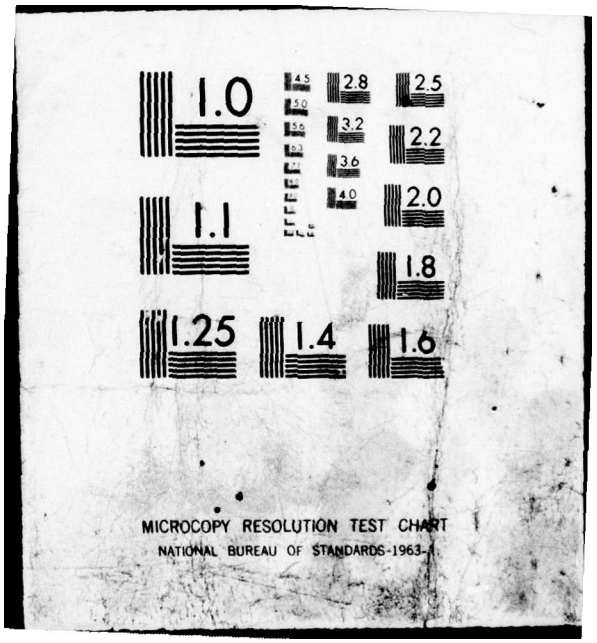
AD-A077 404 CALIFORNIA UNIV LOS ANGELES DEPT OF COMPUTER SCIENCE F/G 17/2.1
ADVANCED TELEPROCESSING SYSTEMS.(U)
JUN 78 L KLEINROCK MDA903-77-C-0272
UNCLASSIFIED NL

1 OF 4
AD
A077404




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD A077404

LEVEL II

ADVANCED TELEPROCESSING SYSTEMS

SEMIANNUAL TECHNICAL REPORT

JUNE 30, 1978

1 Aug 77 - 30 Jun 78

MDA903-77-C-0272

Principal Investigator: Leonard Kleinrock

30 Jun 78

353

DDC FILE COPY

DDC
RECEIVED
NOV 29 1979
A

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

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

79 11 27 022

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Advanced Teleprocessing Systems		5. TYPE OF REPORT & PERIOD COVERED Semiannual Technical Report - 8/1/77-6/30/78 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Dr. Leonard Kleinrock		8. CONTRACT OR GRANT NUMBER(s) MDA903-77-C-0272
9. PERFORMING ORGANIZATION NAME AND ADDRESS School of Engineering and Applied Science University of California Los Angeles, California 90024		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Advanced Research Projects Agency (ARPA) 1400 Wilson Boulevard Arlington, Virginia 22209		12. REPORT DATE 6/30/78 13. NUMBER OF PAGES 350
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ADVANCED TELEPROCESSING SYSTEMS

Sponsored by

ADVANCED RESEARCH PROJECTS AGENCY

SEMIANNUAL TECHNICAL REPORT

June 30, 1978

ARPA Contract MDA903-77-C-0272

Principal Investigator: Leonard Kleinrock

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

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the United States Government.

Accession For
NTIS GRA&I
DDC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or special
A

ADVANCED TELEPROCESSING SYSTEMS

Advanced Research Projects Agency
Semiannual Technical Report

June 30, 1978

INTRODUCTION

This "Semi-Annual" Technical Report covers research carried out by the Advanced Teleprocessing Systems group at UCLA under ARPA Contract MDA903-77-C-0272, covering the period August 1, 1977, through June 30, 1978. The research herein conducted is basically a continuation of our former ARPA Contract Number DAHC-15-73-C-0368, which was also devoted to Advanced Teleprocessing Systems. Under this new contract, we have two designated tasks as follows:

Task 1: Radio Packet Switching Systems -- Advanced studies regarding the fundamental analytic and design considerations for random multiple-access radio packet switching systems. We will investigate the basic performance measures including capacity, stability, control, routing, and the tradeoffs among these quantities for ground and satellite packet radio systems; and

Task 2: Advanced Research in Distributed Communications -- Advanced studies in internetting, flow control, distributed access, fundamental capacity definitions and contours, and investigation of the underlying cost-performance behavior.

We have made significant progress in the two named tasks. In the following paragraphs, we describe the progress and give pointers to those references which represent the published work resulting from this supported research.

Following this short summary is a list of publications produced as a result of the recent research on this contract covering the period being reported. This list contains only those articles and reports which, in fact, did appear in print during this period. Papers which have been submitted (of which there are many) are not listed here, but will be listed in future reports as they appear in the published literature. We devote the main body of this report to the detailed presentation of one aspect of this overall research, and we simply mention the other areas briefly in this summary.

The research reported in the main body of this document discusses various switching and flow control techniques as applied to packet communications networks. Described therein is a new switching technique called "virtual cut-through". It and its performance is analyzed and graphed. Moreover, a comparison is made between this new switching scheme and message (or packet) switching and circuit switching. Furthermore, a window-type flow control procedure is defined and evaluated. We study both static and dynamic flow control procedures and describe an optimal control policy to dynamically select window size. A network algebra for interconnected networks, both in series and in parallel, is developed and evaluated. Further, buffer allocation schemes in which a number of channels converge at a single node which must share a finite buffer in that node are analyzed and evaluated. The particular details in the following report represent the Ph.D. dissertation of Parviz Kermani (Chairman, Leonard Kleinrock) and the work is entitled, "Switching and Flow Control Techniques in Computer Communication Networks."

Turning now to the list of references, we comment that reference #1 gives a global overview of distributed multi-access computer communication systems, placing the various access schemes in a commun framework and identifying the cost which nature will extract for distributed communications. New measures of performance and a reduced set of parameters are identified in this work which have turned out to be extremely useful in evaluating new and/or old access schemes. Reference #2 represents a significant piece of work in the field of stream packet communications (e.g., packetized voice). Reference #3 gives a very effective control method for maintaining slotted ALOHA as an access scheme in a stable mode; simulation experiments are extremely encouraging regarding this very simple control method. Reference #4 examines stability and dynamic control in ground radio packet switching and gives a mathematical evaluation of the various stable and unstable modes and their basic performance. Reference #5 presents some measurement results of the ARPA packet satellite experiment and shows the way in which various access schemes compare to each other, both from a theoretical, simulation and experimentally measured point-of-view. Reference #6 presents a study of ground radio access to a station through a multi-hop tree-type topology; we find that very few buffers suffice at each node due to the half-duplex nature of the transceivers (repeaters). Reference #7 is the main body of the support. Reference #8 refers to a significant work in the evaluation of hierarchically structured wire and broadcast networks; here one observes the gains to be had through hierarchical organization and the various modes in which one can interconnect communicating terminals to increase their capacity at fixed-delay or to minimize delay at a fixed network investment. Reference #9 describes a new

optimal access scheme for ground radio packet switching which promises to be extremely useful for multi-hop ground radio networks. Reference #10 gives an overview of global flow control in a network environment. Here, a new definition of the cost of a flow control scheme as well as some fundamental optimization methods is presented. Reference #11 examines the effect of various acknowledgment mechanisms in packet switched radio channels so that one can sensibly select from among the various schemes available today. The research reported in these published articles is continuing and we are currently investigating new areas as well.

The main report on switching and flow control in packet networks is given following this list of publications.

LIST OF PUBLICATIONS

1. Kleinrock, L., "Performance of Distributed Multi-Access Computer-Communication Systems, Information Processing 77, Proceedings of IFIP Congress, Toronto, Canada, August 1977, North Holland, Amsterdam, 1977, pp. 547-552.

2. Naylor, W.E., "Stream Traffic Communication in Packet Switched Networks," Ph.D. Dissertation, School of Engineering and Applied Science, Computer Science Department, University of California, Los Angeles, September 1977.

3. Gerla, M. and L. Kleinrock, "Closed Loop Stability Controls for S-ALOHA Satellite Communications," Proceedings of the Fifth Data Communications Symposium, Snowbird, Utah, September 1977, pp. 2-10 to 2-19.

4. Tobagi, F. and L. Kleinrock, "Packet Switching in Radio Channels: Part IV -- Stability Considerations and Dynamic Control in Carrier Sense Multiple Access," IEEE Transactions on Communications, Vol. COM-25, October 1977, pp. 1103-1119.

5. Gerla, M., L. Nelson and L. Kleinrock, "Packet Satellite Multiple Access: Models and Measurements, Proceedings of the National Telecommunications Conference, Santa Monica, California, December 1977, pp. 12.2-1 to 12.2-8.

6. Tobagi, F., "Performance Analysis of Packet Radio Communication Systems," Proceedings of the National Telecommunications Conference, Santa Monica, California, December 1977, pp. 12.6-1 to 12.6-7.

7. Kermani, P., "Switching and Flow Control Techniques in Computer Communication Networks," Ph.D. Dissertation, School of Engineering and Applied Science, Computer Science Department, University of California, Los Angeles, December 1977.

