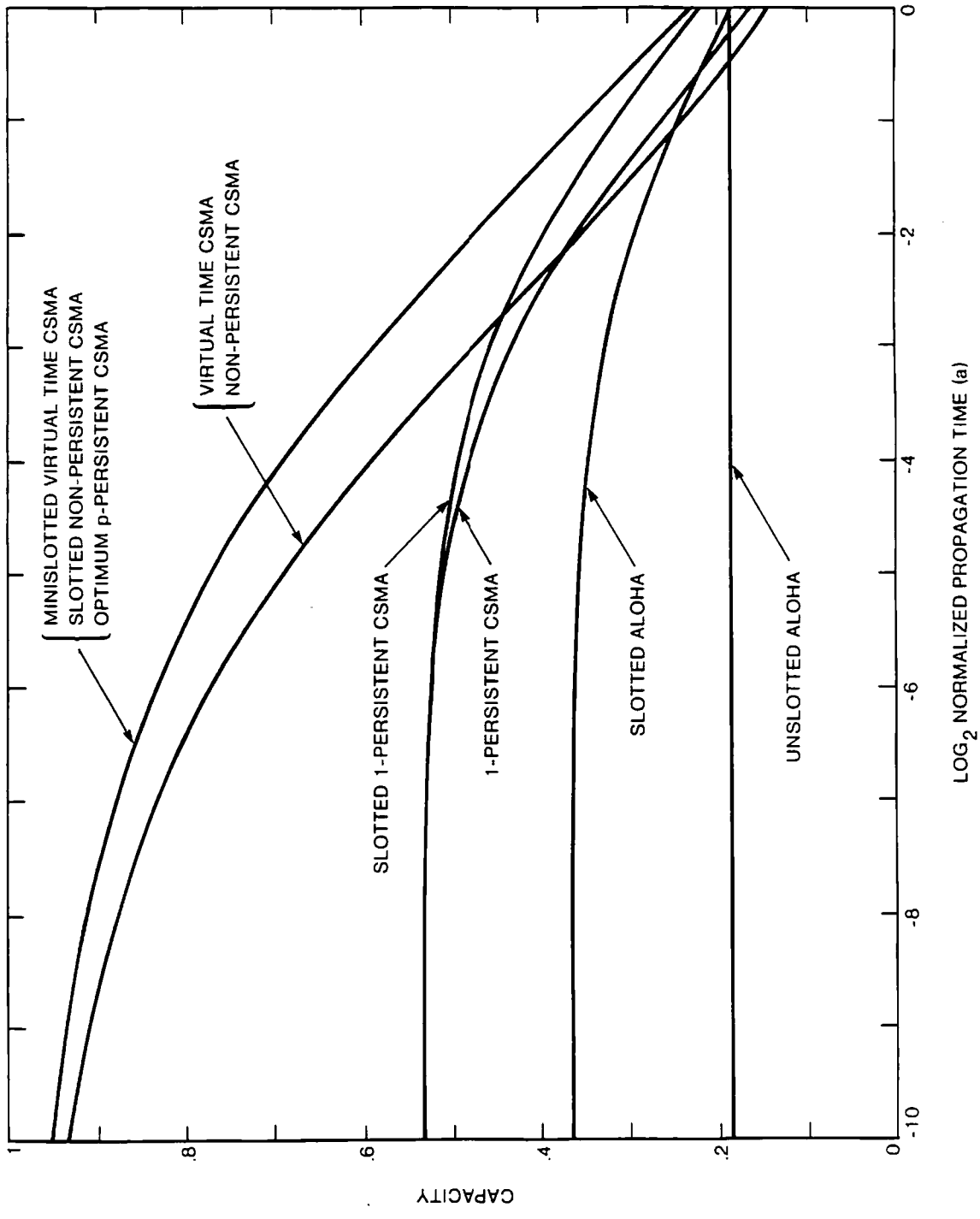


Figure 6.6: Sensitivity of Capacity to Propagation Time

system under light traffic could be improved by gradually reducing $\eta$ to unity as the virtual clock "catches up" to real time. Such a strategy could reduce the instantaneous blocking probability significantly during that transient period — an important consideration when blocked messages must be retransmitted after a random delay. However, we found in our analysis that systems operating below capacity already spent most of their time without a backlog (where the value of $\eta$ is ignored). Consequently, we conjecture that such an extension would offer little real improvement.

Below, we investigate a further property of practical significance. With some loss of fairness, virtual time CSMA protocols can operate with variable rate, imprecise clocks. In fact, there is but one strong requirement for accurate time keeping in virtual time CSMA, and this requirement occurs only in minislotted virtual time CSMA. In every synchronous protocol, all stations must be able to determine the *time* at which each (mini)slot begins with considerable accuracy. Thus, in minislotted virtual time CSMA, synchronism of the times at which the virtual clocks "tick" is critical, but the amount by which the clocks advance at each "tick" is far less important. Since the sum of Poisson streams is always Poisson, the major effect of using imprecise clocks would be to introduce some variability in the apparent offered load. A secondary effect will be to permute the order in which messages were transmitted on the channel. Thus a *priority* mechanism could be provided by varying the virtual clock speed-up functions at each station according to load or to priority class.

Individual, load-dependent speed-up functions, $\eta_i(Q_i)$, could be used to provide a priority mechanism that adjusts the *relative* priority of stations as a function of the current backlog. These "load sensitive" priorities would behave like the time-dependent priorities described by Kleinrock (see § 3.7 in [Klei76a]).

Alternately, one may wish to give *absolute* priority to one class of messages so that, say, class $k$ messages can only be transmitted when there is no backlog of higher priority messages (of class $j$, $j < k$). As a special case, stations can be separated into different priority classes by assuming that all messages generated at a particular station belong to the same priority class. This type of priority mechanism is an example of head-of-the-line priorities [Cobh54a, Klei76a].

Tobagi [Toba80b] has extended the $p$-persistent CSMA protocol to provide this type of absolute priority mechanism. In $p$-persistent P-CSMA, a *priority assessment period* begins whenever "end of carrier" (signifying the completion of a message transmission) is detected on the channel. If no higher priority classes respond, members of the $i^{th}$ priority class are given the opportunity to reserve the channel by transmitting a short burst of energy in the $i^{th}$ *reservation slot* following end of carrier. If no class responds, the system reverts to normal $p$-persistent CSMA until the next transmission occurs. The length of a reservation slot is assumed to exceed $2a$, so that every reservation burst will be received within the reservation slot in which it was transmitted. Once the active priority class has been determined, there is a *channel access period* in which active members of that class apply the $p$-persistent algorithm to determine the start of the *transmission period.* The end of the transmission period provides the end of carrier signal for repeating the algorithm.

In addition to the basic *non-preemptive* algorithm described above, various forms of preemption were also considered in [Toba80b]. In the *semi-preemptive* case, members of higher priority classes can join a channel access period. Both the *partial preemptive* and *fully preemptive* cases include semi-preemptive and also permit members of higher priority classes to interfere with ongoing transmissions of lower priority. In the partial preemptive case, it is assumed that unsuccessful transmissions will be aborted through collision detection. In the fully preemptive case, there is no collision detection, so that unsuccessful transmission will always run to completion.

Tobagi's priority mechanism adds considerable complexity to the $p$-persistent CSMA protocol. The priority assessment periods also add channel overhead that reduces the capacity of the protocol, especially when there are many priority classes. As Tobagi has already suggested [Toba80b], a hierarchical search for the highest active priority class can reduce the priority assessment overhead. Below, however, we show that the HOL priorities can be incorporated into virtual time CSMA by using multiple virtual time clocks. The resulting priority mechanism is similar to Tobagi's semi-preemptive discipline. The fairness property can be preserved within each priority class. Little complexity is added to the basic virtual time CSMA algorithm, and *no* additional channel overhead is required for priority assessment. In fact, with careful tuning of system parameters, there need not be any increase in the overall *mean* message delay — only the *variance* of delay must increase.

We extend the virtual time CSMA protocol to provide an absolute priority mechanism between classes as follows. Let the actual traffic intensity generated by class $i$ be $G_i$. As we have shown in Section 6.5 above, there exists an optimum traffic intensity $G^*(B)$ that provides both maximum capacity and minimum mean delay for unblocked messages given any constraint on the maximum blocking probability per slot, $B$. (It follows that $G \triangleq \sum_i G_i$ must be strictly less than $G^*$ for stability.)

In this prioritized case, we now let each station be equipped with a separate virtual time clock for each priority class. We again assume that the real time clock runs continuously and that the virtual time clock(s) are enabled to run only when the channel is sensed idle. Recall that in the basic virtual time CSMA protocol, the virtual clock speed-up rate, $\eta$, may be thought of as a function of the instantaneous backlog $Q \triangleq \tau - \tau'$ such that $\eta(Q) = G^*/G$ when $Q > 0$ and unity otherwise. To implement the priority mechanism, we now let $\eta_i$, the virtual clock speed-up rate for class $i$, be a function of both $Q_{i-1}$ and $Q_i$, and *independent* of all lower priority classes. More precisely, we let

$$\eta_i(Q_{i-1}, Q_i) = \begin{cases} 0 & Q_{i-1} > 0 \\ \dfrac{G^* - \sum_{j=1}^{i-1} G_j}{G_i} & Q_{i-1} = 0,\ Q_i > 0. \\ 1 & Q_i = 0 \end{cases} \tag{6.6}$$

The behaviour of the set of virtual time clocks can be visualized as a "staggered start". When end of carrier occurs, the virtual clock for the highest priority class starts running at a rate $G^*/G_1$ and all other virtual clocks remain stopped. This grants permission to transmit exclusively to the highest priority class, in FCFS order, at the maximum rate that can be achieved without violating the maximum blocking constraint. If there are no messages from this class in the system, this first virtual clock will catch up to real time and be forced to slow down. As the virtual clock for the $i^{th}$ priority class catches up to real time, the virtual clock for the $i+1^{st}$ priority class is started at a rate just high enough to bring the traffic intensity back up to $G^*$. Messages from the higher classes continue to have *immediate* access to the channel as additional classes gain access to the channel (hence its similarity to Tobagi's semi-preemptive discipline [Toba80b]). The process continues until either some station begins transmitting a message (and the algorithm must be repeated from the beginning when this transmission is over) or else all virtual clocks have caught up to real time. It is clear from the above description that this prioritized virtual time CSMA protocol still generates arrivals at a rate $G^*$ when there is a backlog and $G$ when there is no backlog, so that the addition of priorities has only permuted the order in which messages are transmitted. From Kingman [King62a], we know that this permutation has no effect on the mean delay but that the variance of delay must increase.

