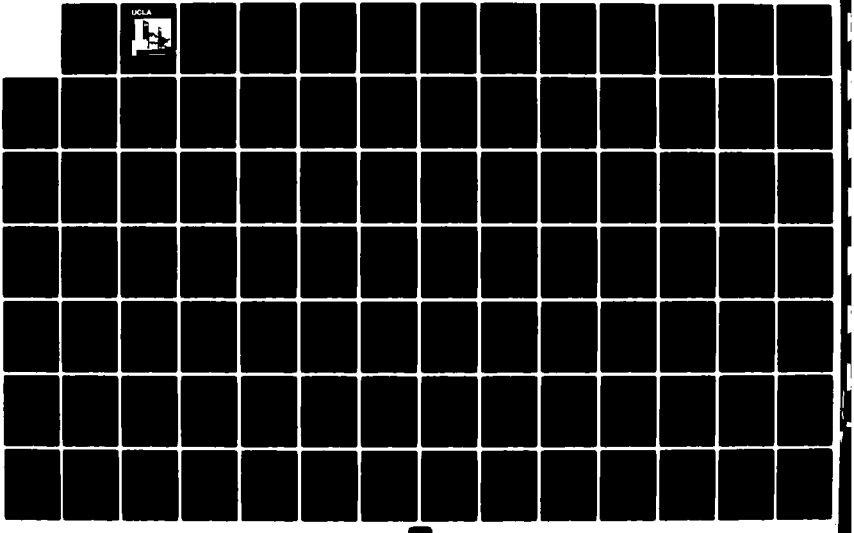
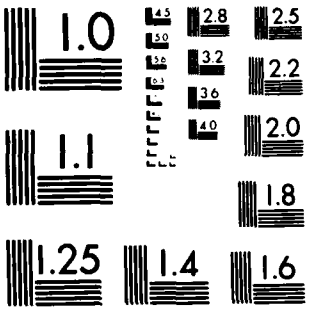
AD-A130 784 FLOW CONTROL IN STORE-AND-FORWARD COMPUTER NETWORKS(U) 1/2
CALIFORNIA UNIV LOS ANGELES DEPT OF COMPUTER SCIENCE
C TSENG APR 81 UCLA-ENG-8110 MDA903-77-C-0272

UNCLASSIFIED F/G 12/1 NL





(12)

UCLA

COMPUTER SCIENCE DEPARTMENT

FLOW CONTROL IN STORE-AND-FORWARD COMPUTER NETWORKS

Contract MDA-903-77-C-0272

Chong-Wei Tseng

April 1981
Report No. CSD-810404

COMPUTER SCIENCE DEPARTMENT OFFICERS

Dr. Gerald Estrin, Chairperson
Dr. Bertram Bussell, Vice Chairperson
Mrs. Arlene C. Weber, Management Services Officer
Room 3731 Boelter Hall

COMPUTER SCIENCE DEPARTMENT RESEARCH LABORATORY

Dr. Leonard Kleinrock, Head
Room 3732 Boelter Hall

This report is part of a continuing series of technical reports initiated in January 1981 and is issued by the Computer Science Department Research Laboratory at UCLA. This technical report presents the latest research results established by the department members and its research staff. It is directed to the professional community and ranges from the presentation of short technical contributions to complete Ph.D. dissertations. Its function is to make available the latest results of our continuing research in a timely fashion. For a complete list of reports in this series you may contact The Department Archivist, at the address below.

University of California, Los Angeles
Computer Science Department
School of Engineering and Applied Science
3732 Boelter Hall
Los Angeles, California 90024

UCLA-ENG-8110
APRIL 1981

FLOW CONTROL IN STORE-AND-FORWARD COMPUTER NETWORKS

by
Chong-Wei Tseng

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

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

ACKNOWLEDGEMENTS

I wish to express my deep thanks to my doctoral committee, consisting of Professors Kirby Baker, Stephen Jacobsen, Leonard Kleinrock (chairman), Mario Gerla, and Charles Stone. I owe my particular gratitude to my research advisor, Professor Leonard Kleinrock, who has stimulated my interest in modeling and analyzing computer communication networks. He has constantly encouraged and guided me through my years in this doctoral research. Without his generous support and valuable advice this research would not have been fruitful.

My study has been helped greatly by Mr. Frank Hanzel with his fast, accurate and patient implementation of our simulation programs.

I also owe a great deal of gratitude to George Ann Hornor and Terry Peters. With her incredible skill and patience in editing and typesetting, George Ann has transformed the confusing English and difficult mathematical expressions into this readable document, and she has always been kind and friendly to me, my wife and my son. Terry has helped in preparing and correcting the many drafts.

The wonderful staff at UCLA, especially Linda Infeld, Lou Nelson, Arlene Weber, Riva Martin and Brenda Ramsey have offered me various kinds of assistance and favors that have greatly eased my life as a student.

I must also thank my wife, Yu-hsiang Elina Tseng, and my mother, Chin-yu Wu, for their constant encouragement and support during the past years. Finally, I feel lucky to have such a nice son, Arvin, who plays quietly so as not to disturb my studies.

ABSTRACT

We study the broad area of flow control and sequencing in computer communication networks.

A special case of message sequencing is shown to reduce to a Markov process. A more general sequencing model is also studied; only approximate results are obtained in this case.

We formally show that an end-to-end flow control scheme in computer communication networks can be modeled as a closed network of queues. We also introduce and analyze a flow control scheme based on limiting the permit generation rates. It turns out that this scheme is very effective. More importantly, this scheme involves essentially no computational complexity. This allows us to measure and adjust the network performance with little difficulty. The simplicity also allows us to incorporate more realistic aspects of network protocols such as piggy-back acknowledgement, time-out and gate functions into our analysis.

Two simple but effective distributed dynamic flow control schemes are proposed and analyzed. It is shown that with either of these two schemes, a network can respond well to the dynamically changing environment.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 The Necessity for Flow Control

Resource scarcity is a common phenomenon in many aspects of our lives. We are accustomed, for example, to finding that there are fewer homes than families needing them, fewer jobs than people of working age, and desperately fewer doctors than patients, etc.

How does this manifest itself in a store-and-forward computer communication network? A computer communication network has only a finite number of buffers, a limited transmission bandwidth, a finite nodal processing capacity, and a limited number of logical channels. Thus if there is no restriction on the entrance rate of data into the network, sooner or later some unpleasant situation will occur. Some of these unpleasant situations can be the loss of information, performance degradation (to almost zero throughput and intolerable delay) [GEIS 76], unfairness of service, or even complete deadlocks [KLEI 76].

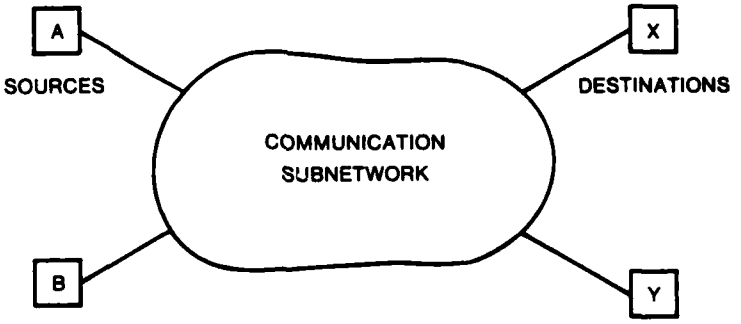


Figure 1.1. A computer communication network.

For example, consider the computer network shown in Figure 1.1, where a set of sources and destinations (host computers) communicate with each other by sending messages (or packets) through the store-and-forward communication subnetwork. Suppose source A sends messages to destination X at a rate faster than X can consume. A queue of messages will soon build up at the destination switching node. If the activity of source A is not stopped quickly enough, then either a large number of buffers in the network will be occupied by source A's messages (preventing the network from serving others), or the destination switching node must discard some messages.

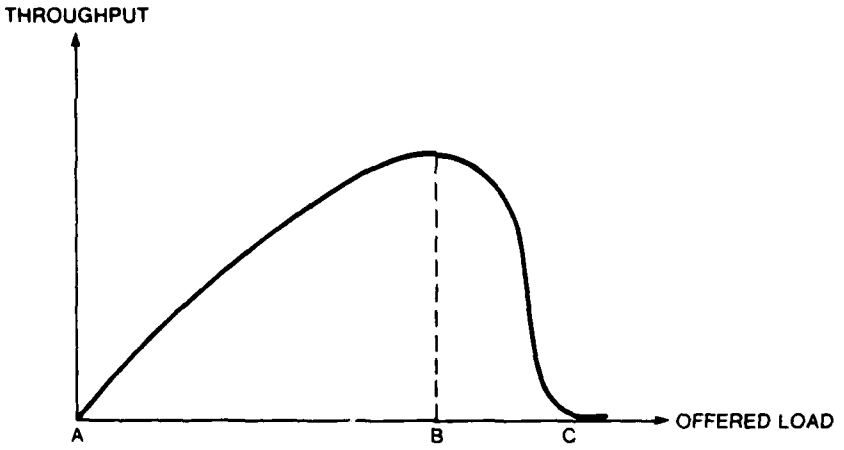


Figure 1.2. Typical throughput as a function of offered load.

Even if the input rate of the sources is slower than the output rate of the destinations, a computer communication network still cannot function without flow control. A typical performance degradation of computer communication networks without flow control is depicted in Fig. 1.2. Here we see that the throughput increases monotonically with the offered load until some threshold is reached. Beyond that level the throughput collapses quickly to essentially zero. The causes of this collapse are multi-fold and are closely related to the network protocols at various levels.

For instance, a *virtual circuit* computer communication network requires that a communication path be established before the communication can start. If the path cannot be established then the request is refused [POUZ 76]; note that this part of the protocol is itself a flow control scheme. Suppose we did *not* refuse the request, but instead let the data from the new link enter the network and occupy buffers at the source node while seeking a path with free buffers available for communication. We would most certainly find that when the externally offered load is high, congestion at the source node will cause the collapse in throughput illustrated in Figure 1.2. (This congestion occurs because the node's buffers are now depleted and it is difficult for data to move.) Even worse, we may have a complete deadlock where no traffic can move. Moreover, simply the fluctuation of both arrival and service rates can cause a large number of messages to wait for service and to accumulate in the network. Indeed, the waiting time may become so long that the sources decide that their messages have been lost and in order to secure the communication, the sources would retransmit those presumably lost messages (this is the so-called "timeout" mechanism, necessary because the communication channels are usually error-prone). These retransmissions may in turn cause the network additional overload and therefore an even longer message waiting time, etc. In fact, a source may make several unnecessary retransmissions before it finally knows that it can stop doing so and thus the throughput is severely degraded since much useless work is done [KAHN 72]. In any case, the fall-off in throughput is caused by wasted resources that are either allocated to repeating work or that are idle because of, say, congested receivers that are in "refusing" states, thus forcing the sender into a "waiting" state.

Studying Fig. 1.2, we see that one should prevent the network from operating in the region (B,C). On the other hand, one should also prevent the network from operating near A while it could operate around B. This can be done by properly throttling the input rate of messages into the network or by placing a limit on the maximum number of messages that can exist in the network simultaneously.

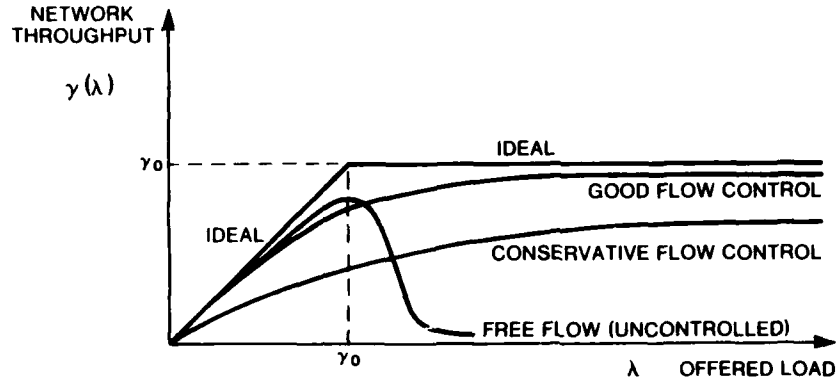


Figure 1.3. The effectiveness of flow control schemes.

A simple measure to test the effectiveness of a flow control scheme is discussed in [KLEI 78], and is shown in Fig. 1.3 where we see that an ideal flow control scheme will accept the total input traffic as long as the offered traffic level is less than the network capacity, say γ_0. Beyond that the input traffic is cut at the level γ_0. The closer to the ideal curve a flow control scheme can perform, the better is the scheme. Note that a more accurate measure in comparing flow control schemes is one that also takes message delay into consideration. One such example is *power*, the ratio of throughput to delay versus offered load [KLEI 79]. For our purpose here we shall assume that γ_0 is set such that the message delay is below some acceptable limit and therefore is not critical. That is, we shall use "throughput versus offered load" as the measure to compare flow control schemes.

In summary, flow control is a set of mechanisms whereby the flow of data can be maintained within limits compatible with the amount of available network resources and whereby a single user or a single user group is prevented from hoarding the resources of the network to the detriment of others (i.e., fairness of service) [RUDI 76].

1.2 Existing Methods of Flow Control

Various flow control schemes have been investigated and implemented in the past; see [GERL 80] and the references therein. They can largely be classified as schemes of *local flow control* or *global flow control*. Examples of local flow control schemes are the "minimal allocation - maximal limit" scheme in the ARPANET [MCQU 72], the "square root scheme" [IRLA 78], the "sharing with minimal allocation" scheme [KAMO 76], the GMD "buffer class" scheme [RAUB 76]. These schemes are intended to prevent local store-and-forward buffer congestion (local throughput degradation) and direct (and indirect) store-and-forward deadlock. While possibly improving the local network performance, a local scheme cannot optimize the

global network performance (high network utilization, high throughput, low delay, fairness of service, etc.). As an example, local flow control stops an overactive user's input only when the backpressure from the local congestion propagates upstream back to the source. By that time, however, this user may have captured an unreasonably large portion of network resources, and other users may be blocked from using the network. To prevent this (and other) unhappy situations, global flow control schemes are often used.

At first thought, one may think that a global flow control scheme should allocate network resources to users in a way such that, well before the network capacity is reached, users' inputs are throttled so that the network will not be overloaded. Indeed, this can be accomplished either by properly allocating a maximal number of permits or by assigning an appropriate window size to each group of users (the group can be as large as that containing all users or as small as that containing a single user). The idea is to keep the total number of messages simultaneously existing in the network below the network's capacity; that is, the network as a whole will not be overloaded. However, computer traffic is bursty and most of the time only a small number of users are active simultaneously. Therefore, the network resources are often overcommitted (e.g., the window size for each user is set such that the sum of all windows for all users is much greater than the network capacity) so that, on the average, the network utilization is kept high enough to be reasonably efficient. When resources are over-allocated, the network is apt to become overloaded from time to time as occasionally a majority of users become active simultaneously. This (infrequent) overloading must be handled by some other mechanism, such as a dynamic or local flow control scheme mentioned above.

In short, local flow control is usually used to resolve local congestion and deadlocks, while global flow control is used to achieve high network performance (high utilization, high overall throughput, tolerable admission delay, lower delivery delay, fairness of service, etc.) most of the time, and further to guarantee that when the network is overloaded it is overloaded gracefully, i.e., network resources are allocated fairly among users.

In this dissertation we concentrate only on global flow control. Existing methods of global flow control can be largely classified into the three categories described below.

1.2.1 Isarithmic Flow Control

This method was originally suggested by Davies [DAVI 72], and has been implemented by Price in his computer simulation model [PRIC 77]. The network is initially provided with a fixed number of "permits", several of which are held in store at each node. As traffic is offered by the local host computer to the subnet, each packet must secure a permit before admission to the subnet is allowed. Each accepted packet temporarily consumes (i.e., seizes) a permit from the accepting node's store. When a packet reaches its destination, the permit is released and is either left at the destination node for its use or shipped elsewhere for the use of others.

The main advantage of this scheme is that it detects network congestion and automatically throttles input into the network (the detection and throttling are automatically done since there are only a fixed number of permits).

The difficulty of this scheme lies in finding a good adaptive way for distributing these permits so as to maximize throughput or minimize delay. It has been shown in [PRIC 73] that the original isarithmic control scheme cannot by itself secure the fairness of service. Moreover, a network usually deals with different types of data. For example (see [OPDE 74]), data may be of the HT (high throughput) type as in the case of file transfer, the LD (low delay) type as in the case of time sharing, or of the RT (real time) type as in the case of speech transmission.

Different types of data require different ways of distributing permits. The HT type prefers the network to allocate a large number of permits to current active sources so that the throughput can be maximized. The LD type would rather the network make sure each source always has several permits at hand so that the admission delay is minimized. The RT type requires that once a source is allowed to transmit its messages, enough permits should be given so that the real-time nature of the traffic is not distorted. So far there exists no easy way to effectively distribute the permits.

1.2.2 End-To-End Flow Control

Typically most networks are based on the concept of logical channels* and each logical channel is end-to-end controlled. Examples of end-to-end flow control are SNA pacing [IBM 75], RFNM in the ARPANET [MCQU 72, KLEI 76] and various window mechanisms [POUZ 76, CERF 74]. Important functions of such end-to-end flow control are to synchronize the source input rate to the sink acceptance rate, and to protect the network from being overloaded by a small number of users. All of these schemes work by limiting the number of messages in a logical channel. Suppose that L_i is the maximum number of messages in logical channel i, and that the network has a total of K logical channels. Then the maximum number of messages permitted to enter the network is

$$N_{max} = L_1 + L_2 + \cdots + L_K$$

Due to the bursty nature of the data sources, N_{max} is usually set higher than the network can actually handle, e.g., at point C of Figure 1.2. Through averaging, the network utilization is hopefully at point B of Figure 1.2. In other words, N_{max} is set according to the overcommitment principle. Consequently there is a finite probability that all sources would be active simultaneously, thereby reaching N_{max}, and the network would be driven into the region beyond point C of Figure 1.2. Thus, end-to-end flow control is not by itself sufficient to keep the network from being congested. We also need a way (perhaps dynamic) to limit the total number of messages in the network.

1.2.3 Hybrid Flow Control

This scheme is essentially the combination of isarithmic and end-to-end control schemes. The communication sources/destinations are divided into exclusive groups and each group is provided with a fixed number of permits. Above this, the total number of messages from all groups is also restricted. An example of this is shown in [WONG 77]. We will discuss this further in the following section.

1.3 Existing Models

Due to the complex multi-variate environment of flow control, quantitative and analytic results have been few. Among them are:

---

* A logical channel is a conceptual connection between a source node and a destination node or a source process and a destination process.

1.3.1 Wong and Unsoy's Two Level Flow Control Model [WONG 77]

J. W. Wong and M. Unsoy considered a fairly general flow control scheme in which flow is controlled at various levels. For example, at the first level, an upper limit L is set for the total number of messages allowed into the network; in the second level, an upper limit L_r is set for the r^{th} disjoint group of source-destination pairs. In fact, this is the hybrid scheme mentioned in the last section. Observe the generality of this model. If $L = \sum_{r=1}^{K} L_r$, where K is the total number of logical channels, then it is an end-to-end flow control scheme. On the other hand, if $L_r = L$ for all $r = 1, 2, \cdots, K$, then it is a pure isarithmic flow control scheme.

Equipped with the tools from the theory of closed queueing networks [GORD 67], [BASK 75], Wong and Unsoy were able to solve for the equilibrium state of a computer communication network with any topology, any loop-free, non-merging (after splitting) routing scheme and any traffic matrix.

Although their work may be considered as an elegant academic achievement, the model implicitly assumes that each node can obtain information (such as "how many messages are in the network now?") at no cost, and unfortunately, this is unrealistic. In a real network, control information is either piggy-backed or sent as an ordinary packet through the network. Thus it takes time for one to gather information and the information one gets is partially "out of date". In fact, this is one of the factors motivating our investigation of new models which take this delay in gathering information into consideration.

1.3.2 Kleinrock and Kermani's End-To-End Flow Control Model [KLEI 80a]

In this model, a network is viewed as a channel whose delay is a function of the offered load (namely, a multi-stage Erlangian delay distribution with parameter as a function of offered load). Under this assumption (and other usual assumptions) Kleinrock and Kermani were able to show the effects of window size, time-out duration, and destination buffer size on network performance. A dynamic scheme based on Markovian decision processes [HOWA 60] that gave a static control policy based on the number of available empty buffers at the destination was also investigated.

Two weak points of the model are (1) it is difficult to find the exact relationship between the network delay and the offered load (on which the entire analysis is based), and (2) the topology of the network (which plays an important role in network performance) is ignored so that the effect of side traffic was not taken into consideration.

1.3.3 Lam and Reiser's Input Buffer Limits Model [LAM 77]

Based on the idea of GMD's buffer-class model [GIES 76], Lam and Reiser modeled a computer communication network as a connection of nodes that were themselves modeled as illustrated in Figure 1.4a, which shows how data input from a channel to a node is routed to other nodes through corresponding channels (for details see [LAM 77]). They further distinguished between two classes of data traffic. Class 1 is the transit traffic, and class 2 is the local input traffic. Class 1 traffic can occupy all buffers N_T of a node, while class 2 traffic can occupy no more than N_I buffers of a node. Lam and Reiser were able to show that this model predicted the congestion-prone behavior (see Figure 1.2) of a network if no control in input was imposed. The effect of various N_I/N_T ratios was also shown (see Figure 1.4b).

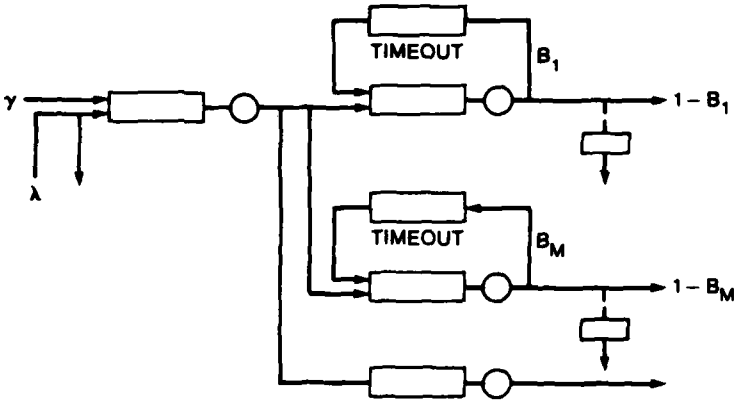


Figure 1.4(a). Lam's *model of a node.*

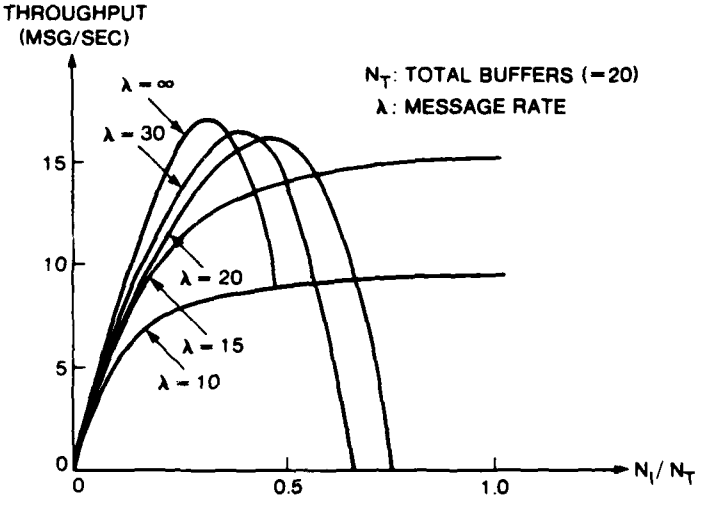


Figure 1.4(b). Throughput vs. N_I/N_T

The disadvantages of this model are: (1) It does not prevent the "capture phenomenon" of a network, that is, the super active sources seize all the resources of a network while the less active sources are prevented from even entering their data into the network; (2) it is a complicated model that requires an enormous amount of computational effort. For the latter reason they assume a "homogeneous network" meaning when one node is congested, all nodes are equally congested, which is unrealistic, especially when the network is large.

1.4 Organization and Results of the Dissertation

In the last section, we saw that there are only a few available flow control schemes and we discussed their mathematical models. Due to the complicated, multi-variate environment, one usually can only use one's intuition to design a flow control scheme; then one verifies and improves this scheme through simulation or, occasionally analysis. If one tries to set up a mathematical model for one's scheme, he usually finds that in order to obtain a model that is analytically solvable, he must exclude certain aspects of his scheme (such as sequencing, time-out mechanism, etc.) and make some convenient (but sometimes unrealistic) assumptions.

Moreover, most tractable existing flow control schemes are static rather than dynamic. They do not adjust themselves according to the variation of a network's state or environment. A more effective flow control scheme should be able to detect the state of the system and its environment, and then make some proper real-time decisions so that the best system performance can be attained.

In our study, we are interested in inventing and analyzing mathematical flow control models based on more realistic assumptions and that incorporate more important aspects of real flow control schemes. Further, we wish to derive some dynamic flow control models that are realistic, yet simple enough to allow mathematical analysis.

In Chapter 2, we look into the problem of *message sequencing*. First we present a simple case that allows us to describe sequencing by a Markovian chain. Then we investigate a more general case for which Markovian analysis can no longer be applied. It turns out that the theory in a $G/G/1$ queueing system provides a handle to deal with this problem. Using this tool, we find approximation results for the throughput and delay of a model network where message sequencing is required before messages can be delivered to their destinations. The throughput degradation caused by sequencing is quantitatively obtained. We leave Chapter 2 with the sequencing problem still not completely solved; however the finished work is a step on the way toward the complete analytic solution and gives a good guideline for engineering design.

In Chapters 3 and 4, we examine two new static flow control methods. The method in Chapter 3 transforms an 'open' computer communication network [JACK 63] (open in the sense that data flows in and out) into a closed queueing network [GORD 67]. Unfortunately because of the complexity of computation involved, only a heuristic solution is possible.

Two schemes in Chapter 4 create a new dimension in flow control methods by placing a limit on the permit generation rates. These schemes assign a fixed *rate* for generating permits (instead of using a fixed window size for unacknowledged permits) for the entire network (as in the first scheme) or for each logical channel (as in the second scheme). It is then shown that by doing so we obtain a flow control scheme that is fairly easy to implement. Later, the second scheme is improved to an ideal one (i.e., it can throttle the input traffic according to the ideal curve as shown in Figure 1.3). With this, the mechanisms of piggy-back acknowledgement and time-out are then easily incorporated into the mathematical model. Two cases of piggy-back

acknowledgements are examined. The first one is based on the number of ACKs that accumulate while they wait to be piggy-backed. Once the number exceeds a limit L, a special control message is generated to piggy-back all of the waiting ACKs. To avoid indefinite waiting of ACKs when the opposite traffic is low, the second case, which is based both on the number of ACKs accumulated and the length of time these ACKs have been waiting, is investigated. In the time-out scheme, the effect of channel error probability and the length of the time-out period are presented. Also in Chapter 4, we move into the area of inter-netting. In this area we show that our proposed flow control scheme can be nicely incorporated into the protocol that governs the flow control function of a gateway through which data of different networks pass.

In Chapter 5, we then look into dynamic flow control schemes. There we propose and analyze two simple distributed dynamic flow control methods. In these methods, no central node is needed to oversee the state of the entire network. The sources themselves, cooperating with the physical channels through which the sources' data pass, can control the flow dynamically and distributedly, based on measurements or information exchanges. To avoid the extreme complexity involved in the exact analysis, only a semi-quantitative analysis is given, and only simulated results are presented. These distributed dynamic schemes adapt to the changing environment effectively. In particular, throughput, delay and permit generation rate allocation versus time-varying external applied loads are presented.

We consider our studies in this dissertation as contributing to the important and neglected frontier of modelling and analyzing the protocols of distributed computer communication networks.

The results we obtain in Chapter 2 shed light on the way to solve the almost hopeless analytic problem of message sequencing, where the different message positions represent different states of a network (in addition to the number of messages existing in the network). The observation made in Chapter 3 enables us to describe a computer communication network as a closed network of queues. This greatly simplifies the network modeling, at least conceptually (regardless of the computational complexity). In Chapter 4, we reveal a new dimension of flow control techniques, namely, those based on limiting the permit generation rates. They are as effective as (if not better than) other existing flow control schemes. Yet they are easier to implement, and it is easier to incorporate other network protocols such as piggy-back acknowledgements, time-outs, etc. into the models. More importantly, these models are much easier to calculate quantitatively. No computational complexity is involved at all — a strong contrast to other existing schemes and models. The dynamic flow control schemes investigated in Chapter 5 provide simple dynamic distributed flow control methods that are a must for today's networks with their rapidly expanding size, arbitrary topology, and heterogeneous traffic matrices.

Our studies provide a tool to control the data flow of large, complicated computer communication networks and to describe, in a simple manner, the effects of flow control schemes on these networks.

CHAPTER 2
THE SEQUENCING PROBLEM

The models discussed in Chapter 1 do not consider the important area of *message sequencing*. Usually, a data terminating device (e.g., a host computer) requires messages be in the correct sequence before they can be accepted for delivery. If the network takes the responsibility for sequencing, messages occupy the buffers of the destination switching node longer than when sequencing is not required, and therefore throughput is reduced while the delay is increased. This has been partially investigated in [SUNS 75] but no significant result was obtained. In the following section we present and analyze two models in order to appreciate the complexity of the sequencing problem and to investigate its effect on the network performance.

As messages pass through the network they may take different paths, or they may be corrupted by noise (in which case they will be retransmitted) so that they may arrive at the destination node out of sequence. What if the destination node refuses to accommodate the out-of-sequence messages, or what if it accommodates them but won't deliver them until they are in the correct sequence? The exact analysis of such a problem is difficult because the state description now involves not only the number of messages in various queues but also their relative positions in these queues. As a network grows large, the number of states also becomes explosively large and soon diminishes our capability to solve, or even describe the system.

2.1 A Simple Example

To appreciate the complexity of the problem, let us first consider a system as shown in Figure 2.1, where we have a loss system with a Poisson message arrival rate λ. Let us assume that

(A1) A window size W (represented by the number of tokens) is imposed to limit the input traffic.

(A2) The round-trip delay time (message plus ACK delay) is exponential with rate μ.

(A3) There is a buffer of size B and an output channel with infinite speed at the destination node.

(A4) The buffer B accommodates the out-of-sequence messages, but when a message (say message a) arrives to find the buffer B full, if any messages in B are more out-of-sequence than message a, the buffer will eject (kick out) the most out-of-sequence message among those in the buffer in favor of message a.

(A5) The output channel will emit messages with infinite speed once they are in sequence.

(A6) When the buffer receives a message, it returns a positive ACK; when it rejects a message because there is no vacancy, it returns a negative ACK.

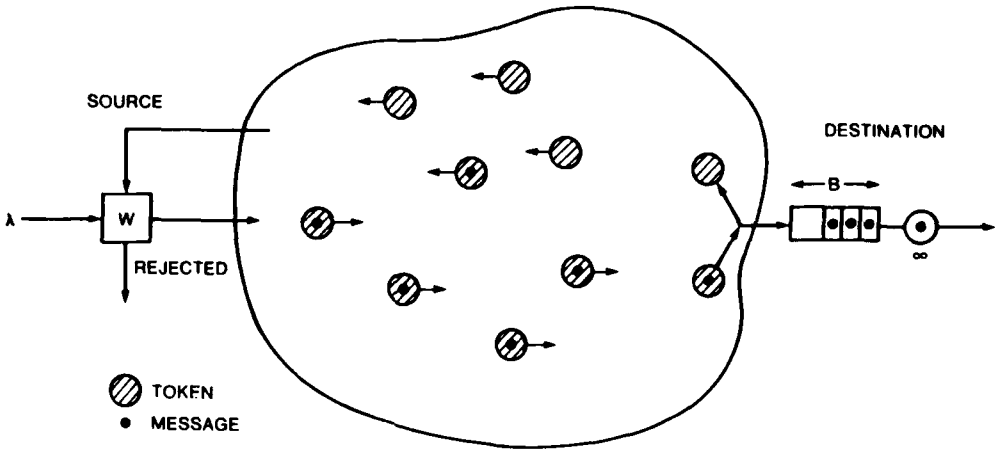


Figure 2.1. An illustrative network model for the sequencing problem.

(A7) The source immediately retransmits the rejected message once it receives the negative ACK.

(A8) When a message is kicked out of the buffer to make room for a less out-of-sequence message, this information is piggy-backed in the positive ACK for the accepted message so that the source will transmit the kicked out one again.

(A9) The channels are noiseless, and the network is reliable.

To illustrate how we can mathematically describe and solve the system, let us consider a simple case in which the window size $W = 4$, and the buffer size $B = 2$.

From the viewpoint of the source, we can describe the state of the system with a state vector, for example (1, 1, 1*, 1*), meaning that the source has sent four messages. Among them are the positive ACKs for the third and fourth messages (we number from left to right) that have come back, and since they are out of sequence, we denote the third and the fourth positions of the state vector by two "1*"s. These two messages are fully occupying the destination buffer (B=2). The simple "1" indicates that a message has been sent but its ACK is not back yet. Note that if now the +ACK for the second message comes back (which also informs the source that the fourth message has been kicked out), the state vector will become (1, 1*, 1*, 1) since the source knows now that the fourth message has been kicked out (by A4) to make room for the second message, the source retransmits the fourth immediately (by A7). If now the +ACK for the first message comes back, the source will change the state vector to (1, 0, 0, 0) since it knows that the first, second, and third message are now in sequence and are zipped out already (by A5), and the fourth message now becomes the first one. Zero here denotes "no activity". Thus the transition diagram of the states can be shown as in Figure 2.2.