8. Akavia, G., "Hierarchical Organization of Distributed Packet-Switching Communication Systems," Ph.D. Dissertation, School of Engineering and Applied Science, Computer Science Department, University of California, Los Angeles, March 1978.

9. Kleinrock, L. and Y. Yemini, "An Optimal Adaptive Scheme for Multiple Access Broadcast Communication," Proceedings of the International Conference on Communications, Vol. I, Toronto, Ontario, June 1978, pp. 7.2.1 to 7.2.5.

10. Kleinrock, L., "On Flow Control," Proceedings of the International Conference on Communications, Vol. II, Toronto, Ontario, June 1978, pp. 27.2.1 to 27.2.5.

11. Tobagi, F. and L. Kleinrock, "The Effect of Acknowledgment Traffic on the Capacity of Packet-Switched Radio Channels," IEEE Transactions on Communications, Vol. COM-26, No. 6, June 1978, pp. 815-826.

JUNE 1978

Switching and Flow Control Techniques

in

Computer Communication Networks

by

Parviz Kermani

This Research,
Conducted Under the Chairmanship of
Professor Leonard Kleinrock
was Sponsored by the
Advanced Research Projects Agency
Department of Defense
Contract No. MDA 903-77-C-0272

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

ABSTRACT

The objective of this research is to develop analytic models for advanced performance evaluation of computer-based communication networks. The emphasis is on flow control mechanisms and on models for switching techniques.

A new switching technique, virtual cut-through, is proposed and its performance is analyzed. Through the comparative study of different switching schemes (virtual cut-through, message (or packet) switching, and circuit switching), we establish certain ground rules which aid the network designer in deciding which switching method to use in a data communication network.

Flow control procedures for computer networks are investigated and analytic models for the "window" mechanism are developed. We study both "static" and "dynamic" flow control. In static flow control, parameters of the system are not adjusted to the stochastic load fluctuations. We study the optimal window size for static flow control which results in maximum throughput. In dynamic flow control, parameters of the system are dynamically adjusted to the resource availability of the network. Based on Markov decision theory, an optimal policy to dynamically decide on the window size is formulated. Because an exact solution to the problem is laborious, an effective heuristic solution to the problem is presented.

Some problems in interconnection of networks are also studied. Based on our study of static flow control, a network algebra for series

and/or parallel-connected networks is developed. The problem of resource allocation in parallel-connected networks is investigated. Finally, a number of buffer allocation schemes in nodes where a number of networks merge together is analyzed. These allocation schemes serve as a further tool for the control of traffic in networks.

CONTENTS

			Page
ACKNOWLEDGEMENTS			iii
ABSTRACT			v
LIST OF FIGURES			xi
LIST OF TABLES			xv
CHAPTER 1		INTRODUCTION	1
	1.1	Switching Techniques in Computer Networks	4
	1.2	Routing and Flow Control in Store-and-Forward Computer Networks	7
	1.3	Scope of This Research and Outline of the Dissertation	9
CHAPTER 2		"VIRTUAL CUT-THROUGH" - A NEW SWITCHING TECHNIQUE	14
	2.1	Assumptions	19
	2.2	Analysis of the Number of Cut-Throughs	20
	2.3	Delay Analysis	23
		2.3.1 Delay Analysis for a Network with Noiseless Channels	25
		2.3.2 Delay Analysis for a Network with Noisy Channels	27
	2.4	Traffic Gain	34
	2.5	Performance Curves and Simulation Results	37
	2.6	Storage Requirement	54
		2.6.1 Buffer Allocation in Message Switching	54
		2.6.2 Buffer Allocation in Cut-Through Switching	57

CONTENTS (continued)

		Page
2.6.3	Storage Requirement for Message Switching	58
2.6.4	Storage Requirement for Cut-Through Switching	59
2.7	A Design Problem	67
2.8	Conclusion	74
CHAPTER 3	A TRADEOFF STUDY OF SWITCHING SYSTEMS	76
3.1	History and Related Works	77
3.2	A Delay Model for Circuit Switching	79
3.3	An Exact Analysis	82
3.4	Analysis of a Path Model for the Circuit Switching System	91
3.5	Discussion of Results	96
3.6	Optimal Number of Channels in Circuit Switching	115
3.7	Conclusion	120
CHAPTER 4	STATIC FLOW CONTROL IN STORE-AND-FORWARD COMPUTER NETWORKS	123
4.1	The Model	126
4.2	A Fluid Approximation	130
4.3	A Simplified Stochastic Analysis	133
4.4	A Generalized Stochastic Model	142
4.4.1	The Acknowledgement and Timeout Strategy.	142
4.4.2	The (Sub)Network Delay Distribution	146
4.4.3	The Destination Node Structure	148
4.4.4	Analysis	148
4.4.5	Numerical Results	155
4.5	Conclusion	174

CONTENTS (continued)

			Page
CHAPTER 5	DYNAMIC FLOW CONTROL IN STORE-AND-FORWARD COMPUTER NETWORKS		177
5.1	The Model		179
	5.1.1	The State Space	181
	5.1.2	The Action Space	185
	5.1.3	The Policy Space	186
	5.1.4	The State Transition Probabilities	186
	5.1.5	The Performance Criteria and the Reward Function	189
	5.1.6	Statement of the Problem	191
5.2	Solution to the Problem - The Look-Ahead Policy		193
	5.2.1	Calculation of the Expected Buffer Occupancy	197
5.3	Implementation of the Look-Ahead Policy - The Decision Table		199
5.4	1-Cycle-Delay and Continuous Time Decision Processes		202
	5.4.1	A Procedure to Find the Set of Input Rates	205
5.5	Numerical Results		208
5.6	Conclusion		217
CHAPTER 6	INTERCONNECTION OF NETWORKS		219
6.1	A Network Algebra		219
	6.1.1	Series Connection of Networks	224
		6.1.1.1 Equal Channel Transmission Rates	224
		6.1.1.2 Non Equal Channel Transmission Rates	232

CONTENTS (continued)

		Page
6.1.2	Parallel Connection of Networks	232
6.2	Optimal Allocation of Resources	234
6.2.1	Optimal Allocation of Window Buffers and Channel Capacities	237
6.2.2	Optimal Window Buffer Allocation	245
6.2.3	Optimal Capacity Allocation	246
6.2.4	Delay Considerations	250
6.3	Buffer Allocation Schemes in a Destination Node .	261
6.4	Conclusion	272
CHAPTER 7	CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH	277
APPENDIX A	LIST OF OFTEN-USED SYMBOLS	280
APPENDIX B	THEOREMS AND PROOFS FOR CHAPTER 2	283
B.1	Multiple Exponential Channels	283
B.2	Theorem 2.3 and Its Proof	284
B.3	Theorem 2.4 and Its Proof	287
APPENDIX C	SOME RESULTS FROM MARKOV DECISION THEORY	289
APPENDIX D	ANALYSIS OF BUFFER ALLOCATION SCHEMES IN A MULTIPLEXING NODE	296
D.1	Complete Sharing (CS)	297
D.2	Complete Partitioning (CP)	302
D.3	Sharing with Maximum Queue Length (SMXQ)	309
D.4	Sharing with Minimum Allocation (SMA)	314
D.5	Sharing with Minimum Allocation and Maximum Queue Length (SMAXQ)	318
BIBLIOGRAPHY	. . . . .	321

LIST OF FIGURES

		Page
1.1	A Computer-Communication Network	3
2.1	Operations of Different Switching Systems	16
2.2	Simplified Model of A Store-and-Forward Node	28
2.3	Example of Retransmission in the Cut-Through System . . .	30
2.4	Sketch of Traffic Gain in the Cut-Through System	35
2.5	Average Number of Cut-Throughs	38
2.6	The Simulated Tandem Queues	39
2.7-a	Delay Vs. Throughput (n_h = 3, P_e = 0.0, α = 0.0)	40
2.7-b	Delay Vs. Throughput (n_h = 3, P_e = 0.0, α = 0.1)	41
2.7-c	Delay Vs. Throughput (n_h = 3, P_e = 0.001, α = 0.0)	42
2.7-d	Delay Vs. Throughput (n_h = 3, P_e = 0.001, α = 0.1)	43
2.7-e	Delay Vs. Throughput (n_h = 3, P_e = 0.05, α = 0.0)	44
2.7-f	Delay Vs. Throughput (n_h = 3, P_e = 0.05, α = 0.1)	45
2.7-g	Delay Vs. Throughput (n_h = 3, P_e = 0.45, α = 0.1)	47
2.8(a-d)	Ratio of the Delay in the Two Switching Systems.	48
2.9	Some Simulation Results	53
2.10	Channel Utilization in the Two Switching Systems for Equal Delays	55
2.11	Traffic Gain	56
2.12	Storage Occupancy in the Cut-Through System	60
2.13	Comparison of Storage Requirements in the Cut-Through System and the Message Switching System	66
2.14-a	Delay Vs. Throughput for the Cut-Through System	72
2.14-b	Probability of No Blocking as a Function of Utilization .	73
3.1	Structure of a Node	81
3.2	A Communication Network	81
3.3	Abstract Structure of a Node	83
3.4	The Simulated Tandem Queues	89

