Nested Window Flow Controls with Packet Fragmentation.

Gerald W. Shapiro

Center for Communications and Signal Processing
Department of Computer Science
North Carolina State University

CCSP-TR-88/21

June 1988

ABSTRACT

SHAPIRO, GERALD W. Nested Window Flow Controls with Packet Fragmentation. (Under the direction of Harry G. Perros).

This thesis presents a methodology for approximating the performance characteristics of a single hop data communications link operating under nested levels of sliding window flow control. Given such a network, we hierarchically reduce each level of flow control to a single queue whose characteristics approximate those of the original flow controlled link. The approximate queues are as simple as possible, so that the hierarchical analysis is as simple as possible. Specific attention is given to the fragmentation and reassembly of packets between protocol layers. In this respect the current analysis breaks new ground in the analysis of this type of queueing network.

The analysis is based on the solution to a set of probability balance equations which approximate the average behavior of the link. We compare the new procedure against exact numerical results, and it is found to work well.

We also compare the approximate solutions obtained under the new procedure for a single level of flow control against the approximation results obtained by using the flow equivalent server approach, first applied to this problem by Avi-Itzhak and Heyman. The new procedure is much superior in the case where packets get fragmented and reassembled.

ACKNOWLEDGMENTS

This research was supported by AIRMICS via the NCSU Center for Communications and Signal Processing, and by the Rome Air Force Development Center grant E-21-669-58 through the Georgia Institute of Technology. My deep thanks to these organizations.

I also wish to acknowledge my indebtedness to Dr.'s Harry Perros and Salah Elmaghraby. Both men have shown me every kindness and inspired me through their excellence as scholars.

TABLE OF CONTENTS

GLOSSARY OF SYMBOLS

In this glossary we define those symbols which are used in more than one Section of this thesis.

Symbol	Definition	Reference Page
W	window size	9
λ	arrival rate	16
B	batch arrival size	16
f	number of transmission packets per high level packet. $f-1$ is the maximum value of n_R	99
Pr{ A }	probability of event A	
Pr{ A | B }	probability of event A given B	
μ_X	transmission time	48
μ_A	acknowledgment time	48
$\pi_W(W)$	Probability that two queue closed network of Figure 3.2, page 55, has all tokens in the acknowledgment queue	54
$\pi_W(W,\gamma_\infty)$	As above when the transmission rate is replaced by $\gamma_\infty\mu_X$	75
n_A	number in acknowledgment queue	57
n_T	number in token queue	57
n_X	number in transmission queue	57
n_H	number in holding queue	57
n_S	number of packets in system	57 103
n_R	number of transmission packets in reassembly buffer	103
(i,j)	state where $n_S=i$, $n_A=j$	57
(i,j,k)	state where $n_S=i$, $n_A=j$, $n_R=k$	103
(j,k)	state where $n_A=j$, $n_R=k$ in closed network of Figure 4.4, page 102	110
$p_{i,j}$	probability of state (i,j)	58
$p_{i,j,k}$	probability of state (i,j,k)	103
$y_{j,k}, \bar{y}_{j,k}$	probability of (j,k) for two models	110 122
$q_{i,j}$	$\Pr\{n_H=i, n_T=j\}$	30
P_i	$\Pr\{ n_S=i \}$	59
γ_∞	state asymptotic value of $\frac{P_{n+1}}{P_n}$	72

b_i	$\Pr\{ n_A=W \mid n_S=i \}$	59
$\bar{b}$	an "average" b_i	61
b_∞	state asymptotic value of b_i	66
$r(\gamma_\infty)$	an estimate of b_∞ from (3.3.10)	76
a_i	$\Pr\{ n_A=W \mid n_S=i \}$ in model of Figure 4.3	105
$\bar{a}$	an "average" a_i	107
a_∞	state asymptotic value of a_i	110

CHAPTER 1

INTRODUCTION

The goal of this thesis is to present a methodology for predicting the performance of a computer communications link operating under a common type of communications protocol known as sliding window flow control. We use this introduction to briefly describe the computer communications environment under study and to introduce the terminology we will be using. We then describe the problem we wish to solve, and indicate how succeeding Chapters address that problem.

As often as possible, we will use boldface type to highlight a term when it is first used. The definition of the term can be found in the surrounding text.

1.1. The Sliding Window Flow Control Environment

There are many issues which must be dealt with in the design of a data communications network. Among these are how the data is to be organized, how to route data through the network, how to encode the data electronically on the transmission medium, and how to detect and recover from failure of the transmission medium to transmit with 100% accuracy. Since all messages accepted to the computer network must be stored, and buffer space is limited, the network designer must also find a way to prevent or recover from users overloading the system.

In order to simplify the task of designing a data communications system, most systems are designed using a "layered" approach. The tasks which must be accomplished to communicate are broken up into groups called layers. The layers are

hierarchical in nature; each layer provides a service to the layer above it, and receives service from the layer below it. While the services provided by each layer are well defined, the actual implementation of the service is hidden from the other layers. This allows layers designed separately to interact correctly as long as the interfaces are agreed upon.

The layers accomplish their functions by exchanging data and messages with the same layer in another machine, using the lower layers to move the data and messages. We refer to the pair of communicating layers as **peers.** Each peer, of course, must be using the same conventions and definitions, referred to as the layer **protocol.** The peer to peer communication at a given layer is transparent to higher layer peers, which only see that the desired service is performed.

Let us make this more concrete by considering an example. The International Standards Organization has developed a data communications standard, the Reference Model for Open Systems Interconnection, which defines seven protocol layers. These layers are depicted in Figure 1.1. The highest layer is the application layer. The lowest is the physical layer. The three highest layers, application, presentation, and session, are concerned with data formats and inter-process interaction, and do not directly provide any communications functions. When communication is established, these layers must exchange messages with their peers so that data is transmitted in a form desired by the receiver. We shall not discuss these layers any further here, as our interest is in the communication protocols.

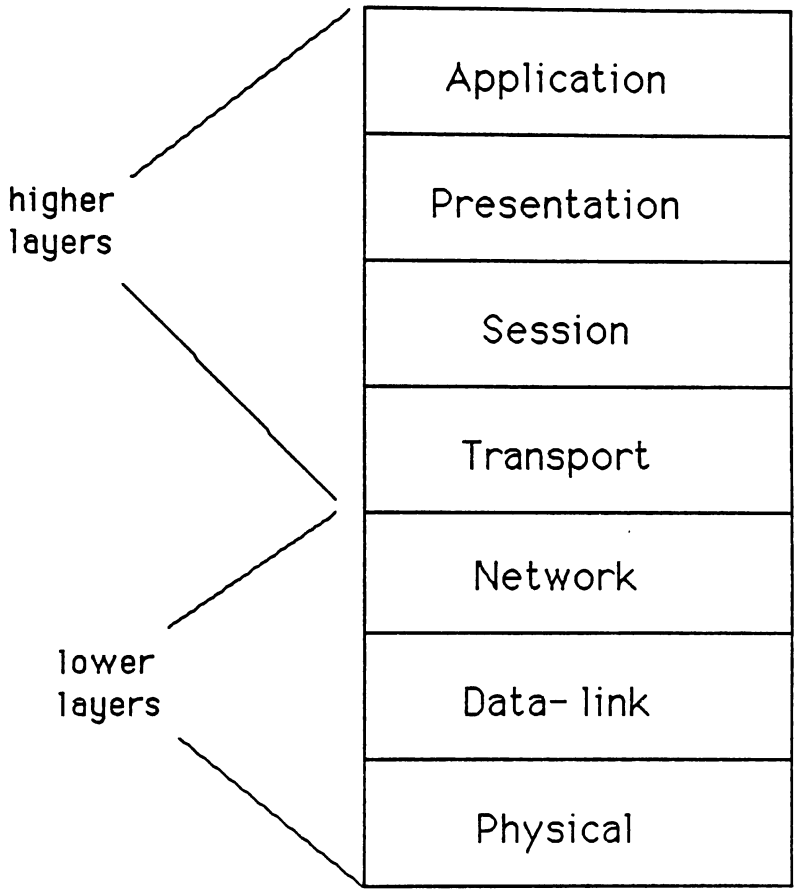


Figure 1.1 - The ISO Layered Architecture

The transport layer is the highest layer of communications protocol. The transport peer at the sending end receives formatted data from the session layer, and uses the lower layers to send that data to the receiving transport peer. The data units handled by the transport layer are called **messages**. In many cases the lower layers, network, data link, and physical, are provided by a service not under the control of the communicating host machines, as when one uses a public data network. The transport layer is thus often the lowest layer of communication protocol running in the hosts, and is designed to perform end to end communication functions relating to the hosts.

A primary function of the transport layer is to prepare the session layer messages for the network layer. Generally speaking, the amount of data that the session layer peers exchange in any one message is determined solely on the basis of organization of the data. For instance, in a file transfer application the session layer message will be a file. The network layer, on the other hand, restricts the length of the data units, called **packets**, that it handles. As we shall discuss in more depth shortly, a major function of the network layer is to avoid having too much work, (i.e. bits to be transmitted), in the network at any time. Having a maximum packet length allows the network layer to control the amount of work in the system by simply limiting the number of packets accepted. It is the function of the transport layer to break the arbitrarily sized session layer message into the necessary number of network sized packets. The transport peer at the receiving host must be aware of this **fragmentation** and reassemble the original session layer message before passing it up to the receiving

session layer peer. In this way the session layer is isolated from the lower layer protocols.

Another transport layer function is pacing. Either by agreeing on a data transmission rate, or by explicit "send"/"don't send" messages, the communicating transport peers limit how fast data is sent to the receiver. This prevents a fast sender from giving a slow receiver more data than it can buffer.

A third transport layer function is error recovery. Some networks may destroy a message in transit if the network becomes too congested. This may be done without notifying the sending transport peer. It is the responsibility of the transport layer to decide that a message has been lost and to re-transmit it.

Another function which can be considered a transport layer function occurs in the context of interconnected networks. We will return to this point after a discussion of the network layer.

A computer network consists of a number of machines interconnected in some fashion, so as to be able to exchange data. The network layer is responsible for routing and flow control within a network. When the transport layer passes a message to the network layer, broken up into one or more packets, the network layer must determine which sequence of transmission links in the network will be used to get those packets to the receiver. Even if there is a direct link the network layer may decide to route the packets through some intermediate nodes to avoid congestion on the direct link. The network functions are duplicated in each node in the network; thus, as opposed to the transport layer, the network layer is not just an end to end protocol.

As we mentioned previously, another means by which the network layer attempts to avoid congestion is by restricting the number of packets a given host may put into the network at any time. When there are too many packets in the network there may not be enough space in intermediate nodes to buffer all the traffic in transit through that node. Queueing delays for transmission may also become excessive, leading to poor service for every network user, not just the users with a lot of traffic. When the network layer limits the number of packets a host has in the system, the network peer at the destination node must send a message back to the sending peer to notify it when more traffic can be allowed in. In the event that the number of packets allowed into the network is limited, this limit applies to network sized packets, and the sending transport peer is responsible for buffering any packets which cannot be allowed into the network immediately.

Just as the transport layer must fragment session layer messages for the network layer, the network layer must also in some instances fragment its packets into units of a size acceptable to the data link layer which runs beneath it. The data units handled by the data link layer are called **frames**. For example, in the ARPANET the network layer has a maximum packet size of 8063 bits and the data link layer has a maximum frame size of 1008 bits. When fragmentation occurs between the network and data link layers each data link frame is transmitted separately through the network. It is the responsibility of the network peer at the exit node to reassemble the original network packet and pass it to the receiving transport peer. The need to reassemble network packets at the exit network node is another important reason for limiting the

number of packets allowed into the network. If the exit node should use all of its available buffer space storing fragments of many different network packets, it would not have any room to accept the completion of any one of them. This problem is referred to as reassembly deadlock.

There are two major classes of network protocols. The **datagram** protocols route each network packet independently, and packets can arrive in arbitrary order. No guarantee of delivery is made in these protocols. The **virtual-circuit** protocols deliver packets in the order which they were transmitted. These protocols offer guaranteed delivery, and attempt to recover from lost packets.

The set of hosts known to a given network is not universal. Each network has its own domain, although these may overlap. A machine which is connected to two different networks and allows messages to be sent from one network to the other is called a **gateway**. In the event that two machines which desire to communicate are located on different networks, the transport layer peers are responsible for routing messages from the sender's network to the receiver's network via gateways. In this case, the transport protocol is the only communication protocol common to both hosts, and the single transport connection will contain a concatenation of network connections beneath it.

The complication of packet fragmentation also arises in an inter-network environment. Different networks will have different maximum packet sizes. If a message must go from a network with a given maximum packet size to one with a smaller maximum packet size, the gateway transport layer must either reassemble the

entire message and then refragment it according to the new packet size, or fragment each of the larger network's packets individually. In the latter case, the fragmented fragments could be reassembled either at the exit node of the network with the smaller packet size or at the transport peer in the final destination.

(We note as an aside that some researchers reserve to the transport layer those functions which run in the host machines only. Since inter-network routing requires processing above the network layer at the gateways, these researchers define another protocol layer, the inter-network layer, which lies between the transport and network layers. In some instances, such as the ISO inter-network standard, the inter-network layer may provide a flow control function).

Below the network layer is the data link layer. The data link layer is responsible for monitoring the physical communication process. Each point-to-point link (**hop**) on the network is controlled by a data link layer; thus, in a network connection with many hops, there will be many independent data link connections. Although each hop has an independent data link connection and each could conceivably use a different protocol, to the best of the author's knowledge of existing networks, within a single network each data link connection uses the same protocol and maximum frame size. (To have data link frame fragmentation between hops within a network would introduce complications which no reasonable network designer would desire, and thus the designers require that all nodes in a network use the same data link protocol). This means that there will be no fragmentation of data link frames between hops, unlike the inter-network environment.

The primary function of the data link layer is to attempt to detect transmission errors. In a typical data link protocol the receiving peer will check an incoming frame's checksum with that computed by the sending peer, and if the frame determined to have transmission errors in it the receiver will either request that the sender re-transmit that frame, or, (as in the IEEE 802.x local area network standards), the receiving data link peer will refrain from passing the faulty frame to the higher layer peers, and depend upon the error recovery functions in the higher layers to cause a re-transmission. If the frame is determined to be correct, an acknowledgment of receipt is sent.

The data link peers also are responsible for managing the buffer space allocated to them by their host machine. In order to prevent its buffer space from being filled with traffic from a fast sender, the receiving data link peer can prohibit the sending peer from transmitting frames temporarily, much as the transport peers do.

The lowest protocol layer, the physical layer, is concerned with the representation of data on the transmission medium. It is of no further interest to this study.

A simple mechanism which is commonly used by designers to help achieve the error recovery and flow control functions of the transport, inter-network, network, and data link layers is a **sliding window flow control**. In a sliding window flow control the layer peers agree upon the maximum number of data units, (frames, packets, or messages), which the sender may transmit without acknowledgment of correct receipt. This maximum number is referred to as the **window** size of that connection. We shall denote the window size of a connection by W.

In the remainder of this thesis, whenever we are discussing the behavior of a sliding window flow control protocol without specific reference to a protocol layer, we shall use the term packet to refer to the data units being exchanged, although strictly speaking the term packet is reserved to network layer data units. When we are specifically speaking of network layer packets, we shall make that clear.

In a connection using sliding window flow control, the sending peer numbers its outgoing packets with successive numbers from 0 to $W-1$, and stores each one in its local buffer space. If the receiving peer finds that a packet is received incorrectly, a request to re-transmit is sent identifying the packet in error by its sequence number. When the receiving peer receives a correct packet, it sends an acknowledgement message to the sender, also indicating the sequence number of the correctly received packet. Upon receipt of an acknowledgment of correct reception, (or simply an **acknowledgment**), the sender can free up the buffer space holding that packet, use the newly freed buffers to accept a new packet, and reassign the sequence number to the new packet.

Whenever the sender has W packets unacknowledged by the receiver, the sender cannot send any more packets until an acknowledgment arrives, and then the sender may only transmit as many new packets as there are acknowledgments received. It is useful to think of the acknowledgment messages as **tokens**. In order to transmit a packet, the sender must have a token available. The token is assigned to a packet, and travels with that packet to the receiving peer. When the receiver receives a correct packet, it returns the token to the sender in an acknowledgment. In some pro-

tocols, a single acknowledgment message may be allowed to confirm correct receipt of more than one packet. In this case, we view the acknowledgment as containing multiple tokens.

Let us see how sliding window flow control helps the different protocol layers achieve their goals. The transport layer (and sometimes the data-link, network, and inter-network layers) need to exchange information with their peers indicating whether or not a transmitted packet was received correctly. This is inherent in sliding window flow control.

Sliding window flow control also protects the buffer space at the receiver, a goal of each of the data link, network, and transport layers. The receiver has to buffer at most W packets, and can delay sending an acknowledgment if its buffers are full. Sliding window flow control also limits the buffer requirements of the sending node in the data link layer. The sending data link peer only needs to keep a copy of the W outstanding frames at any time, and can refuse to send acknowledgments to its senders if its buffers are full.

By limiting the number of packets outstanding on any connection, sliding window flow control helps the network layer limit the number of packets in the network. Packets waiting to get into the network must be stored by the network user, forcing congestion outside the network boundary.

In this section we have briefly touched on network design issues and their protocol solutions. For a more complete discussion of network design the reader is

referred to the text by Tanenbaum, [Tane88]. An excellent discussion of the problems in computer networks which mandate flow control, the paradigms used to address these problems, and actual network implementations of flow control schemes, including sliding window flow control, can be found in Gerla and Kleinrock, [Ger-Kle80].

1.2. Problem to Address

In the previous section we introduced the concept of sliding window flow control. The use of this mechanism, in some variant, is ubiquitous in protocol design. Sliding window flow control offers a convenient means of addressing some of the problems faced by protocol designers, but it is not without its drawbacks.

In the ideal situation, we would like to see the slower process, sender or receiver, always busy while there is work to be completed. If the sender is slower than the receiver, (as when the transmission time is very long), we would like to see the transmitter release each packet as it arrives. Under sliding window flow control this will occur if there is always a token available each time a packet arrives for transmission; however, this will not always be the case for the following reason. Computer traffic tends to be bursty, with packets arriving in close succession followed by a period of relatively few arrivals. This is compounded in the layered environment by fragmentation, which is in effect a simultaneous arrival of a group of packets to the lower layer. This traffic characteristic results in an occasional backlog of packets to be sent larger than the window size of the connection. The sender is then forced to buffer the excess and be idle while all the tokens are in use, until an

acknowledgment arrives.

When the receiver is the slower process, occasional idleness at the sender when there are packets available to transmit is not necessarily a concern. As long as a token returns in time to allow the next packet to be transmitted and arrive at the receiver before the receiver runs out of work, there will be no reduction in the system throughput. Alas, it is the nature of sliding window flow control that this will not be the case. Tokens returning in acknowledgments use the same transmission network as data packets, and are thus subject to unpredictable delays. In the worst case, we could have all of the tokens in acknowledgments making their way back to the sender, while the receiver is idle and the sender has packets to send. In this situation, no work is being done, even though both the sender and receiver are willing and able.

In light of the above discussion, we see that an important issue for network designers to consider is determining what exactly the throughput will be in a system using sliding window flow control. It is this issue which we propose to address, at least in part, in this thesis.

To determine the throughput and other performance measures such as end to end delay and buffer utilization in a network using sliding window flow control, requires an analysis which goes beyond application of the standard queueing network solution techniques, such as the separability results of the BCMP theorem and the approximation algorithm of Marie, as there are features of the sliding window flow control environment which violate the assumptions needed to apply these techniques. One aspect of the sliding window flow control environment which is not accounted for in

the above mentioned procedures is packet fragmentation and reassembly. Typical queueing network solving tools do not allow entities to split and re-coalesce. A second exceptional feature of sliding window flow control is the token/acknowledgment structure. The transmission server is forced to be idle when there are no tokens available. The frequency with which this happens depends upon the load on the system and the rate at which acknowledgments return to the sender, and is thus an integral part of the analysis. The existing general purpose techniques do not allow for the synchronization of two processes, transmission and acknowledgment, in their formulation.

We will begin our investigation in Chapter 2 by reviewing the published literature for modelling and solution techniques of sliding window flow controlled networks.

In Chapters 3 and 4 we present new methods for analyzing sliding window flow control networks. We focus on networks with a single hop and fixed session layer message size. The latter situation occurs in applications such as bulk file transfer where the session layer data units are of constant size. Restriction to a single hop network limits the analysis somewhat. We discuss the full details of the models in these Chapters, but briefly, Chapter 3 addresses a single level of sliding window flow control with arriving packets fragmented. Chapter 4 looks at multiple layers of sliding window flow control. In each of these Chapters an algorithm for approximating the given system's behavior is presented. The approximation results are compared against exact numerical results and simulation experiments.

Our goal in each of the models we consider is to reduce the given sliding window flow controlled network, in which data packets and acknowledgments are transmitted, to a flow-equivalent single server queue which handles only data packets. Acknowledgments do not appear in the approximation model. The service characteristics of the server in the approximation model are calculated to reflect the degradation in data packet transmission capability caused by using sliding window flow control. We attempt to duplicate in the approximate queue the mean and distribution of transmission service time seen by data packets in the actual system, while keeping the characterization of the approximate server as simple as possible.

This approach allows us to replace a sliding window flow controlled subnetwork, which cannot be handled by standard queuing network analysis procedures, by a single flow-equivalent queue which can be handled by these procedures. This effectively extends the class of networks which can be handled by the existing techniques.

The approximation techniques can also be used to hierarchically analyze a link with multiple levels of sliding window flow control. Consider, for example, a link with two levels of sliding window flow control. The higher level has as its delivery mechanism a network which is a sliding window flow controlled network. The lower level has as its delivery mechanism the physical link, which we represent as a single server queue. In the hierarchical method we first reduce the lower level sliding window flow controlled network to a single server queue. We then replace the lower level in the original model by this single server queue. The higher level in the new

model now has a single queue as its delivery mechanism, rather than a sliding window flow controlled network, and can be analyzed as the lower level was. Examples and details of the hierarchical method are presented in Chapters 2 and 4.

We conclude this thesis in Chapter 5 with a summary of the results and some reflections on possible extensions to this research.

1.3. Definition of Symbols

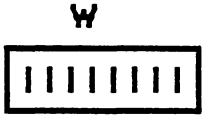
We conclude this Chapter with a set of Figures introducing the symbols which will be used in this thesis.

Figure 1.2(a) is the symbol for a single server queue. We will indicate the characteristics of the server by a notation above the circle. A single parameter, i.e. μ, indicates exponential service time with rate μ. An indexed parameter, i.e. $\mu(n)$, indicates state dependent service with rate $\mu(n)$ when there are n entities in the queue. E(μ,k) indicates Erlang-k service with mean $1/\mu$. The notation $\lambda^{[B]}$ in Figure 1.2(a) indicates that arrivals occur in batches of size B according to a Poisson process with mean rate λ.

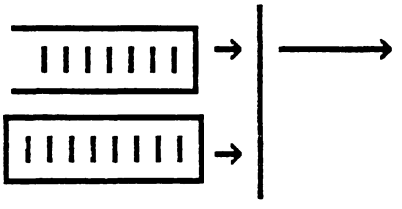
Figure 1.2(b) will be used to represent the queue of available tokens at the sender. A symbol above or below the queue represents the window size.

Figure 1.2(c) is a join symbol. Once an entity is present in each queue, they are joined to form a single entity and released. This symbol is used to indicate assigning a token to a packet. Figure 1.2(d) is a fork symbol. It represents the separation of previously joined entities.

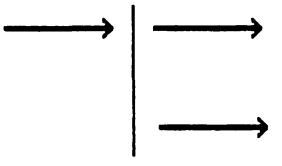
(a) - queue and server



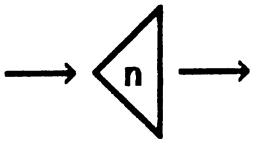
(b) - token queue



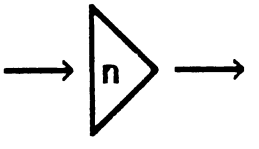
(c) - A join



(d) - A fork



(e) - fragmentation



(f) - reassembly

Figure 1.2 - Symbols

Figure 1.2(e) is a fragmentation symbol. One entity entering on the left becomes n entities upon leaving. There is no delay associated with a fragmentation. Figure 1.2(f) is a reassembly symbol. Once n fragments have arrived at this juncture, a single composite entity is released. There is no delay between the time the n^{th} fragment arrives and the time the composite entity is released.

CHAPTER 2

LITERATURE REVIEW

In this Chapter, we survey papers dealing with end to end window flow control which have appeared in the archival literature. We also survey selected papers from published conference proceedings, and two unpublished manuscripts.

Perhaps the earliest paper analyzing the sliding window flow control problem was published in 1975. Pennotti and Schwartz [PenSch75] analyze a single level of flow control on a multi-hop connection. There is no flow control between hops, only on the end to end connection. A virtual-circuit protocol is assumed, i.e, packets are delivered in the order in which they were transmitted. It is further assumed that order is preserved at each hop along the path. The model for this network is a tandem configuration, (queues in series), as shown in Figure 2.1. Messages on the flow-controlled virtual circuit arrive one at a time to the first queue according to a Poisson process with rate λ_0. After completing service at queue i, i = 1,2,...,M-1, the message queues for service at queue i+1. Messages departing from queue M leave the system. In addition to this traffic, referred to as the link traffic, there are external arrivals to each queue. External arrivals occur according to a Poisson process, with parameter λ_i at queue i. External arrivals leave the system after receiving one service. We assume sufficient buffers so that no packets, link or external, are rejected due to lack of buffers. Both external arrivals and link traffic require an exponentially distributed amount of service at each queue, the service occurring at rate μ_i at queue i. The service time for a given link packet is an independent random variable at each queue.

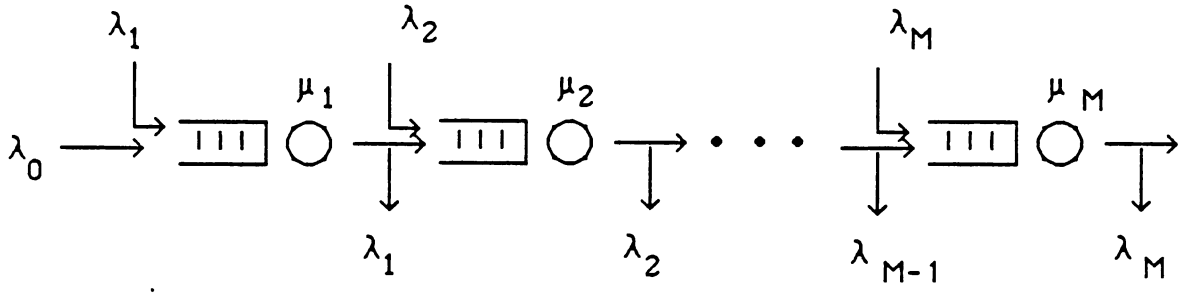


Figure 2.1 - Model of Pennotti and Schwartz

(This is the well known "independence assumption", without which much analysis of computer communication networks would not exist). There is a limit of W link packets allowed in the system at any point in time. There is no explicit acknowledgment mechanism in this model. Link arrivals which occur when there are already W link packets in the system are thrown away. Once there are fewer than W link packets in the system, (once a link packet departs queue M), the next arrival will be accepted. Arrivals will continue to be accepted until the number of link packets in the system reaches W, at which point arriving packets are again rejected. This assumption we shall call the **loss model assumption.**

Under the loss model assumption we can represent the model of Figure 2.1 as the closed network of Figure 2.2, with W link packets circulating. The closed network has the same topology as the original, except that an additional queue, queue M+1, is inserted after queue M and before queue 1. Service rate at queue M+1 is λ_0. When all W link packets are in the transmission queues, (queues 1 through M), queue M+1 is idle. When there are fewer than W link packets in the transmission queues, queue M+1 delivers link packets to queue 1 at rate λ_0. We see then that queue M+1 represents the link packet arrival process under the loss model assumption. Having queue M+1 be idle when all link packets are in transmission is the same as rejecting arrivals which occur during this period.

The network of Figure 2.2, with link packets using a closed network and external arrivals seeing an open network, and having exponential servers at each queue, can easily be solved for the exact probability distribution of the number of link and

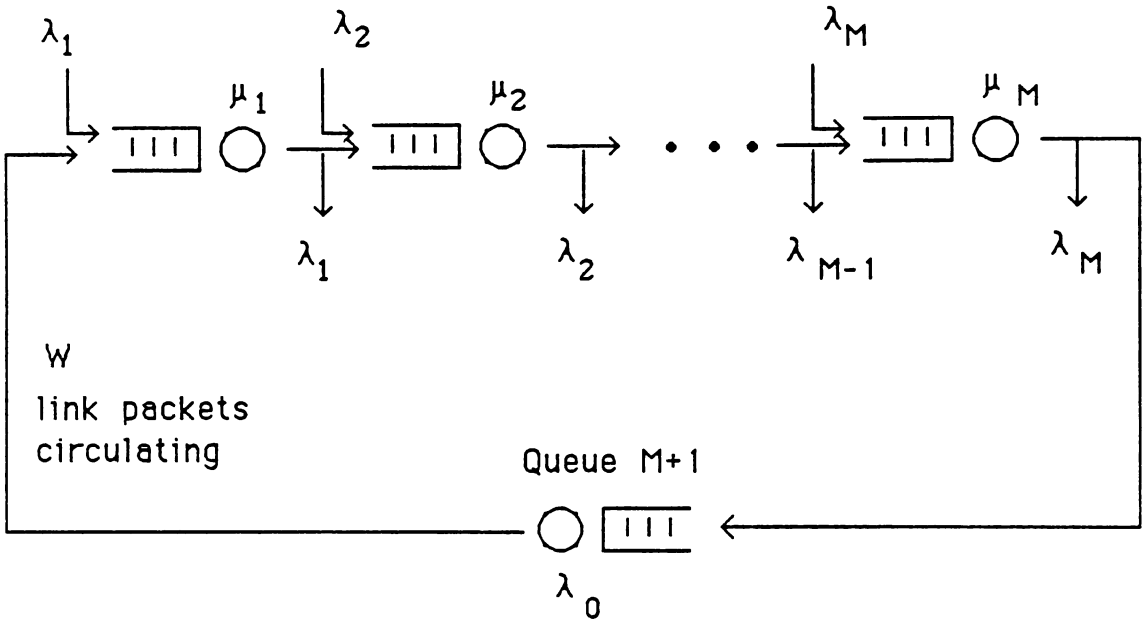


Figure 2.2 - Loss Model

external packets at each queue, from which other performance measures, (e.g. transmission delay and system throughput), can be calculated. If we denote by n_i the number of link packets in queue i, m_i the number of external packets in queue i, $i=1,2,\cdots,M$, and let $\bar{n}$ and $\bar{m}$ be the vectors with components n_i and m_i, respectively, then the probability of the state $(\bar{n},\bar{m})$ is given by

$$\Pr\{\bar{n},\bar{m}\} = \Pr\{\bar{0},\bar{0}\} \prod_{i=1}^{M} \frac{(n_i+m_i)!}{n_i!\, m_i!} \left(\frac{\lambda_0}{\mu_i}\right)^{n_i} \left(\frac{\lambda_i}{\mu_i}\right)^{m_i} \tag{2.1}$$

If we look only at the marginal distribution of link packets, we have

$$\Pr\{\bar{n}\} = G \prod_{i=1}^{M} \left(\frac{\lambda_0}{\mu_i - \lambda_i}\right)^{n_i} \tag{2.2}$$

where G is a constant independent of $\bar{n}$. The form of (2.2) is referred to as **product form**, since the probability of state $\bar{n}$ is a product of M terms each involving only one value of n_i. The i^{th} factor in (2.2) is proportional to the probability of having n_i customers in an M/M/1 queue with arrival rate λ_0 and service rate $\mu_i-\lambda_i$; thus the state probability of $\bar{n}$ has the form of the product of marginal distributions. We note also that (2.2) is the solution of the closed network in Figure 2.2 with no external arrivals, and service rate at queue i of $\mu'_i = \mu_i - \lambda_i$, i.e., we can account for the external arrivals' effect on the distribution of n_i simply by reducing the service rate at queue i by the external arrival rate at that queue! This technique has been referred to as the method of adjusted rates, (see [Reis82]).

Chaterjee, Georganas, and Verma [ChGeVe77] extend the model of [PenSch75] to the situation where the transmission network is a queueing network with random

routing, as opposed to a tandem configuration. This extension models a datagram network protocol. As in the previous model, flow controlled packets arrive one at a time according to a Poisson process, each transmission queue has Poisson external arrivals, the independence assumption is used, and the loss model assumption is made. The original network under these conditions is represented as a closed network with an additional queue linking the network exit node(s) with the input node(s). As before, this additional queue has service rate λ_0 and represents the arrival process of link packets under the loss model assumption. With exponential service rates at each queue, the joint probability distribution of $\bar{n}$ and $\bar{m}$ has a form very much like (2.1).