It is worth noting that this method of using multiple virtual clocks to provide HOL priorities extends to more general classes of infinite population protocols. As is the case with prioritized virtual time CSMA, the introduction of priority classes need not introduce any additional channel overhead. In particular, we may extend to the prioritized case any infinite population tree algorithm that uses intervals of the arrival time axis as their enabled sets, such as the Gallager-Tsybakov algorithm.

As a practical consideration in the implementation of virtual time CSMA protocols with HOL priorities, we note that only one virtual clock per station is required, even with many priority classes. It is clear from Eq. (6.6) that at any time, the virtual clock for at most *one* priority class is in active use. We say that a virtual clock is in active use when it is running faster than real time to provide some flow control for that class. The virtual clocks for every other class merely act as a "gate" which is either completely open (for higher priority classes) or completely closed (for lower priority classes). It is thus sufficient to provide each station with a single virtual clock plus both a gate and a register (to store the current virtual clock reading) for each priority class.

A station in such a network operates as follows. Initially, all gates are open and the virtual clock is assigned to the lowest priority class and running at real time. Whenever the channel changes state from idle to busy, the current virtual time reading is stored in the register for the active class, the registers for all higher priority classes are set to real time, and all gates are closed. Whenever the channel changes state from busy to idle, the virtual clock is loaded from the register from highest priority class and allowed to run. If it catches up to real time before the channel becomes busy, the gate for the active priority class is opened and the virtual clock is reloaded from the register from the next highest priority class and allowed to run, and so on. The above algorithm seems simple enough to be considered for incorporation in a large-scale

integrated circuit.

Another property of practical importance is that stations generating only high priority messages can ignore the lower priority classes entirely — even the *number* of priority classes is unimportant. This is very fortunate for networks that support heterogeneous user classes. One case of particular interest is the mix of many small high priority stations (e.g., interactive terminals or real time instrumentation) with a few large stations (engaged, say, in file transfers, video, or faximile). Because of the transparency of lower priority classes, we can provide each small station with an inexpensive "dumb" network interface, reserving the sophisticated prioritized interfaces for the large stations, without degrading the performance of the network.

# CHAPTER 7
## Hybrid Carrier Sense — Binary Search Protocols

### 7.1: Introduction

When we interpret the "initial propagation time" at the start of each transmission as a binary reservation request, we see that CSMA protocols are a special case of loss protocols with binary reservations. This observation led to the discovery and analysis of the virtual time CSMA protocol, which we examined in considerable detail in Chapter 6. Having shown that no protocol can operate entirely on the binary *reservation* channel, proven that loss protocols are about as efficient (in terms of capacity) as is theoretically possible for any protocol that operates on the *message* channel, and found an optimal synchronous loss protocol (virtual time CSMA), it is tempting to believe that virtual time CSMA is an (almost) optimal protocol for local networks with many stations in which stations can sense carrier but not detect collisions (*i.e.*, packet radio systems). Both the extension of our genie-aided upper bound argument to the case of non-constant slot lengths in § 4.2.3 and Humblet's similar extension of his information theoretic approach [Humb80a] seem to support this claim. However, both of these results are valid *only* when stations are required to transmit only *complete* messages.

When the idle detect time is significantly less than the collision detect time, it is no longer obvious that we suffer no penality in capacity by considering only protocols that transmit only complete messages. When we view sensing carrier as a binary reservation process, it is perhaps easier to see that *hybrid* protocols that use *both* the message channel and the reservation channel must also be considered. Thus, in the context of local ($a \ll 1$) networks, we must also consider protocols in which stations are allowed to transmit either complete messages or short, carefully timed bursts of energy on the channel merely to announce that they have become ready to send a message.

The idea of separating the problem of locating a message from its successful transmission dates back to polling algorithms. Scholl [Scho76a] was one of the first to present a distributed version of polling without central control in his MSAP protocol. The BRAM protocol, independently proposed by Chlamtac [Chla79a], is very similar in concept. Both MSAP and BRAM operate by performing a *linear* search through a *finite* number of stations to locate the next message to be transmitted. Transmissions are synchronized to begin at the start of a minislot. This provides binary feedback to all stations within a propagation time showing whether that station was idle or busy. With these protocols, the overhead in finding messages grows linearly with the number of stations in the network, so that they are unsuitable for networks with a large number of stations.

Another related protocol was devised by Hayes [Haye78a]. In this protocol, ready stations are found by conducting a binary search procedure called *probing*. During the search, ready stations respond to a probe by transmitting a short burst (rather than a complete message). A ready station is not permitted to transmit a complete message until it has been uniquely identified by a sequence of probes. In [Haye78a], it was assumed that the algorithm was centrally controlled. Mark [Mark80a] later proposed a distributed version of this same protocol. It is assumed that the number of stations, $M$, is finite so that the protocols require $\log_2(M)$ probes to find the next ready station. Thus the performance of these protocols degrades as the logarithm of the number of stations as the population increases so that they, too, are infeasible with an infinite number of stations. Furthermore, even in the finite population case each station is required to know both the exact *number* of stations and the *address* of every station in the network. Efficient performance depends on careful assignment of station addresses.

Below, we present new hybrid infinite population multiple access protocols in which stations transmit both binary reservation requests (*i.e.*, energy bursts) and messages to gain access to the channel. This hybrid approach can be applied to CSMA protocols without collision detection to simultaneously reduce the idle time between transmissions and the blocking probability. It can also be used to extend the class of infinite population tree algorithms discussed in Chapter 3. We show that these hybrid tree algorithms can achieve a higher capacity than the hybrid CSMA protocols. Moreover, we show that hybrid tree algorithms can achieve channel utilizations far in excess of any "naive" extensions of the work on upper bounds on capacity to the carrier sense environment, such as the results described in § 4.2.3. Thus these hybrid protocols represent a fundamentally new class of protocols, specifically designed to take advantage of a binary feedback channel.

### 7.2: Efficient Use of Binary Feedback to Reduce Message Location Overhead

It is common in multiple access protocols to scan *linearly* through some candidate set in search of messages. While this is clearly true of such simple protocols as TDMA and ALOHA, it also occurs in tree algorithms. In the Gallager-Tsybakov algorithm, for example, each service epoch begins by selecting an enabled set having the *same* expected number of messages.

When the ratio of idle-detect to collision-detect times is close to unity, using a linear search to locate messages is an entirely reasonable strategy. When this ratio strays very far from unity, however, the optimal linear search is expected to use long sequences of (short) idle slots or collisions (depending on whether the idle- or collision-detect time is smaller) at the start of each service epoch.

While the idle-detect time is usually assumed to be less than the collision-detect time, one can imagine systems in which the reverse is true. For example, consider a "central station" model with many terminals in line of sight of the station but hidden from each other. In [Toba75a] Tobagi and Kleinrock introduce a "busy tone" channel to allow CSMA protocols to be used. If, instead, the station were to broadcast a continuous binary signal on the acknowledgement channel of "collision" or "no collision", then collisions would be detected at

the terminals within a round-trip propagation time, but an idle slot could not be distinguished from a success.

When the idle-detect time is much less than the collision-detect time, we propose the following, super-linear scanning algorithm. (A similar algorithm could be devised for the opposite case.) Let each service epoch begin with a two-level scan for new messages during which only short reservation requests may be transmitted. In the first phase, the algorithm uses a coarse linear search to advance to the neighbourhood of the next message(s). At the start of each minislot, the protocol enables a (large) set, $E_0$, of stations to transmit reservation requests if they have a message to send. If any of the enabled stations respond, the protocol enters the second phase. If no stations respond, that enabled set requires no further service so a new set is enabled in the next minislot. Since each empty minislot provides no new information about the unexamined arrivals, all enabled sets during this first phase should be of constant measure. Thus, we may assume without loss of generality that the distribution of messages in $E_0$ at the end of phase one is Poisson with parameter $\lambda_0$, conditioned on being non-empty, *i.e.*,

$$Pr[E_0 \textit{ contains } k \textit{ messages}] = \frac{\frac{\lambda_0^k}{k!} e^{-\lambda_0}}{1 - e^{-\lambda_0}}.$$

In the second phase, the algorithm uses a binary search to try to locate the first message from $E_0$ more precisely. Since the channel provides only *binary* feedback, the protocol cannot distinguish between a response from one station and a response from more than one station. Thus, there is no reason for the protocol not to use a *constant* number of binary search steps in this second phase.

At the start of the $i^{th}$ minislot in phase two, the protocol enables $E'_{i-1}$, which consists of the *first half* of $E_{i-1}$. If *any* stations respond, $E_i = E'_{i-1}$. Using Bayes' theorem, it is easy to see that the *a posteriori* probability density for messages in $E_{i-1} - E'_{i-1}$ is again (unconditional) Poisson at the original intensity. Thus $E_{i-1} - E'_{i-1}$ is now effectively unexamined, and can be ignored for the duration of the epoch. If *no* stations respond, $E'_{i-1}$ requires no further service, and we let $E_i = E_{i-1} - E'_{i-1}$. In either case, we may apply Bayes' theorem to find the distribution of messages in $E_i$ given that the *a priori* distribution of messages in $E_{i-1}$ was Poisson with parameter $\lambda_{i-1}$ (conditioned on containing at least one message) and that through enabling $E'_{i-1}$, $E_i$ is known to contain at least one message. Let $A$ be the event that $E_{i-1}$ contains $k$ messages, $k > 0$. Let $B$ be the event that both $E_i$ and $E_{i-1}$ each are known to contain at least one message. Using Bayes' theorem, we have

$$Pr[A \mid B] = \frac{Pr[B \mid A] \cdot Pr[A]}{Pr[B]}.$$

Since $A$ implies $B$, $Pr[B \mid A] = 1$. Thus

$$Pr[A \mid B] = \frac{\frac{\lambda_i^k}{k!} e^{-\lambda_i}}{1 - e^{-\lambda_i}},$$

and we see that after $i$ binary search steps, the distribution of messages in $E_i$ is Poisson with parameter $\lambda_i$, conditioned on having at least one message.

In Figure 7.1, we show the operation of a hybrid CSMA protocol in which access to the channel is based on the message arrival times (*i.e.*, each enabled set is an interval on the real time line). For simplicity, we assume that the protocol simply enables each of the intervals found by the scan procedure as they are found, and that any colliding messages lost. Messages arrive to the system from a Poisson source and thereafter 'age' at a rate of one second per second until they leave the system at the end of their (possibly unsuccessful) transmission on the channel. Following Figure 3.2, messages are shown as lines of unit slope and enabled sets are parallelograms 'sweeping out' a range of ages for enabling messages. Each service epoch begins with the first phase of the scan procedure (labelled $S_1$ in the Figure) enabling a sequence of zero or more *idle* intervals followed by one *busy* interval, which terminates this first phase. The second phase (labelled $S_2$ in the Figure) refines this busy interval to $1/8^{th}$ of its original length using three additional minislots, after which the message(s) in this refined interval are transmitted (during the transmission period, labelled *TP* in the Figure). This Figure also illustrates the optimization described below in §7.3. Since our protocol immediately enables the *entire* busy interval found by the scan procedure, one minislot would be saved in some service epochs if our protocol permitted stations to transmit complete messages rather than short reservation bursts during the last minislot of phase two. Comparing the first two service epochs in the Figure, we see one fewer steps is required in the second phase when the last step enabled a busy interval than when it enabled an idle interval.