LIST OF FIGURES (continued)

		Page
3.5	Accuracy of the Analytic Results	90
3.6	Tandem Queue Model of a Communication Path	92
3.7(a-c)	Comparison of Switching Systems	97
3.8	Comparison of Switching Systems	102
3.9(a-b)	Comparison of Switching Systems	104
3.10	Saturation Point of Circuit Switching	107
3.11	MS Vs. CS (n_h = 2)	108
3.12	MS Vs. CS (n_h = 4)	109
3.13	MS Vs. CS (n_h = 8)	110
3.14	CTS Vs. CS (n_h = 8)	112
3.15(a-c)	Dynamic Channel Splitting - Optimal Number of Channels for CS	117
3.16	Static Channel Splitting - Optimal Number of Channels for CS	121
4.1	Structure of a Network	127
4.2	Transmission Interval in a Network When Window size is 1.	131
4.3	Diagram of Network Throughput as a Function of Window Size (*Deterministic Model*)	131
4.4	A Communication Path	131
4.5	Sketch of Throughput and Delay as a Function of Window Size	136
4.6	Sketch of Throughput, Delay and Power of a Network	139
4.7	Structure of Network Components	144
4.8	Diagram of Density and Distribution Functions of Acknowledgement Delay	147
4.9	Dependency of Network Throughput on Timeout	157
4.10-a	End-to-End Delay as a Function of Timeout	160
4.10-b	Network Delay as a Function of Timeout	161
4.10-c	Destination Node Delay as a Function of Timeout	162
4.11	Network Throughput Vs. Window Size	164
4.12	End-to-End Delay Vs. Window Size	166
4.13	Throughput Vs. Network Traffic	167
4.14	Throughput Delay Tradeoff	169
4.15	Throughput-Delay for Different Destination Buffer Sizes	170

LIST OF FIGURES (continued)

		Page
4.16	Contours of Constant Throughputs and End-to-End Delays	172
5.1	Evolution in Time of Input Rate to Network	182
5.2	State Vector at Two Consecutive Time Cycles	188
5.3	Example of a Decision Table	201
5.4	State-Transition-Rate Diagram for the Destination Buffer	206
5.5	Throughput-Delay Performance of Continuous Time Model for Different Buffer Sizes	211
5.6	Throughput-Delay Performance of Look-Ahead Policy	213
5.7	Throughput-Delay Performance of Look-Ahead Policy for Varying Round-Trip Delay	216
6.1	Short Hand Notation	223
6.2-a	Traffic Handling Capacity Vs. Window Buffer Allocation	242
6.2-b	Traffic Handling Capacity Vs. Capacity Allocation	243
6.3	Diagram of Maximum Throughput and Delay	252
6.4-a	Traffic Weighted Delay Vs. Window Buffer Allocation	253
6.4-b	Traffic Weighted Delay Vs. Capacity Allocation	255
6.5	Power of Two Parallel Paths	259
6.6	Storage Allocation and Sharing Schemes	264
6.7	Effect of Allocated Storage in the CP Scheme	268
6.8	Comparison of Different Schemes When One of the Input Rates Increases	270
6.9	Comparison of Schemes: Probability of Being Absent from the System	271
6.10	Comparison of Different Schemes: Throughput	273
6.11	Comparison of Different Schemes: Delay	274
C.1	The Iteration Cycle	295

LIST OF TABLES

		Page
3.1	Comparison of Switching Techniques	113
5.1	Example of Optimal Window Selection for a Continuous Time Model	209
6.1	A Network Algebra	235

CHAPTER 1

INTRODUCTION

In the recent past, we have witnessed an explosion in the analysis, design and implementation of computer communication networks. These efforts reflect the desire to exploit and share the computational capacity of geographically separated computer systems, which results in a considerable economy of scale [ROBE 70].

The demand for remote processing is growing at an enormous rate [KLEI 76B]. For example, it is projected that within less than five years in Europe there will be approximately five million terminals in use and the data traffic will grow up to 10^{12} bits/day [PETE 73].

Computer networks have been in existence since the mid 1950's. One of the earliest computer networks was the SAGE defense network in the 1950's [EVER 57]. The American Airlines SABRE reservation system came shortly thereafter [EVAN 67]. Early computer networks were either designed for special purposes or owned by private corporations. It was only after the conception of the ARPANET [CARR 70], [FRAN 70], [HEAR 70], [KLEI 70], [ROBE 70], that the surge in interest toward computer communications and remote data processing began.

Two general categories of networks can be distinguished: the remote-access networks and the value-added networks [KIMB 75]. Remote-access networks are designed to provide access between a user and a given remote computer, either for terminal access or for remote job entry. Examples of this type of network are the GE information service MARK III

[MAUC 74], [SCHW 77], and TYMNET [TYME 71], [HARC 75]. (This latter network also falls into the second category; see below.) Value-added networks support communications directly between computers to provide more computer power and to distribute the computational load more efficiently; early examples of experimental networks are the ARPANET [references cited above], the British NPL Network [DAVI 73] and the French CYCLADES [POUZ 75]. Examples of more recent networks include Telenet, Datapac and Tranpac. By providing a suitable interface between host computers and user's terminals, value-added networks can easily support remote-access capability. In this dissertation we are mainly concerned with the second category and refer to value-added networks simply as computer networks.

Figure 1.1 shows a simplified computer network. It consists of a number of disjoint "local" networks and a "communication sub-network." A local network consists of one or more computers (HOSTs), each one connected to the communication sub-network through one node. Users' terminals are ususally connected to HOSTs and are part of the local network. (Users' terminals may also be directly connected to the communication sub-net via a TIP.)

The communication sub-net is a network of facilities (hardware and software) to deliver messages from one HOST to another (or from one user to a HOST). The communication sub-net consists of a number of communcation processors or switching centers, connected to each other through communication media. Henceforth we will refer to a switching center simply as a node, and to the communication sub-network as a communication net.

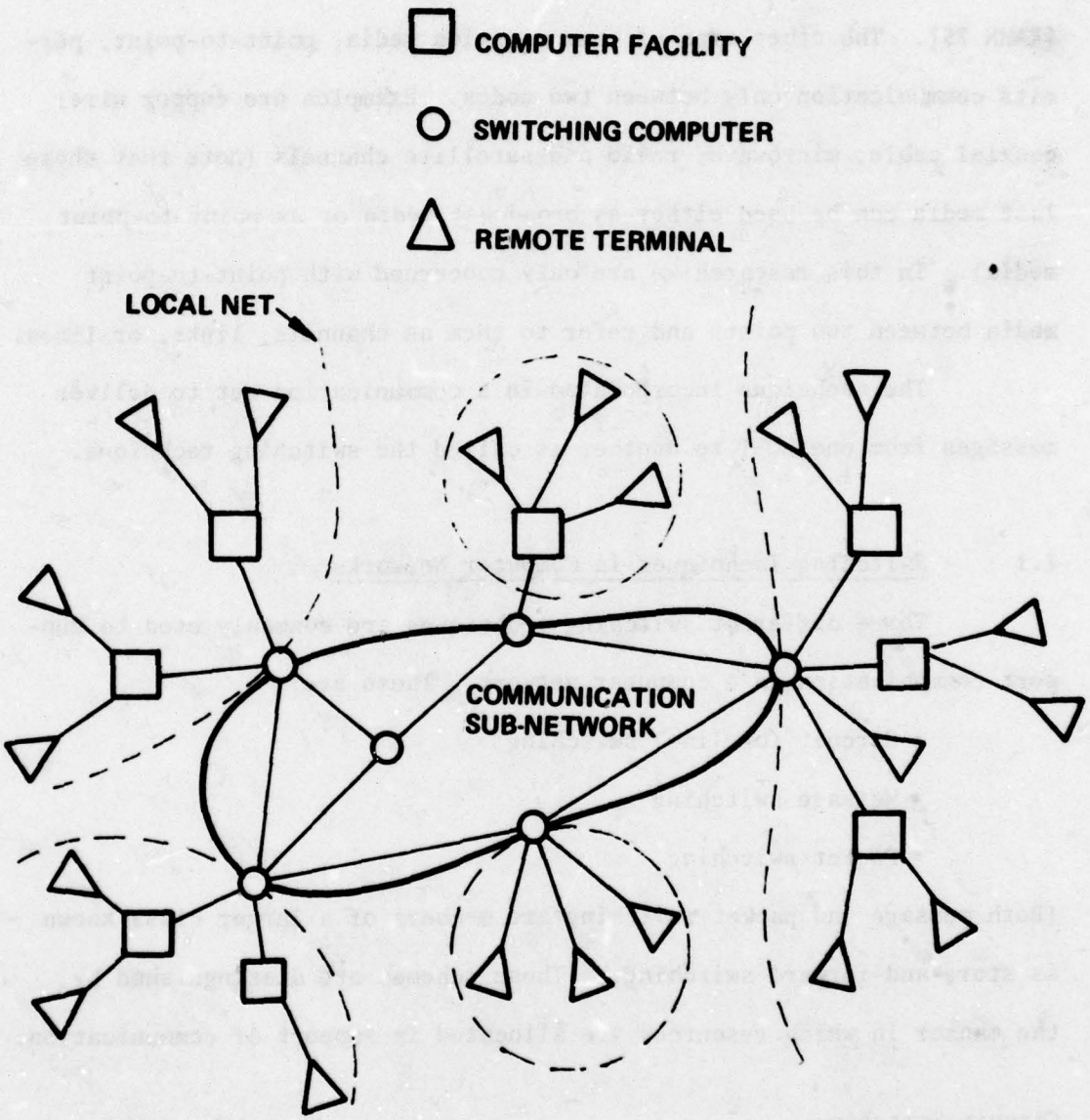


Figure 1.1. A Computer-Communication Network.