Martin Reiser [Reis79] presents a loss model of multiple independent sliding window flow controlled connections sharing a communications network. (A similar analysis appears in [LaPuMi79]). As in [PenSch75] this model assumes that each of the R connections is a virtual circuit, and that its transmission path is thus a series of queues. The independence assumption is used. Different connections have different entry and exit nodes, and use a different sequence of transmission links, although connections share some links. As in [PenSch75] there is no explicit hop level flow control in this model. The author assumes that the delays due to the data link protocol are incorporated into the link service time. Furthermore it is assumed that buffer space at each transmission node is sufficient to ensure that no packets are rejected due to lack of buffers.

A source queue models the arrival process of packets for each connection. As in [PenSch75] the service rate of the source queue for connection r is the packet arrival rate λ_r of connection r. When all W_r tokens of connection r are in transit, the source queue for connection r is empty, and thus the arrival process quiesces.

Acknowledgment packets are included in this model. It is assumed that the acknowledgments use the same transmission path as the data packets, and the service rate at each transmission link is reduced by the mean rate of work required to transmit acknowledgments, (much as Pennotti and Schwartz account for external arrivals in (2.2)). The acknowledgment path is then represented by a simple delay, the length of which is the average delivery delay along the transmission path. This approximation requires an iterative solution, as the average delivery delay is influenced by the average number of tokens in the acknowledgment path, which in turn depends upon the average delivery delay. Acknowledgments are delivered to the source queue, thus linking the output to the source.

As in the previous model, the loss model assumption reduces the network to a closed network. In this model the closed network has R classes of customers, one for each connection. Not surprisingly, this extension to the loss model occurred on the heels of Reiser's development of an efficient algorithm for solving multi-class closed queueing networks. This procedure, called mean value analysis, or MVA, is exact if the network has only BCMP type stations, (see [BCMP75]). The major restriction of BCMP stations in this context is that the service distribution at each transmission queue must be identical for each class if exponential service time distributions are

used. Approximation procedures are presented which remove this restriction. A small test case validation is presented, and the approximation appears to be reasonable, particularly for predicting the throughput of each virtual circuit.

The loss models attempt to analyze a sliding window flow controlled network by assuming that packets never queue for admission to the network. Another approach was taken by Kleinrock and Kermani [KleKer80]. These authors analyze a sliding window flow controlled network under saturation, that is, there is always a packet at the sending station when a token returns. This analysis also differs from the previous ones in that it explicitly models two features of sliding window flow control ignored in the previous models: message rejection at the destination node due to lack of buffers, and retransmissions by the sending station due to time-outs. A single connection is modelled, with no packet fragmentation. The analysis is modular; separate models are developed for the sending station, network, and destination node. The network is not modelled to any precision. It is simply assumed that both the transmission and acknowledgment delivery delays are exponentially distributed, with the same mean. Thus the round trip delay for a token has an Erlang-2 distribution. The probability of a time-out at the sending station can be calculated from this distribution and from the probability of having a packet rejected at the destination node.

The destination node is modelled as a single server queue with an exponential service rate and a finite queue. Packets arriving when the queue is full are rejected (lost). Assuming that the packets are delivered by the transmission network according to a Poisson process, (the mean of which depends upon the number of re-

transmissions), the probability of rejection can be calculated as the probability of an M/M/1/B queue having B customers in its queue. The equations for the probability of retransmission and the probability of destination node rejection both depend upon the two unknowns, the rejection probability and the re-transmission probability. The two equations are solved numerically for the estimates of these values.

Analysis of a sliding window flow controlled link under saturation can give some idea of the maximum throughput achievable by the network, but is not useful for performance prediction under normal loads.

In his 1979 paper, Reiser indicates dissatisfaction with the loss model approach to analyzing sliding window flow control. A primary weakness of this analysis is that it does not allow for true end to end delay prediction, since there is no means of evaluating the queueing delay which packets will incur waiting for a token.

It is also unclear how to set the rate λ in the queue which generates arrivals in these models. The true arrival rate λ_0 is not appropriate since the effective rate at which work enters the system is λ (1 - Pr{ source queue empty }). Reiser suggests solving the model using the true arrival rate λ_0, getting an estimate of Pr{ source queue empty }, and then re-solving the model with source server rate

$$\lambda' = \frac{\lambda_0}{1-\Pr\{\text{ source queue empty }\}}$$

This will allow the rate at which work enters the system to be closer to λ_0. Of course in solving the model a second time a new value of the probability that the source queue is empty will result. The revision of λ' and re-solution of the model can be

repeated until there is convergence.

While such an approach may be useful in raising the admitted traffic rate to the true value, the mechanism of the source queue itself loses some of the flavor of sliding window flow control, as *every* returning token in the loss model waits for a packet, whereas in the actual system some tokens are used immediately by packets enqueued for admission to the network.

In any event, the loss models are better than no analysis at all. Further progress in modelling sliding window flow control awaited further theoretical developments, which were not long in coming.

In 1981, Reiser [Reis81] drew upon the flow equivalent server technique first presented by Avi-Itzhak and Heyman [AviHey73], to develop a model of sliding window flow control which includes admission delay. The model in this paper is of a single connection. Packets arrive one at a time according to a Poisson process to a holding queue, (see Figure 2.3). If there is a token available, the packet is paired with the token, and the pair is delivered to the first transmission queue; otherwise, the packet waits in the holding queue for an acknowledgment. As in [Reis79], the flow controlled connection is a multi-hop virtual circuit represented by a tandem configuration. Service time is exponentially distributed at each queue, and the independence assumption is used. There is no explicit modelling of hop level flow control; it is assumed that the delays due to this level of flow control are incorporated into the link's service time. Packets are individually acknowledged. The acknowledgment path is modelled as a delay, and the effect of the acknowledgments on the

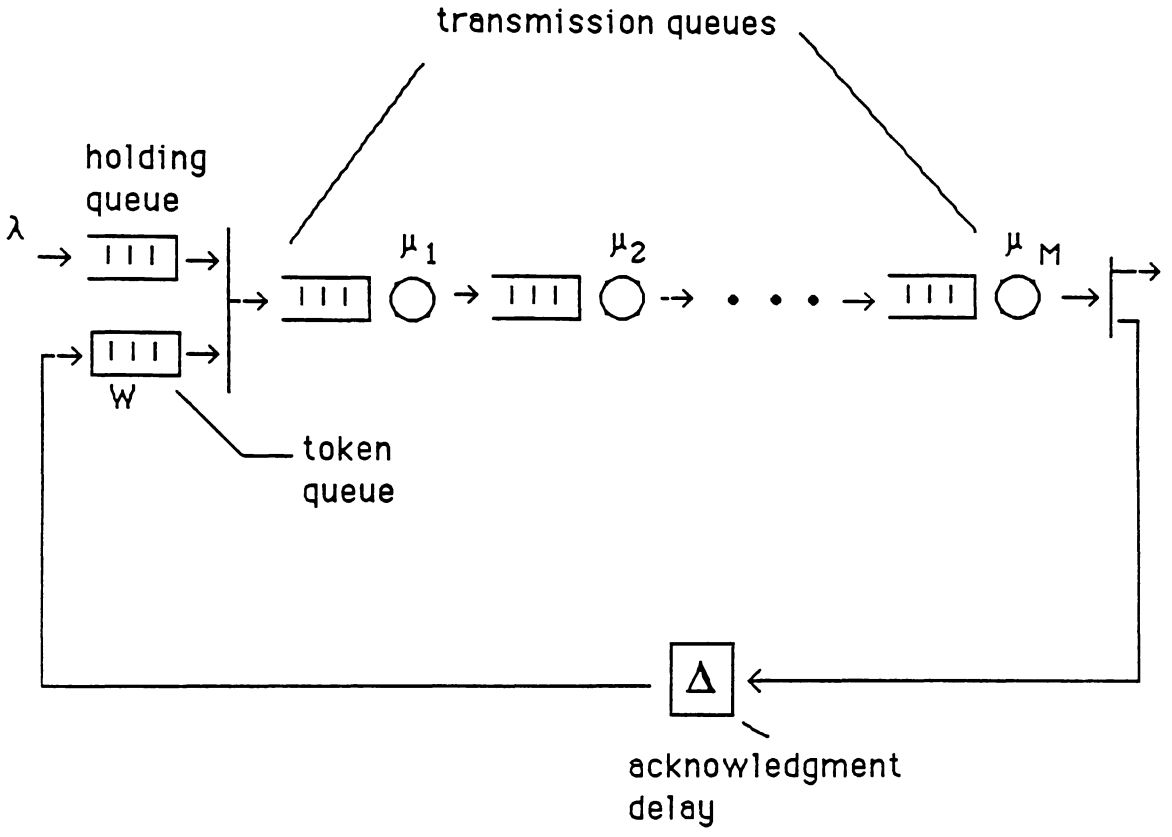


Figure 2.3 - Model of [Reis81]

transmission delay is accounted for by reducing the service rate at the transmission queues.

Denote by n_H the number of packets waiting for admission to the system, and by n_T the number of tokens available at the sending station. W is the window size, and thus $W-n_T$ is the number of packets in transmission plus the number of acknowledgments in transmission.

The model is analyzed by replacing the transmission links and acknowledgment delay by a single queue with state dependent service rate $\gamma(W-n_T)$. The resulting model is shown in Figure 2.4. We note that $\gamma(W-j)$ represents the rate at which tokens return to the token queue when $n_T = j$, and thus the token round trip delay when $n_T = j$ is approximated by $(W-j)/\gamma(W-j)$. The mean data packet delivery delay when $n_T = j$ is thus approximated by $(W-j)/\gamma(W-j) - \Delta$, the mean round trip delay less the acknowledgment delay.

The model of Figure 2.4 is fairly easy to analyze. Observing that n_H and n_T cannot be simultaneously non-zero, the system state probabilities

$$q_{i,j} = \Pr\{ n_H = i, n_T = j \}$$

obey the following steady state balance equations

$$\begin{cases} \lambda q_{0,W} = \gamma(1) q_{0,W-1} \\ (\lambda + \gamma(W-j)) q_{0,j} = \gamma(W-(j-1)) q_{0,j-1} + \lambda q_{0,j+1}, \quad j = W-1, W-2, \cdots, 1 \\ (\lambda + \gamma(W)) q_{0,0} = \lambda q_{0,1} + \gamma(W) q_{1,0} \\ (\lambda + \gamma(W)) q_{i,0} = \gamma(W) q_{i+1,0} + \lambda q_{i-1,0}, \quad i = 1,2,\ldots,\infty \end{cases} \tag{2.3}$$

The values $q_{i,j}$ which solve (2.3) are

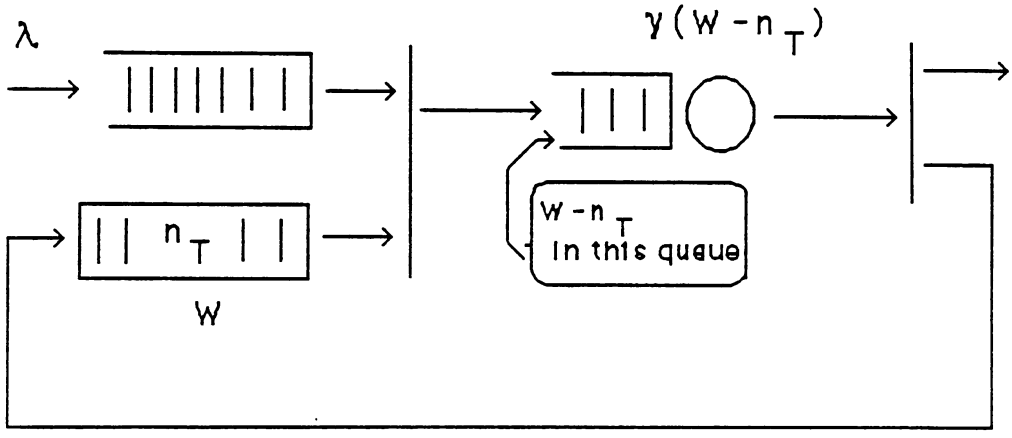


Figure 2.4 - Norton's Approximation Model

$$q_{0,W-j} = \frac{\lambda^j}{\prod_{k=1}^{j} \gamma(k)} q_{0,W} \quad , \quad j=1,2,\cdots,W$$

$$q_{i,0} = \left[\frac{\lambda}{\gamma(W)}\right]^i q_{0,0} \, , \quad i=0,1,\cdots,\infty$$

$$= \left[\frac{\lambda}{\gamma(W)}\right]^i \left[\frac{\lambda^W}{\prod_{k=1}^{W} \gamma(k)}\right] q_{0,W}$$

$q_{0,W}$ is determined from the normalization

$$\sum_{j=0}^{W} q_{0,j} + \sum_{i=1}^{\infty} q_{i,0} = 1$$

The probabilities $q_{i,j}$ give a stochastic description of the model. Using $q_{i,j}$, $\gamma(j)$, and Δ, performance measures can be calculated.

The crucial step in this method is calculation of the values $\gamma(j)$, $j=1,2,\cdots,W$. This is done using the so-called Norton's approximation server for the transmission network. (This name for the flow-equivalent server technique was coined by Chandy, Herzog, and Woo [ChHeWo75], due to its similarity to the Norton's equivalent network theorem in electrical circuit analysis). Using this technique, $\gamma(j)$ is calculated by evaluating the closed network of Figure 2.5, which is the transmission and acknowledgment paths of Figure 2.3, "short-circuited". $\gamma(j)$ is set to the steady state throughput of the network in Figure 2.5 with j packets circulating. These values are easily calculated using MVA, provided the queueing stations are exponential or some other type of BCMP station. Reiser compared the throughput estimates of a flow controlled net using this technique and using the technique of [Reis79], and found the

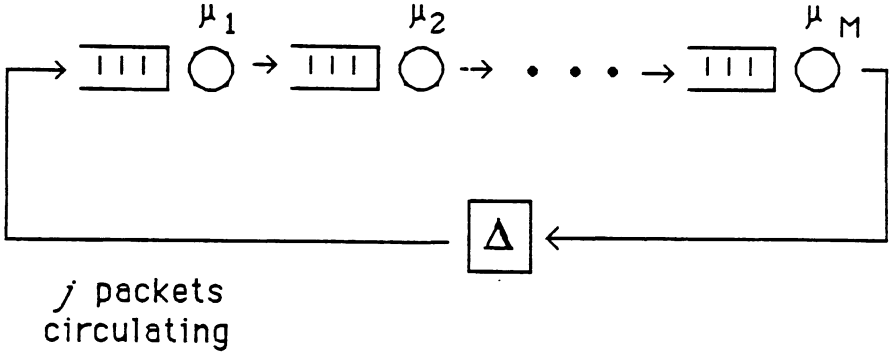


Figure 2.5 - Throughput Model

new method to be superior. We note that the Norton's approximation analysis can be just as easily applied to a datagram network, in which case the transmission network is represented by a network of queues with random routing.

Thomasian and Bay [ThoBay84a] analyze a model with admission delays and multiple virtual circuits. The approach is to generate a separate flow equivalent server with state dependent service rates for each virtual circuit, using approximation methods such as those in [Reis79] to estimate the network throughput characteristics seen by each connection. This decomposes the model of R virtual circuits to R models of the form of [Reis81].

Gihr and Kuehn [GihKue85] extend the model of [Reis81] by pointing out that it can be used for fragmented arrivals to the holding queue by simply modifying the balance equations (2.3) to the case of batch arrivals. (The modified balance equations are presented in Section 3.6). This modifies the form of the solution for $q_{i,j}$, but the method of analysis is the same. [GihKue85] is also of interest in that the transmission server in their model is a single queue representation of a local area network. The local area network was approximated by a single queue in a separate analysis. This is an example of a hierarchical decomposition.

Varghese, Chou, and Nilsson [VaChNi83] analyzed the same network as in [Reis81], but with zero acknowledgment delay. (The last queue in the tandem configuration can be viewed as the acknowledgment queue). These researchers found that the Norton's approximation technique did not work very well, particularly when the window size was small and the system load was high. An alternative means of

representing the transmission network as a single queue with state dependent service is presented, where the Norton's equivalent rate is used for $n_T > 0$, and a Coxian distribution is used when $n_T = 0$. The number of stages in the Coxian distribution is N, the number of hops in the transmission network. This technique was found to yield good results, but it is computationally intensive.

Schwartz [Schw82] also draws upon the Norton's approximation technique in his comparison of sliding window flow control with individual acknowledgments and IBM's SNA pacing control. The sliding window flow control model is the same as in [PenSch75]. Extensive simulation experiments are reported on. Among the most interesting results are that the Norton's approximation does not work well with bulk arrivals, and that the independence assumption substantially underestimates the throughput when the number of hops is larger than 3, particularly when the window size is small. The latter observation agrees with that of Varghese, et al, ([VaChNi83]).

Dallery [Dall87] presents a model of a sliding window flow controlled network with admission delays which is based on the analysis of a closed network model. The model in [Dall87] considers a transmission network with arbitrary topology, random routing, and a general service time distribution at each station. The independence assumption is used. There is no hop level flow control. A single connection is modelled. Packets arrive singly according to a Poisson process with mean λ_a and are acknowledged individually. Packets which arrive when there are no tokens available are enqueued in a holding queue.

Dallery analyzes a closed network consisting of the transmission stations from the open network plus one additional station to represent the sending station. The closed network has W tokens circulating. For a closed queueing network with general service time distributions such as this, there are no fast exact algorithms like MVA. Dallery uses a method due to Marie, [Mari79], which is an iterative numerical approximation scheme for analyzing closed networks with general service time distributions.

In Marie's method, each queue i in the network is analyzed in isolation, by whatever technique is appropriate, for the probability of having n_i customers in the queue, (denoted $p_i(n_i)$), assuming a state dependent arrival rate $\lambda_i(n_i)$. These state dependent arrival rates are the throughput of the network with queue i removed, referred to as the **complementary network** of queue i. Thus if there are W tokens in the system, $\lambda_i(n_i)$ is evaluated by having $W-n_i$ tokens in the complementary network of i. The result of the analysis for each queue i is a set of "conditional throughputs", $\nu_i(n_i)$, which satisfy the equations

$$\nu_i(n_i)p_i(n_i) = \lambda_i(n_i-1)p_i(n_i-1) \tag{2.4}$$

The values $\nu_i(n_i)$ are used as state dependent exponential service rates at queue i when analyzing the complementary network of some other queue $j \neq i$. Representing each queue, regardless of its actual service time distribution, as a queue with state dependent exponential service rate makes the analysis of the λ_i relatively easy.

Marie's procedure is iterative. We initially begin with rough estimates of the $\{\nu_i\}$ for each queue. As we analyze queue i we change these values, which will alter

the λ_j for other queues, and thus their conditional throughputs. This is turn changes the λ_i values, thus we need to re-compute $\{\nu_i\}$ again. The procedure continues until convergence is achieved.

Let us denote the queue representing the sending station as queue T. We shall return in a moment to the values $\nu_T(i)$ used by Dallery in evaluating the closed network, but first let us look at how the solution of the closed network is used to determine the behavior of the original open model.

After convergence of Marie's method, a set of values $\lambda_T(j)$, j=0,1, $\cdots$,W, have been determined. These values are the state dependent rates at which tokens return to the sending station when there are j tokens in the token queue. Dallery then uses equations (2.3) from [Reis81] to evaluate the sending station behavior, setting $\gamma(j)$ to $\lambda_T(W-j)$.

Now let us look at the values $\nu_T(i)$ used by Dallery. The conditional throughputs at the sending station queue should represent the delay tokens incur at the sending station. If there is more than one token at the sending station, then there can be no packets waiting, and thus the rate at which tokens will depart is the arrival rate of packets, λ_a. This is the same value of sending station service rate used in the source queue of [Reis79] and [PenSch75]. The analysis here differs in the conditional throughput assigned when n_T=1. In this case, we use (2.4) to write

$$\nu_T(1) = \lambda_T(0)\frac{p_T(0)}{p_T(1)}$$

$p_T(1)$ and $p_T(0)$ can be determined from the solution to (2.3), using the current esti-

mates of $\lambda_T(W-j)$ for $\gamma(j)$. $p_T(1) = q_{0,1}$ and $p_T(0) = \sum_{i=0}^{\infty} q_{i,0}$. The solution for $\nu_T(1)$ is

$$\left[\nu_T(1)\right]^{-1} = \frac{1}{\lambda_a} - \frac{1}{\lambda_T(0)} \tag{2.5}$$

(2.5) is expressed in terms of the inverse of the conditional throughput, which is the expected delay, to illustrate that the expected delay at the source queue when n_T=1 is less than the delay for an arrival to occur. Thus we see that the closed model to approximate token behavior in [Dall87] modifies the source queue behavior from that of [Reis79] and [PenSch75] to account for the occasional fast departure of a token from the sending station when it returns to find a packet waiting. In fact, (2.5) can be derived as the expected delay of a token at the sending station, conditioned upon it arriving to an empty token queue.

We note that the values of $p_T(0)$ and $p_T(1)$ used to determine $\nu_T(1)$ in (2.5) will change with each iteration of Marie's algorithm as the estimates of $\lambda_T(j) = \gamma(W-j)$ change.

Dallery reports on comparisons of this procedure with simulation experiments. The agreement is good. As with the other methods, the worst cases are when the window size is small and the number of stations is large. We have already pointed out that Marie's method is iterative, continually re-evaluating the conditional throughputs and arrival rates until convergence occurs, and can thus be very computationally intensive.

The paper by Fdida, Perros, and Wilk [FdPeWi87] is the only paper which we shall survey which explicitly treats nested layers of sliding window flow control. The paper presents an approximate solution methodology for a network with admission delay. A single end to end connection is modelled, with no packet fragmentation. Packets are acknowledged individually. Each service station in the network is modelled as a queue and server. The independence assumption is used. It is assumed that the servers are "BCMP-type", [BCMP75]. The significance of this last assumption will be discussed momentarily. Other than the restriction to BCMP-type servers, the network between the sender and receiver allowed under this model is quite general. Parallel levels of flow-controlled networks can be placed in tandem, and levels of flow control may be nested. The transmission and acknowledgment networks are assumed to be separate and independent, but the authors indicate how to remove this assumption.

The analysis of the system is hierarchical. At each step the lowest levels of flow controlled subnetworks are reduced to a single service station. These stations are then used in the analysis of the next lowest level of flow control.

The reduction of a single flow controlled network is performed as follows. Consider the flow controlled network of Figure 2.6. The arrival process to the holding queue is assumed to be Poisson with rate λ. The window size is W. Net1 and Net2 are arbitrary networks of BCMP-type stations; Net1 is the transmission network and Net2 is the acknowledgement network. Similar to [Reis81], the behavior of the flow controlled portion of the network is approximated by the behavior of the closed net-

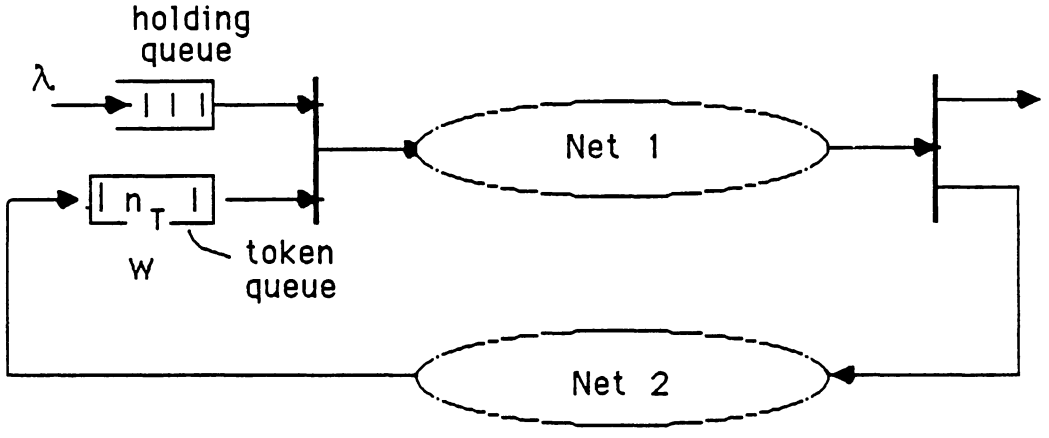


Figure 2.6 - Basic Flow Controlled Network

work formed by linking the outputs of Net1 and Net2 to the inputs of Net2 and Net1, respectively, (see Figure 2.7). Call this new network Net1/2. Since by assumption both Net1 and Net2 are composed of BCMP-type service stations, Net1/2 can be easily analyzed exactly by the MVA algorithm. We perform the analysis of the closed network for each possible network population, 1,2,...,W, and for each population c we calculate two values, $R'(c)$ and $R''(c)$. $R'(c)$ is the mean time to traverse Net1 when there are c packets in Net1/2, and $R''(c)$ is the mean cycle time (time to traverse Net1 and Net2) of Net1/2 when there are c packets in Net1/2. Define $\gamma(c)$ as $c/R''(c)$. $\gamma(c)$ is the rate at which customers leave Net2 when there are c packets in Net1/2, i.e. $n_T=W-c$. The original flow-controlled network of Figure 2.6 is now approximated by an infinite server queue with state dependent service rate $R(c)$ given by

$$R(c)=\begin{cases} R'(c), & c\le W \\ R'(W)+(c-W)/\gamma(W), & c>W \end{cases}$$

$R(c)$ is an approximation to the mean time for a packet to traverse the original flow-controlled network when there are c packets in the system, (holding queue plus transmission network). For $c\le W$ this time is approximated as $R'(c)$. For $c>W$ a packet must wait for $(c-W)$ tokens to return to the sending station before it allowed into the transmission network. The approximation to this delay is $(c-W)/\gamma(W)$, where $1/\gamma(W)$ is the mean time for one token to return when there are W tokens in use. Once access to the transmission network is achieved the estimate of time to traverse Net1 is $R'(W)$.

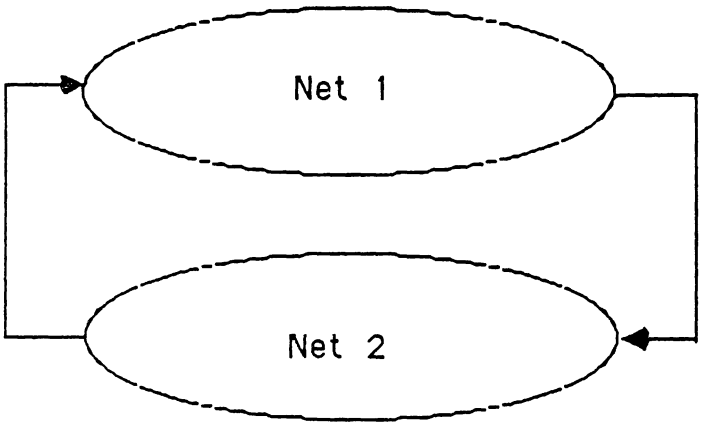


Figure 2.7 - Closed Network (Net 1/2)

Now let us examine how this technique is applied to nested levels of sliding window flow control. Consider the network of Figure 2.8, where S_2 and S_5 are sliding window flow controlled sub-networks, and S_1, S_3, S_4, and S_6 are general networks of BCMP type servers. Using the approach described above, S_2 and S_5 are replaced by single stations, each with a state dependent infinite server. Since such stations are BCMP stations, the transmission and acknowledgment paths of the approximate network are both networks of BCMP stations, and the reduction technique is now applied to the new network.

Once we have hierarchically reduced all of the lower levels of flow control, the values $\gamma(j)$ are computed for the highest level model, and equations (2.3) are used to determine $q_{i,j}$.

Comparisons of this method against simulation results presented by the authors show that the method performs well. The accuracy declines as the utilization of the tokens increases.

The final paper we shall look at analyzes a sliding window flow controlled link in the context of HDLC, a common data-link layer protocol. The basic model analyzed by Labetoulle and Pujolle [LabPuj79] is shown in Figure 2.9. Packets arrive for transmission singly according to a Poisson process with rate λ. Packets in the transmission queue are transmitted as long as there are fewer than W packets in the acknowledgment queue. Once the number of packets in the acknowledgment queue reaches W, transmission is halted until the number in the acknowledgment queue decreases below W again. Transmission time is exponentially distributed with

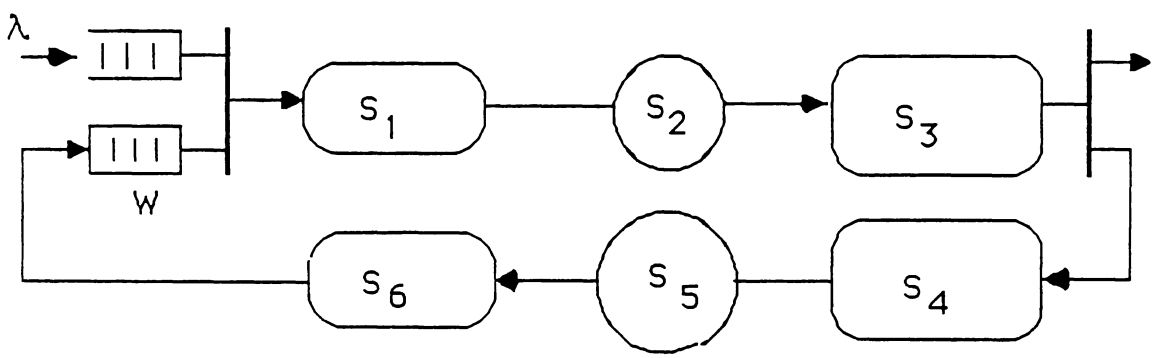


Figure 2.8 - Multiple Flow Controls

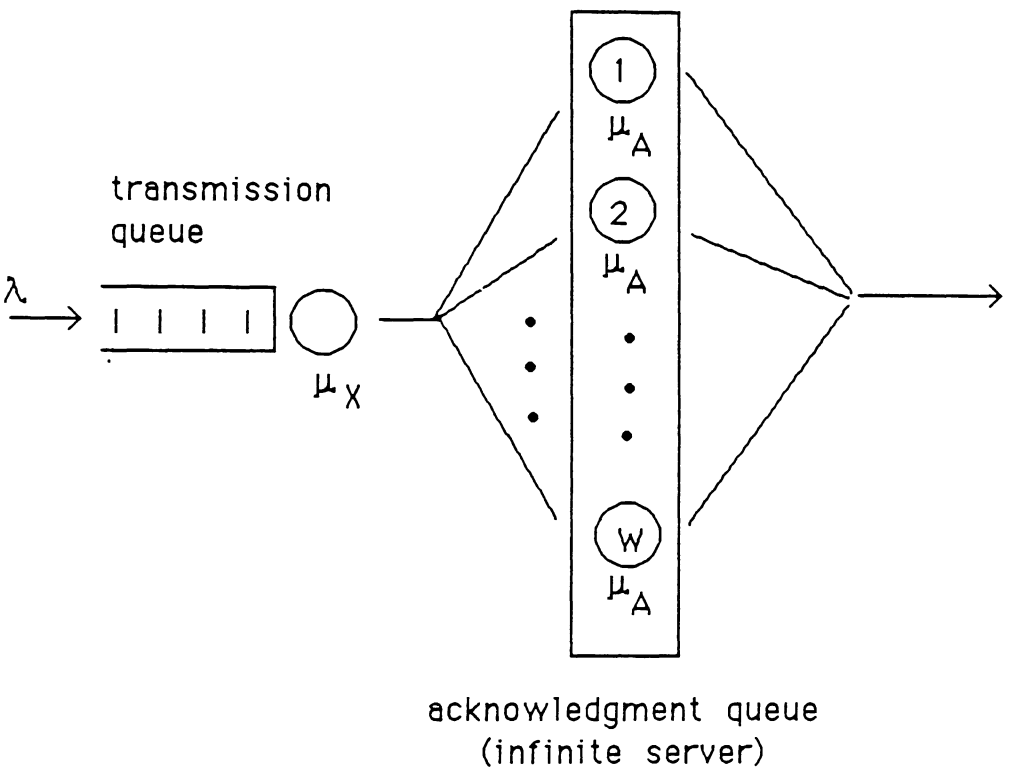


Figure 2.9 - Model of Labetoulle and Pujolle

rate μ_X. After transmission, packets go into an infinite server queue representing the acknowledgment mechanism. Packets leave this queue at the rate $i\,\mu_A$ when there are i packets in it. This state dependent service rate approximates the behavior of HDLC, where acknowledgment generally contain multiple tokens. This queueing model is solved approximately. The probability of having the acknowledgment queue full, (W packets), is estimated, and this is used to develop a service model of the transmission queue. The model of the transmission queue is evaluated for mean delays and queue lengths. No distribution results are obtained. Comparisons to simulation show the approximation to be quite good.

This method should prove useful as a sub-model in approximations which incorporate mean delay of data-link protocol in service time at a queue. Since the final service model of the transmission queue is rather complicated, this method may not be well suited for nested analysis.

We conclude this section with some remarks about related research areas which are not directly suitable for modelling sliding window flow control with admission delays.

One such research area is that of multi-chain queueing networks with chain population constraints. These models have been analyzed approximately by several authors. A review of these approximations can be found in Thomassian and Bay [ThoBay84b]. These models were developed for multi-programming systems, and either deal with closed networks, or require the loss model assumption.