When this two-level scan procedure is applied to infinite population protocols with Poisson arrivals, the first phase of the scan procedure obeys a remarkable conservation law at optimality:

**Theorem 7.1**

Let $\Omega$ be the expected channel overhead per successful message transmission. Then, ignoring the fact that a non-negative *integral* number of binary search steps must be performed in phase two, $\Omega$ is minimized by enabling a Poisson sample with parameter $\lambda_0 \approx .66794$ in each minislot during phase one *independent* of both the cost of binary reservations, idle slots, and collisions, and of the particular multiple access protocol that is employed.

**Proof:**

Each of the times at which the protocol begins the first phase of the scan procedure is a regeneration point for the protocol. Thus each service epoch consists of an initial scan followed by a busy period that lasts until the next regeneration point. We assume that a particular protocol, say $A$, is employed, such that $A$ transmits successfully $k$ messages during a service epoch with

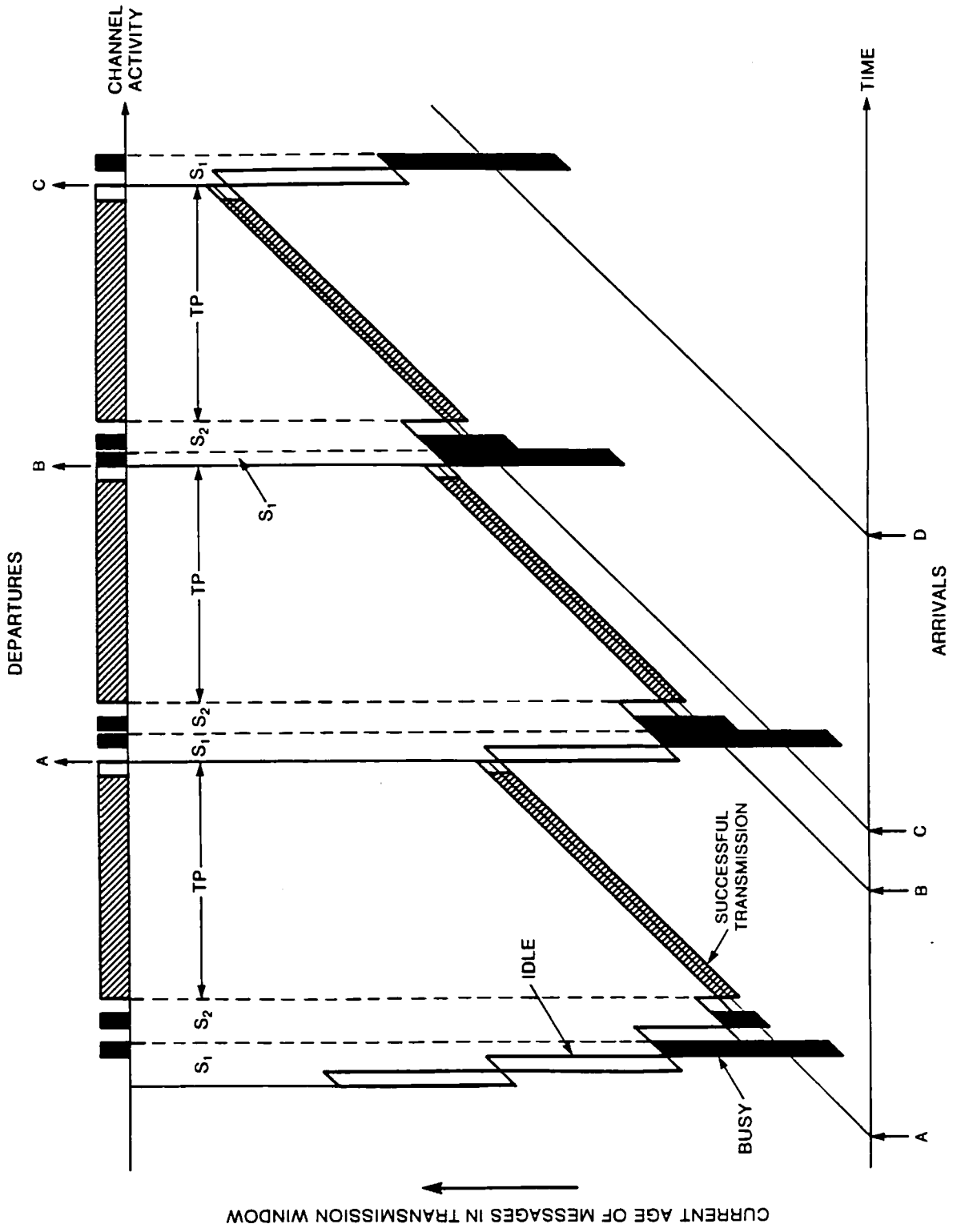


Figure 7.1: The Operation of a Hybrid CSMA Protocol

probability $n_k$. Let $\omega_k$ be the expected channel overhead (not including the cost of the initial scan procedure) during a busy period during which $A$ transmits successfully $k$ messages.

During the initial scan procedure, we assume that Poisson samples with parameter $\lambda_0$ are enabled during phase one, which are refined to be conditional (*i.e.*, non-empty) Poisson samples with parameter $\lambda^*$, a constant, during phase two. To complete the proof, we must show that for *every* choice of $\lambda^*$, and every protocol $A$, the *same* value of $\lambda_0$ is optimal, namely $\lambda_0 \approx .66794$.

For any given choices of $\lambda_0$ and $\lambda^*$, the expected cost per service epoch of the scan procedure may be calculated as follows. The expected number of binary reservations per service epoch in phase one is

$$\frac{1}{1-e^{-\lambda_0}}.$$

Ignoring the requirement that the number of binary search steps must be a non-negative integer, phase two utilizes a constant number,

$$\log_2\left(\frac{\lambda_0}{\lambda^*}\right),$$

of binary reservations.

The activity on the channel may be viewed as sequences of zero or more *unsuccessful* service epochs (during which no messages are transmitted successfully) alternating with *successful* service epochs (during which one or more messages are transmitted successfully). On the average, there will be $\frac{n_0}{1-n_0}$ unsuccessful service epochs between each successful service epoch. Since the probability that the successful transmission of a randomly chosen message occurred in a service epoch during which $k$ messages were transmitted successfully is given by

$$\frac{k \cdot n_k}{\sum_{j=1}^{\infty} j \cdot n_j},$$

we see that

$$\Omega = \sum_{i=1}^{\infty} \frac{1}{i}\left(\frac{i \cdot n_i}{\sum_{j=1}^{\infty} j \cdot n_j}\right)\left[\frac{1}{1-n_0}\left(\frac{1}{1-e^{-\lambda_0}} + \log_2\left(\frac{\lambda_0}{\lambda^*}\right)\right) + \frac{n_0\omega_0}{1-n_0} + \omega_i\right]. \tag{7.1}$$

For any *fixed* value of $\lambda^*$, the minimum value of $\Omega$ must occur when $\lambda_0 = 0$, $\lambda_0 = \infty$, or at some point where the partial derivative with respect to $\lambda_0$ vanishes. Since the expected number of steps in phase one grows to infinity when $\lambda_0 \to 0$, and the number of steps in phase two grows to infinity when $\lambda_0 \to \infty$, only points where $\partial\Omega/\partial\lambda_0 = 0$ need be examined. Since $\lambda^*$ is assumed to be constant, the set $\{n_i\}$ are independent of $\lambda_0$. Furthermore, being probabilities,

any non-negative linear combination of $\{n_i\}$ must also be non-negative so that derivative can be zero only when

$$\frac{\partial}{\partial \lambda_0}\left(\frac{1}{1-e^{-\lambda_0}} + \log_2\left(\frac{\lambda_0}{\lambda^*}\right)\right) = 0. \tag{7.2}$$

Taking the derivative, this condition becomes

$$\frac{e^{-\lambda_0}}{(1-e^{-\lambda_0})^2} = \frac{1}{\lambda_0 \ln(2)}$$

or

$$\ln(2)\lambda_0 e^{-\lambda_0} = (1-e^{-\lambda_0})^2. \tag{7.3}$$

We note the surprising result that Eq. (7.3) depends only on the fact that an even binary search (of unknown depth) will be performed in phase two, and is completely independent of $\lambda$ ', $\{n_i\}$, $\{\omega_i\}$, and thus $A$.

To complete the proof, we now show that there is a unique value of $\lambda_0$ in the range $0 < \lambda_0 < \infty$ that solves Eq. (7.3). Let $\alpha(\lambda_0) \triangleq \ln(2)\lambda_0 e^{-\lambda_0}$ and $\beta(\lambda_0) \triangleq (1-e^{-\lambda_0})^2$ be the left and right hand sides, respectively, of Eq. (7.3). Thus

$$\alpha'(\lambda_0) = \ln(2)e^{-\lambda_0}[1-\lambda_0], \tag{7.4}$$

$$\alpha''(\lambda_0) = \ln(2)e^{-\lambda_0}[\lambda_0 - 2], \tag{7.5}$$

$$\beta'(\lambda_0) = 2e^{-\lambda_0}[1-e^{-\lambda_0}], \tag{7.6}$$

and

$$\beta''(\lambda_0) = 2e^{-\lambda_0}[2e^{-\lambda_0} - 1]. \tag{7.7}$$

Since

$$\alpha(0) = \beta(0) = 0,$$

$$\alpha'(0) = \ln(2) > 0,$$

$$\beta'(0) = 0,$$

and both $\alpha$ and $\beta$ are continuous and differentiable, there must be some neighbourhood of 0 in which $\alpha(\lambda_0) > \beta(\lambda_0)$. However,

$$\lim_{\lambda_0 \to \infty} \alpha(\lambda_0) = 0 < 1 = \lim_{\lambda_0 \to \infty} \beta(\lambda_0),$$

so that at least one solution to Eq. (7.3) must exist. Numerical computation reveals that $\lambda_0 \approx .66794$ is one such solution.

