# A Queueing Analysis of a Buffered Block-Selective S-ARQ Protocol

by

Mohsinuddin Ansari

S.B., Mathematics
Massachusetts Institute of Technology, 1992

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Master of Science

and

Bachelor of Science in Electrical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1993



Author .....
Department of Electrical Engineering and Computer Science
May 17, 1993

Certified by .....
Robert Gallager
Fujitsu Professor of Electrical Engineering
Thesis Supervisor

Certified by .....
Michael Needham
Staff Engineer, Motorola, Inc.
Thesis Supervisor

Accepted by .....
Cam Searle
Chairman, Departmental Committee on Graduate Students

# A Queueing Analysis of a Buffered Block-Selective S-ARQ Protocol

by

Mohsinuddin Ansari

Submitted to the Department of Electrical Engineering and Computer Science
on May 17, 1993, in partial fulfillment of the
requirements for the degrees of
Master of Science
and
Bachelor of Science in Electrical Engineering

## Abstract

An approximate model is created and analyzed for a selective automatic-repeat-request (ARQ) protocol. The ARQ protocol uses a window to control the number of multiple-block messages that have access to the channel at one time. Messages must first be admitted into the window before being allowed to use the channel to send transmissions. The protocol is also block-selective in that the blocks that compose a message can be acknowledged individually, thus allowing following retransmissions of the message to include only the blocks that have not yet been received correctly. The stochastic model is developed as a particular Markov chain known as a quasi-birth-death process. The matrix-geometric method based on Neuts' Theorem is used to obtain the steady-state probability density function of the message occupancy. These results are compared to those found from a simulation of the protocol. The model is also used to observe the relationship between the maximum allowable offered load and the window size.

Thesis Supervisor: Robert Gallager
Title: Fujitsu Professor of Electrical Engineering

Thesis Supervisor: Michael Needham
Title: Staff Engineer, Motorola, Inc.

# Acknowledgments

I would like to thank Mike Needham for his encouragement, patience, and useful suggestions. He was always willing to set aside what he was doing to listen to and comment on my ideas. I could not have asked for a more responsive supervisor.

I appreciate Professor Robert Gallager's advice and perspective on the purpose of doing research and writing a thesis. I hope to carry his insight with me in my future endeavors.

Professor Leonard Gould's influence on my MIT career is immeasurable. For me, he has been an undergraduate advisor, a 6-A advisor, an investment consultant and strategist, and even a car salesman.

Having Arif, Mariam, Pamela, and Tasneem at MIT has made me realize how important it is to have family around. I will always remember the 1991 World Series as the best ever. Hopefully, we will all see each other often in the future.

I thank Irene, Jean, and John for being here this last term making it fun and not all work.

I would like to thank everyone in Chicago for making life there a great time. Inshallah, I will repay them for their hospitality. I thank Quadir Uncle and Kouser Auntie for their tolerance and the busy summers. I will miss being able to call Khalama and Asma Khalajan for support and visit them on the weekends.

At times I wonder if I am Husam's older brother or he is mine. It will be very different now after having him at Harvard for four years. I may actually miss him.

I would like to thank my parents for making me God-conscious and sending me to a private high school and college. Inshallah, from them I will realize that religion and education is more important than money. I dedicate this thesis to them.

# Contents

# List of Figures

# Chapter 1

# Introduction

The Motorola Integrated Radio System (MIRS) is a 64 kbps digital mobile radio system that represents the merging of cellular and trunking technologies [4] [16]. MIRS is being developed to meet the increasing demand for radio systems that provide a variety of new services such as telephone interconnection and data transmission. MIRS uses TDMA as its multiple access mode because, among other reasons, it can support full-duplex operation and has lower base station costs than FDMA. The system also uses a variation of the 16QAM modulation scheme that is more resistant to time dispersion. This new modulation scheme allows high speed signaling without the use of an equalizer. By incorporating TDMA, the new modulation scheme, and digital speech coding, MIRS is designed to support several hundred mobile and fixed-end users efficiently.

The purpose of the data link layer of the Motorola Integrated Radio System (MIRS) [5] is to ensure error-free delivery of packet data in the form of messages from a source node to a destination node. Transmitted data is not time critical but often requires error-free reception, unlike voice which must be sent in real time with or without errors.

Some of the various services that the data link layer can provide to the network layer include unconfirmed connectionless service, confirmed connectionless service, and confirmed connected service. In the unconfirmed service, the source node (sender) sends blocks continuously without expecting acknowledgments from the destination

node (receiver). In the confirmed services, the destination node sends acknowledgments to the source to inform it about which blocks have been received correctly and which need to be retransmitted due to errors. In the connected mode, the sequence of the packets must be preserved when sent to the network layer. Sequence preservation is not a concern in the connectionless mode.

Confirmed services require the destination node to request a retransmission when a message or part of a message is received in error. Retransmission protocols, also known as automatic repeat request (ARQ) schemes, vary in service (connectionless versus connected), efficiency, and complexity. The stop-and-wait protocol, the simplest confirmed-service ARQ protocol, requires the source to transmit a block, then stop transmitting to wait for an acknowledgment message. If a positive acknowledgment (ACK) is received, then the next block is transmitted. If a negative acknowledgment (NAK) is received, the previous block is retransmitted. The stop-and-wait protocol ensures error-free reception and correct sequence delivery but at a large cost in efficiency, since the channel is idle while waiting for an acknowledgment message. This can be a significant factor, especially for communication systems with long propagation delays.

The go-back-$N$ ARQ protocol is a continuous transmission scheme that also provides confirmed connected operation. The source continually sends blocks, not waiting for their acknowledgments. An acknowledgment for a block is expected at most $N$ blocks after it has been transmitted. When an ACK arrives for a previously sent block, the source sends the next block waiting to be transmitted. But when a NAK arrives (or no acknowledgment arrives), the source restarts transmission from the point in the sequence of the NAKed block which was the one sent $N$ blocks previously. Thus, the $N-1$ blocks after the NAKed packet are automatically retransmitted. The factor $N$ is determined by the round trip delay from the time of a block transmission to the time of reception of its acknowledgment message at the source. The continuous transmission feature of this protocol yields greater efficiency than the stop-and-wait procedure, but also adds complexity. Note that this protocol has some inefficiency because $N-1$ additional blocks are retransmitted automatically after a NAK or

time-out, even if some of these blocks were received correctly the first time.

The selective ARQ (S-ARQ) protocol is an even more efficient, but more complex, confirmed-service ARQ scheme. In this protocol, blocks are sent continuously, and, when a NAK arrives for a previously transmitted block, only that block is retransmitted. The added complexity includes the necessity of a resequencing buffer at the destination node, where blocks correctly received can be out of order and must wait for the preceding blocks before they can be sent to the network layer. In the ideal case, with infinite transmit and resequencing buffers, this protocol can achieve the maximum possible throughput of $1 - p$, where $p$ is the block error rate. This ideal case, however, cannot be achieved in practice due, in part, to the finite buffers and overhead data in the blocks and acknowledgments.

## 1.1 The Block-Selective Buffered S-ARQ Protocol

In the data link layer protocol considered for MIRS, data messages arrive at the source and are packetized into multiple-block messages consisting of a header block followed by data blocks. An arriving message must first gain admission to the window (or buffer) before it can use the channel. If there is an open position in the window at the time of its arrival, the message is immediately admitted. If there is no position available upon its arrival, the message must wait in a queue outside the window. After gaining admission, a message requests a transmission. Since there are other admitted messages using the channel, and only one transmission can be sent a time, a line of transmission requests forms. After a transmission has been sent, the message waits for an acknowledgment. Meanwhile, the channel continues to send transmissions from other admitted messages. The block-selective buffered ARQ protocol is a unique selective ARQ scheme in that the multiple-block messages are sent continuously from the source to the destination while bitmap acknowledgments, indicating which blocks of a received message (or partial message) were in error, are being sent from the destination to the source. Upon receiving the bitmap acknowl-

edgment, the message requests a new transmission consisting only of the blocks in error. This retransmission waits in line to be sent over the channel as before. If an acknowledgment has not arrived after a certain time period, a time-out occurs and the message requests its previous transmission again. This process continues until the entire message has been received correctly or a retry limit is reached. When either of these events happens, the message leaves the system creating an open position in the window for another message to gain admission. The block-selective ARQ scheme offers an improvement in efficiency over the normal selective ARQ scheme by reducing the overhead required for transmissions and acknowledgment messages compared to blocks being sent and acknowledged individually. The window is a mechanism of flow control and is used at the source node to limit the number of messages that can use the channel. Consequently, the receiver also needs a window of the same size to hold partial messages.

Various selective ARQ protocols have been analyzed in existing literature, but analysis on block-selective ARQ schemes with window buffering is limited. Konheim [11] creates an exact model of a slotted system with single-block messages. Unfortunately, with his model, obtaining numerical results quickly becomes unwieldy because, as the round trip delay increases, the number of states in his model grows exponentially. As a result of Konheim's work, later analyses focus on creating approximate models. Kaul [9], Anagnostou et al. [2], Saeki et al. [15], and Lee et al. [12] each develop distinct approximate models for a slotted system with single-block messages. Ahmadi [1] analyzes the Checkpoint Mode (CPM) protocol which is a selective ARQ scheme that has a checkpoint block, sent periodically from the receiver to the sender, that positively or negatively acknowledges a fixed number of blocks. Unfortunately, only the throughput of this protocol is studied in the paper. The block-selective buffered ARQ protocol studied here differs from the analyzed protocols mentioned above, except CPM, in that the messages contain multiple blocks that are acknowledged concurrently, as opposed to being acknowledged individually. This protocol differs from CPM in that the acknowledgment messages can acknowledge a variable number of blocks at one time, whereas in CPM, only a fixed number of blocks can be

acknowledged with one checkpoint block.

This thesis presents the background, development, and analysis of an approximate queueing model for the block-selective buffered ARQ protocol. The first few sections discuss relevant queueing theory tools and results. The developed S-ARQ model is explained and used to obtain statistics on message occupancy as it varies with the window size. These results are compared to those obtained from simulation. Finally, the model is used to find a relation between the offered load and the window size which may be useful in determining efficient dynamic bandwidth allocation.

# Chapter 2

# Preliminaries

Queueing systems are a class of a more general group of systems known as flow systems. In a flow system, some commodity moves from a source to a destination through one or more finite-capacity channels. In a queueing system, the commodity is a customer that requires some service, and the channels provide that service. Customers arriving while the server or servers are busy join a queue and wait for a server to become free. One simple example is the teller service at a bank where an arriving customer requires service to complete his or her desired transactions. If a teller is free upon arrival, the customer is served immediately. Otherwise the customer must wait in line. In a communications system, the commodity is packet data and the service is the transmission of these packets from a source to a destination. In most queueing systems, the arrival times and service time of each customer are random variables. Consequently, the number of waiting customers in a system (also known as the queue length) at any given time is random. Occupancy will be used to denote the number of customers both waiting and being served. The objective of queueing theory is to provide analytic tools to obtain statistics about the occupancy and other random variables associated with the system. In addition, queueing analysis can often give insight into the behavior of a system than cannot be gained from simulation.

The understanding of stochastic processes is essential to the study of queueing theory because the random variables of interest are dependent on time. A stochastic process is a sequence of random variables which may also depend on the parameter

indexing the sequence. A particular queueing system has many relevant stochastic processes each of which has its own random variables. For example, the queue length process measures the occupancy of a system as it changes with time and for this reason, the queue length process is often referred to as the occupancy process. Another process is the arrival process which measures the number of customer arrivals to the system over time. In this chapter, a short introduction to stochastic processes, specifically Markov chains, is given with emphasis on those processes pertinent to queueing theory. Certain properties and types of Markov chains that will be useful for subsequent analysis are explained. Also a special class of probability distributions, known as phase-type distributions, are discussed. The hyperexponential distribution, which is a particular phase-type distribution, will be used later in the modeling of the buffered S-ARQ scheme.

## 2.1 Markov Chains

In a communications system, statistics on the number of messages in the system, or message occupancy, are necessary to determine approximately how much space is required to store the messages waiting to be sent. The message occupancy varies randomly with time and can, therefore, be described by a stochastic process. The message occupancy at any time $t$ is considered as a random variable, $X(t)$. A stochastic (or random) process is classified by the set of random variables $X(t)$ (which may be vectors) and the statistical dependencies among the variable $X(t)$ for different values of time $t$. The sample space is the set of possible values that $X(t)$ may take on. When the number of possible values in this sample space is finite or countable, then the random process is a discrete random process. Although the index parameter may be either discrete or continuous, all stochastic processes discussed here will have continuous time as an index parameter.

Markov chains are an important class of random processes. A random process is a Markov chain if the state space is discrete and the probability of the next state value being $X(t_{n+1})$ depends only on the current state $X(t_n)$ and not on previous state

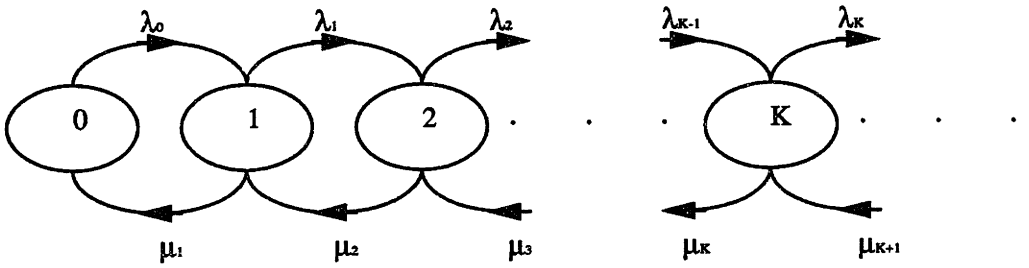



Figure 2-1: The State-Transition Diagram for a Birth-Death Process.

values. In other words, the state of a Markov chain summarizes the effect of the past on the future. The Markov property [10] can be expressed analytically as:

**Definition 1** *The random process $X(t)$ forms a continuous-time Markov chain if, for all integers $n$ and for any sequence $t_1 < t_2 < \cdots < t_{n+1}$,*

$$P[X(t_{n+1}) = j \mid X(t_1) = i_1, X(t_2) = i_2, \ldots, X(t_n) = i_n] = P[X(t_{n+1}) = j \mid X(t_n) = i_n] \tag{2.1}$$

Every continuous-time Markov chain has a transition matrix $Q(t)$ associated with it. The elements $q_{ij}(t)$, $i \neq j$, of $Q(t)$ are the transition rates from state $i$ to state $j$ given that the system is currently in state $i$. By definition, $-q_{ii}(t) = (\sum_{j \neq i} q_{ij})$, is the departure rate from state $i$ given that the system is currently in state $i$. As a result of this definition, the elements in each row of a transition matrix sum to zero and the diagonal terms are non-positive. A homogeneous Markov chain is defined to have a transition matrix $Q(t) = Q$ that is independent of time. As a result, the individual transition rates and departure rates of a homogeneous Markov chain are also independent of time.

Birth-death processes are a class of Markov chains that restrict state transitions to occur only between neighboring states. A partial state-transition diagram of a birth-death process is shown in Figure 2-1. A birth occurs with an arrival of a new customer, so that the state (or queue length) increases by one. A death occurs when a customer currently in the system exits, decreasing the state by one. Only one event, either a single birth or a single death, is allowed to occur at a time. In Figure 2-1,

the bubbles represent the states of the process, and the values in the bubbles indicate the queue length of the system. When the queue length is $K$, the rate at which a new customer will arrive is $\lambda_K$, and the rate at which the customer in service will exit is $\mu_K$. The queue length process of the simplest queue, the M/M/1 queue, is a birth-death process. An M/M/1 queue has interarrival times and service times of customers that are exponentially distributed (but not always with the same mean).

To illustrate an example of a transition matrix, consider a homogeneous birth-death process. Since transitions are only allowed between neighboring states, $q_{ij} = 0$ for $|i-j| > 1$. A birth from state $i$ occurs with rate $q_{ij} = \lambda_i$ for $i = j-1$, $i \geq 0$, and a death from state $i$ occurs with rate $q_{ij} = \mu_i$ for $i = j+1$, $i \geq 1$. Since $-q_{ii} = \sum_{j \neq i} q_{ij}$, $q_{ii} = -(\lambda_i + \mu_i)$ for $i \geq 0$. The transition matrix is tridiagonal

$$Q = \begin{pmatrix} -\lambda_0 & \lambda_0 & & & \\ \mu_1 & -(\lambda_1+\mu_1) & \lambda_1 & & \\ & \mu_2 & -(\lambda_2+\mu_2) & \lambda_2 & \cdots \\ & & \mu_3 & -(\lambda_3+\mu_3) & \cdots \\ & & \cdot & \cdot & \\ & & \cdot & \cdot & \\ & & \cdot & \cdot & \end{pmatrix} \tag{2.2}$$

The death rate $\mu_0$ equals zero since, when the state is zero, no customers are in the system so no deaths can occur.

As mentioned earlier, in analyzing queueing systems, buffer occupancy is of great interest. Ideally, one would like to obtain the probability density of this random variable in steady state. The matrix differential equation for the state probability vector, $x(t)$, and a Markov chain with transition matrix $Q(t)$,

$$\frac{dx(t)}{dt} = x(t)Q(t) \tag{2.3}$$

is called Kolmogoroff forward equation. To find the stationary probability vector $x$, which yields the steady-state probability density of the random variable, the left-

hand side of the above equation must be zero. Assuming that the Markov chain is homogeneous, equation (2.3) reduces to

$$xQ = 0 \tag{2.4}$$

The solution for $x$ in the above equation must be normalized so that the elements of $x$ sum to one.

Earlier it was mentioned that, for a Markov chain, $-q_{ii}$ is the departure rate from state $i$ given that the current state is $i$. Consider the random variable that describes the length of time the stochastic process stays in a particular state before making a transition to a different state. From the Markov property, recall that the entire previous history of the process relevant to its future is summarized in the specification of its current state but not in the specification of the length of time the process has been in that state. As a result, it can be seen that the distribution for the time spent in a particular state must be "memoryless." The only continuous distribution with this memoryless property is the exponential distribution. The memoryless property implies that if the time between the moment the process enters a state and the moment the process exits the state is exponentially distributed, and if the process has not left the state after some time $\tau$, then the time remaining until the process leaves the state is still exponentially distributed with the same mean. The fact that $\tau$ time units have expired since the process has entered the current state does not change the distribution of the time to departure from that state. At the time $\tau$, the process behaves as if it had just entered the state. Using equation (2.1), the memoryless property can be stated as

$$P[X(t_2) = j \mid X(t_1+\tau) = k, X(t_1) = k] = P[X(t_2) = j \mid X(t_1+\tau) = k], \quad t_2 > t_1+\tau > t_1 \tag{2.5}$$

Poisson processes are a special class of birth-death processes that have a constant (state-independent) birth rate and a zero death rate and play an important role in queueing theory. If $\lambda_k = \lambda$ and $\mu_k = 0$ for $k \geq 0$ in Figure 2-1, the process would

be a Poisson process. Since a Poisson process is a Markov chain, it must also exhibit the memoryless property which means that the time spent in a specific state before a transition is exponentially distributed. Because a Poisson process is a pure birth process with a zero death rate, the time spent in each state is also the time between two consecutive arrivals. In most queueing systems, the arrival process is assumed to be a Poisson process. This leads to an important result in queueing theory that, if the arrival process is a Poisson process, then the interarrival times (times between two consecutive arrivals) are exponentially distributed.