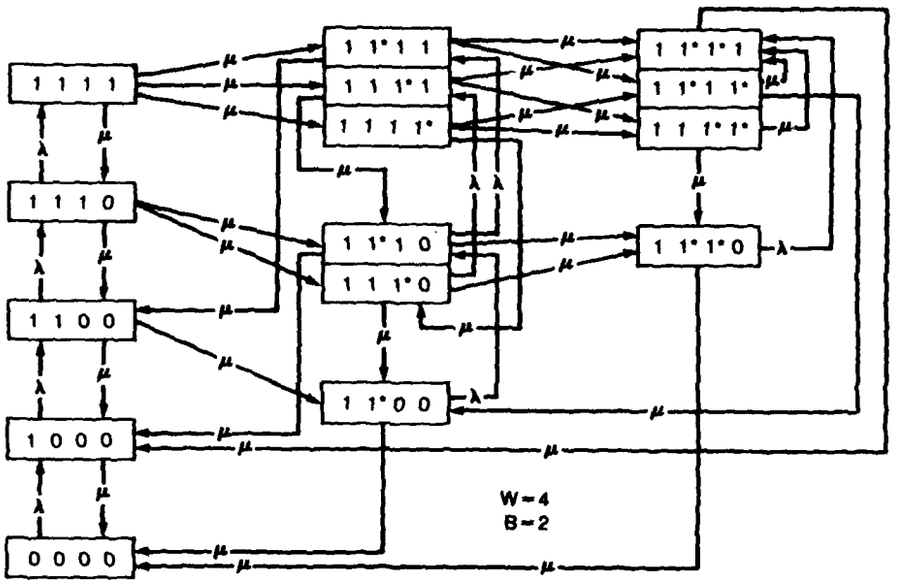


Figure 2.2. The state transition diagram from source's viewpoint.

We see that there are 15 states for this simple example. When W and B are large, we are in trouble. In fact, the number of states is 2^W (when $B > W$). Another difficulty is that we are unable to draw a state transition diagram (no matter how complicated) for a system with general message arrival and network delay processes. This case will be discussed in the next section.

In Figure 2.3 we show the analytically calculated throughput and delay of a system with $W = 8$ and $B = 2,4,6,8$. The solid curves are for throughput, and the dashed curves are for delay. Note that as μ/λ becomes large, (i.e., the network delay becomes shorter than the message interarrival time), the throughput approaches λ and is not affected by the size of the buffer since the chance for messages to get out of order is small. In general, the smaller the buffer size and μ/λ, the smaller is the throughput and the larger is the delay since a lot of kicking out occurs at the buffer.

2.2 A More General Case

Now let us consider a more general case. The system is similar to that in Figure 2.1 except that the the network delay is generalized to an arbitrary one-way delay, X (X is a random variable), and the service time at the output channel is arbitrary and is described by a random variable Y with some distribution.

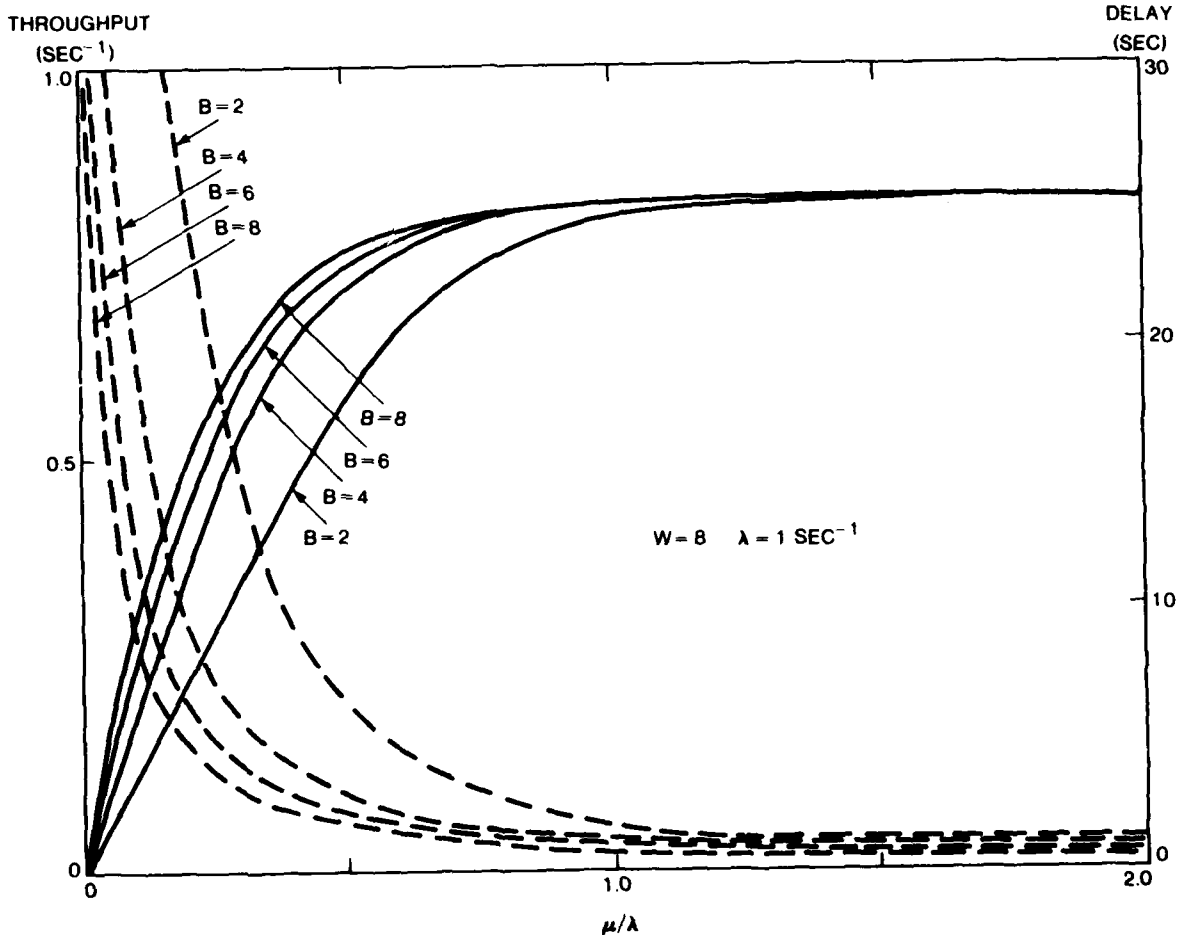


Figure 2.3. Throughput and delay curves for the system in Fig. 2.1.

No state diagram similar to that in Figure 2.2 can be drawn here. To exactly solve this generalized system is almost hopeless. It is more difficult than G/G/1 [KLEI 75] which itself is only solvable using very complicated means. Hence only an approximation (namely, curve fitting) is given here. We first look into the case $B = \infty$ and then we turn to the case, $B=0$. Moreover, we know that if there is no output channel the throughput is zero. We thus obtain three points for three different cases and then we fit a curve exponentially through these points.

Case 1: $B=\infty$

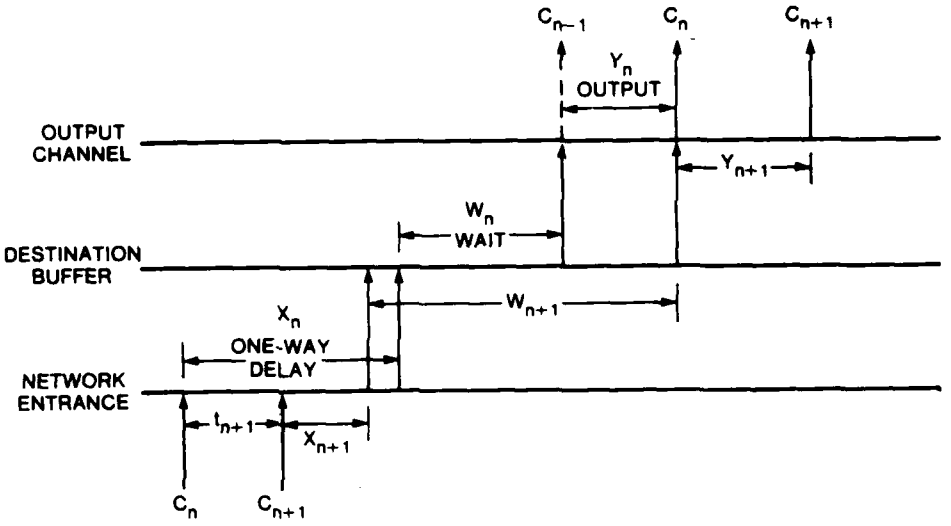


Figure 2.4. The time diagram when B=∞.

The system behavior can be described by the time diagram in Figure 2.4. Here, W_k is the random variable representing the waiting time of the k^{th} customer (accepted to the network) in the destination buffer of infinite size, and t_{j+1} represents the time duration between the j^{th} and the $j+1^{th}$ accepted customers. (Although we use the "W" to represent the token number, window size, and waiting time, the reader should not be confused as the meaning is clear from the context.)

To understand the time diagram shown in Figure 2.4, let us follow the accepted customer C_n. As he arrives at the entrance of the network, he secures a permit and is allowed to enter the network, where he is transmitted toward his destination. It takes him X_n units of time to arrive at the destination. He then joins the buffer B where he waits for W_n units of time to get in the right order, or for his turn to be delivered. It then takes the output channel another Y_n units of time to deliver him. C_{n+1} may arrive at the buffer before C_n (as shown in the figure). Thus, we have

$$W_{n+1} + X_{n+1} = [W_n + X_n + Y_n - t_{n+1}]^{+X_{n+1}} \tag{2.1}$$

where we define

$$[A]^{+B} \triangleq \begin{cases} A, & \text{if } A > B \\ B, & \text{if } A \leqslant B \end{cases}$$

By letting $W' = W + X$ we obtain

$$W_{n+1}' = [W_n' + Y_n - t_{n+1}]^{+X_{n+1}}$$

We assume that message $n+1$ is an "average message" so that X_{n+1} has the value $\bar{X}$. Then the above equation becomes

$$W_{n+1}' \cong [W_n' + U_n]^{+\bar{X}}$$

$$= \max[\bar{X},\ W_n' + U_n] \tag{2.2}$$

where $U_n \triangleq Y_n - t_{n+1}$. Observe that t_{n+1}, the interarrival time between accepted messages $n+1$ and n, is not independent of X_n (hence W_n'). This can be seen at the extreme case where we have only one token for the entire system so that if X_n is large t_{n+1} is also large. Yet the dependence becomes less and less prominent as W, the number of tokens, becomes larger and larger. Therefore, we assume that t_{n+1} is independent of W_n' (i.e., $W >> 1$). This assumption implies the independence of W_n' and U_n. Thus, a method similar to Kingman's Algebra for G/G/1 [KING 66] can be used to solve (2.2). That is, to find $w_{n+1}(w)$, the pdf of W_{n+1}, we calculate the pdf of $W_n + U_n$ by convolution, and sweep the total probability with arguments $< \bar{X}$ up to $\bar{X}$. The iteration continues until it converges [KLEI 75].

To calculate the throughput, S, suppose that we have W tokens for the source, and that these tokens travel between the source and the destination with a round trip delay equal to $DX = X + X$. Then [GROS 74] the system behaves like an M/G/W system with loss; thus,

$$P_i \triangleq Pr[i \text{ tokens are occupied by messages}]$$

$$= \frac{(\lambda \overline{DX})^i / i!}{\sum_{j=0}^{W} [(\lambda \overline{DX})^j / j!]} \tag{2.3}$$

Therefore,

$$S = \lambda(1 - P_W) \tag{2.4}$$

and the average message delay is

$$D = \bar{X} + \bar{W} + \bar{Y} \tag{2.5}$$

Observe that although the external message arrival process is Poisson, the accepted arrival process is not; yet it becomes Poisson when W approaches infinity.

With Equations (2.2), (2.3), (2.4) and (2.5) the system is solved for the case $B = \infty$.

Case 2: $B = 0$

We now have no buffer to store messages. A message is rejected when it arrives at the destination and finds that the output channel is busy. The "-ACK" goes back to the source (with delay X); the source retransmits the message and it may get rejected again. Thus, the message can be viewed as if it is "waiting in the network". When the output channel becomes free, the next copy of the message is somewhere in the network, and is assumed to arrive at the output channel after $\hat{DX}$ ("$\hat{}$" denotes the residual life of a random variable). This time diagram is shown in Figure 2.5. Now, with subscript n denoting the n^{th} message entering the network, we have

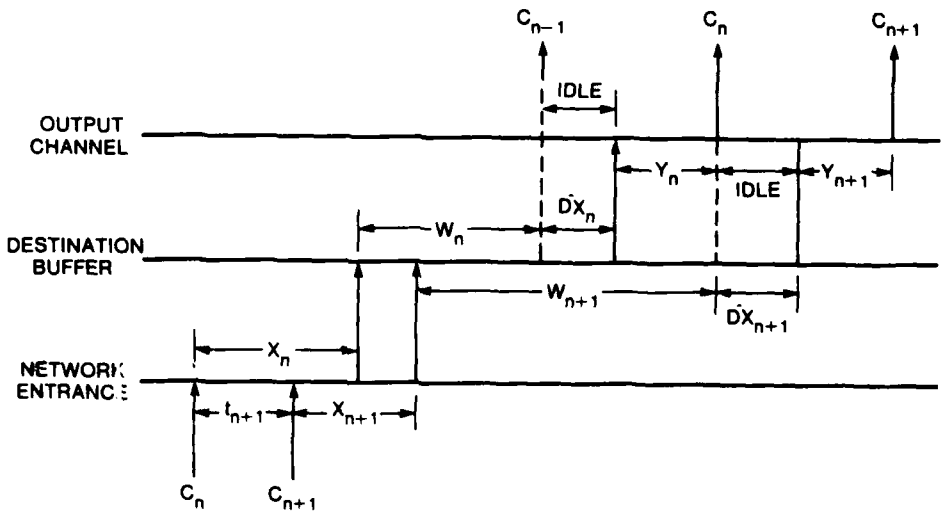


Figure 2.5. The time diagram when B=0.

$$W_{n+1} + X_{n+1} = [W_n + \Delta_{W_n} \cdot D\hat{X}_n + Y_n + X_n - t_{n+1}]^{+X_{n+1}}$$

where again,

$$[A]^{+B} = \begin{cases} A, & \text{if } A > B \\ B, & \text{if } A \leqslant B \end{cases}$$

and,

$$\Delta_{W_n} \triangleq \begin{cases} 1, & \text{if } W_n > 0 \\ 0, & \text{if } W_n \leqslant 0 \end{cases}$$

again define $W' \triangleq W+X$; then

$$\begin{aligned} W_{n+1}' &= [W_n' + \Delta_{W_n} \cdot D\hat{X}_n + Y_n - t_{n+1}]^{+X_{n+1}} \\ &\cong [W_n' + \Delta_{W_n} \cdot D\hat{X}_n + U_n]^{+\bar{X}} \\ &= \max [\bar{X}, W_n' + \Delta_{W_n} \cdot D\hat{X}_n + U_n] \end{aligned} \tag{2.6}$$

We see that $\Delta_{W_n} \cdot \hat{DX}_n$ represents the fact that $\hat{DX}_n$ is needed only if the n^{th} message is rejected at the destination when it arrives the first time (i.e., $W_n > 0$).

Even with the assumption that t_{n+1} is independent of W_n', Eq. (2.6) is still not solvable by the method described for the $B = \infty$ case because of the existence of $\Delta_{W_n}\hat{DX}_n$, which is again not independent of W_n'. For this, we assume that $\Delta_{W_n} \cong 1$. That is, we have a heavy traffic system (with zero buffers) so that when a message arrives at the destination the first time it almost always finds that the server is busy; and/or the distributions of X and t are such that messages easily get out of sequence. Then (2.6) becomes

$$W_{n+1}' = \max [\bar{X},\ W_n' + \hat{DX}_n + U_n] \tag{2.7}$$

We further assume that W_n' and $\hat{DX}_n$ are independent of each other. This is not a bad assumption since the server becomes free at a random instant, and the number of reject/retransmissions of message n (which largely contributes to the value of W_n) is also arbitrary. Eq. (2.7) then becomes

$$W_{n+1}' = \max [\bar{X}, W_n' + U_n'] \tag{2.8}$$

where

$$U_n' = U_n + \hat{DX}_n$$

Eq. (2.8) again can be solved by iteration. To calculate the average throughput we first observe that the average time period D that a token is occupied by a message is

$$D = \bar{X} + \bar{W} + \overline{\hat{DX}} + \bar{X} \tag{2.9}$$

where

$$\bar{W} = \overline{W'} - \bar{X}$$

Thus,

$$P_W = \Pr[\text{all } W \text{ tokens are occupied by messages}]$$

$$= [(\lambda D)^W / W!]/[\sum_{j=0}^{W} (\lambda D)^j/j!] \tag{2.10}$$

Therefore, the average throughput is

$$S = \lambda[1 - P_W] \tag{2.11}$$

and the message delay is

$$T = \bar{X} + \bar{W} + \overline{\hat{DX}} + \bar{Y} \tag{2.12}$$

With Equations (2.8), (2.9), (2.10), (2.11), and (2.12) we solve the system for case $B = 0$. under the assumptions described above.

In Figure 2.6, we show the calculated curves and the simulated results for the case $X \sim \mu e^{\mu x}$ and $Y \sim \gamma e^{\gamma y}$ with $\mu = 0.4$, $\gamma = 1.2$ and $\lambda = 1.0$. Note that the point $S+B=0$ on the horizontal axis corresponds to the case where we have no server; otherwise, $S=1$.

In Figure 2.7 we show the comparison between two systems where one requires sequencing and the other does not. In this figure, as $B \to \infty$ the throughput of both schemes matches nicely except when the window W is small. This discrepancy at small W occurs because the assumption that the duration between two successive acceptances of messages is

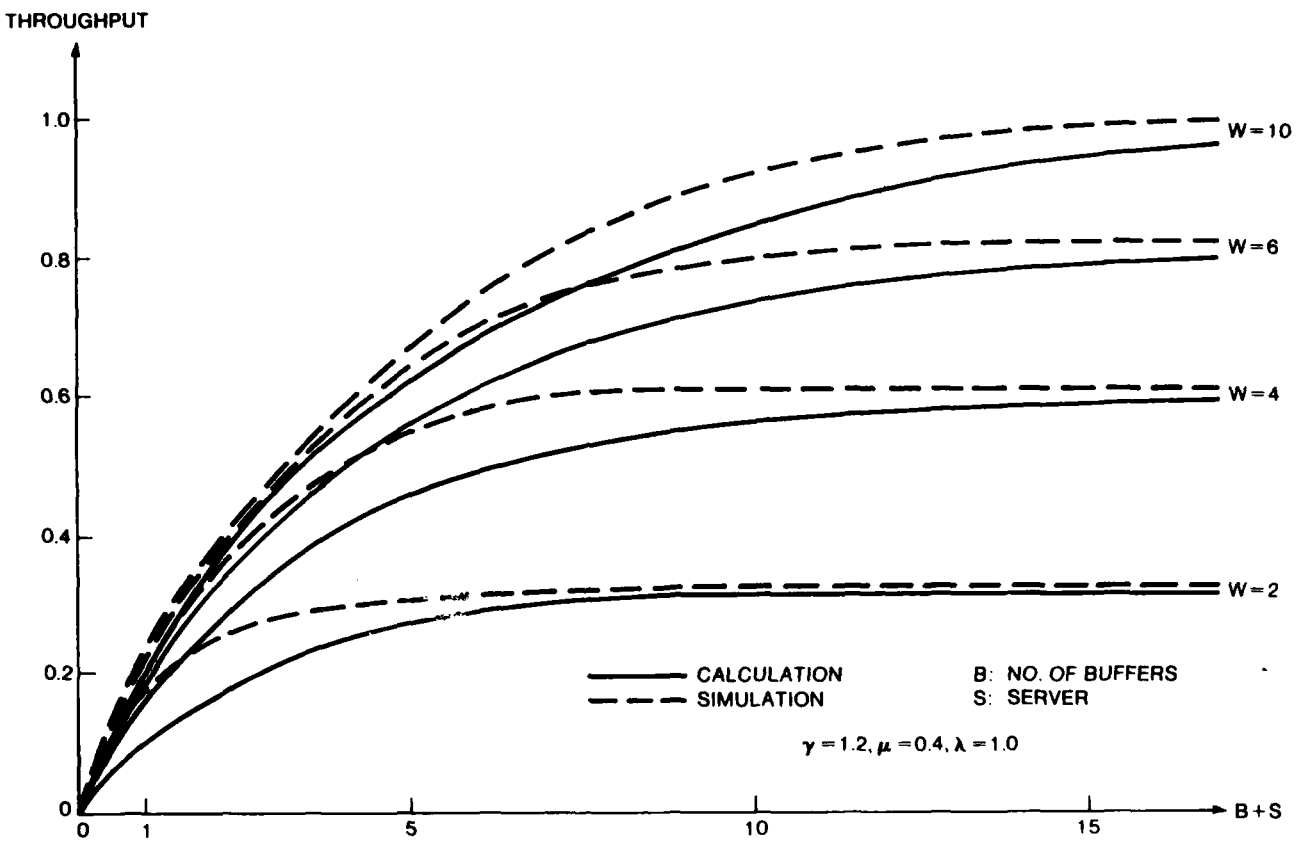


Figure 2.6. The calculated and simulated results for generalized sequencing problems.

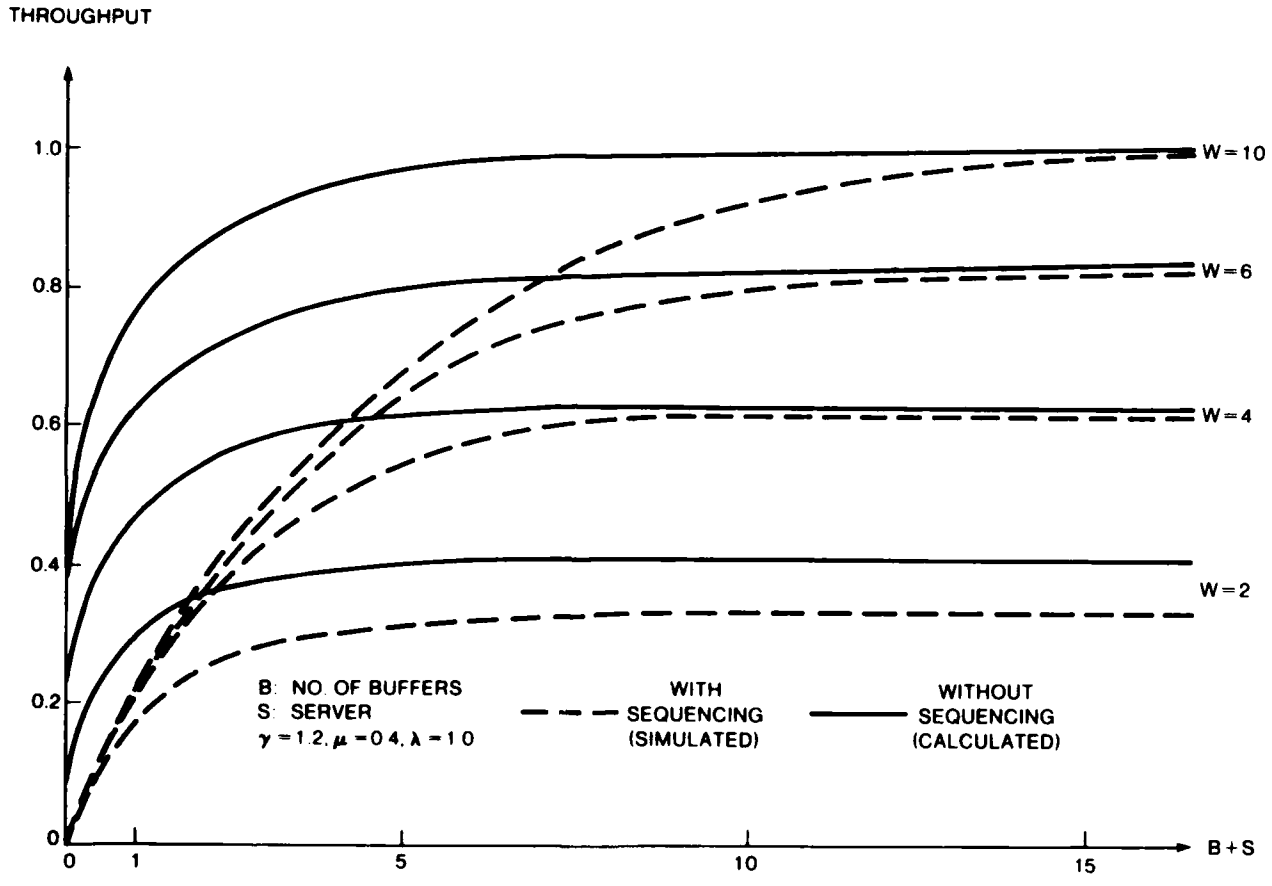


Figure 2.7. The difference in throughput between sequencing and non-sequencing.

exponential and is independent of one way message delay is no longer valid when W is small. We know that when both W and B are large the exponential and independence assumptions become correct and the rejection activity also becomes low. Therefore sequencing causes little reduction in throughput (yet it does cause an increase in delay because of the time required to reassume order).

The careful reader should have noticed that in the curve fitting approximation presented above, the "kicking out" mechanism (see (A4) and (A8) in Sec. 2.1) does not have a chance to play its role. We have only obtained approximated results for $B = \infty, 0$ with $S = 1$, and for $B = 0$ with $S = 0$ also. Yet for the "kicking out" mechanism to be active we need $B > 0$, finite, and $S > 0$.

Despite the weakness of our analysis, we believe that the approach we take here is valuable. Recently, in [KAMO 81] and [BARB 81] fruitful results have been obtained. Interested readers should refer to the above references.

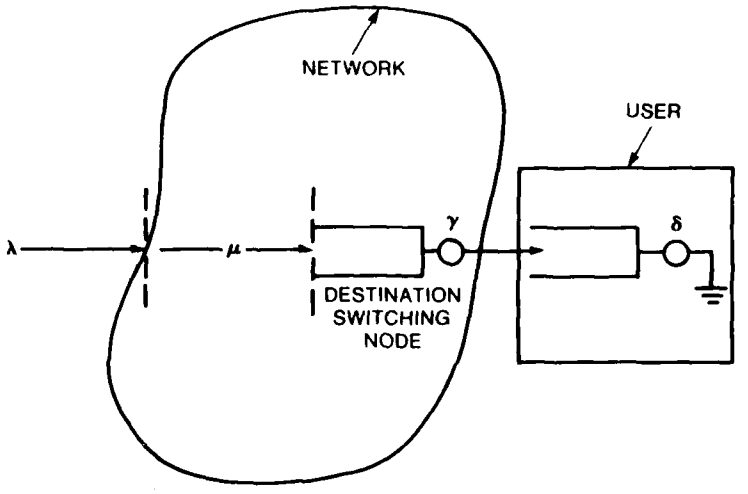


Figure 2.8. Message sequencing nodes.

Another interesting question is as follows: Consider the system in Figure 2.8. Assume that all processes are Poisson (or exponential). Messages enter the network with rate λ. Then they arrive at the destination node after an exponential delay at rate μ. The destination node then delivers (at rate γ) the messages to the user whose consumption rate is δ. The question to ask is which scheme is better - to carry out message sequencing at the destination node or at the user node.

Intuitively, the better place to resume order is at the node where the service rate is slower. This is because a message usually must wait here anyway to be served later and therefore there is little extra cost involved while waiting to resume the correct sequence. Yet the analytic proof is difficult because the departure process from the destination switching node is difficult to find. The interested reader may wish to pursue this.

CHAPTER 3
STATIC FLOW CONTROL BASED ON ASSIGNING A FIXED NUMBER OF PERMITS PER LOGICAL CHANNEL

In Section 1.2.1, we commented on a limitation of the Wong-Unsoy two-level control model; namely, it did not take into account the time needed to gather control information (e.g., the number of messages now in the network).

In this chapter we try to overcome this limitation by considering a new end-to-end flow control model. In this model, each logical channel (a source-destination pair) is provided with a fixed number of permits. A message must secure a permit at its originating node before it can enter the network. After the message reaches its destination, the accompanying permit is then carried back to the original node by the acknowledgement of that message. Thus, a source knows that a message has left the network only when it receives the ACK for that message.

3.1 System Assumptions

Consider a store-and-forward computer communication packet-switching network with any topology and any input traffic. We make the following assumptions.

(A1) Each node has an infinite number of buffers.

(A2) The external traffic arrival rate for the r^{th} logical channel is Poisson with a rate $\mu_e(r)$.

(A3) Each logical channel r is allowed to accommodate no more than L_r unacknowledged messages (i.e., logical channel r has L_r permits).

(A4) We have exponential service time for messages and the same exponential service time for acknowledgements.

(A5) We have a fixed routing scheme.

(A6) Kleinrock's independence assumption [KLEI 76] holds.

3.2 An Interesting Observation

The following observation enables us to transform an open network of queues (for messages) into a closed network of queues (for permits). Consider a queue of permits as shown in Fig. 3.1. Let us assume that the permit-arrival process G_p is an arbitrary one. The permits are queued; the permit at the head of the queue can leave the queue only when a message comes and (immediately) takes it out (when an arriving message finds no permit it will just turn away and never come back). If the arrival process of messages is Poisson, then we can model the queue for permits as a $G_p/M/1$ queueing system as in the lower half of Fig. 3.1.

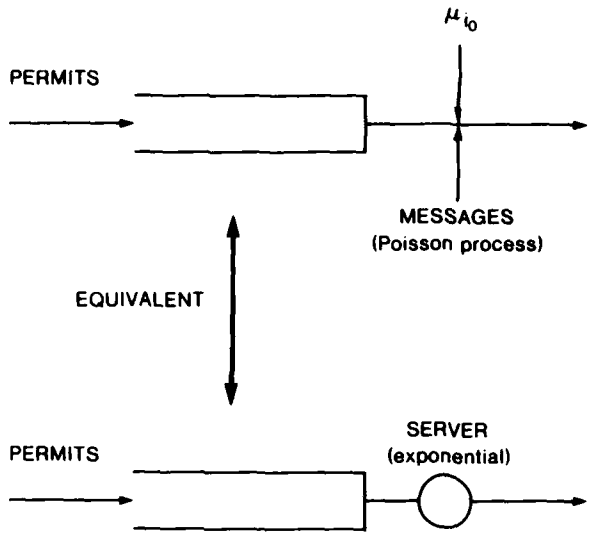


Figure 3.1. Two mathematically equivalent systems.

3.3 The Modeling of the Flow Control Scheme

With the above observation and assumptions we are able to model the flow control scheme as a closed network of queues [GORD 67] *for permits*. The idea is best illustrated by the following simple example. *Consider two connected nodes communicating with each other as* shown in Figure 3.2a, where $\mu_e(i)$, $i = 1,2$, is the external message arrival rate at node i. The queueing network in Figure 3.2b is the model of the flow control scheme.

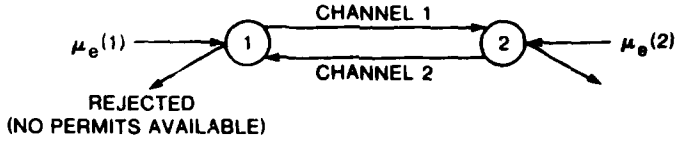
When a message arrives at its source node (say node 1), it must secure a permit (if no permit is available, it is rejected and never returns). After it secures a permit it then enters a queue (queue C in Fig. 3.2b) waiting to be transmitted over channel 1. An accepted message is finally sent to its destination (node 2), where the accompanying permit immediately becomes free, and is sent back with the acknowledgement to node 1 (through queue D into queue A). Thus we have a closed network of queues for the permits (fixed in number), and an open network of queues for messages.

This model is more realistic since the time that permits take to return to the sources is taken into account (remember that Wong-Unsoy model mentioned in Chapter 1 does not incorporate this in the analysis).

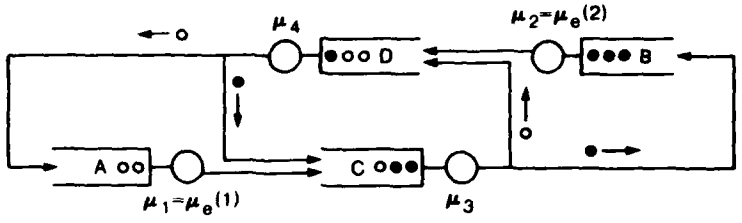
In Figure 3.2b we have two closed chains of permits. One is chain ACDA, the other is BDCB. Chain ACDA is for permits used by node 1 to node 2 messages. Chain BDCB is for permits used by node 2 to node 1 messages. We immediately see that the message throughput (from node 1 to node 2) is

$$\mu_e(1) \cdot Pr \text{ [queue A not empty of permits]} \tag{3.1}$$

and the one-way delay for accepted messages is the system time in queue C. As described in [BASK 75], we can denote a state of the system as



(a)



(b)

Figure 3.2. Illustrative example for the flow control scheme.

$$S = [Z_A, Z_B, Z_C, Z_D]$$

with

$$Z_K = (x_{K_1}, x_{K_2}, \cdots, x_{K_{n_K}})$$

$K = A, B, C, D$, and

$x_K = 1$ or 2

depending on whether the permit x_K belongs to logical channel 1 (node 1 to node 2), or logical channel 2 (node 2 to node 1); and its position in Z_K denotes its actual position in queue K.