Two alternative media can be used to connect the nodes to each other and to host computers: broadcast and point-to-point. Broadcast media permit simultaneous transmission (or reception) to (or from) a number of nodes [KIMB 75]. Two examples are satellite communication [ABRA 73], [LAM 74], [LAM 75C], and packet-switched radio [ABRA 70], [KAHN 75]. The other type of communication media, point-to-point, permits communication only between two nodes. Examples are copper wire; coaxial cable; microwave, radio and satellite channels (note that these last media can be used either as broadcast media or as point-to-point media). In this research we are only concerned with point-to-point media between two points and refer to them as channels, links, or lines.

The technique incorporated in a communication net to deliver messages from one HOST to another is called the switching technique.

1.1 Switching Techniques in Computer Networks

Three different switching techniques are commonly used to support communication in a computer network. These are:

- Circuit (or line) switching
- Message switching
- Packet switching

(Both message and packet switching are members of a larger class known as store-and-forward switching.) These schemes are distinguished by the manner in which resources are allocated in support of communication.

Circuit Switching

With circuit switching, a complete path of communication must be set up between two parties before the communication begins. Path

set-up time plays a critical role in the performance of this switching. Once a path is set up, it is tied up from the time that the first bit of the first message until the last bit of the last message is transmitted between the two parties (this includes idle times as well). Circuit switching is a common method for telephone switching [SYSK 60]; when applied to data communication networks, it suffers from two major drawbacks: slow path set-up and low channel utilization (more about this in Chapters 2 and 3). This type of switching has been the primary technology utilized in such computer networks as CYBERNET [LUTH 72], INFONET and DATRAN [CHEN 75A,B], [RAND 75].

Message Switching

In this technique a message is defined to be the logical unit of information from the viewpoint of transmission. Each message carries with it information regarding its destination. This is done by attaching addressing bits (usually a part of the header) to each message. In this switching technique only one channel is used at a time for a given transmission. A message first travels from its source node to the next node in its path, and when the entire message is received at this node, then the next step in its journey is started by transmitting the message on a selected channel towards the destination; if the selected channel is busy, the message waits in a queue, and finally, when the channel becomes free, transmission starts. Thus the message "hops" from node to node through the network, using one channel at a time, possibly queueing at busy channels as it is successively "store-and-forwarded" through the network [KLEI 64], [KLEI 76B]. Two examples of networks

which use this type of switching are the AUTODIN [PAOL 75] and TYMNET [TYME 71], [HARC 75].

Packet Switching

This switching technique is basically the same as message switching, except that messages are subdivided into packets with a maximum length, prefaced with suitable address information. Through decomposition of messages into packets, many packets of the same message may be in transmission simultaneously, and a message can use a number of links on a path concurrently. Due to more complete resource sharing (here the resource is the communication link), a higher channel utilization and lower network delay is possible [BARA 64], [DAVI 68], [ROBE 67], [KLEI 76B]. The ARPANET is an example of a packet-switched network. Other examples include the British NPL, the French CYCLADES, and also the first large-scale commercial packet-switched network, the TELENET.

Analysis of the tradeoffs which must be considered in evaluating the three switching techniques, and the decision as to which one to use is a difficult task. Some tradeoff studies have been conducted [PORT 71], [CLOS 72A], [CLOS 72B], [CLOW 73], [ITOH 73], [MIYA 75], [ROSN 76], [KUMM 76A] and [KUMM 76B], but the techniques are not quite satisfactory and the agreement on which type of switching is superior is not universal. One thing is clear; namely, if the message to be transmitted is very long, circuit switching is a good choice. On the other hand, if the messages are relatively short, it pays to use some sort of store-and-forward facility. The fact that in practice there is

a broad range of message lengths and message arrival rates motivates the consideration of a hybrid mixture of store-and-forward and circuit switching systems, or the concept of integrated networks incorporating both message (or packet)-switched and circuit-switched capability. A few systems based on integration of switching systems are either in existence or have been proposed [ZAFI 76], [PAOL 75], [GERL 78]; however, no system based on a hybrid switching technique has been yet developed.

1.2 Routing and Flow Control in Store-and-Forward Computer Networks

In the previous section we considered the switching alternatives in computer networks. In many (but not all) cases store-and-forward switching manifests a significant advantage over circuit switching, since it permits simultaneous support of multi-modal traffic. The success of the first packet switched network, the ARPANET, has generated great enthusiasm toward adopting this type of switching.

In the design of a store-and-forward network one faces a multitude of problems, among which are: topological design; capacity allocation; routing procedure; and flow control procedure. Several different formulations of the topological design and capacity allocation problem can be found in the literature [KLEI 64], [FRAN 70], [FRAN 72], [GERL 73A], [KLEI 76B], [SCHW 77], for which solutions have been proposed. In a store-and-forward network, messages (or packets) travel from their sources to their destinations "hopping" from one node to another. Once a packet (or message) enters an intermediate node, the next node to which it should be sent is selected according to a well-

defined decision rule referred to as the routing procedure. There are a variety of approaches to establishing routes for messages (or packets) in a store-and-forward network [SCHW 77]: the routes can be established centrally (as in TYMNET and SITA [CHRE 73]), or locally (as in the ARPANET); they can be fixed and deterministic, chosen on the basis of average traffic statistics (as in SITA); they can be fixed during the entire message interval, although varying with demand (as in TYMNET, in which a central supervisory program establishes a route - fixed throughout the conversation - each time a user connects into the system); or they can be adaptive, following a stochastic control strategy (as in the ARPANET). Several classification schemes have been devised to characterize routing policy [KLEI 64], [FULT 72], [GERL 73B], [MCCO 75], [MCQU 74], [KAMO 76B]. The various routing methods, whether fixed or adaptive, local or central, deterministic or stochastic, attempt to direct messages from source to destination such that:

i. Delay becomes minimal and

ii. Traffic is so distributed in the network that congestion is avoided.

A computer network is a collection of resources (communication links, etc.) to be used by competing users (messages). This collection of resources has a finite capacity which causes conflict to occur between the users of the system. These conflicts can result in a degradation of performance of the system to the point that throughput goes to zero; a typical behavior of "contention" systems [AGNE 74]. Even with the best routing procedure, if the network becomes overloaded, this throughput degradation becomes inevitable. Networks cannot afford

to accept all the traffic that is offered to them without some control. There must be rules which govern the acceptance of traffic from outside and coordinate the flow inside the network. These rules are commonly known as flow control procedures.

The set of rules that control the flow of messages (once they are admitted) inside the network are usually referred to as "local control", as opposed to "global control", which is the set of control procedures which supervise the admission of messages to the network. Most commonly used global flow controls can be divided into two categories: end-to-end flow control, used in the ARPANET [KAHN 7]], [CERF 74]; and isarithmic flow control, as considered in the NPL network [DAVI 71], [DAVI 73], [PRIC 73]. End-to-end flow control is usually described in terms of "window" mechanism [CERF 74], [BELS 75], where the number of "unacknowledged" messages (or packets) between a source and destination is limited to the window size. End-to-end flow control is accomplished by inter-process communication protocols and any attempt to quantitatively study the former should start with the development of an analytic model for the latter. Except for simulation studies [PRIC 73], [GIES 76] and [LELA 76], in most analytic work done so far the interaction of different flow control tools has not been studied [PENN 74], [PENN 75], [SUNS 75], [CHAT 76], [CHOU 76] and [FAYO 77].

1.3 Scope of this Research and Outline of the Dissertation

In the previous sections we examined some issues in computer-based communication networks and (intentionally) elaborated in more detail on the switching techniques and flow control mechanisms in such

networks, the two issues to which this dissertation is addressed.

In Section 1.1 we found that a hybrid mixture of circuit switching and store-and-forward switching may be a good alternative, as it can handle a broad range of message lengths and arrival rates. We also pointed out that very little satisfactory quantitative work has been done regarding the tradeoff study of switching systems. In this dissertation we introduce and analyze a new hybrid switching method and also carry out a tradeoff study of switching systems.