## 2.2 Phase-Type Probability Distributions

Due to the memoryless property of Markov chains, any queueing system with interarrival times and service times exponentially distributed (not necessarily all with the same mean) can be represented by a Markov chain and its stationary probability vector may be found from existing methods[1]. Rarely, however, does a queueing system have such compatible distributions. Imposing such severe restrictions such as exponential interarrival and service time distributions on the models for these systems often yields inadequate results. On the other hand, modeling complex queueing systems with their exact distributions, if they are even known, is usually futile. Phase-type (PH) distributions, which are a special class of probability distributions, allow for the modeling of some complex queueing systems by creating Markov chains that can be analyzed.

Consider the queueing system shown in Figure 2-2. The arrival process is Poisson and the server has two stages. The two stages are cascaded together where the first stage is an exponential server of mean $s_1 = \frac{1}{\mu_1}$ and the second stage is an exponential server of mean $s_2 = \frac{1}{\mu_2}$. The two stages are also said to be in series.

At the beginning of its service, the customer enters the first stage. While the customer is in the first stage, the second stage remains idle. After some length of

[1] Kleinrock discusses methods for solving the stationary probability vectors of Markov chains in [10]

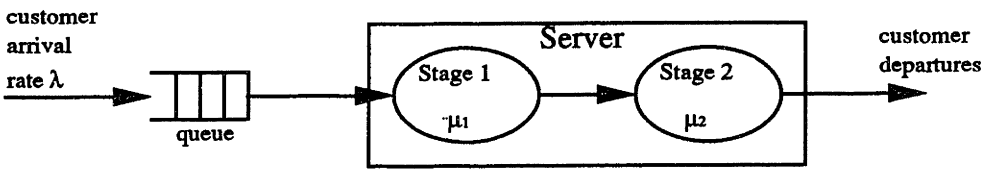



Figure 2-2: The Two-Stage Cascaded Server Queueing System.

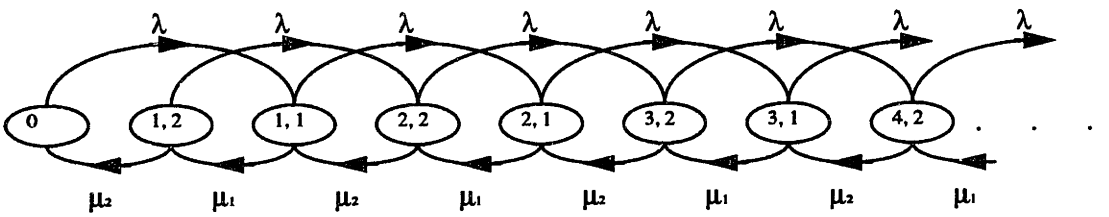



Figure 2-3: The State-Transition Diagram for the Joint Queue Length-Service Stage Process of the Two-Stage Cascaded Server Queueing System.

time that is exponentially distributed with mean $s_1$, the customer leaves the first stage and enters the second stage to continue its service. At this moment, the first stage becomes idle. The first stage does not begin service on the next customer in line, so there is never more than one customer in the two-stage server. After the second stage service is completed, the customer leaves the system, and only now is the next customer permitted to enter its first stage of service. One possible state-transition diagram for this queueing system is shown in Figure 2-3. The first value in each bubble is the occupancy of the system. The second value is the stage of the current customer in service. When a new customer arrives, the occupancy increases by one, but the stage stays the same. When a customer ends stage one, it moves to stage two without changing the occupancy. But when a customer leaves stage two, it also leaves the system, so the occupancy decreases by one, and the customer at the head of the line (if there is one) begins its service by entering stage one. The state of the process is not a scalar quantity but is a two-dimensional vector that maintains the customer occupancy and the stage of the customer currently being served. The process is a Markov chain because the Markov property is satisfied. That is, the effect

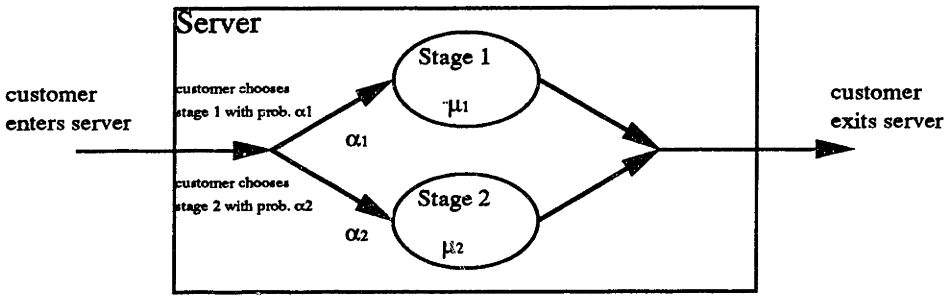



Figure 2-4: The Two-Stage Hyperexponential Server.

of the history of the process on the future is summarized in the current occupancy and the current stage of service because all state departure times are exponentially distributed. In joint processes, the first quantity of the state vector is usually the occupancy of the system, and the second quantity is often referred to as the phase of the system.

The two-stage-in-series service distribution described above where each stage follows an exponential distribution is an example of a phase-type distribution. By allowing the state variable to be a vector that keeps track of the occupancy and the current service stage, a Markov chain is created. This example exhibits the power of phase-type distributions in stochastic modeling. The cascaded server queue will be discussed in more detail in the next chapter.

The hyperexponential distribution is another kind of phase-type distribution. If the service time distribution is a two-stage hyperexponential distribution, then a customer requiring service chooses a type one service with probability $\alpha_1$ or a type two service with probability $\alpha_2 = 1 - \alpha_1$. Type one service is exponentially distributed with mean $s_1$ and type two service is exponentially distributed with mean $s_2$. This server is depicted in Figure 2-4. Similar to the queueing system with the cascaded server, a queueing system with the hyperexponential server can be modeled as a Markov chain by allowing the state vector to maintain the occupancy and the current type of service being performed. This kind of parallel server will be used in the S-ARQ model.

The common characteristic of both the cascaded server and the parallel server is that each stage of the servers is individually exponentially distributed. In fact, a new server with a more complex PH distribution can be created by forming any combination of cascaded and/or parallel servers as long as each stage by itself is exponentially distributed. This requirement makes phase-type distributions useful in queueing analysis because, due to the memoryless property, an appropriate Markov chain and state vector can be found.

Before proceeding, the distinction between the phase of the system and a phase-type distribution must be made clear. The phase is the second variable in the state of a two-dimensional vector Markov chain. The phase does not exist for scalar Markov chains. As explained above, a phase-type distribution is any distribution that can be constructed from exponentially distributed stages in series and/or in parallel. Although there is a relation between the phase and a phase-type distribution, it is not necessary to know this relation to understand the model to be discussed here.

# Chapter 3

# The Matrix-Geometric Method

The arrival and service rates of simple queueing systems such as the M/M/1 queue depend only on the queue length (or customer occupancy) of the system. But in more complex systems, the arrival and service rates depend on other factors as well. In the example of the cascaded server queue, the departure rates depend on the service stage. In another example, the arrival and service rates of a queueing system may depend on the time of day. In the course of the day, there may be busy periods and slow periods. During busy periods, customers arrive and are served at higher rates. During slow periods, customers arrive at a lower rate, and their service rates may also be reduced. The current customer arrival rate and service rate depend on the current phase, busy or slow, of the system.

The matrix-geometric method is an approximate method that allows for the analysis of a certain queueing models by incorporating the phase, which follows a phase process, with the queue length to create a vector random process known as a quasi-birth-death (QBD) process. A QBD process is a Markov chain that can be analyzed using Markov chain theory. This chapter begins by discussing the phase and QBD processes. The matrix-geometric method is then outlined leading to Neuts' Theorem. Next, an application of Neuts' Theorem, developed by Katz, is explained. Finally, the use of phase-type distributions in QBD processes is explained.

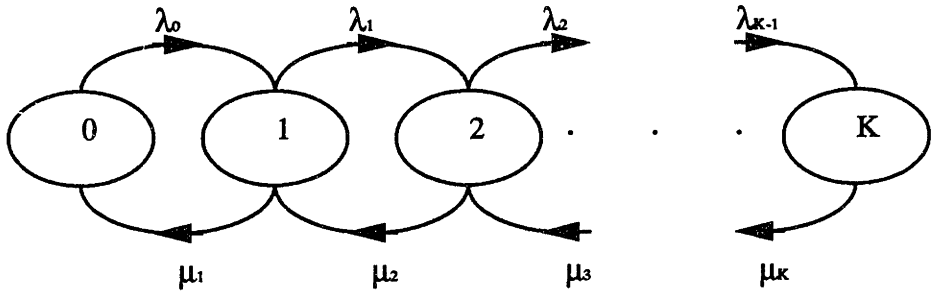



Figure 3-1: A State-Transition Diagram for a Possible Phase Process.

## 3.1 The Phase Process and Quasi-Birth-Death Process

The queue length (or occupancy) process has as its state the number of customers in the system. Some models of queueing systems also include a phase process, which is a continuous-time finite-state Markov chain with a transition matrix $S$. A state-transition diagram of a possible phase process in shown in Figure 3-1. If the phase process has $K+1$ possible phases, then $S$ has dimensions of $(K+1)$x$(K+1)$. In order to use the matrix-geometric method, the number of phases must be finite. Although the example of the phase process depicted in Figure 3-1 is a finite-state birth-death process, in general, the phase process can be any finite-state Markov chain. That is, the matrix-geometric method can still be used when transitions are allowed between non-neighboring phases.

The joint occupancy-phase vector process of a queueing system with the arrival and service rates that depend both on the occupancy and the phase is called a quasi-birth-death (QBD) process[1]. A QBD process is a vector process on the state space $E = \{(i,j) : i \geq 0, 0 \leq j \leq K\}$ where $i$ is the current occupancy and $j$ is the current phase. Since the QBD process is a Markov chain, the memoryless property holds and the residency times (the time the process spends in a state before departing

[1]Daigle [6] gives a good introductory treatment of quasi-birth-death processes. Neuts [13] coverage is more advanced.

from that state) are exponentially distributed for all states. A partial state-transition diagram for a QBD process having the same phase process in Figure 3-1 is illustrated in Figure 3-2. The first component of each bubble is the occupancy of the system and the second component is the phase. For the QBD process of Figure 3-2, the phase transitions (horizontal changes) are independent of the occupancy and the occupancy transitions (vertical changes) are independent of the phase. In general, however, this need not be true for a QBD process. The transition matrix $\tilde{Q}$ of a QBD process has the form

$$\tilde{Q} = \begin{pmatrix} B_0 & A_0 & & & & \\ B_1 & A_1 & A_0 & & & \\ & A_2 & A_1 & A_0 & & \\ & & A_2 & A_1 & A_0 & \cdots \\ & & & A_2 & A_1 & \cdots \\ & & & \cdot & \cdot & \\ & & & \cdot & \cdot & \\ & & & \cdot & \cdot & \end{pmatrix} \tag{3.1}$$

The elements of each row of $\tilde{Q}$ must sum to zero so

$$(B_0 + A_0)e = (B_1 + A_1 + A_0)e = (A_0 + A_1 + A_2)e = 0 \tag{3.2}$$

For brevity in notation, $e$ denotes a column vector of ones of the appropriate length for valid multiplication. In equation (3.2), '0' is a column vector of zeros. Later, '0' will be used to denote the scalar zero, a vector of zeros, or a matrix of zeros, and its exact usage should be clear from the context of the equation. Each element of $\tilde{Q}$ shown in equation (3.1) is a matrix and is referred to as a block matrix or submatrix. $\tilde{Q}$ is considered a block-partitioned matrix.

The key characteristic of a QBD process, which can be observed from $\tilde{Q}$, is that transitions can only take place within an occupancy state (a phase transition alone) or between two neighboring occupancy states. Although the QBD process of Figure 3-2 does not have any simultaneous transitions in occupancy and phase, such transitions

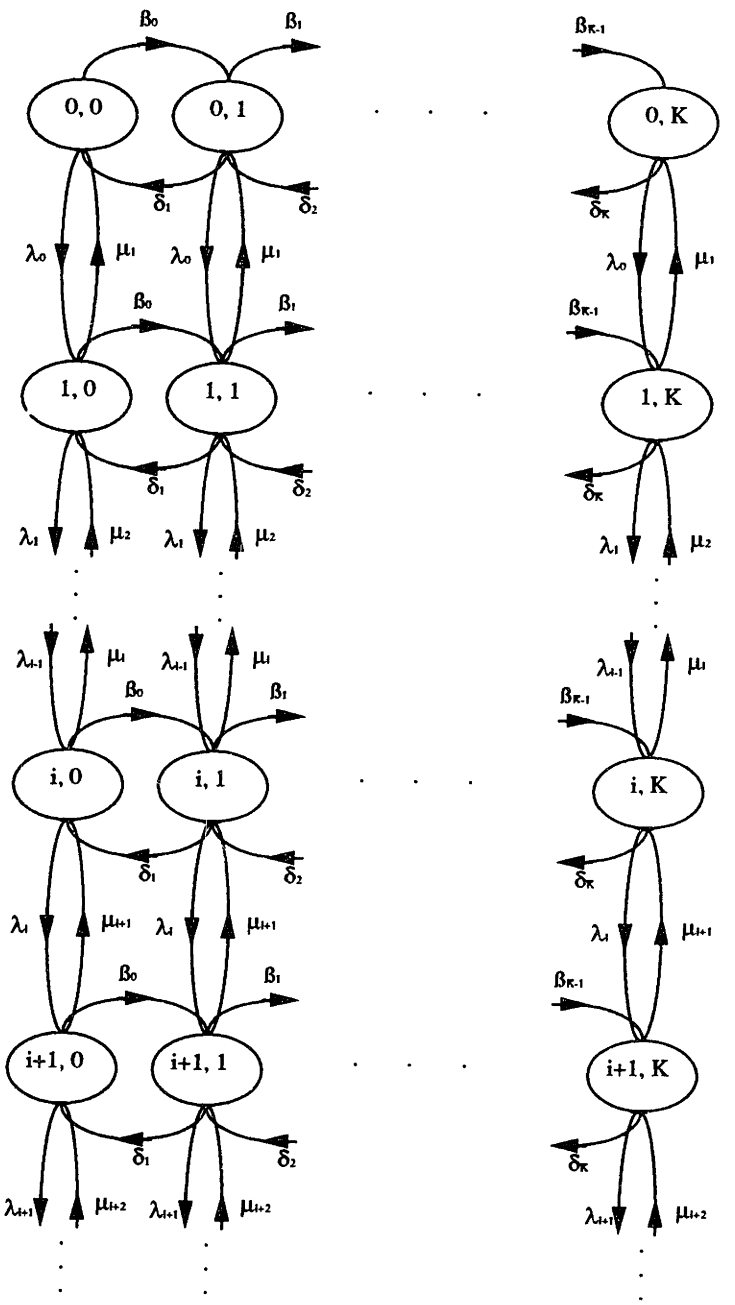



Figure 3-2: A Partial State-Transition Diagram for a Possible Quasi-Birth-Death Process.

are allowed and will be discussed later in the chapter. $\tilde{Q}$ is a block tridiagonal matrix, which parallels the tridiagonal transition matrix of an ordinary birth-death process. Similar to a birth-death process, where the element $q_{i,i+1}$ on the upper diagonal of $Q$ represents the transition rate that increases the occupancy from $i$ to $i+1$, the upper diagonal block matrix $A_0$ of $\tilde{Q}$ contains the rates for all transitions that increase the occupancy from $i$ to $i+1$. Thus, the birth-death aspect of a quasi-birth-death process can be seen. The $(v,w)$th submatrix of $\tilde{Q}$ holds the rates of transitions from any phase of occupancy $v$ to any phase of the occupancy $w$. For example, the (1,3)th submatrix of $\tilde{Q}$ is the zero matrix indicating that there is no allowed state transition that can change the occupancy from one to three. Unlike a simple birth-death process, the diagonal block matrices of $\tilde{Q}$ will not have all negative elements because it is possible to have transitions in which only the phase changes while the current occupancy stays the same. The diagonal elements of the diagonal block matrices will, however, be negative because these elements make up the main diagonal of $\tilde{Q}$. Since, at most, $K+1$ phases exist at each occupancy level, the block matrices are of size $(K+1)$x$(K+1)$. It will later be seen that when complex behavior exists for some boundary occupancy states, some of the phases are extraneous, and the corresponding block matrices may be reduced in size. In the most general QBD process, each block matrix is of size $(K+1)$x$(K+1)$.

## 3.2 Neuts' Theorem

As stated earlier, a QBD process is a vector Markov chain on the state space $E = \{(i,j) : i \geq 0, 0 \leq j \leq K\}$ where $i$ is the occupancy and $j$ is the phase and $K$ is the maximum phase. Therefore, the stationary probability vector satisfies the equilibrium equation,

$$x\tilde{Q} = 0, \tag{3.3}$$

where, in steady-state,

$$x = (x_{00}, x_{01}, ..., x_{0K}, x_{10}, x_{11}, ..., x_{1K}, ...) \tag{3.4}$$

and

$$x_{ij} = P(X = i, Y = j), \quad i \geq 0, \quad 0 \leq j \leq K. \tag{3.5}$$

The stationary probability vector, $x$, can also be partitioned as

$$x = (x_0, x_1, x_2, ...) \tag{3.6}$$

where

$$x_k = (x_{k0}, x_{k1}, ..., x_{kK}), \quad k \geq 0. \tag{3.7}$$

Suppose there exists a matrix $R$ such that

$$x_k = x_0 R^k, \quad k \geq 0 \tag{3.8}$$

A solution of the form of equation (3.8), is called a matrix-geometric solution. Neuts' Theorem [13] states criteria that must be satisfied by the QBD process for a matrix geometric solution to exist.

**Theorem 1 (Neuts)** *The process $\tilde{Q}$ is positive recurrent if and only if the minimal nonnegative solution $R$ to the matrix-quadratic equation*

$$R^2 A_2 + R A_1 + A_0 = 0 \tag{3.9}$$

*has all its eigenvalues inside the unit disk and the finite system of equations*

$$x_0(B_0 + RB_1) = 0 \tag{3.10}$$

$$x_0(I - R)^{-1} e = 1 \tag{3.11}$$

*has a unique positive solution $x_0$. If the matrix $A = A_0 + A_1 + A_2$ is irreducible, then $sp(R) < 1$ if and only if*

$$\pi A_2 e > \pi A_0 e, \tag{3.12}$$

*where $\pi$ is the stationary probability vector of A. The stationary probability vector $x = (x_0, x_1, ...)$ of $\tilde{Q}$ is given by*

$$x_k = x_0 R^k, \ for \ k \geq 0. \tag{3.13}$$

The matrices $A_0$, $A_1$, $A_2$, $B_0$, and $B_1$ refer to the submatrices of $\tilde{Q}$ shown in equation (3.1). The notation $sp(R)$ denotes the spectral radius of the matrix $R$ which is the magnitude of the largest eigenvalue of $R$. For the steady-state probability vector to exist, all of the eigenvalues of $R$ must have a magnitude of less than one.

Positive recurrence means that there are no transient states. For all states that the process can start in, the probability of returning to that state is one. A process is irreducible if any state can be reached from any other state (but not necessarily directly). As long as the transition matrix $\tilde{Q}$ cannot be written in triangular form, the process is irreducible. Since the S-ARQ model to be presented will meet the criteria for positive recurrence and irreducibility, these criteria will not be discussed further. Equation (3.10) is a boundary condition for the first two occupancy levels, and equation (3.11) is the normalization constraint that requires the sum of all steady-state probabilities in $x$ to equal unity. Condition (3.12) is a stability requirement and is equivalent to stating that the maximum average service rate must be greater than the average arrival rate.