Thus, S can be written as

$$S = \left[\begin{bmatrix} x_{A_1} \\ \vdots \\ x_{A_{n_A}} \end{bmatrix}, \begin{bmatrix} x_{B_1} \\ \vdots \\ x_{B_{n_B}} \end{bmatrix}, \begin{bmatrix} x_{C_1} \\ \vdots \\ x_{C_{n_C}} \end{bmatrix}, \begin{bmatrix} x_{D_1} \\ \vdots \\ x_{D_{n_D}} \end{bmatrix} \right]$$

The server at queue K is serving x_{K_1}, the head of the queue. Let $L_1 = L_2 = 2$ for simplicity. Then one of our balance equations becomes

$$P\left[\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}1\\2\end{pmatrix},\begin{pmatrix}2\\1\end{pmatrix}\right](\mu_3+\mu_4)$$

$$= P\left[\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}1\\1\end{pmatrix},\begin{pmatrix}2\\2\\1\end{pmatrix}\right]\mu_4$$

$$+ P\left[\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}1\\1\\2\end{pmatrix},\begin{pmatrix}2\\1\end{pmatrix}\right]\mu_3 \tag{3.2}$$

If we let $y_A(1)$, $y_B(2)$ and $y_C(i)$, $y_D(i)$, $i = 1, 2$ be the solution of the following four equations:

$$\mu_1 y_A(1) = \mu_4 y_D(1) = \mu_3 y_C(1) \tag{3.3}$$

and

$$\mu_2 y_B(2) = \mu_3 y_C(2) = \mu_4 y_D(2) \tag{3.4}$$

then it is easy to see that the solution to Equation (3.2) is

$$G \cdot y_C(1)y_C(2)y_D(2)y_D(1) \quad \text{for } P\left[\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}1\\2\end{pmatrix},\begin{pmatrix}2\\1\end{pmatrix}\right],$$

$$G \cdot y_C(1)y_D(2)^2 y_D(1) \quad \text{for } P\left[\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}1\\1\end{pmatrix},\begin{pmatrix}2\\2\\1\end{pmatrix}\right],$$

and

$$G \cdot y_C(1)^2 y_C(2)y_D(2) \quad \text{for } P\left[\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}0\\0\end{pmatrix},\begin{pmatrix}1\\1\\2\end{pmatrix},\begin{pmatrix}2\\1\end{pmatrix}\right].$$

In general we have R logical channels and N transmission channels, and each logical channel r has L_r permits. The state is

$$S = [Z_1, Z_2, \cdots Z_N]$$

where $Z_i = \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{iL_i} \end{pmatrix}$, with $x_{ij} = 1, 2, \cdots R$

The general balance equation is

$$P[Z_1, \cdots, Z_i, \cdots, Z_j, \cdots, Z_N]\left[\sum_{k=1}^{N} \mu_k \cdot \delta(n_k)\right]$$

$$= \sum_{i=1}^{N}\sum_{j=1}^{N} P[Z_1, \cdots, Z_i(-), \cdots, Z_j(+V_{ij}) \cdots Z_N] \cdot \delta(n_i)\mu_i \phi_i(V_{ij}) \tag{3.5}$$

where

$$\delta(n) \triangleq \begin{cases} 1, \text{ if } n > 0 \\ 0, \text{ otherwise} \end{cases}$$

$$Z_i(-) \triangleq \begin{pmatrix} X_{i_1} \\ \vdots \\ X_{i_{n_i-1}} \end{pmatrix}, \quad Z_i(+X_{n_i}) \triangleq \begin{pmatrix} X_{n_i} \\ X_{i_1} \\ \vdots \\ X_{i_{n_i}} \end{pmatrix}$$

and

$$\phi_{ji}(r) = \begin{cases} 1, & \text{if messages of logical channel } r \text{ are routed} \\ & \text{from physical channel } j \text{ to physical channel } i \\ 0, & \text{otherwise} \end{cases}$$

Let $y_i(r)$, $i = 1, 2, \cdots, N$; $r = 1, 2, \cdots, R$ be the solution of the following equations:

$$\mu_i y_i(r) = \sum_{j=1}^{N} \mu_j y_j(r) \phi_{ji}(r) \tag{3.6}$$

Then the solution of Eq. (3.5) is

$$P(Z_1, \cdots, Z_N) = G \prod_{i=1}^{N} \prod_{k=1}^{R} y_i(x_{ik}) \tag{3.7}$$

where G is a normalization constant that must be adjusted to force the probabilities to sum to unity.

Thus the system is completely solved if the value of G is calculated.

The above flow control model, as innocent as it may look, suffers from great computational difficulty in determining the value of key parameter G, the normalization constant. This is the usual problem with closed queueing networks.

For an open network of queues, G has a simple closed form expression. Yet for a closed network of queues, G must be obtained by summing all the feasible states. Because of the combinatorially growing state space, a naive summation is out of the question except for trivially small networks. Although some efficient algorithms exist in [BUZE 73], [REIS 75], [REIS 76], [REIS 77] and [REIS 78], the computational complexity is still too much to be practical.

For example, in [REIS 76], the normalization constant is given by

$$\begin{aligned} G &= \Pr[\text{population in the corresponding open network} = K] \\ &= \Pr[k_1 + k_2 + \cdots + k_N = K] \\ &= \Pr[k_1] * \Pr[k_2] * \cdots * \Pr[k_N] \end{aligned}$$

where k_i is the queue size of q_i, $Pr[k_i]$ is the queue size probability of queue i, and the asterisk denotes convolution.

One sees that these convolutions are all complex and moreover the intermediate results may easily exceed the floating-point range of most computers even though the final results are all reasonable in magnitude. Had Wong applied his model to a network with a few more nodes than the simple network in his paper [WONG76], he would have found himself confronted by a

solid wall of computational complexity.

This difficulty had made such models infeasible until Martin Reiser's breakthrough in which he invented a heuristic method by using a mean value analysis [REIS 78]. We describe this method in the rest of this chapter.

3.4 The Mean Value Analysis and the Heuristic Solution

3.4.1 Mean Value Analysis in a Closed Cyclic System

The ideas behind the mean value analysis are easiest to describe by using the example of a closed cyclic queue. We denote by $S(K)$ the queueing system with K customers. Furthermore, denote

K: Number of customers in the system

N: Number of queues in the cyclic chain

τ_i: Mean service time of queue i

t_i: Mean system time of queue i (including service)

n_i: Mean queue size of queue i (including customer in service)

λ: Throughput of the chain (= circulation rate)

We will use parenthetical arguments if we wish to emphasize that quantities are for the system with population size K; viz $n_i(K)$, $t_i(K)$ and $\lambda(K)$ are the mean queue size, the mean system time and the throughput of S(K). We can straightforwardly write the equations

$$t_i(K) = \tau_i + \tau_i \cdot [\text{mean number of customers seen upon arrival}] \tag{3.8}$$

$$\lambda(K) = \frac{K}{\sum_{i=1}^{N} t_i(K)} \tag{3.9}$$

$$n_i(K) = \lambda(K) t_i(K) \tag{3.10}$$

Equation (3.8) states that an arriving customer, on the average, must wait for its own service time plus the backlog of work seen upon arrival at queue i. Equations (3.9) and (3.10) are simply Little's formula applied to the entire chain and to each queue respectively. Note that the average number of customers is of course K. If we had an expression for the bracketed term in Eq. (3.8) we could solve for all unknown quantities. Such an expression is found using

Theorem 1: *In a closed queueing network with product-form solution, the probability that an arriving customer finds state* $\mathbf{k}$ *in the system* $S(K)$ *is the same as the long term equilibrium probability of* $\mathbf{k}$ *in* $S(K-1)$.

From the theorem immediately follows the mean waiting time equation

$$t_i(K) = \tau_i + \tau_i n_i(K-1) \tag{3.11}$$

Equation (3.11) together with (3.9) and (3.10) can be solved easily in a recursive manner,

$$n_i(0) = 0 \tag{3.12}$$

$$t_i(K) = \tau_i[1+n_i(K-1)] \tag{3.13}$$

$$\lambda(K) = \frac{K}{\sum_i t_i(K)}, \quad K > 0,\ i=1, 2, \cdots, N \tag{3.14}$$

$$n_i(K) = \lambda(K)t_i(K) \tag{3.15}$$

3.4.2 Generalization to Networks with Many Closed Chains

The recursion (3.12) to (3.15) generalizes easily to a network with a full routing matrix P. Let i^* denote an arbitrarily chosen queue. Then, the quantity θ_i is defined by

$$\theta_{i^*} = 1 \tag{3.16}$$

$$\underline{\theta} = \underline{\theta}P \tag{3.17}$$

where

$$\underline{\theta} = [\theta_1, \theta_2, \cdots, \theta_N]$$

measures the average number of visits a customer makes to queue i between successive visits to the marked queue i^*. Since the number of visits and the throughput are proportional, θ_i also measures the throughput at queue i, λ_i, in units of λ_{i^*}, the throughput of queue i^*, viz

$$\lambda_i = \theta_i\lambda_{i^*}, \quad i=1, 2, \cdots, N \tag{3.18}$$

If t_i measures the system time across queue i, then obviously the average time a customer needs between two successive departures from queue i^* is given by the sum of the products of the form [average number of visits] $\times$ [mean waiting time], namely

$$\sum_{i=1}^{N} \theta_i t_i \tag{3.19}$$

For simplicity of notation, define $\lambda^* = \lambda_{i^*}$. Now Equations (3.13) to (3.15) translate into

$$t_i(K) = \tau_i[1+n_i(K-1)] \tag{3.20}$$

$$\lambda^*(K) = K/\sum_i \theta_i t_i(K) \tag{3.21}$$

$$n_i(K) = \lambda^*(K)\theta_i t_i(K) \tag{3.22}$$

Define $t_i^* = \theta_i t_i$ and as usual traffic intensities $\rho_i = \theta_i\tau_i$. Then we find

$$t_i^* = \rho_i[1+n_i(K-1)] \tag{3.23}$$

$$\lambda^* = K/\sum_i t_i^*(K) \tag{3.24}$$

and

$$n_i(K) = \lambda^*(K)t_i^*(K) \tag{3.25}$$

Equations (3.12) and (3.23) to (3.25) constitute the recursion to evaluate all quantities of $S(K)$ with general routing. They are of the same form as (3.13) to (3.15).

Next we shall discuss the multi-chain case. We assume that there are R closed routing chains. We will use superscripts r ($r = 1, 2, \cdots, R$) to denote that a given quantity belongs to chain r. We do not consider the case where customers change class membership within chains. Note, however, that as shown in Eq. (3.13), this is only a trivial reduction of generality. Also we shall drop the asterisks used in (3.23) to (3.25). As before, we assume constant service rates. We introduce the following notation:

$\underline{K} = (K^1, K^2, \cdots, K^R)$: population vector,

$\underline{K} - \underline{e}_r = (K^1, K^2, \cdots, K^{r-1}, K^r - 1, \cdots, K^R)$: population vector with one less customer in chain r,

R(i): Set of chains visiting queue i,

Q(r): Set of queues in chain r,

λ^r: Throughput of marked queue in chain r,

τ_i^r: Mean service time of a chain r customer in queue i,

t_i^r: Mean time a chain r customer spends at queue i between successive visits to the marked queue of chain r,

n_i^r: Mean number of chain r customers at queue i,

$n_i = \Sigma_r n_i^r$: Mean queue size of queue i,

$\rho_i^r = \theta_i^r \tau_i^r$: Traffic intensity of chain r at queue i,

θ_i^r: Mean number of visits a chain r customer makes to queue i between successive visits to an arbitrarily marked queue in chain r.

The mean waiting time equations follow from Theorem 1.

$$t_i^r(\underline{K}) = \rho_i^r + \theta_i^r \Sigma_r \tau_i^r n_i^r(\underline{K} - \underline{e}_r) \tag{3.26}$$

The first term in Eq. (3.26) is the customer's own service demand and the sum measures the backlog of work he encounters upon arrival.

The recursion now follows, analogous to (3.13) to (3.15) or (3.23) to (3.25) as:

$$t_i^r(\underline{K}) = \rho_i^r [1 + n_i(\underline{K} - \underline{e}_r)] \tag{3.27}$$

$$\lambda^r(\underline{K}) = K^r / \sum_{i \in Q(r)} t_i^r \tag{3.28}$$

$$n_i(\underline{K}) = \sum_{r \in R(i)} \lambda^r(\underline{K}) t_i^r(\underline{K}) \tag{3.29}$$

where (as usual) $i = 1, 2, \cdots, N$; $r = 1, 2, \cdots, R$; $K \geq 0$; and $n_i(0) = 0$

Equations (3.27) and (3.29) allow an easy recursive evaluation of unknown quantities.

3.4.3 The Heuristic Solution

The complexity of recursion (3.27) to (3.29) for both storage and operation is of the order

$$\prod_{r=1}^{R} K^r \tag{3.30}$$

Obviously, it will be impossible to solve for more than a small number of chains (clearly less than 10). The following heuristic method, derived from the mean value analysis, overcomes the complexity barrier of Eq. (3.30).

We note that the recursion is the source of the complexity. It is the goal of the heuristic method to replace the recursion by an iteration that is performed at the point K only. We define quantities ξ_i^r as follows

$$\xi_i^r(K) = n_i(K) - n_i(K-e_r) \tag{3.31}$$

Thus, ξ_i^r measures how the customer added to $S(K-e_r)$ is distributed over the individual queues.

Obviously we have (all i = 1, 2, ... , N, r = 1, 2, ... , R)

$$\sum_{i=1}^{N} \xi_i^r = 1 \tag{3.32}$$

and also

$$0 \leqslant \xi_i^r \leqslant 1 \tag{3.33}$$

Suppose now that we had a function that would yield ξ_i^r in terms of quantities of S(K), for example

$$\xi_i^r = f(i, r, \{\lambda^j,\ j = 1, 2, \cdots, R\}) \tag{3.34}$$

where we have omitted the arguments denoting the chain population. We may now rewrite Eqs. (3.27) to (3.29), again dropping the arguments (K), as follows

$$\xi_i^r = f(i, r, \{\lambda^j\}) \tag{3.35}$$

$$t_i^r = \rho_i^r[1+n_i-\xi_i^r] \tag{3.36}$$

$$\lambda^r = K^r / \sum_{i \in Q(r)} t_i^r \tag{3.37}$$

$$n_i = \sum_{j \in R(i)} \lambda^j t_i^j \tag{3.38}$$

We have now obtained a nonlinear system of equations that is independent of K, the source of the high operation count in Eq. (3.30). We would solve Eqs. (3.35) to (3.38) by a simple iteration method, starting with initial values for n_i and λ^r (i = 1, 2, ... , N, r = 1, 2, ... , R) and then iterating through Eqs. (3.35) to (3.38) in a cyclic fashion until convergence is obtained (or divergence established).

In its exact form, the function (3.34) is no less complex than the original problem. Hence, it is our goal to obtain an efficient heuristic for Eq. (3.34). The proposed heuristic will coalesce the R chains into a single chain that can be solved efficiently by the recursion (3.23) to (3.25).

If a customer is removed from a chain, then all of the values n_i^r, $i = 1, 2, \cdots, N$, $r = 1, 2, \cdots, R$ are affected. However, since

$$\sum_{i \in Q(j)} n_i^j(K) - n_i^j(K - e_r) = 0 \text{ for all } j \neq r \tag{3.39}$$

(where r is the chain with one less customer) we may assume that ξ_i^r is affected mostly by chain r. Therefore, we estimate ξ_i^r from a single chain problem with redefined parameters. The capacity of queue i's server devoted to chains $j = 1, 2, \cdots, R$, $j \neq r$ is given by

$$\sum \lambda^j \tau_i^j$$

where the sum is over $j = 1, 2, \cdots, R$ but $j \neq r$. Taking the point of view of the fluid approximation of queueing systems, we may agree that a chain r customer "sees" a server with a reduced rate given by Eq. (3.40) below. Thus his mean service times are not τ_i^r but adjusted values

$$\hat{\tau}_i^r = \frac{\tau_i^r}{(1 + \lambda^r \tau^r - \sum_{i \in R(i)} \lambda^j \tau_i^j)} \tag{3.40}$$

Suppose that $\hat{n}_i^r(K)$ $(i = 1, 2, \cdots, N)$ are the mean queue sizes of a single chain queueing problem with population K and with parameters given by Eq. (3.40). Then we set

$$\xi_i^r = \hat{n}_i(K^r) - \hat{n}_i(K^r - 1) \tag{3.41}$$

$\hat{n}_i(K^r)$ and $\hat{n}_i^r(K^r - 1)$ are obtained from Equations (3.12) to (3.15). This completes the heuristic method.

The computational complexity of one iteration step through Eqs. (3.35) to (3.38) is now

$$\sum_{r=1}^{R} K^r \tag{3.42}$$

clearly an affordable effort even for a large number of closed chains with sizable populations.

Note that the heuristic method provided by Reiser has reduced the computational complexity tremendously. For an R-chain network, if N_i is the number of queues in the i^{th} chain, then the original complexity in determining G is

$$\prod_{i=1}^{R} \binom{N_i + K_i - 1}{N_i - 1}$$

where K_i is the total number of tokens in chain i. The above parenthesis represents the total number of ways to put K_i tokens in N_i queues. With Reiser's heuristic method this complexity reduces to that in Eq. (3.42).

We leave Chapter 3 here without giving an example of applying this control scheme. Dr. Reiser gives examples in great detail and therefore we need not repeat the same effort here. The interested reader may refer to [REIS 78].

CHAPTER 4
STATIC FLOW CONTROL BASED ON LIMITING THE PERMIT GENERATION RATES

In this chapter we present a new class of flow control techniques. Two schemes are proposed and their models analyzed. Both are based on limiting the permit generation *rates* of logical channels. It is found that the second model gives a performance very close to the ideal. Both schemes are very easy to implement; more importantly they involve no computational complexity (one can calculate the entire network performance on the back of an envelope) and other network features such as "piggy-back" acknowledgements and "time-outs" can easily be incorporated in the models. It turns out that the schemes can be modeled by Jackson's open network of queues [JACK 63] and the network of queues presented by Baskett et al. [BASK 75]. The first scheme serves as an introductory one which has an inherent shortcoming. The second scheme overcomes this shortcoming and appears to be an "ideal" flow control method.

4.1 The Currency Circulation Model - Scheme 1

The name of this scheme comes from the analogy between the way that the scheme controls the flow in a network (by properly generating, circulating and destroying permits) and the way that our society controls the flow of commodities (by issuing, circulating and destroying currency). Fortunately we manage to resist the temptation to inflate the currency!

4.1.1 Description of the Scheme

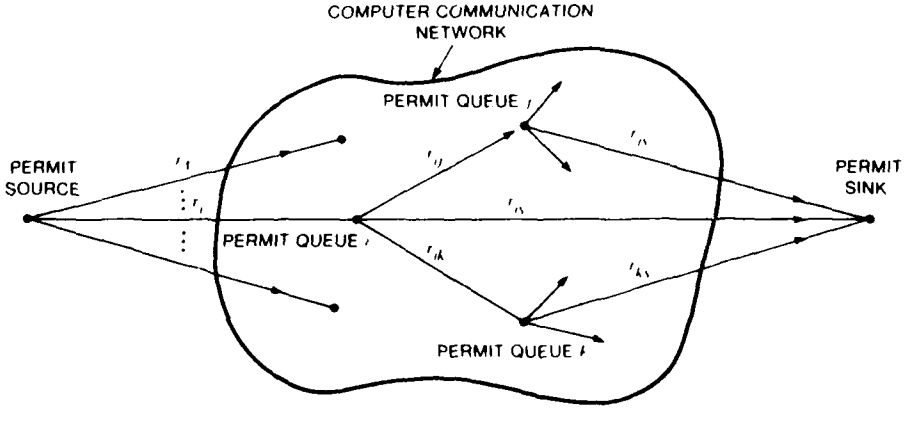


Figure 4.1 The currency circulation model.

Consider the computer communication network shown in Figure 4.1. This network may be described as follows:

(1) The network consists of a number of nodes inter-connected by communication channels.

(2) There is an external source of permits which generates permits at a rate g. Assume that the generation process is Poisson. Further, assume these permits are distributed to *permit queue* i with a probability r_i. Permit queues are associated with physical channels. For example, permit queue i is associated with the physical channel i which connects node x to node y. The permits in queue i are to be used by messages which *originate* at node x and are *routed* through node y as the first hop on the way to their eventual destination.

(3) An externally arriving message at a node must secure a permit from the corresponding permit queue at that node before it can be launched into the network on the first leg of its journey. If the message finds no permit, it is immediately rejected and "lost". Note that externally arriving messages may not queue at their origin while awaiting entry into the net. Once a message is accepted into the net it is stored-and-forwarded over some fixed route to its destination. We assume that messages arrive at logical channel jk (origin node j, destination node k) according to a Poisson process at rate γ_{jk}. (A logical channel is associated with each origin-destination pair).

(4) There is also a sink which destroys any permits which find their way to the sink.

(5) After launching a message into the network, a permit at queue i is then immediately passed to a neighbor permit queue j with probability r_{ij} (where queue j could be the sink).

(6) To prevent stalling of extra permits at queue i when the message traffic is low, we have a basic *circulation* rate μ_{i_0} to pump the permits out of queue i. That is, when there is zero message traffic, the permit at the head of the permit queue i will remain there for an exponentially distributed length of time with mean $1/\mu_{i_0}$ (after which it will move to neighbor permit queue j with probability r_{ij}). This basic rate is added to the permit circulation rate because of message arrivals.

Before going any further, let us give an example to see how the control scheme can be modeled for a network.

Figure 4.2 shows a simple 5-node computer communication network. Compare this with the permit queue model in Figure 4.3. Note that we have ten physical communication channels (i.e., five full-duplex channels) and twenty logical (origin-destination) channels. Each physical channel corresponds to a permit queue. In Figure 4.3 we have a conceptual source and a conceptual sink. The permits generated at the source (with rate g) are distributed to queue i with probability r_i. Permits at queue i are removed at a rate

$$\mu_{P_i} = \mu_{i_0} + \sum_{r \in O_i} \gamma_r$$

where γ_r is the external message arrival rate of logical channel r and O_i is the set of logical channels which originate at x and are routed through y as their first hop enroute to their destination (Nodes x,y are connected by physical channel i. See assumption (2) in this section).

After its removal from queue i, a permit enters queue j with probability r_{ij}.

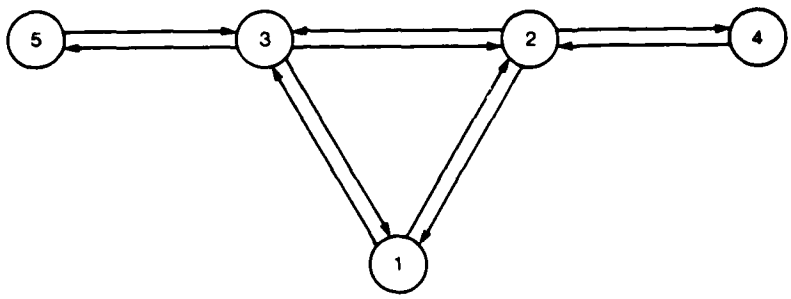


Figure 4.2. A simple computer communication network.

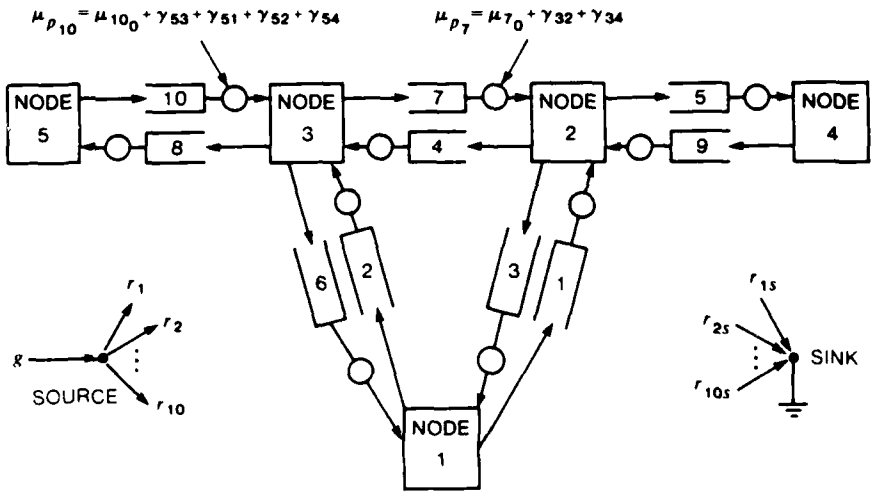


Figure 4.3. The currency circulation model of the simple network in Fig. 4.2.

As an example, let us assign the following values to the r_i's and r_{ij}'s:

$$r_i = 0.1 \qquad \forall \text{ queue } i, \quad i = 1,2,\cdots,10$$

and

$$r_{ij} = \frac{1}{n_i} \qquad \forall\, i = 1,2,\cdots,10\,.$$

where n_i = 1 + the number of adjacent queues. (The extra "one" is for the sink.) We shall continue this example later.

4.1.2 The Queueing Model for Permits

Let γ_{jk} be the external input rate of messages to logical channel jk (i.e., the logical path between node j and node k). The external arrival rate of permits at queue i is $g_i = gr_i$ and the departure rate of permits at queue i is

$$\mu_{p_i} = \mu_{i_0} + \gamma_i \tag{4.1}$$

where $\gamma_i \triangleq \sum_{r \in O_i} \gamma_r$.

The total arrival rate of permits to queue i (from outside the network and from other queues) is

$$\lambda_{p_i} = gr_i + \sum_{j=1}^{N} \lambda_{p_j} r_{ji} \tag{4.2}$$

where g is the total permit generation rate.

Thus, we have a Jackson-type network of M queues for permits. The joint probability for the number of permits distributed at all of the queues is then [KLEI 75]

$$P_p[n_1, n_2, \ldots, n_M] = P_{p_1}(n_1) P_{p_2}(n_2) \cdots P_{p_M}(n_M) \tag{4.3}$$

where n_i denotes the number of permits in queue i, and $P_{p_i}(n_i)$ is the probability that queue i has n_i permits. $P_{p_i}(n_i)$ is obtained from the solution to an M/M/1 queue with input rate λ_{p_i} as in Eq. (4.2) and output rate μ_{p_i} as in Eq. (4.1), i.e.,

$$P_{p_i}(n_i) = (1 - \rho_{p_i}) \rho_{p_i}^{n_i} \quad \text{where } \rho_{p_i} = \lambda_{p_i} / \mu_{p_i}$$

Observe that the permits do not necessarily enter or leave a queue according to a Poisson process. However, we shall assume this (and point it out as we do so) whenever it is convenient.

4.1.3 The Queueing Model for Messages

Observe that each logical channel r is associated with a queue at its originating node (e.g., in Figure 4.3, logical channel 5-4 is associated with queue 10 at node 5).

Denote the throughput (message/sec) of logical channel r as S_r. Then,

$$S_r = [1 - P_{p_i}(0)] \gamma_r = \left(\frac{\lambda_{p_i}}{\mu_{p_i}} \right) \gamma_r = \frac{\lambda_{p_i}}{\mu_{i_0} + \gamma_i} \gamma_r \tag{4.4}$$

where $P_{p_i}(0)$ is the probability of no permit in queue i such that $r \in O_i$.

After entering the network, a message is routed through a set of nodes and physical channels to its destination. Upon arriving at a node, the message joins the message queue associated with the next physical channel in its path, and waits there for transmission through the channel.

We describe the state of the system for messages by using "customer classes" as in [BASK 75]. Each class corresponds to a logical channel. In particular, consider the vector of column vectors

$$\left[\left(\begin{matrix}K_{11}\\K_{12}\\\vdots\\K_{1R}\end{matrix}\right) \cdots \left(\begin{matrix}K_{i1}\\K_{i2}\\\vdots\\K_{iR}\end{matrix}\right) \cdots \left(\begin{matrix}K_{M1}\\K_{M2}\\\vdots\\K_{MR}\end{matrix}\right)\right]$$

where

(a) M is the total number of physical channels

(b) R is the total number of logical channels

(c) Column vector i denotes the state of message queue i corresponding to physical channel i

(d) K_{ir} $(r = 1, 2, \cdots, R)$ denotes the number of messages at queue i from logical channel r, regardless of their actual relative positions with respect to other messages in queue i.

If we assume that the messages enter the network from a Poisson process,* then

$$P\left[\left(\begin{matrix}K_{11}\\\vdots\\K_{1R}\end{matrix}\right) \cdots \left(\begin{matrix}K_{M1}\\\vdots\\K_{MR}\end{matrix}\right)\right] = G \prod_{i=1}^{M} \frac{K_i!}{K_{i1}! \cdots K_{iR}!} \prod_{r=1}^{R} \left[\frac{\xi_{ir}}{\mu C_i}\right]^{K_{ir}} \tag{4.5}$$

where

G is the appropriate normalization constant, and

$$K_i = \sum_{r=1}^{R} K_{ir} \tag{4.6}$$

C_i is the service rate of channel i (bits/sec),

$1/\mu$ is the average message length (bits)
which is assumed to have an exponential distribution,

and

$$\xi_{ir} = \begin{cases} S_r, & \text{if messages of logical channel } r \text{ are routed through physical channel } i \\ 0, & \text{otherwise} \end{cases}$$

We also define $0^0 = 1$, for convenience.

---

* It is easy to prove that in our system, if permits arrive at permit queues from Poisson processes, then messages enter the network with Poisson processes. Unfortunately, as mentioned in Sec. 4.1.2, permits do not arrive at queues from Poisson processes and hence messages do not enter the network with Poisson processes.

Let $\rho_{ir} \triangleq \frac{\xi_{ir}}{\mu C_i}$ for $i = 1, 2, \ldots N$, $r = 1, 2, \ldots, R$. Then,

$$P\left[\begin{bmatrix} K_{11} \\ \vdots \\ K_{1R} \end{bmatrix} \cdots \begin{bmatrix} K_{M1} \\ \vdots \\ K_{MR} \end{bmatrix}\right] = G \prod_{i=1}^{M} \frac{K_i!}{K_{i1}! \cdots K_{iR}!} \prod_{r=1}^{R} \rho_{ir}^{K_{ir}}$$

Therefore,

$$G^{-1} = \sum_{\text{all feasible states}} \prod_{i=1}^{M} \frac{K_i!}{K_{i1}! \cdots K_{iR}!} \prod_{r=1}^{R} \rho_{ir}^{K_{ir}}$$

$$= \prod_{i=1}^{M} \sum_{K_i=0}^{\infty} \left(\rho_{i1} + \rho_{i2} + \cdots + \rho_{iR}\right)^{K_i}$$

The last equality can be justified by proving that for any term in the right hand side, there is an equivalent term in the left hand side and vice versa. For example, $\prod_{i=1}^{M} \rho_{i1}^2$ is a term belonging to the right hand side, but it corresponds to the feasible state

$$\left[\begin{bmatrix} 2 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} 2 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \cdots \begin{bmatrix} 2 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}\right]$$

and therefore it is a term in the left hand side.
Define

$$\rho_i = \sum_{r=1}^{R} \rho_{ir}$$

Then

$$G^{-1} = \prod_{i=1}^{M} \sum_{K_i=0}^{\infty} \rho_i^{K_i} = \prod_{i=1}^{M} \frac{1}{1-\rho_i}$$

i.e.,

$$G = \prod_{i=1}^{M} (1-\rho_i) \tag{4.7}$$

Let $P_i(K_i)$ denote the marginal probability that the i^{th} physical channel has K_i messages, then [BASK 75]

$$P_i(K_i) = \sum_{K_1=1}^{\infty} \cdots \sum_{K_{i-1}=1}^{\infty} \sum_{K_{i+1}=1}^{\infty} \cdots \sum_{K_M=1}^{\infty} G \rho_i^{K_i} \rho_j^{K_j} = (1-\rho_i)\rho_i^{K_i} \tag{4.8}$$

Thus the system is completely solved. Moreover, from J. W. Wong's work for the Laplace transform of the density for the delay of class r messages with fixed routing (under some restrictions) [WONG 78]

$$T_r^*(s) = \prod_{i \in I(r)} \frac{\mu C_i (1-\rho_i)}{s + \mu C_i (1-\rho_i)} \tag{4.9}^*$$

where $I(r)$ = {queue i | queue i is in the path of logical channel r}.

Thus, the average delay is

$$T_r = \sum_{i \in I(r)} \frac{1}{\mu C_i (1 - \rho_i)} \tag{4.10}$$

and the variance of delay is

$$\sigma_{T_r}^2 = \sum_{i \in I(r)} \frac{1}{[\mu C_i (1 - \rho_i)]^2} \tag{4.11}$$

4.1.4 Result of a Simple Example

We apply the above scheme to the network shown in Figures 4.2 and 4.3. The calculated throughput and the simulated throughput are both plotted in Figure 4.4, where we have (see Sec. 4.1.1)

$g = 30$ (total generation rate of permits)

$r_i = 1/10$

$r_{ii} = \frac{1}{n_i}$ = [1 + number of queues adjacent to queue i]

and $\mu_{i0} = 5, \forall i$ (the basic circulation rate).