It has been realized that the behavior of a computer network is similar to a "contention" system in that as traffic increases, throughput of the system degrades (possibly to zero) and its delay escalates [RUDI 76]. Very few analytic works can demonstrate this phenomenon. In our study of flow control, we develop analytic models for flow control based on window mechanisms which can predict this behavior. The control schemes that we study are categorized as "static" and "dynamic" flow controls. In static flow control, parameters of the system are not adjusted to the stochastic load fluctuations on the system. We study the optimal window size for static flow control which results in the maximum throughput. In dynamic flow control, parameters of the system are dynamically adjusted to the resources available in the network. Based on Markov decision theory, an optimal policy to dynamically decide on the window size is formulated. Because an exact solution to the problem is laborious, a heuristic solution to the problem is presented.

The issue of interconnection of computer networks (otherwise known as internetting) has recently become of major interest. In this

dissertation we study some problems related to internetting. Based on our study of static flow control, we develop a network algebra for series and/or parallel-connected networks and also investigate the problem of resource allocation in parallel-connected networks. Finally, we analyze a number of buffer allocation schemes in nodes where a number of networks merge together. These allocation schemes serve as a further tool for the control of traffic in networks.

In general, the goal of this dissertation is to develop analytic tools for the performance study and design of computer networks. In many cases the models which we use are rather simplistic; nevertheless, they help us understand the implications of real system behavior.

Chapters 2 and 3 are concerned with the switching problems in computer nets.

In Chapter 2 a hybrid switching technique, called "virtual cut-through", is introduced and its performance is compared with that of message switching. The delay analysis of "virtual cut-through" is carried out for two cases: noiseless channels and noisy channels. It is shown that in most cases, the network delay of virtual cut-through is less than that of message switching. At the end of this chapter we study an optimal design problem for this switching technique.

In Chapter 3 we carry out a tradeoff study of three switching techniques: virtual cut-through, message switching, and circuit switching. We first formulate a network model for delay in a circuit switched system; however, no simple solution to the transform equation which results from this model has been found. Next, we generalize the independence assumption [KLEI 64] and develop an analytic model for the

delay of a communication path between two users in a circuit switching network. Having thus developed expressions for delay in circuit switching, we compare the performance of the three switching techniques. While our study confirms the previous understandings that circuit switching is favorable for long and infrequent message traffic, it also shows that performance of circuit switching depends very much on the topology of the network and also on the number of channels per trunk. In general, for long paths, if the number of channels per trunk is properly selected, network delay for circuit switching is less than message switching and cut-through switching. Our study indicates that the selection of switching technique should be based upon a careful study of parameters of the network and characteristics of the traffic. At the end of this chapter we study an optimal design problem based on our comparison study between message switching and circuit switching.

Chapters 4 and 5 are concerned with the flow control problems in computer networks.

In Chapter 4 we develop analytic models for end-to-end communication protocols and study the window mechanism for flow control in store-and-forward computer-based communication networks. We start with a very simple, deterministic model in which there are no stochastic fluctuations in the load on the system. Then, step by step, we improve this basic model and develop a static flow control model in which the parameters of the system are not dynamically adjusted to the stochastic fluctuations of the system load.

In Chapter 5 we use the results of our study in Chapter 4 to develop a dynamic flow control in which window size and certain other

parameters of a network are dynamically adjusted to the stochastic fluctuations in the system load. The dynamic decision on optimum window size is formulated as a Markov decision process. Having thus formulated the process, we develop a heuristic solution to the problem as an alternative to the laborious exact solution. Based on the heuristic solution, a (sub)optimal policy to decide on window sizes is introduced and its performance is compared with the optimal one.

In Chapter 6 we study some issues related to the interconnection of computer networks. We begin by establishing a network algebra as an aid in the systematic study of computer networks interconnection; we then look at some design problems for allocation of resources among a number of parallel-connected networks. Finally we discuss buffer allocation strategies in nodes where a number of networks merge together.

Appendix A contains a list of often-used symbols.

In Appendix B we present proofs for some of the theorems stated in Chapter 2.

In Appendix C some results from Markov decision theory are presented. These results are used in Chapter 5.

In Appendix D we study various buffer sharing and/or allocation schemes for a multiplexing node [KERM 77]. The study shows that in order to provide a fair service to different streams of data, some non-trivial sharing schemes should be implemented. Chapter 6 contains performance curves for different sharing schemes presented in this appendix.

CHAPTER 2

"VIRTUAL CUT-THROUGH"

A NEW SWITCHING TECHNIQUE

One of the basic problems in a computer communication network design is to select the switching method. Early computer networks were built either over the existing telephone networks or with leased lines, using principles of telephone switching, namely line (circuit) switching. With circuit switching, a complete path of communication links must be set up between two parties before the real communication begins. This setup is accomplished via a signalling message. The path is tied up from the time the first bit of the first message until the last bit of the last message is transmitted between the two parties. This includes any idle periods during which the two parties are silent. Once a path is set up, no further signalling for addressing purposes is necessary; thus, in a circuit-switched network, a path, once set up, essentially provides all the addressing information.

When applied to data communication networks, circuit switching suffers from some drawbacks. One is the slow-call setup which delays the transfer of messages from sender to receiver. The fact is, circuit switching is essentially used in telephone networks which have been designed for human communication. A telephone conversation is typically an order of magnitude longer than the signalling delay. On the other hand, measurement studies [JACK 69, FUCH 70] conducted on time-sharing systems indicate that in computer communication, data streams are bursty.

That is, they consist of frequent short messages with sporadic long ones. The excessive signalling delay, especially for short messages, is a serious disadvantage of this switching technique. Another drawback is that channel utilization may be low due to the fact that the channels on a path are tied up but actually are not being used during the idle periods. Furthermore, as we will see in Chapter 3, the communication network saturates very fast due to excessive holding time of channels. Fig. 2.1-a shows the network delay in a circuit-switched system. It is assumed that there is no interfering traffic and that the number of intermediate nodes in the path is two.

In order to achieve a better channel utilization, one may think of relinquishing the channels on a path during periods in which the parties are silent. This brings us the idea of (store-and-forward) message switching. In this method, messages are routed toward their destination node without establishing a path beforehand. Through provision of a storage facility at each node, messages are stored in intermediate nodes and then are sent forward to a selected adjacent node (hence the name, store-and-forward). This selection is made by a well-defined decision rule referred to as the routing algorithm. The process is repeated until the message reaches the destination node. Each message carries with it the information regarding its destination. This is done by attaching addressing bits (usually a part of the header) to each message. In this switching method, by not allocating the communication links into complete paths for specific source-destination pairs of nodes, each link is statistically shared by many messages. Fig. 2.1-b shows the network delay in a message-switching system.