## 3.3 An Example of a Stochastic Matrix-Geometric Model

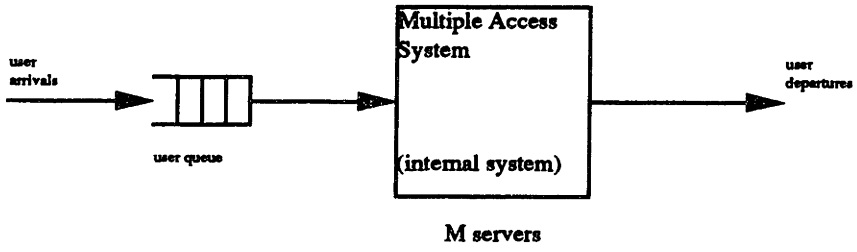
Sharlene Katz [8], in her Ph. D. dissertation, models a multi-access communications system in which multiple users share a channel and a buffer (or window) to send activities of random length. In her analysis, Katz uses the matrix-geometric method to study the effects of session admission control on the multiple access channel.

The session admission control is a window that limits the number of users that can share the channel at any one time. A user arrives randomly and, if there is an

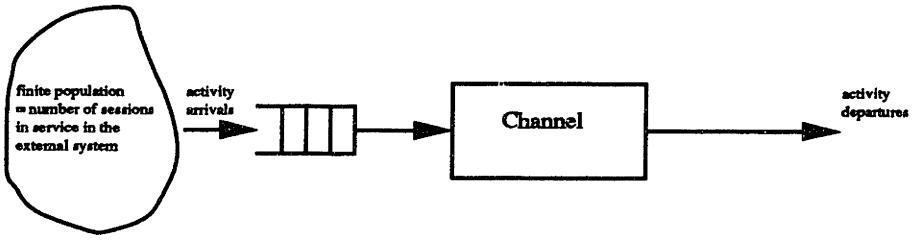
open position in the window, enters the window. If no position is open, the user joins a queue outside the window, waiting for a spot in the window to open so that it can use the channel. Queued users are admitted into the window on a first-come-first-serve basis. Once a user has been admitted to the window, it can generate an activity to send over the channel. The channel can only send one activity at a time, so when the channel is busy, newly generated activities from other admitted users must wait in line and receive service on a first-come-first-serve fashion by the channel. An admitted user is allowed to have at most one outstanding activity, which is either currently being handled by the channel or is waiting in line for the channel. Consequently, if the maximum window size is $M$, there can be no more than $M$ outstanding messages. A user with an outstanding activity is considered active and cannot generate additional activities. Once the user's activity has been completely sent, the user becomes inactive and can either exit the system or generate another message to send over the channel. When an admitted user exits the system, its position in the window is filled by a user at the head of the outside queue if the queue is not empty.

### 3.3.1 Description of the Model

Katz models the system described above by considering it as a combination of an external system and an internal system. The user occupancy of the external system (or external user occupancy) is the number of users in the entire system. The user occupancy includes the users admitted in the window as well as those queued waiting to enter the window. The phase of the internal system (or internal phase) is the number of outstanding activities (or active user occupancy of) in the window waiting to be served including the one currently being served by the channel. Because each outstanding activity comes from one active user in the window, the internal phase is also the number of active users in the window. Since the window size is fixed, the number of active users is finite and bounded by the size of the window. The external and internal queueing systems are shown in Figure 3-3. Katz assumes that all interarrival and service times are exponentially distributed (but not with the same




(a) External System




(b) Internal System

Figure 3-3: The External and Internal Systems of Katz's Model.

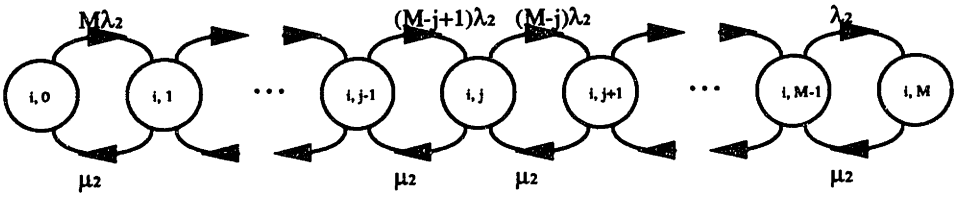



Figure 3-4: The State-Transition Diagram for Katz's Phase Process with a full window ($i \geq M$).

mean). Consequently, Katz finds a QBD process that maintains the user occupancy and active user occupancy (which is the phase). Katz uses a variation of Neuts' Theorem to analyze the system and find a matrix-geometric solution.

Let the window size be $M$ so that a maximum of $M$ users can share the channel at one time. The window is considered full when all $M$ slots are occupied, which occurs when the external user occupancy is at least $M$. The arrival rate of new users to the external system is $\lambda_1$. An admitted user can exit the system (and the window) only when it is inactive and does so at a rate of $\mu_1$. Also, an admitted user can generate a new activity only when it is inactive at a rate of $\lambda_2$. When the internal phase is $j$, and the window is full (so the external user occupancy is $i > M$), the overall user departure rate is $(M-j)\mu_1$ because there are $M-j$ inactive users in the window, each behaving independently. For the same reason, the overall activity generation rate is $(M-j)\lambda_2$ when the window is full and the internal phase is $j$. The state-transition diagram for the phase process when the window is full is shown in Figure 3-4. When the external user occupancy $i$ is less than $M$, however, the window is not full, and the number of inactive users in the window is only $i-j$. Consequently, the user departure rate and activity generation rate are $(i-j)\mu_1$ and $(i-j)\lambda_2$ respectively. Because the internal system exhibits different behavior when $0 \leq i < M$, these occupancy states are called boundary states. A partial state-transition diagram of Katz's QBD process is illustrated in Figure 3-5. In the next section, it is shown that slight modifications are required to do the analysis of Katz's QBD process to find a matrix-geometric solution due to the existence of the boundary states.

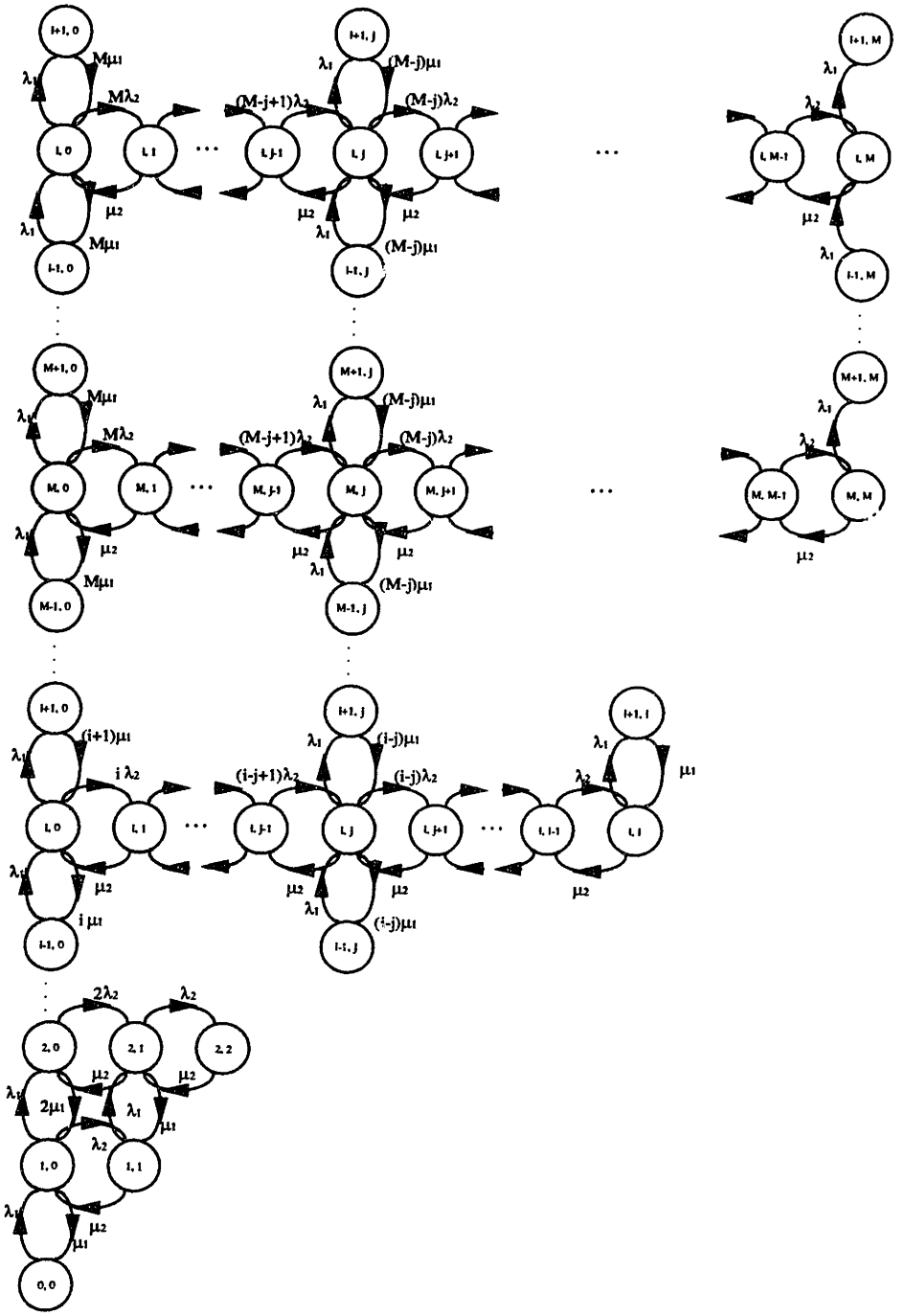


Figure 3-5: The Partial State-Transition Diagram for Katz's Quasi-Birth-Death Process.

### 3.3.2 Analysis of the Model

The quasi-birth-death process of Figure 3-5 is a two-dimensional Markov chain with a state space of $E = \{(i,j) : i \geq 0, 0 \leq j \leq M\}$, where $i$ is the external user occupancy (both in the window and in the queue), and the phase $j$ is the number of outstanding activities (or active users) in the internal system.

Let $x$ be the stationary probability vector, where

$$x = \{x_{00}, x_{01}, ..., x_{0M}, x_{10}, ..., x_{1M}, ..., x_{M0}, ..., x_{MM}, x_{M+1,0}, ..., x_{M+1,M}, ..., x_{i0}, ..., x_{iM}, ...\} \tag{3.14}$$

so that $x_{ij}$ equals the stationary probability of the system having a user occupancy $i$ and active user occupancy (or phase) $j$. As can be seen from Figure 3-5, $j$ can never exceed $i$ since each admitted user is allowed at most one outstanding activity at any time. As a result, the steady-state probabilities associated with these states will be zero in the matrix-geometric solution. As before, the vector $x$ can also be partitioned as

$$x = \{x_0, x_1, ..., x_{M-1}, x_M, ...\} \tag{3.15}$$

where

$$x_k = \{x_{k0}, x_{k1}, ..., x_{kM}\} \tag{3.16}$$

The stationary probability vector $x$ must satisfy the steady-state matrix equation

$x\tilde{Q} = 0$, where

$$\tilde{Q} = \begin{pmatrix} B_{00} & B_{01} & & & & & & & \\ B_{10} & B_{11} & B_{12} & & & & & & \\ & B_{21} & B_{22} & B_{23} & \cdots & & & & \\ & & B_{32} & B_{33} & \cdots & & & & \\ & & \cdot & \cdot & & & & & \\ & & \cdot & \cdot & & & & & \\ & & \cdot & \cdot & & & & & \\ & & & B_{M-1,M-2} & B_{M-1,M-1} & B_{M-1,M} & & & \\ & & & & A_2 & A_1 & A_0 & & \\ & & & & & A_2 & A_1 & A_0 & \cdots \\ & & & & & & A_2 & A_1 & \cdots \\ & & & & & & \cdot & \cdot & \\ & & & & & & \cdot & \cdot & \\ & & & & & & \cdot & \cdot & \end{pmatrix} \tag{3.17}$$

is the transition matrix of the QBD process. The submatrices $B_{ij}$, $A_2$, $A_1$, $A_0$ are $(M+1)$x$(M+1)$ matrices. This transition matrix has a different form than that shown in equation (3.1) to account for the boundary states of Katz's QBD process. The $(m,n)$th element of $B_{gh}$ is the transition rate from the external user occupancy $g$ and the internal phase $m$ to the external user occupancy $h$ and the internal phase $n$. Similarly, the $(m,n)$th elements of $A_2$, $A_1$, and $A_0$ are the transition rates from the external user occupancy $i$ to the external user occupancy $i-1$, $i$, and $i+1$, respectively, for $i \geq M$ and from the internal phase $m$ to the internal phase $n$. For a submatrix, $m$ refers to the $(m+1)$th row of the submatrix, and $n$ refers to the $(n+1)$th column of the submatrix for $0 \leq m,n \leq M$. The elements of the submatrices of $\tilde{Q}$ can be found from the state-transition diagram in Figure 3-5. The submatrices are (for $0 \leq m,n \leq M$)

$$B_{00}(m,n) = \begin{cases} -\lambda_1, & m = n = 0 \\ 0, & otherwise \end{cases} \tag{3.18}$$

$$B_{01}(m,n) = \begin{cases} \lambda_1, & m = n = 0 \\ 0, & otherwise \end{cases} \tag{3.19}$$

$$B_{k,k-1}(m,n) = \begin{cases} (k-m)\mu_1, & m = n < k \\ 0, & otherwise \end{cases} \qquad for\ 1 \le k \le M \tag{3.20}$$

$$B_{k,k}(m,n) = \begin{cases} (k-m)\lambda_2, & m = n-1,\ 1 \le n \le k+1 \\ \mu_2, & m = n+1,\ 1 \le m \le k \\ -(\lambda_1 + \mu_2 + (k-m)(\mu_1 + \lambda_2)), & m = n,\ 1 \le m \le k \\ -(\lambda_1 + k(\mu_1 + \lambda_2)), & m = n = 0 \\ 0, & otherwise \\ for\ 1 \le k \le M \end{cases} \tag{3.21}$$

$$B_{k,k+1}(m,n) = \begin{cases} \lambda_1, & m = n \le k \\ 0, & otherwise \end{cases} \qquad for\ 1 \le k \le M \tag{3.22}$$

Also, $A_0 = B_{M,M+1}$, $A_1 = B_{M,M}$, and $A_2 = B_{M,M-1}$.

In Katz's dissertation [8], she gives a variation of Neuts' Theorem to account for the boundary occupancy states.

**Theorem 2** *Provided*

$$\pi A_2 e > \pi A_0 e, \tag{3.23}$$

*where $\pi$ is the stationary probability vector of $A = A_0 + A_1 + A_2$, the queue is stable. The steady-state vector*

$$x = \{x_0, x_1, x_2, ...\} \tag{3.24}$$

*is given by*

$$x_k = x_{M-1} R^{k-(M-1)}, \qquad k \ge M, \tag{3.25}$$

*where $R$ is the unique solution in the set of non-negative matrices of spectral radius less than one of the equation*

$$R^2 A_2 + R A_1 + A_0 = 0 \tag{3.26}$$

*The matrix T, given by*

$$T = \begin{pmatrix} B_{00} & B_{01} & & & & \\ B_{10} & B_{11} & B_{12} & & & \\ & B_{21} & B_{22} & B_{23} & \cdots & \\ & & B_{32} & B_{33} & \cdots & \\ & & \cdot & \cdot & & \\ & & \cdot & \cdot & & \\ & & \cdot & \cdot & & \\ & & & B_{M-1,M-2} & B_{M-1,M-1} + RA_2 & \end{pmatrix} \tag{3.27}$$

*is an irreducible square matrix of size $M(M+1)$. The vector $\{x_0, x_1, ..., x_{M-1}\}$ is the left eigenvector corresponding to the zero eigenvalue of T. It is normalized so that,*

$$x_0 e + ... + x_{M-2} e + x_{M-1}(I - R)^{-1} e = 1 \tag{3.28}$$

The matrix $R$ is found iteratively from the following equation:

$$R_{NEW} = -A_0 A_1^{-1} - R_{OLD}^2 A_2 A_1^{-1} \tag{3.29}$$

where, initially, $R_{OLD} = 0$. Since $A_1 = B_{M,M}$, from equation (3.21), $A_1$ has full rank and is invertible. The number of iterations required to achieve a certain tolerance $\delta R = R_{NEW} - R_{OLD}$ increases with the size of $R$ and as the ratio of $\pi A_2 e$ and $\pi A_0 e$ approaches one from below. As this ratio reaches one, the system becomes marginally stable and the spectral radius of $R$ approaches one. Once $R$ is found, the matrix $T$ can be formed and $(x_0, x_1, ...x_{M-1})$ can be determined by finding the appropriate left eigenvector and normalizing it. Then, by Theorem 2, $x_k$, for $k \geq M$, form a matrix-geometric solution and can be generated from $x_{M-1}$ and $R$.

In doing queueing analysis of communications systems, it is often desired to obtain a probability distribution of the queue length of the system. For the system modeled by Katz, this is the external user occupancy. Once the joint probability distribution

$x = (x_0, x_1, ...)$ has been found, the marginal probability distribution of the external user occupancy can be found by summing over the phases of the internal system. Let $P(X = i)$ be the steady-state probability of having $i$ users in the external system.

$$P(X = i) = \sum_{j=0}^{M} x_{ij} = x_i e. \tag{3.30}$$

## 3.4 Phase-Type Distributions in QBD Processes

The QBD process depicted in Figure 3-5 allowed transitions in occupancy alone (vertical transitions) or in phase alone (horizontal transitions), but not in both. A more complex QBD process would allow simultaneous transitions in occupancy and phase. A matrix-geometric solution for the stationary probability vector of such a process can be found from Theorem 2. One way to create a QBD process with simultaneous changes in occupancy and phase is to include service times that follow a phase-type distribution. In this instance, a service completion results in a decrease in occupancy and a change in phase as the server begins servicing the next customer. The underlying vector process is a Markov chain because the residency times for all states are exponentially distributed. Therefore, the memoryless property holds, and the future of the process only depends on the current value of the state vector.

The model of the two-stage server queue is a simple example of a QBD process with a phase-type distribution. The state vector keeps track of both the occupancy and the stage of service of the current customer being served. A partial state-transition diagram for this queueing system that is different than the earlier one is shown in Figure 3-6. An example of a simultaneous transition occurs when the customer currently being served leaves the system while other customers are waiting in the queue for the server. Let the occupancy be $i$ ($i > 1$) at the instant before departure. The customer must be in stage two immediately before his exit. Upon that customer's departure from stage two (and, therefore, from the entire system), the next customer in line enters stage one of the server. Therefore, this transition causes the occupancy to decrease by one and the stage to change from phase two to phase one.