With these parameter values, we have (see Eq. (4.2))

$$\sum_{i=1}^{10} \lambda_{p_i} = 96$$

We observe in Figure 4.4 that the asymptotic value of total throughput is also 96. This is not an accident and we will explain why in the next section. Also note that we did not obtain results in the low input region since in that region permits enter a queue faster than they are removed (with the basic circulation rate $\mu_{i0} = 5$ fixed), i.e., we obtained an unstable system for the permits. In fact, if $\mu_{i0} \leqslant \lambda_{p_i}$ for any i, this will occur. This is one of the problems which motivated us to investigate the second scheme in Section 4.2.

4.1.5 Some Observations of the Model

(1) *Upper limit of the throughput:* Let γ_r denote the external message input rate of logical channel r. Let $\gamma_i = \sum_{r \in O_i} \gamma_r$ where O_i is the set of all logical channels whose messages are routed through physical channel i as the first hop. Then the total throughput of those logical channels is

$$S_i = [1 - P_{p_i}(0)]\gamma_i \tag{4.12}$$

*This equation holds only for looping free and non-merging after splitting routing schemes.

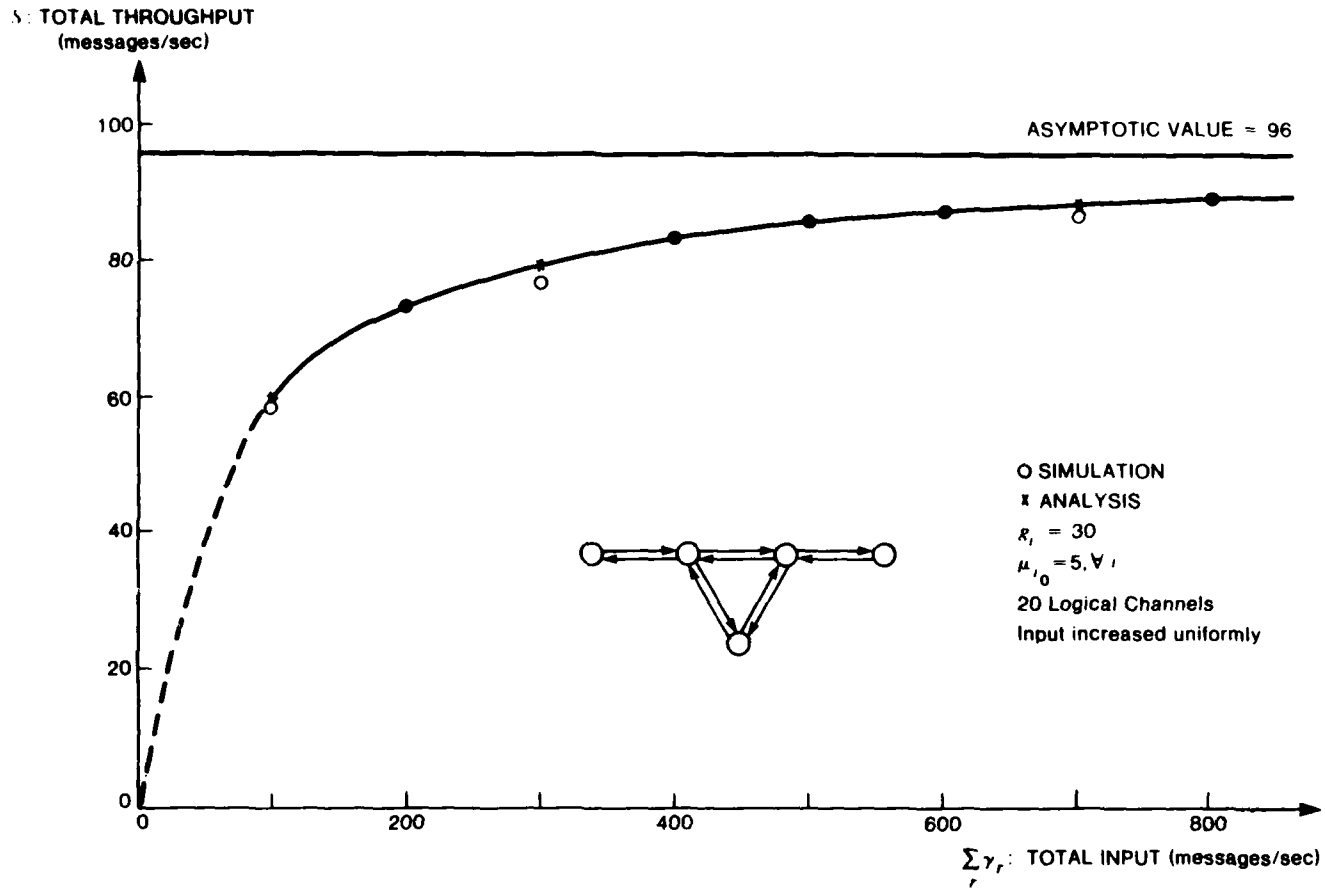


Figure 4.4. Total throughput vs. total input.

$$= \frac{\lambda_{p_j}}{\mu_{j_0}+\gamma_j}\,\gamma_j \tag{4.13}$$

Here we have made use of Eq. (4.4), and λ_{p_j} is a constant determined by Eq. (4.2). Note that as $\gamma_j \to \infty$, $S_j \to \lambda_{p_j}$. This shows that our scheme does put a ceiling ($\sum_i \lambda_{p_i}$) on the acceptance rate of the external input to protect the network from becoming overloaded. This explains why the asymptotic value of the total throughput shown in Figure 4.4 is 96 which is $\sum_{i=1}^{10} \lambda_{p_i}$.

(2) *Inefficiency of the scheme:* In order to prevent permits infinitely accumulating in a queue when external traffic is near zero, we must let

$$\mu_{j_0} > \lambda_{p_i}, \quad \forall j$$

Since

$$S_j = \frac{\lambda_{p_j}}{\mu_{j_0}+\gamma_j}\,\gamma_j$$

we have

$$S_j \to \frac{\lambda_{p_j}}{\mu_{j_0}}\,\gamma_j \text{ as } \gamma_j \to 0$$

which approaches zero linearly with γ_j. And when $\gamma_j = k\mu_{j_0}$, $k = 1, 2, \cdots$

$$S_j\Big|_{\gamma_j = k\mu_{j_0}} = \frac{\lambda_{p_j}}{(k+1)\mu_{j_0}}\,k\mu_{j_0} = \frac{k}{k+1}\,\lambda_{p_j}$$

Figure 4.5 shows this family of performance curves for this control scheme.

We see that the best we can do (without the danger of having an infinite number of permits at a queue so that the arriving messages thereafter are not flow controlled) is to let $\mu_{j_0} > \lambda_{p_i}$. Therefore the best performance we can obtain is the uppermost curve under the ideal one (with throughput $\simeq \lambda_{p_j}/2$ when input $= \lambda_{p_j}$, $\forall j$). This is apparently not satisfactory because it is too conservative. In the next section we will attack this weak point and obtain a nearly perfect static control scheme.

(3) *The bottleneck of a network:* If we define the bottleneck of a network as the physical channel whose associated queue has the largest average backlog, then we are able to identify the bottleneck with respect to a given set of network parameters. Let N_i denote the backlog at message queue i. Then

$$\max_i E[N_i] = \max_i \sum_{K_i=1}^{\infty} K_i P_i(K_i) \qquad \text{(see Eq. (4.8) for } P_i(K_i)\text{)}$$

$$= \max_i (1-\rho_i) \sum_{K_i=1}^{\infty} K_i \rho_i^{K_i}$$

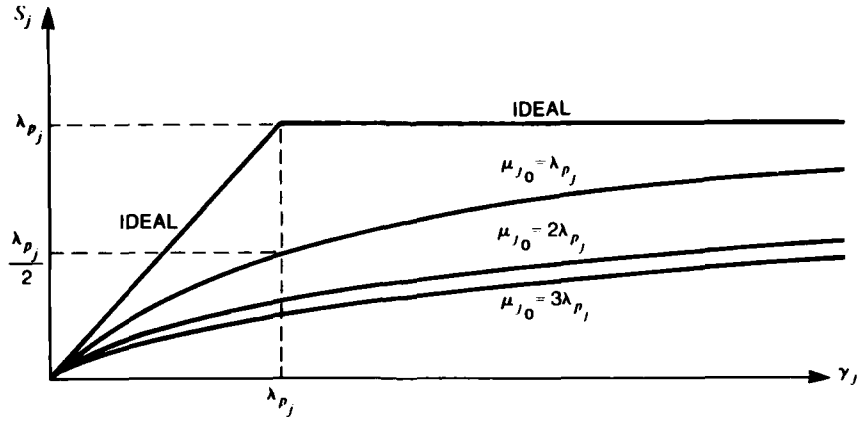


Figure 4.5. Performance of the currency circulation model.

$$= \max_i \frac{1}{1-\rho_i}$$

but,

$$\rho_i = \sum_{j \in R(i)} \frac{S_j}{\mu C}$$

where $R(i) = \{\text{logical channels using physical channel } i\}$. Therefore,

$$\max_i E[N_i] = \max_i \frac{1}{1 - \sum_{j \in R(i)} S_j/\mu C}$$

where S_j is given in Eq. (4.4).

4.2 A More Interesting Model - Scheme 2

The control scheme described in Section 4.1 has some weak points. First, the performance curve is far below the ideal one (see Figure 4.5). Second, we do not know how to optimally specify the transition matrix for permits. In this section we look into a more interesting scheme which produces much more desirable results.

4.2.1 The Model Description

This scheme works similar to the one in the last section except that:

(1) After launching a message into the network, a permit is *destroyed immediately* instead of being circulated to other permit queues.

(2) Permits for logical channel r are generated (from a Poisson process) at rate g_r and are queued in permit queue q_r. Note that there is a permit queue for each logical channel.

(3) We have a finite buffer size N_r for each permit queue q_r. When the buffer is full, the newly generated permits are destroyed thus controlling the unlimited growth of permit queues. The permit queue is shown in Figure 4.6a.

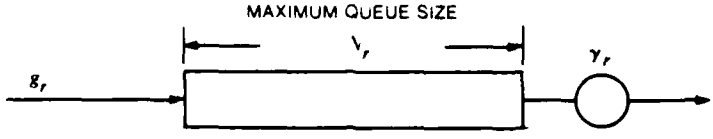


Figure 4.6(a) Permit queue r for logical channel r.

4.2.2 Analysis of the Scheme

The model for permits: Each permit queue q_r (for logical channel r) can be described by the finite Markov chain in Figure 4.6b.

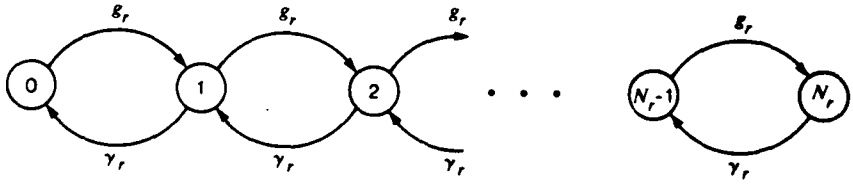


Figure 4.6(b) The Markov chain describing the permit queue.

Let $P_{k_r} = P[k \text{ permits in } q_r]$. We then have

$$P_{k_r} = \left(\frac{g_r}{\gamma_r}\right)^k \cdot P_{o_r} \quad \text{for } k = 0, 1, 2, \cdots, N_r \tag{4.14}$$

$$= P_{o_r}\rho_r^k$$

where $\rho_r \triangleq g_r/\gamma_r$, and

$$P_{0_r} = \frac{1 - (g_r/\gamma_r)}{1 - (g_r/\gamma_r)^{N_r+1}} = \frac{(\gamma_r - g_r)}{(\gamma_r^{N_r+1} - g_r^{N_r+1})}\,\gamma_r^{N_r} \tag{4.15}$$

Note that for $g_r > 0$, as $\gamma_r \to 0$ (i.e., the message input rate is near zero)

$$P_{0_r} \to 0 \tag{4.16}$$

That is, q_r is always full.

As $\gamma_r \to g_r$ we have

$$P_{0_r} \to \frac{1}{N_r + 1} \tag{4.17}$$

As $\gamma_r \to kg_r$ for $k = 2, 3, \cdots$ we have

$$P_{0_r} \to \frac{1 + 1/k}{1 + (1/k)^{N_r+1}} \tag{4.18}$$

and, as $k \to \infty$ (i.e., the message input rate is very large) $P_{0_r} \to 1$.

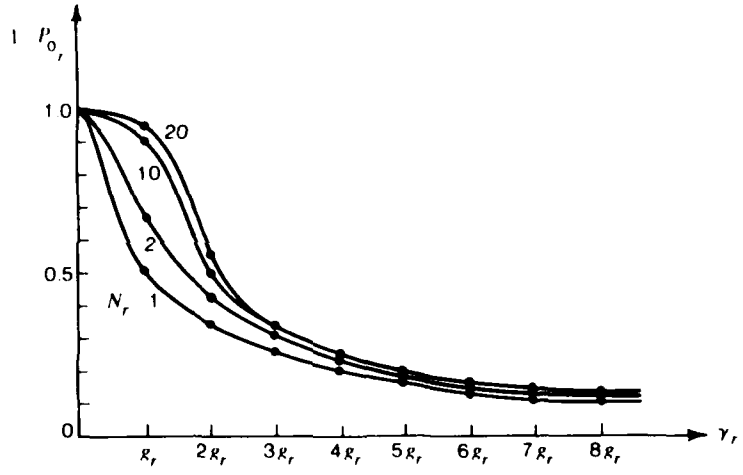


Figure 4.7 The non-empty probability of a permit queue.

In Figure 4.7, we plot $1 - P_{0_r}$ (the non-empty probability of permits for logical channel r) versus γ_r. Observe that $1 - P_{0_r}$ at $\gamma_r = g_r$ is larger than 0.9 at $N_r = 10$. This means that less than one out of ten messages is rejected for lack of permit. The total throughput for logical channel r is

$$S_r = \left(1 - P_{0_r}\right)\gamma_r = \frac{\rho_r - \rho_r^{N_r+1}}{1 - \rho_r^{N_r+1}}\,\gamma_r \tag{4.19}$$

For large N_r, as $\gamma_r \to \infty$ (i.e., heavy applied load for logical channel r) we have $\rho_r \to 0$, and

$$S_r \to \rho_r \gamma_r = \frac{g_r}{\gamma_r} \gamma_r = g_r \tag{4.20}$$

as $\gamma_r \to g_r$, we have $\rho_r \to 1$, and

$$S_r \to \frac{N_r}{N_r + 1} g_r \tag{4.21}$$

and as $\gamma_r \to 0$ (i.e., almost zero applied load for logical channel r) we have $\rho_r \to \infty$, and

$$S_r \to \gamma_r \tag{4.22}$$

4.2.3 Result of the Simple Example

In Figure 4.8, we show the result of applying the scheme to our example network (Figure 4.2) with $g_r = 5$. The plot is for total throughput versus input rate per logical channel.

We see that we can bring our performance curve as close to the ideal curve as we desire by letting N_r be sufficiently large. This is an extremely desirable property. Also, the "danger" mentioned in Sec. 4.1.5 is avoided here. That is, an idle logical channel can no longer make itself uncontrolled later (because of the large number of permits accumulated during the idle period). The computer communication network can now be viewed as a network of queues with external input S_r (which is assumed to be a Poisson process) and with some routing scheme (fixed for our case). The delay can easily be obtained through the application of J. Wong's delay equations. The network model is thus completely solved. Observe again that S_r is not a Poisson process, but it becomes Poisson asymptotically when $N_r \to \infty$. We assume S_r is Poisson in the following two sections.

4.3 The Piggy-back Acknowledgement Scheme

The "ideal" flow control scheme in the previous section operates under the simplified assumption that the transmission channels are noiseless and thus neither time-out nor acknowledgement schemes are necessary. We now relax this assumption to obtain a more realistic model. As the assumption is removed, a destination will issue an ACK for each message it receives and a source will retransmit a message when the ACK for it does not return in the allotted time. The ACKs are usually piggy-backed by messages.

We first look into the piggy-back acknowledgment schemes. In order to maximize the throughput, a computer communication network usually arranges the acknowledgements for messages in one flow to be piggy-backed by the messages in the opposite flow. Yet the traffic in the two mutually opposite flows is usually not symmetric, so that we usually allow a message in one flow to simultaneously piggy-back ACKs for many messages in the opposite flow. To prevent an indefinite delay of ACKs (which may happen when the opposite flow is idle), we usually set upper limits both on the number of ACKs accumulated (waiting to be piggybacked) and on the time that the first ACK has spent waiting for a message to pick it up. If these limits are reached before a message comes up, a special control packet is generated to pick all the waiting ACKs. We first investigate the case where we set a limit only on the number of ACKs waiting for piggyback. Then we look into a more complicated case where limits on both the number of waiting ACKs and the time that ACKs have been waiting are imposed.

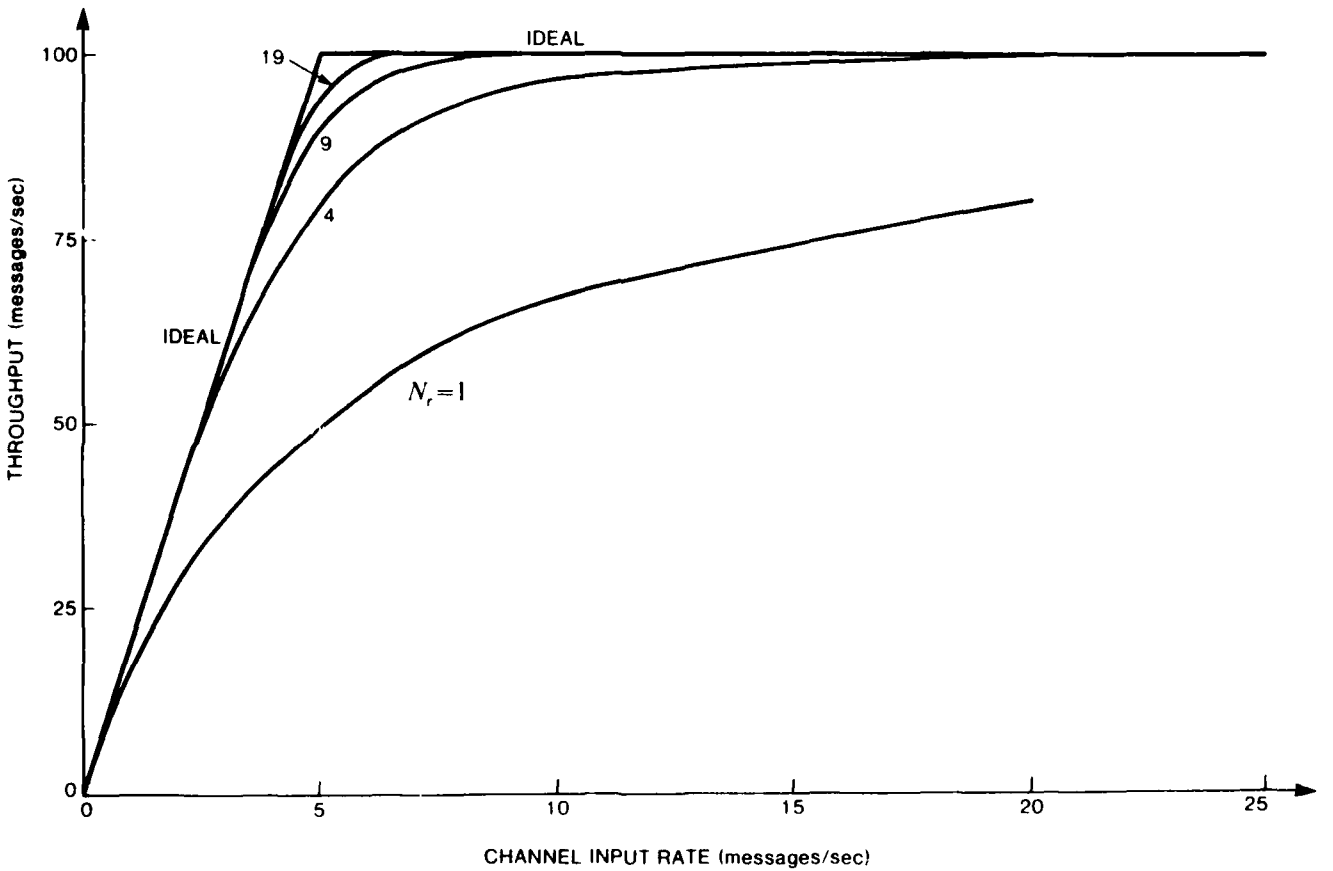


Figure 4.8. The performance curves of Scheme 2.

4.3.1 Piggy-back Scheme with Limit on Number of ACKs Waiting to be Piggy-backed

Consider a computer communication network as shown in Fig. 4.9. The logical channels $S-D$ (the forward channel) and $D-S$ (the backward channel) have external message input rates γ_f, and γ_b. Only a portion of them, S_f and S_b, can be accepted by the network (because of the limited permit generation rate).

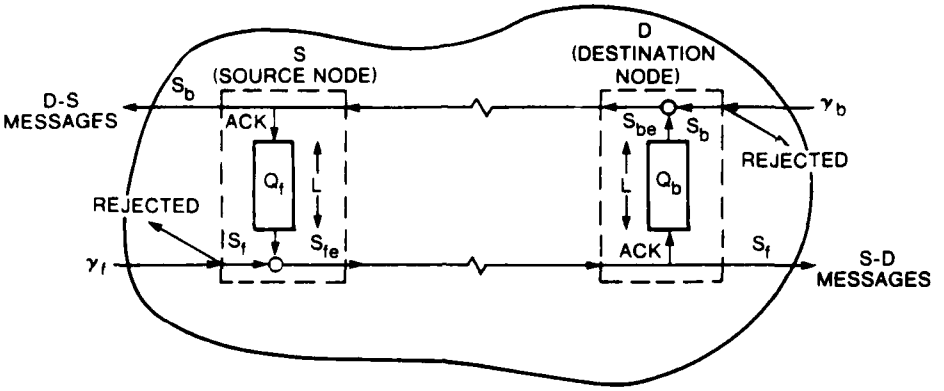


Figure 4.9 Model of piggy-back scheme with limit on number of waiting ACKs.

The accepted messages, say the S-D messages, are routed through the network to the destination D. The ACKs for these messages then enter a queue, Q_b, where they are picked up by the $D-S$ messages. If L ACKs are accumulated in Q_b before a $D-S$ message comes up, then node D generates a special control packet to carry these L ACKs. These control messages, together with S_b, make the effective backward traffic S_{be}. Among S_{be}, S_b need to be acknowledged by node S. By assuming S_f, and S_b are Poisson processes Q_b can be described by the following Markov chain. (See Figure 4.10.) Let $P_k \triangleq P[k \text{ ACKS in } Q_b]$. Then

$$P_k = \left(\frac{S_f}{S_f + S_b} \right) P_{k-1}, \quad 1 \leqslant k \leqslant L-1 \tag{4.23}$$

Let

$$\rho_b \triangleq \frac{S_f}{S_f + S_b} \tag{4.24}$$

then

$$P_k = \rho_b^k P_0 \tag{4.25}$$

$$P_0 = \frac{1}{1 + \rho_b + \cdots + \rho_b^{L-1}} = \frac{1-\rho_b}{1-\rho_b^L} \tag{4.26}$$

Thus,

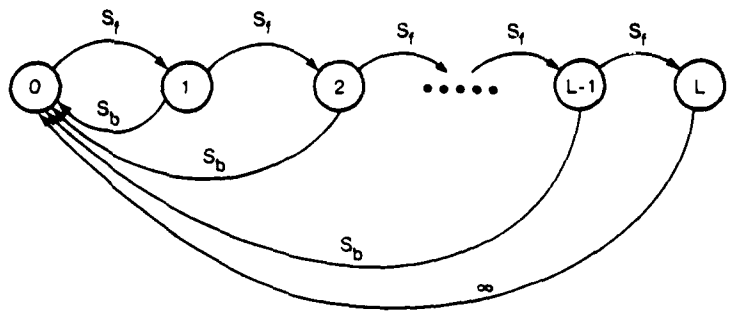


Figure 4.10 The state diagram for Q_b in Figure 4.9.

$$P_k = \left\{ \frac{1-\rho_b}{1-\rho_b^L} \right\} \rho_b^k, \quad 0 \leqslant k \leqslant L-1 \tag{4.27}$$

Note that k can not be equal to L because a special control packet is generated immediately once there are L ACKs. Let us now calculate T_b, the average time an ACK waits in Q_b. As shown in Fig. 4.11, a random tagged customer (an ACK for an S_f message) arrives at Q_b at time τ and sees that K ACKs are already there. He must wait at Q_b until either the L^{th} ACK arrives or until a $D-S$ message is accepted by the network (which picks up all the ACKs).

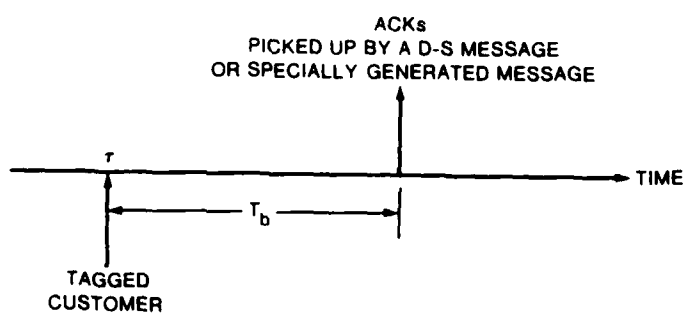


Figure 4.11. The time period for a tagged ACK to wait in Q_b.

Let $P[T_b = t|K] \triangleq$ Pr[tagged customer waits in Q_b for t sec > 0 given he found K ACKs in Q_b for $0 \leqslant K \leqslant L-2$]. Then we have,

$$P[T_b = t|K]dt = \frac{(S_f t)^{L-K-2} e^{-S_f t}}{(L-K-2)!} S_f\, dt\, e^{-S_b t} + S_b\, dt\, e^{-S_b t} \sum_{j=0}^{L-K-2} \frac{(S_f t)^j e^{-S_f t}}{j!}$$

$$= \left[\frac{\alpha(\alpha t)^{L-K-2} e^{-\alpha t}}{(L-K-2)!} \rho_b^{L-K-1} + \sum_{j=0}^{L-K-2} (1-\rho_b)\rho_b^j \frac{\alpha(\alpha t)^j e^{-\alpha t}}{j!} \right] dt \qquad (4.28)$$

where $\alpha \triangleq S_f + S_b$.

The first term of the above equation means that the Lth ACK arrives at $(t, t+dt)$ and before any D-S message arrives; the second terms means that a $D-S$ message picks up all the ACKs at $(t, t+dt)$ and before the Lth ACK arrives.

The conditional expected value of T_b is

$$E[T_b|K] = \int_0^\infty tP[T_b = t|K]\, dt$$

$$= \rho_b^{L-K-1} \frac{L-K-1}{\alpha} + (1-\rho_b) \sum_{j=0}^{L-K-2} \rho_b^j \frac{j+1}{\alpha}$$

$$= \frac{1}{\alpha} [1 + 2\rho_b + 3\rho_b^2 + \cdots + (L-K-1)\rho_b^{L-K-2}]$$

$$- \frac{1}{\alpha} [\rho_b + 2\rho_b^2 + \cdots + (L-K-1)\rho_b^{L-K-1} - (L-K-1)\rho_b^{L-K-1}]$$

$$= \frac{1}{\alpha} [1 + \rho_b + \rho_b^2 + \cdots + \rho_b^{L-K-2}]$$

$$= \frac{1}{\alpha} \frac{1-\rho_b^{L-K-1}}{1-\rho_b} \qquad (4.29)$$

We have used the mathematics developed in Appendix A in deriving Eq. (4.29). The unconditional expected value of T_b is

$$E[T_b] = \sum_{K=0}^{L-2} E[T_b|K]P_K = \sum_{K=0}^{L-2} \frac{1}{\alpha} \frac{1-\rho_b^{L-K-1}}{1-\rho_b} \frac{(1-\rho_b)\rho_b^K}{1-\rho_b^L}$$

$$= \frac{1}{\alpha} \sum_{K=0}^{L-2} \frac{\rho_b^K - \rho_b^{L-1}}{1-\rho_b^L}$$

$$= \frac{1}{\alpha(1-\rho_b^L)} \left\{ [1 + \rho_b + \cdots + \rho_b^{L-2}] - (L-1)\rho_b^{L-1} \right\}$$

$$= \frac{1}{\alpha(1-\rho_b^L)} \left\{ \frac{1-\rho_b^{L-1}}{1-\rho_b} - (L-1)\rho_b^{L-1} \right\} \qquad (4.30)$$

We have used Eq. (4.27) and the fact that $E[T_b|\ L-1] = 0$ in deriving Eq. (4.30).

Note that as $S_b \rightarrow 0$, i.e., the $D-S$ traffic $\rightarrow 0$,

$$\rho_b = \frac{S_f}{S_f + S_b} \rightarrow 1$$

From Eq. (4.30), we have

$$E[T_b] \rightarrow \frac{1}{S_f} \frac{L-1}{2} \tag{4.31}$$

This is intuitively true because when $S_b \rightarrow 0$, the only group output from Q_b is caused by the arrival of the L^{th} $S-D$ message. On the average, a random ACK arriving at Q_b sees $(L-1)/2$ ACKs in front of it and $(L-1)/2$ ACKs arrive after it does. Each inter-arrival is, on the average, $1/S_f$ units of time, i.e.,

$$E[T_b] = \frac{L-1}{2} \frac{1}{S_f}$$

which is Eq. (4.31).

On the other hand, as $S_b \rightarrow \infty$, we have $\rho_b \rightarrow 0$, $\alpha \rightarrow S_b$ and

$$E[T_b] \rightarrow \frac{1}{\alpha} \rightarrow \frac{1}{S_b} \tag{4.32}$$

Again, this is intuitively clear since as $S_b \rightarrow \infty$, no special control message needs to be generated. Each ACK is picked up by a $D-S$ message, which arrives each $1/S_b$ units of time after our ACK joins the queue (the $D-S$ message generation process is assumed to be Poisson).

The effective traffic rate is

$$S_{b'} = S_b + S_f P_{L-1}$$

The first term is of course the D-S message rate, while the second term is the extra traffic generated because the L limit is reached (i.e., the ACK for S-D message finds L-1 ACKs in Q_b)

$$P_{L-1} = (1 - \rho_b)\rho_b^{L-1}/(1-\rho_b^L)$$

Therefore

$$S_{b'} = S_b + S_f \frac{(1-\rho_b)\rho_b^{L-1}}{1-\rho_b^L} \tag{4.33}$$

As $S_b \rightarrow 0$, $\rho_b \rightarrow 1$, and $S_{b'} \rightarrow S_f/L$. That is every L S-D messages contribute an extra D-S message; as $S_b \rightarrow \infty$, $S_{b'} \rightarrow S_b$. This is again true since no special message needs to be generated at all as $S_b \rightarrow \infty$.

Now a computer communication network can be described as if it were a network of queues with external traffic $S_{b'}$'s.

The one-way message delays can again be obtained from L. Kleinrock's equation [KLEI 76]. The round-trip delay (message delay plus ACK delay) of a particular flow, e.g., $S-D$ flow, is

$$\bar{T}_{S-D} + \bar{T}_b + \bar{T}_{D-S}$$

where $\bar{T}_{S-D}$ ($\bar{T}_{D-S}$) is the message delay in the network, and $\bar{T}_b$ is the waiting time in Q_b.

In Fig. 4.12 we show the round trip delay T_r (normalized with respect to the message length $\bar{x}$) of logical channel 5-4 when we apply the piggy-back scheme to our little network in Fig. 4.2. In the figure ρ^* is the utilization factor of the most heavily loaded physical channel when $L=1$ * (e.g., physical channels 5-3, 3-2, and 2-4). We increase traffic uniformly so that ρ^* increases from 0.16 to 1.28. The most we can do is to increase traffic to the extent that $\rho^* \leqslant 2$ since at this point, with $L=\infty$, the effective traffic (in this case no ACKs contribute to the effective traffic) just loads up the most heavily loaded channels (i.e., the channel utilizations are one).

Observe that when ρ^* is small, ACKs for one flow are seldom picked up by messages of the opposite flow (since the traffic is low). Therefore, if L is too large, then so is T_b. As traffic increases, ρ^* increases and we have more intensive traffic to pick up ACKs. It begins to be beneficial to have $L>1$. In our little network's case, we see this occurs around $\rho^* = 0.64$. Thus, as ρ^* increases we want L to increase so as to make the round trip delay small. Of course, as ρ^* increases more the network starts to be saturated (as shown by the curve corresponding to $\rho^* = 1.28$).

4.3.2 Piggy-back Scheme with Limit on Number of Accumulated ACKs and the Length of ACK-waiting Time

Let us again consider the network in Fig. 4.9. We have everything remaining the same except that we add one more condition; namely, that if the first ACK in Q_b has been there for T_a units of time, then a special control message is generated to pick up all the ACKs in Q_b. Remember that this is to prevent the indefinite waiting of ACKs in Q_b.

To describe Q_b, no state diagram such as that in Fig. 4.10 can be drawn because of the T_a limit. Let us refer to the time diagram in Fig. 4.13.

As time goes on, Q_b undergoes repeated cycles, each of which consists of an empty period followed by a filling period. An empty period is ended by the first arriving ACK (for a $S-D$ message) which starts the filling period. The filling period is ended when a $D-S$ message is accepted (by the network), when the T_a time limit expires, or when L ACKs are collected in Q_b.