Open queueing networks with population constraints have also been considered. For two node queueing networks see [Perr83] and the references within. Lam [Lam77] considers multi-chain queueing networks with state dependent lost and triggered arrivals. This paper extends the class of queueing networks which are known to have a product form solution. Goto, Takehashi, and Hasegawa [GoTaHa83] analyze an open tandem configuration with finite buffers and an overall population constraint. All of these models ignore acknowledgment delays and use the loss model assumption.

Two other related areas are the models of simultaneous resource possession and serialization delays. These models either use closed network analysis, or are specifically tailored to multiprogramming systems. For papers dealing with simulatneous resource possession see [Perr81], [JacLaz82], and [FreBex83]. Serialization delays are dealt with in [AgrBuz83], [Thom83], and [JacLaz83].

CHAPTER 3

THE LOWEST LEVEL MODEL

In this Chapter we present a model of a one hop single connection with sliding window flow control, and an algorithm for approximating its throughput and delay characteristics.

3.1. The Model

The model under consideration is shown in Figure 3.1. Packets for transmission arrive in groups of B to an infinite capacity holding queue at the sending station. The arrival process is assumed to be Poisson with rate λ. If a token is available at the token queue, then that token is paired with the first waiting packet in the holding queue and the token/packet pair is delivered to the transmission queue. This pairing of tokens and packets and subsequent delivery to the transmission queue is assumed to take place instantaneously, and continues until either the token queue or the holding queue becomes empty.

The packet/token pairs at the transmission queue are served one at a time in a first-come first-served fashion. Service times at the transmission queue are exponentially distributed with a mean service time of $1/\mu_X$. Upon completion of service at the transmission queue, the packet leaves the system and the token is delivered instantaneously to the acknowledgment queue. Tokens at this queue are served one at a time, first-come-first-served, with an exponentially distributed service time of mean $1/\mu_A$. Upon completion of service at the acknowledgment queue, the token is instantaneously delivered to the token queue, where it is again available for pairing

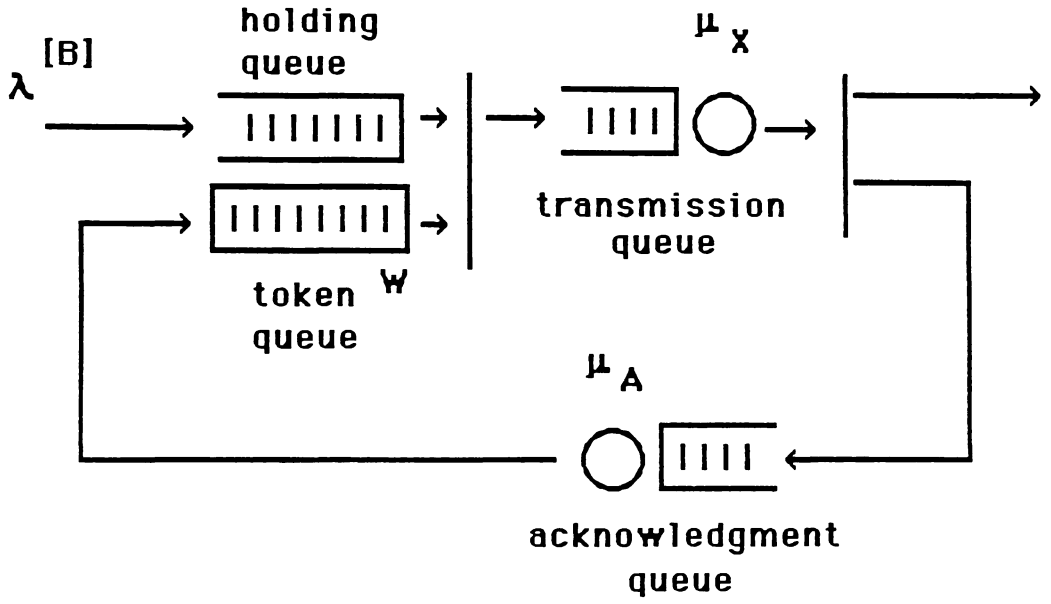


Figure 3.1 - The Model

with a packet. There are a total of W tokens circulating in this model, W being the window size of the flow control.

Let us discuss the features of this model as they relate to actual data communication systems. We assume that the arrival process is Poisson. Limited empirical evidence suggests that this may indeed be the case in computer communication environments, (see [Schw77]), as it is in telephony. In any event, queueing analysis is consideralbly more difficult without this assumption, and it is thus a common one in communications modelling.

The holding queue represents the packets the higher layer has ready once the modelled layer has a token available. The batch arrivals correspond to fragmentation of the higher layer data unit.

We model the packet delivery mechanism as a single server queue with an exponential service time distribution. In what follows we shall refer to the service time in this queue as the transmission time, although there are other components to the delivery delay than just transmission. For instance, if there are other connections using the transmission channel, as will generally be the case, packets in the connection under study will suffer queueing delays while other connection's packets are being transmitted, in addition to the queuing delays among packets of the connection under study. We do not explicitly model other connections sharing the transmission facility, but assume that the exponential service time in the transmission queue includes the queueing delays due to these other packets. There will also be delays in successful delivery due to the need to re-transmit lost or damaged packets. We do

not model these delays here, but only assume that each packet is delivered correctly, and that the time to delivery includes any required re-transmissions.

Our model also assumes that packets are delivered in the order in which they are transmitted. This assumption is true in the case of virtual-circuit protocols, or protocols which use a "go-back-n" scheme for re-transmissions. Although this assumption may be violated in some systems, throughput and buffer requirements will not be affected (as long as the time between packet deliveries regardless of order is exponentially distributed) since packets are assumed to be statistically identical; however, violation of this assumption will affect our delay analysis, in which we assume that each packet must wait for delivery of each packet which arrived to the system before it.

We do not explicitly model the time required for such protocol functions as checksum computation, adding headers and trailers, and generating acknowledgments. It can be assumed that the time for these functions is included in the delivery time, but our model does not allow for these tasks to occur in parallel with the transmission of another packet as is the actual case. The choice of whether to ignore these protocol times or include them in the transmission time depends upon the utilization of the system. If the system has a fairly high utilization, much of the protocol functions will be done while other packets are being transmitted. If packets typically need to queue for transmission after protocol overhead functions are completed, then we may as well ignore the time these functions take, since they are not the determining factor in delivery delay. On the other hand, if packets generally find an idle

server after protocol functions, as will be the case when the system is lightly loaded, the inclusion of the protocol times in the delivery delay is appropriate. In any event, if the time for protocol functions is negligible with respect to the transmission time, either assumption will not severly affect the analysis.

To conclude, the assumption that the time between packet deliveries is exponentially distributed is made without modelling many important features of actual data communication systems. The decision not to explicitly model these features was made to keep the model of focus as simple as possible. This decision is based on two reasons. First, we wish to examine and evaluate the effects of sliding window flow control protocols. The model to be considered, while idealized, can still give insight into the effects of sliding window flow control. Secondly, we wish to examine a model with as few analytical complications as possible at this point. Should we find a successful approach for the simple model, it can point to means which may be successful in more complicated models, as we will see in Chapter 4.

We also assume that the acknowledgments have an exponentially distributed delivery time, but with a different mean than the data packets. The ratio of these means, μ_X/μ_A, is a measure of the relative delivery times of the data packets and acknowledgments. If the ratio is much less than 1.0 this indicates that acknowledgments return to the sender much more rapidly than data travels to the receiver. This will be the case when data packets are large relative to the acknowledgments. When the ratio is close to or greater than 1.0 this implies that either the data packets and acknowledgments are roughly the same size, or that the receiver is slow and thus

delays releasing acknowledgments. We do not model the receiver release delay and acknowledgment transmission delay separately, but lump them together in the acknowledgment server.

It is assumed that data packets are acknowledged one at a time. The situation in actual networks is more complicated; packets can be acknowledged in groups or separately, even in the same protocol.

In our model the data packets and acknowledgments do not interfere with each other's transmission, despite sharing the same transmission network. In the case where acknowledgments have priority, it appears to the acknowledgments that there is no interference from the data. In this case, the interference of the acknowledgments on the data transmission can be roughly accounted for by decreasing the data transmission service rate proportionally to the amount of time required to transmit acknowledgments. In networks where the acknowledgments do not have priority, we decrease both the acknowledgment service rate and the data transmission rate to account for the time spent transmitting the other type of information. (See the discussion of effective service rates in the review of [PenSch75] in Chapter 2 for an example of this technique).

In summary, our model is an idealized version of an actual sliding window flow controlled connection, but retains the essential feature of such a connection, to wit, transmission of data is suspended when all the tokens are in use. While the model as presented may be assailed for its simplifications, we note in partial defense that the simplifications are no more severe than in other models, (see Chapter 2).

3.2. Maximum Throughput of the Model (Stability Condition)

One parameter of interest in looking at any queueing model is its maximum throughput. This gives us an upper bound on the input traffic rate which the system can handle. The bound is stated as an inequality, arrival rate < maximum throughput. Any arrival rate above the maximum throughput bound will result in an unstable system, so we wish to keep the input traffic below this bound.

The maximum throughput can be evaluated by examining the system under saturation. We posit an infinite queue of customers awaiting service by the system, and observe the rate at which customers leave the system under this condition. This rate is the maximum throughput.

This means of evaluating the maximum throughput is particularly easy for the model of Section 3.1. With an infinite queue of packets waiting for transmission the token queue becomes irrelevant, as each token returning to the sending station will always find a packet waiting, and will thus immediately be used. The queueing model for this situation is shown in Figure 3.2. The output of the acknowledgment queue is connected to the input of the transmission queue, forming a closed network of two exponential server queues with W tokens circulating.

Let us define

$$\pi_i(j) = \Pr\{\ j \text{ tokens in ack. queue} \mid i \text{ tokens in model of Figure 3.2 }\}. \quad (3.2.1)$$

The throughput of the model in Figure 3.2, (and thus the maximum throughput of the model in Figure 3.1), is $\mu_X(1-\pi_W(W))$. Whenever there are less than W tokens in

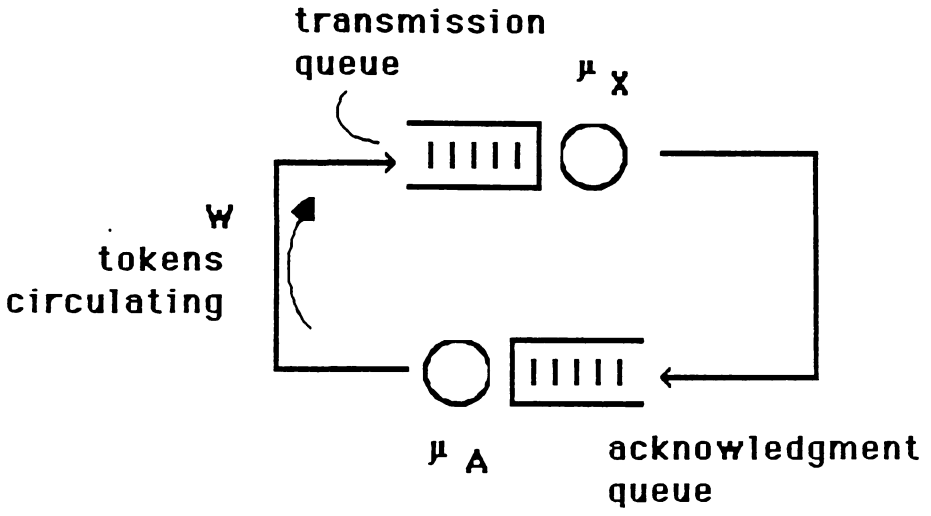


Figure 3.2 - Model of the System Under Saturation

the acknowledgment queue the transmission server is delivering packets at the rate μ_X. When all W tokens are in the acknowledgment queue the transmission server is idle, and thus has an effective service rate of 0. The closed system throughput is then

$$\mu_X(1-\pi_W(W)) + 0\pi_W(W) = \mu_X(1-\pi_W(W)).$$

All that remains to find the maximum throughput is to calculate $\pi_W(W)$. The local steady state probability balance equations for $\pi_W(i)$ in the model of Figure 3.2 are

$$\mu_X \pi_W(i) = \mu_A \pi_W(i+1), \quad i=0,1,2,...,W-1$$

and a unique solution is obtained by using the normalizing sum

$$\sum_{i=0}^{W} \pi_W(i) = 1.$$

The solution to the above system of equations is

$$\begin{cases} \pi_W(i) = \pi_W(0)\left(\mu_X/\mu_A\right)^i, & i=1,2,...,W \\ \pi_W(0) = \dfrac{1-\left(\mu_X/\mu_A\right)}{1-\left(\mu_X/\mu_A\right)^{W+1}} \end{cases} \tag{3.2.2}$$

Thus

$$1-\pi_W(W) = \frac{1-\left(\mu_X/\mu_A\right)^W}{1-\left(\mu_X/\mu_A\right)^{W+1}}$$

and the stability condition is

$$\lambda B < \mu_X \frac{1-\left(\mu_X/\mu_A\right)^W}{1-\left(\mu_X/\mu_A\right)^{W+1}}. \tag{3.2.3}$$

3.3. Approximating the Throughput Characteristics of the Connection

As we mentioned in Chapter 1, our goal is to reduce the model of Figure 3.1 to a single server queue. The characteristics of the server should be as simple as possible, so as to reduce the computational burden when using the approximate queue in place of the sliding window flow controlled network in a hierarchical analysis.

We begin by examining the behavior of the model. The probabilistic behavior of the model described in Section 3.1, and depicted in Figure 3.1, can be completely described using a two-dimensional state space. We shall define the state to be the pair (n_S, n_A), where n_S is the number of packets in the system, (combined number of packets in the holding and transmission queues), and n_A is the number of tokens in the acknowledgment queue. (We shall use the notation (n_S, n_A) to refer to a general state. When we are discussing particular values of n_S and n_A, we will use the notation (i, j) to describe the state).

From the state description (n_S, n_A) we can calculate the number of tokens in the token queue and the number of packets in each of the holding and transmission queues. Let n_H denote the number of packets in the holding queue, n_T the number of tokens in the token queue, and n_X the number of token/packet pairs in the transmission queue. $(n_S = n_H + n_X)$. Recall that W is the total number of tokens in the system, and does not vary with time. We now prove

Theorem 1

Given the state (n_S, n_A). Let $k = W - n_A$. Then

(1) $n_X=\min(n_S,k)$,

(2) $n_H=n_S-n_X$,

(3) $n_T=k-n_X$,.

Proof:

(2) is the definition of n_S. (3) follows from the identity $n_A + n_T + n_X = W$.

To prove (1), note that k is the number of tokens available for packets, and n_S is the number of packets to which tokens can be assigned. Since the operation of the system is such that tokens are assigned to packets and delivered to the transmission queue as long as both tokens and packets are available, it follows that $n_X=\min(n_S,k)$, i.e., either all tokens are used ($n_X=k$) or the holding queue is emptied ($n_X=n_S$).

QED

Now let us examine the equations describing the steady state behavior of the model. Let p_{ij} denote the steady state probability of finding the system in the state (i,j). The steady state balance equations for this model are:

$$i=0 \begin{cases} \lambda p_{0,0} = \mu_A p_{0,1} \\ (\lambda+\mu_A)p_{0,j} = \mu_A p_{0,j+1} + \mu_X p_{1,j-1}, \ j = 1,2,\ldots,W-1 \\ (\lambda+\mu_A)p_{0,W} = \mu_X p_{1,W-1} \end{cases} \tag{3.3.1}$$

$$i=1,2,\ldots,B-1 \begin{cases} (\lambda+\mu_X)p_{i,0} = \mu_A p_{i,1} \\ (\lambda+\mu_A+\mu_X)p_{i,j} = \mu_A p_{i,j+1} + \mu_X p_{i+1,j-1}, \ j=1,2,\ldots,W-1 \\ (\lambda+\mu_A)p_{i,W} = \mu_X p_{i+1,W-1} \end{cases}$$

$$i \geq B \begin{cases} (\lambda+\mu_X)p_{i,0} = \lambda p_{i-B,0} + \mu_A p_{i,1} \\ (\lambda+\mu_A+\mu_X)p_{i,j} = \lambda p_{i-B,j} + \mu_A p_{i,j+1} + \mu_X p_{i+1,j-1}, \quad j=1,2,\ldots,W-1. \\ (\lambda+\mu_A)p_{i,W} = \lambda p_{i-B,W} + \mu_X p_{i+1,W-1} \end{cases}$$

As it is our desire to develop a single queue approximation for the data packet transmission process, which does not explicitly include the acknowledgment/token scheme, let us look at the probability balance equations governing the marginal probability distribution of n_S. Define

$$P_i = \Pr\{n_S=i\} = \sum_{j=0}^{W} p_{i,j}.$$

We can obtain the probability balance equations for P_i by summing each equation in (3.3.1) in which i appears on the left-hand side. This operation results in the equations

$$\begin{cases} \lambda P_0 = \mu_X (P_1 - p_{1,W}) \\ \lambda P_i + \mu_X (P_i - p_{i,W}) = \mu_X (P_{i+1} - p_{i+1,W}), \quad i=1,2,\ldots,B-1. \\ \lambda P_i + \mu_X (P_i - p_{i,W}) = \lambda P_{i-B} + \mu_X (P_{i+1} - p_{i+1,W}), \quad i \geq B \end{cases} \tag{3.3.2}$$

Define

$$b_i = \Pr\{ n_A=W \mid n_S=i\} = \frac{p_{i,W}}{P_i}.$$

b_i is the conditional probability of having all tokens in the acknowledgment queue when there are i packets in the system. We can re-write (3.3.2) as

$$\begin{cases} \lambda P_0 = \mu_X (1-b_1) P_1 \\ \lambda P_i + \mu_X (1-b_i) P_i = \mu_X (1-b_{i+1}) P_{i+1}, \quad i=1,2,\ldots,B-1. \\ \lambda P_i + \mu_X (1-b_i) P_i = \lambda P_{i-B} + \mu_X (1-b_{i+1}) P_{i+1}, \quad i \geq B \end{cases} \tag{3.3.3}$$

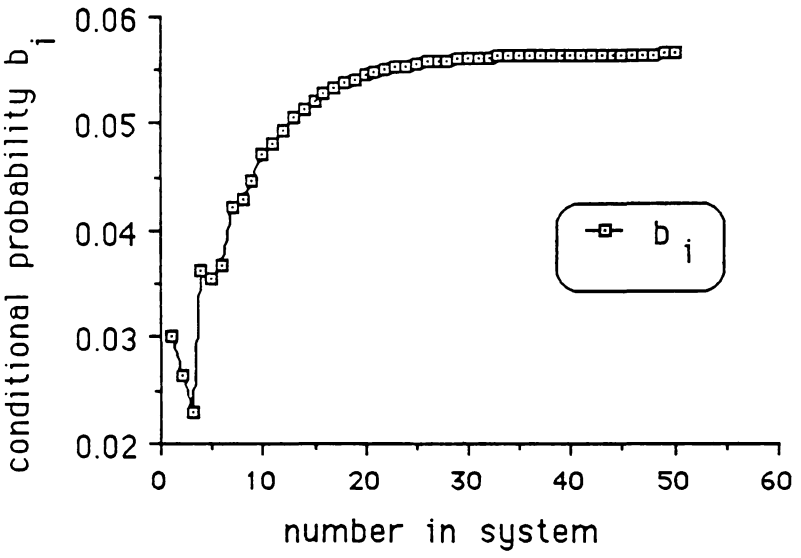
These are the global balance equations for a single server queue with batched Poisson arrivals of size B, and state dependent exponential service rate $\mu_X(1-b_i)$.

At this point one could construct an exact equivalent single server queue for the data packet transmission process by using the state dependent service rates $\mu_X(1-b_i)$. The b_i can be calculated from the solution to (3.3.1); however, there is no known closed form solution to these equations. Due to the regular structure of the rate equations (3.3.1) for $n_S \geq B$, numerical solution of the equations is rather straightforward using the matrix geometric procedure, ([Neut81]). This procedure was implemented for these equations, and the exact solutions are used to evaluate the approximation algorithms. Unfortunately, exact numerical solution is not practical for all instances of the model, as it is an iterative procedure which requires $O(B^2W^3)$ multiplications per iteration. Furthermore, the $\{b_i\}$ represent an infinite family of parameters, and the time required by the numerical procedure to compute b_0 through b_N is proportional to N. Since numerical solution for the b_i is not practical in all instances, we resort to approximation.

Rather than attempting to estimate each b_i, and construct a state dependent server, we take another approach. We will approximate the transmission service process by a single server queue with exponential service at a rate which is in some sense an average of the state dependent service rates $\mu_X(1-b_i)$. The assumption that the effective service from the sliding window flow controlled network is exponentially distributed makes the approximation queue very simple, but it is also incorrect given the model assumptions. The transmission service itself has an exponential distribution. Each time the acknowledgment queue becomes full, (the probability of which is state dependent), the next service completion will occur after two exponen-

tially distributed delays; the first of mean $1/\mu_A$ for return of a token to the sending station, and the second of mean $1/\mu_X$ for actual delivery. We see that the actual delivery time distribution under sliding window flow control is both state dependent and non-exponential. We suspect, however, that if the acknowledgment queue does not become full too often then the effective delivery time distribution may not be too far from the service time distribution at the transmission queue, i.e. exponential, in which case the approximation should prove satisfactory. In our experiments we shall specifically examine the effects of the probability that the acknowledgment queue is full on the accuracy of the approximation.

It remains for us to determine the service rate of our approximate queue. We will denote the approximate service rate by $\bar{\mu}_X = \mu_X(1-\bar{b})$, in conformance with the structure of the state dependent rates. We wish to choose $\bar{b}$, (and thus $\bar{\mu}_X$), such that the approximate queue has roughly the same behavior as the actual system; $\bar{b}$ should be an "average" of the b_i. Before we proceed any further, let us examine two typical examples of the state dependent behavior of b_i. These examples are taken from the exact solution to the model of Figure 3.1 computed by Neuts' matrix-geometric procedure, and are shown in Figures 3.3 and 3.4. We see that the value of b_i is lowest for $i=B$. The situation where $i=B$ can occur when a batch arrival of size B occurs to an empty system. At this point, since the holding queue has been empty, the sending station will have had a chance to "stockpile" some tokens, and thus the probability of having all tokens in the acknowledgment queue will be relatively low. In Figure 3.4 $W=3$, and we note that b_i is small for $i = B, B-1$, and $B-2$. For $i = B-3$, b_i is much



Data Set Parameters

$\lambda = 2.5$

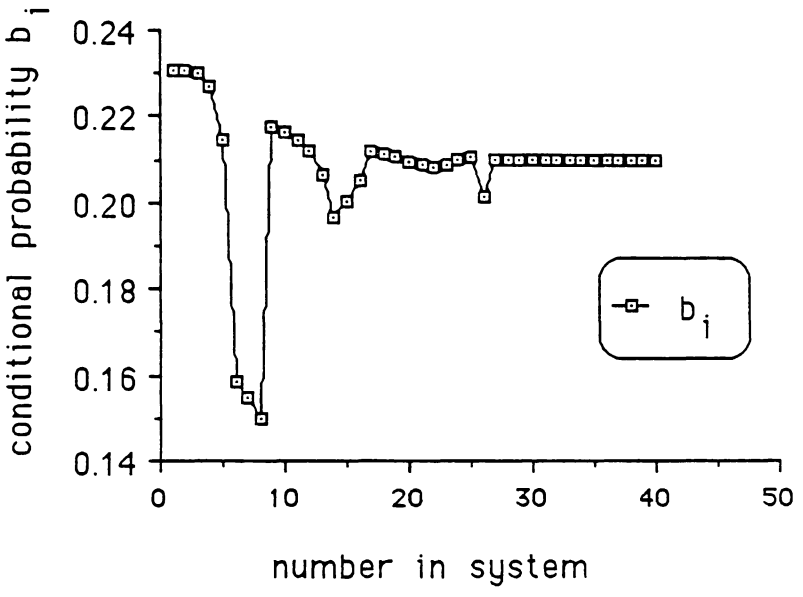
$\mu_X = 10$

$\mu_A = 11$

$W = 8$

$B = 3$

Figure 3.3 - Plot of b_i for $B < W$



Data Set Parameters

$\lambda = 0.85$

$\mu_X = 9$

$\mu_A = 10$

$W = 3$

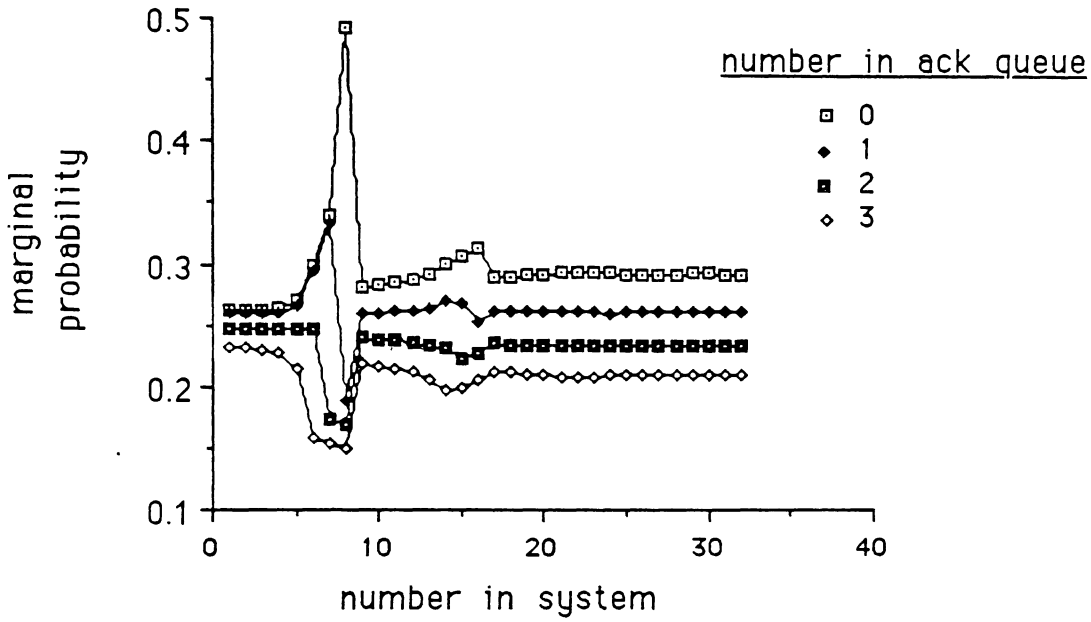
$B = 8$

Figure 3.4 - Plot of b_i for $B > W$

larger, as at this point all tokens are likely to be in use by the first three packets in the batch. In Figure 3.4 we also see small dips in b_i for i close to multiples of $2B$ and $3B$, due to multiple arrivals to an empty system.

We also observe that b_i becomes nearly constant as i gets large. This behavior is exhibited by every choice of parameters we have examined. Figure 3.3 is typical of the cases where $B < W$. The b_i values beyond $i = 2B$ increase with i, tending to the constant value. Figure 3.4 is an example of the situation when $B > W$. The values of b_i fluctuate for small values of i, the lowest value occurring at $i = B$. The fluctuations diminish as i increases, and b_i eventually tend to a constant value.

Let us refer to the settling down of b_i to a constant as i increases as a **state-asymptotic** property. (This is to distinguish from time asymptotic behavior). That all systems which we examined show this behavior is not surprising. The state transition structure of our model is aperiodic. When there are a large number of packets in the system, the system will behave much like the closed model of Figure 3.2 considered earlier, in that each token returning to the sending station will find a packet waiting. As the holding queue is being depleted, the tokens will be continuously moving through the system and we would expect that the distribution of the tokens between the acknowledgment and transmission queues will reach a statistical equilibrium. This expectation is confirmed by the empirical evidence from exact solutions. Figure 3.5 illustrates the exact marginal probability of each possible acknowledgment queue population for the data set of Figure 3.4. Results similar to this are observed for every example which we considered.



Data Set Parameters

$\lambda = 0.85$

$\mu_X = 9$

$\mu_A = 10$

$W = 3$

$B = 8$

Figure 3.5 - Marginal probability of ack. queue population

Let us denote the state-asymptotic value of b_i by b_∞.

We also note from empirical observation that $b_B < b_\infty$. This is also to be expected. As we discussed previously, b_B will be small due to the events of a batch arriving to an empty system which has had time to "stockpile" tokens. b_∞ is observed when the system is under congestion, and there is no opportunity for the sender to build up a supply of tokens. Since the probability of $n_S = B$ is higher than other probabilities in most systems, (see for Example Figures 3.6 and 3.7 which give the probability distribution for the examples in Figures 3.3 and 3.4, respectively), we expect that the "average" value $\bar{b}$ will be between b_B and b_∞.

Let us first attempt to estimate b_∞. Doing that will give us an upper bound on $\bar{b}$. Since under congestion the system behaves like the closed model of Figure 3.2, it seems that a good estimate of b_∞ would be $\pi_W(W)$. (See (3.2.1) and (3.2.2)). Surprisingly, this approximation is not very good. $\pi_W(W)$ always overestimates b_∞. Table 3.1 gives some sample results, which are typical. The data set numbers in this table refer to the data sets solved exactly in our experiments. The parameters for these data sets are in Appendix A, Table A.1. The values of b_∞ in Table 3.1 were obtained by examining the exact b_i values from the sampled systems, and choosing as b_∞ the first value b_i for which $\dfrac{|b_i - b_{i+k}|}{b_i} < 0.001$ for $k = 1,2,\cdots,20$.