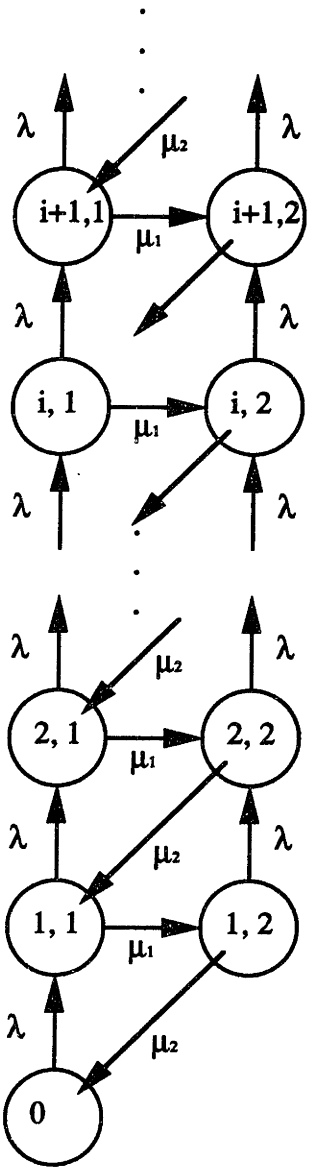



Figure 3-6: The Partial State-Transition Diagram for the Quasi-Birth-Death Process of the Two-Stage Cascaded Server.

# Chapter 4

# Design of the Buffered S-ARQ Model

Previous analyses of selective ARQ schemes have led to the conclusion that it is infeasible, and in many cases impossible, to develop an exact Markov-state model for any selective ARQ protocol because the number of possible states grows exponentially with the window size or round trip delay [2] [11] [12]. As a result, approximate models for some selective ARQ protocols have been created [2] [12]. In order to create models, certain simplifications and assumptions of the original protocol are necessary.

In this chapter, an approximate model for the buffered S-ARQ protocol is developed. First, the buffered S-ARQ protocol will be summarized and compared with the multi-access communications system studied by Katz. Next, the assumptions and simplifications that were necessary to create the model of the S-ARQ protocol are explained. Then, the design of the model is discussed. The model uses a hyperexponential distribution instead of the real transmission service time distribution. Finally, the derivation of the actual service time distribution for a random transmission and the method used to find a suitable hyperexponential distribution are discussed. In the next chapter, the model will be used to find a matrix-geometric solution which will yield a steady-state occupancy distribution.

## 4.1 Buffered S-ARQ and its Similarities with Katz's Multi-Access System

Recall that the buffered S-ARQ scheme is a block-selective protocol. The data is formed into messages of multiple fixed-length blocks. The message transmissions are sent continuously while bitmap acknowledgments are sent back to the source from the destination for each received transmission stating which of its blocks arrived at the destination in error and must be retransmitted. Upon receiving the bitmap acknowledgment, the source resends a partial message transmission consisting of only the blocks that require a retransmission. A message keeps requesting retransmissions until the entire message has been received correctly or the retry limit has been reached. The retry limit is the maximum number of transmissions a message can attempt before it is forced to leave the system.

The S-ARQ protocol resembles the multi-access communications system modeled by Katz discussed in the previous chapter. In Katz's communications system, users enter the system in order to send activities through a shared channel. Before being allowed to use the channel, a user must first be admitted into a window. Once admitted, the user can generate an activity of random length. After generating an activity, the user is considered active until its outstanding activity is completely sent over the channel. Because an active user is not allowed to generate a new activity, an admitted user can have at most one outstanding activity (either waiting for the channel or currently being sent over the channel) at any one time. After an activity is completed, the user that generated the activity enters an inactive period of random length. The inactive period ends with the generation of a new activity or the departure of the user from the system.

In the buffered S-ARQ queueing system, messages arrive at the source node to send transmissions over a channel shared by multiple messages. A message must first be admitted into the buffer (or window). After being admitted, the message requires an initial transmission of random length. Since a retransmission cannot be issued for a particular message until an acknowledgment for that message's previous

transmission has been received, each message in the window can have, at most, one outstanding transmission at any time. After a transmission has been sent, there is a round trip delay for that transmission during which the message that generated the transmission is inactive and waits for an acknowledgment. The channel meanwhile continues to send transmissions, if there are any, from other admitted messages. Upon receiving an acknowledgment, the inactive message either becomes active by requesting another transmission (if the acknowledgment is negative) or exits the system (if the acknowledgment is positive or the retry limit is reached).

The S-ARQ model, like Katz's model, considers the protocol to have an external system which keeps track of the overall message occupancy and an internal system that maintains the status of the window. In the QBD process to be developed, the occupancy variable will be called the external message occupancy referring to the occupancy of the external system, and the phase will be called the internal phase referring to the status of the internal system. The internal system of the S-ARQ model is more complicated than that of Katz's model. In Katz's model, generated activities are exponentially distributed in length, whereas, in the S-ARQ model, transmission lengths follow a hyperexponential distribution, as mentioned earlier. The internal phase of Katz's model is simply the number of active users in the window. The internal phase of the S-ARQ model is not only the number of active messages in the window (or, equivalently, the number of outstanding transmissions), but also the type of the transmission currently being sent as will be discussed later.

One shortcoming of Katz's model is that when a user is admitted into the window, it is immediately inactive. Therefore, since an inactive user decides either to leave the system or to generate a new activity, it is possible that a user may enter the window and leave the system without requesting a single activity. Clearly, in the S-ARQ protocol, this course of events is not allowed. An admitted message must immediately request a transmission and become active. Fortunately, incorporating such a modification does not add any difficulty when creating a matrix-geometric model. A message is admitted into the window either when it arrives and finds an open position or when an already admitted message exits the system leaving an

open position in the window. In the former case, the external message occupancy and active message occupancy (the number of active messages in the window) both increase by one because a newly admitted message generates a transmission. In the latter case, the external message occupancy decreases by one, but the active message occupancy increases by one because the departing message was inactive at the time. Note the difference between the external message occupancy and the active message occupancy. Therefore, both transitions involve a simultaneous change in the external message occupancy and the internal phase.

## 4.2 Assumptions and Simplifications of the S-ARQ Protocol

In this chapter, an approximate matrix-geometric model is created to analyze the buffered S-ARQ scheme. As with any model, this model requires simplifying assumptions about the protocol, some of which are discussed in this section.

Clearly, the lengths of retransmissions for a particular message become shorter and shorter since only the erred blocks of the last transmission are resent. Consequently, there is a strong correlation among the lengths of a message's initial transmission and its retransmissions. For a window size of one, this correlation is exhibited between the lengths of consecutive transmissions sent over the channel because the transmission requests are generated from the single admitted message until that message leaves the system and a new message enters the window. For a large window size in steady-state, however, the correlation between the lengths of consecutive transmissions sent over the channel is small since consecutive transmission requests are generated from different messages most of the time. This observation leads to an important assumption used in the development and analysis of the S-ARQ model. The lengths of transmissions over the channel (which are also the service times) are assumed to follow a hyperexponential distribution and to be independent and identically distributed. As was discussed in Chapter 2, the hyperexponential distribution is a phase-type distribution. This assumption permits the S-ARQ system to be modeled as a QBD

process while allowing some flexibility in the random transmission length distribution by not restricting it to follow an exponential distribution. Recall that a QBD process is memoryless. This assumption effectively ignores the memory required in the actual protocol to keep track of the previous transmission lengths of messages. Because of this assumption, the S-ARQ model will not be valid for light loads, but may be appropriate for moderate to heavy loads.

Another assumption that is made to create a QBD process for the S-ARQ model is that the round trip delay, which is the delay between the end of a transmission and the reception of its acknowledgment at the source, is exponentially distributed. For the same reason the assumption regarding hyperexponentially distributed transmission service times is necessary, the round trip delay assumption is required to maintain the memoryless property of the QBD process. Unfortunately, this assumption imposes severe restrictions on the model because often, in communications systems, the round trip delay is not exponentially distributed. The round trip delay in the real S-ARQ system represents the sum of the random access delay experienced at the medium access control (MAC) sublayer[1] and the acknowledgment turn-around time. The model presented here does not explicitly model the MAC sublayer, but appropriate values for the round trip delay obtained from existing simulations of the MAC sublayer will be used for the analysis.

The buffered S-ARQ protocol is based on a discrete-time system where one block of a message is sent per time slot. The model developed in this chapter is, however, a continuous-time model. The justification for creating a continuous-time model is that most times of importance in the real system, such as the transmission service times and round trip delays, contain many time slots and can appear as continuous variables. Although discrete-time matrix-geometric models exist, developing one for the S-ARQ protocol would be difficult because the required state vector would have an unmanageable state-space.

[1]Tanenbaum [17] gives a detailed discussion of the function of the MAC sublayer.

## 4.3 Description of the Buffered S-ARQ Model

Arrivals of messages which wish to be admitted to the window correspond to the arrivals to the external system. These messages arrive at a rate of $\lambda_1$. The interarrival times between consecutive messages are exponentially distributed with mean $\frac{1}{\lambda_1}$. If there are less than $M$ messages in the system, where $M$ is the window size, a new message is admitted upon arrival. If $M$ messages are already admitted, the arriving message waits outside the window in a queue. The queued messages are admitted into the window on a first-come-first-serve basis. When a message is admitted, it generates a transmission request immediately and is considered active. Only after its transmission has been sent, does the message become inactive. This ensures that every message generates at least one transmission before exiting the system. While inactive, a message leaves the system at a rate of $\mu_1$. Since there can be no more than $M$ inactive messages at any time, a finite population creates the message departure process.

The internal system models the generation and sending of transmissions from admitted messages. Each admitted message is allowed at most one outstanding transmission. When a message is inactive, it generates a transmission request at a rate of $\lambda_2$. Similar to message departures from the external system, the actual transmission arrival process depends on the number of inactive messages and is created by this finite population. The generated transmissions are serviced by a single channel on a first-come-first-serve basis. The transmission lengths (which are the service times) follow a hyperexponential distribution and are independent and identically distributed. Upon seizing the channel, a transmission can be one of two types. With probability $\alpha_1$, the transmission is of type one and requires a service time exponentially distributed with mean $s_1 = \frac{1}{\mu_{21}}$. With probability $\alpha_2 = 1 - \alpha_1$, the transmission is of type two and requires a service time exponentially distributed with mean $s_2 = \frac{1}{\mu_{22}}$. Since the size of the finite population for the internal system is a random variable that depends on, among other things, the external system, conventional analysis of finite population queueing systems cannot be used.

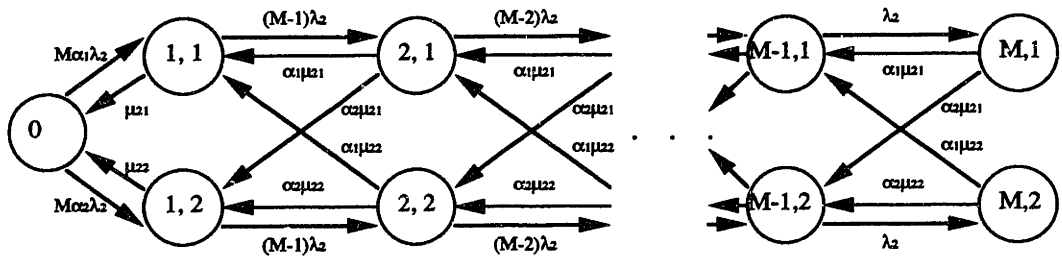



Figure 4-1: The Partial State-Transition Diagram for the Internal Phase Process of the S-ARQ Model with a Full Window ($i \geq M$).

The underlying vector stochastic process is a QBD process. The state vector can be partitioned into a two-dimensional vector, where the first variable is the external message occupancy $i$, and the second variable (or phase) is both the number of outstanding transmissions (or the active message occupancy) $j$ and the type $k$ of the current transmission being served. The state-space for this vector process is

$$E = \{(i,(j,k)) : 0 \leq i, 0 \leq j \leq M, k = 1 \text{ or } k = 2\} \tag{4.1}$$

Although the state-transition diagram of the entire S-ARQ model is too complicated to illustrate, a partial state-transition diagram of the internal system is given in Figure 4-1 where it is assumed that the window is full with $M$ admitted messages. Only the transitions where the external message occupancy stays the same (which occurs when no message enters or exits the external system) are shown in the figure.

Recall that in order to use the matrix-geometric method, the process of the internal system must have a finite number of phases. If $M$ is the window size of the system, then there can be $m$ active messages in the window, where $0 \leq m \leq M$. For $0 < m \leq M$, the transmission currently being sent can be of type one or of type two. When $m = 0$, however, no transmission is being sent so the type is not needed. Consequently, the number of possible internal phases is $2M + 1$. Actually, because the number of active messages cannot exceed the external message occupancy, when the external message occupancy is $i < M$, only $2i + 1$ possible internal phases exist.

### 4.3.1 Summary of Model Parameters

Before discussing the model for the S-ARQ protocol further, it is useful to find the roles of the various model parameters in the real system. First, $\lambda_1$ is the new message arrival rate to the external system which is the offered load to the system.

Immediately after the transmission of a message has been completed, the message becomes inactive and waits for an acknowledgment to come from the receiver. After a round trip delay, the message decides either to leave the system (if a positive acknowledgment or ACK arrives) or request another transmission (if a negative acknowledgment or NAK arrives). Unlike the real S-ARQ protocol which has bitmap acknowledgments that can acknowledge parts of a transmission, the acknowledgments in the model positively or negatively acknowledge the entire transmission due to the assumption that transmission lengths are independent and identically distributed. For an inactive message in the model, a negative acknowledgment comes at a rate of $\lambda_2$ because this is the rate at which a retransmission is generated. Likewise, a positive acknowledgment arrives at a rate of $\mu_1$ because this is the rate at which an inactive message leaves the system. The interarrival times of both kinds of acknowledgments are exponentially distributed.

In order to complete the model, a relationship between the round trip delay (the time taken for either an ACK or NAK to arrive), and the rates, $\lambda_2$ and $\mu_1$, is necessary. The round trip delay is a random variable that is the minimum of the ACK time (exponentially distributed with rate $\mu_1$) and the NAK time (exponentially distributed with rate $\lambda_2$). The minimum of two exponentially distributed random variables with rates $a$ and $b$ is an exponentially distributed random variable with a rate $a + b$. Therefore, the round trip delay is exponentially distributed with mean

$$T_{RTD} = \frac{1}{\lambda_2 + \mu_1} \tag{4.2}$$

Also,

$$P_{ACK} = \frac{\mu_1}{\lambda_2 + \mu_1} = \mu_1 T_{RTD} \tag{4.3}$$

is the probability that the arriving acknowledgment is an ACK and

$$P_{NAK} = 1 - P_{ACK} = \lambda_2 T_{RTD} \tag{4.4}$$

is the probability that the arriving acknowledgment is a NAK. In the analysis of the real transmission service time distribution, the expected number of required transmissions per message, $E[T]$, will be determined for a specific block error rate and initial message size distribution. The probability of a random transmission being successful is simply $\frac{1}{E[T]}$. To find appropriate values for $\lambda_2$ and $\mu_1$ to use in the model, specify the mean round trip delay $T_{RTD}$, calculate $E[T]$, set

$$P_{ACK} = \mu_1 T_{RTD} = \frac{1}{E[T]} \tag{4.5}$$

and

$$P_{NAK} = \lambda_2 T_{RTD} = 1 - \frac{1}{E[T]} \tag{4.6}$$

and solve for $\lambda_2$ and $\mu_1$.

### 4.3.2 Explanation of Permitted Transitions in the Model

This section explains the state transitions that are allowed in the QBD process of the S-ARQ model. The first $M + 1$ external message occupancy states of the QBD process are boundary states, and their transitions must be considered separately. Katz's model only had $M$ boundary occupancy states. The additional boundary state of the S-ARQ model, which is for the external message occupancy $M$, is a result of the modification needed to require newly admitted messages to become active immediately and request a transmission. Because the underlying Markov process of

the model is a QBD process, the transition matrix is block tridiagonal

$$\tilde{Q} = \begin{pmatrix} B_{00} & B_{01} & & & & & & & & \\ B_{10} & B_{11} & B_{12} & & & & & & & \\ & B_{21} & B_{22} & B_{23} & \cdots & & & & & \\ & & B_{32} & B_{33} & \cdots & & & & & \\ & & \cdot & \cdot & & & & & & \\ & & \cdot & \cdot & & & & & & \\ & & \cdot & \cdot & & & & & & \\ & & & & B_{M,M-1} & B_{M,M} & B_{M,M+1} & & & \\ & & & & & A_2 & A_1 & A_0 & & \\ & & & & & & A_2 & A_1 & A_0 & \cdots \\ & & & & & & & A_2 & A_1 & \cdots \\ & & & & & & & \cdot & \cdot & \\ & & & & & & & \cdot & \cdot & \\ & & & & & & & \cdot & \cdot & \end{pmatrix} \tag{4.7}$$

Although, all the above submatrices could be $(2M+1)$x$(2M+1)$, many of the rows and columns of the $B_{ij}$ submatrices are zeros. By deleting these rows and columns, the computation time required for analysis is reduced. The forms and sizes of each of the submatrices used in the final implementation are described below.

The rows and columns of the submatrices in $\tilde{Q}$ refer to different phases of the internal system. For example, the first row of the $(r,s)$th submatrix of $\tilde{Q}$ holds all the transition rates from the internal phase of zero outstanding transmissions as the external message occupancy changes from $r$ to $s$. Similarly, the first column of the $(r,s)$th submatrix of $\tilde{Q}$ holds all the transition rates to the internal phase of zero outstanding transmissions as the external message occupancy changes from $r$ to $s$. In general, row $2g+h-1$, for $1 \leq g \leq M$ and $h = 1$ or 2, of the $(r,s)$th submatrix of $\tilde{Q}$ holds the transition rates from the internal phase of $(g,h)$ where $g$ is the number of active messages in the window and $h$ is the type of transmission currently being sent over the channel as the external message occupancy changes from $r$ to $s$. Also,

column $2g+h-1$, for $1 \leq g \leq M$ and $h = 1$ or 2, of the $(r,s)$th submatrix of $\tilde{Q}$ holds the transition rates to the internal phase of $(g,h)$ as the external message occupancy changes from $r$ to $s$.