Refer to Fig. 4.14(a); let $P_{sb} \triangleq$ Pr[a filling period is ended by a $D-S$ message] = Pr[a $D-S$ message is accepted into the network before (L-1) ACKs for $S-D$ messages arrive in node D in the time period $(0, T_a)$ after a filling period starts]. Note that we only need $(L-1)$ ACKs to end a filling period because the period is started by an ACK at time zero. Let A be the event that fewer than $(L-1)$ $S-D$ messages arrive at D in $(0,t)$; and let B be the event that the first $D-S$ message is accepted into the network at $t + dt$. Then

$$P_{sb} = \int_0^{T_a} Pr[A] \cdot Pr[B]\, dt$$

* Remember that when $L=1$, each ACK is sent back to the source as an ordinary message. These messages are also included in calculating ρ^*.

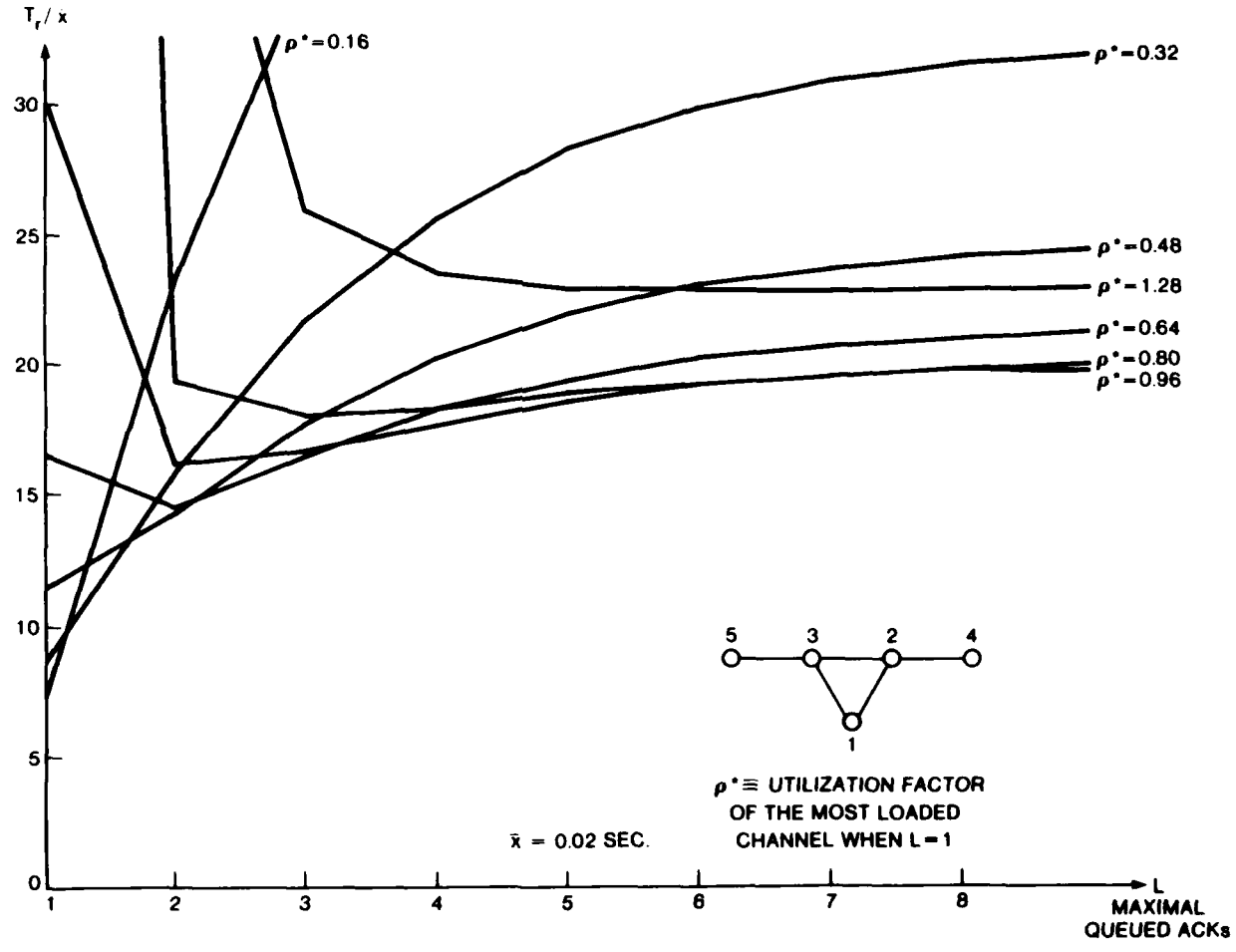


Figure 4.12. Delay of logical channel 5-4 as a function of the maximal allowable ACKs accumulated.

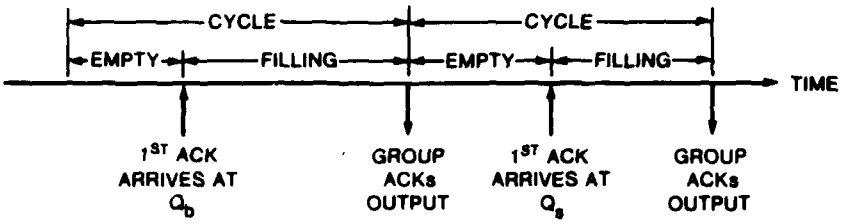


Figure 4.13. Time diagram of Q_b activity.

$$= \int_0^{T_a} \left[\sum_{k=0}^{L-2} \frac{(S_f t)^K}{K!} e^{-S_f t} \right] \cdot e^{-S_b t} S_b \, dt$$

$$= \int_0^{T_a} \sum_{K=0}^{L-2} \frac{S_b S_f^K}{\alpha^{K+1}} \frac{\alpha(\alpha t)^K}{K!} e^{-\alpha t} \, dt \qquad \text{where } \alpha \triangleq S_f + S_b$$

$$= \sum_{K=0}^{L-2} (1-\rho_b)\rho_b^K I_k(0, T_a) \tag{4.34}$$

where

$$\rho_b \triangleq \frac{S_f}{S_f + S_b} \text{ and}$$

$I_K(0, T_a)$ is defined in Appendix A

Refer to Figure 4.14(b). Let $P_L \triangleq$ Pr[a filling period is ended because L ACKs are collected in Q_b]. Then let G be the event that no $D-S$ message is accepted in $(0,t)$; let H be the event that $(L-2)$ ACKs for $S-D$ message arrive at Q_b in $(0,t)$; and let F be that one ACK for an $S-D$ message arrives in $(t, t+dt)$. Then,

$$P_L = \int_0^{T_a} Pr[G] \cdot Pr[H] \cdot Pr[F] \, dt$$

$$= \int_0^{T_a} e^{-S_b t} \cdot \frac{(S_f t)^{L-2} e^{-S_f t}}{(L-2)!} S_f \, dt$$

$$= \rho_b^{L-1} I_{L-2}(0, T_a) \tag{4.35}$$

And finally refer to Fig. 4.14(c); let $P_{T_a} \triangleq$ Pr[a filling period is ended because the first ACK in Q_b has waited T_a units of time]. Also let X be the event that no $D-S$ message is accepted in $(0, T_a)$, and Y be that less than $(L-1)$ ACKs for $S-D$ messages arrive at Q_b in $(0, T_a)$. Then,

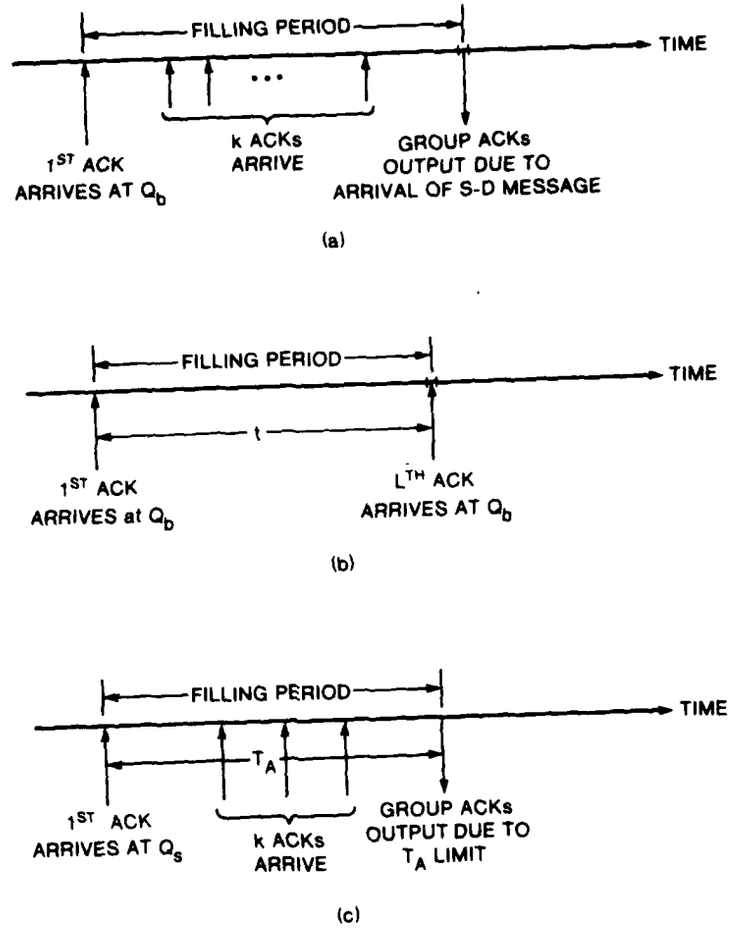


Figure 4.14. Three types of filling periods.

$$P_{T_a} = P[X]\,P[Y] = e^{-S_b T_a} \sum_{K=0}^{L-2} \frac{(S_f T_a)^K}{K!}\, e^{-S_f T_a}$$

$$= \sum_{K=0}^{L-2} \frac{S_f^K}{\alpha^K} \frac{(\alpha T_a)^K}{K!}\, e^{-\alpha T_a}$$

$$= \sum_{K=0}^{L-2} \rho_b^K \frac{(\alpha T_a)^K}{K!}\, e^{-\alpha T_a} \tag{4.36}$$

Observe that as $T_a \to \infty$,

$$P_{S_b} \to 1-\rho_b^{L-1} \tag{4.37a}$$

$$P_L \to \rho_b^{L-1} \tag{4.37b}$$

$$P_{T_a} \to 0; \tag{4.37c}$$

We have (as it should be)

$$P_{S_b} + P_L + P_{T_a} = 1$$

as $T_a \to 0$,

$$P_{S_b} \to 0 \tag{4.38a}$$

$$P_L \to 0 \tag{4.38b}$$

$$P_{T_a} \to 1 \tag{4.38c}$$

Again,

$$P_{S_b} + P_L + P_{T_a} = 1$$

In any case, regardless of the value of T_a, the quantity

$$P \triangleq P_{S_b} + P_L + P_{T_a}$$

must equal to 1. Indeed, from (4.34), (4.35), (4.36) we confirm this as follows:

$$P = P_{S_b} + P_L + P_T$$

$$= \left\{ \sum_{K=0}^{L-2} (\rho_b^K - \rho_b^{K+1}) \left[1 - \sum_{j=0}^{K} \frac{(\alpha T_a)^j}{j!}\, e^{-\alpha T_a} \right] \right\}$$

$$+ \rho_b^{L-1} \left\{ 1 - \sum_{K=0}^{L-2} \frac{(\alpha T_a)^K}{K!}\, e^{-\alpha T_a} \right\} + \left\{ \sum_{K=0}^{L-2} \rho_b^K \frac{(\alpha T_a)^K}{K!}\, e^{-\alpha T_a} \right\}$$

$$= \left\{ 1 - e^{-\alpha T_a} + \sum_{K=1}^{L-1} \rho_b^K \left[1 - \sum_{j=0}^{K} \frac{(\alpha T_a)^j}{j!}\, e^{-\alpha T_a} \right] - \rho_b^{L-1} \left[1 - \sum_{j=0}^{L-1} \frac{(\alpha T_a)^j}{j!}\, e^{-\alpha T_a} \right] \right.$$

$$\left. - \left[\sum_{K'=1}^{L-1} \rho_b^{K'} \left[1 - \sum_{j=0}^{K'} \frac{(\alpha T_a)^j}{j!}\, e^{-\alpha T_a} \right] \right] + \sum_{K'=1}^{L-1} \rho_b^{K'} \left[\frac{(\alpha T_a)^{K'}}{K'!}\, e^{-\alpha T_a} \right] \right\}$$

$$+\rho_b^{L-1}\left\{1-\sum_{K=0}^{L-2}\frac{(\alpha T_a)^K}{K!}\,e^{-\alpha T_a}\right\}+\left\{\sum_{K=0}^{L-2}\rho_b^K\,\frac{(\alpha T_a)^K}{K!}\,e^{-\alpha T_a}\right\}$$

$$=1-e^{-\alpha T_a}+\rho_b^{L-1}\,\frac{(\alpha T_a)^{L-1}}{(L-1)!}\,e^{-\alpha T_a}+e^{-\alpha T_a}$$

$$-\rho_b^{L-1}\,\frac{(\alpha T_a)^{L-1}}{(L-1)!}\,e^{-\alpha T_a}$$

$$=1 \tag{4.39}$$

We now calculate the expected value of the filling period T_f.

Let $T_4 \triangleq E[T_f \mid T_a] \triangleq$ expected value of T_f given the filling period is ended because the T_a limit is reached. Of course,

$$T_4 = T_a \tag{4.40}$$

Let $T_{S_b} \triangleq E[T_f|S_b] \triangleq$ expected value of T_f given that the filling period is ended by an $D-S$ message. Then,

$$T_{S_b}=\int_0^{T_a} t\cdot dP[t|S_b]=\int_0^{T_a} t\,\frac{e^{-S_f t}\cdot S_b\,dt\left[\sum_{K=0}^{L-2}\frac{(S_f t)^K}{K!}\,e^{-S_f t}\right]}{P_{S_b}}$$

$$=\frac{\sum_{K=0}^{L-2}\int_0^{T_a}\frac{S_b S_f^K}{\alpha^{K+1}}\,\frac{K+1}{\alpha}\,\frac{\alpha(\alpha t)^{K+1}}{(K+1)!}\,e^{-\alpha t}\,dt}{P_{S_b}}$$

$$=\frac{\sum_{K=0}^{L-2}\frac{K+1}{\alpha}(1-\rho_b)\rho_b^K\,I_{K+1}(0,T_a)}{P_{S_b}} \tag{4.41}$$

And let $T_L \triangleq E[T_f|L] \triangleq$ expected value of T_f given the filling period ended because L ACKs accumulated in Q_b.

$$T_L=\frac{\int_0^{T_a}\frac{(S_f t)^{L-2}e^{-S_f t}}{(L-2)!}\,S_f\,dt\cdot e^{-S_b t}}{P_L}$$

$$=\frac{\frac{L-1}{\alpha}\,\rho_b^{L-1}\,I_{L-1}(0,T_a)}{P_L} \tag{4.42}$$

From (4.40), (4.41), (4.42)

$$T_f = E[T_f|S_h]\cdot P_{S_h} + E[T_f|L]\cdot P_L + E[T_f|T_a]\cdot P_{T_a}$$

$$= \sum_{K=0}^{L-2} \frac{K+1}{\alpha}(1-\rho_b)\rho_b^K I_{K+1}(0,T_a)$$

$$+ \frac{L-1}{\alpha}\rho_b^{L-1} I_{L-1}(0,T_a) + T_a \sum_{K=0}^{L-2} \rho_b^K \frac{(\alpha T_a)^K}{K!} e^{-\alpha T_a} \tag{4.43}$$

Observe that as $T_a \to \infty$, we have

$$T_f \to \sum_{K=0}^{L-2} \frac{K+1}{\alpha}(1-\rho_b)\rho_b^K + \frac{L-1}{\alpha}\rho_b^{L-1}$$

$$= \frac{1-\rho_b}{\alpha}(1 + 2\rho_b + 3\rho_b^2 + \cdots + (L-1)\rho_b^{L-2}) + \frac{L-1}{\alpha}\rho_b^{L-1}$$

$$= \frac{1}{\alpha}(1 + \rho_b + \rho_b^2 + \cdots + \rho_b^{L-2}) = \frac{1}{\alpha}\frac{1-\rho_b^{L-1}}{1-\rho_b} \tag{4.44}$$

And, as $T_a \to 0$

$$T_f \to 0$$

The effective traffic S_{he} (S_h + the special control messages generated when L or T_a limits are reached) is derived as follows: Consider N (a large number) cycles (remember each cycle consists of an empty period followed by a filling period). Among N cycles only $(P_{T_a}+P_L)N$ of them will generate extra messages.

Therefore, for large N,

$$S_{he} = S_h + \frac{N\cdot(P_{T_a}+P_L)}{P_L N(T_L + \frac{1}{S_f}) + P_{S_h}N(T_{S_h} + \frac{1}{S_f}) + P_{T_a}N(T_a + \frac{1}{S_f})} \tag{4.45}$$

where the denominator of the second term is the average time length of the N cycles. Thus,

$$S_{he} = S_h + \frac{P_{T_a}+P_L}{(P_L T_L + P_{S_h}T_{S_h} + P_{T_a}T_a) + \frac{1}{S_f}}$$

$$= S_h + \frac{P_{T_a}+P_L}{T_f + \frac{1}{S_f}} \tag{4.46}$$

We observe that as $T_a \to \infty$, from Equations (4.37), (4.44)

$$S_{he} = S_h + \frac{1}{\left(\frac{1-\rho_b^{L-1}}{\alpha(1-\rho_b)} + \frac{1}{S_f}\right)}\left(\rho_b^{L-1} + 0\right)$$

$$= S_b + \frac{S_f(1-\rho_b)\rho_b^{L-1}}{1-\rho_b^L} \tag{4.47}$$

Equation (4.47) is exactly the same as Equation (4.33) (where we had no limit on time) as it should be. On the other hand, if $T_A \to 0$, then

$$S_{be} \to S_b + (1+0) \,/\, [0+1/\, S_f] = S_b + S_f$$

This is correct since each ACK for a $D-S$ message now generates a special control message.

Having done the above calculations, we are ready to derive the average delay time T_b for an ACK to stay in Q_b. Refer to Fig. 4.14a. Let $P[K,t|S_b] \triangleq$ Pr[the filling period lasts t units of time and ends with $K+1$ ACKs in Q_b | it is ended by a $D-S$ message] = $P[K,t,S_b]/P_{S_b}$, where "1" is the first ACK which starts the filling period, and

$$P[K,t|S_b] = \begin{cases} e^{-S_b t} S_b \, dt \, \dfrac{(S_f t)^K \, e^{-S_f t}}{K!} & \text{for } 0 \leqslant K \leqslant L-2,\ t < T_a \\ 0 \quad \text{for } K \geqslant L-1 \text{ or } t \geqslant T_a \end{cases} \tag{4.48}$$

Let $P[K,t|L] \triangleq$ Pr[the filling period lasts t units of time and ends with $K+1$ ACKs in Q_b | it ends because the $L-$limit is reached] = $P[K,t,L]/P_L$

$$P[K,t|L] = \begin{cases} e^{-S_b t} \, \dfrac{(S_f t)^{L-2}}{(L-2)!} \, e^{-S_f t} \, S_f \, dt\,, & \text{for } K = L-1,\ t < T_a \\ 0, \quad \text{otherwise} \end{cases} \tag{4.49}$$

Let $P[K|T_a] \triangleq$ Pr[the filling period ends with $K+1$ ACKs in Q_b given that it ends because the T_a limit is reached] = $P[K,T_a]/P_{T_a}$

$$P[K|T_a] = \begin{cases} e^{-S_b T_a} \, \dfrac{(S_f T_a)^K}{K!} \, e^{-S_f T_a} / P_{T_a}\,, & \text{for } 0 \leqslant K \leqslant L-2,\ t = T_a \\ 0, \quad \text{for } t < T_a \text{ or } K \geqslant L-1 \end{cases} \tag{4.50}$$

Before proceeding further, we make an observation (which is easy to prove):

If we know that during a time period [0,t], we have k Poisson arrivals, and at t they are all served together, then the average waiting time is $\frac{t}{2}$.

Now, to calculate T_b, the average waiting time in Q_b, consider a large number N of filling periods. Define $N_{sb}(K,t) \triangleq$ the number of filling periods ended by a $D-S$ message at $(t,t+dt)$, with $K+1$ ACKs in Q_b (we call this type of filling period the (K,t,S_b)-type). Then,

$$N_{sb}(K,t) = N \cdot P_{sb} \cdot P[K,t|S_b] \tag{4.51}$$

Let

$$t_{sb}(K,t) = N_{sb}(K,t)[K\,\frac{t}{2} + t] \tag{4.52}$$

Then t_{s_b} is the total time each (K,t,S_b)-type ACK filling periods wait in Q_b. Note that the second term, t, in Eq. (4.52) is the time that the first arrival (who starts the filling period) waits in Q_b.

Similarly, $N_{Ta}(K) \triangleq$ the number of filling periods that end because the T_a limit is reached and K ACKs have collected in Q_b ((K,T_a) type).

$$N_{T_a}(K) = N \cdot P_{T_a} \cdot P[K \mid T_a] \tag{4.53}$$

Let $t_a(K) \triangleq$ total amount of time the ACKs of (K,T_a)-type filling periods wait in Q_b collectively. Then,

$$t_a(K) = N_{Ta}(K) \cdot \left[K \cdot \frac{T_a}{2} + T_a \right] \tag{4.54}$$

Similarly, let $N_L(t) \triangleq$ the number of filling periods that end at t because the L-limit is reached. Then,

$$N_L(t) = N \cdot P_L \cdot P[t \mid L]$$

$$t_L(t) = N_L(t) \cdot [t + 0 + (L-2)\ \frac{t}{2}\ ] \tag{4.55}$$

The first term, t, is the length of time that the first ACK waits; the second term, 0, is the length of time that the L^{th} ACK waits. Thus,

$$T_b = \frac{\sum_{K=0}^{L-2} \int_0^{T_a} t_{s_b}(K,t)\,dt + \sum_{K=0}^{L-2} t_a(K) + \int_0^{T_a} t_L(t)\,dt}{\sum_{K=0}^{L-2} \left[\int_0^{T_a} (K+1) N_{s_b}(K,t)\,dt \right] + \sum_{K=0}^{L-2} \left[N_{T_a}(K) \cdot (K+1) \right] + \int_0^{T_a} N_L(t) L\ dt} \tag{4.56}$$

The denominator is the total number of ACKs gathered in the N filling periods and the numerator is the total amount of time that all the ACKs stay in Q_b. T_b can be rewritten as

$$T_b \triangleq \frac{A_1+A_2+A_3}{B_1+B_2+B_3} \tag{4.57}$$

with each term corresponding to that of the last equation.

We have, after a little calculation,

$$A_1 = N \sum_{K=0}^{L-2} (1-\rho_b)\rho_b^K \frac{(K+1)(K+2)}{2\alpha} \cdot I_{K+1}(0,T_a) \tag{4.58}$$

$$B_1 = N \sum_{K=0}^{L-2} (K+1)(1-\rho_b)\rho_b^K \cdot I_K(0,T_a) \tag{4.59}$$

$$A_2 = N \sum_{K=0}^{L-2} \frac{(K+1)(K+2)}{2\alpha}\ \rho_b^K \frac{(\alpha T_a)^{K+1}}{(K+1)!}\ e^{-\alpha T_a} \tag{4.60}$$

$$B_2 = N \sum_{K=0}^{L-2} (K+1)\rho_b^K \frac{(\alpha T_a)^K}{K!}\ e^{-\alpha T_a} \tag{4.61}$$

$$A_3 = N \frac{L(L-1)}{2\alpha} \rho_b^{L-1} I_{L-1}(0, T_a) \tag{4.62}$$

and

$$B_3 = N L \rho_b^{L-1} I_{L-2}(0, T_a) \tag{4.63}$$

We again use the mathematics in Appendix A to derive the above equations.

Let us now check the limiting value of T_b. When $T_a \to \infty$,

$$T_b \to \frac{\sum_{K=0}^{L-2} (1-\rho_b)\rho_b^K \frac{(K+1)(K+2)}{2\alpha} + \frac{L(L-1)}{2\alpha} \rho_b^{L-1}}{\sum_{K=0}^{L-2} (K+1)(1-\rho_b)\rho_b^K + L\rho_b^{L-1}}$$

$$= \frac{\frac{1}{2\alpha}\left[2 + \sum_{K=1}^{L-2} \rho_b^K [(K+1)(K+2) - K(K+1)]\right]}{1 + \rho_b + \rho_b^2 + \cdots + \rho_b^{L-2} - (L-1)\rho_b^{L-1} + L\rho_b^{L-1}}$$

$$= \frac{(1 + 2\rho_b + 3\rho_b^2 + \cdots + (L-1)\rho_b^{L-2})}{\alpha(1-\rho_b^L)/(1-\rho_b)}$$

$$= \frac{(1 + \rho_b + \rho_b^2 + \cdots + \rho_b^{L-2} - (L-1)\rho_b^{L-1})/(1-\rho_b)}{\alpha(1-\rho_b^L)/(1-\rho_b)}$$

$$= \frac{\frac{1-\rho_b^{L-1}}{1-\rho_b} - (L-1)\rho_b^{L-1}}{\alpha(1-\rho_b^L)} \tag{4.64}$$

This is exactly (4.30) where we have no waiting time limit!

From Appendix A, when $T_a \to 0$,

$$I_K(0, T_a) = 1 - \sum_{i=0}^{K} \frac{(\alpha T_a)^i}{i!} e^{-\alpha T_a} \to 1 - \left[1 - \alpha T_a + \frac{(\alpha T_a)^2}{2} + \cdots\right]\left[1 + \alpha T + \cdots\right]$$

$$= 1 - \left[1 - \frac{\alpha T_a)^2}{2} + \cdots\right] \to \frac{(\alpha T_a{}^2)}{2} \tag{4.65}$$

Therefore,

$$T_b \rightarrow$$

$$\frac{\sum_{k=0}^{L-2}(1-\rho_b)\rho_b^k(K+1)(K+2)(\alpha T_a)^2 \cdot 2\alpha + \frac{1}{\alpha}(\alpha T_a)\left[1-\alpha T_a+\frac{(\alpha T_a)^2}{2}+\cdots\right]+\frac{L(L-1)}{2}\rho_b^{L-1}\frac{(\alpha T_a)^2}{2}}{\sum_{k=0}^{L-2}(K+1)(1-\rho_b)\rho_b^k\frac{(\alpha T_a)^2}{2}+\left[1-\alpha T_a+\frac{(\alpha T_a)^2}{2}+\cdots\right]+L\rho_b^L\frac{(\alpha T_a)^2}{2}}$$

$$\rightarrow \frac{\frac{1}{\alpha}(\alpha T_a-(\alpha T_a)^2+\cdots)}{1-\alpha T_a+\frac{(\alpha T_a)^2}{2}} \rightarrow T_a$$

$\rightarrow 0$ linearly with T_a as expected.

Figures 4.15 and 4.16 show the results (T_b and round trip delay of logical channel 5-4) of applying the scheme to our little network (Figure 4.2).

Note that when the externally applied load is small (e.g., ρ^*=0.16) we want to keep L and T_a small. The reason again is that when traffic is low, it does not pay to wait at Q_b. On the other hand, when the applied load is high (e.g., ρ^*=0.96), there exists an optimal (L, T_a) value to minimize T_r . In Figure 4.17, the minimal round trip delay T_r is plotted (obtained by adjusting L and T_a) versus ρ^* . Observe that as ρ^* increases from zero, T_r increases linearly with ρ^* at first; then the piggy-back scheme begins to become effective in reducing T_r as ρ^* becomes medium large (0.75 to 1.25); finally the network is saturated as ρ^* becomes too large ($>$ 1.25). The above derivation provides a method (static policy) for a network to dynamically adjust the number of ACKs to be piggy-backed together and/or the time length of delayed piggy-backs.

4.4 The Time Out Scheme

In the real world, transmission channels are always noisy and hardware and software at the nodes are error prone. In a computer communication network, data may become lost (e.g., because of uncorrectable errors in the header) along the path from its source to its destination. Therefore a time-out scheme is usually needed to automatically retransmit those messages whose ACKs do not come back to the source within a specified period of time and are thus presumed lost.

In this section, we try to incorporate the time-out scheme into our flow control model.

The following assumptions are made

(1) Newly arrived messages are accepted or rejected as before, i.e., accepted only if there are permits.

(2) The probability that a message gets lost at hop i is P_i .

(3) When the ACK of a message does not come back to the source of the message within the time-out limit T_0 since the instant of the last transmission of that message, the source will immediately retransmit it

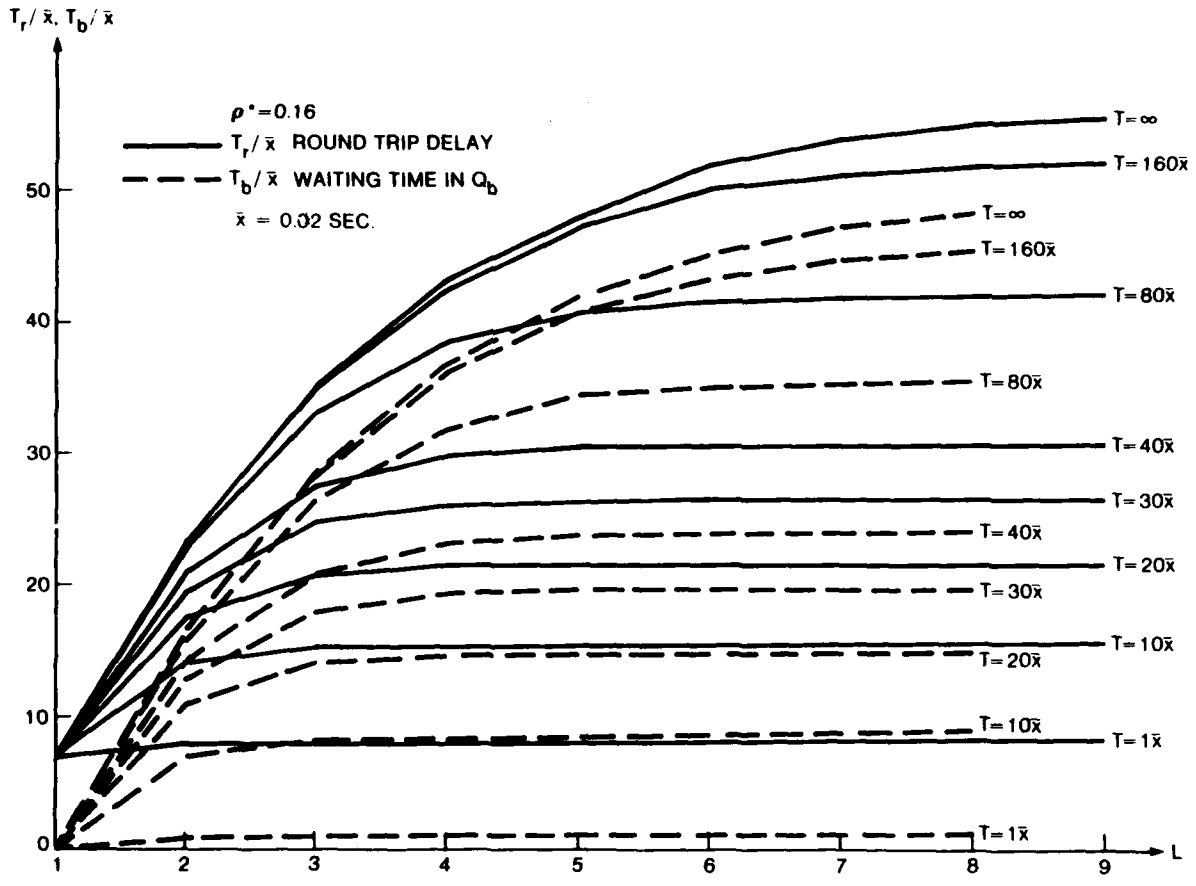


Figure 4.15. Delay of logical channel 5-4 as function of L and T at $\rho^*=0.16$.