Why does $\pi_W(W)$ overestimate b_∞ ? This can be explained by examining the rate diagram for the closed system of Figure 3.2 used to calculate $\pi_W(W)$ shown in Figure 3.8, and the rate diagram for the actual system, a portion of which is also

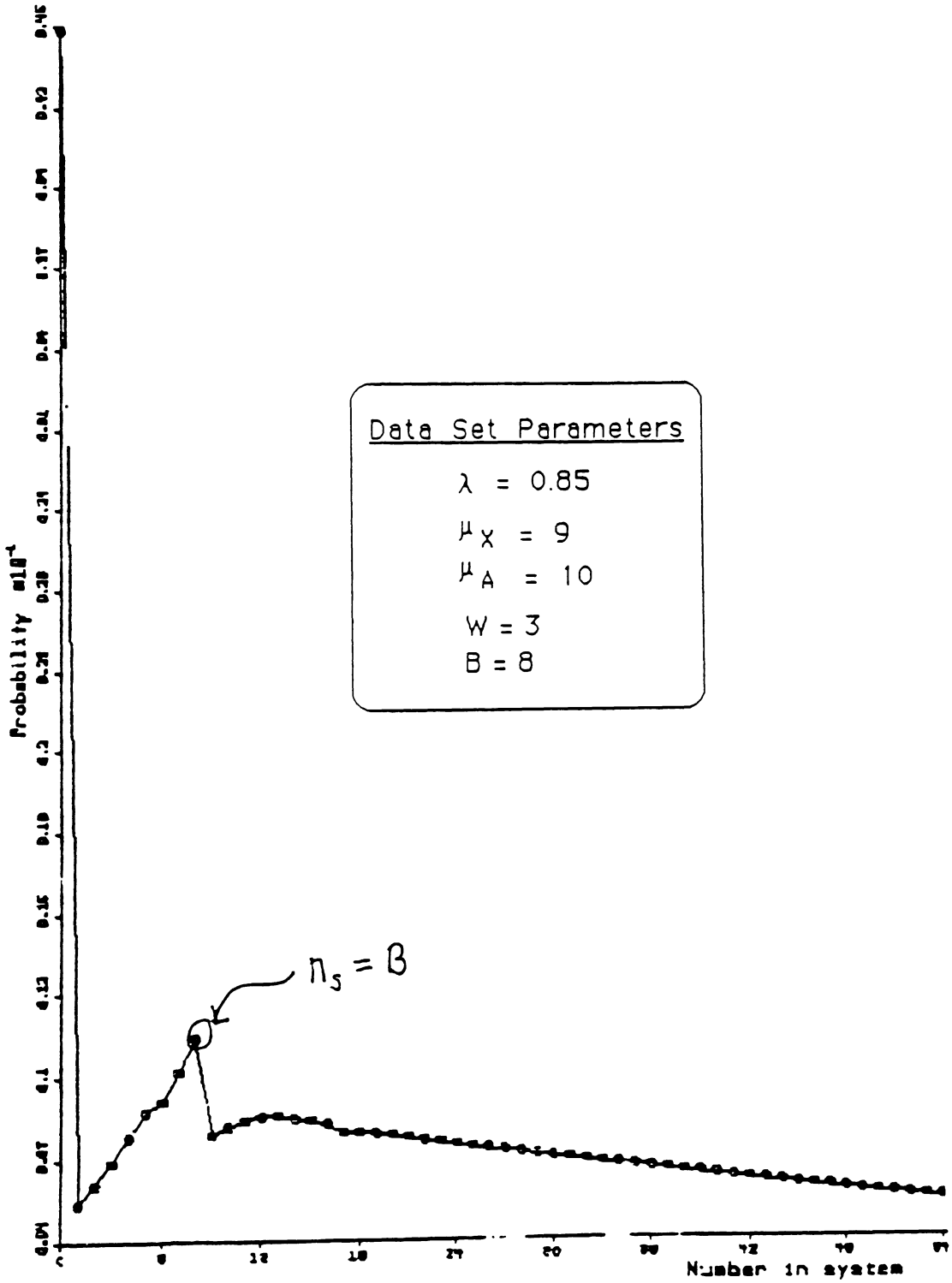


Figure 3.6 - Exact Probability of number in system for example where B > W

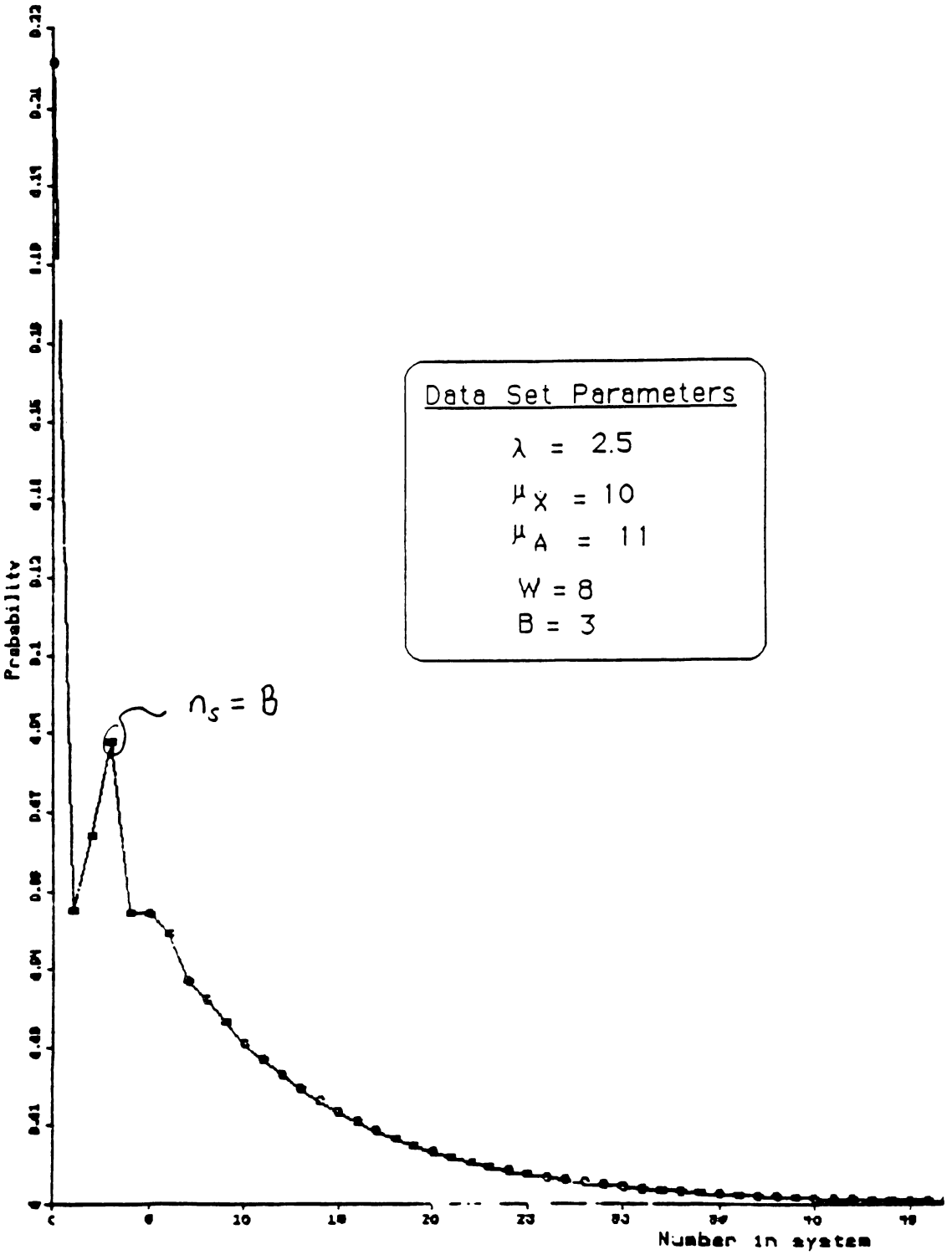
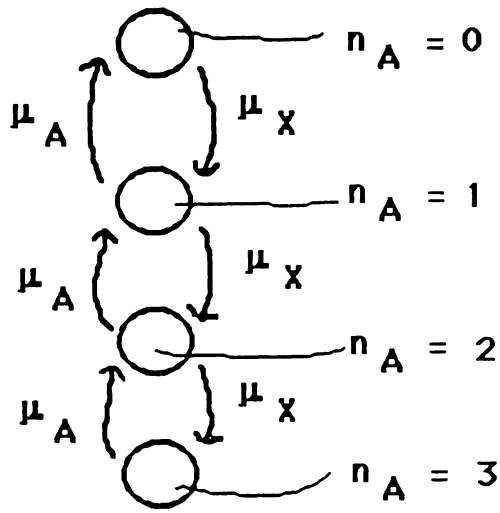


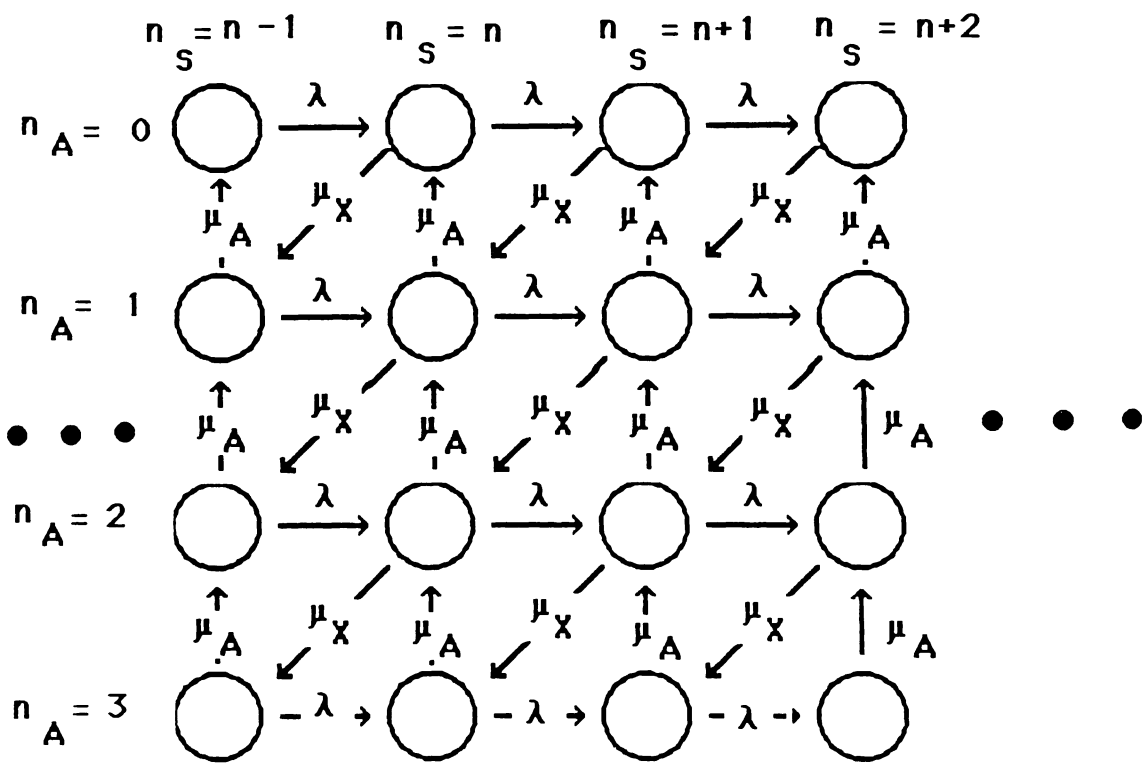
Figure 3.7 - Exact Probability of number in system for example where B < W

Table 3.1 - Observed b_∞ vs. $\pi_W(W)$

Data Set	$\pi_W(W)$	b_∞	% error
2	5.76 e-03	6.22 e-04	826%
8	2.58 e-07	3.57 e-08	623%
10	9.00 e-05	6.27 e-06	1335%
13	0.122	0.107	14%
14	0.196	0.178	10%
15	9.00 e-05	3.00 e-05	200%
16	9.00 e-05	6.04 e-05	49%
18	9.00 e-05	5.45 e-09	1.65 e06%
20	0.0155	0.0017	812%
21	0.0155	0.0109	42%
22	0.0323	0.0192	68%



Closed model rate diagram (W=3)



Portion of actual rate diagram (W=3, B=1)

Figure 3.8 - Comparison of rate diagrams

shown in Figure 3.8. Note that in the actual system each completion at the transmission queue takes us from a state where $n_S=n+1$ to a state where $n_S=n$. In the closed system there is no explicit value of n_S. We assume that n_S is "large", and that we increase the number in the acknowledgment queue at the rate at which transmission completions occur, and decrease it by the rate at which acknowledgment completions occur. Thus the closed model estimates for large n_S that the flux into state $(n,j+1)$ due to a transmission completion is $\mu_X p_{n,j}$, the rate at which transmission completions occur weighted by the probability of having j in the acknowledgment queue with large n_S. In the actual system, the flux into state $(n,j+1)$ due to transmission completions is $\mu_X p_{n+1,j}$. If $p_{n,j}$ and $p_{n+1,j}$ were equal, then the closed model would accurately predict the probability distribution of the acknowledgment queue for large n_S; however, in general these probabilities are not equal. While we have already seen that for large n_S the marginal probabilities of having j in the acknowledgment queue are nearly equal, i.e.,

$$\frac{p_{n,j}}{P_n} \approx \frac{p_{n+1,j}}{P_{n+1}}, \tag{3.3.4}$$

we should not expect that $p_{n,j} = p_{n+1,j}$.

In a stable queueing system we must have that $\Pr\{n_S=n\}>\Pr\{n_S=n+1\}$ for infinitely many values of n, (except for the special cases where an infinite number of $P_n = 0$). We shall assume that this inequality holds in the system under study for *all* values of n sufficiently large, and that furthermore these probabilities are in a constant ratio, i.e.,

$$\frac{P_{n+1}}{P_n} = \gamma_\infty \tag{3.3.5}$$

for some $\gamma_\infty \in (0,1)$ and all n sufficiently large.

Let us look at this assumption. (3.3.5) is true for an M/M/1 queueing model for all $n \geq 0$. It is also approximately true for the $M^{[B]}/M/1$ queue (which is how we are approximating the model of Figure 3.1), for large values of n. For the latter queueing model, the probability of having n in the system is a weighted sum of geometric distributions, ([Klei75]). This weighted sum can be expressed as

$$P_n = \sum_{i=1}^{B} c_i z_i^{\,n} \tag{3.3.6}$$

where z_i are the poles of the z-transform of $\{P_n\}$, and c_i are constants. For large values of n the largest z_i dominates the other terms in the sum (3.3.6), and thus the ratio of the probability of having $n+1$ in the system to the probability of having n is roughly $\max_i z_i$. While we have no proof that (3.3.5) also holds in the sliding window flow-controlled model, there is empirical evidence that it does. Figures 3.9 and 3.10 show plots of the ratios $\frac{P_{n+1}}{P_n}$ vs. n_S for the data sets of Figures 3.6 and 3.7. We see that the ratios do become nearly constant as n_S gets large. This behavior was observed in every example which we examined.

Given that (3.3.5) holds, we can show why $\pi_W(W)$ overestimates b_∞. From (3.3.4) we can write

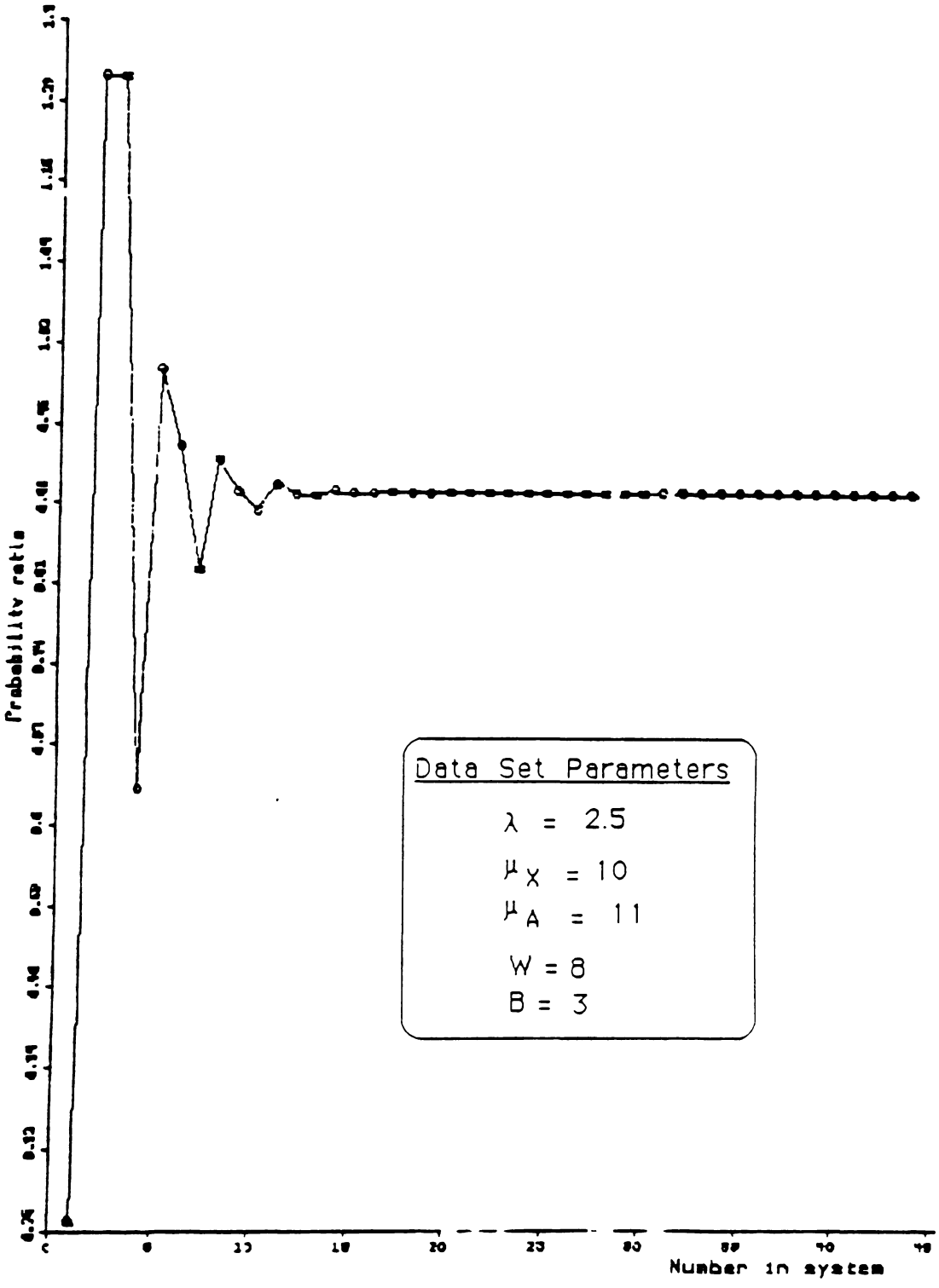


Figure 3.9 - Probability Ratios $B < W$

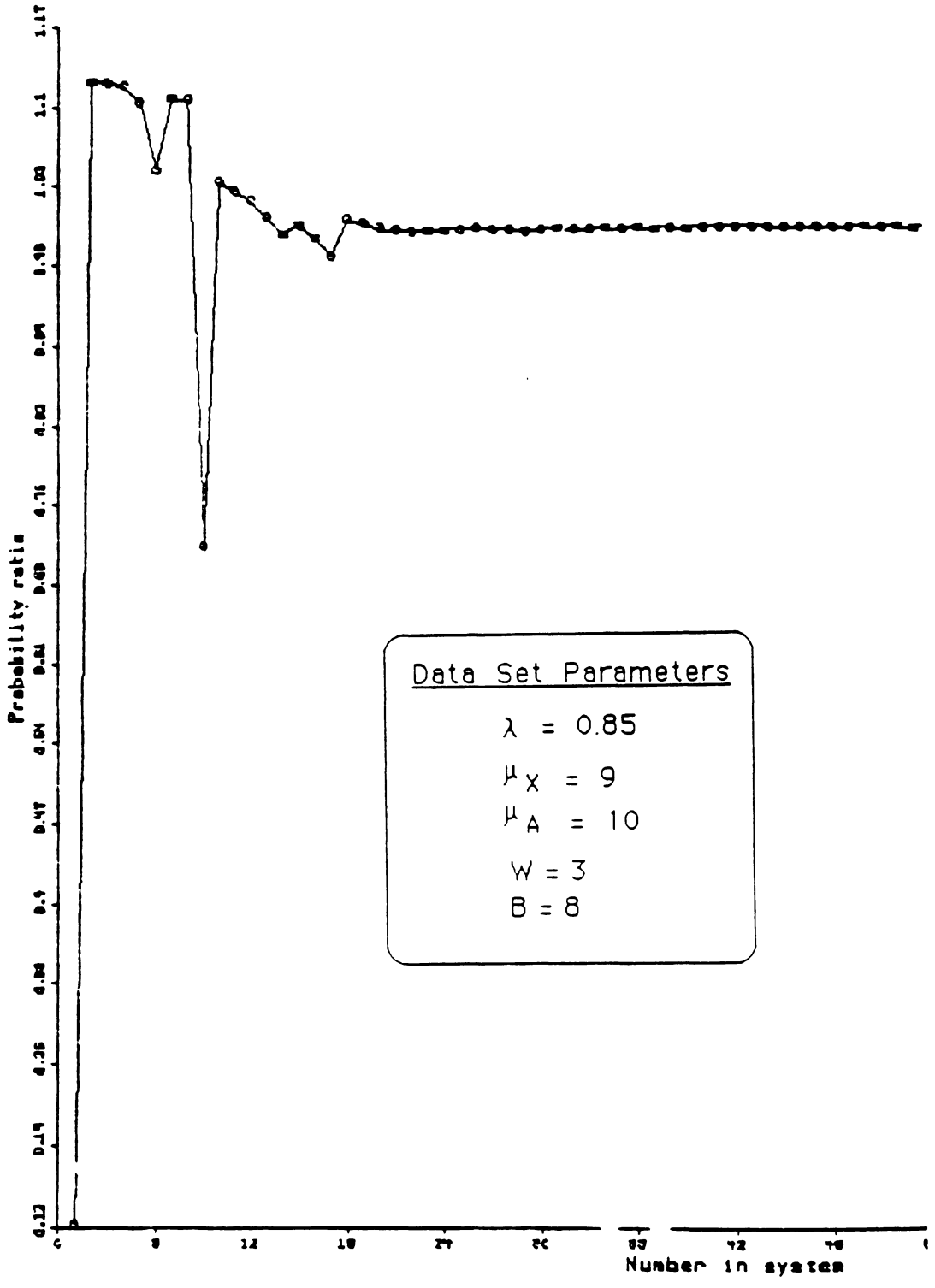


Figure 3.10- Probability Ratios B > W

$$p_{n+1,j} \simeq \frac{P_{n+1}}{P_n} p_{n,j}$$

$$\simeq \gamma_\infty p_{n,j}.$$

Since $\gamma_\infty < 1$ we see that the rate at which we enter state $(n,j+1)$ due to transmission completions in the actual system is less than the rate used in the closed approximation model. Since the closed model overestimates the rate at which we increase the number in the acknowledgment queue, it predicts a higher probability of having the acknowledgment queue full.

Based on the above discussion, we can modify the model of the closed system to more accurately estimate b_∞ by replacing the transmission service rate μ_X in the closed model with $\gamma_\infty \mu_X$. For this network with modified transmission service rate, let $\pi_W(W,\gamma_\infty)$ denote the probability of having all tokens in the acknowledgment queue. As in our previous look at the closed model, this probability has a convenient closed form expression. It is

$$b_\infty \simeq \pi_W(W,\gamma_\infty) = \left[\frac{\gamma_\infty \mu_X}{\mu_A}\right]^W \frac{1-\left[\gamma_\infty \mu_X/\mu_A\right]}{1-\left[\gamma_\infty \mu_X/\mu_A\right]^{W+1}}. \qquad (3.3.7)$$

We note that $\pi_W(W,1)$ in (3.3.7) is the same as $\pi_W(W)$ in (3.2.2).

We have now replaced the problem of estimating b_∞ by one of estimating γ_∞. Towards this end, let us look again at the aggregate rate equations (3.3.3). Summing the left hand sides and right hand sides of (3.3.3) for $i=0,1,2,\ldots,n_1+B-1$, for any $n_1>0$, and cancelling like terms, we get

$$\lambda \sum_{i=n_1}^{n_1+B-1} P_i = \mu_X (1-b_{n_1+B}) P_{n_1+B} . \tag{3.3.8}$$

(3.3.8) has the interpretation that the instantaneous probability of going from states (n_S, n_A) where $n_S < n_1+B$ to states (n'_S, n_A) where $n'_S \geq n_1+B$ is equal to the instantaneous probability of going from states (n'_S, n_A) to states (n_S, n_A). This is the type of flow-balance which we expect in steady state.

Under assumption (3.3.5), (3.3.8) becomes

$$\lambda \sum_{i=0}^{B-1} \gamma_\infty{}^i P_{n_1} = \mu_X (1-b_{n_1+B}) \gamma_\infty{}^B P_{n_1}$$

or

$$1-b_{n_1+B} = \frac{\lambda}{\mu_X} \frac{1-\gamma_\infty{}^B}{\gamma_\infty{}^B (1-\gamma_\infty)} . \tag{3.3.9}$$

Since the right-hand side of (3.3.9) is independent of n_1, we see that assumption (3.3.5) implies our assumption that the marginal probability of state (n, W) given n is a constant for large n. We thus re-write (3.3.9) as

$$b_\infty \simeq 1-\frac{\lambda}{\mu_X} \frac{1-\gamma_\infty{}^B}{\gamma_\infty{}^B (1-\gamma_\infty)} = r(\gamma_\infty). \tag{3.3.10}$$

We now have two equations, (3.3.7) and (3.3.10) relating the two unknown quantities b_∞ and γ_∞. (3.3.10) is based on balancing the flow between the aggregate states P_n and applying assumption (3.3.5), whereas (3.3.7) is based on estimating the marginal probability of state (n_S, W) based on assumptions about the rate of transitions between states (n, j), for large values of n. We can solve for the two unknowns by solving the equation $r(\gamma) = \pi_W(W, \gamma)$. The following algorithm computes this

solution.

Algorithm 1

1. Set γ_0=1. Set k=0.

2. Evaluate $\pi_W(W,\gamma_k)$ by (3.3.7).

3. Solve (3.3.10) for γ_{k+1} by finding γ_{k+1} for which $r(\gamma_{k+1}) = \pi_W(W,\gamma_k)$.

4. If $\gamma_k - \gamma_{k+1} < \varepsilon$ then terminate with $\gamma_\infty = \gamma_k$, and $b_\infty = \pi_W(W,\gamma_k)$. Otherwise, set $k=k+1$, go to 2.

To prove the convergence of the algorithm, we note the following,

a) $\dfrac{d\pi_W(W,\gamma)}{d\gamma} > 0$, $\gamma \in (0,1)$, (from (3.3.7)).

b) $\dfrac{dr(\gamma)}{d\gamma} > 0$, $\gamma \in (0,1)$, (from (3.3.10)).

c) $\lim_{\gamma \to 1} r(\gamma) = 1 - \dfrac{\lambda B}{\mu_X}$.

We now demonstrate that the sequences $\{\gamma_k\}$ and $\{\pi_W(W,\gamma_k)\}$ generated by the algorithm are monotone decreasing. We will prove this by induction. Assume that each sequence is decreasing for $k \le N$, and that both γ_N and $\pi_W(W,\gamma_N)$ are between zero and one. γ_{N+1} is determined from $\pi_W(W,\gamma_N)$ in step 3. By b) and the assumption that $\pi_W(W,\gamma_N) < \pi_W(W,\gamma_{N-1})$, $\gamma_{N+1} < \gamma_N$. Now a) and this result imply that $\pi_W(W,\gamma_{N+1}) < \pi_W(W,\gamma_N)$. We complete the induction proof by showing that $\gamma_1 < \gamma_0$=1. From the stability condition (3.2.3) we have

$$1-\frac{\lambda B}{\mu_X} > 1-\frac{1-\left(\mu_X/\mu_A\right)^W}{1-\left(\mu_X/\mu_A\right)^{W+1}} = \pi_W(W,1).$$

Now since $r(\gamma)$ is a continuous function of γ for $\gamma \in (0,1)$, and $\pi_W(W,1) < \lim_{\gamma \to 1} r(\gamma)$, then the value γ_1 for which 3) is satisfied with $\pi_W(W,1)$ must be less than 1.

To complete the proof of convergence of the algorithm, we show that the sequences $\{\gamma_k\}$ and $\{\pi_W(W,\gamma_k)\}$ are bounded from below by a positive number. This together with the proof of monotone decreasing sequences implies the existence of limiting values. We note from (3.3.7) that if $\gamma_k > 0$ then $\pi_W(W,\gamma_k) > 0$. This in turn implies that γ_{k+1} determined in step 3 is greater than γ^+, where we define γ^+ to be the value of γ for which $r(\gamma) = 0$. γ^+ exists and is strictly positive between 0 and 1, since $\lim_{\gamma \to 0} r(\gamma) = -\infty$, $\lim_{\gamma \to 1} r(\gamma) > 0$, and $r(\gamma)$ is continuous. We see then that

$$\lim_{k \to \infty} \gamma_k \geq \gamma^+ > 0$$

which implies that

$$\lim_{k \to \infty} \pi_W(W,\gamma_k) \geq \pi_W(W,\gamma^+) > 0.$$

This completes the proof.

Algorithm 1 was compared with empirical results, and was found to consistently *underestimate* the state-asymptotic value b_∞. This is actually not an undesirable result, since as discussed previously our estimate $\bar{b}$ should be less than b_∞. In fact, taking the value of $\pi_W(W,\gamma_1)$ from Algorithm 1 as an estimate of $\bar{b}$ proves to be very effective (!) and this is the algorithm used to compute the service rate in the approximate model. We state this here as Algorithm 2.

Algorithm 2

1. Set $b = \pi_W(W,1)$.
2. Find γ_1 to solve $r(\gamma_1) = b$.
3. Set $\bar{b} = \pi_W(W,\gamma_1)$.

At this point, we can only view the success of Algorithm 2 as a fortuitous accident. Our stated goal was to find a procedure for estimating b_∞, yet we stumble upon a good value for $\bar{b}$. In actual practice, determining b_∞ is not of great importance, as no engineer would design a system to operate with n_S very large; the value of $\bar{b}$ is what we desire. However, in the next Section we present a procedure which succeeds in estimating b_∞ accurately. By examining this procedure, we will gain insight into why Algorithm 2 works.

3.4. An Algorithm for b_∞

Equation (3.3.7) is based on a closed network approximation of the original network. A better approximation comes from applying assumptions (3.3.4) and (3.3.5) directly to the rate equations of the original system, (3.3.1). Recall from the previous Section, that under (3.3.4) and (3.3.5)

$$\frac{p_{n+1,j}}{p_{n,j}} = \frac{P_{n+1}}{P_n} = \gamma_\infty , \quad j=0,1,2,...,W. \tag{3.4.1}$$

Given (3.4.1), the rate equations (3.3.1) for $i \geq B$ become

$$\begin{cases} (\lambda+\mu_X)p_{n,0} = \mu_A p_{n,1} + \lambda\gamma_\infty^{-B}\, p_{n,0} \\ (\lambda+\mu_X+\mu_A)p_{n,j} = \mu_A p_{n,j+1} + \lambda\gamma_\infty^{-B} p_{n,j} + \mu_X \gamma_\infty p_{n,j-1}, \\ \qquad j = 1,2,\ldots,W-1 \\ (\lambda+\mu_A)p_{n,W} = \mu_X \gamma_\infty\, p_{n,W-1} + \lambda\gamma_\infty^{-B} p_{n,W} \end{cases} \tag{3.4.2}$$

Equations (3.4.2) are the rate equations of a closed network with the topology of Figure 3.2 and W entities circulating, but with state dependent service rates at each queue. $p_{n,j}$ in this interpretation is the probability of having $n_A = j$, and the index n is superfluous.

The system (3.4.2) can be solved for the conditional probability $\Pr\{n_A=W \mid n_S=n\}$ for any given value of γ_∞ by first assuming an arbitrary positive value for $p_{n,W}$, solving for each $p_{n,j}$, $j=0,1,2,\ldots,W-1$, and then calculating $p_{n,W}/\left[\sum_{j=0}^{W} p_{n,j}\right]$. The details of this procedure follow.

From (3.4.2) we can write

$$p_{n,0} = \frac{\mu_A}{\lambda(1-\gamma_\infty^{-B}) + \mu_X} p_{n,1}.$$

Define the coefficient of $p_{n,1}$ above as $f_1(\gamma_\infty)$. Each $p_{n,j}$, $j=1,2,\ldots,W-1$ can be expressed in terms of $p_{n,j+1}$ by

$$p_{n,j} = f_{j+1}(\gamma_\infty) p_{n,j+1} \tag{3.4.3}$$

where

$$f_{j+1}(\gamma_\infty) = \frac{\mu_A}{\lambda(1-\gamma_\infty^{-B}) + \mu_A + \mu_X(1-\gamma_\infty f_j(\gamma_\infty))}. \tag{3.4.4}$$

From (3.4.3)

$$p_{n,j} = \left[\prod_{t=j+1}^{W} f_t(\gamma_\infty) \right] p_{n,W}, \quad j=0,1,2,...,W-1.$$

Thus

$$b_\infty \simeq \frac{p_{n,W}}{\sum_{j=0}^{W} p_{n,j}} = \left[1 + \sum_{j=0}^{W-1} \prod_{t=j+1}^{W} f_t(\gamma_\infty) \right]^{-1} = g(\gamma_\infty). \tag{3.4.5}$$

We find an estimate of b_∞ and the associated γ_∞ by solving numerically for the unique solution on $\gamma > \gamma^+$ of $g(\gamma) = r(\gamma)$. Compare this to Algorithm 1, where we solve $\pi_W(W,\gamma) = r(\gamma)$. The new algorithm then replaces the estimate of b_∞ from a closed network approximation which uses a reduced transmission rate with an estimate from a closed network with state dependent service rates at the transmission and acknowledgment servers. For reference, we refer to the new procedure as Algorithm 3.

Algorithm 3 (a better estimate of b_∞).

1. Find the unique γ in $(\gamma^+,1)$ which solves $r(\gamma) = g(\gamma)$. Set γ_∞ to this value.

   (Step 1 uses the secant method to find a zero of $r(\gamma)-g(\gamma)$).

2. Set $b_\infty = g(\gamma_\infty)$.

The approximation of b_∞ obtained from Algorithm 3 is excellent, as is documented in Section 3.5.

We now have a good means of estimating b_∞. It was mentioned previously that the estimate of b_∞ from Algorithm 1 underestimated the true value. In fact we can prove that the estimate of b_∞ from Algorithm 1 is less than the value $g(\gamma_\infty)$ obtained from Algorithm 3. Let us begin by re-writing (3.3.7) as

$$\pi_W(W,\gamma)=\left[1+\sum_{j=0}^{W-1}\left(\frac{\mu_A}{\gamma\mu_X}\right)^{W-j}\right]^{-1}. \tag{3.4.6}$$

In this form, $\pi_W(W,\gamma)$ is easily compared to $g(\gamma)$ in (3.4.5). We shall next show that $f_j(\gamma) < (\mu_A/\gamma\mu_X)$ for $j=1,2,...,W$ and $\gamma^+<\gamma<1$. From this it follows that

$$\prod_{t=j+1}^{W} f_t(\gamma) < (\mu_A/\gamma\mu_X)^{W-j}. \tag{3.4.7}$$

Now comparing (3.4.5) and (3.4.6) using (3.4.7) we see that $\pi_W(W,\gamma)<g(\gamma)$. Since $r(\gamma)$ is monotonically increasing in γ, it follows that the value γ_{A1} for which $r(\gamma_{A1}) = \pi_W(W,\gamma_{A1})$ is less than the value γ_{A3} for which $r(\gamma_{A3}) = g(\gamma_{A3})$, and thus b_∞ from Algorithm 1 = $r(\gamma_{A1}) < r(\gamma_{A3})$ = value of b_∞ from Algorithm 3, (see Figure 3.11). We see then that the closed network approximation used in Algorithm 1 is not adequate to capture the state asymptotic behavior. Algorithm 1 always calculates an estimate of b_∞ less than the true state asymptotic value, which makes it useful for estimating $\bar{b}$.

Let us now wrap up this Section by proving that $f_j(\gamma) < (\mu_A/\gamma\mu_X)$, and thus proving (3.4.7). We first need

Lemma 1: For $\gamma^+<\gamma<1$, $\lambda(1-\gamma^{-B})+\mu_X > \gamma\mu_X$.