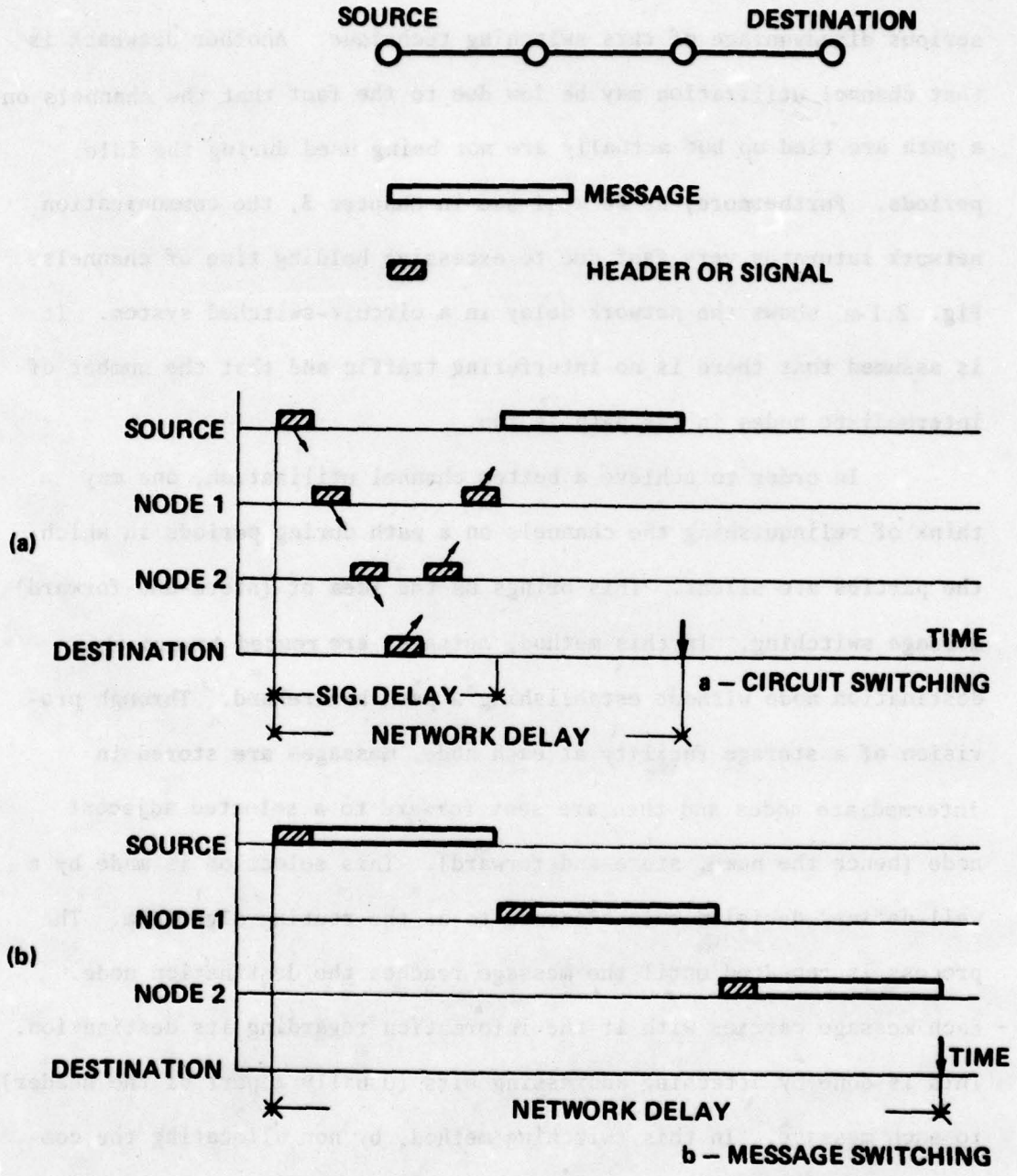
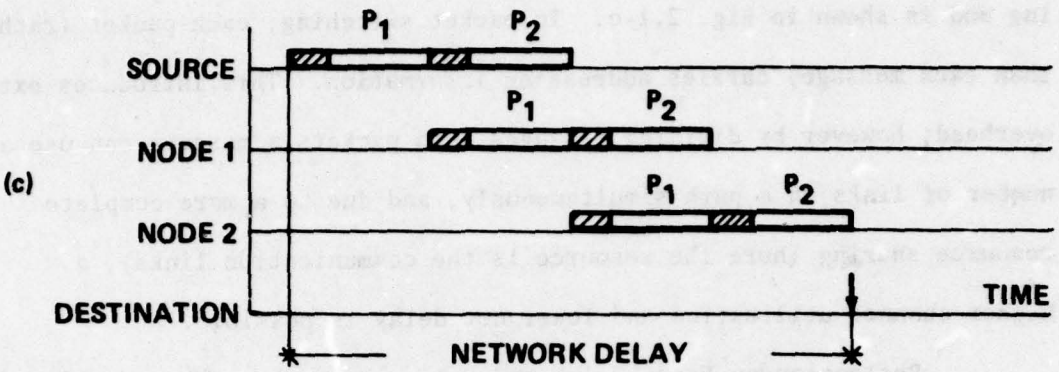
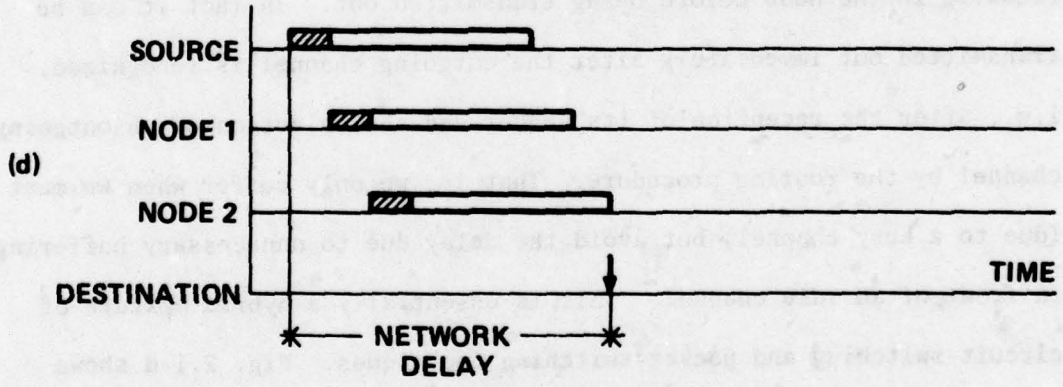


Fig. 2.1 Operations of Different Switching Systems.



c – PACKET SWITCHING



d – CUT-THROUGH SWITCHING

Fig. 2.1 (cont.)

Examination of Fig. 2.1-b suggests that a better utilization and further reduction in network delay is possible by dividing a message into smaller pieces called packets. This method is called packet switching and is shown in Fig. 2.1-c. In packet switching, each packet (rather than each message) carries addressing information. This introduces extra overhead; however by dividing messages into packets a message can use a number of links on a path simultaneously, and due to a more complete resource sharing (here the resource is the communication links), a higher channel utilization and lower net delay is possible.

Reflection on Figs. 2.1-b and 2.1-c shows that the extra delay in message switching is caused by the fact that we do not permit a message to be transmitted out of a node before it is received completely. We notice that when a message arrives in an intermediate node and its outgoing channel is free, it actually does not need to be completely received in the node before being transmitted out. In fact it can be transmitted out immediately after the outgoing channel is recognized, i.e., after the reception of its header and the selection of an outgoing channel by the routing procedure. That is, we only buffer when we must (due to a busy channel) but avoid the delay due to unnecessary buffering in front of an idle channel. This is essentially a hybrid mixture of circuit-switching and packet-switching techniques. Fig. 2.1-d shows the delay when this "virtual cut-through" method is used.

If the packet encounters busy channels at all of the intermediate nodes, the outcome is exactly the same as a packet-switched network. On the other hand, if all of the intermediate channels are free, the outcome is similar to a circuit-switched system. In general,

the delay of the cut-through system is somewhere between packet-switched and circuit-switched systems, and as Fig. 2.1-d shows, this delay is less for smaller packets.

In this chapter we will study and analyze the performance of this switching technique and compare it with the packet-switching method. We start with some assumptions and terminology.

2.1 Assumptions

In the opening of this chapter we discussed message- and packet-switching systems. The essential difference between these two systems is that in packet switching a message consists of multiple packets. In this chapter we assume that messages consist of single packets and we no longer differentiate between these two systems.

Consider a "tagged" message in a message-switching communication net. Whenever the message arrives in an intermediate node in which the outgoing channel is free, we say that the message arrives in a "free" node (this happens with probability $(1 - P_w)$, P_w being the probability of having to wait). Furthermore, because the message can be sent out immediately after its header is received (rather than the entire message), we say a "cut-through" has occurred. If after the reception of the header, the message cannot be sent out immediately (due to busy channels) then it must be received completely before being transmitted out (i.e., partial cuts are not allowed). The reason for this assumption is two-fold. First, in the case of noisy channels the message can be error-checked when it is blocked (more about this in Section 2.3.2), and second, this assumption makes the analysis simpler.

The following assumptions are accepted throughout the analysis:

1. External Poisson arrivals.
2. Exponentially distributed message lengths.
3. Infinite nodal capacity.
4. Deterministic routing.
5. Independence assumption [KLEI 64].
6. Balanced network (defined below).
7. Negligible propagation delay.

Definition 2.1

A network is said to be balanced if the utilization factor of all of its channels is the same. Note that this property does not imply any assumption on the topology of the network.

A detailed discussion and justification of the above assumptions can be found in [KLEI 64], [GERL 73A] and [KAMO 76B].

The analysis is carried out for multiple channel links. We use the word link for the communication media between nodes. Each link can be split into one or more channels.

Appendix A contains a list of notation used in this chapter.

2.2 Analysis of the Number of Cut-Throughs

Although the average number of cut-throughs is not a good measure of performance it is used to evaluate the network delay. In this section we investigate some of its properties.

Theorem 2.1

The average number of cut-throughs in a balanced network with

channel utilization ρ is given by

$$\bar{n}_c = (\bar{n}_h - 1)(1 - P_w) \tag{2.1}$$

where $\bar{n}_c$ is the average number of cut-throughs, $\bar{n}_h$ is the average number of hops (or path length), and P_w is the probability of waiting (no cut).