To show uniqueness, we consider two regions bounded by $\lambda_0 = \ln(2) \approx .693$. We see from Eq. (7.7) that $\beta(\lambda_0)$ is convex for $\lambda_0 < \ln(2)$ and concave for $\lambda_0 \geq \ln(2)$, and from Eq. (7.5) that $\alpha(\lambda_0)$ is concave for $\lambda_0 < 2$ and convex for $\lambda_0 \geq 2$. A concave function can intersect a convex function at most twice. (See Theorem 4.2. in Chapter 4.) Thus the two intersections at $\lambda_0 = 0$ and $\lambda_0 \approx .66794$, respectively, must be the only intersections in the region $0 \leq \lambda_0 < \ln(2)$. By continuity, it follows that $\alpha(\lambda_0) < \beta(\lambda_0)$ must hold for $.66794 \leq \lambda_0 \leq \ln(2)$. Comparing Eqs. (7.4) and (7.6), we see that for all $\lambda_0 \geq \ln(2)$, $\alpha'(\lambda_0) \geq 1$ while $\beta'(\lambda_0) < 1$. Since $\alpha(\lambda_0) < \beta(\lambda_0)$ for $\lambda_0 = \ln(2)$, it must also be the case that $\alpha(\lambda') < \beta(\lambda')$ for all $\lambda' > \ln(2)$. Thus $\lambda_0 \approx .66794$ must be the *unique* solution to Eq. (7.3) in the region $0 < \lambda_0 < \infty$.

■

We have thus shown that the optimum value of $\lambda_0$ for the two-level scan procedure defined above is a constant. Thus, whenever it makes sense to consider binary reservation-aided infinite population protocols (e.g., in local packet radio networks where the ratio of idle- to collision-detect times is small), the parameters in the scan procedure can be optimized by solving a *one* dimensional optimization.

### 7.3: Improved Performance with Hybrid CSMA and Tree Algorithms

Using Eq. (7.1), we may calculate the capacity of some representative hybrid protocols to determine their sensitivity to the propagation time, $a$, and to compare their performance to similar "non-reservation" protocols. We will examine both a hybrid CSMA loss protocol and a hybrid tree algorithm.

Let us define the following hybrid CSMA protocol, which is an extension of minislotted virtual time CSMA. For simplicity, let us assume that it is a loss protocol, *i.e.*, messages are discarded after one unsuccessful attempt is made to transmit the *entire* message, so that the arrival process will be a stationary Poisson process. We shall again assume that permission to transmit either a reservation burst or a complete message is controlled by a set of synchronous virtual clocks. However, the behaviour of the virtual clock must be slightly more complex in order to implement the two level scan procedure described above.

This hybrid CSMA protocol differs from virtual time CSMA during the scan procedure that begins each service epoch. During this scan procedure, each station transmits a reservation burst whenever a "tick" of the virtual clock passes the arrival time of one of its messages. During the first phase, the size of the ticks is chosen to enable a Poisson set with parameter $\lambda_0 \approx .66794$. As soon as one or more stations respond with a reservation burst, the second phase of the scan procedure is begun. The binary search in phase two is performed according to the following pair of rules. First, the virtual clock must "back up" one "tick" after each minislot in which one or more stations responded with a reservation burst. Second, the amount by which the virtual clock advances at each tick is halved before each minislot.

For this hybrid CSMA loss protocol, it is easy to calculate the system parameters for Eq. (7.1). At most one message is transmitted successfully in any service epoch, so that

$$n_1 = \frac{\lambda^* e^{-\lambda^*}}{1 - e^{-\lambda^*}} = \frac{\lambda^*}{e^{\lambda^*} - 1}$$

and

$$n_0 = 1 - n_1.$$

To allow for the propagation time, each message transmission slot is assumed to be of duration $1+a$. Thus

$$\omega_0 = \frac{1}{a} + 1$$

and

$$\omega_1 = a.$$

Substituting these values into Eq. (7.1) gives

$$\Omega = \frac{1}{n_1}\left[1 + \frac{1}{1 - e^{-\lambda_0}} + \log_2\left(\frac{\lambda_0}{\lambda^*}\right) + \frac{n_0}{a}\right]. \tag{7.8}$$

This hybrid CSMA protocol immediately enables the *entire* set of stations found by the scan procedure to transmit their complete messages. Thus, it is easy to see that the overhead could be reduced by allowing the stations to transmit *complete messages* rather than reservation bursts in the last ($n^{th}$) step of the binary search procedure, when the first half of $E_{n-1}$ is examined. If any stations respond with a message transmission, we will have saved one reservation request time. If no stations respond, it becomes known that the *second* half of $E_{n-1}$ must contain one or more messages. In that case, all stations in that set are enabled to transmit their complete messages in the next slot. Since the times for all stations to become aware that no stations began transmitting a message and that no stations transmitted a reservation request are equal, this improvement will reduce the overhead by one end-to-end propagation time, $a$, with probability

$$1 - \frac{e^{-\lambda^*} - e^{-2\lambda^*}}{1 - e^{-2\lambda^*}}.$$

Thus, the average overhead per successful transmission will be reduced to

$$\Omega = \frac{1}{n_1}\left[1 + \frac{1}{1 - e^{-\lambda_0}} + \log_2\left(\frac{\lambda_0}{\lambda^*}\right) - 1 + \frac{e^{-\lambda^*} - e^{-2\lambda^*}}{1 - e^{-2\lambda^*}} + \frac{n_0}{a}\right]. \tag{7.9}$$

Figure 7.2 shows the capacity of the improved hybrid CSMA protocol described above as a function of the propagation time, $a$, when there is no collision detection (*i.e.*, $b = 1$). The capacity curve for minislotted virtual time CSMA from Figure 6.5 has been included for comparison. It is clear from the figure that the hybrid CSMA protocol performs significantly better for small values of $a$ (e.g., $a \lesssim .1$), but the added overhead of the reservation bursts makes the

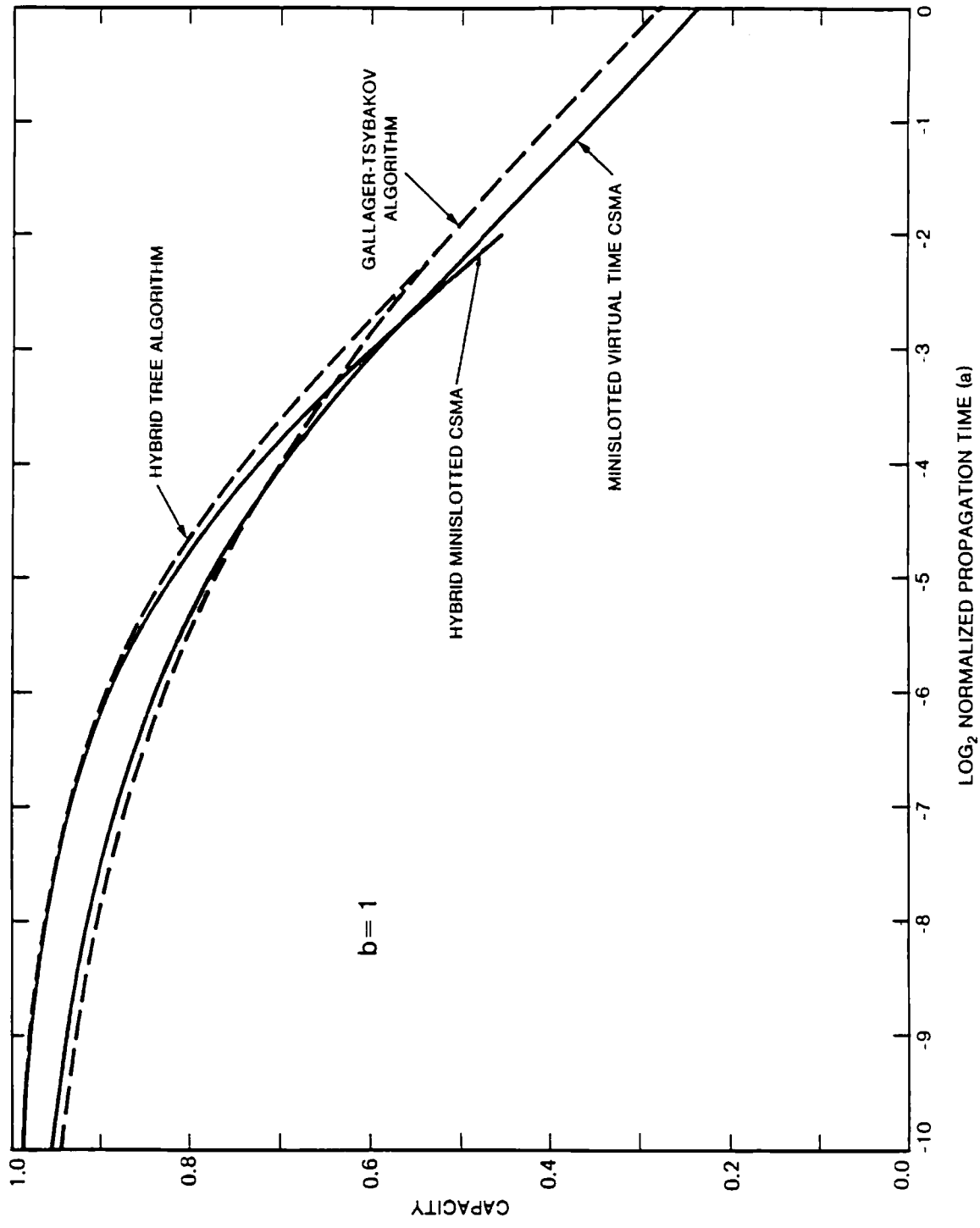


Figure 7.2: Sensitivity of Hybrid Protocols to Propagation Time

protocol significantly less efficient than minislotted virtual time CSMA as the propagation time grows closer to the message transmission time.

We have already seen in Chapter 5 that CSMA loss protocols are about as efficient as the best collision resolution algorithms that operate primarily on the message channel when the ratio of idle- to collision-detect times is small. Because of the high cost of message collisions, it does not seem worthwhile to try to resolve collisions on the message channel. We thus propose the following hybrid tree algorithm that requires only *binary* feedback during much of the collision resolution procedure. We note that this new hybrid tree algorithm is related to a "breadth-first" traversal of the same collision resolution tree that is employed by the original Capetanakis algorithm.

This hybrid tree algorithm may be described as follows. We shall again assume that at the end of the two-level scan procedure, the algorithm is supplied with a set, $E_n$, known to contain one or more messages according to a conditional Poisson distribution with parameter $\lambda^*$. Whenever the scan procedure produces such a set, the algorithm immediately grants permission to all stations included in $E_n$ to transmit their entire message in the next slot. (Since the entire set $E_n$ is enabled, we can immediately incorporate the same improvement described above to (possibly) save one reservation request per service epoch.) If enabling $E_n$ yields a successful transmission, the service epoch is ended and the next scan procedure begins. However, in this hybrid tree algorithm we shall resolve all collisions rather than discard any colliding messages, thereby transmitting all messages in $E_n$ before ending the service epoch.