First, consider the transitions that can occur when the external message occupancy is $i > M$. The relevant submatrices are $A_0$, $A_1$, and $A_2$. The submatrix $A_0$ holds the rates for all transitions that increase the external message occupancy by one. The submatrix $A_1$ holds the transition rates that do not change the external message occupancy. The elements of $A_2$ are the rates of transitions that decrease the external message occupancy by one. All three submatrices are square of size $2M+1$. The message occupancy can only increase from an external message arrival which occurs at an average rate of $\lambda_1$. When $i > M$, some messages are already queued waiting for an open position in the window. Because a newly arriving message will go to the end of the queue in the external system, the internal phase will not be disturbed upon its arrival. Therefore, the submatrix $A_0$ is diagonal because a positive off-diagonal term would represent a transition in which the internal phase changes as well. That is,

$$A_0 = \begin{pmatrix} \lambda_1 & & & & & & \\ & \lambda_1 & & & & & \\ & & \cdot & & & & \\ & & & \cdot & & & \\ & & & & \cdot & & \\ & & & & & \lambda_1 & \\ & & & & & & \lambda_1 \end{pmatrix} \tag{4.8}$$

A decrease in the external message occupancy occurs when an inactive message in the window exits the system. If there are $m$ active messages, then the departure rate is $(M-m)\mu_1$ because the window is full when $i > M$. Unlike the transitions for an external message occupancy increase, when a message exits the system with $i > M$, the state of the internal system also changes. Upon a message departure, the message at the head of the queue is admitted into the window filling the vacant position and immediately becomes active by requesting an initial transmission. Therefore, a

message departure decreases the message occupancy by one and increases the number of active messages in the internal system by one. If an inactive message leaves the system when all $M$ admitted messages are inactive, the newly admitted active message will generate a transmission that will immediately begin service because it is the only outstanding transmission. This transmission is type one with probability $\alpha_1$ or type two with probability of $\alpha_2$. The submatrix associated with a decrease in the external message occupancy is

$$A_2 = \begin{pmatrix} 0 & M\alpha_1\mu_1 & M\alpha_2\mu_1 & & & & & \\ & & & \Upsilon_1 & & & & \\ & & & & \Upsilon_2 & & & \\ & & & & & \cdot & & \\ & & & & & & \cdot & \\ & & & & & & & \Upsilon_{M-1} \\ & & & & & & & 0 \\ & & & & & & & 0 \end{pmatrix} \tag{4.9}$$

where

$$\Upsilon_k = \begin{pmatrix} (M-k)\mu_1 & 0 \\ 0 & (M-k)\mu_1 \end{pmatrix} \quad for\ 1 \leq k \leq M-1 \tag{4.10}$$

When $i > M$, the transitions in which only the internal phase changes and the message occupancy stays the same are more complicated. Refer to Figure 4-1 for a

partial state-transition diagram. The submatrix that holds the transition rates is

$$A_1 = \begin{pmatrix} C_0 & D_0 & & & & & \\ F_1 & C_1 & D_1 & & & & \\ & F_2 & C_2 & D_2 & & & \\ & & F_3 & C_3 & D_3 & \cdots & \\ & & & F_4 & C_4 & \cdots & \\ & & & \cdot & \cdot & & \\ & & & \cdot & \cdot & & \\ & & & \cdot & \cdot & & \\ & & & & & F_{M-1} & C_{M-1} & D_{M-1} \\ & & & & & & F_M & C_M \end{pmatrix} \tag{4.11}$$

where

$$C_0 = -(\lambda_1 + M(\lambda_2 + \mu_1)) \tag{4.12}$$

and, for $1 \leq k \leq M$,

$$C_k = \begin{pmatrix} -(\lambda_1 + \mu_{21} + (M-k)(\lambda_2 + \mu_1)) & 0 \\ 0 & -(\lambda_1 + \mu_{22} + (M-k)(\lambda_2 + \mu_1)) \end{pmatrix} \tag{4.13}$$

Also,

$$D_0 = \begin{pmatrix} M\alpha_1\lambda_2 & M\alpha_2\lambda_2 \end{pmatrix} \tag{4.14}$$

$$D_k = \begin{pmatrix} (M-k)\lambda_2 & 0 \\ 0 & (M-k)\lambda_2 \end{pmatrix} \quad for\ 1 \leq k \leq M-1 \tag{4.15}$$

and

$$F_1 = \begin{pmatrix} \mu_{21} \\ \mu_{22} \end{pmatrix} \tag{4.16}$$

$$F_k = \begin{pmatrix} \alpha_1\mu_{21} & \alpha_2\mu_{21} \\ \alpha_1\mu_{22} & \alpha_2\mu_{22} \end{pmatrix} \quad for\ 2 \leq k \leq M \tag{4.17}$$

Recall that the rows in $A_1$ that correspond to the rates of transitions starting with

$k$ active messages in the window are row $2k$ (the transmission currently being served is type one) and row $2k+1$ (the transmission currently being served is type two). Likewise, the columns in $A_1$ corresponding to the rates of transitions ending with $k$ active messages are column $2k$ (the transmission that will be served is type one) and column $2k+1$ (the transmission that will be served is type two). The first row and column of $A_1$ correspond to the rates of transitions starting and ending with zero active messages, respectively.

The $D_k$ matrices hold the transition rates of transmission generations from inactive messages. Messages do not enter or exit the system during these transitions, so the external message occupancy does not change. Since the elements in the first row of $A_1$ are the rates for transitions starting with zero active messages in the window (i.e. all $M$ admitted messages are inactive), the rate of a transmission generation is $M\lambda_2$. But any transmission that is generated in this case will be served immediately and must choose between type one service or type two service. Therefore, the transition rate is $M\alpha_1\lambda_2$ for the generation of a type one transmission and is $M\alpha_2\lambda_2$ for a generation of a type two transmission. When $k$ active messages ($k > 0$) reside in the window, the rate for a transmission generation is $(M-k)\lambda_2$. The new transmission will wait for the channel and does not choose its type of service yet.

The elements of the $F_k$ matrices are the transition rates of transmission completions by the channel. A message does not leave the system after its transmission has been finished. Instead, the message becomes inactive, and the external message occupancy remains the same. If there is only one active message ($k = 1$), its transmission is currently being served, and, upon its completion, the number of active messages reduces to zero which is why the element appears in the first column of $A_1$. If the service is type one, the transition rate is $\mu_{21}$. If the service is type two, the transition rate is $\mu_{22}$. When $k$ active messages ($k > 1$) exist in the window, a transmission completion (which will occur at a rate of $\mu_{21}$ if it is of type one or at a rate of $\mu_{22}$ if it is of type two) reduces the number of active messages by one and also allows for the service of the next transmission in line to begin. The transmission at the head of the line is type one with a probability $\alpha_1$ or type two with a probability of $\alpha_2$.

Finally, the diagonal elements of $A_1$, which are held in the $C_k$ matrices where $k$ is the number of active messages, are also diagonal elements of $\tilde{Q}$. Therefore, these elements are the negative sums of the non-diagonal elements in their row of $\tilde{Q}$ which include some elements of $A_0$ and $A_2$.

For the boundary external message occupancy states, $i < M$, the transition submatrices are different. First, since there can be, at most, $i$ active messages, there are only $2i + 1$ internal phases, and, consequently, the size of these submatrices can be smaller than $(2M + 1)$x$(2M + 1)$. The submatrix $B_{ij}$, which is reduced to a $(2i + 1)$x$(2j + 1)$ matrix after removing the extraneous rows and columns of zeros, holds the transition rates from an external message occupancy $i$ to an external message occupancy $j$. For $i < M$, a newly arriving message sees an open position in the window and is immediately admitted and active. Therefore, unlike the case of $i > M$, the active message occupancy of the internal system increases by one, when the message occupancy increases by one. On the other hand, when an inactive message departs, with $i < M$, no message is waiting to enter the window, so the internal phase (the number of active messages and the type of service) remains the same.

The submatrix $B_{01}$ describes the possible transitions from an external message occupancy of zero to an external message occupancy of one. When the external message occupancy is zero, the channel is idle. Consequently, not only is an arriving message admitted into the window right away, but its transmission is also immediately handled by the channel. Again, the transmission can be of type one or type two. $B_{00}$, a scalar that is a diagonal element of the transition matrix $\tilde{Q}$, is the negative of the sum of the remaining elements in that row of $\tilde{Q}$.

$$B_{00} = -\lambda_1 \tag{4.18}$$

$$B_{01} = \begin{pmatrix} 0 & \alpha_1\lambda_1 & \alpha_2\lambda_1 \end{pmatrix} \tag{4.19}$$

A transition that changes the external message occupancy from one to zero occurs only when the number of active messages (or outstanding transmissions) is zero because, if the sole message were active, it could not exit the system. When the message is

inactive, it exits the system at a rate of $\mu_1$.

$$B_{10} = \begin{pmatrix} \mu_1 \\ 0 \\ 0 \end{pmatrix} \tag{4.20}$$

Transitions that do not change the external message occupancy can occur in a few ways. When the external message occupancy is one, an inactive message can generate a transmission that will be handled immediately by the channel since there are no transmissions before it. This new transmission may be type one or type two. These transitions correspond to elements $B_{11}(1,2)$ and $B_{11}(1,3)$. A transition is also possible when the transmission of the admitted message is completed, so that the message becomes inactive. If the transmission is type one, the rate of such a transition is $\mu_{21}$. If the transmission is type two, the rate is $\mu_{22}$. These transitions are elements $B_{11}(2,1)$ and $B_{11}(3,1)$. The diagonal terms of $B_{11}$ are the negative sum of the remaining elements in that row of $\tilde{Q}$.

$$B_{11} = \begin{pmatrix} -(\lambda_1 + \mu_1 + \lambda_2) & \alpha_1\lambda_2 & \alpha_2\lambda_2 \\ \mu_{21} & -(\lambda_1 + \mu_{21}) & 0 \\ \mu_{22} & 0 & -(\lambda_1 + \mu_{22}) \end{pmatrix} \tag{4.21}$$

Transitions that increase the external message occupancy from one to two occur only when a new message arrives. If $M > 1$, the new message becomes active and changes the internal phase. If there are no outstanding transmissions before the arrival, the new transmission is served immediately and chooses type one or two (elements $B_{12}(1,2)$ and $B_{12}(1,3)$). If, however, there is an outstanding transmission, the new transmission waits in line for service (elements $B_{12}(2,4)$ and $B_{12}(3,5)$). In all of these transitions, the number of active messages is increased by one.

$$B_{12} = \begin{pmatrix} 0 & \alpha_1\lambda_1 & \alpha_2\lambda_1 & & \\ & & & \lambda_1 & \\ & & & & \lambda_1 \end{pmatrix} \tag{4.22}$$

The remaining $B_{ij}$ submatrices for $i < M$ can be found in a similar manner by carefully considering all possible transitions. For brevity, these matrices are not shown here, but the Mathematica code in Appendix A shows how to find them recursively.

The external message occupancy state $i = M$ is also a boundary state. In this case, when an inactive message exits the system, no message is waiting to be admitted, so the internal phase stays the same as it does when $i < M$. Unlike the case where $i < M$, when a new message arrives with $i = M$, the window is full, and the new message begins a queue for admittance. The internal phase remains the same here as well. Consequently, both matrices $B_{M,M-1}$ (which is of size $(2M+1)$x$(2M-1)$) and $B_{M,M+1}$ (which is a square matrix of size $2M+1$) have nonzero elements only on their diagonals. In fact, $B_{M,M+1} = A_0$, and $B_{M,M-1} =$

$$\begin{pmatrix}
M\mu_1 & & & & & & & & \\
 & (M-1)\mu_1 & & & & & & & \\
 & & (M-1)\mu_1 & & & & & & \\
 & & & (M-2)\mu_1 & & & & & \\
 & & & & (M-2)\mu_1 & & & & \\
 & & & & & \ddots & & & \\
 & & & & & & \ddots & & \\
 & & & & & & & \ddots & \\
 & & & & & & & & \mu_1 \\
 & & & & & & & & & \mu_1 \\
 & & & & & & & & & 0 \\
 & & & & & & & & & 0
\end{pmatrix} \tag{4.23}$$

For transitions where the external message occupancy does not change, the system behaves like the case when $i > M$ and $B_{M,M} = A_1$.

## 4.4 Derivation of the Service Time Distribution for a Random Transmission

Before analyzing the S-ARQ model with the matrix-geometric method, a hyperexponential distribution that approximates the real service distribution must be found. First, the real service time distribution for a random transmission as seen as by the channel must be determined.

Assume the initial distribution of the number of blocks in a message is a geometric distribution with mean $f$, and the block error rate is a constant $p$ with block errors occurring independently. The length of a transmission (or the number of blocks in the transmission) is equal to the service time the transmission requires. The analysis can be done easily in the Z-transform domain.

Let $N_1^0(z)$ be the Z-transform of the geometric distribution of the length of an initial message transmission[2].

$$N_1^0(z) = \frac{\frac{z}{f}}{1 - \frac{f-1}{f}z} = \frac{z}{f - (f-1)z} \tag{4.24}$$

Consider the random variable $g$ that describes the number of blocks that must be retransmitted after a single block has been transmitted once. Clearly, $g$ can only take on a value of one (if the block was in error) or zero (if the block was received correctly). Let $G(z)$ be the Z-transform for the distribution of $g$, so

$$G(z) = 1 - p + pz \tag{4.25}$$

Let $N_2^0(z)$ be the Z-transform of the distribution for the number of blocks that need to be resent in the second transmission (or first retransmission) of a message. For each block sent in the initial message, the random variable $g$ describes whether or not that block has to be retransmitted. Since there is a random number of blocks in the initial transmission, the total number of blocks that must be retransmitted in the

[2]Drake [7] gives a good treatment of Z-transforms.

second transmission is a random sum of random variables $g$. From a known theorem for the Z-transform of a random sum of random variables [7],

$$N_2^0(z) = N_1^0(G(z)) \tag{4.26}$$

The probability of not needing the second transmission is $N_2^0(0)$, which is the coefficient of the $z^0$ term in the Z-transform. By definition, this coefficient is the probability that zero blocks need to be retransmitted.

The argument for finding the Z-transform of the distribution of the length of the second transmission can be extended to finding the Z-transform of the distribution of the length of the $i$th transmission, $N_i^0(z)$, for $i > 2$. In general,

$$N_i^0(z) = N_{i-1}^0(G(z)) = N_1^0(p^{i-1}z + 1 - p^{i-1}) \tag{4.27}$$

$N_i^0(0)$ is the probability that the $i$th transmission is not needed, and $1 - N_i^0(0)$ is the probability that the $i$th transmission is needed. In fact, it is easy to see that $N_i^0(0) - N_{i-1}^0(0)$ is the probability that the $(i-1)$th transmission is the last transmission required for the entire message to be sent correctly.

Let $N_i(z)$ be the Z-transform of the distribution of the length of the $i$th transmission of a message given that the length is not zero. $N_i(z)$ can be found explicitly from $N_i^0(z)$ using a generalization of Bayes' Theorem

$$N_i(z) = \frac{N_i^0(z) - N_i^0(0)}{1 - N_i^0(0)} \tag{4.28}$$

Also, let $P(RT = i)$ be the probability that a random transmission is the $i$th transmission of a message. $P(RT = i)$ is the probability of needing the $i$th transmission, $1 - N_i^0(0)$, divided by the average number of transmissions required for complete successful reception of a message.

$$P(RT = i) = \frac{1 - N_i^0(0)}{E[T]} \tag{4.29}$$

Two formulas for $E[T]$ are given below.

$$E[T] = \sum_{i=1}^{\infty} i(N_{i+1}^0(0) - N_i^0(0)) = \sum_{i=1}^{\infty}(1 - N_i^0(0)) \tag{4.30}$$

The Z-transform of the distribution of the length of a random transmission is

$$N(z) = \sum_{i=1}^{\infty} P(RT = i)N_i(z) \tag{4.31}$$

$N(z)$ is the weighted sum of the Z-transforms of the distributions of the length of first, second, third, etc. transmissions given that the lengths of these transmissions are not zero. The weight of the $i$th term is the probability of a random transmission being the $i$th transmission of a message. After the appropriate substitutions, equation (4.31) simplifies to

$$N(z) = \frac{1}{E[T]} \sum_{i=1}^{\infty}(N_i^0(z) - N_i^0(0)) \tag{4.32}$$

The above formulas for $E[T]$ and $N(z)$ are not in closed form but can be solved by a computer by assuming an upper limit on the number of transmissions allowed per message. To impose a retry limit of $R$ (which includes the initial transmission of a message), after which the message is dropped, let $N_{R+1}^0(z) = 1$. Equations (4.30) and (4.32) reduce to

$$E[T] = R - \sum_{i=1}^{R} N_i^0(0) \tag{4.33}$$

$$N(z) = \frac{1}{E[T]} \sum_{i=1}^{R}(N_i^0(z) - N_i^0(0)) \tag{4.34}$$

After finding $N(z)$, the first three noncentral moments can be calculated easily.

$$E[N] = \frac{dN(z)}{dz} \mid_{z=1} \tag{4.35}$$

$$E[N^2] = \frac{d^2N(z)}{dz^2} \mid_{z=1} + E[N] \tag{4.36}$$

$$E[N^3] = \frac{d^3N(z)}{dz^3} \mid_{z=1} + 3E[N^2] - 2E[N] \tag{4.37}$$

A hyperexponential distribution that approximates the real service time distribution can now be found by matching these first three moments using an iterative method to solve a system of three nonlinear equations and three unknowns. This method along with the Mathematica code that performs the iterations are explained in Appendix B. The iterations converge to yield four parameters $\alpha_1$, $\alpha_2$, $s_1$, and $s_2$, where $\alpha_1$ is the probability the service time for a transmission is exponentially distributed with mean $s_1$ and $\alpha_2 = 1 - \alpha_1$ is the probability the service time for a transmission is exponentially distributed with mean $s_2$. This server is shown in Figure 2-4, where $\mu_1 = \frac{1}{s_1}$ and $\mu_2 = \frac{1}{s_2}$.

# Chapter 5

# Analysis of the S-ARQ Model

Finding the matrix-geometric solution to a QBD process is usually computationally intensive. To obtain statistics on the external message occupancy, the S-ARQ model developed in the previous chapter was implemented in Mathematica. The Mathematica code with documentation is included in Appendix A. This chapter begins by describing a simulation of the buffered S-ARQ protocol. The analytical external message occupancy (referred to hereafter as the message occupancy) results are compared with those acquired from the simulation. Statistics on the message occupancy as a function of the window size are also presented. The S-ARQ model can also be used to study the effects of other parameters such as the message arrival rate, the block error rate, and the round trip delay on the message occupancy, but these results will not be presented here. Finally, the S-ARQ model is used to perform stability analysis of the queueing system by finding a relationship between the maximum message arrival rate (or offered load) and the window size.

## 5.1 Occupancy Analysis and Simulation of the Buffered S-ARQ Protocol

A simulation of the S-ARQ protocol was created with the General Purpose Simulation Software (GPSS) package. The simulation code is included in Appendix C. GPSS is

a flexible package that simulates basic queueing systems quickly and efficiently. The GPSS simulation of the S-ARQ protocol includes some of the assumptions made in the development of the S-ARQ model such as restricting the round trip delay to follow an exponential distribution. This assumption was kept in the simulation model so that the validity of the independent-transmission-length assumption could be tested when comparing the analytical and simulation results. The simulation, however, uses the real service time distribution for transmission lengths rather than the hyperexponential assumption used in the model. In other words, the simulation has a memory of the size of and the number of transmissions each message in the system has generated. In the simulation, a retry limit of eight is used but is not significant for the results shown here because the number of messages that reached this limit is negligible.

In the results presented in this chapter, the analytical model uses a hyperexponential service distribution that approximated the length of a random transmission given that a message's initial transmission is geometrically distributed with a mean of 20 blocks and block error rate of 10 percent. The simulation derives this geometric distribution by discretizing an exponentially distributed random variable of mean 19.496 instead of 20. The continuous random variable is converted to a discrete random variable by adding one to the integer part of the continuous random variable. For example, if a message has an initial size of 5.14 blocks, in order to have an integer number of blocks, the simulation uses an initial size of 6 blocks for that transmission. It can be shown that discretizing an exponentially distributed random variable with mean 19.496 in this manner yields a geometric distribution of mean 20.