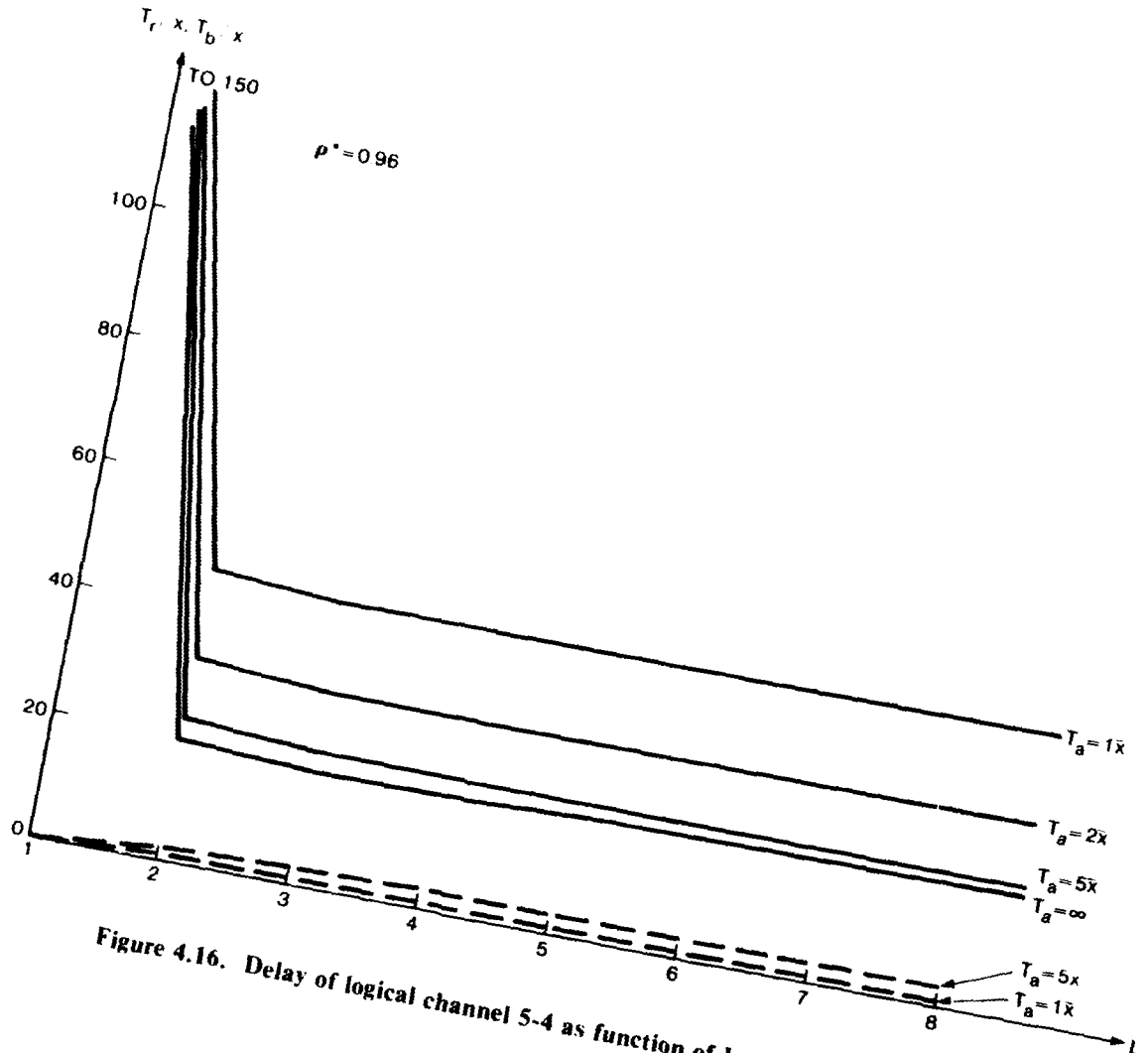


Figure 4.16. Delay of logical channel 5-4 as function of L and T at $\rho^* = 0.96$.

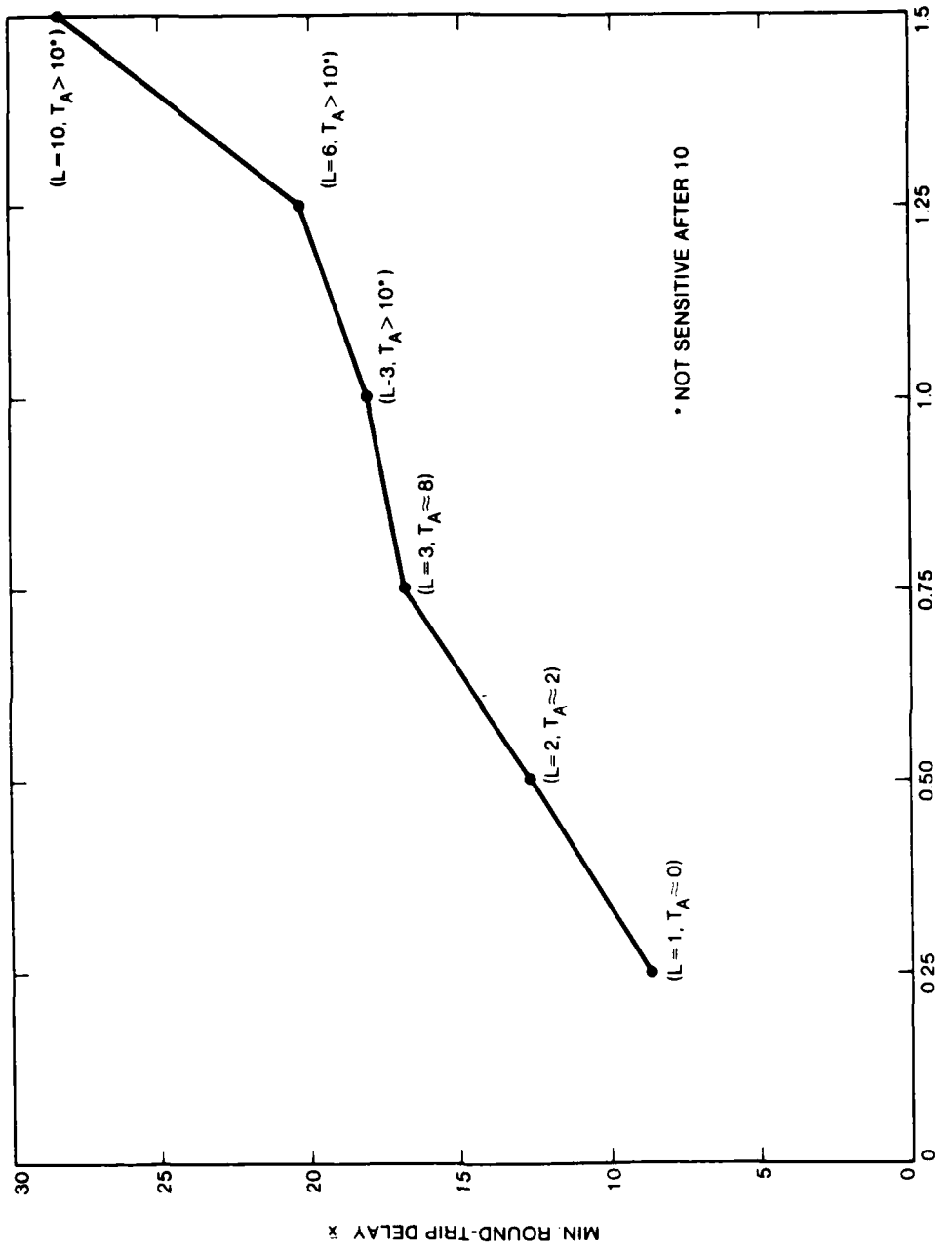


Figure 4.17. The minimal delay of logical channel 5-4 by adjusting L and T.

(4) The source will stop retransmitting a message once it receives an ACK for that message whether the ACK is or is not for the most recent transmission of that message. The subsequent ACKs (if any) for the same message are just ignored.

(5) An ACK is treated here as an ordinary message except that it does not consume a permit.

(6) Each retransmission of a message uses up one permit. If there are no permits the retransmission will use a "credit card" i.e., it will consume a permit beforehand; the next generated permit then makes up this deficit. Thus, a node will destroy the next n generated permits if there are n permits consumed in advance by the retransmitted messages.

(7) The destination acknowledges the correct messages only. And it accepts only the first correct copy of a message.

(8) For simplicity we make the approximation that the round trip delay of a message (message delay + ACK delay) never exceeds $3\ T_0$ if the message does not get lost. This is a reasonable assumption since T_0 is usually set to be greater than the average round trip delay of messages. Notice that for an exponential distribution with average σ, the distribution function $F[3\sigma]=\int_0^{3\sigma} \frac{1}{\sigma} e^{(-x/\sigma)}\, dx = 1-e^{-3} \simeq 0.95$. Therefore, Pr[round trip delay $\geqslant 3\sigma$] $\simeq 0.05$. And T_0 is usually $>>\sigma$.

(9) For easy mathematical manipulation, we assume that for each logical channel, the external input traffic plus the retransmitted traffic is always larger than the permit generation rate g_r (this implies that the total traffic of logical channel r that enters the network can be approximated by g_r, i.e., the heavy traffic situation).

Under the above assumptions we now analyze our time-out scheme. We first calculate the average number of transmissions N_r of a message. Then the throughput of a logical channel r is

$$S_r = \frac{g_r}{N_r} \tag{4.66}$$

Next, we derive the average delay D_r of a message, as the source of the message sees it (i.e., the time period between the instant of the first transmission and the instant of receival of the first ACK of a message). Let $P_i \triangleq$ Pr[the round trip time of a particular transmission lies in the interval $[(i-1)T_0, iT_0]$ given the transmission is not lost], i = 1, 2, 3.

Note that by assumption (8), we have

$$P_1 + P_2 + P_3 = 1$$

Define $P_{re} \triangleq$ Pr[a transmission results in retransmission] and $P_l \triangleq$ Pr[a transmission gets lost]. We have

$$P_{re} = 1 - \prod_{i \in I(r)} (1 - P_{l_i}) \tag{4.67}$$

where $I(r) = \{\text{hop } i | \text{hop } i \text{ is in the round trip path of logical channel } r\}$

and

$$P_{rc} = P_l + \bar{P}_l(1-P_1) \tag{4.68}$$

where $\bar{X} \triangleq 1-X$

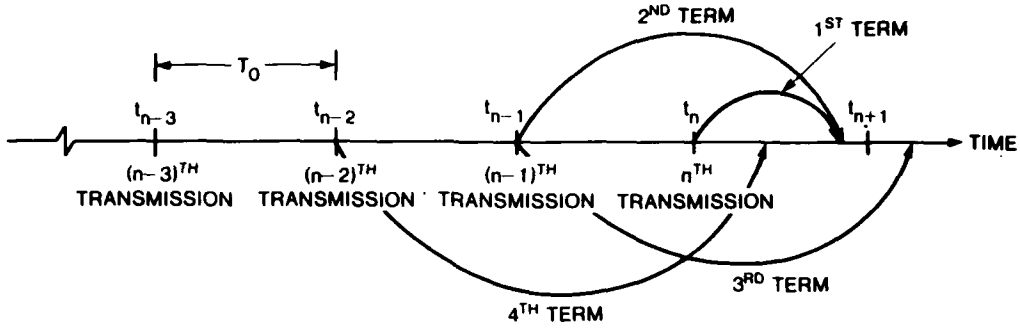


Figure 4.18. The event time diagram for $P[N_t=n]$.

Now refer to Figure 4.18. We have

$$P[N_t=n] = P_l^{n-1}\cdot\bar{P}_{rc} + P_l^{n-2}\cdot\bar{P}_l\cdot P_2 + P_l^{n-2}\cdot\bar{P}_l\cdot P_3\cdot\bar{P}_{rc} + P_l^{n-2}\cdot\bar{P}_l\cdot P_3\cdot P_{rc} \tag{4.69}$$

The first term means that the only 'not-lost' transmission is the n^{th} one, and whose ACK comes back within the time-out limit T_0, i.e., within (t_n, t_{n+1}). The second term says that the first transmission not lost is the $(n-1)^{th}$ one, whose ACK comes back within 2 T_0, i.e., in (t_n, t_{n+1}) . Thus the n^{th} transmission must be made (but we do not care about the result of this transmission). The third term represents the case where the first transmission not lost is the $(n-1)^{th}$ one, and its ACK comes back within 3 T_0 (i.e., in (t_{n+1}, t_{n+2})), and the ACK of the n^{th} transmission comes back within T_0. The last term tells us that the first transmission not lost is the $(n-2)$ one, but its ACK comes back within 3 T_0, and the $(n-1)^{th}$ transmission results in retransmission. We have no more terms because of assumption (8). Since we have fixed routing and the messages are first-come-first-served, if the ACKs are not lost they are never returned out of sequence; therefore, the third term is impossible, and is dropped. Thus, Equation (4.69) can be rewritten for later use as

$$\begin{aligned} P[N_t=n] &= P_l^{n-1}\cdot\bar{P}_{rc} + P_l^{n-2}\cdot\bar{P}_l\cdot P_2 + P_l^{n-3}\cdot\bar{P}_l\cdot P_3\cdot P_{rc} \\ &\triangleq P_{n1} + P_{n2} + P_{n3} \end{aligned} \tag{4.70}$$

with P_{ni} corresponding to the i^{th} term. The average value of N_t is

$$E[N_t] = \sum_{n=1}^{\infty} n\ P[N_t=n] = P_2 + 2P_3P_{rc} + \frac{P_2+P_3P_{rc}}{\bar{P}_l} + \frac{\bar{P}_{rc}}{\bar{P}_l^2} \tag{4.71}$$

Let us now try to find the round trip delay D_r as the source sees it, i.e., the time period between the instances of the 1st transmission of the message and the reception of the 1st ACK for that message.

Let $E[D_r|P_{nj}]$ represent the average of D_r given $P_{nj} = 1$, $j = 1$, 2, or 3. Let T_i denote the time period between $(i-1)T_0$ and the time that the ACK for a particular transmission comes back to the source given that it comes back in $((i-1)T_0, iT_0)$ $i = 1, 2, 3$. And finally let T_t denote the round trip delay of a particular transmission i.e., the time period between the instant a transmission finishes and the instant the ACK for that particular transmission returns. Observe the difference between T_i and D_r . T_t can be ∞ if that particular transmission gets lost.

Referring to the time diagram in Figure 4.18, we have

$$E[D_r|P_{n1}] = (n-1)T_0 + E[T_1]$$

$$E[D_r|P_{n2}] = (n-1)T_0 + E[\min\{T_2, T_t\}]$$

$$E[D_r|P_{n3}] = (n-1)T_0 + E[\min\{T_3, [T_2|P_{re}], T_t\}]$$

The same assumption that we have fixed routing and messages are first-come-first-served (hence messages (and ACKs) never arrive out of sequence) enable us to simplify the above equations to

$$E[D_r|P_{n1}] = (n-1)T_0 + E[T_1] \tag{4.72}$$

$$E[D_r|P_{n2}] = (n-1)T_0 + E[T_2] \tag{4.73}$$

$$E[D_r|P_{n3}] = (n-1)T_0 + E[T_3] \tag{4.74}$$

Thus,

$$E[D_r] = \sum_{n_i=1}^{3} E[D_r|r_{n_i}]\cdot P_{n_i} \tag{4.75}$$

With Eq. (4.66) for throughput, (4.71) for average number of transmissions, and (4.75) for the average message round-trip delay, we have solved the system. The only thing left is to derive $E[T_1]$, $E[T_2]$, and $E[T_3]$.

This can be done as follows: We first define the following quantities. $\lambda_i \triangleq$ the sum of all traffic which is routed through physical channel i. By the heavy traffic assumption

$$\lambda_i = \sum_{r \in R(i)} g_r \tag{4.76}$$

where
$R(i) = \{$logical channel r | message of logical channel r are routed through physical channel $i\}$.

Let $T_{ri} \triangleq$ the round trip delay of a message given $P_{ri} = 0$, and there is no time-out scheme.

We have from J. Wong's equation

$$T_r^*(S) = \prod_{i \in I(r)} \frac{\mu C_i - \lambda_i}{S + (\mu C_i - \lambda_i)} \tag{4.77}$$

for a particular logical channel r. From this, the probability density $f_{T_r}(t)$ can be obtained by inverting $T_r^*(s)$ Thus, for $t = 1, 2, 3$

$$E[T_t] = \int_{(t-1)T_0}^{tT_0} t \, f_{T_r}(t | t \in [(t-1)T_0, tT_0]) \, dt \tag{4.78}$$

We apply the above time-out scheme to our little network again. The results are shown in Figures 4.19 and 4.20.

Fig. 4.19 shows the throughput and delay of logical channel 5-4 in the little network. Observe the sharp decrease of throughput and sharp increase in delay at heavy external load.

The long delay is because of the nearly saturated network capacity while the decrease in throughput is because of the excessive retransmission of messages which is caused by the time-out scheme. Note that the curves corresponding to $T_0 = 0.5\ T_r$ (T_r is the average round-trip delay of logical channel 5-4 when P_b=0 and there is no time out mechanism) are not accurate since assumption (7) no longer holds. The dashed line for the throughput shows the true behavior.

Fig. 4.20 shows the "power" (defined below) versus T_0/T_r for different channel error rates P_b's. The optimal (maximum power) time-out line is also shown as the broken curve. The power for a logical channel is defined as

$$Power = \frac{\left\{\dfrac{\text{throughput with time-out scheme}}{\text{throughput without time-out scheme}}\right\}}{\left\{\dfrac{\text{round trip delay with time-out scheme and error}}{\text{round trip delay without time-out scheme and error}}\right\}} = \frac{(S_r/g_r)}{(D_r/T_r)} \tag{4.79}$$

This is a measure of the effectiveness of the time-out scheme under a set of network parameters. The maximal value of power is 1.

4.5 Buffered Input Model and Inter-netting

Networks are usually connected by *gateways* where differences in protocols are resolved, and where billing and flow control between networks takes place.

In this section we discuss how to apply the idea of flow control based on limiting the permit generation rate to the inter-netting case.

Consider two networks, N_1 and N_2, connected by a gateway G as illustrated in Fig. 4.21(a). Nodes $a, b, c, \cdots$ communicate with nodes $i, j, k, \cdots$ by first sending their messages to the gateway G. After processing these messages the gateway G will send them to their destination i, j, or k, etc.

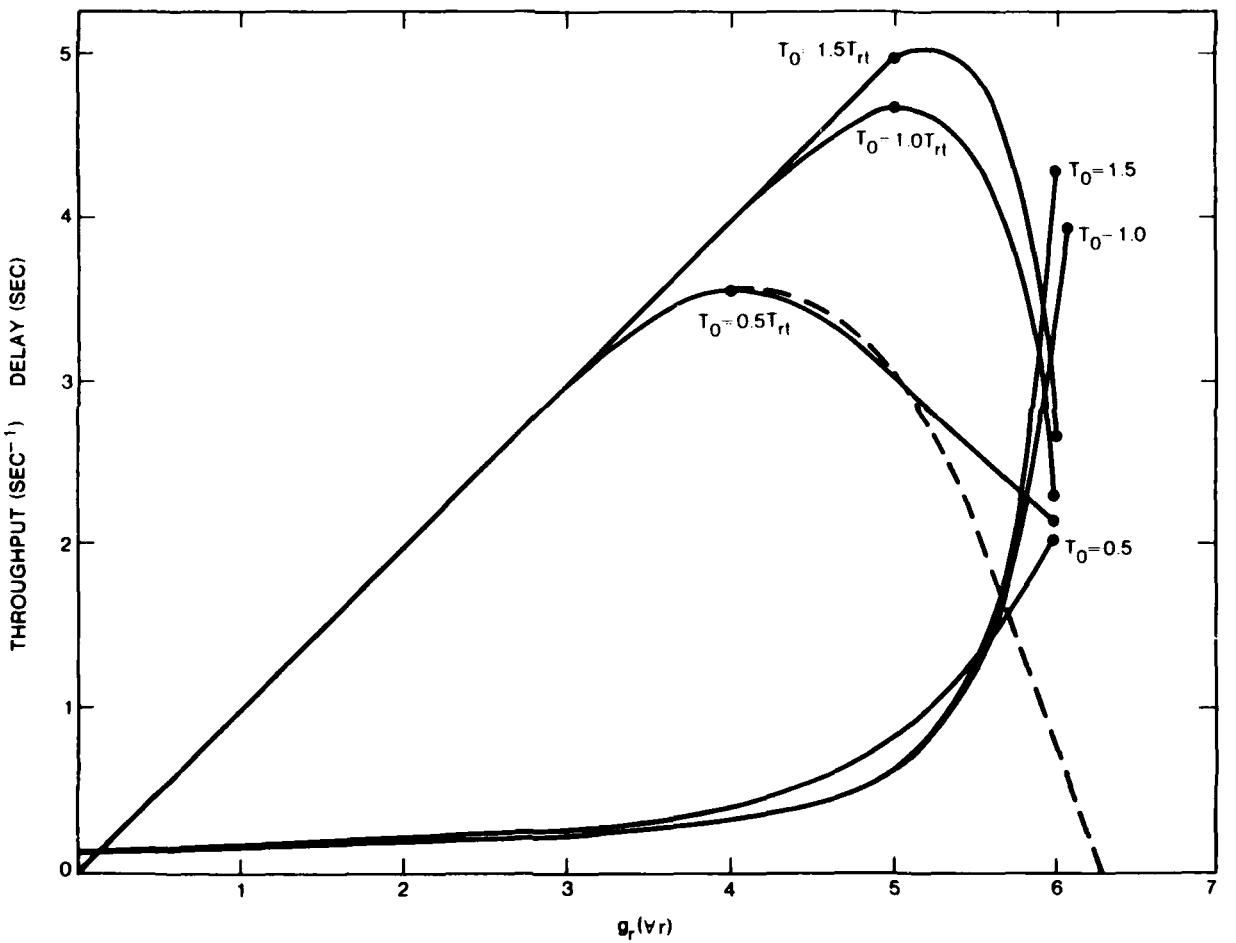


Figure 4.19. Throughput and delay as a function of time-out length as $P_{L_i} = 10^{-3}$.

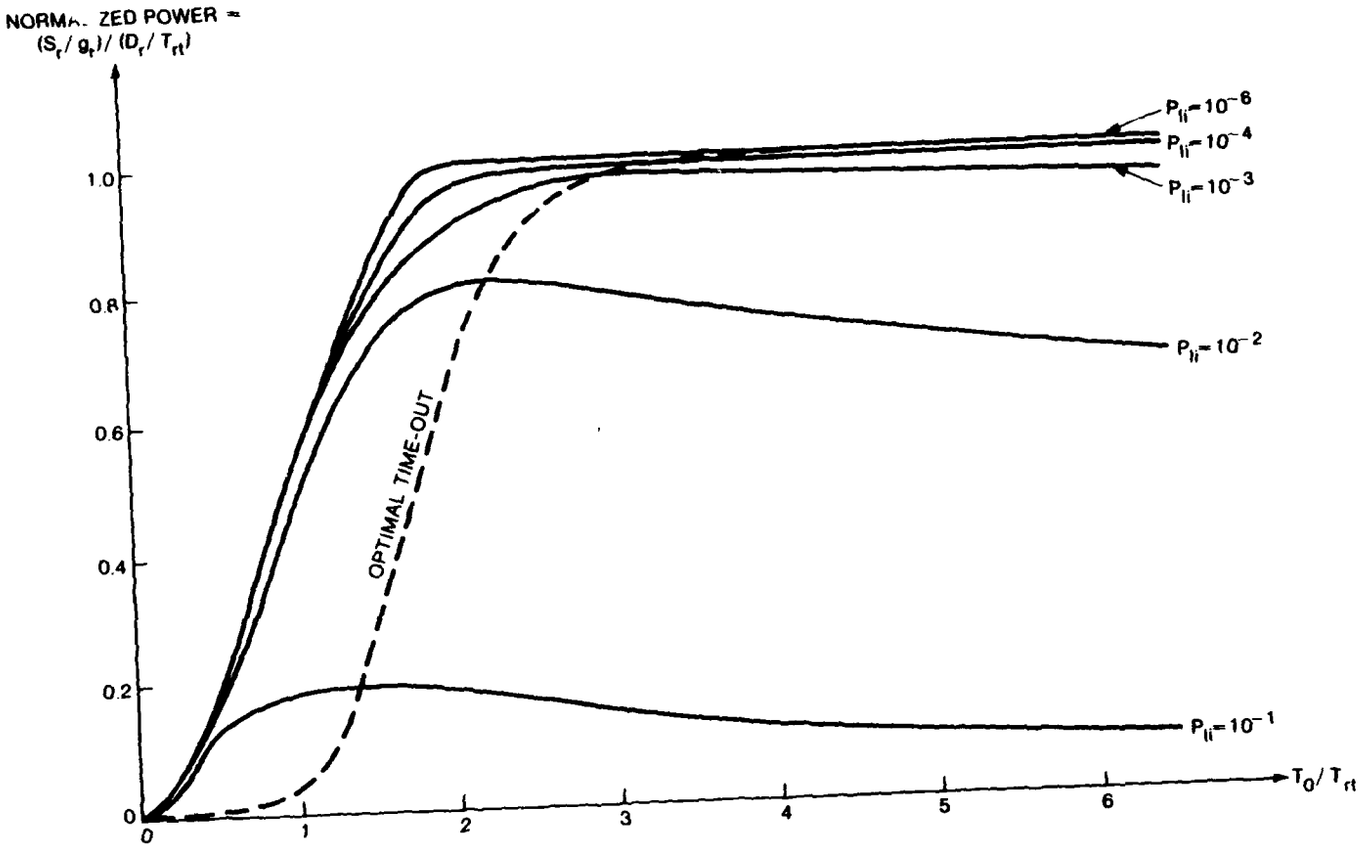


Figure 4.20. The optimal time-out scheme.

For simplicity let us assume that all flows from N_1 to N_2 are controlled by the permit queue q_p and a permit generation rate $g_{N_1N_2}$. The procedure used is similar to that used in Sec. 4.2 except that, contrary to Sec. 4.2, when an N_1–N_2 message finds no permit in queue q_p, it is not rejected. Instead, it will be queued in $q_{N_1N_2}$ to wait for a permit to be generated. The reason messages are not rejected at G is to assure the users that once the network accepts their messages it will do its best to deliver the messages to the destination. Also, for the sake of network utilization, it is not a good thing to throw away messages in which we have invested so much of the resources (by sending them from their sources to the gateway G).

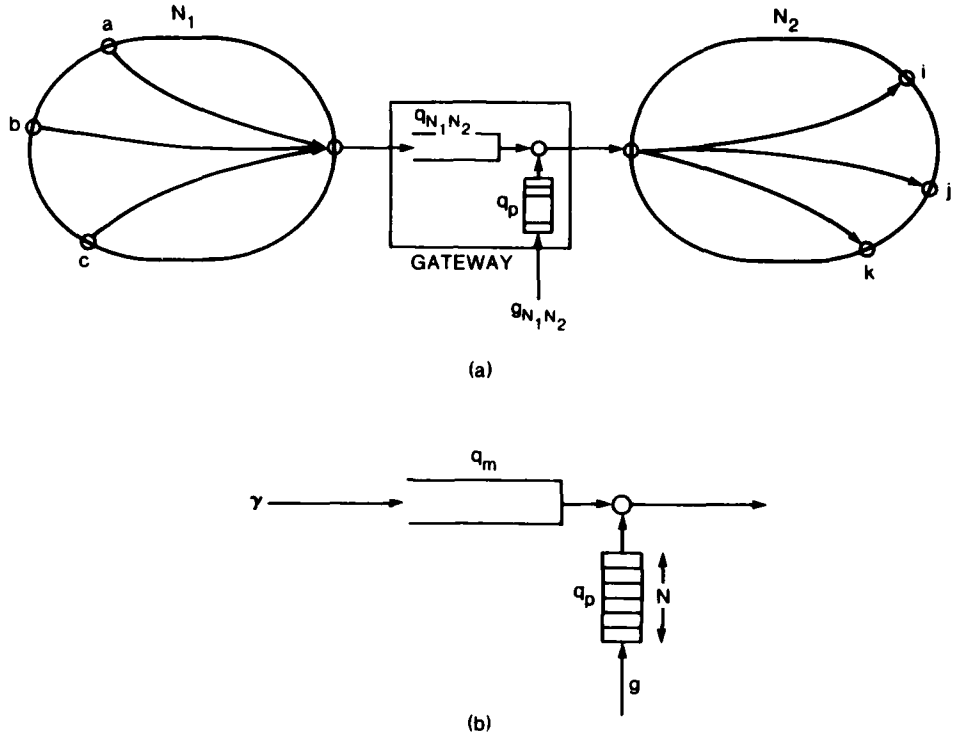


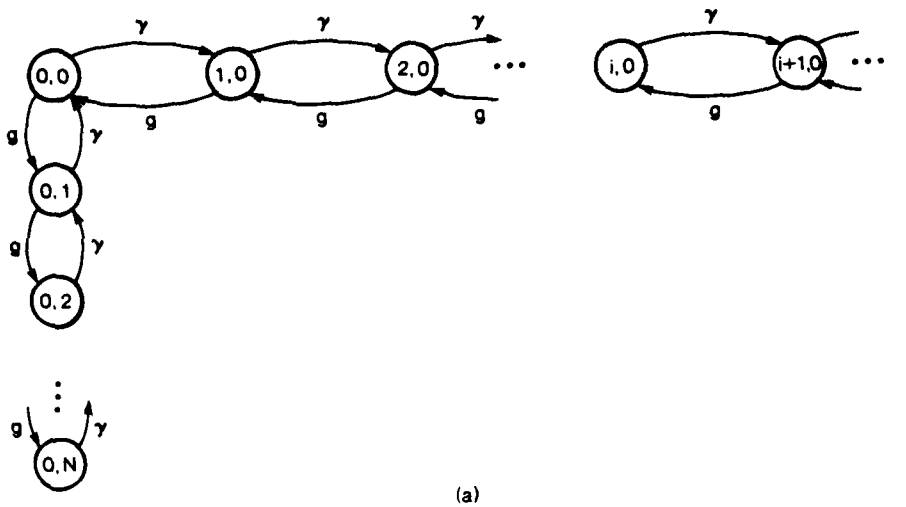
Figure 4.21. Flow control between networks.

The above discussion leads us to investigate the following scheme.

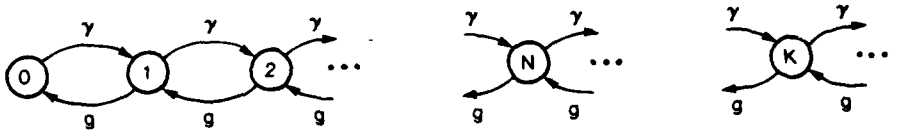
4.5.1 The Buffered Input Scheme

Consider the control scheme shown in Fig. 4.21(b). Assume that N_1 to N_2 messages arrive at q_m at a rate γ. They are accepted by network N_2 only if they can secure permits from q_p, which is a finite buffer of size N, and which accommodates permits generated at a rate g.

The system can be described by a Markov Chain as shown in Fig. 4.22(a). In the diagram, a state (i,j) means that there are i messages in q_m, and j permits in q_p. Observe that it is impossible for $i \neq 0$ and $j \neq 0$ simultaneously; and that j is always less or equal to N. Thus, a state is either $(i,0)$, or $(0,j)$ with $j \leq N$.



(a)



(b)

Figure 4.22. The state diagrams of $q_{N_1N_2}$ and q_m.

By transforming

$$(0,j) \rightarrow (N-j)$$

and

$$(i,0) \rightarrow (N+i)$$

the diagram in Fig. 4.22(a) becomes Figure 4.22(b).

This is just the state diagram of an M/M/1 queue! Therefore,

$$P'_K = (1-\frac{\gamma}{g})(\frac{\gamma}{g})^K = (1-\rho)\rho^K \tag{4.80}$$

where $\rho \triangleq \frac{\gamma}{g}$

After transforming back to the original system, we obtain

$$P_{0j} = P'_{N-j} = (1-\rho)\rho^{N-j}, \quad j = 0, 1, \cdots, N \tag{4.81}$$

and

$$P_{i0} = P'_{N+i} = (1-\rho)\rho^{N+i}, \quad i = 1, 2, \cdots \tag{4.82}$$

Thus, the average number of messages in q_m is

$$N_m = \sum_{i=1}^{\infty} i\, P_{i0} = (1-\rho)\rho^N \sum_{i=1}^{\infty} i\rho^i$$

$$= \frac{\rho^{N+1}}{(1-\rho)} \tag{4.83}$$

The average delay in Q_m is

$$D_m = \frac{(1/\gamma)\rho^{N+1}}{1-\rho} = \frac{\rho^N}{g-\gamma} \tag{4.84}$$

Observe that the inter-departure process from q_m is not Poisson. This can be seen as follows:

Case 1: The departure message sees q_m empty and q_p not empty; then the Laplace transform of the probability distribution of the interdeparture time is

$$A^*(s \mid \text{case 1}) = \frac{\gamma}{s+\gamma} \tag{4.85}$$

$$P[\text{case 1}] = \sum_{j=1}^{N-1} P_{0j} = (1-\rho) \sum_{j=1}^{N-1} \rho^{N-j} = \rho - \rho^N \tag{4.86}$$

Case 2: The departure message sees Q_m not empty and Q_p empty.

$$A^*(s \mid \text{case 2}) = \frac{g}{s+g} \tag{4.87}$$

$$P[\text{case 2}] = \sum_{i=1}^{\infty} P_{i0} = (1-\rho)\rho^N \sum_{i=1}^{\infty} \rho^i = \rho^{N+1} \tag{4.88}$$

Case 3: The departure sees both Q_m and Q_p empty.

$$A^*(s \mid \text{case 3}) = \frac{g}{\gamma+g} \cdot \frac{g}{s+g} + \frac{\gamma}{\gamma+g} \cdot \frac{\gamma}{s+\gamma} \tag{4.89}$$

$$P[\text{case 3}] = P_{00} = (1-\rho)\rho^N \tag{4.90}$$

Therefore,

$$A^*(s) = \sum_{i=1}^{3} A^*(s|\text{case } i)\; P[\text{case } i]$$

$$= \frac{N(s)}{(s+\gamma)(s+g)(s+\gamma+g)} \tag{4.91}$$

where $N(s)$ is a polynomial.

We see that $A^*(s)$ cannot be Poisson, since if it were, the $N(s)$ must cancel the denominator in such a way that only $S+\gamma$, $S+g$, or $S+\gamma+g$ remains.

This implies that if the inter-departure process is Poisson, then it is Poisson with a parameter γ, g, or $\gamma+g$. None of these can be the case for our system and therefore it is not Poisson.

We see that we can apply the above model to the flow control at gateway G in Figure 4.21(a). The entire system can be analyzed by separately analyzing N_1 and N_2. The only convenient assumption we have to make is that the traffic which enters N_2 from N_1 is Poisson (and which we have just shown is not true).

Since it does not generate any more insight to analyze N_1 and N_2 in detail, we shall leave this section here.