Suppose a collision occurs when all stations in some set $E$ are enabled to transmit messages. This situation is illustrated in Figure 7.3. A collision resolution period (labelled *CR* in the Figure) follows the first transmission period on the Figure. During the first minislot, all stations in $E'$, the first half of $E$ are enabled to transmit reservation requests. If no station responds (as is the case in the Figure), it becomes known (at the cost of *one* binary reservation request) that $E'' \triangleq E - E'$ contains two or more messages, and the collision resolution algorithm is repeated starting with $E''$. If at least one station responds (which occurs after one more step of the algorithm in the Figure), all stations in $E''$ are enabled to transmit *complete messages*. We consider two cases. First, if there are no stations in this second enabled set, it becomes known (at the cost of *two* binary reservation requests) that $E'$ contains two or more messages, and the algorithm is repeated starting with $E'$. Second, if at least one station in $E''$ responds (as is the case in the Figure), we must consider two further cases. First, if enabling $E''$ yields a successful transmission, that set requires no further processing and the algorithm continues by processing $E'$. (In this manner, the service epoch in the Figure ends after two successful message transmission.) Second, if enabling $E''$ yields a collision, the collision resolution procedure is applied recursively to that set, after which it continues by processing $E'$. Using Bayes' rule, it is not difficult to show that in either case the distribution of messages in $E'$ is Poisson with half the parameter of $E$, conditioned on there being one or more messages. Thus $E'$ is even more likely to contain exactly one message than are the sets with parameter $\lambda^*$ produced by the two-level scan procedure that initiates each service epoch. The algorithm proceeds by granting all stations in $E'$ permission to transmit their entire message(s), possibly

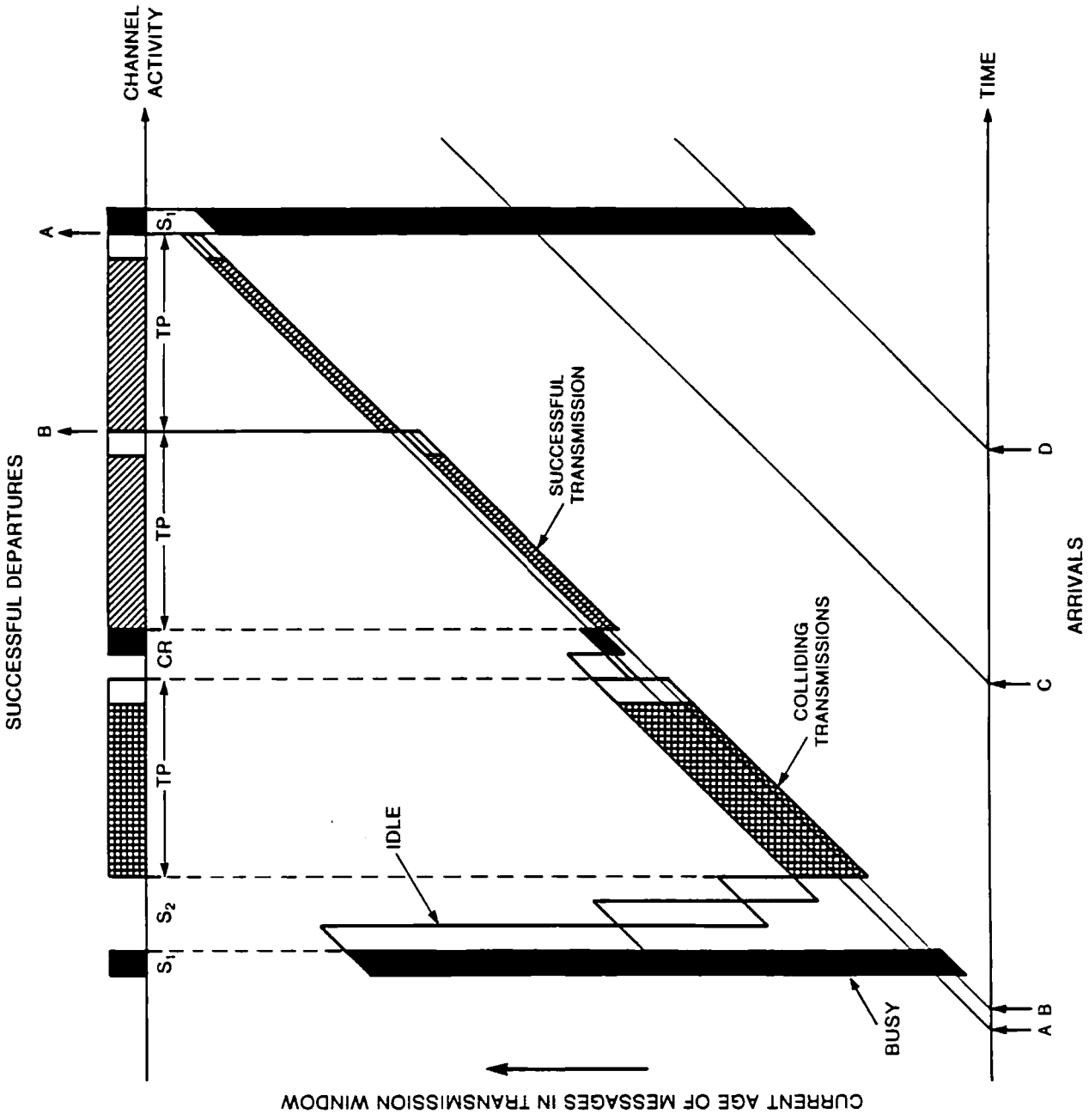


Figure 7.3: The Operation of a Hybrid Tree Algorithm

causing a further collision to be resolved by the above algorithm.

Since many of the algorithmic steps are performed by transmitting (short) binary reservation bursts rather than complete messages, we expect this algorithm to be more efficient than, say, the Gallager-Tsybakov algorithm when $a$ is small. Furthermore, since an average of about *four* binary reservations used to find *two* busy sets, this procedure is also more efficient than the hybrid CSMA protocol in which colliding messages are discarded immediately.

To calculate the capacity of this protocol numerically, it remains to establish a set of recursive equations to define the sets $\{n_k\}$ and $\{\omega_k\}$. We note that like Capetanakis' original algorithm and unlike the Gallager-Tsybakov algorithm, all messages that are contained in an enabled set are transmitted during the same service epoch. Thus, $\{n_k\}$ is given by the conditional Poisson distribution with parameter $\lambda^*$, *i.e.*,

$$n_k = \frac{\lambda^{*k}}{k!} e^{-\lambda^*}.$$

To calculate the set $\{\omega_k\}$, we observe that the collision resolution algorithm alternates between two sets of states, depending upon whether the current set $E$ is known to contain one or more, or two or more messages, respectively. Thus $\{\omega_k\}$ can be calculated by simultaneously calculating the set $\{\mu_k\}$, the average overhead to service a set of $k$ messages given that it is known to contain *two* or more messages. Following the notation of Chapter 3, let $p_{i,j}$ be the probability that exactly $j$ messages are found in the first half of a set that contains exactly $i$ messages. Hence

$$\omega_k = \frac{1}{a} + 1 + \mu_k \qquad k \geq 2,$$

with initial conditions $\omega_0 = \omega_1 = 1$, and

$$\mu_k = p_{k,0}(1+\mu_k) + p_{k,k}(1+\mu_k) + \sum_{i=1}^{k-1} p_{k,i}(1+\omega_i+\omega_{k-i})$$

$$= \frac{1 + \sum_{i=1}^{k-1} p_{k,i}(\omega_i+\omega_{k-i})}{1 - p_{k,0} - p_{k,k}} \qquad k \geq 2,$$

with initial conditions $\mu_0 = \mu_1 = 0$. Since we have incorporated the improvement described above to (possibly) save the last binary search step, we must rewrite Eq. (7.1) as

$$\Omega = \sum_{i=1}^{\infty} \frac{1}{i} \left( \frac{i \cdot n_i}{\sum_{j=1}^{\infty} j \cdot n_j} \right) \left[ \frac{1}{1-n_0} \left( \frac{1}{1-e^{-\lambda_0}} + \log_2\left(\frac{\lambda_0}{\lambda^*}\right) - 1 + \frac{e^{-\lambda^*} - e^{-2\lambda^*}}{1-e^{-2\lambda^*}} \right) + \frac{n_0\omega_0}{1-n_0} + \omega_i \right]. \qquad (7.10)$$

For comparison, the performance of the Gallager-Tsybakov algorithm with idle-detection is also shown in Figure 7.1. Unfortunately, with this algorithm the number of messages actually transmitted out of an original set of $k$ messages is a random number, so it is difficult to calculate the parameters for Eq. (7.1). However, Eq. (3.1), giving the recursion for

the expected length of a busy period that started with $k$ messages, can readily be extended to this local network environment. In this case, we find

$$w_k = p_{k,0}(a+w_k) + p_{k,k}(1+a+w_k) + p_{k,1}(2(1+a)+w_{k-1}) + \sum_{i=2}^{k-1} p_{k,i}(1+a+w_i)$$

$$= \frac{1+a-p_{k,0}+p_{k,1}(1+a+w_{k-1}) + \sum_{i=2}^{k-1} p_{k,i}\, w_i}{1-p_{k,0}-p_{k,k}}, \tag{7.11}$$

with boundary conditions $w_0 = a$, $w_1 = 1+a$, $\{n_k\}$ are given be Eq. (3.2), and the throughput is given by Eq. (3.3).

As was the case with the hybrid CSMA protocol, we again observe from Figure 7.1 that this new hybrid tree algorithm is inferior to the Gallager-Tsybakov algorithm for large values of the idle-detect time, and more efficient when the idle detect time grows smaller. In particular, we see from the Figure that when $a = .01$, this hybrid tree algorithm can achieve a stable throughput of $\approx .92$. In § 4.2.3, we defined a "naive" upper bound on the performance of protocols in the carrier sense environment, in which we assumed that all exchange of information over the channel was accomplished by transmitting only complete messages. Under this assumption, we showed that no protocol can attain a higher throughput than $\approx .88$ with these system parameters (*i.e.*, $a = .01$ and $b = 1$). Since the hybrid tree algorithm defined above can achieve a significantly higher stable throughput, it is clear that the class of hybrid protocols offers fundamental performance advantages over the "standard" infinite population tree algorithms when the length of an idle slot is much less than the length of a collision.

# CHAPTER 8
## Conclusions and Suggestions for Future Work

### 8.1: Protocol Recommendations for Local Networks

Several factors need to be considered in the selection of a multiple access protocol for local networks. While at first it might seem that networks should be built using the most efficient multiple access protocol that has yet been devised, one must keep in mind that optimized protocols are inherently less robust and possibly more complex, and thus more sensitive to errors.