### 5.1.1 Message Occupancy Analysis

The underlying Markov chain of the S-ARQ model is a quasi-birth-death process with boundary occupancy states and, therefore, a matrix-geometric solution can be found by applying Theorem 2. The $T$ matrix of the S-ARQ model differs slightly from that

of Katz's model.

$$T = \begin{pmatrix} B_{00} & B_{01} & & & & \\ B_{10} & B_{11} & B_{12} & & & \\ & B_{21} & B_{22} & B_{23} & \cdots & \\ & & B_{32} & B_{33} & \cdots & \\ & & \cdot & \cdot & & \\ & & \cdot & \cdot & & \\ & & \cdot & \cdot & & \\ & & & & B_{M,M-1} & B_{M,M} + RA_2 \end{pmatrix} \tag{5.1}$$

The matrix $R$ is found iteratively from the equation

$$R_{NEW} = -A_0 A_1^{-1} - R_{OLD}^2 A_2 A_1^{-1} \tag{5.2}$$

The left eigenvector, $x_B$, associated with the zero eigenvalue of $T$ is determined and normalized, where

$$x_B = (x_0, x_1, ..., x_{M-1}, x_M) \tag{5.3}$$

As a result of the deletion of the extraneous rows and columns of zeros from the submatrices $B_{ij}$, $T$ is a square matrix of size $(M+1)^2$. Consequently, $x_B$ is a row vector of length $(M+1)^2$, and $x_i$, for $0 \leq i \leq M$, are row vectors of size $2i+1$. The normalization equation of Theorem 2 which assumes all $x_i$ sre the same size requires a slight modification. After normalization, the remaining part of the steady-state distribution is found from

$$x_k = x_M R^{k-M}, \qquad for\ k \geq M \tag{5.4}$$

The matrix-geometric solution yields the joint probability density for the exterrnal message occupancy and the internal phase (the active message occupancy and type of transmission being sent). The row vector $x_i$ holds the stationary probabilities for all internal phases for an external message occupancy $i$. The external message occupancy

density is just the marginal density

$$P(X = i) = \sum_{j=1}^{2i+1} x_i(j) \tag{5.5}$$

where $X$ is the message occupancy and $x_i(j)$ is the $j$th term of the row vector $x_i$.

The values for the parameters used in most of the analysis are presented here. As mentioned earlier, the initial message length is geometrically distributed with a mean of 20 blocks. Block errors are independent and occur at a rate of 10 percent. The message arrival rate is 0.02 messages/slot. The round trip delay is assumed to be exponentially distributed with a mean duration of 40 slots.

The message occupancy density functions for both the analysis and the simulation are shown in Figure 5-1 for a window size of eight. The mean message occupancy from the analysis was 2.4347 which underestimated the simulation mean of 2.6430 by 7.88 percent. One explanation for the underestimate is that the S-ARQ model does not keep track of how many transmissions a particular message has had and, therefore applies the same probability of success, $P_{ACK}$, to each transmission. As a result, many messages exit the system after the initial transmission, spend less time in the system, and reduce the message occupancy. Explained in a different way, the model has a transmission success probability that is independent of the length of the transmission so long transmissions may be successful in the model more frequently than they are in the simulation. Clearly, in the real protocol and the simulation, there is a correlation between the transmission length and the probability of its success.

The window is the distinctive feature of this S-ARQ protocol. It is of interest to observe the effects the window size has on the message occupancy. Figure 5-2 shows the mean message occupancy as it varies with the window size. The 99th percentile of the occupancy distribution versus the window size is graphed in Figure 5-3. From this graph, a rough estimate (even though it would be an underestimate) can be made for the amount of storage space is required at the source to hold messages in the window and the queue. In Figure 5-4, the probability of the queue length (message occupancy) $X$ being less than or equal to the window size $M$ is shown versus the

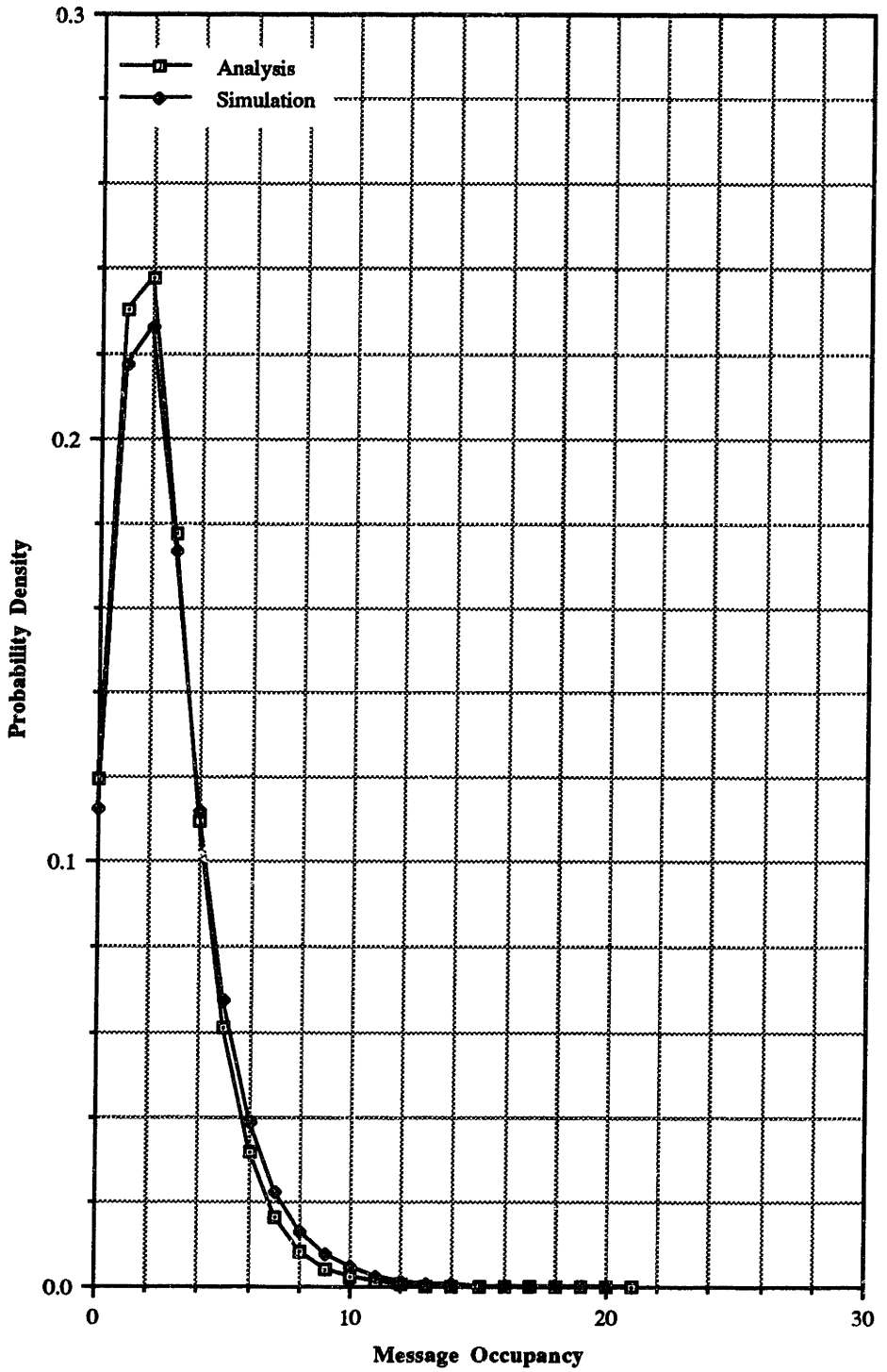



Figure 5-1: Message Occupancy Density Functions from Analysis (the S-ARQ Model) and Simulation: Avg. Initial Message = 20 Blocks, Block Error Rate = 0.1, Offered Load = 0.02 Messages/Slot, $T_{RTD}$ = 40 Slots, $M$ = 8.

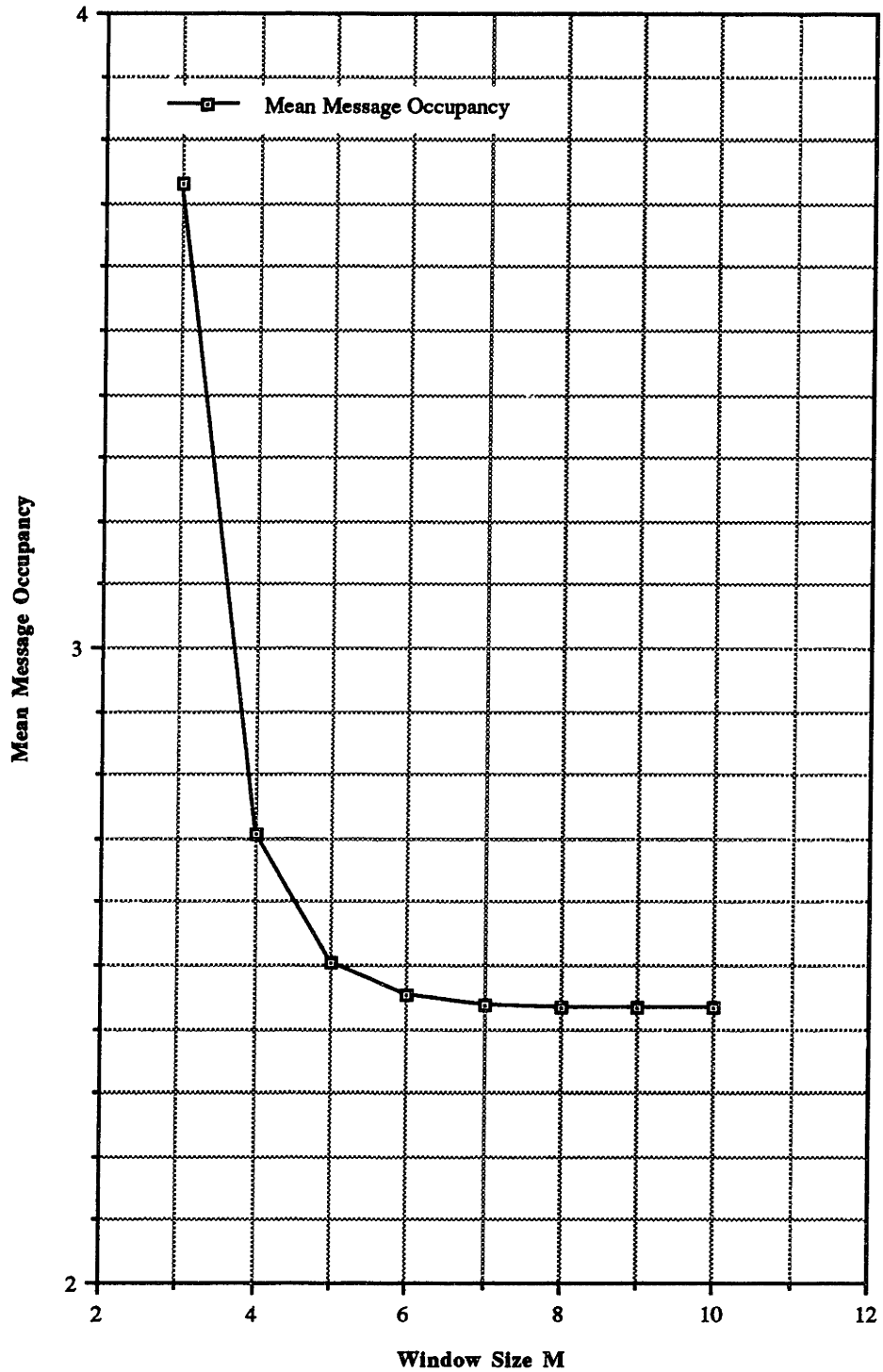



Figure 5-2: Mean Message Occupancy versus Window Size: Avg. Initial Message = 20 Blocks, Block Error Rate = 0.1, Offered Load = 0.02 Messages/Slot, $T_{RTD}$ = 40 Slots.

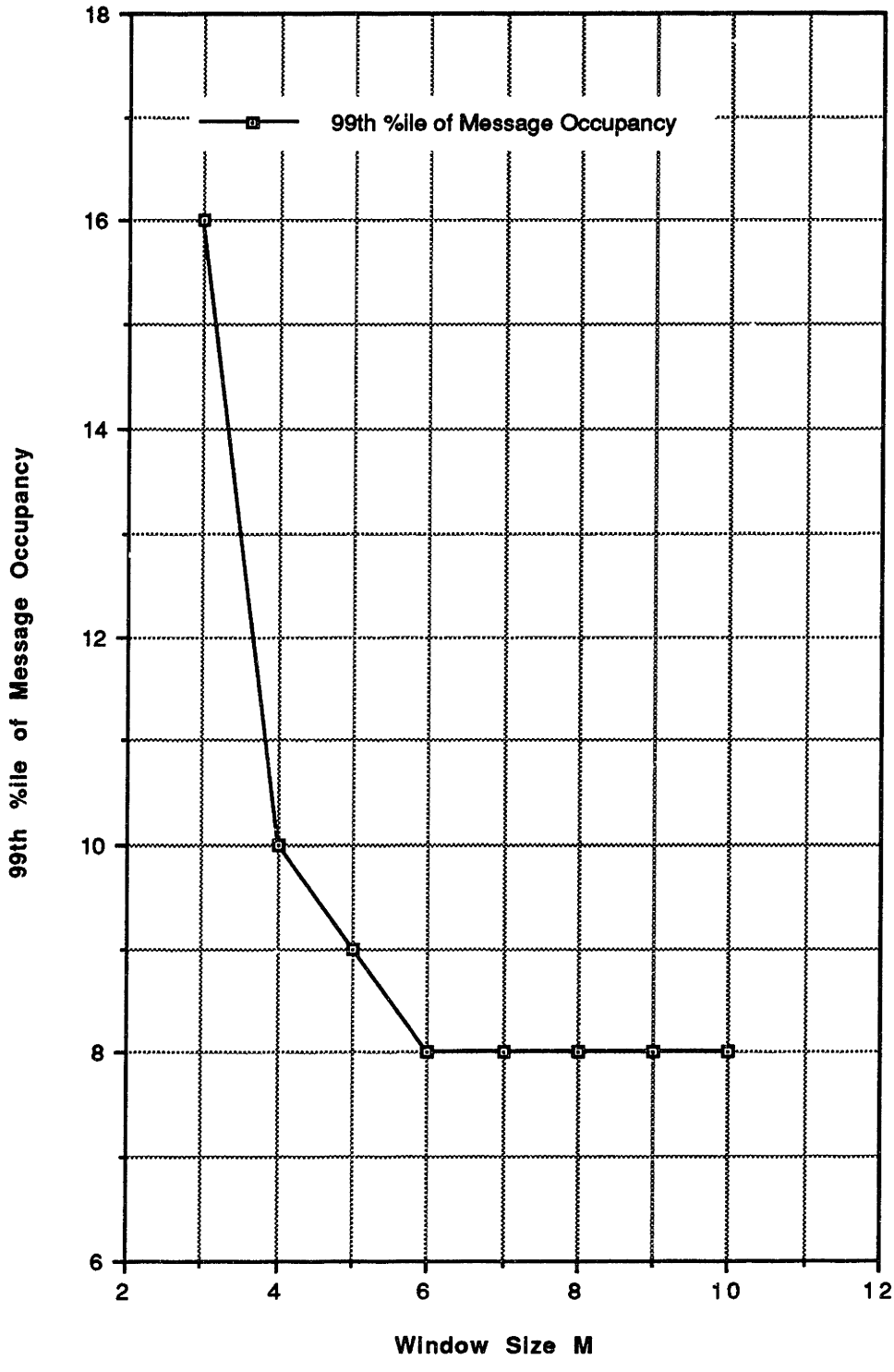



Figure 5-3: 99th Percentile of Message Occupancy Distribution versus Window Size: Avg. Initial Message = 20 Blocks, Block Error Rate = 0.1, Offered Load = 0.02 Messages/Slot, $T_{RTD}$ = 40 Slots.

window size. This graph is useful in determining the probability of having a window overflow. The results from the model shown in Figures 5-2, 5-3, 5-4 are as expected. At small window sizes, there is a great deal of congestion because the channel is shared by only a few messages at a time and is often idle even though messages may be waiting in the queue to use the channel. At first, an increase in the window size alleviates the traffic congestion in the system significantly. At large window sizes, however, further increases in the window size has diminishing returns.

## 5.2 Stability Analysis

Stability is an important issue in the study of queueing systems. If customers (or messages) arrive at a faster rate than they can be serviced on the average, the system becomes unstable as the occupancy of the system grows to infinity. Neuts' Theorem provides a condition for stability for a quasi-birth-death process. A queueing system described by a QBD process with matrices $A_2$ and $A_0$ is stable if

$$\pi A_2 e > \pi A_0 e \tag{5.6}$$

where

$$\pi A = 0, \qquad A = A_0 + A_1 + A_2, \tag{5.7}$$

and

$$\sum_{i=0}^{2M+1} \pi_i = 1 \tag{5.8}$$

The vector $\pi$ is the normalized left eigenvector for the zero eigenvalue of $A$ and is the steady-state probability vector of the phase process (with the state-transition diagram shown in Figure 4-1 when the message occupancy is greater than $M$). The left-hand side of the inequality is simply the average message service rate of the system, and the right-hand side is the average message arrival rate of the system. Therefore, the system remains stable if the message arrival rate $\lambda_1$ is less than $\pi A_2 e$ (which is independent of $\lambda_1$). The maximum allowable offered load is $\pi A_2 e$ and is graphed as a function of window size in Figure 5-5. As the window size increases, the maximum

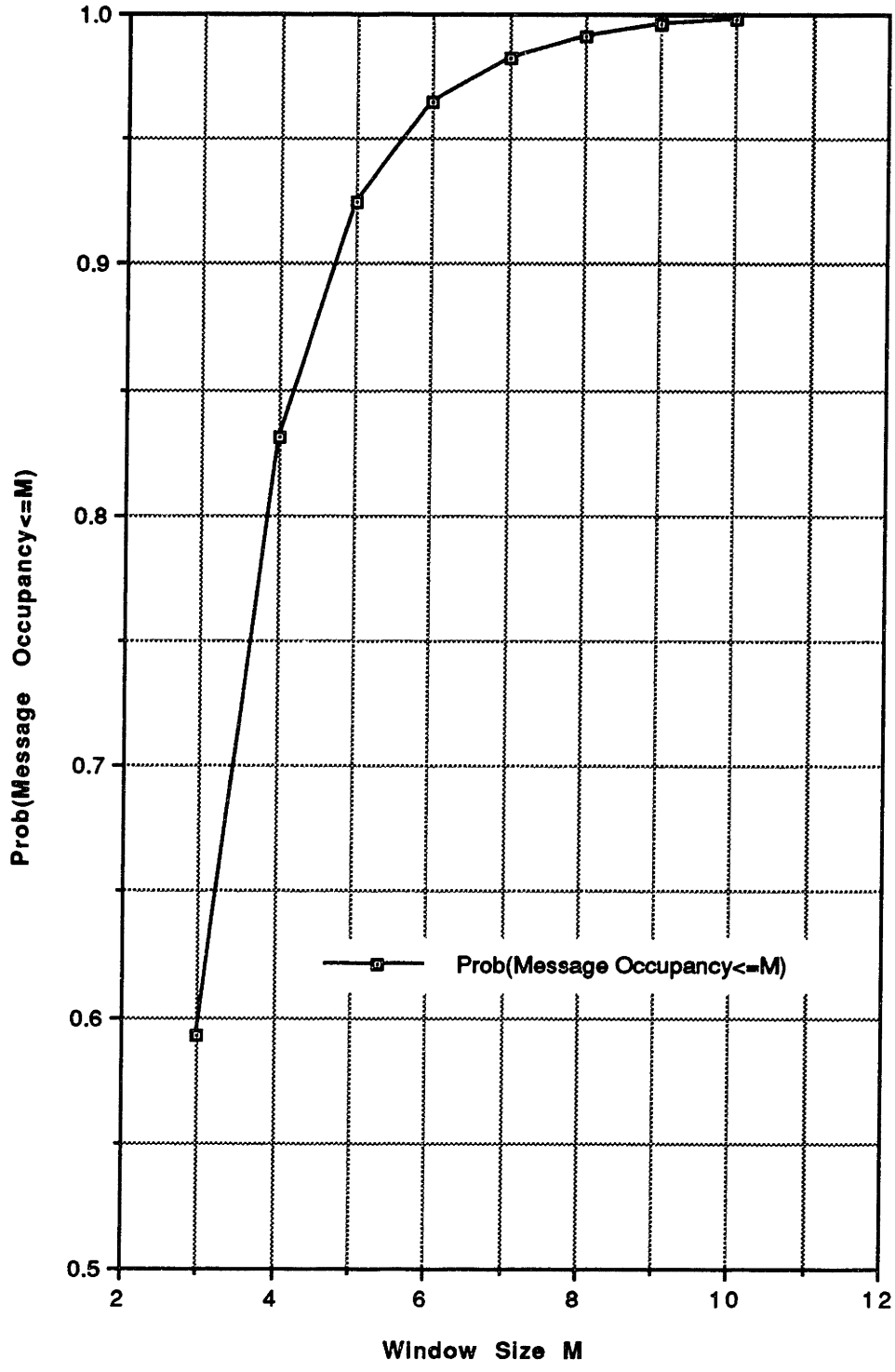



Figure 5-4: Probability that Message Occupancy $\leq$ Window Size versus Window Size: Avg. Initial Message = 20 Blocks, Block Error Rate = 0.1, Offered Load = 0.02 Messages/Slot, $T_{RTD}$ = 40 Slots.