Proof:

Whenever a message enters a free node it can make a cut. This event occurs with probability $1 - P_w$. Due to the independence assumption, the number of cuts has a binomial distribution [FELL 50] and we have:

$$Pr[\tilde{n}_c = k \mid \tilde{n}_h = n, \rho] = \binom{n-1}{k}(1 - P_w)^k P_w^{n-k-1} \qquad 0 \leq k \leq n-1$$

The conditional generating function of the number of cut-throughs may be written as

$$C_{n,\rho}(z) = E[z^{\tilde{n}_c} \mid \tilde{n}_h = n, \rho] = \sum_{k=0}^{n-1} z^k \binom{n-1}{k} (1 - P_w)^k P_w^{n-k-1}$$

So,

$$C_{n,\rho}(z) = [P_w + z(1 - P_w)]^{n-1}$$

We may now find the conditional mean number of cut-throughs:

$$\bar{n}_c^n = E[\tilde{n}_c \mid \tilde{n}_h = n, \rho] = \frac{dC_{n,\rho}(z)}{dz}\bigg|_{z=1} = (n-1)(1 - P_w)$$

$$\bar{n}_c^n = (n-1)(1 - P_w)$$

The mean number of cut-throughs is therefore

$$\bar{n}_c = \sum_n E[\tilde{n}_c \mid \tilde{n}_h = n, \rho] Pr[\tilde{n}_h = n] = (\bar{n}_h - 1)(1 - P_w)$$

and so

$$\bar{n}_c = (\bar{n}_h - 1)(1 - P_w)$$

Q.E.D.

The unconditional generating function of the number of cut-throughs is derived below:

$$C_\rho(z) = \sum_n C_{n,\rho}(z) Pr[\tilde{n}_h = n] = \sum_n [P_w + z(1 - P_w)]^{n-1} Pr[\tilde{n}_h = n]$$

or

$$C_\rho(z) = \frac{N[P_w + z(1 - P_w)]}{P_w + z(1 - P_w)} \qquad (2.2)$$

where $N(z) \triangleq E[z^{\tilde{n}_h}]$ is the generating function of the number of hops. Notice that

$$\bar{n}_c = \frac{dC_\rho(z)}{dz}\bigg|_{z=1} = \frac{N^{(1)}(1) - N(1)}{1}(1 - P_w)$$

but

$$N^{(1)}(1) = \bar{n}_h$$

so

$$\bar{n}_c = (\bar{n}_h - 1)(1 - P_w)$$

as we had before.

For the special case of single channel links, Eqs. (2.1) and (2.2) are reduced to

$$\bar{n}_c = (\bar{n}_h - 1)(1 - \rho) \qquad (2.3)$$

$$C_\rho(z) = \frac{N[\rho + z(1 - \rho)]}{\rho + z(1 - \rho)} \qquad (2.4)$$

which were first reported in [KLEI 76A].

These equations show that when the average number of hops ($\bar{n}_h$) increases, so does the average number of cut-throughs. This is intuitively clear when we notice that as the number of intermediate nodes increases there is a higher chance of experiencing more cuts. They also show that $\bar{n}_c$ is a decreasing function of ρ which means that in a less crowded network the average number of cuts is larger.

Special Cases

(I) $\rho = 1$ This implies $P_w = 1$ and $\bar{n}_c = 0$, i.e, no cut-throughs are made.

(II) $\rho = 0$ This implies $P_w = 0$ and $\bar{n}_c = \bar{n}_h - 1$.

In case (I), with probability one, all of the intermediate nodes are busy upon arrival of the message. The network then behaves like a pure message-switched system. In (II), with probability one, all of the intermediate nodes are free and all of them are cut through; thus it resembles a circuit-switched system. Later in this chapter we will show performance curves for the system.

2.3 Delay Analysis

An important performance measure for a computer communication net is the average source to destination message delay T, defined below:

$$T = \sum_{i,j} \frac{\gamma_{ij}}{\gamma} Z_{ij}$$

where γ_{ij} is the average number of messages entering the network per second with origin i and destination j, γ is the total arrival rate of messages from external sources, and Z_{ij} is the average message delay for messages with origin i and destination j.

For a message-switched system, a straightforward application of Little's result [LITT 61] to the queueing model leads to the following expression for T [KLEI 64]. (Here we use the symbol T_m to indicate that this delay is for message-switching systems.)

$$T_m = \sum_i \frac{\lambda_i}{\gamma} T_i$$

where T_i is the average delay at node i.

Calculation of T_i is in general a nontrivial and currently unachieved task; however, by accepting the assumptions of Section 2.1, especially the independence assumption of Kleinrock [KLEI 64], we are in a position to reduce the network of queues model to the model first studied by Jackson [JACK 57]. By virtue of his results, each node behaves stochastically independent of the other nodes and similar to an M/M/m system, for which the average delay is reported in Appendix B.1. For the special case of uniform ρ, all of the nodal delays are identical and we have

$$T_m = T_i \sum_i \frac{\lambda_i}{\gamma}$$

but we have [KLEI 64]

$$\sum_i \frac{\lambda_i}{\gamma} = \bar{n}_h$$

so we get

$$T_m = \bar{n}_h T_i \qquad (2.5)$$

So far we have not made any assumption regarding the channel error rate. We will initially give an analysis of delay for a network with noiseless channels. It turns out that for such a system the

analysis is fairly simple and a closed form expression for network delay is obtainable. However, when the network contains noisy channels, as we will see shortly, the network delay can be calculated through a fairly complicated iterative routine. We will present a delay analysis of these two systems; however, later we deal only with networks with noiseless channels.

2.3.1 Delay Analysis for a Network with Noiseless Channels

When channels are error free, each nodal delay becomes similar to the system delay of an M/M/m queueing system (Appendix B.1). So we have

$$T_i = \frac{N_{ch}}{\mu C} + \frac{P_w}{\mu C(1-\rho)}$$

and

$$T_m = \left(\frac{N_{ch}}{\mu C} + \frac{P_w}{\mu C(1-\rho)} \right) \bar{n}_h \tag{2.6}$$

where N_{ch} is the number of channels per link, $1/\mu$ is the average message length, and C is the total channel capacity.

For the special case of $N_{ch} = 1$ we get

$$T_m = \left(\frac{1}{\mu C(1-\rho)} \right) \bar{n}_h = \frac{\bar{n}_h}{\mu C - \lambda}$$

For the average network delay in the cut-through system we have the following result:

Theorem 2.2

The average network delay in the cut-through system is given by

$$T_c = T_m - (\bar{n}_h - 1)(1 - P_w)(1 - \alpha)t_0 \tag{2.7}$$

where t_0 ($= 1/\mu C$) is the average transmission time of a message on a channel, t_h is the average transmission time of a header and $\alpha = t_h/t_0$.

Proof:

For each node at which a cut-through is made, a nodal service time is saved. However, this service time is conditioned on the event that the waiting time is zero. So we have

$$T_m - T_c = \bar{n}_c E[\tilde{s} \mid \tilde{w} = 0]$$

where $\bar{n}_c$ is the average number of cuts, $\tilde{s}$ is the total delay in a node and $\tilde{w}$ is the waiting time in a node.

However, $E[\tilde{s} \mid \tilde{w} = 0] = t_0$, so

$$T_m - T_c = \bar{n}_c t_0$$

If we assume that a message can be sent out only after its header is received, then the previous equation is changed to

$$T_m - T_c = \bar{n}_c (t_0 - t_h)$$

where $(t_0 - t_h)$ is the saving in delay at each node when the cut-through method is used. Using the value of $\bar{n}_c$ from Eq. (2.1), we get

$$T_c = T_m - (\bar{n}_h - 1)(1 - P_w)(1 - \alpha)t_0$$

Q.E.D.

For the special case of $N_{ch} = 1$ and $\alpha = 0$, we have $T_m = \dfrac{\bar{n}_h t_0}{1 - \rho}$ and $P_w = \rho$, and Eq. (2.7) is reduced to

$$T_c = \frac{\bar{n}_h t_0}{1 - \rho} - (\bar{n}_h - 1)(1 - \rho)t_0 \qquad (2.8)$$

From Eq. (2.7) we have

$$T_m - T_c = \begin{cases} (\bar{n}_h - 1)(1 - \alpha)t_0 & \text{for } \rho = 0 \\ 0 & \text{for } \rho = 1 \end{cases} \qquad (2.9)$$

which shows that if all of the intermediate nodes are busy ($\rho = 1$ and $P_w = 1$), no reduction in delay is made. On the other hand, when $\rho = 0$ and there is no intermediate busy node, the transmission times of the intermediate channels are saved. This fact becomes clearer when we set $\alpha = 0$.

For $\alpha = 1$, Eq. (2.7) gives $T_c = T_m$; this is intuitively clear because $\alpha = 1$ implies that a message is completely composed of its header and the cut-through system is identical to message switching.

2.3.2 Delay Analysis for a Network with Noisy Channels

A detailed analysis of a network with noisy channels can be very complicated. The difficulty is that in a network with unreliable channels there should be acknowledgement and time-out mechanisms to guarantee correct delivery of messages; modeling of such mechanisms is usually an involved task [KAMO 76B], [LAM 75A]. In a later chapter of this dissertation, on dealing with flow control techniques, we will present a detailed model of an acknowledgement mechanism; however in this section we accept a very simplified model for this mechanism. For message switching we assume if a message is received with some bits in error (which happens with a constant probability P_e), it is retransmitted immediately (i.e., instantaneous negative and positive acknowledgement). We further assume that with the retransmission traffic, the network is still balanced (Definition 2.1). A logical picture of each node is given in Fig. 2.2.