Proof: By definition of γ^+, for the given range of γ,

$$g(\gamma) = 1-\frac{\lambda}{\mu_X}\frac{1-\gamma^B}{\gamma^B(1-\gamma)}>0$$

i.e.,

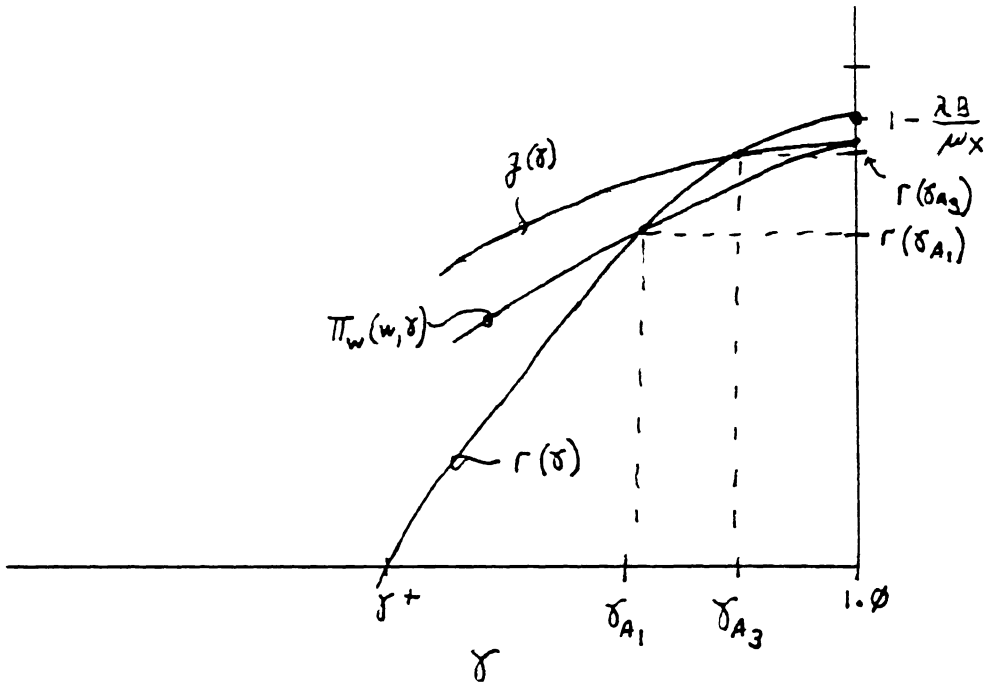


Figure 3.11 - Comparing Algorithms 1 and 3

$$1+\frac{\lambda(1-\gamma^{-B})}{\mu_X(1-\gamma)}>0.$$

Simple manipulation of this last equation gives the desired result. *QED*.

We now use induction to prove $f_j(\gamma)<(\mu_A/\gamma\mu_X)$. For $j=1$,

$$f_1(\gamma)=\frac{\mu_A}{\lambda(1-\gamma^{-B})+\mu_X}<\frac{\mu_A}{\gamma\mu_X}$$

by Lemma 1. Assume that $f_j(\gamma)<(\mu_A/\gamma\mu_X)$ for $j\leq J$. $f_{J+1}(\gamma)$ is given in (3.4.4). By the induction assumption, $\mu_A-\mu_X\gamma f_J(\gamma)>0$. Thus

$$f_{J+1}(\gamma)=\frac{\mu_A}{\lambda(1-\gamma^{-B})+\mu_X+[\mu_A-\mu_X\gamma f_J(\gamma)]}<\frac{\mu_A}{\lambda(1-\gamma^{-B})+\mu_X}. \qquad (3.4.8)$$

From (3.4.8) and Lemma 1

$$f_{J+1}(\gamma)<\frac{\mu_A}{\lambda(1-\gamma^{-B})+\mu_X}<\frac{\mu_A}{\gamma\mu_X}.$$

This completes the induction proof.

3.5. Approximation Results

In this section we discuss tables and figures comparing the queue length distribution of our approximate model with exact results. To improve the readability of this Section, all tables are presented in Appendix A.

Sixty-one test cases were run. Twenty-four of the cases have $B=1$, i.e., no fragmentation of arriving packets, and the other thirty-seven cases have $B>1$. For each example we calculated the mean service rate of the approximate queue by Algorithm 2, solved that queue numerically for the queue length distribution, and compared the results to the exact results calculated from a numerical procedure. The various

system parameters were varied so as to evaluate the performance of the approximation scheme under a wide set of conditions. One parameter of interest is the nominal workload $\lambda B / \mu_X$. This was varied from light load (about 30% of the stability bound) to very heavy load, (nearly 100% of the stability bound). Another parameter is the ratio μ_X/μ_A, which strongly influences the probability of having all tokens in the acknowledgment queue. The higher the ratio, the greater that probability. (See (3.3.6)). μ_X/μ_A was varied from low (.25) to high (.9). We also varied the batch arrival and window sizes. Due to the limitations of the numerical procedure, these parameters were kept fairly small. We looked at $W = 3, 4, 8, 16$, and 20, and $B = 1, 3$, and 8. Tables A.1 and A.2 detail the parameters used in each example. In both tables we present the system utilization, which is the ratio of the offered load to the maximum throughput, (see (3.2.3)). In Table A.1, we also present the ratio λ/μ_X, which is the nominal load on the transmission facility. The difference between the nominal load and the system utilization is a measure of the degradation of transmission capability introduced by sliding window flow control.

Let us examine the results of the approximation algorithm. Tables A.3 and A.4 compare the exact and predicted average value of n_S for each data set. The errors are in general quite small. As has been observed by other researchers, the worst cases are when W is small and the system utilization is high, e.g. data sets 14, 30, and 31.

We also note the importance of the ratio μ_X/μ_A on the accuracy of the approximation. Data sets 25 and 32 both have a high utilization and a small window size, but μ_X/μ_A is low and the prediction is good. A combination of all three conditions,

small window size, high utilization, and μ_X/μ_A close to 1 together lead to a fairly large value of b_∞. As we mentioned in Section 3.3, it is precisely this condition which we would expect to lead to poor performance. Figure 3.12 gives a plot of error vs. the empirically observed value of b_∞ from the exact solutions, for the data sets with $B=1$. We shall explain in our discussion of the performance of Algorithm 3 how this empirically observed value of b_∞ was determined.

It is possible to have good agreement in the predicted and actual value of the mean number in the system, while still having poor approximation of the distribution of the number in the system. To evaluate the prediction of the distribution, the exact probability of having n in the system was compared to the predicted values for every n with probability larger than 0.001. We then extracted from this comparison two measures of closeness, the maximum absolute deviation, and the maximum relative error, (error/(actual value)). The results of these comparisons are shown in Tables A.5 and A.6. We see that the actual and predicted results are quite close for most cases. Again the worst performance is in those data sets where b_∞ is large. In some cases (e.g. data sets 12-14), the maximum relative error is quite large. We note with some comfort that these large relative errors occur in the tail of the probability distribution, and that the corresponding absolute deviation is not excessive. Figure 3.13 is a plot of maximum relative error vs. b_∞ for the data sets with $B=1$. Figure 3.14 shows the actual probabilities and the predicted probabilities for data set 31, which was one of the worst examples. In this data set the window size is 3, the load is 96%, and $\mu_X/\mu_A = .91$.

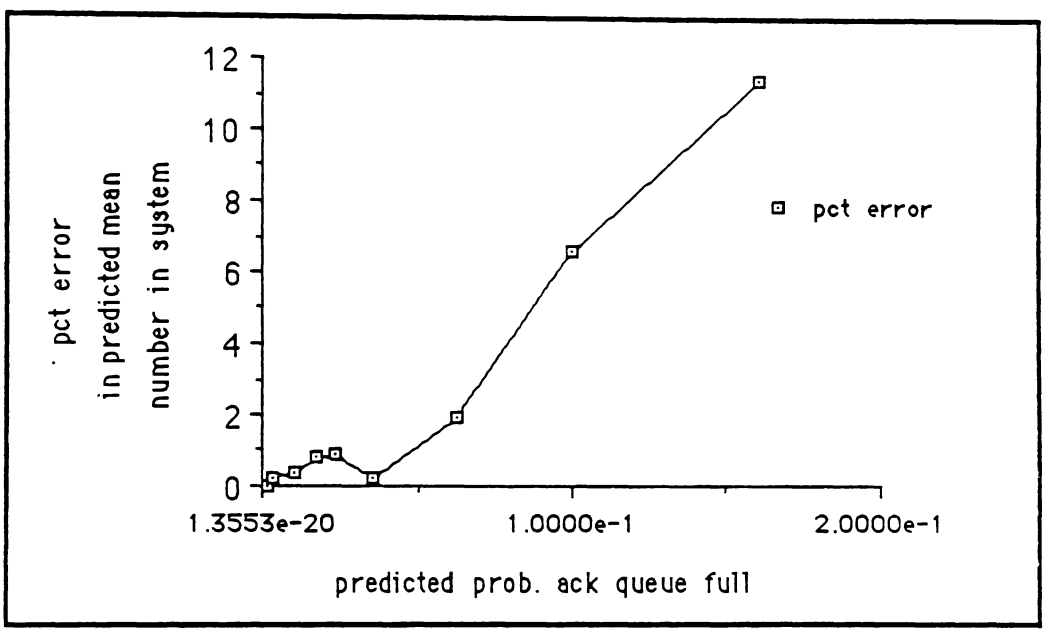


Figure 3.12 - Prediction error for mean in system vs. observed b_∞

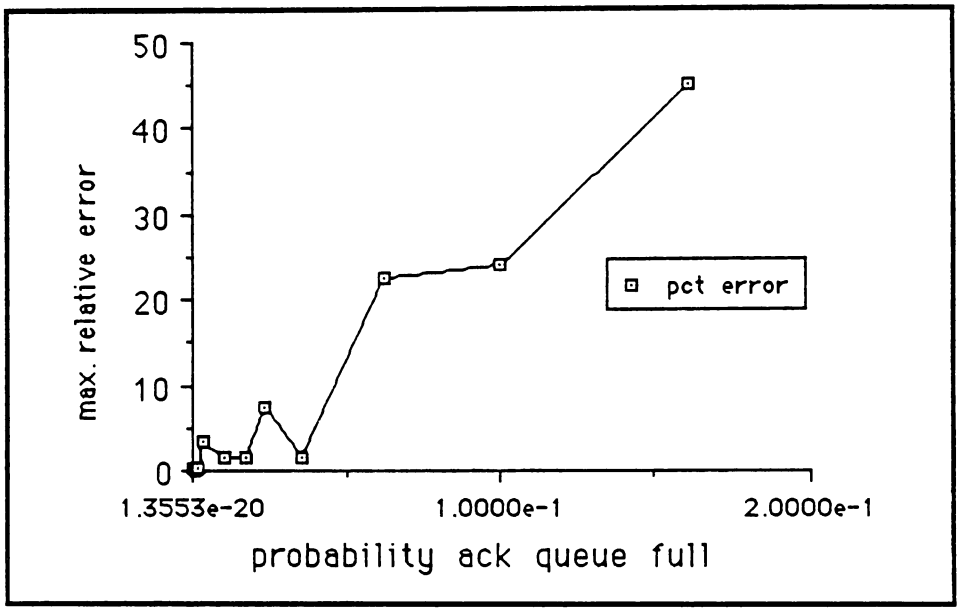
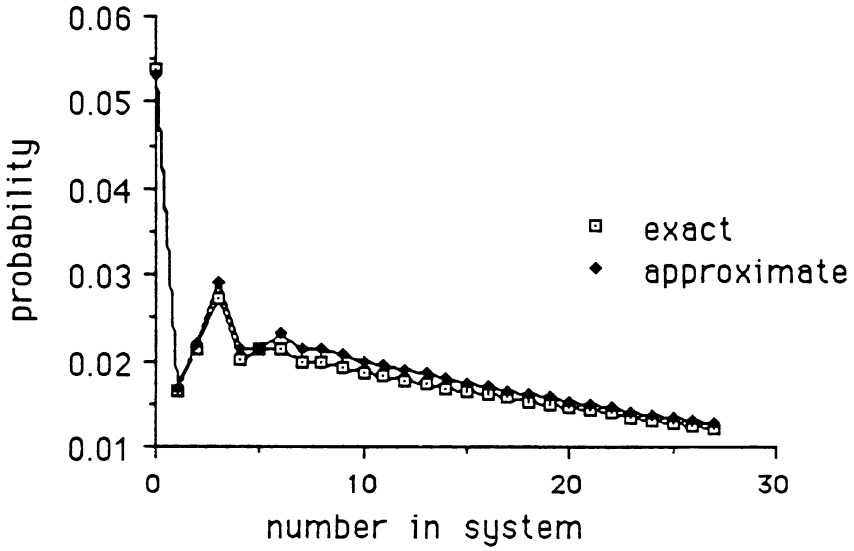


Figure 3.13 - Maximum relative error vs. observed b_∞



Data Set Parameters

$\lambda = 2.5$

$\mu_X = 10$

$\mu_A = 11$

$W = 3$

$B = 3$

Figure 3.14 - Comparison of exact and aproximate solutions

Finally, we compare the predictions of Algorithm 3 for b_∞ with the empirically observed values from the exact solutions. (The comparisons which we present are stated in terms of the limiting throughput. This is an artifact of the methodology employed at the time these comparisons were made. The nature of the results is not affected, as the throughput is simply the transmission rate times 1-b_∞).

Our empirically observed values of b_∞ were determined from the exact solution as follows. As b_∞ is the state asymptotic value of b_i, it was necessary to define a "limiting" state. The definition used was:

> The first limiting state is the value of n_S for which the 20 succeeding states have throughputs with less than 0.001% variation.

The throughput in the state with n_S ten higher than the first limiting state was used as the true value of $\mu_X(1-b_\infty)$. The comparison of the approximation and the "true" limiting values is shown in Table A.7 for the data sets with $B>1$, along with the value of the first "limiting" state.

3.6. Comparison to Flow Equivalent Server Technique

Another means of evaluating the effectiveness of the approximation procedure developed in Section 3.3 is to compare it to other approaches. The most effective current techniques appear to be those using the Norton's approximation server approach to determine the characteristics of the transmission network, ([Reis81], [FdPeWi87], [Dall87]). In the case of the simple model of Figure 3.1, all of these approaches are the same. The closed model of Figure 3.2 is evaluated for its

throughput characteristics with populations 1,2,...,W, and the population dependent throughput values $\gamma(j)$ are used in (2.3) to determine the values $q_{i,j}$. Since the network in Figure 3.2 has exponential servers, in all of the approaches the value $\gamma(j)$ is calculated as

$$\gamma(j) = \mu_X\ (1-\pi_j(j)).$$

(See (3.2.1) for the definition of $\pi_i(i)$).

Before comparison can be made, some equations need to be developed which relate $q_{i,j}$ calculated by these procedures to P_n used to evaluate the new approximation. Recall that $q_{i,j}$ is the probability of having i packets in the holding queue and j tokens in the token queue. P_n is the probability of having n total packets in the holding and transmission queues.

We develop estimates of P_n from the $q_{i,j}$ as follows.

$$P_n = \Pr\{n_S = n\} = \sum_{i=0}^{n} \Pr\{n_H = i\}\Pr\{n_X = n-i \mid n_H=i\} \tag{3.6.1}$$

$$= \Pr\{n_H = 0\}\Pr\{n_X = n \mid n_H = 0\} + \sum_{i=1}^{n} \Pr\{n_H = i\}\Pr\{n_X = n-i \mid n_H=i\}.$$

Let us examine the first term in the above sum

$$\Pr\{n_X = n \mid n_H = 0\} = \sum_{j=0}^{W} \Pr\{n_X=n \mid n_T=j\}\Pr\{n_T = j \mid n_H=0\}. \tag{3.6.2}$$

Using the definition of conditional probability and the definition of $q_{0,j}$, we rewrite (3.6.2) as

$$\Pr\{n_X = n \mid n_H = 0\} = \sum_{j=0}^{W} \Pr\{n_X=n \mid n_T=j\} \frac{q_{0,j}}{\Pr\{n_H=0\}} \tag{3.6.3}$$

$$= \sum_{j=0}^{W-n} \Pr\{n_X=n \mid n_T=j\} \frac{q_{0,j}}{\Pr\{n_H=0\}}$$

the last step above reflecting the fact that $n_X + n_T \leq W$. In the spirit of the Norton's approximation methodology, we assume that $\Pr\{n_X=n \mid n_T=j\}$ is the steady state probability of having $n_X = n$ when there are $W-j$ tokens in the closed network of Figure 3.2. Defining $\rho=\mu_A/\mu_X$, this assumption results in

$$\Pr\{n_X=n \mid n_T=j\} = \rho^n \frac{1-\rho}{1-\rho^{W-j+1}}. \tag{3.6.4}$$

In the same spirit, we assume that $\Pr\{n_X = n-i \mid n_H=i\}$ for $i>0$ is the probability of having $n-i$ in the transmission queue when there are W in the closed network of Figure 3.2 (If $n_H > 0$ then all tokens must be in use). This leads to

$$\Pr\{n_X = n-i \mid n_H = i\ , i>0\} = \begin{cases} \rho^{n-i} \dfrac{1-\rho}{1-\rho^{W+1}}, & 0 \leq n-i \leq W \\ 0, & \text{otherwise} \end{cases} \tag{3.6.5}$$

We note that since $n_H > 0$ only when $n_T = 0$, $\Pr\{n_H = i\} = q_{i,0}$ for each $i>0$. Using this observation and formulae (3.6.2) - (3.6.5), (3.6.1) can be re-written as

$$P_n = \begin{cases} \displaystyle\sum_{j=0}^{W-n} \rho^n \frac{1-\rho}{1-\rho^{W+1-j}} q_{0,j} + \sum_{i=1}^{n} \rho^{n-i} \frac{1-\rho}{1-\rho^{W+1}} q_{i,0}, & n \leq W \\ \displaystyle\sum_{i=n-W}^{n} \rho^{n-i} \frac{1-\rho}{1-\rho^{W+1}} q_{i,0}, & n > W \end{cases} \tag{3.6.6}$$

We thus have an expression for P_n in terms of the $q_{i,j}$.

The equations (2.3) for determining $q_{i,j}$ are for the case of single arrivals to the holding queue. If we define $q_{0,j} = 0$ for $j > W$, then for batch arrivals of size B the appropriate balance equations are:

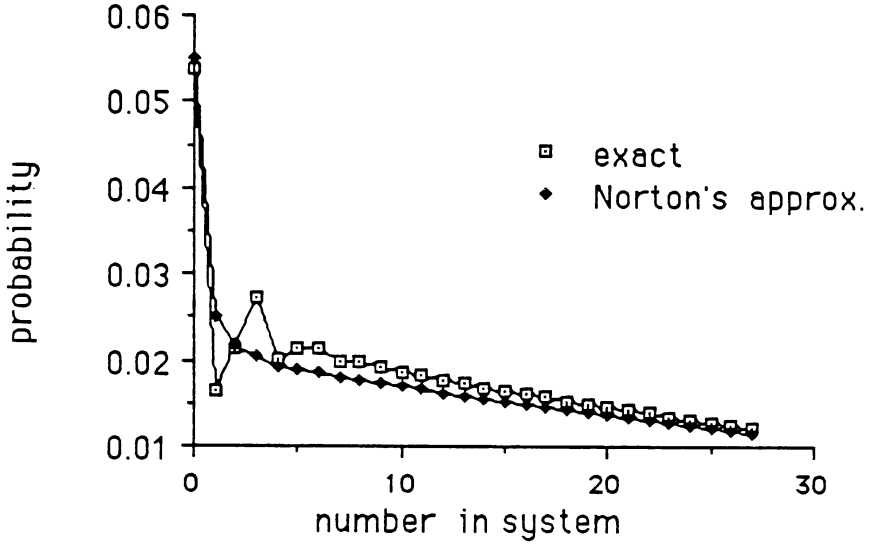
$$\begin{cases} \lambda \sum_{k=j+1}^{j+B} q_{0,k} = \gamma(W-j) q_{0,j}\ , \quad j=W-1,W-2,\ldots,1 \\ \lambda \left[\sum_{j=1}^{B-i} q_{0,j} + \sum_{k=0}^{i-1} q_{k,0} \right] = \gamma(W) q_{i,0}\ , \quad 0<i<B \\ \lambda \sum_{k=i-B}^{i-1} q_{k,0} = \gamma(W) q_{i,0}\ , \quad i \geq B \end{cases} \tag{3.6.7}$$

There is no known closed form solution for the above equations, but due to the very regular structure for $i \geq B$, the system is easily solved using the numerical procedure of Neuts, [Neut81]. We note that (3.6.6) for evaluating P_n still holds with values $q_{i,j}$ derived from (3.6.7).

Tables A.8 and A.9 give the comparisons of the mean value of P_n predicted by the Norton's approximation method and the exact solution. For $B=1$, the results are comparable to those in Table A.3, but consistently slightly worse. Table A.9 gives the comparison of mean values for cases where $B>1$. We see that the results are not as good here, and also not as good as the results of the new method, shown in Table A.4.

Tables A.10 and A.11 compare the distribution of the number in the system predicted vs. actual, using the same measures as in Section 3.5. Table A.10 shows fairly good results for the data sets where $B=1$, although again not as good as the comparable Table A.5 In Table A.11 we do not show the maximum relative error, as

in all cases it corresponded to the same data point as the maximum deviation. We see that in all examples the Norton's approximation technique severely overestimated P_1. The mean values of Table A.9 did not show such large deviation, since the Norton's approximation underestimated nearly every other P_n. Usually P_B was the most severly underestimated. Figure 3.15 shows a comparison of the predicted distribution of n_S for the same data set as Figure 3.14. The poor performance of the Norton's approximation procedure with $B > 1$ tallies with the observation of Schwartz in [Schw82].



Data Set Parameters

$\lambda = 2.5$

$\mu_X = 10$

$\mu_A = 11$

$W = 3$

$B = 3$

Figure 3.15 - Comparison of Exact and Norton's Approximation

CHAPTER 4

NESTED WINDOW FLOW CONTROLS

The analysis of Chapter 3 provides us with a means of constructing an equivalent server for a single hop sliding window flow controlled network with bulk (fragmented) arrivals. We next consider the situation where such a network is itself nested in another level of flow control. Consider the network in Figure 4.1. Here the connection between the sender and the receiver has two levels of flow control. The lower level has a window size of W_1. Each higher level packet is fragmented into B_1 lower level packets, thus the lower level sees bulk arrivals of size B_1. We shall analyze this type of network by a hierarchical methodoldgy. Using Algorithm 2 of Chapter 3 we can reduce the lower level sub-network to a single server queue. (The appropriate arrival process to use in the lower level model is batches of size B_1 arriving at rate λB_2). We embed the approximate queue obtained from the lower level model in the higher level model to form the network of Figure 4.2. It is this network which we analyze in this chapter.

Our analysis will parallel that of Chapter 3, with a modification at the end to account for the fragmentation and re-assembly in the transmission path. We will draw upon the observations and methodology in Chapter 3, thus the expository material in this Chapter will be somewhat less. We also note that we do not have a Section in this Chapter evaluating the flow-equivalent server techniques, as we have already seen that that approach is poor when arrivals occur in batches.

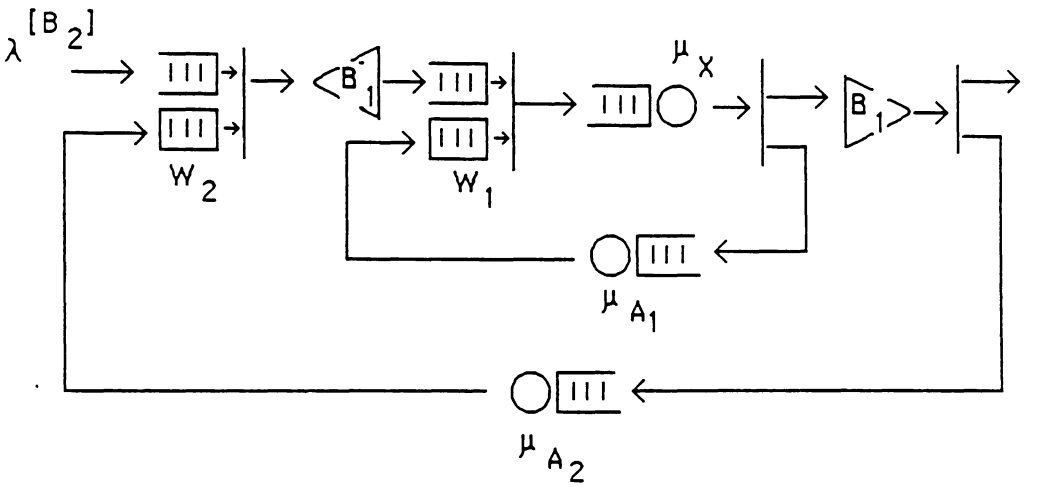


Figure 4.1 - A network with two levels of flow control

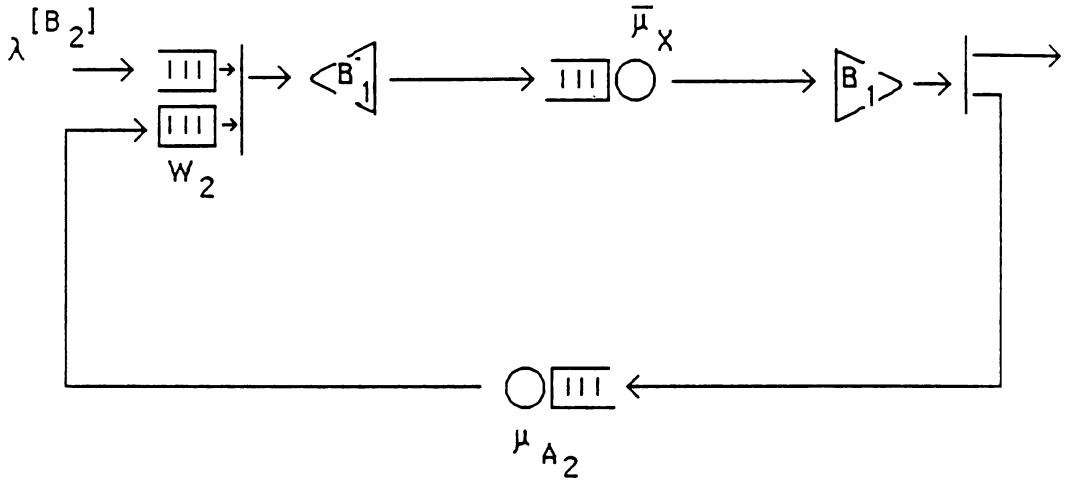


Figure 4.2 - Network with lower level reduced

4.1. The Model

The modelling issues related to approximating the behavior of a communications system raised in Chapter 3 are equally valid with the model presented here. Enough said.

For simplicity of notation, we redraw Figure 4.2 with new notation indicated in Figure 4.3. Arrivals to the network occur according to a Poisson process with rate λ. Arrivals occur as batches of B **high-level packets**. The arriving packets are placed in the holding queue and are each instantaneously paired with a token, until either the token queue is emptied or all packets have a token. Each token/packet pair is fragmented into f **transmission packets**. Each transmission packet is enqueued in the transmission queue. The process of assigning tokens, fragmenting, and placing packets in the transmission queue can either be assumed to take no time, or be included in the delivery time. The reader is referred to the discussion in Section 3.1 of the modelling implications.

Service time for each transmission packet at the transmission server is exponentially distributed with a mean service time of $1/\mu_X$. Upon completion of transmission service, the transmission packet is placed in the reassembly buffer. When each of the f transmission packets forming the original higher level packet is in the reassembly buffer, they are reformed into the original packet/token pair and released. We assume that every fragment of a high level packet is delivered to the reassembly buffer before any fragments of a succeeding high level packet. This preservation of order corresponds to a virtual-circuit transmission protocol. After reassembly, the

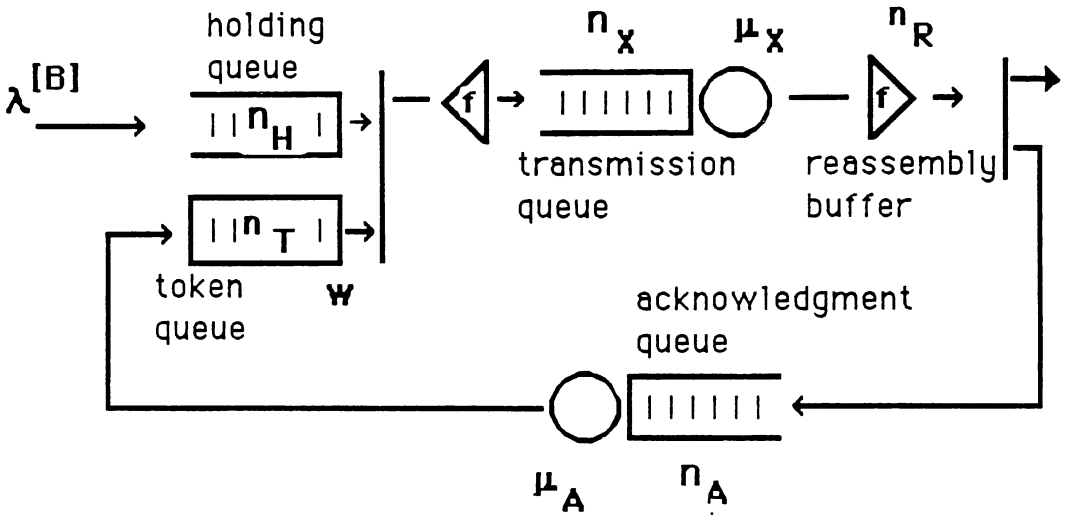


Figure 4.3 - Notation for model of Chapter 4

high-level packet is delivered to the receiving peer, and the token is placed in the acknowledgment queue. Reassembly, delivery to the receiving peer, and acknowledgment generation are assumed to take no time, or can be assumed to be part of the transmission time.

Tokens are served one at a time on a first-come-first served basis at the acknowledgment queue. Service at the acknowledgment queue is exponentially distributed with a mean of $1/\mu_A$. Upon completion of acknowledgment service the token is returned to the token queue. If there is a packet waiting in the holding queue, the token is immediately paired with that packet, and the token/packet pair is delivered to the transmission subnetwork as described above.

4.2. Maximum Throughput

As was discussed in Chapter 3, the maximum throughput of the model is given by the throughput of the model under saturation. The closed network used to represent the saturated system is shown in Figure 4.4. Again it consists of the transmission and acknowledgment paths of the original model with W entities circulating.

Unlike the model of Chapter 3, there is no convenient closed form expression for the maximum throughput of this model. In Section 4.3 we will present the steady state probability balance equations for the closed network, and indicate how they can be solved. The probability needed to evaluate the maximum throughput can be obtained from that solution. We will point out how this is done when we look at the balance equations.

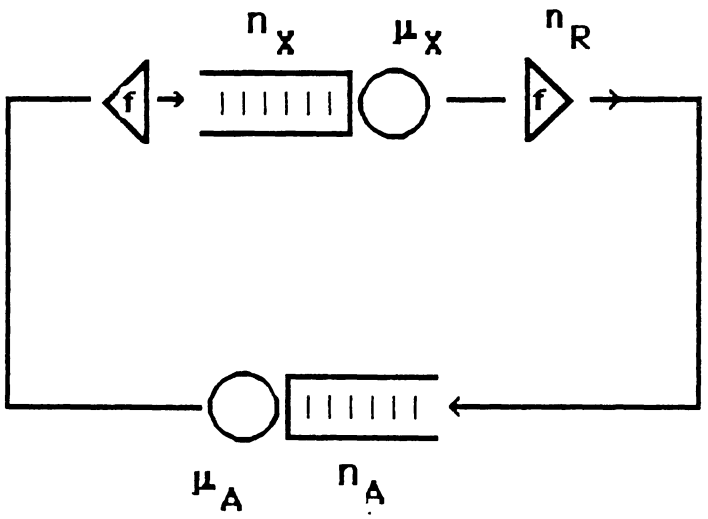


Figure 4.4 - Closed Model Used in Approximation Procedure

4.3. Approximation Procedure

Let us define the following notation:

n_H :number of high level packets in holding queue

n_T : number of tokens in token queue

n_A : number of tokens in acknowledgment queue

n_X : number of transmission packets in transmission queue

n_R : number of transmission packets in reassembly buffer

n_S : number of high level packets undelivered ,
(i.e., the number of packets "in the system").

$$n_S = n_H + \frac{1}{f}(n_X + n_R).$$

Given the triple (n_S, n_A, n_R) we can calculate the other quantities above using the following relations:

$$n_X = f \min(n_S, W - n_A) - n_R,$$

$$n_T = W - n_A - (n_X + n_R)/f,$$

$$n_H = n_S - (n_X + n_R)/f..$$

The triple (n_S, n_A, n_R) then suffices as the state of the system. We will denote a specific state by (i, j, k).