In local networks, it is commonly assumed that the duration of idle slots (and possibly of collisions) is shorter than the duration of successful transmissions. Having variable slot sizes makes it possible even for crude protocols to achieve high maximum channel utilizations in local network applications. (For example, see how efficient virtual time CSMA is, even though it is no more than a variation of ALOHA with variable length slots.) Thus, the difference in the capacity of a local network using, say, the Capetanakis protocol instead of the Gallager-Tsybakov algorithm would be small.

However, achieving maximum capacity is by no means the only goal in choosing a protocol for a local network. Long-term stability and good delay characteristics are certainly just as important. Thus, since provably stable tree algorithms that guarantee first-come first-served delivery of messages exist, it seems difficult to justify using CSMA protocols with their inherent stability problems, random delivery of messages, and poor delay characteristics. At the very least, when CSMA protocols are used, we feel that virtual time CSMA should be strongly considered. We also note that since tree algorithms are inherently stable and CSMA protocols are inherently unstable and thus cannot operate without some additional control procedures, it is doubtful that CSMA protocols are any less complex to implement.

When choosing a tree algorithm, one must take care to study the robustness of the algorithm with respect to errors in the feedback information. Massey [Mass80a] has done a careful study of several variations of the original Capetanakis algorithm. He showed that many of the more clever algorithms can deadlock when an idle slot is mistakenly interpreted as a collision, but that some simple versions of the protocol are immune from this problem. Similar problems can also occur when the characteristics of the traffic change. For example, trying to support packetized speech traffic will be difficult for the more clever tree algorithms. It is an implicit assumption in such protocols that information about the set of waiting messages can be gathered from the history of past activity on the channel. Thus, any message that has been involved in a collision is expected to remain in the system until it is granted permission to transmit. However, speech is an example of real-time traffic. Furthermore, speech traffic

usually contains redundant information. If a particular message gets delayed too long, the receiver will attempt to reconstruct the speech without it. Thus, late messages are of no use to the communicating speech processes, and are likely to be dropped. Paradoxically, the very issue that Massey has shown to be a potential deadlock hazard is an inherent feature of one important class of traffic!

In summary, the most reasonable protocol choice for a local coaxial cable networks at this time seems to be a slight modification of the Capetanakis tree algorithm. This modification is to control access to the channel at the start of each service epoch explicitly using a fixed size window, (or the virtual clock mechanism from virtual time CSMA). We note that this modified protocol avoids a nasty correlation between neighbouring service epochs. It is also possible to support priority classes in a very neat manner based on our results in section 6.6.

For local packet radio networks, we have seen that CSMA protocols can have a higher capacity than even the Gallager-Tsybakov algorithm. Radio networks also have much noisier channels, making it more difficult to ensure that all stations receive accurate feedback information. Furthermore, because of the possibility of mobile stations, it is more difficult to gather information about messages. Indeed, since mobile stations may periodically lose contact with the network (when passing though a tunnel, say), it may be unreasonable to assume that the stations can follow any synchronous tree algorithm. Thus, the virtual time CSMA protocol may be best suited for such networks. In addition, it can operate asynchronously — impossible with every known tree algorithm. We note, however, that the hybrid protocols discussed in Chapter 7 offer higher capacity in the packet radio environment and should also be considered.

## 8.2: Extensions

### 8.2.1: General Open Problems

Synchronous tree collision resolution algorithms are now becoming well understood in the "standard" case in which all slots are of constant length. However, much work remains to be done in extending these types of protocols to other operating environments. In particular, we see the following general open problems.

To date, all of the tree algorithms have relied on the fact that the channel is slotted. A synchronous channel facilitates gathering information about the distribution of messages in enabled sets. All stations know precisely which enabled set corresponds to a specific (ternary) "bit" of feedback information. Unfortunately, it is difficult to maintain global time synchronization in a network. Thus, for practical reasons, it is desirable to use asynchronous "unslotted" protocols. At the present time, only variations of the ALOHA-based protocols (such as the CSMA protocols) have been proposed as unslotted infinite population protocols. An important area of further research would be to determine whether any form of "tree algorithm" can operate on an unslotted channel, and to define such a protocol.

Another problem of considerable practical interest in packet radio networks is the extension of tree algorithms to the hidden station environment. To be practical, several limitations need to be overcome. First, the information gathering process is more difficult because the sending and receiving stations no longer share the same environment. Thus, some outside agent must supply the sender with the feedback information. Second, given the correct feedback information from the receiver's environment, making reasonable decisions about which stations to enable is more difficult. For example, what should be the decision if it is known that a particular message could be transmitted successfully to its intended destination, but its transmission would destroy some other successful transmissions [Yemi79a].

### 8.2.2: Specific Extensions of this Work

In Chapter 6, we proposed a method for incorporating head-of-line priorities into the virtual time CSMA protocol. We showed that such an extension will not affect the capacity of the protocol or the *overall* mean message delay. However, we did not show how priorities affect the mean delays for different classes of stations. We conjecture that a close approximation to the delay performance could be obtained by extending the discrete time M/D/1 model for minislotted virtual time CSMA in a similar manner to the way in which the delay expressions for HOL priority queues are obtained from the corresponding M/G/1 queueing results [Klei76a].

In Chapter 7, we introduced hybrid carrier sense − binary search protocols for systems in which binary (but not ternary) feedback is available at low cost. Such is the case in local radio networks, where rapid idle detection is possible but there is no collision detection. We calculated the capacity of a hybrid CSMA and a hybrid tree protocol. However, we again did not present any results for their delay performance. We believe that a similar, albeit more tedious, numerical approach to the second delay model for minislotted virtual time CSMA is solvable, and should be investigated.

In addition, we have shown that the maximum throughput for the hybrid protocols that we described in Chapter 7 can exceed a "naive" upper bound on capacity obtained by a straightforward extension of work described in Chapter 4. When the ratio of idle-detect to collision-detect times is unity, these bounds are correct. However, when this ratio is far from unity (as we have assumed in Chapter 7), further work is required to properly bound the capacity. Trivially, a weak upper bound can be obtained by assuming that both idle-detection and collision-detection can be accomplished in a time equal to the *minimum* of the two, but its results can be too weak to be of any value. A more promising approach would be to extend the work of Tsybakov and Mikhailov (§ 4.3) to account for the fact that the protocol could choose to send short binary messages that could increase the size of the accepted set but not the number of successful transmissions.

# APPENDIX A
## The Discrete Time M/G/1 Queue

### A.1: The Behaviour of the Discrete Time M/G/1 Queue

The M/G/1 queue in *continuous* time has been studied extensively in the queueing theoretic literature [Klei75a]. This queueing system has a memoryless *Poisson* interarrival distribution with parameter $\lambda$, and a general, independent service time distribution $B(x)$ with mean $\bar{x}$ and Laplace transform $B^*(s)$. It is well known that the Pollaczek-Khinchin mean value formula gives the average time in system and that the Pollaczek-Khinchin transform equation gives the transform of the distribution of the number of customers in the system for the continuous time M/G/1 queueing system.

In the study of synchronous protocols, we will have need of the analogous results for *discrete* time queueing systems, which for completeness we present below. In particular, we shall assume that time advances in discrete steps such that events may occur *only* at some time $k \cdot \tau$, $k = 0, 1, \cdots$. Each such event time will be called an *arrival point*. Clearly the interarrival distribution will exhibit the memoryless property if each arrival point is independently *busy* with probability $p$ and *idle* with probability $1-p$. This corresponds to a *geometric* interarrival distribution with parameter $p$. We shall permit a general, independent *discrete* service time distribution, requiring only that each service time be an integer multiple of $\tau$. We thus define $b_k$, with $z$-transform $B(z)$, to be the probability that the service time of the next customer is $k \cdot \tau$, $k = 1, 2, \cdots$, and $B_k = \sum_{i=1}^{k} b_i$ to be the service time distribution function. Since $\{b_k\}$ is a discrete probability density, $\sum_{k=1}^{\infty} b_k = 1$. As with the continuous time case, our analysis requires that the service time for each customer be independent of the service times of all previous customers.

We proceed using the method of imbedded Markov chains ([Klei75a], pp. 174ff.). Let $q_k$, with $z$-transform $Q(z)$, be the equilibrium probability of $k$ customers left behind by a departure from service. Since the number of customers in the system changes in unit steps,[1] $\{q_k\}$ must also describe the number in system as seen by a new arrival to the system (see [Klei75a], problem 5.6). Furthermore, since the arrival process is *memoryless*, $q_k$ must also be the equilibrium probability of $k$ customers in the system over all time.

[1] This property holds only if we assume that $b_0 = 0$, *i.e.*, every customer requires a non-zero time in service.

Let $v_k$, with $z$-transform $V(z)$, be the equilibrium probability of $k$ arrivals during a service time. Thus

$$\begin{aligned} V(z) &\triangleq \sum_{i=0}^{\infty} v_i z^i \\ &= \sum_{i=0}^{\infty}\sum_{j=i}^{\infty} \binom{j}{i} p^i (1-p)^{j-i} b_j z^i \\ &= \sum_{j=0}^{\infty} b_j \sum_{i=0}^{j} \binom{j}{i} (pz)^i (1-p)^{j-i} \\ &= \sum_{j=0}^{\infty} b_j (1-p+pz)^j \\ &\triangleq B(1-p+pz). \end{aligned} \tag{A.1}$$

It is a property of $z$-transforms of discrete probability density functions [Klei75a] that the average number of arrivals per service time, $\bar{v}$, can be found as

$$\bar{v} = V'(1) \triangleq \frac{d}{dz} V(z)\Big|_{z=1}.$$

Applying Eq. (A.1), we obtain

$$\bar{v} = B'(1) \cdot \frac{d(1-p+pz)}{dz} = p\bar{x}. \tag{A.2}$$

If we define the discrete convolution $f \circledast g$ of two non-negative discrete density functions to be the sequence whose $n^{th}$ term is

$$(f \circledast g)_n \triangleq \sum_{j=0}^{n} f_j g_{n-j},$$

then the balance equations for $\{q_k\}$ at the imbedded points are simply

$$q_k = (q \circledast v)_{k+1} - q_0 v_{k+1} + q_0 v_k. \tag{A.3}$$

Applying the convolutional property of $z$-transforms [Klei75a] to this set of equations, we obtain

$$\begin{aligned} Q(z) \triangleq \sum_{k=0}^{\infty} q_k z^k &= \sum_{k=0}^{\infty} \left[ (q \circledast v)_{k+1} z^k - q_0 v_{k+1} z^k + q_0 v_k z^k \right] \\ &= \frac{Q(z)V(z) - q_0 v_0 - q_0(V(z) - v_0)}{z} + q_0 V(z) \\ &= \frac{q_0 V(z)(1-z)}{V(z) - z}. \end{aligned} \tag{A.4}$$

Since $Q(1) \triangleq 1$, one application of l'Hôpital's rule gives $q_0 = 1 - \bar{v}$, and hence that $\rho = \bar{v}$. Thus, Eq. (A.4) becomes

$$Q(z) = \frac{V(z)(1-\bar{v})(1-z)}{V(z)-z}. \tag{A.5}$$

We note that Eq. (A.5) is identical to the Pollaczek-Khinchin transform equation for M/G/1 queues in continuous time (e.g., Eq. (5.85) in [Klei75a]), except that $V(z) = B^*(\lambda - \lambda z)$ in continuous time, rather than obeying Eq. (A.1).