4.6. Conclusion

We have presented two flow control schemes, each based on limiting the permit generation rates for logical channels.

The advantages of these kinds of flow control schemes are:

(1) They require no end-to-end acknowledgements. We have shown that our schemes provide good network performance most of the time. Only at infrequent instants, when the traffic pattern and/or the service pattern deviates significantly from the normal situation, will the network have to invoke a special handling mechanism. In the usual case, no ACK's are needed for flow control purposes since our schemes use no windows. This may save considerable processing capacity as well as transmission bandwidth. Furthermore, our schemes become particularly suitable if, for some security reason, the destination node is not allowed to know the identity of the source, and hence no ACK can be routed back.

(2) The second scheme can easily be adjusted to the ideal flow control scheme as closely as we desire. With most other flow control schemes, we usually have no idea how to achieve such behavior.

(3) Our schemes are easy to implement. No complicated software or algorithms are involved.

(4) Most importantly, these procedures are easily evaluated analytically. They can save considerable effort in the network development phase. One need not write large, cumbersome, hard-to-debug programs to determine the performance impact of these flow control schemes on the network performance. Also, the computational simplicity of our schemes have allowed us to easily incorporate many other aspects of network protocols (such as piggy-back ACK's, and time-outs) into our model.

We wish to comment on some of our basic assumptions. First we have assumed that we have an infinite buffer in each node. This assumption liberates us from the painful storage arrangement (for flow control purpose) as well as the awful computational complexity. As storage elements become cheaper and cheaper, our assumption becomes more and more acceptable. Second, we have assumed that a node takes "no time" to process messages so that the only delay is because of transmission. Though this is quite true today, we would like to warn the readers that this might be reversed as faster and faster channels such as digitized channels, and optical fibers [RAWS 78] introduce themselves into the world. Third, we have neglected to model the point to point acknowledgement because this results in a simpler mathematical model. With better and better channels for digital communication available now and in the future we believe people may simply abandon the point to point acknowledgement and let errors, if any, be recovered end-to-end, or by a time-out scheme. Lastly, we wish to comment that the control scheme in this chapter aims to control the average behavior of a network only.

It does not stop input when occasionally the network is in a congested status. Hence some complimentary methods such as local congestion or dynamic flow control schemes are needed to overcome this weak point. However, as pointed out before, this is a common shortage shared by almost all global flow control schemes. Some dynamic flow control schemes are discussed in Chapter 5.

CHAPTER 5
DYNAMIC FLOW CONTROL SCHEMES

There is a Chinese saying, "No flower can be fresh for a hundred days, and no man can be sound for a hundred weeks." Indeed, nothing in this world is everlasting. Therefore no long-term plan can be carried out without dynamically adapting itself to the changing environment.

The flow control schemes discussed in Chapter 3 and 4 are static. They work well if the throttling thresholds (e.g., permit-generation rates) at various entry points of the network are set properly according to the external demands (input rates) at these entry points as well as to the network status.

Now, if we can dynamically adjust the throttling thresholds according to *changing* external demands and the internal state, we shall have a high performance network.

Four problems arise immediately. First, in a geographically distributed network, who is responsible for obtaining the information about the situation of external demands and the network status? Second, given that these responsible "administrative centers" have the information, what should they do? Third, how often should we readjust the throttling thresholds? And fourth, should the adjustment be done globally or in part of the network at a time?

These are questions with no simple answers. In fact there has been a lack in the literature in solving these problems. Kermani and Kleinrock [KERM 80] investigated a dynamic control scheme in which the destination for a single flow tries to control the source input rate according to its own buffer utilization. This scheme is intended to match the speed of the source and the destination of a *single* flow. A better control scheme must take the speed of the network and the effect of side traffic into account since a destination may cry for more input while the network is congested. Also it is worthwhile to mention that to analyze a single flow scheme as simple as the one that Kermani and Kleinrock investigated, they had to use the theory of "Markov decision processes", and only a heuristic solution could be obtained. One is then easily convinced that there hardly exists an easy way to analyze dynamic flow control for a large, geographically distributed, multi-variate network. With this in mind one may be less reluctant to accept the dynamic control schemes that we propose and analyze later in this chapter.

In this chapter we will assume that although the external demands do keep changing, they do not change too rapidly (compared to the relaxation time of the system*). This means that the rapid changes are considered as fluctuations of traffic, and only the D.C. shifts are considered as traffic changes.

---

* The relaxation time can be regarded as the time period for a system to enter into the new steady state from the old steady state because of a step change in the input rate. See [KLEI 76].

In the schemes we discuss below, we always require that the adjustment be done at regular time intervals, each of which is much larger than the relaxation time of the system. We call these adjusting moments, the "control points."

5.1 A General Dynamic Control Scheme

To answer the four questions raised above, one might suggest the following. Let some node in the network be the central administrative node to which every node in the network constantly reports the external demand at its site. The central node then regularly examines the external demands of the whole network, and then tries to allocate the network resources (e.g., permit generation rates) properly. By 'properly' we mean to

(1) satisfy the demands most fairly;

(2) maximize the total throughput of the network; and

(3) limit the delay below a certain acceptable value.

One way to do this is to solve the following nonlinear programming.

$$\textit{minimize} \left[\sum_r \left(\frac{\gamma_r - g_r}{\gamma_r} \right)^2 \right] / \sum_r g_r$$

subject to

$$\alpha \mu C_i \geq \sum_{r \in R(i)} g_r \quad , \forall \text{ physical channel } i$$

and

$$g_r \leq \gamma_r \ \forall \text{ logical channel } r$$

where γ_r is the external input rate of logical channel r, and g_r is the allocated permit generation rate for that logical channel; α is a multiplier < 1 to keep utilization of channels < 1; μC_i is the service rate at physical channel i; and

$$R(i) = \{\text{logical channel } r \mid \text{whose messages route through physical channel } i\}$$

The above nonlinear program tries to minimize the sum of the squares of the percentage deficit $\left(\frac{\gamma_r - g_r}{\gamma_r} \right)$ that the network owes to a logical channel r while keeping the total throughput $\sum_r g_r$ maximal. And this is under the condition that none of the physical channels is overloaded and that none of the g_r's are over allocated ($\leq \gamma_r$).

The advantage of a scheme such as this one is that the network resources can be allocated most efficiently because the entire situation of a network is taken into account in the allocation decision process. Yet there are two disadvantages. First, because it is centralized, if the central node is down, the flow control cannot function at all. Second, if the network is large, with hundreds of nodes and channels, then the calculation, transmission and processing of updates is so difficult and time consuming that the scheme may end up with no ability to control the network in real time. In fact, these considerations motivate us to investigate the following two *distributed dynamic* flow control schemes.

5.2 Distributed Dynamic Flow Control -- Scheme 1

5.2.1 Some Observations from Kleinrock's Delay Equation [KLEI 76]

We start with the formula

$$T_r = \sum_{i \in I(r)} \frac{1}{\mu C_i - S_i} \tag{5.1}$$

where T_r is the average round trip delay of logical channel r; S_i is the sum of the traffic over logical channels whose messages are routed through physical channel i; and $I(r) = \{\text{physical channel } i \mid \text{physical channel } i \text{ belongs to the path for logical channel } r\}$

Consider the simplest (but non-realistic) case that the logical channel consists of only one physical channel i^*. Then

$$T_r = T_{i^*} = \frac{1}{\mu C_{i^*} - S_{i^*}} \tag{5.2}$$

i.e.,

$$S_{i^*} = \mu C_{i^*} - \frac{1}{T_r} \tag{5.3}$$

Let $\rho^* < 1$ be the maximal allowed utilization factor for channel i^* (for a finite maximal average delay). Then corresponding to ρ^*, we have the maximum traffic (through channel i^*)

$$S^* = \rho^* \mu C_{i^*} \tag{5.4}$$

and the maximum channel delay

$$T^* = \frac{1}{\mu C_{i^*} - S^*} \tag{5.5}$$

i.e.,

$$S^* = \mu C_{i^*} - \frac{1}{T^*} \tag{5.6}$$

From Eqs. (5.3) and (5.6) we have

$$\delta S_{i^*} \triangleq S^* - S_{i^*} = \frac{1}{T_r} - \frac{1}{T^*} \tag{5.7}$$

Equation (5.7) tells us that in the one hop case, by measuring T_r, if $T_r < T^*$, one can increase the input rate by

$$\frac{1}{T_r} - \frac{1}{T^*}$$

without exceeding the maximum channel delay T^*.

This observation is the cornerstone of the dynamic distributed flow control scheme that we investigate in this section.

Usually the path of a logical channel consists of K ($K>1$) physical channels. Imagine that we sit at the source of logical channel r. We measure the average round trip delay T_r there. We assume that T_r is mainly because of the delay at some single physical channel i^* along the path of logical channel r †. Then if $T_r < T_{r_0}$ (T_{r_0} is the maximal allowed round trip delay of logical channel r), we will increase S_r by δS_r such that

$$\delta S_r = \frac{1}{T_r} - \frac{1}{T_{r_0}} \tag{5.8}$$

Let us illustrate the effect of doing this in Fig. 5.1.

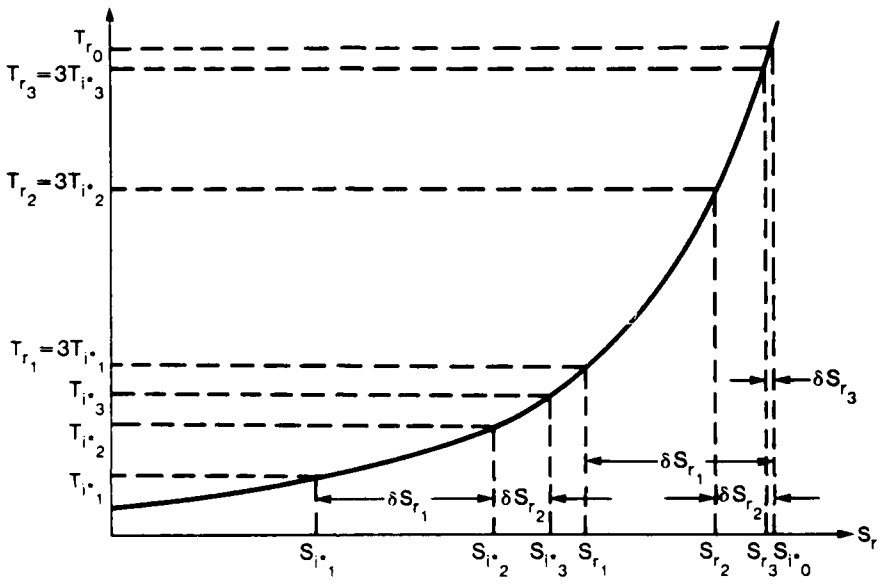


Figure 5.1. Multi-step adjustment of input rate.

Assume $K = 3$ (again, "3" is chosen just for simplicity; it may be unrealistic since round-trip paths are more likely to consist of an even number of hops). And for the sake of easy illustration let us assume that the channel utilizations and the channel speeds in these three channels are the same so that the actual delay in each channel T_i is

$$T_i = T_r/3$$

Let T_{r_i} be the measured average round-trip delay at the i^{th} control point. Now since at the source we think T_{r_1} ($= 3T_{i^*_1}$, see Fig. 5.1) is the delay at channel i^*, we also would think that the traffic at channel i^* is S_{r_1} while it is actually $S_{i^*_1}$. Thus, we conclude that we can increase S_r by

† This is not an unrealistic assumption. The channel capacities and the traffic matrix in a network are usually heterogeneous. Therefore, some channel may be saturated before other channels are.

$$\delta S_{i_1} = \frac{1}{T_{i_1}} - \frac{1}{T_{i_0}}$$

which brings S_{i_1} to S_{i_2}. At the second adjusting moment, the measured delay would become $T_{i_2} (= 3\,T_{i_2})$. This will give us δS_{i_2}. By adding δS_{i_2} to S_i we will have $S_{i_2} = S_{i_3}$. And in turn, at the third adjusting moment we will obtain the measured delay $T_{i_3} (= 3\,T_{i_3})$. Eventually we will reach T_{i_0}.

Based on the above observation, the following distributed dynamic control scheme is proposed. We again emphasize that the assumption that delay is lumped at a single channel is not a bad one since the network capacities are usually heterogeneous and the traffic uneven.

5.2.2 The Measure-and-Control Scheme

In this section we present a scheme that controls the flow by having each source measure the round-trip delay of its message, and according to this measurement, set an appropriate permit generation rate (see Chapter 4) for itself unless it obtains a control signal telling it not to do so. We describe the scheme in detail by presenting the following rules:

(1) Originally each logical channel i is assigned a basic legal permit-generation rate g_{i_0}. This is the rate at which a source can operate whenever it desires (e.g., as the input rate exceeds g_{i_0}). In the long run, as long as each source operates at this rate, the network is not going to be overloaded.

(2) Each source measures the external input rate γ_i, and the average round trip delay T_i.

(3) At each control point (i.e., adjusting moment) if a source finds that $\gamma_i > g_i$ (g_i is the current permit-generation rate) and $T_i < T_{i_0}$, it increases g_i by

$$\delta g_i = \frac{1}{T_i} - \frac{1}{T_{i_0}}$$

But if $T_i > T_{i_0}$, it does nothing.

(4) If it finds

$$\gamma_i < g_i$$

then it reduces g_i to γ_i.

(5) Whenever a physical channel (not necessarily at control points) finds its backlog is larger than N_u, it issues an instruction to all the sources whose messages are routed through it. This instruction tells each source to reset its g_i to g_{i_0} (if $g_i > g_{i_0}$) until further notification.

(6) A channel, when its backlog is reduced to N_l from N_u, will issue a relief signal to the relevant sources so that they can start the measure-and-control procedure again.

(7) Control instructions are sent with the highest priority (i.e., they bypass queues for data messages). The reason for points (5), (6) and (7) is that when T_i is large (i.e., somewhere the backlog is large), the source can no longer control the traffic in real time because it receives the information too late. The channel with the critical backlog has to immediately instruct the source to reduce its input rate by cutting its permit

generation rate to a safe value.

This scheme is certainly distributed and dynamic. Failures in some parts of a network do not affect (in the flow control sense) the other irrelevant parts of the network. No complicated computation is involved and information is not transferred back and forth globally, but only among relevant parties. Fig. 5.2 shows a typical performance curve of two logical channels sharing some common path that we may see when we apply the scheme to a network.

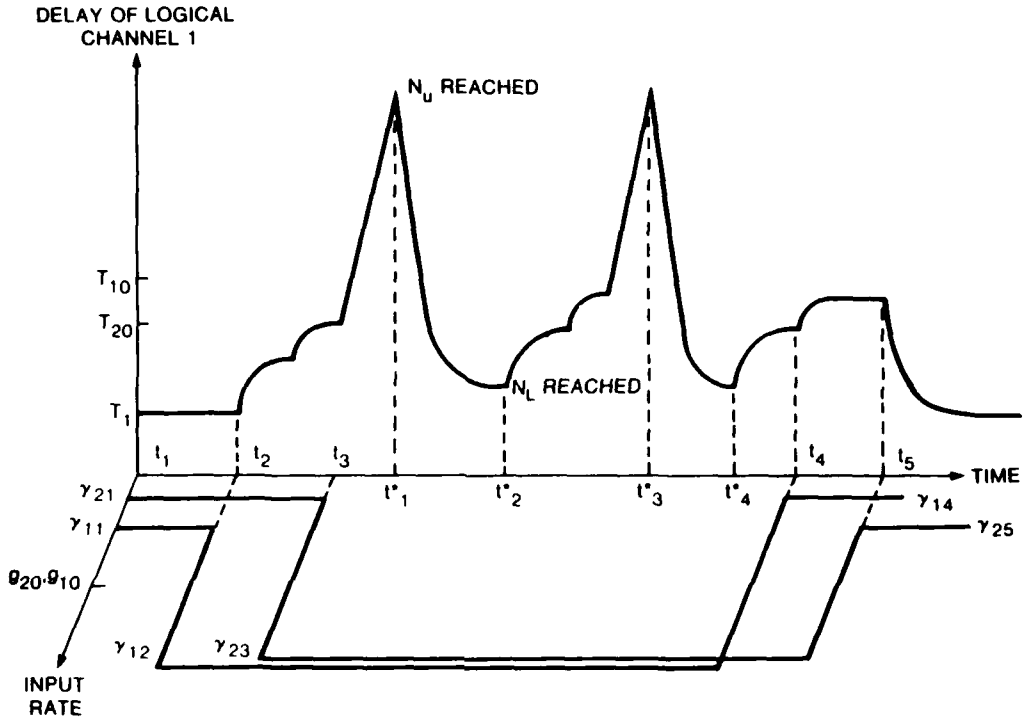


Figure 5.2. Typical delay curve of Scheme 1.

In this diagram, at a control point t_i, the measured input rate of logical channel k (k=1,2) is γ_{ki}. Between (t_1, t_2), the input rates are small ($<<g_{i0}$), so that the measured delay T_1 of logical channel 1 is also small and $g_i=\gamma_i$ (rule 4). At t_2, the measured input rate of logical channel 1 steps to γ_{12} ($>g_{10}$) and it first steps its g-value to g_{10} (rule 1). Then logical channel 1 increases its g-value according to rule 3 (but if the resultant g value is smaller than g_{10}, it is set to $g = g_{10}$, according to rule 1). It would take several control points for logical channel 1 to increase its g-value to the final value (corresponding to T_{10}). But before this final value is reached, the measured input rate of logical channel 2 increases to γ_{23} at t_3, and the measured delay T_2 is near T_{20}. Logical channel 2 would just operate at g_{20} according to rule 1. This will cause some physical channel to be overloaded ($\rho_{i^*} > 1$), and the delay increases linearly until t^*_1 where the backlog in physical channel i^* reaches the limit N_u. The g-values of logical channel 1 and 2 are then reset to g_{10} and g_{20} respectively (rule 5). This causes the backlog to decay. At t^*_2 the relief threshold N_L is reached, and the measure-and-control

mechanism starts again (rule 6). The same cycle repeats during (t_2^*, t_4^*). At t_4 the input rate of logical channel 1 steps down, while that of logical channel 2 is still high. Logical channel 2 starts to gain its g-value at each control point. The delay of logical channel 1 still increases since they share some physical channels in their paths. At t_4 the input rate of logical channel 2 also steps down. Therefore the delay also decays to a lower value.

5.3 The Simulation Results

We tested the measure-and-control scheme by simulation on our little network in Fig. 4.2. Here we change the capacity of physical channel 5-3 to 48 (messages per second) in order to make the network a little bit non-uniform. Other physical channels have capacities of 50. The results are shown in Figs. 5.3(a), (b), (c) and (d).

In the simulation we kept the input rates of all logical channels (other than 5-4 and 5-2) randomly under 3 messages per second, in order to single out the effects of changing the input rate of logical channels 5-2 and 5-4. The g_{r0} values are set at 5.7 so that when all logical channels operate at this value the most loaded physical channel has traffic at $5.7 \times 8 = 45.6$ messages per second (8 comes from the fact that, for example, through physical channel 3-2 we have four forward logical channels 5-2, 5-4, 3-2, 3-4 and four acknowledgement channels for logical channels 4-3, 4-5, 2-3, 2-5). The maximum utilization factor $\rho^* = \frac{45.6}{48} = 0.95$ (for physical channel 5-3).

The input rates of logical channels 5-2 and 5-4 are shown in Fig. 5.3(a). The measured delays are shown in Fig. 5.3(b). The control point intervals are one minute long †. Observe the similarity between Fig. 5.3(b) and Fig. 5.2. The reset instants are indicated by the vertical lines. The upper limit N_u on the backlog of any physical channel is 60, and the relief threshold is 30. The path of logical channel 5-4 is longer than that of 5-2. Hence the delay of 5-4 is longer than that of 5-2. However, at the peaks the two delay curves come very close to each other. This is because one of the physical channels is overloaded while others still have a utilization factor much smaller than one. This justifies our assumption of the existence of a critical physical channel.

In Fig. 5.3(c) we show the adjustments of the permit generation rates of the two logical channels. The vertical lines again indicate the resetting moments. Observe that logical channel 5-2 always gains in its g-value faster than logical channel 5-4. The reason behind this is explained below.

Let r_1 and r_2 be two logical channels such that π_{r_2} contains π_{r_1}, where π_{r_i}, $i=1,2$, is the path of logical channel r_i. Suppose that at the same control point the measured average delays are T_{r_1} and T_{r_2} respectively, and they are somewhat smaller than T_{r1_0} and T_{r2_0} (the maximum allowable values) respectively.

Then,

† If $\rho^* = 0.95$ for our little network then the relaxation time for the queue at physical channel i^* is $\tau = \frac{\sigma_{i^*}^2}{(\rho^* - 1)^2} = \frac{45.6(1/48)^2}{(0.05)^2} \simeq 7$seconds. 1/48 seconds is the service time of physical channel 5-3. This shows that one minute is much larger than the relaxation time.

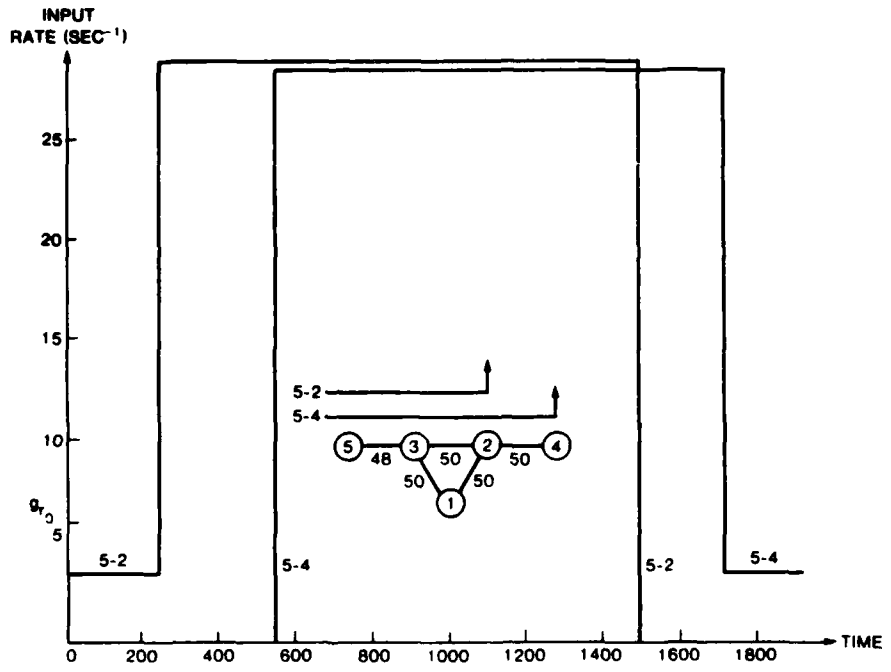


Figure 5.3(a). The input rates of logical channels 5-2 and 5-4.

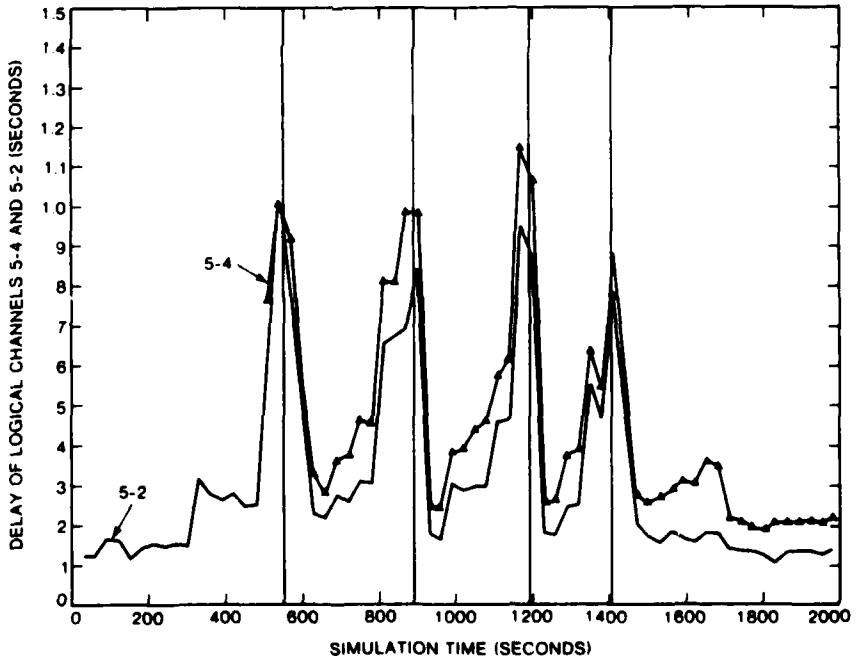


Figure 5.3(b). The simulated delay of Scheme 1.

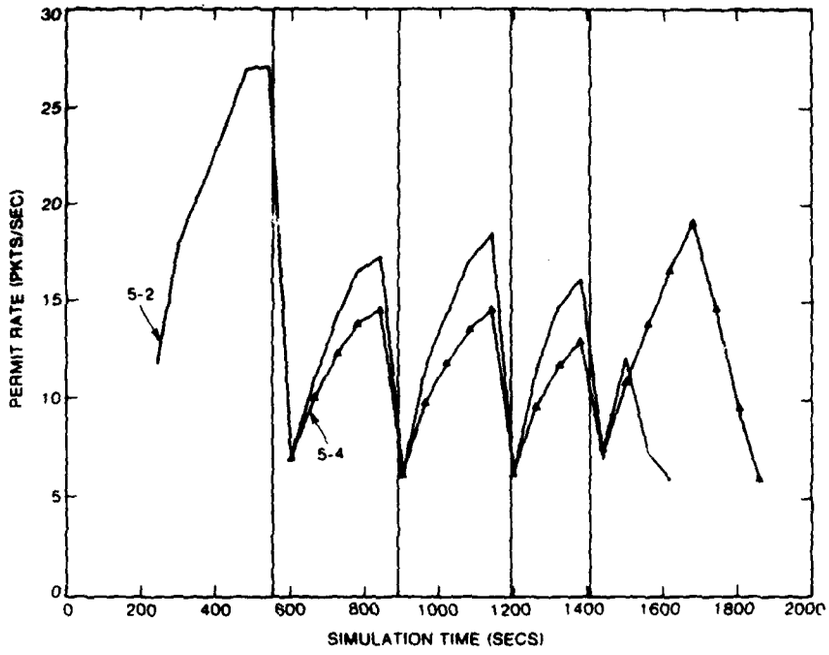


Figure 5.3(c). The simulated permit gain of Scheme 1.

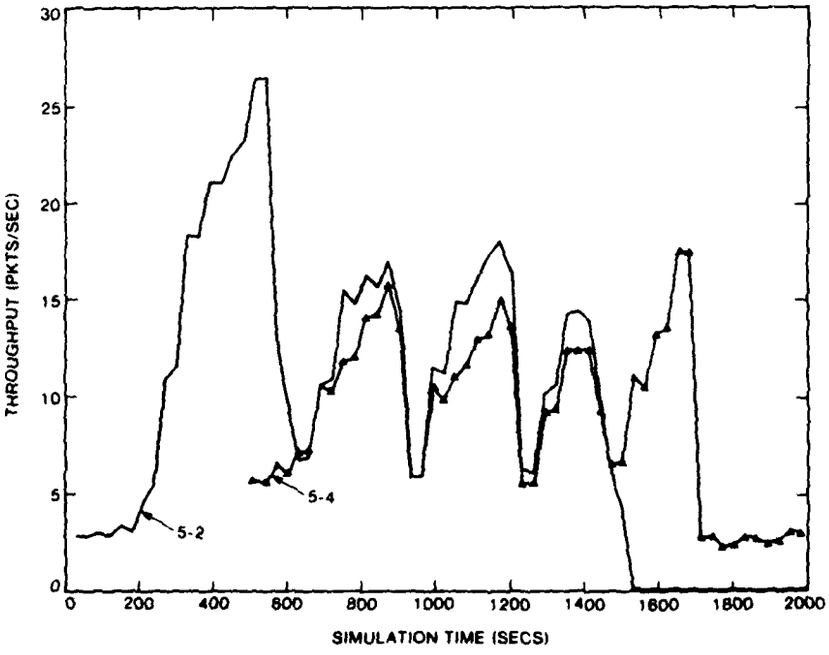


Figure 5.3(d). The simulated throughput of Scheme 1.

$$T_{r_1} = \frac{1}{\mu C_{i^*} - S_{i^*}} + \sum_{j \in \pi_{r_1} \cap \pi_{r_2}} \frac{1}{\mu C_j - S_j} < T_{r_2}$$

$$= \frac{1}{\mu C_{i^*} - S_{i^*}} + \sum_{j \in \pi_{r_1} \cap \pi_{r_2}} \frac{1}{\mu C_j - S_j}$$

$$+ \sum_{j \in \pi_{r_2},\, j \notin \pi_{r_1} \cap \pi_{r_2}} \frac{1}{\mu C_j - S_j}$$

Let $a = \dfrac{1}{\mu C_{i^*} - S_{i^*}}$

$$b = \sum_{j \in \pi_{r_1} \cap \pi_{r_2}} \frac{1}{\mu C_j - S_j}$$

$$c = \sum_{j \in \pi_{r_2},\, j \notin \pi_{r_1} \cap \pi_{r_2}} \frac{1}{\mu C_j - S_j}$$

Then

$$\delta g_{r_2} - \delta g_{r_1} = \left(\frac{1}{T_{r_2}} - \frac{1}{T_{r2_0}}\right) - \left(\frac{1}{T_{r_1}} - \frac{1}{T_{r1_0}}\right)$$

$$\simeq \frac{1}{T_{r_2}} - \frac{1}{T_{r_1}} = \frac{1}{a+b+c} - \frac{1}{a+b} < 0$$

i.e., $\delta g_{r_2} < \delta g_{r_1}$. We have used the fact that $\dfrac{1}{T_{r1_0}} \simeq \dfrac{1}{T_{r2_0}} \simeq 0$.

This is exactly what we observed in Fig. 5.3 since π_{5-4} contains π_{5-2}.

Intuitively speaking, this is because a logical channel with a longer path tends to be more inaccurate than one with a shorter path when it assumes that the delay is lumped at some single critical physical channel, while actually every physical channel in the path contributes to the delay.

One may wonder if, in order to eliminate the above mentioned "capture" effect, we should take the path length into consideration, e.g., to multiply the incremental g-value with a parameter $\alpha_r \geq 1$, that is a function of path length of the logical channel r. Indeed, we can do so, except for the fact that it is easier to make a misjudgement for the incremental g-value since α_r is also a function of the traffic matrix, which may change dramatically with time. Thus, there is no simple optimal way to specify the value of α_r.

The difficulty in accurately adjusting the g-value is an inherent nature of distributed decision processes.

Fig. 5.3(d) shows the throughput of logical channels 5-2 and 5-4. One should compare this with the input rates. The scheme does dynamically control the flow reasonably well (except for the capture effect indicated before).

The oscillation can be minimized by a smoother resetting mechanism. For example, a scheme with multi-threshold levels and multi-steps for resetting g-values can be used. Indeed, we shall use this type of resetting scheme in an example of scheme 2.

5.4 Distributed Dynamic Flow Control - Scheme 2

In this section we present a less distributed dynamic control scheme that allocates the g-values more effectively.

In this scheme, at τ seconds prior to each control point, the sources and physical channels exchange information for N_c cycles. N_c is a proper number chosen by the designer of the control scheme. At each cycle a source passes information about the current demand and δg-value gained from the last cycle to the physical channels along the path of the logical channel, and the physical channels pass the information about their allocation of current δg-values to the source. The details are described below.

5.4.1 The Overall Rules

(1) τ seconds prior to each control point, execute the "g-value allocation" mechanism in Sec. 5.4.2 for N_c cycles.

(2) At each control point, each source adjusts its g-value to that obtained by rule (1), and the network operates at these new g-values.

(3) Whenever the input rate γ_r of a source r steps up from a low value ($<g_{r0}$) to a high one ($>g_{r0}$), it can operate at g_{r0} immediately whether or not the next control point has arrived.

(4) *If rule (3) ever causes any physical channel to be overloaded, the physical channel will activate a resetting mechanism as various thresholds of backlog are reached.*

5.4.2 g Value Allocation Rules at Each Cycle

(1) Each source j passes $(\gamma_j, \delta g_j)$ to each physical channel i along its path to its destination, where γ_j is the source's current demand and δg_j is the incremental g-value gained during the last cycle. Note that for the first cycle, γ_j is the external input rate and $\delta g_j = 0$. The demand of the next cycle of source j is $\gamma_j = \gamma_j$ of the current cycle less δg_j gained in this cycle.

(2) A physical channel i, upon receiving $(\gamma_j, \delta g_j)$ $\forall$ source $j \in R(i)$, allocates its residual current capacity C_i to these sources such that source j receives capacity A_{ji} according to the allocation algorithm in Sec. 5.4.3. Again, note that C_i at the first cycle is the capacity of channel i. C_i at the current cycle is

$$C_i = (C_i \text{ of last cycle}) - \sum_{j \in R(i)} \delta g_j \text{ (last cycle)}$$

(3) Channel i then passes A_{ji} to each source j. Source j picks up its new δg_j value with $\delta g_j = \min_{i \in I(j)} A_{ji}$. δg_j is then added to the g-value allocated to source j so far (up to the last cycle).

5.4.3 The A_{ji} Allocation Algorithm

For each iteration k in a cycle m,

S_k = {source | its demand in this iteration $\neq 0$}
M_k = $|S_k|$, the Cardinal number of S_k
γ_{j_k} = demand of source j in this iteration (k)
C_{ik} = residual capacity of channel i in this iteration (k)
C_{i0} = The beginning capacity of the cycle (m) (i.e., that left after cycle $m-1$).
and let A_{ji} = 0 initially, for all $j \in S_0$.