Let $p_{i,j,k}$ be the probability of finding the system in state (i, j, k) in steady state. The steady state probability balance equations which determine $p_{i,j,k}$ are:

$$\mathbf{n_S=0}\quad \begin{cases} \lambda p_{0,0,0}=\mu_A p_{0,1,0} \\ (\lambda+\mu_A)p_{0,j,0}=\mu_A p_{0,j+1,0}+\mu_X p_{1,j,f-1}, & j=1,2,\cdots,W-1 \\ (\lambda+\mu_A)p_{0,W,0}=\mu_X p_{1,W-1,f-1} \end{cases} \quad (4.3.1)$$

$$\mathbf{0<n_S<B}\quad \begin{cases} (\lambda+\mu_X)p_{i,0,0}=\mu_A p_{i,1,0} \\ (\lambda+\mu_X)p_{i,0,k}=\mu_A p_{i,1,k}+\mu_X p_{i,0,k-1}, & k=1,2,\cdots,f-1 \\ (\lambda+\mu_A+\mu_X)p_{i,j,0}=\mu_A p_{i,j+1,0}+\mu_X p_{i+1,j-1,f-1}, & j=1,2,\cdots,W-1 \\ (\lambda+\mu_A+\mu_X)p_{i,j,k}=\mu_A p_{i,j+1,k}+\mu_X p_{i,j,k-1}, & \\ j=1,2,\cdots,W-2 \quad k=1,2,\cdots,f-1 & \\ (\lambda+\mu_A+\mu_X)p_{i,W-1,k}=\mu_X p_{i,W-1,k-1}, & k=1,2,\cdots,f-1 \\ (\lambda+\mu_A)p_{i,W,0}=\mu_X p_{i+1,W-1,f-1} \end{cases}$$

$$\mathbf{n_S\geq B}\quad \begin{cases} (\lambda+\mu_X)p_{i,0,0}=\mu_A p_{i,1,0}+\lambda p_{i-B,0,0} \\ (\lambda+\mu_X)p_{i,0,k}=\mu_A p_{i,1,k}+\mu_X p_{i,0,k-1}+\lambda p_{i-B,0,k}, & k=1,2,\cdots,f-1 \\ (\lambda+\mu_A+\mu_X)p_{i,j,0}=\mu_A p_{i,j+1,0}+\mu_X p_{i+1,j-1,f-1}+\lambda p_{i-B,j,0}, & j=1,2,\cdots,W-1 \\ (\lambda+\mu_A+\mu_X)p_{i,j,k}=\mu_A p_{i,j+1,k}+\mu_X p_{i,j,k-1}+\lambda p_{i-B,j,k}, & \\ j=1,2,\cdots,W-2 \quad k=1,2,\cdots,f-1 & \\ (\lambda+\mu_A+\mu_X)p_{i,W-1,k}=\mu_X p_{i,W-1,k-1}+\lambda p_{i-B,W-1,k}, & k=1,2,\cdots,f-1 \\ (\lambda+\mu_A)p_{i,W,0}=\mu_X p_{i+1,W-1,f-1}+\lambda p_{i-B,W,0} \end{cases}$$

As in the analysis of the previous model, we wish to reduce the flow controlled network of Figure 4.3 to a single server queue, and to that end we look at the aggregate rate equations governing the probabilities P_i,

$$P_i=\sum_{j=0}^{W}\sum_{k=0}^{f-1}p_{i,j,k}.$$

P_i is the probability of having i high level packets in the system. We obtain the rate equations governing P_i by summing all of the rate equations (4.3.1) with i as a subscript on the left hand side. These equations are, (with like terms on left and right hand sides cancelled),

$$\begin{cases} \lambda P_0 = \mu_X \sum_{j=0}^{W-1} p_{1,j\,f-1} \\ \lambda P_i = \mu_X \sum_{j=0}^{W-1} p_{i+1,j\,f-1} - \mu_X \sum_{j=0}^{W-1} p_{i,j\,f-1}, \quad 0<i<B \\ \lambda P_i = \lambda P_{i-b} + \mu_X \sum_{j=0}^{W-1} p_{i+1,j\,f-1} - \mu_X \sum_{j=0}^{W-1} p_{i,j\,f-1}, \quad i \geq B \end{cases} \tag{4.3.2}$$

Comparing (4.3.2) to its counterpart in Chapter 3, (3.3.2), we see that the summation $\sum_{j=0}^{W-1} p_{i,j\,f-1}$ replaces $(P_i - p_{i,W})$. These two quantities share a common probabilistic interpretation. Each is the probability of being in those states from which an immediate transition to a state with one less high level packet in the system is possible when there are i packets in the system. In the model of Chapter 3, each transmission completion causes a reduction of the number of packets in the system. The only state from which this transition could not occur is the state where all tokens are in the acknowledgment queue, state (i,W). In the current model, the number of high level packets in the system only decreases when the final fragment of an original high level packet completes transmission. This only occurs when the number in the reassembly buffer is $f-1$.

We saw in Chapter 3 that the marginal probability of having all the tokens in the acknowledgment queue, b_i, played a pivitol role. Let us define the corresponding quantity here as a_i,

$$a_i = \frac{p_{i,W,0}}{P_i}. \tag{4.3.3}$$

We can represent $\sum_{j=0}^{W-1} p_{i,j\,f-1}$ in terms of a_i by assuming that, for $i>0$,

$$\sum_{j=0}^{W-1} p_{i,j,k} = \sum_{j=0}^{W-1} p_{i,j,k'} \tag{4.3.4}$$

for each pair $k,k' \in \{0,1,\cdots,f-1\}$. Assumption (4.3.4) states that it is equally likely to find any of the possible reassembly buffer populations when there are $i>0$ packets in the system, as long as $n_A < W$, i.e., as long as there are packets in the transmission queue. Since the mean time spent with k in the reassembly buffer, (given that a packet is in transmission), is $1/\mu_X$ for each possible value of k, assumption (4.3.4) seems reasonable.

(In fact, while it is true that

$$\sum_{i=1}^{\infty}\sum_{j=0}^{W-1} p_{i,j,k} = \sum_{i=1}^{\infty}\sum_{j=0}^{W-1} p_{i,j,k'}$$

by the argument used above, (4.3.4) is *not* true. We will examine the true state of affairs in Section 4.5, but for now assumption (4.3.4) is a good assumption for the type of "average" behavior we are seeking to replicate).

Using assumption (4.3.4) we can write

$$P_i = p_{i,W,0} + \sum_{k=0}^{f-1}\sum_{j=0}^{W-1} p_{i,j,k}$$

$$= p_{i,W,0} + f\sum_{j=0}^{W-1} p_{i,j,f-1}$$

thus

$$\sum_{j=0}^{W-1} p_{i,j,f-1} = (P_i - p_{i,W,0})/f \tag{4.3.5}$$

$$= P_i(1-a_i)/f.$$

Substituting this expression for $\sum_{j=0}^{W-1} p_{i,j,f-1}$ into (4.3.2), we see that under assumption

(4.3.3), (4.3.2) is the set of rate equations for a bulk arrival queue with state dependent exponential service rate $\frac{\mu_X}{f}(1-a_i)$.

As with the b_i, empirical observation shows that the conditional probabilities a_i stabilize to a constant as i gets large. Figure 4.5 illustrates this for a typical data set. Let us denote this state asymptotic value by a_∞.

We wish to construct a single queue approximation to the model under study, and we would thus like to find a value $\bar{a}$ which is an average of the a_i. Let us try to mimic the procedure used in Chapter 3.

We have seen in Chapter 3 that the aggregate state probabilities P_i tend to a constant ratio as i gets large. The same phenomenon is observed in the current model. (See Figure 4.6). We thus posit that for i large

$$P_{i+1} = \gamma_\infty P_i \tag{4.3.6}$$

for some $\gamma_\infty \in (0,1)$. A formula for γ_∞ comes from summing the equations (4.3.2) for $i=0,1,2,...,N+B$, for any large N, which gives us

$$\lambda \sum_{i=0}^{N+B} P_i = \lambda \sum_{i=0}^{N} P_i + \mu_X \sum_{j=0}^{W-1} p_{N+B+1,j} f_{-1}$$

or

$$\lambda \sum_{i=N+1}^{N+B} P_i = \mu_X \sum_{j=0}^{W-1} p_{N+B+1,j} f_{-1}. \tag{4.3.7}$$

Using assumption (4.3.6) and (4.3.5), (4.3.7) becomes

$$\lambda \sum_{i=0}^{B-1} \gamma_\infty^i P_{N+1} = \frac{\mu_X}{f}(1-a_{N+B+1})\gamma_\infty^B P_{N+1}$$

or

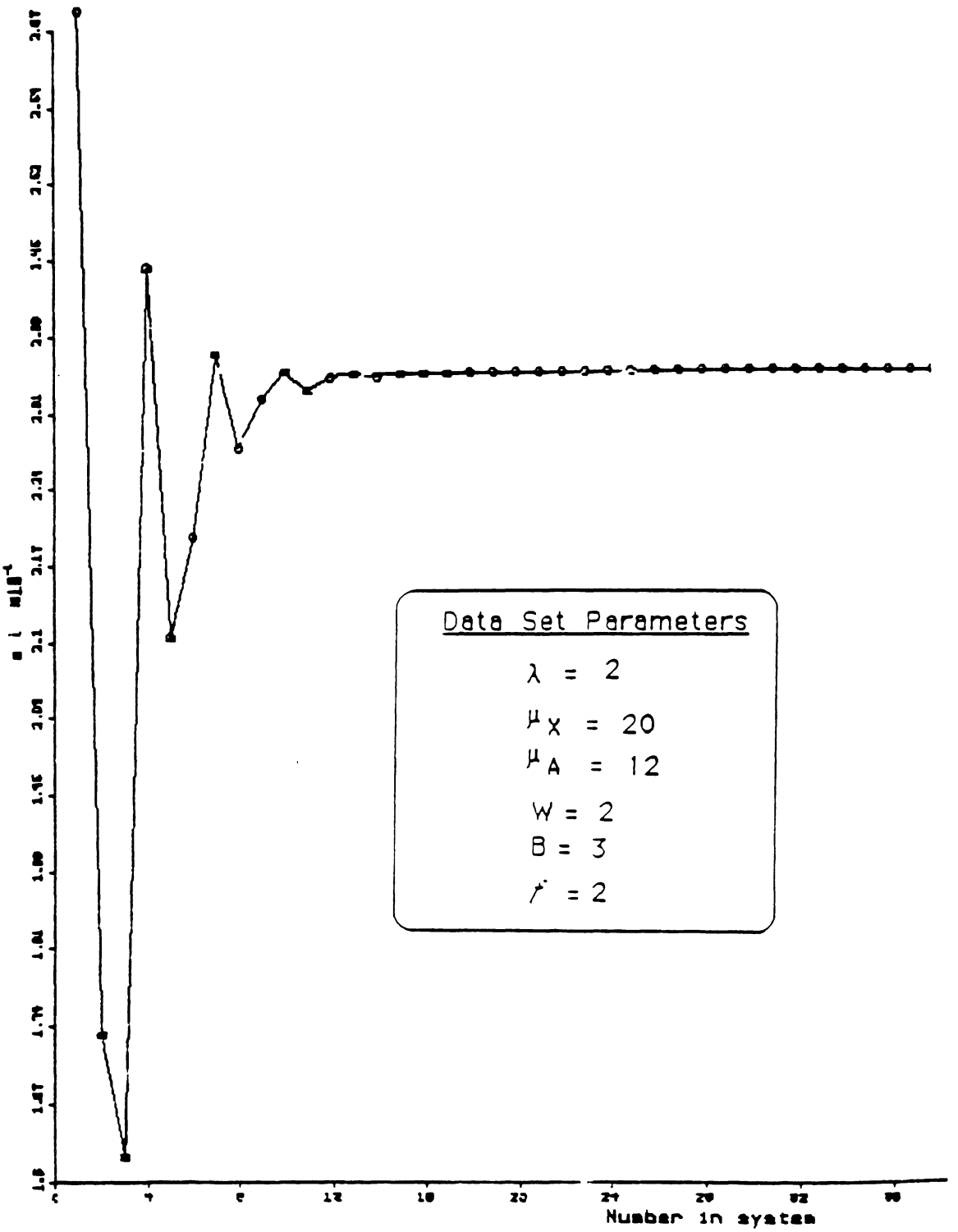


Figure 4.5 - a_i vs. i

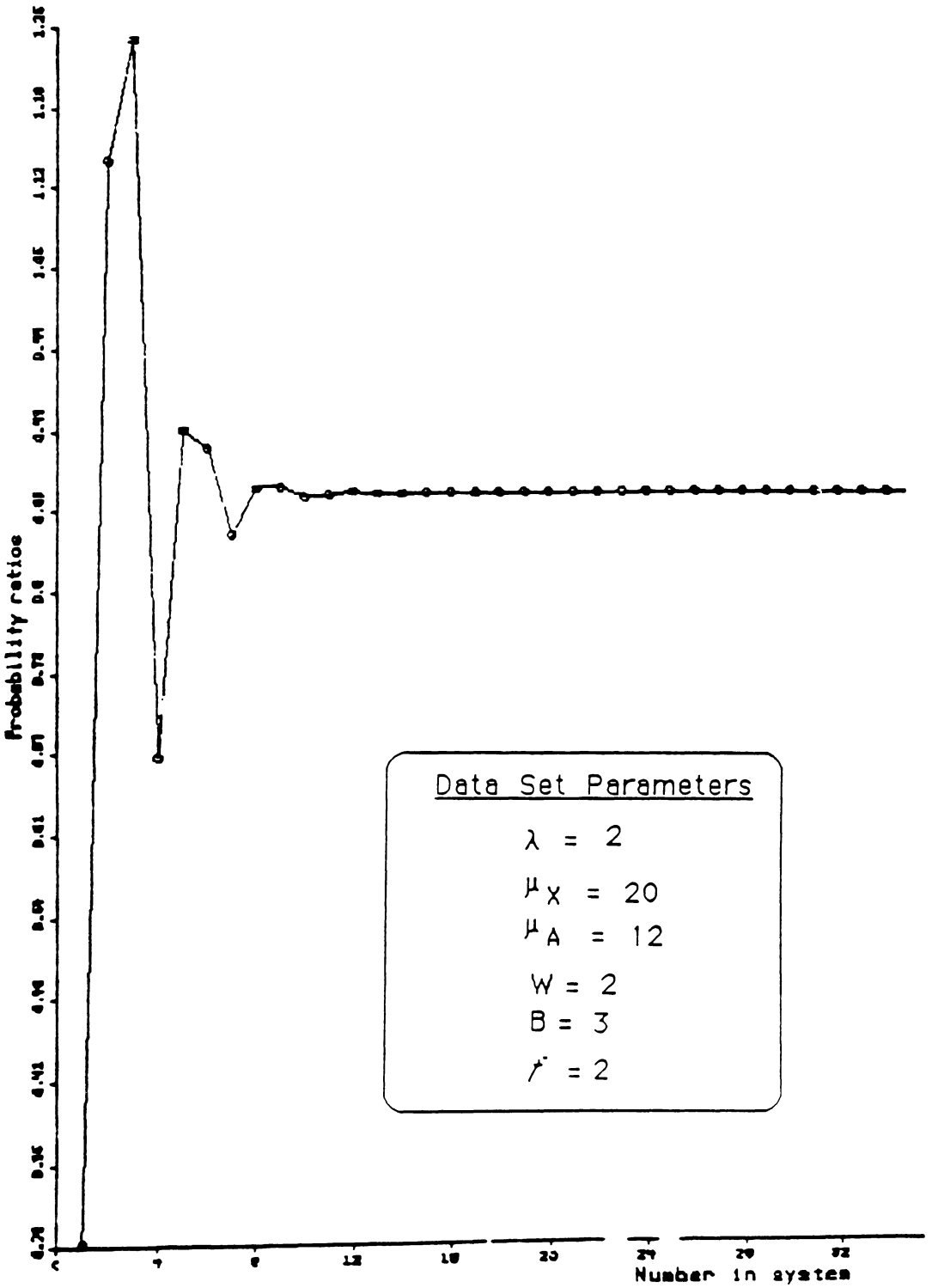


Figure 4.6 - State Probability Ratios

$$a_{N+B+1} = 1 - \frac{\lambda}{(\mu_X/f)} \frac{1-\gamma_\infty^B}{\gamma_\infty^B(1-\gamma_\infty)}. \tag{4.3.8}$$

We see that the assumption (4.3.6) implies the existance of a_∞, since the right-hand side above is independent of N. We thus have one equation relating a_∞ and γ_∞

$$a_\infty = 1 - \frac{\lambda}{(\mu_X/f)} \frac{1-\gamma_\infty^B}{\gamma_\infty^B(1-\gamma_\infty)}. \tag{4.3.9}$$

This is identical to (3.3.10), except that a_∞ replaces b_∞, and we replace μ_X by μ_X/f.

As in the analysis of the previous model, we obtain another equation relating a_∞ and γ_∞ from an analysis of the closed network consisting of the transmission and acknowledgment paths. This network was shown in Figure 4.4. The rate diagram for this closed network is shown in Figure 4.7, for the case where $f=3$ and $W=4$. We take for the state of the closed system the pair (n_A, n_R). We denote the probability of state (j,k) by $y_{j,k}$.

Following the analysis of Chapter 3, we will modify the rate at which tokens enter the acknowledgment queue in the closed model in order to reflect more accurately the true dynamics of the system. Since the only service completions which result in a token entering the acknowledgment queue are those which occur when $n_R = f-1$, we will replace the rate μ_X at which transmission completions occur in these states by $\gamma_\infty\mu_X$, and leave the other service rates alone. This results in the modified rate diagram of Figure 4.8.

The probability balance equations for the model of Figure 4.8 are:

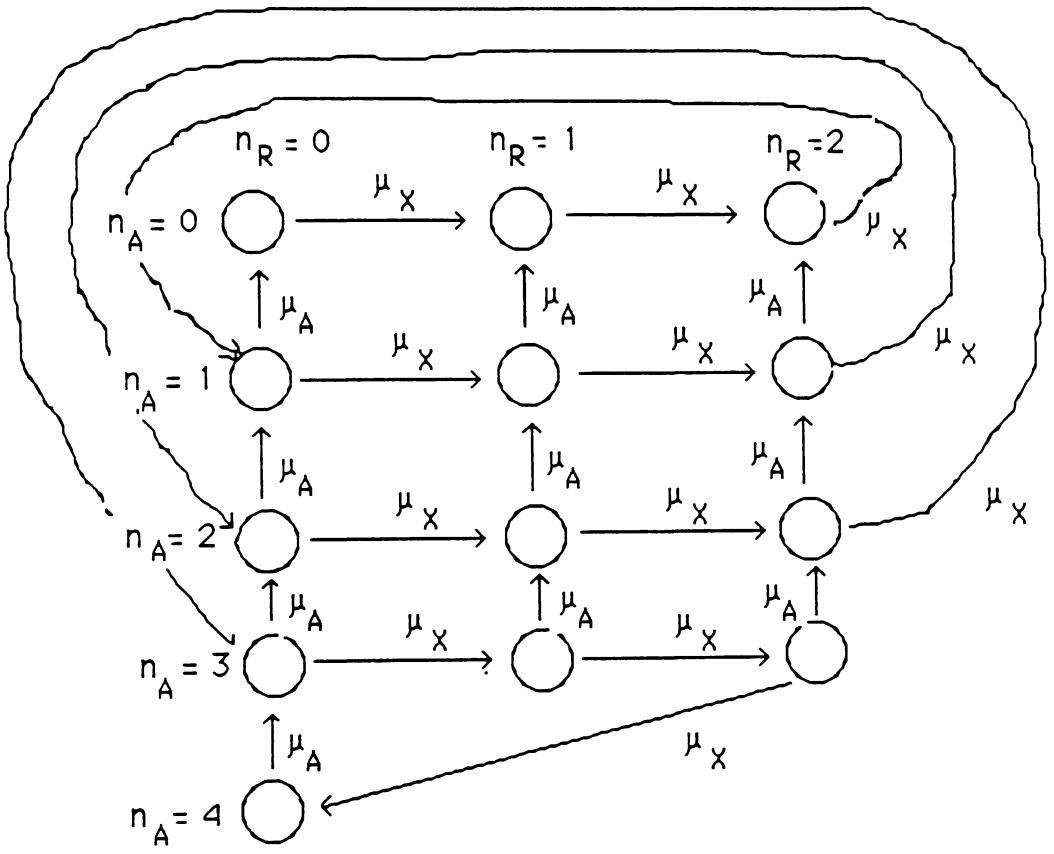
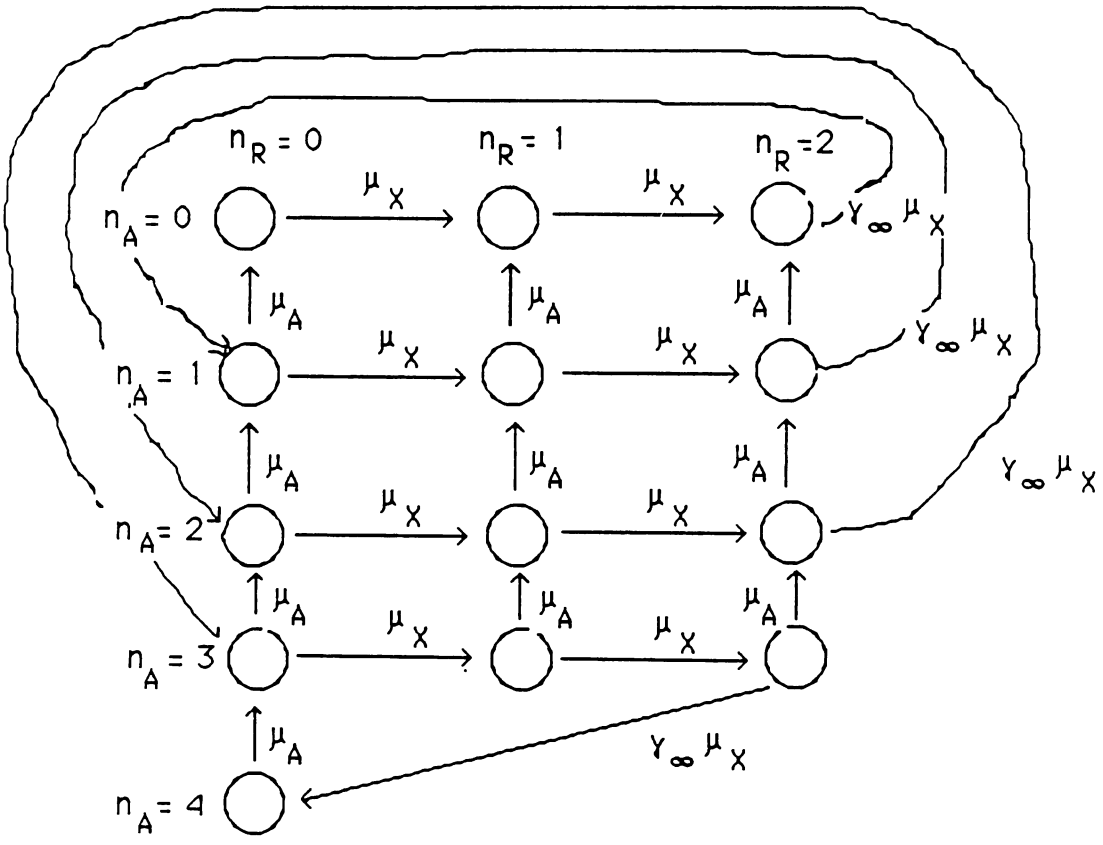


Figure 4.7 - Rate diagram of closed model



$W = 4$

$r = 3$

Figure 4.8 - Rate diagram with modified rates

$$\mathbf{n_A=W-1}\quad\begin{cases}\gamma_\infty\mu_X y_{W-1,f-1}=\mu_A y_{W,0}\\ \mu_X y_{W-1,f-2}=(\mu_A+\gamma_\infty\mu_X)y_{W-1,f-1}\\ \mu_X y_{W-1,k}=(\mu_A+\mu_X)y_{W-1,k+1}\,,\quad k=f-3,\ldots,0\end{cases}\tag{4.3.10}$$

$$\mathbf{n_A=W-2,\ldots,1}\quad\begin{cases}\gamma_\infty\mu_X y_{j,f-1}=\mu_A\sum_{k=0}^{f-1}y_{j+1,k}\\ \mu_X y_{j,f-2}=(\mu_A+\gamma_\infty\mu_X)y_{j,f-1}-\mu_A y_{j+1,f-1}\\ \mu_X y_{j,k}=(\mu_A+\mu_X)y_{j,k+1}-\mu_A y_{j+1,k+1}\,,\quad k=f-3,\ldots,0\end{cases}$$

$$\mathbf{n_A=0}\quad\begin{cases}\gamma_\infty\mu_X y_{0,f-1}=\mu_A\sum_{k=0}^{f-1}y_{1,k}\\ \mu_X y_{0,f-2}=\gamma_\infty\mu_X y_{0,f-1}-\mu_A y_{1,f-1}\\ \mu_X y_{0,k}=\mu_X y_{0,k+1}-\mu_A y_{1,k+1}\,,\quad k=f-3,\ldots,0\end{cases}$$

These equations can be solved by assuming an arbitrary positive value for $y_{W,0}$, solving for each $y_{j,k}$ in the order the equations are presented above, and then normalizing by dividing each probability by

$$\sum_{j=0}^{W}\sum_{k=0}^{f-1}y_{j,k}$$

The estimate of a_∞ from the solution to these equations is

$$a_\infty = 1-y_{W,0}$$

using the normalized value of $y_{W,0}$.

We note that the throughput of the closed network represented by Figure 4.8 is $\mu_X(1-\sum_{j=0}^{W-1}y_{j,k})$, for any k. If the equations (4.3.10) are solved with $\gamma_\infty = 1$, then the throughput value obtained is the maximum throughput of the model of Figure 4.3.

An alternative to solving equations (4.3.10) recursively as described above is to note that the closed network of Figure 4.4, on which the rate diagram in Figure 4.8 and equations (4.3.10) are based, can be cast as a Coxian server with a state dependent arrival rate. Marie, [Mari80], has developed an algorithm for solving for the state probabilities of such a queue. This algorithm is likely to have better stability behavior than the recursive solution.

Let us show how the network of Figure 4.4 can be described as a Coxian server with state dependent arrival rates. Let $n_C = W - n_A$. n_C is the number of high level packets in the transmission queue, (including the high level packet in service). Arrivals to the transmission queue occur at rate μ_A whenever the acknowledgment queue is not empty, thus the state dependent arrival process to the transmission queue, denoted by $\lambda(n_C)$ is given by

$$\lambda(n_C) = \begin{cases} \mu_A , & n_C = 0,1,\cdots,W-1 \\ 0 , & n_C \geq W \end{cases} . \qquad (4.3.11)$$

The transmission server can be thought of as a Coxian-f server of high level packets. A high level packet beginning service at the transmission server requires f sequential exponential service phases before completing service. In our model with modified rates, (i.e. the model in Figure 4.8), the first $f-1$ service phases have rate μ_X and the last phase has rate $\gamma_\infty \mu_X$. $y_{W,0}$, the probability of having all tokens in the acknowledgment queue in the closed model, is equal to the probability that $n_C = 0$ in the Coxian server with state dependent arrival rate (4.3.11). This probability can be calculated from the algorithm in [Mari80].

Although this means of calculating $y_{W,0}$ may be more numerically stable than the recursive solution of (4.3.10), it was not implemented for this thesis.

Using the solution to (4.3.10) together with (4.3.9) gives us two equations relating a_∞ and γ_∞. We estimate $\bar{a}$ with an algorithm based on these two equations and modelled after Algorithm 2. Our estimate of $\bar{a}$ is calculated from:

Algorithm 4

1) Let γ_∞=1. Solve (4.3.10) for a_∞.

2) Use the value of a_∞ from 1) to calculate a new γ_∞, using (4.3.9).

3) Using γ_∞ from 2), calculate a new a_∞ from the closed network equations (4.3.10). Use this value of a_∞ for $\bar{a}$.

At this point, we followed the paradigm of Chapter 3, and used a single server queue with exponential service rate $\frac{\mu_X}{f}(1-\bar{a})$ to represent the sliding window flow controlled network of Figure 4.3. The results were quite dissapointing, with predicted mean number in queue errors exceeding the actual values by over 20% in many cases. The situation was greatly improved by considering more closely the service behavior of the transmission channel in Figure 4.3, and modifying the approximate server appropriately.

The model of Figure 4.3 has a transmission mechanism which must complete f exponentially distributed transmissions to deliver a high level packet. The transmission time for the high level packet thus has an Erlang-f distribution. We know from the Pollaczek-Khinchin formula, (see [Klei75]), that if two queues have the same

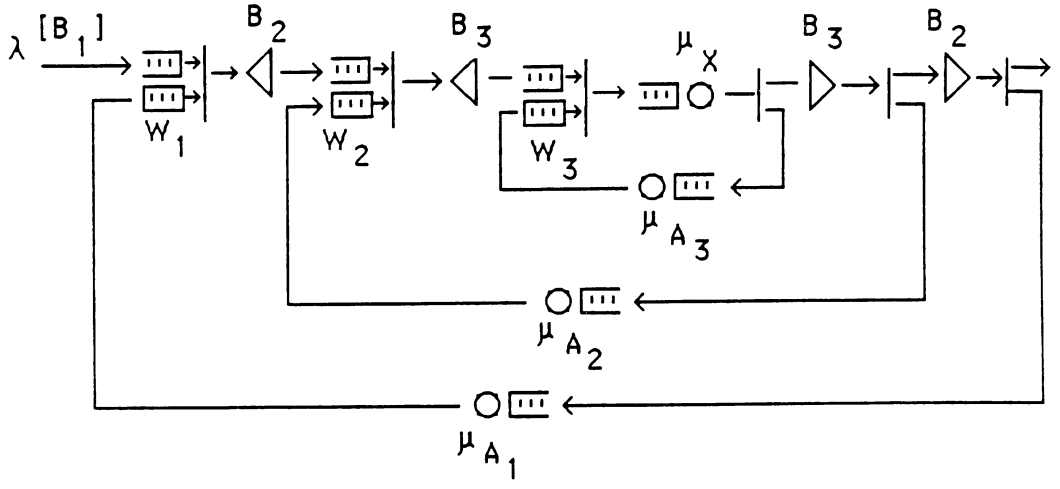
Poisson arrival pattern and same mean service time, the queue with the larger variance in the service time distribution will have the longer average queue length. Since the exponential distribution used to approximate the transmission behavior has more variance than an Erlang-f distribution (for $f>1$) perhaps this is the source of the errors above.

We repeated the experiments, using an Erlang-h server with mean service rate $\frac{\mu_X}{f}(1-\bar{a})$, (the same as in the exponential server case) to represent the transmission network, and the results were a vast improvement. The results are presented in Section 4.5.

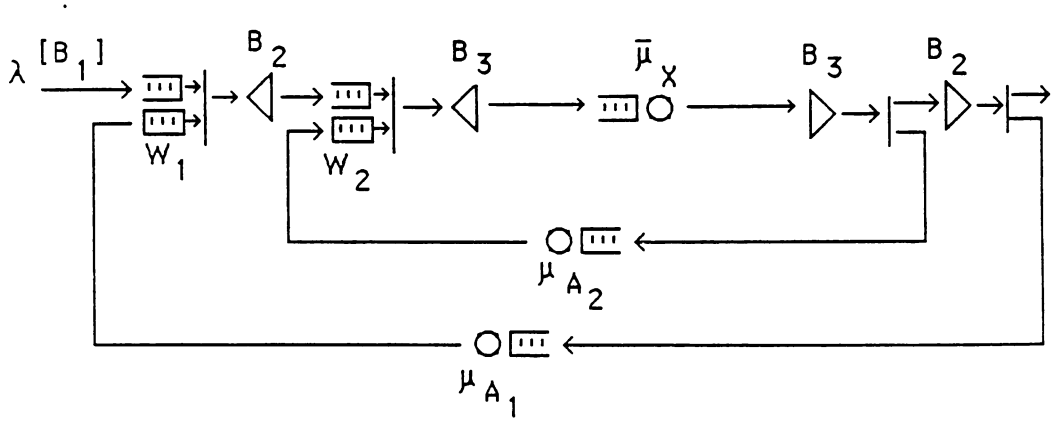
We wish to note here how the number of stages, h, in the approximate Erlang server were determined. If f was 2 or 3, $h = 2$ was used. This choice is based on comparison of experiments with $h=1$, 2, and 3 in the approximation algorithm against exact solutions computed by Neuts' matrix-geometric procedure. For any $f>3$, we used $h=3$. Since the sliding window flow control mechanism introduces variability into the effective delivery time distribution beyond that of the transmission time distribution, we use less than f stages in our approximate model, since the Eralng-k family of distributions has increasing variance as k decreases. The exception to this rule is when $f=2$. In this case the Erlang-2 distribution and the exponential, or Erlang-1, gave comaprable results. We limited the number of stages in our approximate model to three, since the Erlang-k distributions become less distinct as k increases. We hoped that using no more than three stages would suffice.