We continue the analysis by finding the average queue size at customer departure instants from the relation $\bar{q} = Q'(1)$. Since $Q'(1)$ has an indeterminate form, we define $\alpha(z) \triangleq B(z-p+pz)(1-\bar{v})(1-z)$ and $\beta(z) \triangleq B(z-p+pz) - z$ so that $Q(z) = \frac{\alpha(z)}{\beta(z)}$, and

$$Q'(z) = \frac{\alpha'(z)\beta(z) - \alpha(z)\beta'(z)}{\beta^2(z)}. \tag{A.6}$$

Since $\alpha(1) = \beta(1) = 0$, applying l'Hôpital's rule to Eq. (A.6) we obtain

$$\begin{aligned} \bar{q} &= \left. \frac{\alpha''(z)\beta'(z) - \alpha'\beta''}{2\beta'^2} \right|_{z=1} \\ &= \frac{(1-\bar{v})(-2p\bar{x}(p\bar{x}-1) + p^2(\overline{x^2}-\bar{x}))}{2(p\bar{x}-1)^2}. \end{aligned} \tag{A.7}$$

But since $\rho = \bar{v} = p\bar{x}$, Eq. (A.7) may be rewritten as

$$\bar{q} = \rho + \frac{p^2(\overline{x^2}-\bar{x})}{2(1-\rho)}. \tag{A.8}$$

Define the coefficient of variation of a random variable $X$ with mean $\bar{X}$, second central moment $\overline{X^2}$, and standard deviation $\sigma_X$ to be

$$C_X \triangleq \frac{\sigma_X}{\bar{X}} = \frac{\sqrt{\overline{X^2} - (\bar{X})^2}}{\bar{X}}.$$

Then Eq. (A.8) may be rewritten as

$$\bar{q} = \rho + \rho^2 \frac{(1 + C_b^2 - 1/\bar{x})}{2(1-\rho)}, \tag{A.9}$$

or, applying Little's result [Litt61a],

$$\frac{T}{\bar{x}} = 1 + \frac{\rho(1 + C_b^2) - \rho}{2(1-\rho)}. \tag{A.10}$$

The Poisson limit is obtained by simultaneously letting $p \to 0$ and $\tau \to 0$ while preserving the ratio $p/\tau \triangleq \lambda$. Since $\bar{x}$ is measured units of $\tau$, $\bar{x}$ must diverge to $+\infty$ in the Poisson limit if we are to have a non-zero service time. Thus, Eqs. (A.9) and (A.10) are consistent with the analogous forms of the Pollaczek-Khinchin mean value formula in continuous time (e.g., Eqs. (5.63) and (5.71), respectively, in [Klei75a]).

## A.2: Residual Life of the Customer in Service

In the previous section, we have shown that Pollaczek-Khinchin equations for the M/G/1 queueing system in discrete time differ only slightly from the corresponding equations in continuous time. Below, we show from the residual life calculation why this discrepancy occurs.

In the set of balance equations defined by Eq. (A.3), it was assumed that arrivals occur just before a (possible) service completion. A new arrival may enter service immediately if no other customers are waiting. Consequently, a customer arriving during a service time of total length $k \cdot \tau$ can wait *at most* $(k-1) \cdot \tau$ before the service ends. Since we have memoryless arrivals, it follows that

$$Pr[\textit{residual life} = i \,|\, \textit{service time} = j] = \frac{1}{j}, \tag{A.11}$$

and

$$Pr[\textit{residual life} \leqslant i \,|\, \textit{service time} = j] = \frac{i+1}{j} \tag{A.12}$$

for all $i = 0, 1, \ldots, j-1$. The probability that a randomly chosen customer arrives during a service time of length $k \cdot \tau$ is proportional to its length and relative frequency of occurrence, *i.e.*,

$$Pr[\textit{lifetime} = j] = \frac{j \cdot b_j}{\sum_{i=1}^{\infty} i \cdot b_i} = \frac{j \cdot b_j}{\bar{x}}. \tag{A.13}$$

Combining Eqs. (A.11) — (A.13) we obtain

$$\begin{aligned} \hat{f}_i &= \sum_{j=i+1}^{\infty} \frac{1}{j} \cdot \frac{j \cdot b_j}{\bar{x}} \\ &= \sum_{j=i+1}^{\infty} \frac{b_j}{\bar{x}} \\ &= \frac{1 - B_i}{\bar{x}}, \end{aligned} \tag{A.14}$$

the density of the residual life of the customer in service. It follows that the transform of the residual life density must be

$$\begin{aligned} \hat{F}(z) &\triangleq \sum_{i=0}^{\infty} \hat{f}_i z^i \\ &= \frac{1}{\bar{x}} \sum_{i=0}^{\infty} (1 - B_i) z^i \\ &= \frac{1 - B(z)}{\bar{x}(1-z)}. \end{aligned} \tag{A.15}$$

Thus the mean residual life of the customer in service will be $\bar{y} \triangleq \hat{F}'(1)$. Substituting the value of $\hat{F}(z)$ from Eq. (A.15) and applying l'Hôpital's rule, we obtain

$$\bar{y} = \frac{1}{2}\left[\frac{\overline{x^2}}{\bar{x}} - 1\right]. \qquad \text{(A.16)}$$

Comparing Eq. (A.16) with the corresponding equation in continuous time (e.g., Eq. 5.15 in [Klei75a]), we find that the mean residual life is shorter by exactly $\tau/2$ in discrete time than in continuous time. This is to be expected because we assumed that arrivals occur as a service time is ending (rather than just beginning). This result also confirms the observation from Eq. (A.10): the average time in system must be *shorter* in discrete time than in continuous time.

References


[Abra70a] N. Abramson, "The ALOHA System — Another Alternative for Computer Communications," *AFIPS Conference Proceedings, FJCC* **37**, pp.281-285 (1970).

[Abra73a] N. Abramson, "Packet Switching with Satellites," *AFIPS Conference Proceedings, NCC* **42**, pp.695-703 (June 1973).

[Akav78a] G. Y. Akavia, *Hierarchical Organizations of Distributed Packet-Switching Communication Systems*, Computer Science Department, University of California, Los Angeles (March 1978). Ph.D. Dissertation.

[Avri76a] M. Avriel, *Nonlinear Programming: Analysis and Methods*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1976).

[Bell57a] R. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, N. J. (1957).

[Berg80a] T. Berger, *The Poisson Multiple-Access Conflict Resolution Problem*, School of Electrical Engineering, Cornell University, Ithaca, N. Y. (April 1980). Lectures delivered at CISM summer school, Udine Italy, July 1979.

[Berg81a] T. Berger and N. Mehravari, *An Improved Upper Bound to the Throughput of a Multi-Access Broadcast Channel*, School of Electrical Engineering, Cornell University, Ithaca, N. Y. (February 1981). Submitted to *IEEE Transactions on Information Theory*.

[Cape78a] J. I. Capetanakis, "The Multiple Access Broadcast Channel: Protocol and Capacity Considerations," ESL-R-806, Electronic Systems Laboratory, MIT, Cambridge, Mass. (March 1978).

[Cape79a] J. I. Capetanakis, "Generalized TDMA: The Multi-Accessing Tree Protocol," *IEEE Transactions on Communications* **COM-27**, pp.1476-1484 (October 1979).

[Chla79a] I. Chlamtac, W. R. Franta, and K. D. Levin, "BRAM: The Broadcast Recognizing Access Method," *IEEE Transactions on Communications* **COM-27**, pp.1183-1190 (August 1979).

[Chri78a] G. S. Christensen and W. R. Franta, "Design and Analysis of the Access Protocol for HYPERchannel Networks," *3rd USA-Japan Computer Conference*, pp.86-93 (October 1978).

[Cobh54a] A. Cobham, "Priority Assignment in Waiting Line Problems," *Operations Research* **2**, pp.70-76 (1954).

[Cohe69a] J. W. Cohen, *The Single Server Queue*, Wiley-Interscience, New York (1969).

[Cruz80a] R. Cruz and B. Hajek, *A New Upper Bound to the Throughput of a Multi-Access Broadcast Channel*, Department of Electrical Engineering, University of Illinois, Urbana, Ill. (1980). to appear in *IEEE Transations on Information Theory*.

[Ferg75a] M. Ferguson, "On the Control, Stability and Waiting Time in a Slotted ALOHA Random Access System," *IEEE Transactions on Communications* **COM-23** (November 1975).

[Gall78a] R. G. Gallager, "Conflict Resolution in Random Access Broadcast Networks," *Proceedings of the AFOSR Workshop in Communication Theory and Applications*, pp.74-76 (Sept. 17-20, 1978).

[Gies78a] A. Giessler, J. Hanle, A. Konig, and E. Pade, "Free Buffer Allocation — An Investigation by Simulation," *Computer Networks* 1(3), pp.191-204 (July 1978).

[Haje81a] B. Hajek, "Information of Partitions with Applications to Random Access," *Abstracts of Papers, Sixth International Symposium on Information Theory*, p.106 (February 9-12, 1981).

[Hama80a] V. C. Hamacher and G. S. Shedler, "Local Area Bus Network Performance Analysis," *Conference Record, National Telecommunications Conference*, pp.37.2.1-37.2.6 (December 1980).

[Haye78a] J. F. Hayes, "An Adaptive Technique for Local Distribution," *IEEE Transactions on Communications* **COM-26** (August 1978).

[Heit76a] C. L. Heitmeyer, J. H. Kullback, and J. E. Shore, "A Survey of Packet Switching Techniques for Broadcast Media," NRL Report 8035, Naval Research Laboratory, Washington (October, 1976).

[Howa60a] R. A. Howard, *Dynamic Programming and Markov Processes*, M.I.T. Press, Cambridge, Mass. (1960).

[Humb80a] P. A. Humblet, "Bounds on the Utilization of ALOHA-like Multiple-Access Broadcast Channels," LIDS-P-1000, MIT Laboratory for Information and Decision Systems (June, 1980).

[Jaco78a] I. M. Jacobs, R. Binder, and E. V. Hoversten, "General Purpose Packet Satellite Networks," *Proceedings of the IEEE* **66** (November 1978).

[Kahn77a] R. E. Kahn, "The Organization of Computer Resources into a Packet Radio Network," *IEEE Transactions on Communications* **COM-25**, pp.169-178 (January 1977).