The following flowchart illustrates the algorithm:

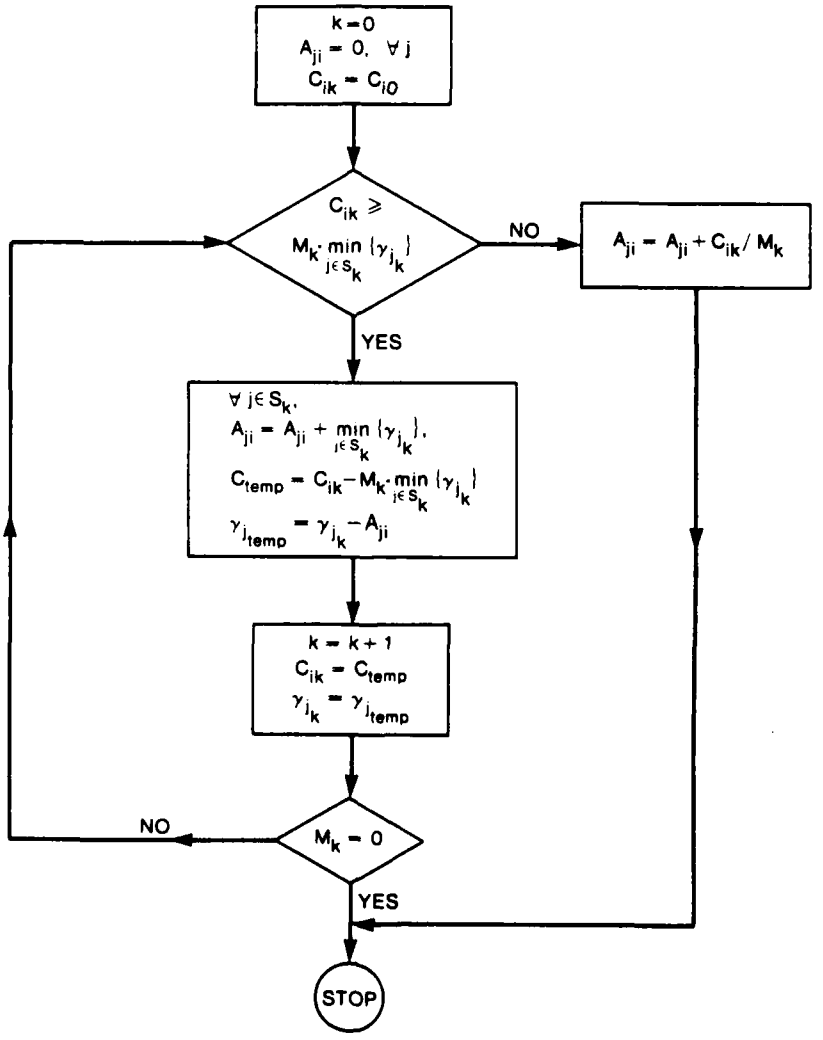


Table 5.1 Illustrative example of A_μ allocation algorithm.

	Initial	1st Iteration	2nd Iteration	3rd Iteration	Final Values
Demand	5_0	0_5	0_0	0_0	0_5
Subscript	7_0	2_5	0_2	0_0	0_7
g Allocated	9_0	4_5	2_2	0_2	0_9
	12_0	7_5	5_2	3_2	3_9
Capacity Remaining	30	10	4	0	0

Table 5.1 shows the result of a capacity allocation example where a physical channel tries to allocate its capacity to four logical channels. In this example, the four sources route their messages through a physical channel with a capacity of 30. The initial demands of these logical channels are 5, 7, 9 and 12, respectively. The initial allocated g-values are 0. After the first iteration (see the allocation flow chart) the allocated g-values are 5,5,5,5. The residual capacity of the physical channel is 10, and the current demands become 0,2,4,7. After the second iteration the allocated incremental g-values are 0,2,2,2, the remaining capacity is 4, and the current demands are 0,0,2,5. After the third iteration, the incremented g-values are 0,0,2,2, the residual capacity is 0, and the current demands are 0,0,0,3. So, finally the four sources have g-values 5,7,9,9 with the fourth source not satisfied by 3 units. The capacity of the physical channel is used up.

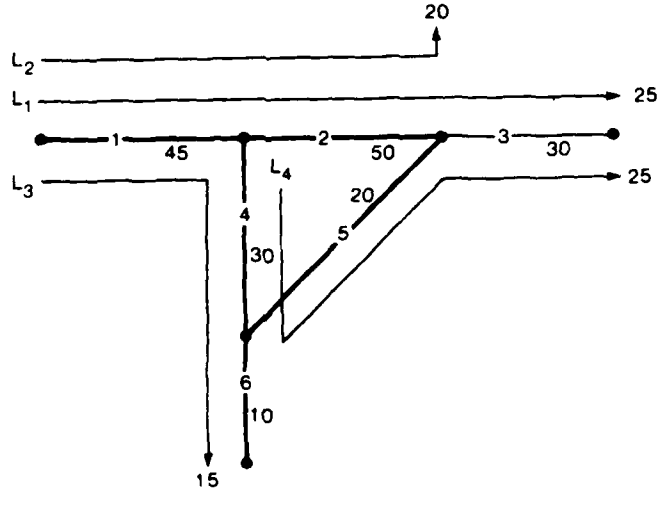


Figure 5.4. Illustrative network for dynamic control scheme 2.

To illustrate how the flow control scheme works, we apply 3 cycles of the set of rules in Sec. 5.4.2 to the network shown in Fig. 5.4. Here we show a network with six physical channels having capacities as denoted, and we show four logical channels L_1, L_2, L_3, and L_4 with initial demands of 25, 20, 15, and 25. The results of applying 3 cycles of the g-value allocation rules are summarized in Table 5.2.

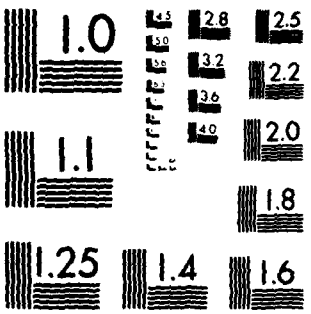
Table 5.2 Results of applying 3 cycles of g-value allocation rules.

	1st Cycle		2nd Cycle		3rd Cycle		
Logical Channel	$(\gamma_l, \delta g_l)$	δg_l obtained	$(\gamma_l, \delta g_l)$	δg_l obtained	$(\gamma_l, \delta g_l)$	δg_l obtained	Total g_l obtained
L_1	(25,0)	15	(10,15)	0	(10,0)	0	15
L_2	(20,0)	15	(5,15)	$\frac{5}{3}$	$\left(\frac{10}{3}, \frac{5}{3}\right)$	$\frac{10}{9}$	17.78
L_3	(15,0)	10	(5,10)	0	(5,0)	0	10
L_4	(25,0)	15	(10,15)	0	(10,0)	0	15
Physical Channel	Capacity Left	Allocation	Capacity Left	Allocation	Capacity Left	Allocation	Capacity Left
1	45	15,15,15	5	$\frac{5}{3}, \frac{5}{3}, \frac{5}{3}$	$\frac{10}{3}$	$\frac{10}{9}, \frac{10}{9}, \frac{10}{9}$	$\frac{20}{9}$
2	50	20,25	20	10,5	$\frac{55}{3}$	$10, \frac{10}{3}$	$\frac{155}{9}$
3	30	15,15	0	0,0	0	0,0	0
4	30	15,15	5	$\frac{5}{2}, \frac{5}{2}$	5	$\frac{5}{2}, \frac{5}{2}$	5
5	20	20	5	5	5	5	5
6	10	10	0	0	0	0	0

Let us follow the activity of physical channel 2. Initially it has a capacity of 50. At the first cycle it allocates 20 and 25 to L_1 and L_2 respectively. Yet L_1 and L_2 obtain only 15 and 15 of the g-values because the allocation made by physical channel 1 is 15, 15, 15, for L_1, L_2, L_3. At the beginning of the second cycle, physical channel 2 has capacity of 20 (=50-15-15) remaining. The residual demands of L_1 and L_2 are 10 and 5 so it allocates 10,5 to L_1 and L_2 respectively. But since channel 1 allocates 5/3, 5/3, 5/3 to L_1, L_2, L_3; and channel 3 allocates 0 and 0 to L_1 and L_4, the δg-values obtained by L_1 and L_2 in this cycle are 0, 5/3. Now, at the beginning of the third cycle channel 2 has a capacity of 55/3 (=20-5/3) remaining. It in turn allocates 10 and 10/3 to L_1 and L_2. But δg gained by L_1 and L_2 in this cycle is 0,

AD-A130 784 FLOW CONTROL IN STORE-AND-FORWARD COMPUTER NETWORKS(U) 2/2
CALIFORNIA UNIV LOS ANGELES DEPT OF COMPUTER SCIENCE
C TSENG APR 81 UCLA-ENG-8110 MDA903-77-C-0272

UNCLASSIFIED F/G 12/1 NL



It in turn allocates 10 and 10/3 to L_1 and L_2. But δg gained by L_1 and L_2 in this cycle is 0, 10/9. Therefore, finally channel 2 has capacity of 155/9 (= 55/3 - 10/9) remaining. The activities of other physical channels can be followed similarly.

The resulting g-values obtained by L_1, L_2, L_3 and L_4 are 15, 17.78, 10, and 15.

The final g-values are the combined products of several factors, namely, the fairness consideration, the topology constraint, the channel capacities and the routing algorithm. Observe that as $N_c \rightarrow \infty$, g-values converge to their asymptotic values.

Since rule (3) in Sec. 5.4.1 allows a source to step up to its legal g_{r_0}-value, overload occasionally occurs. Therefore a resetting mechanism as stated in rule (4) is necessary.

One example of a resetting mechanism is shown in Fig. 5.5. In the figure, the horizontal lines represent the backlog of the physical channel. If every physical channel is in region (a,b), (b,d), (e,f), or (h,i) the g-values are decided at each control point by the rules in Sec. 5.4.1. If a physical channel is on (b,c) or (c,e), the g-values of all sources whose messages pass through it are instructed to reduce its g-value by 10%. But, if source r has a g-value g_r smaller than g_{r0}, it would still operate at the g_r level even if it is instructed to reduce 10%. Finally, if it is on (c,g) or (g,h), the relevant sources can only operate at g_{r0}'s (or lower) to force the network to be stabilized.

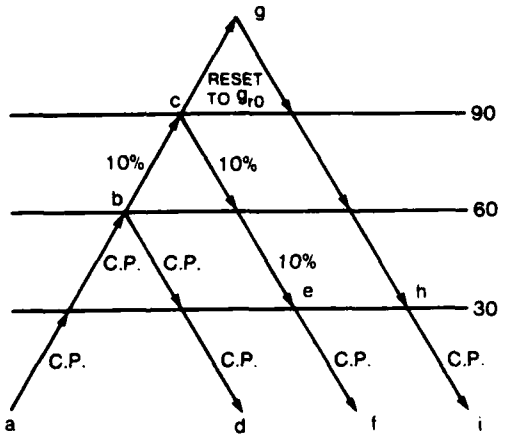


Figure 5.5. A resetting mechanism.

5.5 The Simulation Results

We tested scheme 2 on the little network in Fig. 4.2 that we have mentioned so many times. We use the same parameter values as in Sec. 5.3.

the g-values of 5-4 and 5-2 allocated at control points. Table 5.3 shows the resetting activity during the simulation. Figs. 5.6(c) and 5.6(d) show the throughputs and delays of the logical channels 5-4 and 5-2.

Table 5.3. Reset activity.

Time (sec.)	Physical Channel	Reset Activity	Queue Length
502.710	5-3	10%	65
638.384	5-3	10%	62
672.481	5-3	10%	61
749.069	5-3	10%	62
813.486	5-3	10%	61
883.949	5-3	10%	61
901.196	3-5	10%	62
909.225	3-5		93
1361.256	3-5	10%	61
1453.903	5-3	10%	66
1504.270	5-3	10%	61
1706.956	5-3	10%	61

One observes that indeed there is no more capture effect in this scheme. g-values are assigned according to the demand and the network capacity. The reader is urged to compare the input rates and the g-values assigned. Also note that it takes only one control point to allocate the g-values, while in the first scheme it takes several control points. Yet the effectiveness of allocating capacity is gained at the cost of passing information back and forth several rounds between the relevant parties. (*In our simulated case the cost is 1.5 sec. per control point.*)

Refer to Table 5.2. It is observed that if in a cycle, a physical channel i receives a $(\gamma_j, \delta g_j)$ from source j with $\delta g_j = 0$, then the physical channel can actually determine that some other physical channel assigned a zero g-value to source j during the last cycle, and that physical channel is to assign a zero g-value to source j again in this cycle. This means that physical channel i can regard the residual demand γ_j of source j to be zero even though it is not. By doing this, channel i can allocate more of its residual capacity to other sources (whose messages pass through it) with $\delta g_j \neq 0$ and $\gamma_j \neq 0$. This results in a more effective allocation mechanism.

5.6 Conclusion and Remarks

People have long recognized that dynamic flow control is needed to accommodate a changing environment. Yet the difficulty lies in the fact that the geographic distribution of the system makes it difficult to obtain the information needed in time for a real-time dynamic control. Also, the largeness of the system may make the computational effort too much to be practical. Furthermore, it is also realized that in a distributed system, we should not rely on one or a few centers to make decisions that are important for the whole system. Otherwise the whole system may be seriously affected when these centers are down.

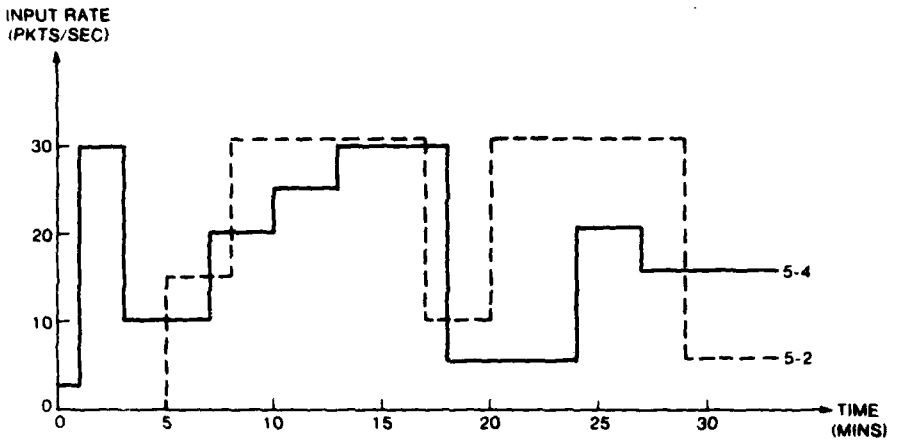


Figure 5.6(a). The input of Scheme 2.

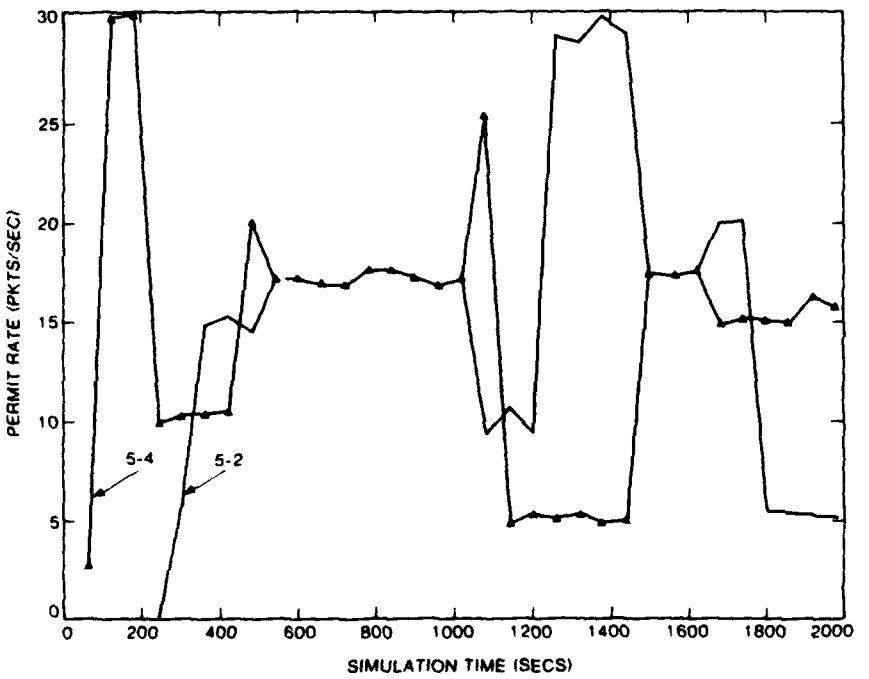


Figure 5.6(b). The permit gain of Scheme 2.

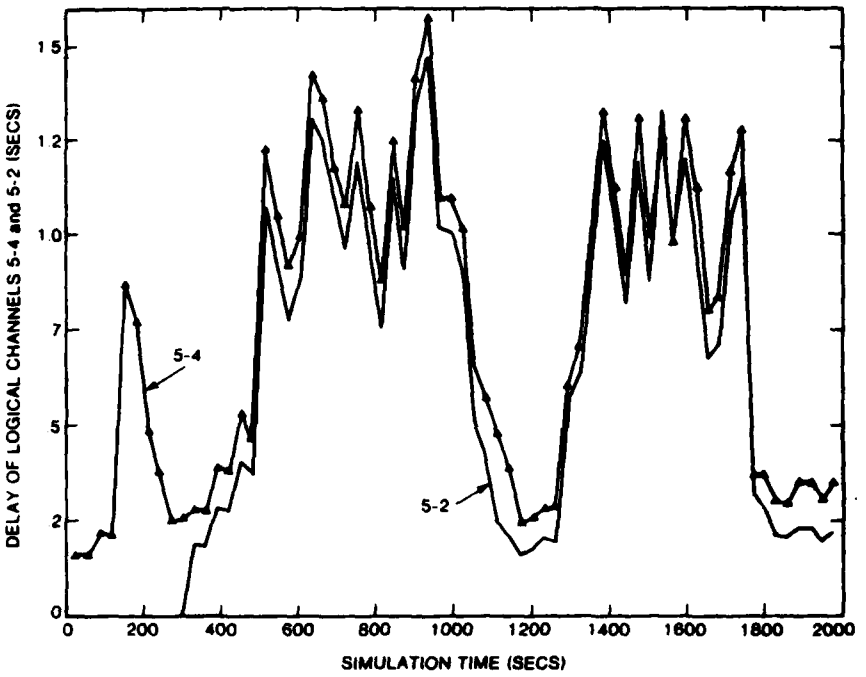


Figure 5.6(c). The delay of Scheme 2.

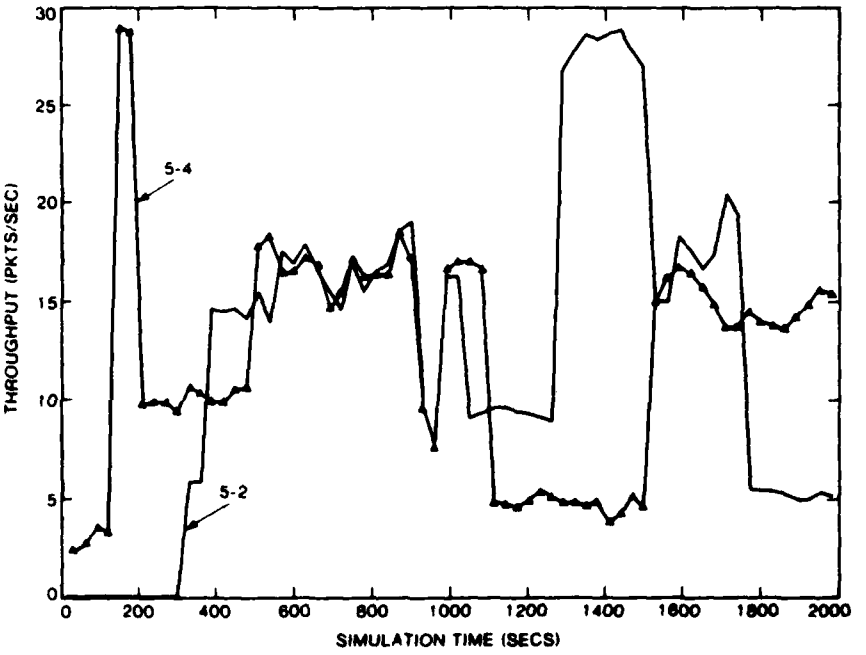


Figure 5.6(d). The throughput of Scheme 2.

The above thoughts then motivate our research in distributed dynamic control. What we really want is to seek methods whereby each individual party of the system can act properly by consulting as few other parties of the system as possible.

We immediately see the contradiction here. In a large dynamically changing system if you want proper behavior you must keep track of the state of the system and that requires some consulting between parties.

In the first scheme (Sec. 5.2) the "consulting" is imbedded in the measurement of the round trip delay, that tells the source about the collective activity level of the physical channels along its path. The only one who makes a decision for the source is the source itself. (Of course, when sources make mistakes we need someone to correct them. We let the critical channel do this.)

In the second scheme (Sec. 5.4) both sources and physical channels consult with each other to make decisions. One thing we see is that the first scheme involves fewer parties and less information than the second scheme in making decisions. Yet the second produces better decisions than the first scheme.

The above discussion seems to persuade us to believe that there exists an uncertainty principle in the area of distributed dynamic flow control, that may roughly be stated as follows.

> Let $I \triangleq$ degree of independence of an individual part to make a decision and let $E \triangleq$ effectiveness of the decision. Then for a given network, we conjecture that $I \times E \simeq C$ where E, the effectiveness of the decision, is compared to the optimal decision based on some measure set by the designer of a system (for example, the measure may be the nonlinear equation in Sec. 5.1); and I, the degree of independence in making the decision may be defined, for example, as $I =$ (amount of information needed)$^{-1}$; and C, the constant, may be a function of network topology, routing, size, etc.

This is an interesting and important research area. Unfortunately, we have not yet carried out that investigation. The interested reader is urged to pursue it.

APPENDIX A

Let

$$I_n(\tau,T) \triangleq \int_\tau^T \frac{\alpha(\alpha t)^n e^{-\alpha t}}{n!}\,dt \tag{A.1}$$

and,

$$A_n(\tau,T) \triangleq \int_\tau^T t\,\frac{\alpha(\alpha t)^n e^{-\alpha t}}{n!}\,dt \tag{A.2}$$

Then

$$I_0(\tau,T) = \int_\tau^T \alpha e^{-\alpha t}dt = e^{-\alpha t}\Big|_\tau^T = e^{-\alpha\tau}-e^{-\alpha T} \tag{A.3}$$

$$\begin{aligned} I_n(\tau,T) &= \frac{-(\alpha t)^n e^{-\alpha t}}{n!}\Bigg|_\tau^T + \int_\tau^T \frac{\alpha(\alpha t)^{n-1}e^{-\alpha t}}{(n-1)!}\,dt \\ &= \left[\frac{(\alpha\tau)^n e^{-\alpha\tau}}{n!} - \frac{(\alpha T)^n e^{-\alpha T}}{n!}\right] + I_{n-1}(\tau,T) \\ &\vdots \\ &= \sum_{k=0}^{n}\left[\frac{(\alpha\tau)^k e^{-\alpha\tau}}{k!} - \frac{(\alpha T)^k e^{-\alpha T}}{k!}\right] \end{aligned} \tag{A.4}$$

and

$$\begin{aligned} A_n(\tau,T) &= \frac{n+1}{\alpha}\int_\tau^T \frac{\alpha(\alpha t)^{n+1}e^{-\alpha t}}{(n+1)!}\,dt \\ &= \frac{n+1}{\alpha}\,I_{n+1}(\tau,T) \end{aligned} \tag{A.5}$$

Let

$$A_n[(\tau,T)\,|\,t\in(\tau,T)] \triangleq \int_\tau^T t\,\frac{\left[\dfrac{\alpha(\alpha t)^n e^{-\alpha t}}{n!}\right]}{I_n(\tau,T)}\,dt \tag{A.6}$$

Then

$$A_n[(\tau,T)|t\in(\tau,T)] = \frac{n+1}{\alpha}\,\frac{I_{n+1}(\tau,T)}{I_n(\tau,T)} \tag{A.7}$$

As $\tau \to 0$, from Eqs. (A.4), (A.5) and (A.7), we have

$$I_n(\tau,T) \to 1 - e^{-\alpha T}\sum_{k=0}^{n}\frac{(\alpha T)^k}{k!} \tag{A.8}$$

$$A_n(\tau,T) \to \frac{n+1}{\alpha}\left[1 - e^{-\alpha T}\sum_{k=0}^{n+1}\frac{(\alpha T)^k}{k!}\right] \tag{A.9}$$

and

$$A_n[\tau,T|t\in(\tau,T)] \to \frac{n+1}{\alpha}\,\frac{1 - e^{-\alpha T}\sum_{k=0}^{n+1}\frac{(\alpha T)^k}{k!}}{1 - e^{-\alpha T}\sum_{k=0}^{n}\frac{(\alpha T)^k}{k!}} \tag{A.10}$$

As $T \to \infty$ also, we have

$$I_n(\tau,T) \to 1 \tag{A.11}$$

$$A_n(\tau,T) \to \frac{n+1}{\alpha} \tag{A.12}$$

$$A_n[\tau,T|t\in(\tau,T)] \to A_n(\tau,T) \to \frac{n+1}{\alpha} \tag{A.13}$$

BIBLIOGRAPHY

BARB 81 Barberis, G., "Buffer Sizing of a Packet-Voice Receiver," *IEEE Transactions on Communications*, Vol. COM-29, February 1981, pp. 152-156.

BASK 75 Baskett F., K. Chandy, R. Muntz and F. Palacios. "Open, Closed, and Mixed Networks of Queues with Different Classes of Customers," *Journal of the ACM*, Vol. 22, No. 2, April 1975, pp. 248-260.

BUZE 73 Buzen, J.P. "Computational Algorithms for Closed Queueing Networks with Exponential Servers," *Communications of the ACM*, Vol. 116, No. 9, September 1973, pp. 527-531.

CERF 74 Cerf, V.G. and R.E. Kahn. "A Protocol for Packet Network Interconnection," *IEEE Transactions on Communications*, Vol. COM-22, No. 5, May 1974, pp. 637-648.

DAVI 72 Davies, D.W. "The Control of Congestion in Packet Switching Networks," *IEEE Transactions on Communications*, Vol. COM-20, June 1972, pp. 546-550.

GERL 80 Gerla, M. and L. Kleinrock. "Flow Control — A Comparative Survey," *IEEE Transactions on Communications*, Vol. COM-28, April 1980, pp. 553-574.

GIES 78 Giessler, A., J. Haenle, A. Koenig and E. Pade. "Free Buffer Allocation — An Investigation by Simulation," *Computer Networks*, Vol. 2, No. 3, July 1978, pp. 191-208.

GORD 67 Gordon, W. J. and G. F. Newell. "Closed Queueing Systems with Exponential Servers," *Operations Research*, Vol. 15, No. 2, 1967, pp. 254-265.

GROS 74 Gross, D. and G.M. Harris. *Fundamentals of Queueing Theory*, John Wiley and Sons, Inc., New York, 1974.

HOWA 60 Howard, R. *Dynamic Programming and Markov Processes*, M.I.T. Press, Cambridge, Massachusetts, 1960.

IBM 75 IBM Corporation, *Systems Network Architecture General Information*, GA27-3102-0, January 1975.

IRLA 78 Irland, M. "Buffer Management in a Packet Switch," *IEEE Transactions on Communications*, Vol. COM-26, March 1978, pp. 328-337.

JACK 63 Jackson, J.R. "Jobshop-Like Queueing Systems," *Management Science*, Vol. 10, 1963, pp. 131-142.

KAHN 75 Kahn, R.E. "The Organization of Computer Resources into a Packet Radio Network," *AFIPS Conference Proceedings*, National Computer Conference, Anaheim, California, May 1975, Vol. 44, pp. 177-186.

KAMO 76 Kamoun, F. "Design Considerations for Large Computer Communication Networks," Computer Science Department, School of Engineering and Applied Science, University of California, Los Angeles, UCLA-ENG-7642, April 1976. (Also published as Ph.D. dissertation.)

KAMO 81 Kamoun, F., L. Kleinrock and R. Muntz, "Queueing Analysis of the Ordering Issue in a Distributed Database Concurrency Control Mechanism," *to appear in* Proceedings of Second International Conference on *Distributed Computing Systems*, 1981.

KERM 80 Kermani, P. and L. Kleinrock, "Dynamic Flow Control in Store and Forward Computer Networks," *IEEE Transactions on Communications*, Vol. COM-28, February 1980, pp. 263-271.

KING 66 Kingman, J.F.C. *On the Algebra of Queues*, Methuen and Company, Ltd., London, 1966.

KLEI 75 Kleinrock, L. *Queueing Systems, Vol. I: Theory*, Wiley-Interscience, New York 1975.

KLEI 76 Kleinrock, L. *Queueing Systems, Vol. II: Computer Applications*, Wiley-Interscience, New York, 1976.

KLEI 78 Kleinrock, L. "On Flow Control in Computer Networks," *Conference Record, International Conference on Communications*, Toronto, June 1978, pp. 27.2.1-27.2.5.

KLEI 79 Kleinrock, L. "Power and Deterministic Rules of Thumb for Probabilistic Problems in Computer Communications," *Conference Record, International Conference on Communications*, Boston, Massachusetts, June 1979, pp. 43.1.1 to 43.1.10.

KLEI 80a Kleinrock, L. and P. Kermani. "Static Flow Control in Store-and-Forward Computer Networks," *IEEE Transactions on Communications*, Vol. COM-28, No. 2, February 1980, pp. 271-286.

KLEI 80b Kleinrock, L. and C. W. Tseng, "Flow Control Based on Limiting Permit Generation Rates," *Proceedings of the Fifth International Conference on Computer Communication*, Atlanta, Georgia, October 1980, pp. 785-790.

LAM 77 Lam, S. and M. Reiser, "Congestion Control of Store-and-Forward Networks by Input Buffer Limits," *Conference Record, National Telecommunications Conference*, Los Angeles, December 1977, pp. 12.1.

MCQU 72 McQuillan, J.M., W.R. Crowther, B.P. Cossell, D.C. Walden and F.E. Heart. "Improvements in the Design and Performance of the ARPA Network," *AFIPS Conference Proceedings*, Vol. 41, Fall Joint Computer Conference, Anaheim, California, December 1972, pp. 741-754.

OPDE 74 Opderbeck, H. and L. Kleinrock. "The Influence of Control Procedures on the Performance of Packet-Switched Networks," *Proceedings of the National Telecommunications Conference*, San Diego, California, December 1974, pp. 810-817.

POUZ 76 Pouzin, L. "Flow Control in Data Networks-Methods and Tools," *International Conference on Computer Communication Proceedings*, Toronto, August 1976, pp. 467-474.

PRIC 73 Price, W.L. "Simulation of Packet-Switching Networks Controlled on Isarithmic Principles," *Proceedings of the Third IEEE Symposium on Data Networks Analysis and Designs*, St. Petersburg, Florida, November 13-15, 1973, pp. 44-49.

PRIC 77 Price, W.L. "Data Network Simulation Experiments at the National Physical Laboratory 1968-1976," *Computer Networks*, Vol. 1, No. 4, May 1977, pp. 199-210.

RAUB 76 Raubold E. and J. Haenle. "A Method of Deadlock-Free Resource Allocation and Flow Control in Packet Networks," *Proceedings of the Third International Conference on Computer Communication*, Toronto, August 1976, pp. 483-487.

RAWS 78 Rawson, E.G. "Fibernet: Multimode Optical Fibers for Local Computer Networks," *IEEE Transactions on Communications*, Vol. COM-26, July 1978, pp. 983-990.

REIS 75 Reiser, M. and H. Kobayashi. "Queueing Networks with Multiple Closed Chains: Theory and Computational Algorithms," *IBM Journal of Research and Development*, Vol. 19, May 1975, pp. 283-294.

REIS 76 Reiser, M. and H. Kobayashi. "On the Convolution Algorithm for Separable Queueing Networks," *Proceedings of the International Symposium on Computer Performance Modeling, Measurement and Evolution*, Cambridge, Massachusetts, March 1976, pp. 109-117.

REIS 77 Reiser, M., "Numerical Methods in Separable Queueing Networks," *Studies in the Management Science*, Vol. 7, 1977, pp. 113,142.

REIS 78 Reiser, M., "Mean Value Analysis of Queueing Networks, a New Look at an Old Problem," IBM, Yorktown Heights, New York, Research Report RC7228 (#31125) July 1978.

RUDI 76 Rudin, H. "Flow Control: Introduction," *International Conference on Computer Communication Proceedings*, Toronto, August 1976, pp. 463-466.

SUNS 75 Sunshine, C.A. "Interprocess Communication Protocols for Computer Networks," Stanford Electronics Laboratories, Stanford University, Technical Report #105, December 1975.

WONG 77 Wong, J. and M. Unsoy, "Analysis of Flow Control in Switched Data Networks," *Proceedings, IFIP Congress '77*, Toronto, August 1977.

WONG 78 J. Wong, "Distribution of End-to-End Delay in Message-Switched Networks," *Computer Networks*, Vol. 2, February 1978, pp. 44-49.


DAT
FILM