Limiting the number of stages in our approximate Erlang server also helps to keep the computational burden of the approximation scheme down when more than two nested levels of sliding window flow control are analyzed. Consider the network of Figure 4.9(a). This network has three layers of sliding window flow control. Analyzing the performance of this network hierarchically by our scheme, we would first reduce the lowest level to a single exponential server. We use Algorithm 2 to compute the service rate of this server, analyzing the lowest level model with arrivals at rate $\lambda B_1 B_2$ in batches of size B_3. After this step we have the approximate network of Figure 4.9(b). Now we use Algorithm 4 to reduce the lowest level of flow control in this network, using an arrival process of rate λB_1 with batches of size B_2. This results in the network of Figure 4.9(c), where the Erlang-h server has $h = 2$ or 3, depending upon the magnitude of B_3.

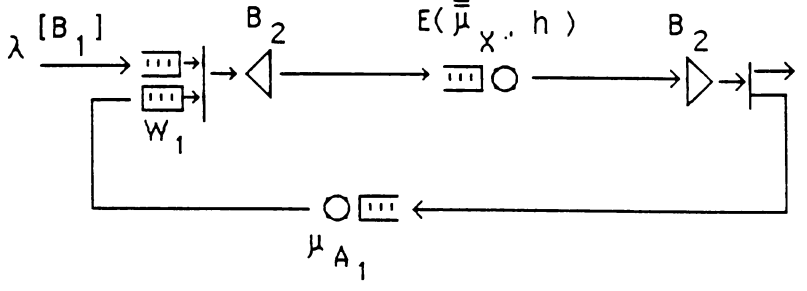
The network of Figure 4.9(c) appears to be a model which we have not yet considered. It differs from the model of Figure 4.3 by having an Erlang rather than an exponential server. From a modelling point of view, however, these two networks have the same structure. In the model of Figure 4.3 f exponential services must be completed to deliver a high level packet. In Figure 4.9(c), B_2 Erlang services must be completed to deliver a high level packet. Since one Erlang-h service is composed of the sum of h exponential random variables, delivery of a high level packet in the model of Figure 4.9(c) takes hB_2 exponential delays; thus the model of Figure 4.9(c) is equivalent to the model of Figure 4.3 when $f = hB_2$. Casting the problem in this form, we can reduce the network of Figure 4.9(c) to a single queue using Algorithm



(a) - Original model



(b) - lowest level reduced



(c) - two lowest levels reduced

Figure 4.9 - Three levels of flow control

4.

We now see why keeping the number of Erlang stages in our approximation server small is desirable. The effort to analyze (4.3.10) for Figure 4.9(c) is $O(hW_1B_2)$.

4.4. A Better Algorithm for a_∞

As in Chapter 3, better estimates of a_∞ may be obtained by modifying the rate structure of the closed network to more closely reflect the state asymptotic behavior. While we do not present a proof here, as in Chapter 3, that the value of a_∞ from the "better" procedure is larger than $\bar{a}$, this was observed in all cases. We present the better algorithm as interesting in its own right.

Referring to Figure 4.7, let S_k denote the aggregate state $\bigcup_{j=0}^{W}(j,k)$, $k=0,1,\cdots,f-1$, and let Y_k denote the probability of this state. From the rate diagram it is clear that for any pair l and k, $Y_l=Y_k$, since the aggregate rate equations are

$$\mu_X Y_k=\mu_X Y_{k+1},\ k=0,1,\cdots,f-2.$$

This property is not true however in the actual system (Figure 4.3). By summing the rate equations (4.3.1) over j for $i > B$, we get

$$(\lambda+\mu_X)\sum_{j=0}^{W-1} p_{i,j,k}=\mu_X\sum_{j=0}^{W-1} p_{i,j,k-1}+\lambda\sum_{j=0}^{W-1} p_{i-B,j,k},\ k=1,2,\cdots,f-1. \qquad (4.4.1)$$

If we assume that for i large the conditional probability of having k transmission packets in the reassembly buffer reaches a state asymptotic constant for each k, we

have

$$\frac{\sum_{j=0}^{W-1} p_{i,j,k}}{P_i} = \frac{\sum_{j=0}^{W-1} p_{i+1,j,k}}{P_{i+1}}. \tag{4.4.2}$$

This property has been observed in all examples. Figure 4.10 gives a typical plot. As in the previous model, the existence of such state asymptotic values is due to the regular structure of the rate equations for i large.

Manipulating (4.4.2) and using assumption (4.3.5) we have

$$\frac{P_{i+1}}{P_i} = \frac{\sum_{j=0}^{W-1} p_{i+1,j,k}}{\sum_{j=0}^{W-1} p_{i,j,k}} = \gamma_\infty.$$

Thus,

$$\sum_{j=0}^{W-1} p_{i,j,k} = \gamma_\infty{}^{B} \sum_{j=0}^{W-1} p_{i-B,j,k}. \tag{4.4.3}$$

Using (4.4.3), (4.4.1) becomes

$$\sum_{j=0}^{W-1} p_{i,j,k} = \frac{\mu_X}{\lambda(1-\gamma_\infty^{-B})+\mu_X} \sum_{j=0}^{W-1} p_{i,j,k-1}$$

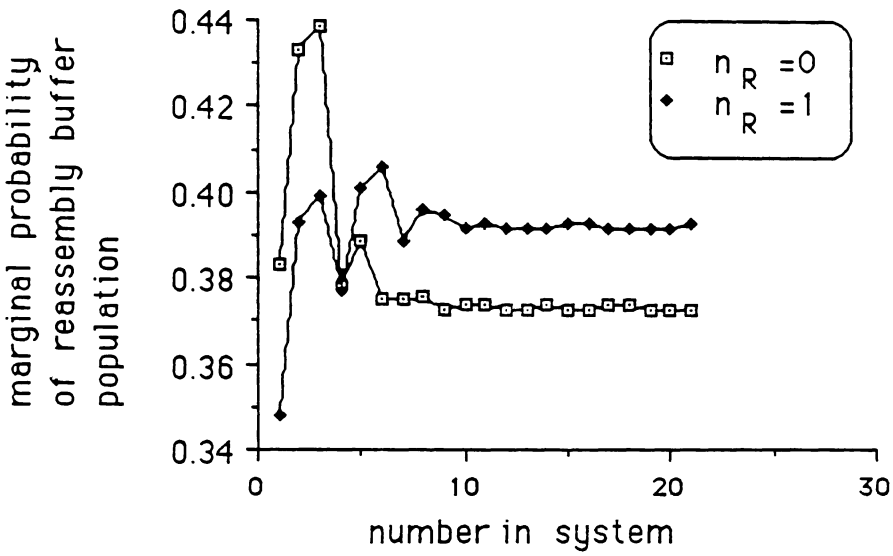
and as $1-\gamma_\infty^{-B}<0$ for $\gamma_\infty<1$,

$$\sum_{j=0}^{W-1} p_{i,j,k} > \sum_{j=0}^{W-1} p_{i,j,k-1}$$

for i large. We see then that in the actual system

$$\Pr\{n_R=f-1\} = \xi\Pr\{n_R=f-2\} = \xi^2\Pr\{n_R=f-3\} = \cdots = \xi^{f-1}\Pr\{n_R=0\}$$

where $\xi=\mu_X/(\lambda(1-\gamma_\infty{}^{-B})+\mu_X) < 1$. We can make the closed system have this property by replacing the rate μ_X at which we leave S_k by μ_X/ξ^k. This results in the rate



Data Set Parameters

λ = 2.0

μ_X = 20.0

μ_A = 12.0

W = 2

B = 3

r = 2

Figure 4.10 - Marginal Probabilities of Reassembly Buffer Population

diagram in Figure 4.11. The balance equations among states S_k in this model are

$$\frac{\mu_X}{\xi^k} Y_k = \frac{\mu_X}{\xi^{k+1}} Y_{k+1}$$

or

$$Y_{k+1} = \xi Y_k .$$

Let $\bar{y}_{j,k}$ denote the probability of state (j,k) in the new closed model. We can solve for each of the $\bar{y}_{j,k}$ in terms of $\bar{y}_{W,0}$ recursively, much as we solved (4.3.10). The rate equations for the closed network of Figure 4.11, in the order which they are easily solved, are:

$$\bar{y}_{W-1,f-1} = \xi^{f-1} \mu_A / \mu_X \bar{y}_{W,0}$$

$$\bar{y}_{W-1,k} = (\xi^k \mu_A / \mu_X + \xi^{-1}) \bar{y}_{W-1,k+1}, \quad k = f-2, f-3, \cdots, 1, 0.$$

For each $j = W-2, W-3, \cdots, 2, 1$ we solve

$$\bar{y}_{j,f-1} = \xi^{f-1} \mu_A / \mu_X \sum_{k=0}^{f-1} \bar{y}_{j+1,k}$$

$$\bar{y}_{j,k} = (\xi^k \mu_A / \mu_X + \xi^{-1}) \bar{y}_{j,k+1} - \xi^k \mu_A / \mu_X \bar{y}_{j+1,k+1}, \quad k = f-2, f-3, \cdots, 1, 0$$

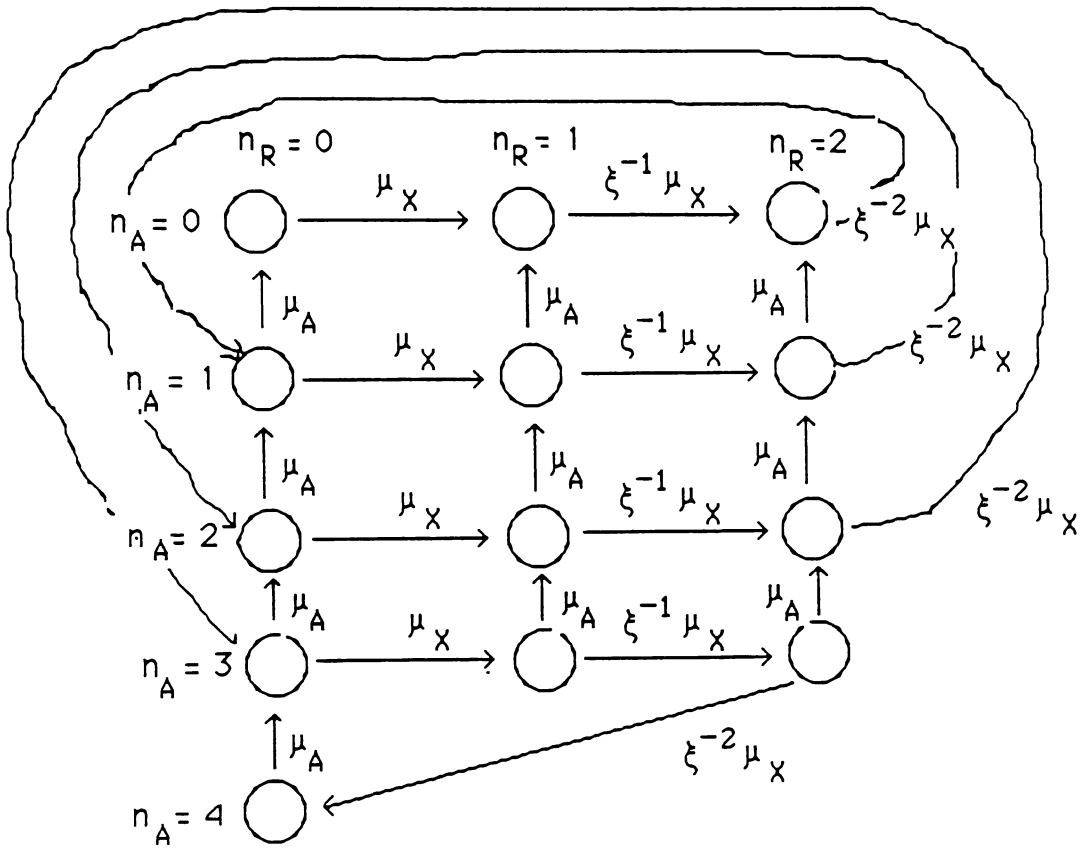
and for $j = 0$ we solve

$$\bar{y}_{0,f-1} = \xi^{f-1} \mu_A / \mu_X \sum_{k=0}^{f-1} \bar{y}_{2,k}$$

$$\bar{y}_{0,k} = \xi^{-1} \bar{y}_{0,k+1} - \xi^k \mu_A / \mu_X \bar{y}_{1,k+1}, \quad k = f-2, f-3, \cdots, 1, 0.$$

We normalize the probabilities by summing them and dividing each $\bar{y}_{j,k}$ by that sum. a_∞ is estimated by $1 - \bar{y}_{W,0}$. In Table A.15 we present results of this procedure, using a similar methodology to Table A.7.

As discussed in Section 4.3, the algorithm of Marie in [Mari80] can be used as an alternative means of solving for $\bar{y}_{W,0}$.



$W = 4$

$r = 3$

Figure 4.11 - Rate diagram for a_∞

4.5. Approximation Results

We evaluated Algorithm 4 for 32 data sets of networks with the structure of Figure 4.3. The parameters for these data sets are shown in Table A.12 in Appendix A. In each case the approximate results were compared to exact solutions obtained numerically. The numerical procedure for these models required working with matrices of the order of fBW, so we were limited in the range of parameters we could test. Within these limits, we varied the system utilization, the window size, f, and the ratio of high level packet transmission time to acknowledgment transmission time, $\mu_X/f\,\mu_A$.

Results for the prediction error of the mean number of high level packets in the system are shown in Table A.13. The errors are fairly low. In Figure 4.12 we show the error in mean number vs. the predicted probability of having the acknowledgment queue full. (The probability of having the acknowledgment queue full is a measure of how far from a true Erlang distribution the transmission time distribution will be). The implications of this Figure are not as clear as in the corresponding figure in Chapter 3. In addition to the errors introduced by not having an Erlang transmission distribution, there are also errors due to approximating a system with $f>5$ by an Erlang-3. Figure 4.13 shows the error in predicted mean vs. probability of having the acknowledgment queue full for those data sets where $f=2$. (Due to the limitations of the numerical procedure this is the only value of f for which we have many data points). In this case, we see that the errors are roughly correlated with the probability of having the acknowledgment queue full.

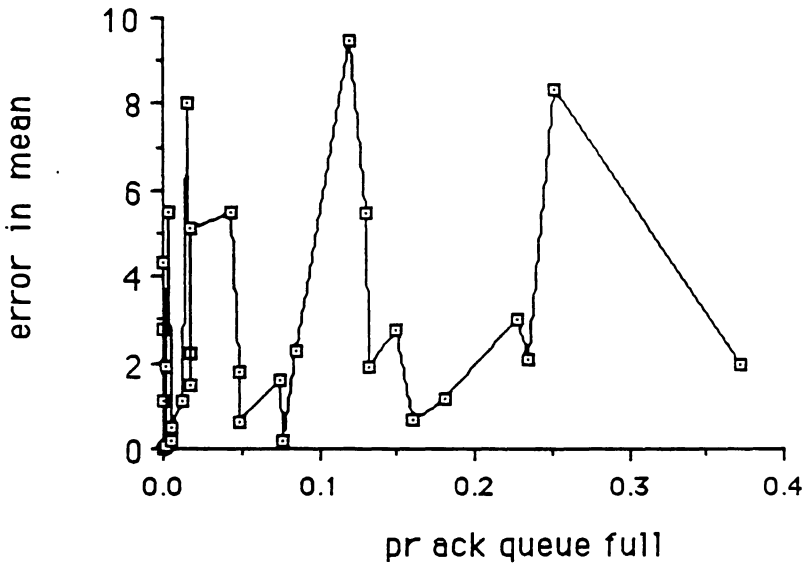


Figure 4.12 - Errors in mean prediction

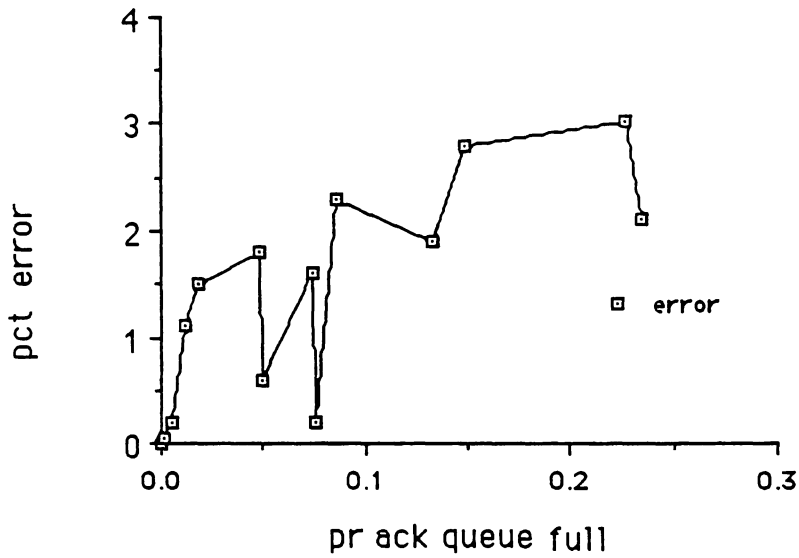


Figure 4.13 - Errors in mean prediction for f=2

While errors due to Erlang "fit" problems cloud the picture, it is still true that for a given f, high utilization, high ratio $\mu_X/f\,\mu_A$, and small window size will be the cases with the worst performance.

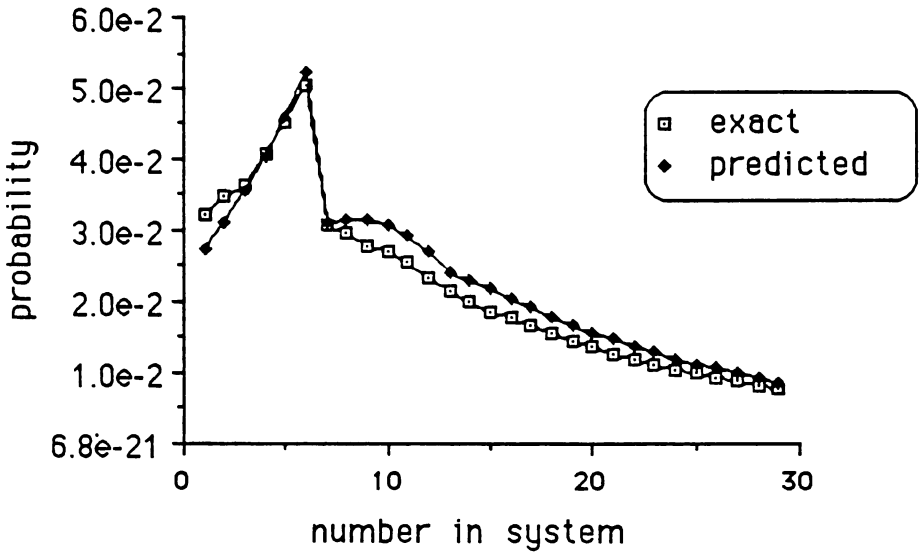
Table A.14 shows the maximum relative error and maximum deviation measures for the data sets of Table A.12. The results are fairly good, though not as good as in the exponential transmission time examples.

Figure 4.14 shows the predicted and actual P_i probabilities for data set 54, which was one of the poorest predictions.

4.6. Multi-layer Simulation vs. Approximation Results

In the previous Section we compared the approximation algorithm (Algorithm 4) results to exact results for a model with the structure of Figure 4.2. Recall that the transmission queue in this model represents a sliding window flow controlled link which has been reduced using Algorithm 2, (see Figure 4.1). The comparisons of Algorithm 4 results to exact results in the previous Section assumes that the transmission queue is an *exact* representation of the lower level network, and thus allows us to evaluate Algorithm 4 apart from errors in the lower level approximation. In this Section we evaluate the performance of the hierarchical method, as described in Sections 4.1 and 4.3.

We performed the hierarchical reduction for 36 examples with the structure of Figure 4.1. In each case Algorithm 2 was used to reduce the lower level flow controlled link to a single queue, assuming arrivals at the rate λB_2. The resulting



Data Set Parameters

$\lambda = 1.0$

$\mu_X = 50.0$

$\mu_A = 8.0$

$W = 4$

$B = 6$

$r = 5$

Figure 4.14 - Predicted vs. actual for data set 54

approximation queue was then used as the transmission queue in a network with the structure of Figure 4.2, and this latter network was analyzed by Algorithm 4.

The data set parameters used are shown in Table A.16 in Appendix A. We attempted to use enough data sets to examine the various possible relationships between W_1, W_2, B_1, and B_2. The lower level models in these examples are taken from data sets previously examined in Chapter 3. The data set number of the lower level model used is indicated in Table A.16. We note that data sets 1c, 1d, 2c, and 2d use the same lower level parameters as data set 12, except that the window size is increased to 16.

Due to the large state space associated with an exact probabilistic model of Figure 4.1, we are not able to generate exact solutions against which to compare the approximation results. We thus constructed a simulation model of the network in Figure 4.1, with the rates, window sizes, and amount of fragmentation as parameters. We use the results from the simulation model as a basis of comparison for the approximation results. For each data set in Table A.16, a number of independent replications of the simulation were run. That number is shown in Table A.17. Each replication began with no packets in the system, and all tokens at both levels in the token queue. Each replication was terminated after 500 high level packets were delivered. We calculated the mean number of high level packets in the system for each replication, and from these values calculated an approximate 95% confidence interval for the mean number of high level packets in the system, using a value from the t-distribution table.

The comparison of the simulation mean number of high level packets in the system and the predicted mean number from the hierarchical approximation is shown in Table A.17. We see that in most cases the approximation mean exceeds the simulation mean, but with the exception of cases 3a, 3c, 3f, 3h, 4a, and 4c, the approximate means lie within the 95% confidence intervals. Over-estimation of the mean number in the system, (as long as it is not severe), is generally considered a desirable result, as it provides a "conservative" estimate of the true value.

We also show in Table A.17 the average number of "busy periods" in each of the six simulation runs. A busy period begins when a packet arrives to a system with all tokens in the token queues, and ends the next time this state is reached. Since the number of busy periods is fairly large in most cases, we would expect that the confidence intervals are close to the true 95% confidence intervals for the mean.

We also used the simulation data to construct point estimates of the probability of having any given number of high level packets in the system, along with the associated approximate 95% confidence intervals for these estimates. Figures 4.15 and 4.16 show comparisons of the estimate of the probability distribution of high level packets in the system thus obtained against the predicted probability distribution from the approximate queue generated by Algorithm 4. Figure 4.15 is for case 2b, and Figure 4.16 is for case 4m. These two plots were selected as being typical.

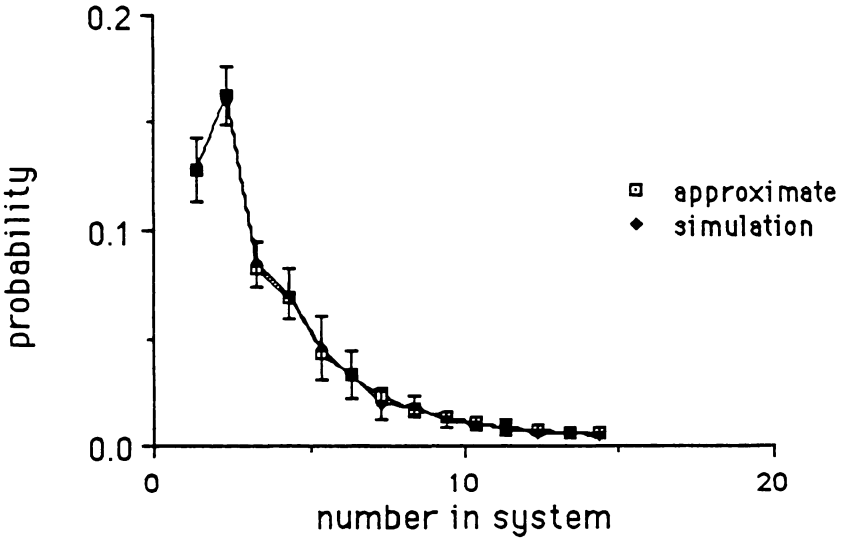


Figure 4.15 - Comparison of simulation and approximation distributions (2b)

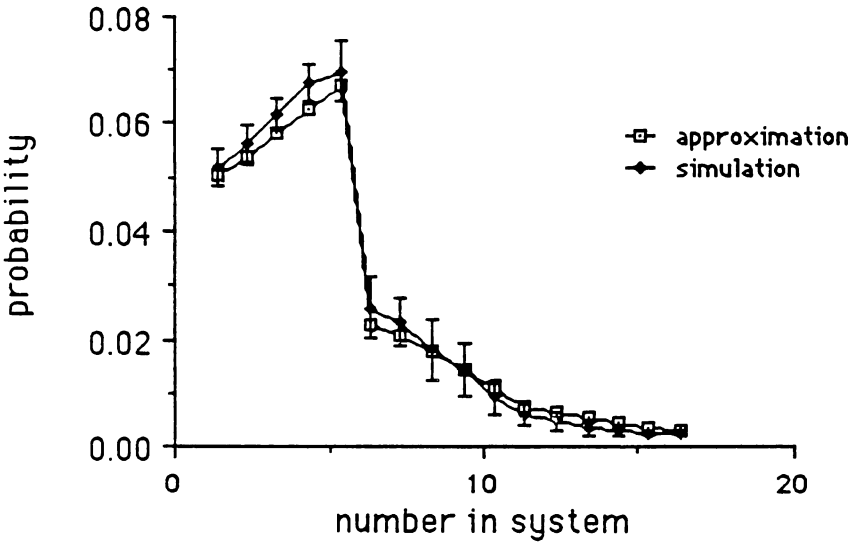


Figure 4.16 - Comparison of simulation and approximation distributions (4m)

CHAPTER 5

CONCLUSIONS AND SUGGESTIONS FOR FUTURE RESEARCH

In this thesis we have presented a new procedure for approximating the behavior of a single hop network with multiple layers of sliding window flow control. The method differs from previous research presented in the published literature in that it explicitly takes into account the fragmentation of packets which occurs in actual systems. We have shown that the method is better than existing procedures for cases where there is no packet fragmentation, and much better in models where there is fragmentation. The procedure was compared to exact numerical solutions, and found to perform well.

The procedure is easy to implement, and easy to solve on a computer. Its speed makes it feasible to think of including the procedure in an interactive network designer expert system, as a module to evaluate a proposed network which has sliding window flow control.

The procedure reports its results as the service rate and number of exponential service stages of a queue. This should make it a useful add-on to general purpose queuing network solution software packages. These packages typically cannot handle the special features of a sliding window flow control. By reducing the portions of the queueing network which have sliding window flow control to a single queue, the procedure approximates the flow controlled sub-net by a structure which can be handled by the software package.

The analysis uses the state behavior of the network in high numbered states in order to predict its behavior over all states. The behavior in high numbered states, referred to as the state asymptotic properties of the system, is relatively easy to analyze. Although we do not expect to see these states occurring often in actual practice, given the structure of the system, its behavior in low numbered states will not be independent of its asymptotic behavior.

We see particularly that for systems with batched arrivals that the state asymptotic approach yields better results than procedures based on the Norton's approximation. The Norton's approximation technique assumes an "instantaneous steady state" behavior of the network, that is, the network with a given population behaves like a closed network in steady state with that population. This technique is successful in the models where the state does not change dramatically in any one instantaneous transition. This is clearly not the case in systems with batched arrivals.

This thesis is a contribution towards analyzing sliding window flow controlled networks, but it is by no means the last word on this subject. Let us look at some extensions to the analysis presented here which would be useful.

The current procedure provides an approximate solution to the performance measures of a sliding window flow controlled channel, but it does not provide any estimate of how well it has done. The experiments which have been done provide some general guidelines as to when the solutions can be expected to be good, and when they may be not so good. A useful addition to this analysis would be derivation of bounds on the performance characteristics of the network.

Extension of the analysis to a random number of batches arriving from the session layer would allow us to model communication links with more general traffic characteristics. The probability distribution of the batch arrival size to the transport layer would describe the distribution of session layer message sizes. A large session layer message would cause a large batch arrival to the transport layer, while a small message would have a batch size of one. Our current analysis is restricted to fixed session layer message sizes, and thus a fixed batch arrival size to the transport layer.

Another extension which would prove useful would be to analyze the network with a transmission server with a Coxian distribution. We already can handle the cases of an exponential and Erlang transmission times. The family of Coxian distributions is capable of representing a wider range of transmission characteristics than the exponential and Erlang servers. Since the Coxian server is based on a weighted sum of exponential delays, it is usually a tractable modelling device.

Even if we continue to restrict ourselves to an exponential delivery time, the Coxian distribution would seem to be particularly appropriate in the cases in which the exponential service approximation performs the worst, i.e., when the acknowledgment queue is frequently full. In these cases the packet at the head of the holding queue when the acknowledgment queue becomes full will have two exponentially distributed delays before arriving at the sending station. A Coxian service distribution, with one phase at rate μ_X and a second phase at rate μ_A weighted by the probability of finding the acknowledgment queue full, should be a good modelling approach.

The Coxian server should also be useful in analysis of multi-hop connections. The work by Varghese, Chou, and Nilsson [VaChNi83] is already a step in this direction. In this work the characteristics of an N-hop transmission link is captured in a state dependent server with Coxian service when $n_S > W$. The authors of this paper mention that the approximation is numerically demanding, particularly in evaluating the parameters of the Coxian service; nonetheless, the indication that the Coxian distribution is better than Norton's approximation for multi-hop networks warrants further investigation.

Another feature of sliding window flow control which would be useful to analyze is multiple token acknowledgments. These are a common feature in actual networks. This problem may be particularly tricky, since the number of tokens in an acknowledgment depends upon the number of data packets which have arrived between acknowledgment releases.

Another feature, which is likely even more demanding to model, is non-Poisson arrivals. The arrival process to a lower level model is not likely to be Poisson. Using a more accurate representation of the arrival process will make the analysis of nested levels of flow control more accurate. Determining the characteristics of the arrival process to the lower layers of a flow controlled link, given the characteristics of the external arrival process, would be an interesting study.

Finally, we note that sliding window flow control has a structural parallel to kan-ban type manufacturing systems. In these systems, there are a limited number of inventory units between each production stage. When a unit is consumed by a stage,

the downstream stage is signalled to replace it. If all inventory units for a stage are being replenished, that stage is forced to stop working. The inventory units thus resemble tokens in sliding window flow control. It will be interesting to see how the results developed in this thesis for sliding window flow control can be applied to kan-ban systems.

REFERENCES

[AgrBuz83] Agrawal, S.C., and Buzen, J.P., "The Aggregate Server Method for Analyzing Serialization Delays in Computer Systems", *ACM Transactions on Computer Systems,* vol. 1, no. 2, 1983.

[AviHey73] Avi-Itzhak, B., and Heyman, D.P., "Approximate Queueing Models for Multiprogramming Computer Systems", *Operations Research,* vol. 21, no. 6, 1973.

[BCMP75] Baskett, F., Chandy, K.M., Muntz, R.R., and Palacios, F.G., "Open, Closed, and Mixed Networks of Queues with Different Classes of Customers", *Journal of the ACM,* vol. 22, no. 2, 1975.

[ChHeWo75] Chandy, K.M., Herzog, U., and Woo, L.S., "Parametric Analysis of Queueing Networks", *IBM Journal of Research and Development,* vol. 19, no. 1, 1975.

[ChGeVe77] Chaterjee, A., Georganas, N.D., and Verma, P.K., "Analysis of a Packet Switched Network with End to End Congestion Control and Random Routing", *IEEE Transactions on Communications,* vol. 25, no. 12, December 1977.

[Dall87], Dallery, Y., "Approximate Analysis of General Open Queueing Networks with Restricted Capacity", unpublished manuscript.

[FdPeWi87], Fdida, S., Perros, H., and Wilk, A., "Semaphore Queues: Modelling Multi-layered Window Flow Controls Mechanisms", unpublished manuscript.

[FreBex83] Freund, D.J., and Bexfield, J.N., "A New Aggregation Approximation Procedure for Solving Closed Queueing Networks with Simultaneous Resource Possession", in *Proceedings of the ACM SIGMETRICS Conference,* Minneapolis, August 1983.

[GerKle80] Gerla, M. and Kleinrock, L., "Flow Control: A Comparative Survey", *IEEE Transactions on Communications,* vol. 28, no. 4, April 1980.

[GihKue85] Gihr, O. and Kuehn, P.J., "Comparison of Communication Services with Connection-Oriented and Connectionless Data Transmission", *Proceedings of the International Seminar on Computer Networking and Performance Evaluation,* Tokyo, September 18-20, 1985.

[GoTaHa83] Goto, K., Takahashi, Y., and Hasegawa, J., "An Approximate Analysis of Controlled Tandem Queues", in *Proceedings of the International Seminar on Modelling and Performance Evaluation Methodology,* Paris, 1983.

[JacLaz82] Jacobson, P.A., and Lazowska, E.D., "Analyzing Queueing Networks with Simultaneous Resource Possession", *Communications of the ACM,* vol. 25, no. 2, February, 1982.

[JacLaz83] Jacobson, P.A., and Lazowska, E.D., "A Reduction Technique for Evaluating Queueing Networks with Serialization Delays", in *Performance '83,* Agrawala and Tripathi, eds, North Holland, New York, NY, 1983.

[Klei75] Kleinrock, L., *Queueing Theory, Volume 1,* John Wiley and Sons, New York, 1975.

[KleKer80] Kleinrock, L. and Kermani, P., "Static Flow Control in Store and Forward Computer Networks", *IEEE Transactions on Communications,* vol. 28, no. 2, February 1980.