[Kahn78a] R. E. Kahn, S. A. Gronemeyer, J. Burchfiel, and R. C. Kunzelman, "Advances in Packet Radio Technology," *Proceedings of the IEEE* **66**, pp.1468-1496 (November 1978).

[Karl75a] S. Karlin and H. M. Taylor, *A First Course in Stochastic Processes*, Academic Press, New York (1975). Second Edition.

[King62a] J. F. C. Kingman, "The Effect of Queue Discipline on Waiting Time Variance," *Proceedings of the Cambridge Philosophical Society* **58**, pp.163-164 (1962).

[Klei75a] L. Kleinrock, *Queueing Systems, Vol I., Theory*, Wiley-Interscience, New York (1975).

[Klei75b] L. Kleinrock and S. S. Lam, "Packet Switching in a Multiaccess Broadcast Channel: Performance Evaluation," *IEEE Transactions on Communications* **COM-23**, pp.410-423 (April 1975).

[Klei75c] 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), pp.1400-1416 (December 1975).

[Klei76a] L. Kleinrock, *Queueing Systems, Vol II., Computer Applications*, Wiley-Interscience, New York (1976).

[Klei77a] L. Kleinrock and M. Scholl, "Packet Switching in Radio Channels: New Conflict-Free Multiple Access Schemes for a Small Number of Data Users," *Conference Record, International Conference on Communications*, pp.22.1-105 to 111 (June 1977).

[Klei77b] L. Kleinrock, "Performance of Distributed Multi-Access Computer-Communication Systems," *Information Processing 77, Proceedings of IFIP Congress 77*, pp.547-552 (August 1977).

[Klei78a] L. Kleinrock and Y. Yemini, "An Optimal Adaptive Scheme for Multiple Access Broadcast Communication," *Conference Record, International Conference on Communications*, pp.7.2.1-7.2.5 (June 1978).

[Klei78b] L. Kleinrock, "On Flow Control in Computer Networks," *Conference Record, International Conference on Communications* **2**, pp.27.2.1-27.2.5 (June 1978).

[Klei78c] L. Kleinrock and J. Silvester, "Optimum Transmission Radii for Packet Radio Networks or Why Six is a Magic Number," *Conference Record, National Telecommunications Conference*, pp.4.3.1-4.3.5 (December 1978).

[Klei79a] L. Kleinrock, "Power and Deterministic Rules of Thumb for Probabilistic Problems in Computer Communications," *Conference Record, International Conference on Communications*, pp.43.1.1-43.1.10 (June 1979).

[Klei79b] L. Kleinrock, "On a New Class of Queueing Models for Distributed Environments," *Ninth International Teletraffic Congress* (October 1979).

[Konh74a] A. G. Konheim and B. Meister, "Waiting Lines and Times in a System with Polling," *Journal of the ACM* **21**(3), pp.470-490 (July 1974).

[Lam74a] S. S. Lam, "Packet Switching in a Multi-Access Broadcast Channel with Application to Satellite Communication in a Computer Network," UCLA-ENG-7429, Computer Science Department, University of California, Los Angeles (April 1974). Ph.D. dissertation.

[Lam75a] S. S. Lam and L. Kleinrock, "Packet Switching in a Multiaccess Broadcast Channel: Dynamic Control Procedures," *IEEE Transactions on Communications* **COM-23**, pp.891-904 (September 1975).

[Lam78a] S. S. Lam, "A New Measure for Characterizing Data Traffic," *IEEE Transactions on Communications* **COM-26**, pp.137-140 (January 1978).

[Lam79a] S. S. Lam, "Multiple Access Protocols," TR-86, Department of Computer Sciences, The University of Texas at Austin (January 1979). to appear in *Computer Communication: State of the Art and Direction for the Future*, W. Chou, editor, Prentice-Hall.

[Lam80a] S. S. Lam, "Packet Broadcast Networks — A Performance Analysis of the R-ALOHA Protocol," *IEEE Transactions on Communications* **COM-29(7)** (July 1980).

[Litt61a] J. Little, "A Proof of the Queueing Formula $L = \lambda W$," *Operations Research* **9(2)**, pp.383-387 (March 1961).

[Luen73a] D. G. Luenberger, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley, Reading, Mass. (1973).

[Mara80a] M. Marathe, "Design Analysis of a Local Area Network," *Proceedings of the Computer Networking Symposium*, pp.67-81 (December 10, 1980).

[Mara81a] M. Marathe, *private communications*, June 1981.

[Mark80a] J. W. Mark, "Distributed Scheduling Conflict-Free Multiple Access for Local Area Communication Networks," *IEEE Transactions on Communications* **COM-28(12)**, pp.1968-1976 (December 1980).

[Mart70a] J. Martin, *Teleprocessing Network Organization*, Prentice-Hall, Englewood Cliffs, N. J. (1970).

[Mass80a] J. L. Massey, "Collision-Resolution Algorithms and Random-Access Communications," UCLA-ENG-8016, School of Engineering and Applied Science, University of California, Los Angeles (April 1980).

[Metc76a] R. M. Metcalfe and D. R. Boggs, "Ethernet: Distributed Packet Switching for Local Computer Networks," *Communications of the ACM* **19(7)** (July 1976).

[Mitt81a] 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)**, pp.962-970 (July 1981). preprint: Department of Electrical Engineering, University of Toronto, 1980.

[Moll80a] M. L. Molle, *On the Capacity of Infinite Population Multiple Access Protocols*, UCLA Computer Science Department, University of California, Los Angeles (April, 1980). to appear in *IEEE Transactions on Information Theory*.

[Moll81a] M. L. Molle, "Bounds on the Capacity of Infinite Population Multiple Access Protocols," *Abstracts of Papers, Sixth International Symposium on Information Theory*, p.120 (February 9-12, 1981).

[Mose79a] J. Mosely, "An Efficient Contention Resolution Algorithm for Multiple Access Channels," LIDS-TH-918, Laboratory for Information and Decision Systems, MIT, Cambridge, Mass. (June 1979).

[Yosh77a] Y. Yoshioka, T. Nakamura, and R. Sato, "An Optimum Solution of the Queuing System," *Electronics and Communications in Japan* 60-B(8), pp.590-591 (August 1977).

[Pipp81a] N. Pippenger, "Bounds on the Performance of Protocols for a Multiple-Access Broadcast Channel," *IEEE Transactions on Information Theory* IT-27(2), pp.145-151 (March 1981). preprint: Research Report RC 7742(#33525), Mathematical Sciences Department, IBM Thomas J. Watson Research Center Yorktown Heights, New York (June 1979)

[Raws78a] E. G. Rawson and R. M. Metcalfe, "Fibernet: Multimode Optical Fibers for Local Computer Networks," *IEEE Transactions on Communications* COM-26(7), pp.983-990 (July 1978).

[Robe72a] L. G. Roberts, "ALOHA Packet System With and Without Slots and Capture," ASS Note 8 (NIC 11290), ARPA Network Information Center, Stanford Res. Inst., Menlo Park, Ca. (June 1972). reprinted in *Computer Communication Review*, Vol. 5, pp. 28-42 (April 1975).

[Robe72b] L. G. Roberts, "Extensions of Packet Communication Technology to a Hand Held Personal Terminal," *AFIPS Conference Proceedings, SJCC* 40, pp.295-298 (1972).

[Scho76a] M. Scholl, "Multiplexing Techniques for Data Transmission of Packet-Switched Radio Systems," UCLA-ENG-76123, UCLA Computer Science Department, University of California, Los Angeles (December 1976). Ph.D. dissertation.

[Schw77a] M. Schwartz, *Computer-Communication Network Design and Analysis*, Prentice-Hall, Englewood Cliffs, N.J. (1977).

[Toba74a] F. A. Tobagi, "Random Access Techniques for Data Transmission over Packet Switched Radio Networks," UCLA-ENG-7499, Computer Science Department, University of California, Los Angeles (December 1974). Ph.D. Dissertation.

[Toba75a] F. A. Tobagi and L. Kleinrock, "Packet Switching in Radio Channels: Part II. — The Hidden Terminal Problem in Carrier Sense Multiple-Access and the Busy Tone Solution," *IEEE Transactions on Communications* COM-23(12), pp.1417-1433 (December 1975).

[Toba79a] F. A. Tobagi and V. B. Hunt, "Performance Analysis of Carrier Sense Multiple Access with Collision Detection," *Proceedings of the Local Area Communication Network Symposium* (May 1979). also in Computer Networks, November 1980.

[Toba80a] F. A. Tobagi, "Multiaccess Protocols in Packet Communication Systems," *IEEE Transactions on Communications*, **COM-28**, pp.468-488 (April 1980).

[Toba80b] F. A. Tobagi, "Carrier Sense Multiple Access With Message-Based Priority Functions," Technical Report 200, Computer Systems Laboratory, Stanford University, Stanford, Ca. (December 1, 1980).

[Tsyb79a] B. S. Tsybakov, M. A. Berkovskii, N. D. Vvedenskaja, V. A. Mikhailov, and S. P. Fedorzov, "Methods of Random Multiple Access," *Fifth International Symposium on Information Theory* (July 7-9, 1979).

[Tsyb80a] B. S. Tsybakov and V. A. Mikhailov, "Ergodicity of a Slotted Aloha System," *Problems of Information Transmission*, Plenum Publishing Corp. (1980). translated from Russian.

[Tsyb80b] B. S. Tsybakov and V. A. Mikhailov, "Free Synchronous Packet Access in a Broadcast Channel with Feedback," *Problems of Information Transmission*, Plenum Publishing Company (1980). translated from Russian, original article in *Problemy Peredachi Informatsii*, Vol. 14, No. 4, pp. 32-59, October-December 1978.

[Tsyb80c] B. S. Tsybakov and M. A. Berkovskii, "Multiple Access with Reservation," *Problems of Information Transmission*, Plenum Publishing Corp. (1980). translated from Russian.

[Tsyb81a] B. S. Tsybakov and V. A. Mikhailov, "Packet Communications of Random Multiple Access Broadcast Channels," *Sixth International Symposium on Information Theory* (February 9-12, 1981).

[Uzga74a] R. C. Uzgalis, "Programming, Problem Solving and Communication, A Philosophical Look at the Software Crisis," Modeling and Measurement Note No. 28, Computer Science Department, University of California, Los Angeles (June 1974).

[Yemi79a] Y. Yemini and L. Kleinrock, "On a General Rule for Access Control or, Silence is Golden...," *Proceedings of the International Symposium on Flow Control in Computer Networks*, pp.335-347, North Holland Press (1979).

[Yemi80a] Y. Yemini, "On Channel Sharing in Discrete-Time Packet Switched, Multiaccess Broadcast Communication," UCLA-ENG-8061, Computer Science Department, University of California, Los Angeles (September 1980). Ph.D. Dissertation.