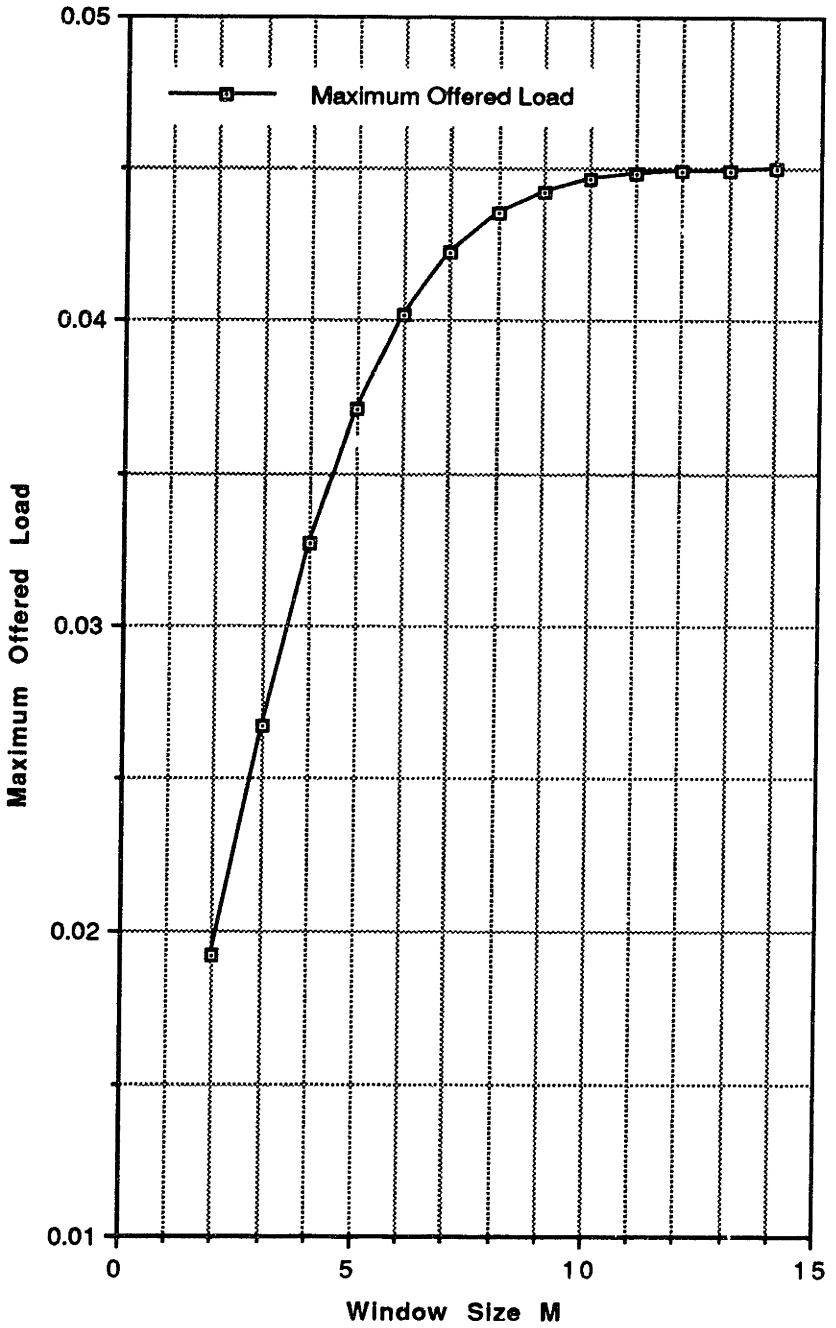



Figure 5-5: Maximum Stable Offered Load versus Window Size: Avg. Initial Message = 20 Blocks, Block Error Rate = 0.1, $T_{RTD}$ = 40 Slots.

offered load reaches a theoretical threshold value of $\frac{1-p}{f}$, where $p$ is the block error rate and $f$ is the mean initial message size. For $p = 0.1$ and $f = 20$, the threshold value is 0.045 which is confirmed by Figure 5-5. To check the results of Figure 5-5, for a given window size, the message arrival rate in the model was increased so that it exceeded the maximum for that window size, and, as predicted from the analysis, the probability of the message occupancy staying finite dropped to near zero indicating the system had become unstable.

One possible application of the stability analysis of the S-ARQ protocol is that the relationship found between the maximum stable offered load and the window size may be useful in allocating bandwidth to multiple users. If the MAC sublayer has a global buffer that holds reservation requests from two users, $C$ and $D$ and it is desired to allow user $C$ to give more requests than user $D$ (and consequently, to give $C$ more bandwidth than $D$), then perhaps this can be achieved by setting the window size of user $C$ to a value greater than that of user $D$ so that user $C$'s throughput at the Data Link layer is more than user $D$'s throughput. Future study may concentrate on the possibilities of this dynamic bandwidth allocation application.

# Chapter 6

# Conclusion

In comparing the analytical results with the simulation results, the validity of the independent-transmission-length assumption is checked. In other words, the results from a memoryless QBD model are being compared to those of a simulation that has a memory of the status of every message in the system. For the parameters chosen in the previous chapter, the analytical results underestimate, but are close to, the simulation results. Unfortunately, the analytical results are reasonable only for low error rates and large window sizes. When the error rate is too high, the model uses a value for $P_{ACK} = \frac{\mu_1}{\lambda_2+\mu_1}$ that is an unrealistic approximation. The model cannot emulate the real system because the model permits too many initial transmissions to be successful. Consequently, the messages associated with these transmissions leave the system too early and the analytical message occupancy further underestimates the simulation results. When the window size is small and under light load assumptions, the independent-transmission-length assumption also loses its validity because there is a high correlation between the lengths of successive transmissions.

The phase of the S-ARQ QBD process contained two variables, the number of active messages in the window and the type of transmission being served. At first, it may seem attractive to develop a more accurate model by creating a more complex phase. This could be done by adding a third variable to the phase or increasing the number of transmission types. Such a model, however, quickly becomes unmanageable as the phase becomes more detailed. Deciding what variable to add to the phase and

how to model that variable accurately are additional issues that have to be considered.

Some of the other assumptions required in developing the S-ARQ model may be unrealistic. For example, rarely is it a reasonable approximation to require the round trip delay of a communications system to be exponentially distributed. This assumption was necessary in the model to maintain the memoryless property of the QBD process. In many systems, the round trip delay actually is more or less deterministic and not very stochastic. Neuts [13] [14] has done a great deal of work in the development of stochastic matrix models for complex queueing systems, including models that are not restricted to QBD processes and can, therefore, deal with non-exponential distributions. These models are more general but are also more computationally intensive.

In conclusion, although a reasonably accurate model for the buffered S-ARQ protocol was developed, the range of the parameters for which the model remains valid is restricted. It may be possible to create a more accurate model, but one must seriously consider if the additional insight gained outweighs the computational costs and complexity of this new model when a simulation can often mimic the real protocol more easily and accurately.

# Appendix A

# Implementation of the S-ARQ Model in Mathematica

The Mathematica code in this appendix performs the matrix-geometric method on the S-ARQ model. The first program creates the matrix $T$ and finds the matrix $R$ that is required to obtain a matrix-geometric solution. The second section of code explains how to find the steady-state probabilities for the boundary message occupancy levels. The third section is a program that finds the entire message occupancy distribution as well as other statistics. The code uses $W$, instead of $M$, to denote the window size and $x_1$ and $x_2$, instead of $s_1$ and $s_2$, to denote the mean service times of the exponential distributions in a hyperexponential distribution.

```
(* This program calculates the matrix R for the matrix-geometric
solution:
x(k) = x(W)*(R^(k-W)) for k > W where W is the Window Size.
This program does not calculate x(k); that calculation is done in the
next section of code. *)


(* Need to include this file to use the BlockMatrix command *)
<<LinearAlgebra`MatrixManipulation`
```

```
(* W is the window size of the system. *)
W = 8;

(* a1, a2, x1, x2 are parameters that yield an approximate
distribution for the actual distribution of the length of a random
transmission.  The actual distribution depends upon the block error
rate and the initial message length distribution and is found
numerically.  The values of these parameters were determined by
iteration scheme for solving simultaneous nonlinear equations; see
Appendix B. *)

(* The approximate distribution is a hyperexponential distribution
which is a weighted sum of two exponential distributions.
   a1 is the probability of choosing the server of mean x1,
   a2 is the probability of choosing the server of mean x2 *)

a1
a2
x1
x2

(* u21 is the service rate of server one
   u22 is the service rate of server two *)

u21 = N[1/x1, 20];
u22 = N[1/x2, 20];

(* u1 is the message departure rate, exponentially distributed (i.e.
ACK rate)
l2 is the message retransmission request rate, exp. dist.(i.e. NAK rate)
```

```
l1 is the external message arrival rate, exp. dist. *)

(* u1 and l2 are related to the round trip delay which is also assumed
to be exponentially distributed in this model.  Let RTD be the mean
interarrival time of the acknowledgments (either ACKs or NAKs).  Then,
RTD = 1/(u1 + l2) because the random variable that is the minimum of
two exponential random variables is also exponential with the rate
being the sum of the rates of the initial two random variables.  Also,
u1/(u1+l2) is the probability of an acknowledgment being an ACK, and
l2/(u1+l2) is the probability of an acknowledgment being a NAK.
Et is the expected number of transmissions per message and is found
from the code in Appendix B. *)

RTD = 120;
u1 = 1/Et/RTD;
l2 = (1- 1/Et)/RTD;
l1 = 0.005;

(* e is a column vector of size 2W+1 with all elements being one *)

e = Table[1,{2 W+1},{1}];

(* z is a function that creates a zero matrix of size i,j *)

z[i_,j_] := Table[0,{i},{j}];

(* The matrix T is a square matrix of size (W+1)^2 and has the
following form:

T = |B11    B12                                                  |
```

```
|B21    B22     B23                                          |
|       B32     B33     B34                                  |
|               B43     B44     B45                          |
|                       .       .       .                    |
|                               .       .       .            |
|                                       .       .       .    |
|                                               BW(W-1)  BWW+R*A2 |
```

Bij is a matrix of size 2i-1 by 2j-1 that holds the transition rates for all transitions from occupancy state i-1 to occupancy level j-1.

Note that transitions are only allowed between neighboring occupancy levels since the matrix T is block tridiagonal.
For all |i-j|>1, Bij is a zero matrix.
In the last block row of T shown above, the matrix R and A2 are found first, then multiplied and added to BWW. *)

```
(* B11 = {{ -l1 }}
   B12 = {{ 0, a1*l1, a2*l1 }}
   f = B21
   g = B22
   h = B23 *)

f = {{u1},{0},{0}};
g = {{-(l1+u1+l2),a1 l2,a2 l2},{u21,-(l1+u21),0},{u22,0,-(l1+u22)}};
h = {{0,a1 l1,a2 l1,0,0},{0,0,0,l1,0},{0,0,0,0,l1}};

(* The next two lines are creating the first two block rows of T *)
(* First, create the following matrix

        |       B11     B12     0       |
```

```
        |        B21      B22      B23      | *)

T = BlockMatrix[{{{{-l1,0,a1 l1,a2 l1,0,0,0,0,0}}},{f,g,h}}];

(* Padding the first two block rows with the necessary number of
columns of zeros *)

T = BlockMatrix[{{T,z[4,(W+1)^2-9]}}];

(* The following for loop calculates the remaining B matrices and
appends them to the exist matrix T, adding another block row to T
every time it goes through the loop except for the last time.  m
starts from 3 because the first two block rows for T have already been
found and ends with W+1 because this corresponds to an occupancy level
of W. *)

For[m=3, m<(W+2), m++,

(* Creates Bm(m-1) from B(m-1)(m-2) which is stored in f *)

Btemp = BlockMatrix[{{f,z[2 m-3,2]}}];
Btemp = Btemp + u1 IdentityMatrix[2 m-3];

(* Sets f to Bm(m-1) for next run through loop *)

f = BlockMatrix[{{Btemp},{z[2,2 m-3]}}];

(* Creates Bmm from B(m-1)(m-1) which is stored in g *)

Btemp = g - (l2+u1) IdentityMatrix[2 m -3];
```

```
temp = BlockMatrix[{{z[2 m-3,2], l2 IdentityMatrix[2 m-3]}}];
temp[[1,2]] = temp[[1,2]] + a1 l2;
temp[[1,3]] = temp[[1,3]] - a1 l2;
Btemp = BlockMatrix[{{Btemp, z[2 m-3,2]}}] + temp;
temp2 = {{a1 u21, a2 u21, -(l1+u21), 0},
                {a1 u22, a2 u22, 0, -(l1+u22)}};
temp2 = BlockMatrix[{{z[2,2 m -5], temp2}}];

(* Sets g to Bmm for next run through loop *)

g = BlockMatrix[{{Btemp},{temp2}}];

(* Creates Bm(m+1) from B(m-1)m which is stored in h *)
(* Sets h to Bm(m+1) for next run through loop *)

h = BlockMatrix[{{h,z[2 m-3,2]}, {z[2,2 m -1], l1 IdentityMatrix[2]}}];

(* If m does not equal W+1, then append next block row to T, but if m
equals W+1, skip this step and exit the for loop *)

If[m<(W+1),T = BlockMatrix[{{T},
                {z[2 m-1,(m-2)^2],f,g,h,z[2 m-1,(W+1)^2-(m+1)^2]}}]];
        ];

(* Done with for loop.  *)
(* Now calculate A0, A1, and A2 *)

A0 = l1 IdentityMatrix[2 W+1];
A1 = g;
temp3 = BlockMatrix[{{z[2 W+1,2],f}}];
```

```
temp3[[1,2]] = temp3[[1,2]] + W a1 u1;
temp3[[1,3]] = temp3[[1,3]] - W a1 u1;
A2 = temp3;

(* A is the sum of A0, A1, and A2 *)

A = A0 + A1 + A2;

(* Need to find the zero left eigenvector of A which is just the
transpose of the zero right eigenvector of the transpose of A. So
first let AT be the Transpose of A.  *)

AT = Transpose[A];

(* Calculating the following two matrices now as opposed to during the
iterations for R will reduce the computation time. *)

C1 = -A0 . Inverse[A1];
C2 = -A2 . Inverse[A1];

(* For initial iteration, set R equal to a matrix of zeros *)

R = z[2 W+1,2 W+1];

(* The following for loop finds R iteratively.  One thing to know is
that as l1 (external message arrival rate) approaches the maximum
allowable offered load for the given conditions, the number of
iterations required for convergence increases.  This means, in heavy
traffic, you need to do more iterations on R to get good results.
Currently, 250 iterations are performed. *)
```

```
For[k=1, k<251, k++,
        Rold = R;
        R = C1 + R . R . C2;
        ];

(* Finding Rdif allows one to tell if the solution for R has
converged. Rdif should be a matrix of zeros. *)

Rdif = R - Rold;

(* Adding R*A2 to BWW *)

temp4 = g + R . A2;

(* Appending BW(W-1) and BWW + R*A2 to T which yields the desired form
of T. *)

T = BlockMatrix[{{T},{z[2 W+1,(W-1)^2], f, temp4}}];

(* Need to find the zero left eigenvector of T which is just the
transpose of the zero right eigenvector of the transpose of T. So
first let V be the Transpose of T. *)

V = Transpose[T];
{pysd, pys} = Eigensystem[AT];
{Tvals, Tvecs} = Eigensystem[V];

(* After running this program in Mathematica, the eigenvalues of AT
are stored in pysd, and the eigenvalues of V will be stored in Tvals.
```

```
At this point, the lists pysd and Tvals have to be searched to find
the zero eigenvalue.  The corresponding eigenvector can then found as
the kth row of pys or Tvecs, if the kth eigenvalue is zero.  Note that
the way Mathematica stores the eigenvectors, they are already in row
form, so no transpose is needed to get the zero left eigenvector of A
or T.  As a check, let X be the determined zero left eigenvector.
Then doing the following command should yield a result of a zero row
vector. *)

(* Type 'X . A'  (without the quotes) if X is found from AT.
   Type 'X . T'  (without the quotes) if X is found from V. *)
```

The next section of code must be performed from the command line in Mathematica because the position of the zero eigenvalue of A and T has to be found by inspection.

```
(* This code will discuss how to use the data from the previous
program to do two things. *)

(* First, it explains how to find the maximum stable offered load from
the matrix A calculated above.  The left eigenvector of the zero
eigenvalue of A must be found and normalized.  Second, the left
eigenvector X of the zero eigenvalue of the matrix T found above must
also be calculated. *)

(* In order to find the maximum stable offered load, first the left
eigenvector of A that corresponds to the eigenvalue of zero must be
found.  Mathematica only finds the right eigenvectors of a square
matrix.  It is easy to show that the left eigenvectors are the
transposes of the right eigenvectors of the transpose of the original
matrix. *)

(* The following command was performed in the previous program but is
included here for reference. *)

{pysval, pysvec} = Eigensystem[AT];
```

```
(* Now list pysval to find out where in the list the zero eigenvalue
is.  (i.e. is it the first, second, third, etc.) This can be done by
typing: *)

pysval

(* After finding its spot (say it in the ith place), let py be the ith
right eigenvector of AT which can be done as follows: (Note that i in
the statement below should be replaced by the integer corresponding to
the position of the zero eigenvalue.) *)

py = {pys[[i]]}

(* Note that the way Mathematica stores the eigenvectors, py is
already a row vector and does not need to be transposed.  Now the py
vector has to be normalized.  Set up a column vector of ones of size
2W+1 if it is not already made. *)

ones = Table[1, {2 W+1}, {1}]
{{temp}} = py . e
py = py / temp

(* The last statement divides py by its length so that the new py has
length unity. At this point, all the elements of py should be positive. *)

(* To find the maximum offered for which the message occupancy
remains stable, type the following command: *)

{{maxt}} = py . A2 . e

(* Also, as a verification, the following command yields the current
external message arrival rate (which is the offered load): *)

{{curt}} = py . A0 . e

(* If curt > maxt, then the buffer occupancy is unstable (i.e. has a
non-zero probability of being infinity).  This implies that the
external message arrival rate (which is the reciprocal of the mean
interarrival time) should be reduced so that curt < maxt *)

(* The rest of the code here explains how to find the vector X which
is the left eigenvector of the zero eigenvalue of T. *)

(* In order to find the vector X that can be used to find the message
occupancy distribution and other statistics, the position of the zero
```

```
eigenvalue of V (which is the transpose of T) has to be found by
listing them as follows (If there is no zero eigenvalue then the
current system is unstable and the external message arrival rate
should be reduced.): First, list the eigenvalues of V (and T) by
typing *)

Tvals

(* After finding the zero eigenvalue of V, the corresponding right
eigenvector of V needs to be extracted from the matrix Tvecs of the
right eigenvectors of V (which are stored as a row vectors so the
transpose does not have to be taken):
Note: i in the formula below should be replaced by the integer
corresponding to the place of the zero eigenvalue. *)

X = {Tvecs[[i]]}

(* At this point, X may be a negative row vector, in which case the
following command should be typed: *)

X = - X

(* Once X is a positive row vector, it can be used in the following
program to find statistics like the message occupancy distribution and
mean waiting time. *)
```