[LabPuj79] Labetoulle, J., and Pujolle, G., "Modelling and Performance Evaluation of the Protocol HDLC", *Flow Control in Computer Networks,* J.-L. Grange and M.Gien, eds, North Holland, New York, NY, 1979.

[LaPuMi79] Labetoulle, J., Pujolle, G., and Mikou, N., "A Study of Flows in an X25 Environment", *Flow Control in Computer Networks,* J.-L. Grange and M.Gien, eds, North Holland, New York, NY, 1979.

[Lam77] Lam, S., "Queueing Networks with Population Size Constraint", *IBM Journal of Research and Development,* vol. 21, no. 4, 1977.

[Mari79], Marie, R., "An Approximate Analytical Method for General Queueing Networks", *IEEE Transactions on Software Engineering,* vol. 5, no. 5, September 1979.

[Mari80] Marie, R., "Calculating Equilibrium Probabilities for $\lambda(n)/C_k/1/N$ Queues", in *Proceedings of Performance '80,* Toronto, 1980.

[Neut81] Neuts, M., *Matrix-Geometric Solutions in Stochastic Models* , Johns Hopkins University Press, Baltimore, Md., 1981.

[PenSch75] Pennotti, M.C. and Schwartz, M., "Congestion Control in Store and Forward Tandem Links", *IEEE Transactions on Communications,* vol. 23, no. 12, December 1975.

[Perr81] Perros, H.G., "A Symmetrical Exponential Open Queue Network with

Blocking and Feedback", *IEEE Transactions on Software Engineering,* vol. 7, no. 4, 1981.

[Perr83] Perros, H.G., "A Two-Node Queueing Network with a Maximum Number of Allowable Jobs", in *Performance '83,* Agrawala and Tripathi, eds, North Holland, New York, NY, 1983.

[Reis79] Reiser, M., "A Queueing Network Analysis of Computer Communication Networks with Window Flow Control", *IEEE Transactions on Communications,* vol. 27, no. 8, August 1979.

[Reis81] Reiser, M., "Admission Delays on Virtual Routes with Window Flow Control", *Performance of Data Communications Systems and Their Applications,* G. Pujolle, ed., North Holland Publishing Co., New York, NY, 1981.

[Reis82] Reiser, M., "Performance Evaluation of Data Communication Systems", *Proceedings of the IEEE,* vol. 70, no. 2, 1982.

[Schw77] Schwartz, M., *Computer-Communication Network Design and Analysis,* Prentice-Hall, Inc, Englewood Cliffs, New Jersey, 1977.

[Schw82], Schwartz, M., "Performance Analysis of the SNA Virtual Route Pacing Control", *IEEE Transactions on Communications,* vol. 30, no. 1, January 1982.

[Tane88] Tanenbaum, A. S., *Computer Networks,* second edition, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1988.

[Thom83] Thomassian, A., "Queueing Network Models to Estimate Serializa-

tion Delays in Computer Systems", in *Performance '83*, Agrawala and Tripathi, eds, North Holland, New York, NY, 1983.

[ThoBay84a], Thomasian, A. and Bay, P., "Performance Analysis of Window Flow Control for Multiple Virtual Routes", *Proceedings IEEE INFOCOM '84*, San Francisco, California, April 9, 1984.

[ThoBay84b], Thomasian, A. and Bay, P., "Analysis of Queueing Network Models with Population Size Constraints and Delayed Blocked Customers", in *Proceedings of the 1984 ACM SIGMETRICS Conference on Measurement and Modelling of Computer Systems*, a special issue of *Performance Evaluation Review*, vol. 12, no. 3, 1984.

[VaChNi83], Varghese, G., Chou, W., and Nilsson, A.A., "Queueing Delays on Virtual Circuits using a Sliding Window Flow Control Scheme", *Proceedings ACM SIGMETRICS Conference*, Minneapolis, 1983.

APPENDIX A

TABLES

Table A.1 - Data Set Parameters ($B = 1$)

Data Set Number	λ/μ_X	System utilization	Window size	μ_X/μ_A size
1	.125	.125	16	.4
2	.813	.817	16	.8
3	.737	.774	16	.95
4	.750	.750	16	.1
5	.125	.126	16	.8
6	.158	.160	16	.95
7	.150	.150	16	.1
8	.875	.875	16	.4
9	.500	.503	16	.8
10	.500	.500	4	.1
11	.125	.127	4	.4
12	.526	.628	4	.95
13	.813	.925	4	.91
14	.737	.911	4	.95
15	.750	.750	4	.1
16	.900	.900	4	.1
17	.158	.188	4	.95
18	.150	.150	4	.1
19	.250	.285	4	.8
20	.500	.508	4	.4
21	.875	.889	4	.4
22	.800	.827	4	.5
23	.500	.569	4	.8
24	.500	.536	6	.8

Table A.2 - Data Set Parameters ($B > 1$)

Data Set Number	$\lambda B / \mu_X$ as % of maximum	Batch size	Window size	μ_X / μ_A
25	91%	3	3	.25
26	31%	3	3	.4
27	87%	3	3	.4
28	36%	3	3	.8
29	38%	3	3	.91
30	87%	3	3	.91
31	96%	3	3	.91
32	90%	8	3	.25
33	33%	8	3	.33
34	73%	8	3	.4
35	97%	8	3	.9
36	41%	8	3	.91
37	90%	8	3	.91
38	90%	3	8	.25
39	30%	3	8	.4
40	68%	3	8	.4
41	32%	3	8	.91
42	73%	3	8	.91
43	81%	3	8	.91
44	89%	8	8	.25
45	32%	8	8	.33
46	70%	8	8	.4
47	81%	8	8	.9
48	34%	8	8	.91
49	75%	8	8	.91
50	90%	3	20	.25
51	30%	3	20	.4
52	68%	3	20	.4
53	31%	3	20	.91
54	69%	3	20	.91
55	77%	3	20	.91
56	89%	8	20	.25
57	32%	8	20	.33
58	70%	8	20	.4
59	77%	8	20	.9
60	33%	8	20	.91
61	71%	8	20	.91

Table A.3 - Exact vs. Predicted Mean Number in System ($B = 1$)

Data Set Number	Exact Mean in System	Predicted Mean in System	Relative Error
1	0.14286	0.14286	0.0%
2	4.3460	4.3423	-0.0%
3	2.7774	2.7706	-0.2%
4	3.0000	3.0000	0.0%
5	0.14286	0.14286	0.0%
6	0.17857	0.17857	0.0%
7	0.17647	0.17647	0.0%
8	7.0000	7.0000	0.0%
9	1.0000	1.0000	0.0%
10	1.0000	1.0000	0.0%
11	0.14286	0.14286	0.0%
12	1.1898	1.1675	-1.9%
13	9.9725	9.3140	-6.6%
14	7.7390	6.8585	-11.4%
15	3.0004	3.0004	0.00%
16	9.0054	9.0054	0.00%
17	0.17876	0.17878	0.00%
18	0.17647	0.17647	0.00%
19	0.33439	0.33426	-0.00%
20	1.00326	1.0027	-0.00%
21	7.6665	7.63222	-0.4%
22	4.4170	4.3795	-0.8%
23	1.0593	1.0502	-0.9%
24	1.0090	1.0071	-0.2%

Table A.4 - Exact vs. Predicted Mean Number in System ($B > 1$)

Data Set Number	Exact Mean in System	Predicted Mean in System	Relative Error
25	2.011338e+01	2.008561e+01	0.14%
26	8.705217e-01	8.693995e-01	0.13%
27	4.530612e+00	4.510478e+00	0.44%
28	9.511480e-01	9.472546e-01	0.41%
29	9.92130e-01	9.88023e-01	0.41%
30	1.070386e+01	1.003069e+01	6.29%
31	3.855895e+01	3.548236e+01	7.98%
32	4.000568e+01	3.997672e+01	0.07%
33	2.166652e+00	2.162462e+00	0.19%
34	1.184953e+01	1.181387e+01	0.30%
35	9.950218e+01	9.532734e+01	4.20%
36	2.780606e+00	2.756917e+00	0.85%
37	3.513755e+01	3.381809e+01	3.76%
38	1.800140e+01	1.800138e+01	0.0001%
39	8.571508e-01	8.571503e-01	5.56e-05%
40	4.155155e+00	4.155070e+00	0.002%
41	8.60001e-01	8.612843e-01	-0.15%
42	4.674847e+00	4.631739e+00	0.92%
43	7.442701e+00	7.283353e+00	2.14%
44	3.600305e+01	3.600303e+01	5.30e-05%
45	2.117704e+00	2.117705e+00	-3.38e-05%
46	1.050800e+01	1.050788e+01	0.001%
47	1.829872e+01	1.809593e+01	1.11%
48	2.171922e+00	2.196188e+00	-1.12%
49	1.293453e+01	1.286309e+01	0.55%
50	1.800000e+01	1.800000e+01	2.12e-05%
51	8.571428e-01	8.571428e-01	0
52	4.153846e+00	4.153846e+00	0
53	8.57143e-01	8.57144e-01	-0.0001%
54	4.164457e+00	4.165776e+00	-0.03%
55	6.052628e+00	6.051785e+00	0.01%
56	3.599999e+01	3.599999e+01	-1.06e-05%
57	2.117647e+00	2.117647e+00	-1.13e-05%
58	1.050000e+01	1.050000e+01	0
59	1.422668e+01	1.425892e+01	-0.23%
60	2.117969e+00	2.119141e+00	-0.06%
61	1.066026e+01	1.069734e+01	-0.35%

Table A.5 - Predicted vs. Actual $\Pr\{n_S=i\}$, $B = 1$.

Data Set Number	Maximum deviation	Corresp. rel. error	Maximum rel.error *	Corresp. deviation
1	-3.7 e-09	-0.00%	-	-
2	6.6 e-05	0.04%	-0.4%	-4.4 e-06
3	-3.9 e-04	-0.15%	-3.3%	-3.5 e-05
4	-6.1 e-10	-0.00%	-	-
5	-1.7 e-10	-0.00%	-	-1.45 e-04
6	-1.2 e-10	0.00%	-	-
7	-1.8 e-09	0.00%	-	-
8	-3.2 e-10	0.00%	-	-
9	6.2 e-08	0.00%	-0.0002%	-3.9 e-09
10	1.4 e-07	0.00%	-0.0002%	-4.2 e-09
11	2.2 e-07	0.00%	-0.0007%	-1.1 e-08
12	3.7 e-03	1.52%	-22.6%	-2.8 e-04
13	3.7 e-03	4.87%	-24.0%	-2.5 e-04
14	7.6 e-03	9.85%	-45.0%	-4.5 e-04
15	5.5 e-07	0.00%	-0.0013%	-1.4 e-08
16	5.4 e-07	0.00%	-0.0022%	-2.6 e-08
17	2.4 e-05	0.02%	-0.0477%	-1.4 e-06
18	-1.1 e-09	-0.00%	-	-
19	2.6 e-05	0.01%	-0.29%	-8.6 e-06
20	1.1 e-04	0.02%	-0.22%	-4.4 e-06
21	3.4 e-04	0.33%	-1.6%	-1.8 e-05
22	8.2 e-04	0.54%	-1.6%	-1.7 e-04
23	1.2 e-03	0.47%	-7.4%	-9.4 e-05
24	1.9 e-04	0.76%	-1.6%	-1.6 e-05

* taken from states with actual probability greater than 0.001 .
- same data point as maximum deviation

Table A.6 - Predicted vs. Actual $\Pr\{n_S=i\}$, $B > 1$

Data Set Number	Maximum deviation	Corresp. rel. error	Maximum rel.error *	Corresp. deviation
25	1.15 e-04	0.13%	-0.34%	-4.69 e-06
26	3.68 e-04	0.05%	-0.65%	-7.63 e-06
27	6.72 e-04	0.22%	-2.01%	-1.69 e-05
28	1.59 e-03	1.82%	-5.47%	-9.04 e-05
29	2.45 e-03	2.77%	-8.03%	-1.45 e-04
30	4.51 e-03	6.35%	-24.02%	-2.21 e-03
31	2.00 e-03	7.35%	8.23%	1.76 e-03
32	1.95 e-04	0.19%	0.37%	8.89 e-05
33	1.36 e-03	0.20%	-1.22%	-3.67 e-04
34	1.28 e-03	0.47%	1.28%	5.79 e-04
35	9.23 e-04	8.44%	8.44%	9.23 e-04
36	6.29 e-03	1.02%	-8.89%	-1.00 e-03
37	2.11 e-03	8.52%	8.52%	2.11 e-04
38	1.27 e-07	0.00%	-0.00%	-4.89 e-09
39	2.98 e-07	0.00%	-0.00%	-5.24 e-09
40	3.04 e-06	0.00%	-0.01%	-6.80 e-08
41	-3.83 e-04	-0.06%	0.25%	3.23 e-05
42	-3.12 e-03	-1.02%	-9.74%	-9.65 e-05
43	-3.09 e-03	-1.41%	-13.87%	-1.27 e-04
44	8.20 e-08	0.00%	0.00%	6.71 e-08
45	5.36 e-07	0.00%	-0.00%	-3.19 e-07
46	3.55 e-06	0.01%	0.01%	3.55 e-06
47	-3.93 e-03	-1.93%	-5.60%	-5.60 e-05
48	-2.86 e-03	-0.42%	1.86%	9.15 e-05
49	-4.82 e-03	-1.83%	-5.25%	-5.19 e-05
50	-1.49 e-08	-0.00%	-0.00%	-1.49 e-08
51	0.00 e-99	0.00%	0.00%	0.00 e-99
52	0.00 e-99	0.00%	0.00%	0.00 e-99
53	-2.98 e-07	-0.00%	0.00%	5.59 e-09
54	-2.29 e-04	-0.07%	-0.26%	-1.61 e-06
55	-4.34 e-04	-0.17%	-0.82%	-7.42 e-06
56	-1.49 e-08	-0.00%	-0.00%	-1.49 e-08
57	4.66 e-10	0.00%	0.00%	4.66 e-10
58	-3.73 e-09	-0.00%	-0.00%	-3.73 e-09
59	-1.50 e-03	-0.62%	-0.62%	-1.50 e-03
60	-1.26 e-04	-0.02%	0.12%	-1.30 e-06
61	-1.61 e-03	-0.54%	0.66%	4.58 e-05

* taken from states with actual probability greater than 0.001 .

Table A.7 - Actual vs. Predicted Limiting Throughput

Data Set Number	Predicted Limiting Throughput	First "Limiting" State	Throughput in first limiting State + 10	Relative Error (predicted vs. actual)
25	9.89488	14	9.89486	0.0002%
26	19.77111	21	19.75959	0.06%
27	19.45762	17	19.44808	0.05%
28	18.2176	33	18.01599	1.11%
29	17.43412	35	17.16138	1.56%
30	15.96260	23	15.95428	0.05%
31	7.88559	22	7.88550	0.001%
32	8.89990	32	8.89986	0.0005%
33	49.26478	51	49.23563	0.06%
34	19.31167	40	19.31112	0.003%
35	7.10645	40	7.10635	0.001%
36	41.34323	53	40.70657	1.54%
37	15.77529	40	15.77477	0.003%
38	9.99992	1	9.99992	0%
39	19.99985	1	19.99985	0%
40	19.99791	6	19.99787	0.0002%
41	19.87652	*	*	*
42	19.14781	47	19.02328	0.65%
43	9.48505	43	9.43576	0.52%
44	8.99991	1	8.99991	0%
45	49.99902	10	49.99900	3.8e-05 %
46	19.99530	18	19.99530	0%
47	8.43359	59	8.43292	0.008%
48	48.44222	101	47.88927	1.14%
49	18.72449	63	18.72259	0.01%
50	10.00000	1	10.00000	0%
51	20.00000	1	20.00000	0%
52	20.00000	1	20.00000	0%
53	19.99995	1	20.00000	-0.0002%
54	19.97113	50	19.95912	0.06%
55	9.97016	55	9.95995	0.10%
56	9.00000	1	9.00000	0%
57	50.00000	1	50.00000	0%
58	20.00000	1	20.00000	0%
59	8.92896	85	8.92815	0.009%
60	49.96501	68	49.95691	0.02%
61	19.84181	95	19.83888	0.01%

* - system did not reach limiting state among states with significant probability

Table A.8 - Exact vs. Norton's Equivalent Predicted Mean Number in System $B=1$.

Data Set Number	Exact Mean in System	Predicted Mean in System	Relative Error
1	0.14286	0.14286	0.0%
2	4.3460	4.3617	0.36%
3	2.7774	2.8054	1.0%
4	3.0000	3.0000	0.0%
5	0.14286	0.14286	0.0%
6	0.17857	0.17857	0.0%
7	0.17647	0.17647	0.0%
8	7.0000	7.0000	0.0%
9	1.0000	1.0000	0.0%
10	1.0000	1.0000	0.0%
11	0.14286	0.14286	0.0%
12	1.1898	1.2638	6.2%
13	9.9725	11.7111	17.4%
14	7.7390	9.2560	19.6%
15	3.0004	3.0007	0.01%
16	9.0054	9.0075	0.02%
17	0.17876	0.17890	0.08%
18	0.17647	0.17647	0.00%
19	0.33439	0.33529	0.3%
20	1.00326	1.0071	0.4%
21	7.6665	7.8995	3.0%
22	4.4170	4.6154	4.5%
23	1.0593	1.0957	3.4%
24	1.0090	1.1070	0.8%

Table A.9 - Exact vs. Norton's Equiv. Predicted Mean Number in System $B > 1$.

Data Set Number	Exact Mean in System	Predicted Mean in System	Relative Error
25	2.0113e+01	2.0208e+01	0.47%
26	8.7052e-01	8.0070e-01	-8.02%
27	4.531e+00	4.462e+00	-1.52%
28	9.51148e-01	8.44844e-01	-11.18%
29	9.9213e-01	8.8064e-01	-11.24%
30	1.07039e+01	1.13802e+01	6.32%
31	3.856e+01	4.236e+01	9.85%
32	4.001e+01	4.001+01	0.00%
33	2.16665e+00	2.07235e+00	-4.35%
34	1.1850e+01	1.1729e+01	-1.02%
35	9.9502e+01	1.03511e+02	4.03%
36	2.78061e+00	2.60421e+00	-6.34%
37	3.5138e+01	3.6127e+01	2.81%
38	1.8001e+01	1.7835e+01	- 0.92%
39	8.5715e-01	7.7556e-01	-9.52%
40	4.155e+00	3.950e+00	-4.93%
41	8.6000e-01	6.7241e-01	-21.81%
42	4.6748e+00	4.2167e+00	-9.80%
43	7.443e+00	7.067e+00	-5.05%
44	3.600e+01	3.575e+01	-0.69%
45	2.11770e+00	1.99247e+00	-5.91%
46	1.0508e+01	1.0144e+01	-3.46%
47	1.830e+01	1.738e+01	-5.03%
48	2.17192e+00	1.63941e+00	-24.53%
49	1.29345e+01	1.19422e+01	-7.67%
50	1.8000e+01	1.7833e+01	-0.93%
51	8.5714e-01	6.6341e-01	-22.60%
52	4.154e+00	3.947e+00	-4.98%
53	8.5714e-01	7.7552e-01	-9.52%
54	4.1645e+00	3.3823e+00	-18.78%
55	6.053e+00	5.060e+00	-16.41%
56	3.600e+01	3.575e+01	-0.69%
57	2.11765e+00	1.99231e+00	-5.92%
58	1.0500e+01	1.0131e+01	-3.51%
59	1.42267e+01	1.19134e+01	-16.26%
60	2.11797e+00	1.45052e+00	-31.51%
61	1.066026e+01	8.5444e+00	-19.85%

Table A.10 - Norton's Equiv. Predicted vs. Actual $\Pr\{n_S=j\}$, $B=1$.

Data Set Number	Maximum deviation	Corresp. rel. error	Maximum rel.error *	Corresp. deviation
1	-3.7 e-09	-0.00%	-	-
2	1.2 e-04	-0.2%	5.0%	5.3 e-05
3	6.1 e-04	0.2%	19.5%	2.1 e-04
4	-6.1 e-10	-0.00%	-	-
5	-1.7 e-10	-0.00%	-	-
6	-1.2 e-10	0.00%	-	-
7	-1.8 e-09	0.00%	-	-
8	1.1 e-8	0.00%	0.0004%	4.5 e-09
9	7.2 e-08	0.00%	-	-
10	-1.5 e-06	0.00%	0.03%	6.2 e-07
11	-8.3 e-07	0.00%	0.02%	2.7 e-08
12	-7.6 e-03	-5.8%	89.5%	1.1 e-03
13	-9.5 e-03	-15.6%	72.9%	7.7 e-04
14	-1.3 e-02	-16.6%	91.4%	9.1 e-04
15	-9.2 e-06	0.01%	0.08%	8.3 e-07
16	-1.5 e-05	-0.02%	0.09%	1.0 e-06
17	-8.0 e-05	-0.06%	0.3%	9.6 e-06
18	-5.5 e-09	-0.00%	-	-
19	-3.0 e-04	-0.2%	2.4%	7.2 e-05
20	-3.4 e-04	-0.3%	5.5%	1.1 e-04
21	-2.1 e-03	-3.0%	13.7%	1.5 e-04
22	-3.9 e-03	-3.9%	26.8%	3.0 e-04
23	-3.9 e-03	-3.0%	56.4%	7.2 e-04
24	-6.7 e-04	-0.5%	22.9%	2.3 e-04

* taken from states with actual probability greater than 0.001 .
- same data point as maximum deviation

Table A.11 - Norton's Equiv. Predicted vs. Actual $\Pr\{n_S=j\}$ $B>1$.

Data Set Number	Maximum deviation	State Number	Corresp. rel. error
25	6.4637167e-03	1	23.5301
26	2.6311457e-02	1	37.2528
27	2.5155077e-02	1	35.3699
28	4.4043558e-02	1	60.1404
29	4.7076593e-02	1	63.4376
30	2.4025493e-02	1	52.6990
31	8.4478229e-03	1	51.1369
32	2.7466008e-03	1	24.1141
33	8.7536791e-03	1	31.5826
34	9.1328443e-03	1	36.2414
35	3.0417046e-03	1	55.4045
36	1.8513402e-02	1	58.5487
37	7.4239023e-03	1	55.9029
38	7.5746448e-03	1	25.2488
39	2.7658448e-02	1	39.5121
40	2.9534638e-02	1	40.3920
41	5.9375637e-02	1	84.6852
42	6.4470039e-02	1	91.8485
43	5.2414219e-02	1	93.0551
44	3.2792440e-03	1	26.5741
45	9.9546774e-03	1	36.5981
46	1.2076001e-02	1	46.0213
47	2.9237108e-02	1	145.4583
48	4.0958105e-02	1	148.6718
49	3.5667190e-02	1	147.5041
50	7.5773234e-03	1	25.2577
51	2.7658929e-02	1	39.5128
52	2.9548740e-02	1	40.4058
53	5.9748383e-02	1	85.3548
54	7.2658525e-02	1	99.4441
55	6.4272429e-02	1	103.2212
56	3.2704942e-03	1	26.4817
57	9.9554996e-03	1	36.6011
58	1.2092640e-02	1	46.0672
59	4.1831007e-02	1	183.0022
60	4.4176968e-02	1	162.4051
61	4.8030736e-02	1	183.9586

Table A.12 - Data Set Parameters (Chapter 4)

Data Set Number	$\lambda Bf/\mu_X$ as % of maximum	Batch size	Window size	Frag	$\mu_X/f\,\mu_A$
1	22%	3	2	2	.43
1a	26%	3	2	2	.83
3	20%	3	4	2	.43
5	20%	3	15	2	.43
7	17%	3	2	4	.23
9	17%	3	4	4	.23
13	24%	3	2	16	.06
19	35%	8	2	2	.45
21	32%	8	4	2	.45
25	48%	8	2	6	.15
35	79%	3	2	2	.83
36	67%	3	4	2	.83
37	62%	3	8	2	.83
38	70%	3	2	4	.63
39	62%	3	4	4	.63
40	60%	3	8	4	.63
41	69%	3	2	10	.63
42	61%	3	4	10	.63
50	80%	6	2	2	.83
51	67%	6	4	2	.83
52	62%	6	8	2	.83
53	96%	6	2	5	1.25
54	81%	6	4	5	1.25
45	93%	3	4	2	.94
46	85%	3	8	2	.94
47	98%	3	2	8	.75
48	84%	3	4	8	.75
49	80%	3	8	8	.75
57	89%	5	4	2	.93
58	82%	5	8	2	.93
59	86%	5	2	5	.70
60	85%	5	4	5	.70

Table A.13 - Exact vs. Predicted Mean Number in System (Chapter 4)

Data Set Number	Exact Mean in System	Predicted Mean in System	Relative Error
1	0.50881	0.5116	0.6%
1a	0.5793	0.5966	2.8%
3	0.4879	0.4881	0.04%
5	0.4875	0.4875	0.00%
7	0.4025	0.40785	0.5%
9	0.4005	0.4049	1.1%
13	0.59605	0.6126	2.8%
19	2.334	2.339	0.2%
21	2.0936	2.0973	0.2%
25	3.9804	4.05525	1.9%
35	6.470	6.273	3.0%
36	3.329	3.382	1.6%
37	2.825	2.855	1.1%
38	3.827	4.037	5.5%
39	2.759	2.901	5.1%
40	2.664	2.778	4.3%
41	3.569	3.907	9.5%
42	2.663	2.877	8.0%
50	12.286	12.030	-2.1%
51	6.228	6.370	2.3%
52	5.175	5.253	1.5%
53	71.582	70.147	-2.0%
54	12.365	13.402	8.4%
45	23.767	22.50	-5.3%
46	9.422	9.480	0.6%
47	62.555	63.30	1.2%
48	8.357	8.82	5.5%
49	6.7101	7.076	5.5%
57	22.289	21.859	-1.9%
58	11.566	11.778	1.8%
59	15.709	15.823	0.7%
60	14.780	15.112	2.2%

Table A.14 - Predicted vs. Actual $\Pr\{n_S=j\}$

Data Set Number	Maximum deviation	Corresp. rel. error	Maximum rel.error *	Corresp. deviation
1	2.98 e-03	5.1%	5.1%	2.98 e-03
1a	8.45 e-03	12.1%	12.1%	8.45 e-03
3	6.2 e-05	0.008%	0.1%	4.7 e-06
5	1.8 e-10	0.0%	0.0%	1.8 e-10
7	7.0 e-04	1.4%	14.1%	4.0 e-04
9	4.3 e-04	0.8%	13.5%	3.8 e-04
13	1.5 e-03	2.0%	34.8%	3.6 e-04
19	1.8 e-03	4.9%	4.9%	1.8 e-03
21	1.6 e-04	0.02%	0.48%	3.2 e-05
25	6.5 e-04	1.3%	10.4%	1.1 e-04
35	7.0 e-03	3.1%	19.1%	2.2 e-04
36	1.4 e-02	3.7%	8.1%	9.5 e-05
37	4.1 e-03	1.0%	1.8%	2.0 e-04
38	9.5 e-03	3.0%	17.1%	1.9 e-04
39	4.0 e-03	1.0%	26.7%	2.8 e-04
40	4.4 e-03	3.4%	26.4%	3.2 e-04
41	1.3 e-03	3.9%	37.9%	3.9 e-04
42	6.6 e-03	5.1%	46.3%	5.4 e-04
50	5.0 e-03	10.0%	10.0%	5.0 e-03
51	1.4 e-03	4.0%	5.25%	1.5 e-03
52	6.1 e-03	1.5%	2.4%	2.3 e-04
53	3.5 e-03	7.2%	9.5%	1.4 e-03
54	5.0 e-02	20.3%	20.3%	5.0 e-02
45	9.2 e-03	11.2%	18.3%	1.9 e-04
46	1.3 e-02	7.6%	11.0%	1.2 e-04
47	2.1 e-03	7.4%	9.4%	9.7 e-04
48	7.9 e-03	4.6%	15.5%	1.6 e-04
49	3.0 e-03	3.9%	22.7%	2.5 e-04
57	1.3 e-02	10.4%	10.4%	1.3 e-02
58	1.4 e-02	6.8%	6.8%	1.4 e-02
59	4.8 e-03	3.2%	4.7%	1.8 e-03
60	2.1 e-03	1.4%	6.3%	6.5 e-05

* taken from states with actual probability greater than 0.001 .

Table A.15 - Observed vs. Predicted a_∞

Data Set Number	Observed a_∞	Predicted a_∞	Relative Error
35	.234524	.23863	1.8%
36	.08910	.089185	0.1%
38	.13057	.12589	-3.6%
39	.02027	.01636	-19.3%
41	.12009	.106326	-11.5%
42	.01665	.009537	-42.7%
50	.239666	.24165	0.8%
51	.095028	.09479	-0.3%
52	.02298	.023105	0.5%
53	.372935	.37181	-0.3%
54	.263342	.24894	-5.5%
45	.141384	.140718	-0.5%
46	.058793	.055713	-5.2%
47	.179003	.1778	-0.7%
48	.046603	.038885	-16.6%
49	.0048167	.0028725	-40.4%
56	.332445	.33245	1e-6%
57	.1376119	.136884	0.5%
58	.05589	.053589	-4.1%
59	.16403	.16111	-1.8%
60	.017157	.015943	-7.1%

Table A.16 - Simulation Data Set Parameters

Simulation Data Set Number	B_2	B_1	λ	μ_X	μ_{A_1}	μ_{A_2}	W_1	W_2	Low Level data set Number
1a	1	1	10.0	19.8	20.0	19.0	4	3	12
1b							4	8	12
1c							16	3	*
1d							16	8	*
2a	2	1	5.0	19.8	20.0	19.0	4	3	12
2b							4	8	12
2c							16	3	*
2d							16	8	*
2e	5	1	2.0	19.8	20.0	19.0	4	3	12
2f							4	8	12
2g							16	3	*
2h							16	8	*
3a	1	3	4.5	20.0	22.0	21.0	3	3	30
3b							8	3	42
3c							3	8	30
3d							8	8	42
3e	1	8	2.0	50.0	55.0	53.0	3	3	36
3f							8	3	48
3g							3	8	36
3h							8	8	48
4a	2	3	2.25	20.0	22.0	21.0	3	3	30
4b							8	3	42
4c							3	8	30
4d							8	8	42
4e	2	8	1.0	50.0	55.0	53.0	3	3	36
4f							8	3	48
4g							3	8	36
4h							8	8	48
4i	5	3	0.9	20.0	22.0	21.0	3	3	30
4j							8	3	42
4k							3	8	30
4l							8	8	42
4m	5	8	0.4	50.0	55.0	53.0	3	3	36
4n							8	3	48
4o							3	8	36
4p							8	8	48

Table A.17 - Comparison of Approximate and Simulation Means

Data Set	Approx. mean	Simulation mean	95% conf. interval	Number of runs	Average num. busy periods
1a	1.64086	1.609	+-0.3516	6	72.5
1b	1.182	1.1778	+-0.1997	6	78.5
1c	1.4297	1.445	+-0.2807	6	74.5
1d	1.0355	1.001	+-0.1308	6	80.2
2a	2.7365	2.3068	+-0.5234	6	54.3
2b	1.8230	1.7338	+-0.3324	6	59.7
2c	2.4025	2.033	+-0.4543	6	56.3
2d	1.6075	1.4247	+-0.2428	6	62.5
2e	6.2412	6.108	+-1.610	15	27.3
2f	3.9034	4.1997	+-0.8466	15	29.3
2g	5.5368	5.4561	+-1.4297	15	26.8
2h	3.5031	2.8488	+-0.6891	6	32.0
3a	4.0051	3.2045	+-0.4435	15	61.9
3b	1.9267	1.7681	+-0.2595	6	86.7
3c	3.9699	3.2013	+-0.4440	15	62.0
3d	1.9115	1.7655	+-0.2593	6	87.0
3e	0.5351	0.5244	+-0.0286	6	283.7
3f	0.4347	0.4127	+-0.0194	6	299.8
3g	0.5351	0.5245	+-0.0286	6	283.7
3h	0.4347	0.4127	+-0.0194	6	299.8
4a	6.5422	5.6219	+-0.8189	25	35.3
4b	3.1004	2.7855	+-0.4104	15	56.8
4c	6.4776	5.6043	+-0.8200	25	35.4
4d	3.0694	2.7749	+-0.4123	15	56.9
4e	0.8414	0.8300	+-0.0724	6	149.7
4f	0.6787	0.6452	+-0.0505	6	159.5
4g	0.8414	0.8300	+-0.0724	6	149.7
4h	0.6787	0.6452	+-0.0505	6	159.5
4i	11.416	13.693	+-3.1462	25	14.9
4j	6.6247	7.4186	+-1.4554	25	24.5
4k	14.001	13.675	+-3.1474	25	15.0
4l	6.5432	7.3887	+-1.4545	25	24.6
4m	1.7604	1.6871	+-0.2271	6	63.0
4n	1.4108	1.3066	+-0.1541	6	67.8
4o	1.7604	1.6871	+-0.2271	6	63.0
4p	1.4108	1.3066	+-0.1541	6	67.8