The last program in the appendix takes the unnormalized vector $X$, which is the steady-state probability vector for the boundary occupancy states, and the matrix $R$ to find the matrix-geometric solution and the message occupancy distribution. Other statistics of interest are also calculated.

```
(* This code is a function that finds various statistics such as
message occupancy distribution, mean message occupancy, mean waiting
delay, mean system time (total time in the system), and probability of
the message occupancy not exceeding the window size. *)

(* A function that makes zero matrices of various sizes has been
created in mgm.m *)
(* z[i_,j_] := Table[0, {i}, {j}]; *)
(* Create a function that creates a matrix of ones *)
o[i_,j_] := Table[1, {i}, {j}];
```

```
(* N is a temporary vector that is used to find the normalizing factor *)

NV = z[1,2 W+1];
Q = BlockMatrix[{{ X[[Range[1,1], Range[1,1]]], z[1,2 W]}}];
NV = NV + Q;

(* In this loop, the vector X is partitioned into W+1 row vectors
each of size 2W+1.  Q is a matrix that stores these row vectors. *)

For[k=2, k<(W+2), k++,
    temp = BlockMatrix[{{ X[[Range[1,1], Range[(k-1)^2+1,k^2]]],
                        z[1,2 W+1-(2 k-1)]}}];
                        NV = NV + temp;
    Q = BlockMatrix[{{Q},{temp}}];
    ];

(* e is a column vector of size 2W+1 with all entries equal to one. *)
(* e = Table[1, {2 W+1}, {1}]; *)

(* NF is the normalizing factor.  temp holds the value of the last row
(i.e. buffer occupancy of W). *)

NF = (NV - temp) . e + temp . Inverse[IdentityMatrix[2 W+1] - R] . e;

(* Scale Q by NF which is a scalar. *)

Q = Q/NF[[1,1]];

(* Z is the probability that message occupancy does not exceed the
window size. *)

Z = o[1,W+1] . Q . e;

(* S will be the probability that the message occupancy does not
exceed the parameter chosen in the next for loop for maximum k.  To
get a good approximation on expected buffer occupancy, S should be
near one at the end of the for loop. *)

S = Z;

(* Scale and use temp in determining S and future rows of Q. *)
temp = temp/NF[[1,1]]

(* In this for loop, the remainder of the Q matrix and the message
occupancy distribution are found up to the maximum message occupancy
```

```
used in the for loop.  Here it is set at 101, so all calculations are
done through message occupancy of 100. *)

For [k=W+2, k<101, k++,
        temp = temp . R;
        Q = BlockMatrix[{{Q},{temp}}];
        S = S + temp . e;
        ];

{{Z}} = Z;
{{S}} = S;
(* S should be close to one.  If not, the maximum k has to be
increased to get accurate results on message occupancy moments. *)

(* QD is a row vector that maintains the probability distribution of
the system's message occupancy. *)

QD = Transpose[Q . e];

(* Need to find the mean message occupancy, mean delay, and mean system
time. To find the mean message occupancy, S must be very close to one. *)

temp = 0;
For [k=1, k<101, k++,
        temp = temp + (k-1) QD[[1,k]];
        ];
EBO = temp;

(* Let EW be the expected number of messages in the window. *)
temp2 = 0;
For [k=1, k<W+2, k++,
        temp2 = temp2 + (k-1) QD[[1,k]];
        ];
temp2 = temp2 + W (1-Z);
EW = temp2;

(* Let EQ be the expected number of messages waiting to enter the
queue. Then, EQ = EBO - EW. *)
EQ = EBO - EW;

(* Now we can find the expected waiting time and system time. *)
Wait = EQ/l1;
Syst = EBO/l1;
```

# Appendix B

# Calculation of the Real Service Time and Hyperexponential Distributions using Mathematica

This appendix describes the procedure and Mathematica code for finding a hyperexponential distribution that approximates the actual service time distribution for a random transmission. First, the code that calculates the first three moments of the actual random transmission service time distribution is given. Then, an iterative method for solving a system of nonlinear equations used to match these moments to those of a hyperexponential distribution. Finally, the Mathematica code that performs this iteration method is included. In the first program, the mean initial message transmission length is denoted by $m$, instead of $f$ which was used in Chapter 4.

```
(* This program will find the distribution of the length of a random
transmission with a block error rate of p and an initial message
transmission length that is geometrically distributed with mean m.
Although the actual distribution is messy and cannot be found in
closed form, the first three moments can be solved for (as shown
below) which can be used to get an approximate distribution, which is
a weighted sum of two exponential distributions by matching these
moments. *)

(* Actually, the Z-Transform of the distribution is found, from which
```

```
the moments can easily be calculated. *)

(* Mean initial message size in blocks *)

m = 20;

(* Block error rate, where errors are independent
The block error rate can either be set before running the program or
set here *)

p

(* Z-T for the random variable g of a block being in error *)

G[z_] = 1-p + p z;

(* Z-T for an initial message transmission, assumed to be a geometric
distribution *)

N0[z_] = z/(m (1-(z (m-1)/m)));

(* Z-T for the length of a message before its (i+1)th transmission
given that no transmissions have been made yet *)

T[i_,p_,z_] := T[i,p,z] = Compose[N0, Nest[G,z,i]];

(* To find the probability of the message being done by the ith
transmission, use the following command: (note that i should be
replaced by desired integer) (Also note that this answer does not give
the probability of success of the ith transmission) *)

i = 1 T[i,p,0]

(* Let R be the maximum number of transmissions allowed for a message
before being dropped.  (This is the retry limit.) *)

R = 8;

(* Formula for the expected number of transmissions for a message with
a retry limit of R *)

Et = R - Sum[T[k,p,0], {k,0,R-1}];

(* Z-T for the length of a random transmission, the derivation of
which is included in Chapter 4 *)
```

```
RT[z_] = Sum[T[k,p,z]-T[k,p,0], {k,0,R-1}]/Et;

(* Finding the first three noncentral moments of RT[z], saved in
M1, M2, M3 *)

DRT1[z_] = D[RT[z], z];
M1 = DRT1[1];

DRT2[z_] = D[DRT1[z], z];
M2 = DRT2[1]+M1;

DRT3[z_] = D[DRT2[z], z];
M3 = DRT3[1] + 3 M2 -2 M1;
```

Let a system of nonlinear equations be described by

$$\bar{f}(\bar{u}) = 0 \tag{B.1}$$

where $\bar{f}$ is a vector of equations, and $\bar{u}$ is a vector of unknowns. An iterative method for finding the solution $\hat{u}$ of the above system is outlined here.

Select an initialization vector $\bar{u}_0$. Then, perform the following series of calculations with $k = 0$ to find $\bar{u}_1$.

$$y_k = -\bar{f}(\bar{u}_k) \tag{B.2}$$

$$D_k = \nabla \bar{f}(\bar{u}_k) \tag{B.3}$$

$$\tilde{u}_k = D_k^T (D_k D_k^T)^{-1} y_k \tag{B.4}$$

$$\hat{u}_{k+1} = \hat{u}_k + \tilde{u}_k \tag{B.5}$$

Repeatedly increment $k$ by one and perform the calculations (B.2), (B.3), (B.4), (B.5) until $\bar{u}_k$ converges to the solution $\tilde{u}$. The necessary requirements to guarantee convergence of this method were not studied in detail. A basic observation is that the method is not guaranteed to converge if the number of unknowns is greater than the number of equations. Also, if the number of unknowns is less than the number of equations, there may be multiple solutions, and the solution the method yields

will depend on the point of initialization. The method is used here to find the four parameters $\alpha_1$, $\alpha_2$, $x_1$, and $x_2$ of a hyperexponential distribution that matches the first three moments, $\mu_1$, $\mu_2$, and $\mu_3$, of the real transmission service distribution. The parameter $\alpha_1$ is the probability of the transmission being of type one and having an exponentially distributed service time with mean $x_1$. Likewise, $\alpha_2$ is the probability of the transmission being of type two having an exponentially distributed service time with mean $x_2$.

The system of nonlinear equations that need to be solved is found by subtracting the first three moments of the hyperexponential distribution by those of the real service distribution to yield

$$\bar{f}(\alpha_1, \alpha_2, x_1, x_2) = \begin{bmatrix} \alpha_1 x_1 + \alpha_2 x_2 - \mu_1 \\ 2\alpha_1 x_1^2 + 2\alpha_2 x_2^2 - \mu_2 \\ 6\alpha_1 x_1^3 + 6\alpha_2 x_2^3 - \mu_3 \end{bmatrix} = 0 \tag{B.6}$$

Note that $\alpha_2 = 1 - \alpha_1$ so that there are only three independent parameters of the hyperexponential distribution, and the problem reduces to finding the solution to three equations with three unknowns. The Mathematica code for performing the iterations follows.

```
(* This file calculates the parameters a1, a2, x1, x2 for a
hyperexponential service distribution that matches the first three
(non-central) moments of the actual service distribution. *)

(* Need to put initial values here *)

mu1 = M1;
mu2 = M2;
mu3 = M3;
u = {{0.2}, {25}, {1}};
a1 = u[[1,1]];
x1 = u[[2,1]];
x2 = u[[3,1]];
a2 = N[1 - a1,30];

(* Start iterations.  Here, 100 iterations performed.  The necessary
```

```
number of iterations depends on the nonlinear system. *)

For[k=1, k<101, k++,
        f = {{a1 x1+a2 x2 - mu1}, {2 a1 x1 x1+2 a2 x2 x2 - mu2},
                {6 a1 x1 x1 x1+6 a2 x2 x2 x2 - mu3}};
        y = -f;
        Dee = {{x1-x2,a1,a2},{2 x1 x1-2 x2 x2,4 a1 x1,4 a2 x2},
                {6 x1 x1 x1-6 x2 x2 x2,18 a1 x1 x1,18 a2 x2 x2}};
        uadd = Transpose[Dee] . Inverse[Dee . Transpose[Dee]] . y;
        u = u + uadd;
        a1 = u[[1,1]];
        x1 = u[[2,1]];
        x2 = u[[3,1]];
        a2 = 1 - a1;
        ];
```

# Appendix C

# GPSS Simulation Code for the Buffered S-ARQ Protocol

The simulation for the buffered S-ARQ protocol used to obtain an external message occupancy distribution is given below with comments.

```
        SIMULATE
        REALLOCATE COM,50000

* SET WINDOW SIZE BY SETTING LAST OPERAND IN NEXT LINE
* CURRENTLY WINDOW SIZE = 8

        STORAGE   S(WINDOW),8
QDIST   TABLE     Q(BUFFER),0,1,102

* SET MEAN INTERARRIVAL TIME WHICH IS THE RECIPROCAL OF THE MEAN
* ARRIVAL RATE BY CHANGING THE SECOND OPERAND IN THE PARENTHESES IN
* THE NEXT LINE
* CURRENTLY THE MEAN ARRIVAL RATE = 0.02 = 1/50

        GENERATE  RVEXPO(1,50),,,,,1PF,2PL

* TABULATING THE QUEUE SIZE SEEN BY ARRIVING MESSAGES YIELDS THE SAME
* RESULTS AS THOSE TAKEN OVER TIME AVERAGES BECAUSE THERE ARE POISSION
* ARRIVALS

        TABULATE  QDIST
```

```
* COUNT KEEPS TRACK OF THE NUMBER OF TRANSMISSIONS A MESSAGE HAS HAD

        ASSIGN    COUNT,0,PF

* ERRBLOC KEEPS TRACK OF THE NUMBER OF BLOCKS IN ERROR DURING A
* MESSAGE TRANSMISSION

        ASSIGN    ERRBLOC,0.0,PL

* CURBLOC KEEPS TRACK OF THE NUMBER OR BLOCKS YET TO BE TRANSMITTED IN
* THE CURRENT MESSAGE TRANSMISSION

        ASSIGN    CURBLOC,RVEXPO(2,19.496),PL

* ARRIVING MESSAGES ENTER THE BUFFER

LINE    QUEUE     BUFFER

* WHEN A SPOT IN THE WINDOW IS OPEN THE NEXT WAITING MESSAGE GETS THAT
* SPOT AND DOES NOT LEAVE IT UNTIL THE ENTIRE MESSAGE IS RECEIVED
* CORRECTLY OR THE RETRY LIMIT IS REACHED

BWINDOW ENTER     WINDOW

* ONLY ONE MESSAGE CAN OCCUPY THE CHANNEL AT A TIME

SERVER  SEIZE     CHANNEL

* TEST TO SEE IF THERE ARE ANY MORE BLOCKS TO BE SENT IN THE CURRENT
* TRANSMISSION

NEXTB   TEST GE   PL(CURBLOC),0.0,TDONE

* IF THERE ARE REMAINING BLOCKS, THEN DECREMENT CURBLOC BY ONE AND
* SEND IT EACH BLOCK TAKES A FIXED TIME UNIT OF LENGTH ONE FOR
* TRANSMISSION

        ASSIGN    CURBLOC-,1.0,PL
        ADVANCE   1

* THE BLOCK ERROR RATE IS ONE MINUS THE FIRST OPERAND OF THE NEXT
* STATEMENT
* IF BLOCK SUCCESSFUL, GO SEND THE NEXT BLOCK

        TRANSFER  .9,,NEXTB
```

```
* IF THE BLOCK IS IN ERROR, THEN INCREMENT ERRBLOC AND GO SEND THE NEXT
* BLOCK

        ASSIGN    ERRBLOC+,1.0,PL
        TRANSFER  ,NEXTB

* RELEASE THE CHANNEL WHEN THE CURRENT TRANSMISSION IS FINISHED AND
* INCREMENT COUNT TO SHOW THAT THE MESSAGE HAS GONE THROUGH ONE MORE
* TRANSMISSION

TDONE   RELEASE   CHANNEL
        ASSIGN    COUNT+,1,PF

* THE MESSAGE THAT HAS JUST BEEN SENT WILL GO THROUGH A ROUND TRIP
* DELAY WHICH HERE IS ASSUMED TO BE EXPONENTIAL WITH MEAN LENGTH
* EQUAL TO THE SECOND OPERAND IN THE PARENTHESES

RTDELAY ADVANCE   RVEXPO(3,40)

* CHECK TO SEE IF ALL THE BLOCKS OF THE TRANSMISSION WERE SUCCESSFUL
* IF SO, THE MESSAGE IS DONE AND MUST LEAVE THE SYSTEM

        TEST G    PL(ERRBLOC),0.0,MESSDON

* IF NOT, THEN CHECK TO SEE IF THE RETRY LIMIT HAS BEEN REACHED
* IF SO, THEN THE MESSAGE MUST LEAVE THE SYSTEM

        TEST L    PF(COUNT),8,MESSDON

* IF NOT, THEN CURBLOC SHOULD BE SET TO ERRBLOC AND ERRBLOC SHOULD BE
* SET TO ZERO TO GET READY FOR THE NEXT TRANSMISSION

        ASSIGN    CURBLOC,PL(ERRBLOC),PL
        ASSIGN    ERRBLOC,0.0,PL

* THE MESSAGE WILL GO BACK AND WAIT FOR THE CHANNEL FOR A RETRY

        TRANSFER  ,SERVER

* ONLY WHEN A MESSAGE LEAVES THE SYSTEM DOES IT FREE UP A SPOT IN THE
* WINDOW
MESSDON LEAVE     WINDOW
        DEPART    BUFFER
        TERMINATE 1
```

```
START     10000
RESET
START     1000000
END
```

# Bibliography


[1] Ahmadi, Hamid and Parviz Kermani. "Throughput Analysis of a Class of Selective Repeat Protocols in High-Speed Environments." *IEEE GLOBECOM '89,* 26.4.1-26.4.9, 1989.

[2] Anagnostou, Miltiades and Emmanuel Protonotarios, "Performance Analysis of the Selective Repeat ARQ Protocol." *IEEE Transactions on Communications,* 34(2):127-135, February, 1986.

[3] Bertsekas, Dimitri and Robert Gallager. *Data Networks.* Prentice-Hall, Englewood Cliffs, NJ, 1987.

[4] Birchler, Mark and Steve Jasper. "A 64 Kbps Digital Land Mobile Radio System Employing M-16QAM." *1992 IEEE International Conference on Selected Topics in Wireless Communications,* 158-162, 1992.

[5] Crisler, Ken. "MIRS Packet Data Protocol Link Layer Specification." *Motorola,* Internal Document, 1992.

[6] Daigle, John. *Queueing Theory for Telecommunications.* Addison-Wesley, Reading, MA, 1992.

[7] Drake, Alvin. *Fundamentals in Applied Probability Theory.* McGraw-Hill, New York, 1967.

[8] Katz, Sharlene. "Session Admission Control for Multiple Access Communications Systems." *Ph. D. Dissertation, UCLA,* 1987.

[9] Kaul, A. "A Model for a Packet Switching Node including Packet Retransmission Effects." *Comsat Tech. Review,* 9(2B):669-688, 1979.

[10] Kleinrock, Leonard. *Queueing Systems, Vol. I.* John Wiley, New York, 1974.

[11] Konheim, Alan. "A Queueing Analysis of Two ARQ Protocols." *IEEE Transactions on Communications,* 28(7):1004-1014, July, 1980.

[12] Lee, H. and B. Ngo. "Queueing Analysis of the Selective Repeat ARQ Scheme." *IEE Proceedings-I,* 138(6):487-493, 1991.

[13] Neuts, Marcel. *Matrix-Geometric Solutions in Stochastic Models.* Johns Hopkins University Press, Baltimore, 1981.

[14] Neuts, Marcel. *Structured Stochastic Matrices of M/G/1 Type and Their Applications.* Marcel Dekker, New York, 1989.

[15] Saeki, Billy and Izhak Rubin. "An Analysis of a TDMA Channel Using Stop-and-Wait, Block, and Select-and-Repeat ARQ Error Control." *Transactions on Communications,* 30(5):1162-1173, May, 1982.

[16] Sundharadas, Reba. "Simulation of the Access Protocol for a Digital Radio System." *S.M. Dissertation, MIT,* 1992.

[17] Tanenbaum, Andrew. *Computer Networks.* Prentice-Hall, Englewood Cliffs, NJ, 1989.