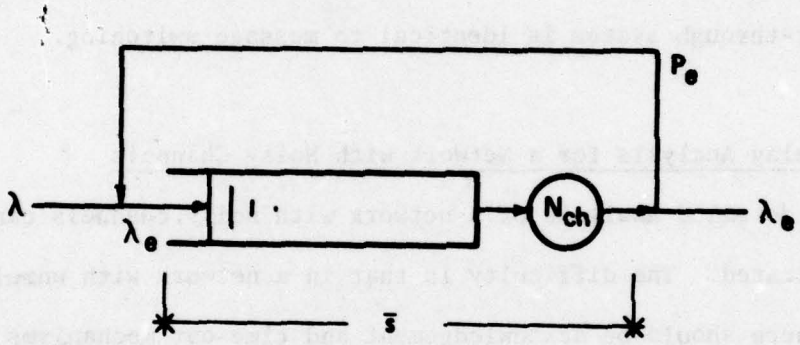


Fig. 2.2 Simplified Model of A Store-and-Forward Node.

If the average delay for a one-pass transmission is $\bar{s}$, the total nodal delay becomes

$$T_i = \frac{\bar{s}}{1 - P_e} = \left[\frac{N_{ch}}{\mu C} + \frac{P_w}{\mu C(1 - \rho_e)} \right] / (1 - P_e)$$

and the network delay for message switching will be

$$T_m = \left[\frac{N_{ch}}{\mu C} + \frac{P_w}{\mu C(1 - \rho_e)} \right] \bar{n}_h / (1 - P_e) \qquad (2.10)$$

In the above expression ρ_e is the effective network traffic. The useful traffic is given by

$$\rho = (1 - P_e)\rho_e \qquad (2.11)$$

For the cut-through system the operation is somewhat more involved. We assume that a message is error-checked only in its destination node or any intermediate node which it can not cut through. (We call this type of node a "final node" which is not necessarily the same as the destination node, although the destination node is always a final node). It is not that a message could not be checked for error in an intermediate node which it cuts through, but even if an error were detected (this can be done only after the last bit of a message arrives in a node) there is no way to stop the transmission and notify the next node(s) not to accept the message because the message has already begun to be transmitted out of the node. In a final node error checking can be done because the message must be received completely. If it is recognized that the message is erroneous, it has to be retransmitted through all of the intermediate nodes (including the first node on the cut-through path). As before, we further assume that acknowledgements (positive and/or negative) are instantaneous. Figure 2.3 makes the

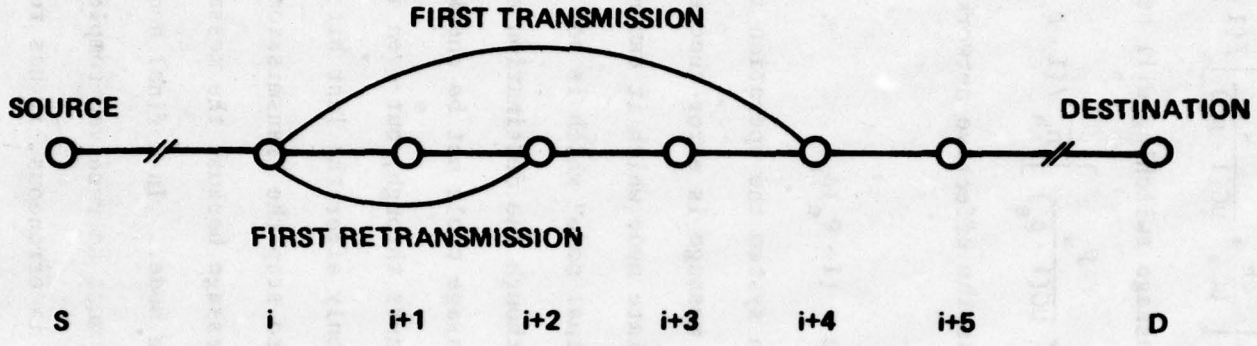


Fig. 2.3 Example of Retransmission in the Cut-Through System.

operations more clear. In this figure, a message is to be transmitted from source node S to destination node D. The message has already made its way to an intermediate node i (in which it has been blocked). When it starts transmission from this node, it makes a cut through nodes $i+1$, $i+2$ and $i+3$ and is again blocked at node $i+4$. At this node, after it is checked, the message is found to have some bits in error. Through negative acknowledgement (which we assume is instantaneous), it is retransmitted from node i. In the retransmission process, however, it can make a cut through node $i+1$ only and is blocked at node $i+2$. Depending on whether it is received incorrectly or not, further retransmission from node i is necessary or it can start making its way toward its destination node D (after queueing delay).

The delay analysis for the cut-through system in a network with noisy channels is fairly complicated, and in fact no closed form expression for it could be derived. The conditional delay, when the number of hops is constant ($\tilde{n}_h = n$), is given below.

Theorem 2.3

The average network delay for the cut-through system in a network with noisy channels is given by

$$T_c = T_c^{(n)}(n)$$

where

$$T_c^{(j)}(i) \triangleq \begin{cases} \sum_{k=0}^{j-1} \left\{ T_{cc}^{(k)}(i,j) + T_c^{(j-k+1)}(i)(1-P_e)^{k+1} + T_c^{(j)}(j)\left[1-(1-P_e)^{k+1}\right] \right\} P_c^{(j)}(k) & (i > j > 0) \\ \dfrac{\sum_{k=0}^{j-1} \left[T_{cc}^{(k)}(i,j) + T_c^{(j-k-1)}(i)(1-P_e)^{k+1} \right] P_c^{(j)}(k)}{\sum_{k=0}^{j-1} (1-P_e)^{k+1} P_c^{(j)}(k)} & (i = j > 0) \\ 0 & (j = 0) \end{cases}$$

where (2.12)

$$T_{cc}^{(k)}(i,j) \triangleq \begin{cases} E[\tilde{s}] + kt_h = \left[\dfrac{N_{ch}}{\mu C} + \dfrac{P_w}{\mu C(1-\rho_e)} \right] + kt_h & (i = j) \\ E[\tilde{s} \mid \tilde{w} > 0] + kt_h = \dfrac{N_{ch}}{\mu C} + \dfrac{1}{\mu C(1-\rho_e)} + kt_h & (i > j) \end{cases}$$

and

$$P_c^{(j)}(k) = \begin{cases} (1-P_w)^k P_w & (0 \leq k < j-1 \quad \text{and} \quad j > 1) \\ (1-P_w)^k & (k = j-1) \end{cases}$$

where ρ_e is the effective traffic and n is the path length. (See Appendix A for the description of notations.)

Proof: Appendix B.2

If we set $P_e = 0$, after some algebra Eq. (2.12) reduces to Eq. (2.7), the delay for the cut-through system in a network with noiseless channels.

The useful traffic is related to effective traffic according to the following theorem.

Theorem 2.4

For the cut-through system in a balanced network with noisy channels, the useful traffic is determined according to the following relationships

$$\rho = \frac{n}{N_t(n)} \rho_e \tag{2.13}$$

where

$$N_t(i) \triangleq \begin{cases} \dfrac{\sum_{k=0}^{i-1} \left[(k+1) + N_t(i-k-1)(1-P_e)^{k+1} \right] P_c^{(i)}(k)}{\sum_{k=0}^{i-1} (1-P_e)^{k+1} P_c^{(i)}(k)} & i > 0 \\ 0 & i = 0 \end{cases}$$

where ρ_e is effective channel traffic and n is the path length.

Proof: Appendix B.3.

Some special cases are of interest:

I. If $P_w \uparrow 1$, then Eqs. (2.10) and (2.12) become identical. This is intuitively clear as in this case $\bar{n}_c = 0$ and retransmissions take place only over one hop in both systems.

II. If $n = 1$, again Eqs. (2.10) and (2.12) become identical.

III. If $P_e = 0$ then both Eqs. (2.11) and (2.13) reduce to $\rho = \rho_e$. Clearly in this case there is no retransmission and effective traffic is the same as useful traffic.

In Section 2.5 we will present some performance curves and will see the effect of channel errors on network performance. In the remaining sections of this chapter we will assume that network channels are noiseless.

2.4 Traffic Gain

Analysis of Eq. (2.7) shows that

$$T_m(\rho) - T_c(\rho) \geq 0 \qquad 0 \leq \rho \leq 1$$

(Here we use $T_m(\rho)$ and $T_c(\rho)$ to indicate the delay at a certain traffic.)

Later in this chapter we present some performance curves which explicitly show this fact. This property indicates that at the same traffic level the network delay in the cut-through system is less than in message switching. Equivalently, for the same network delay the cut-through system can handle more traffic; see the sketch, Fig. 2.4. It is the purpose of this section to calculate this throughput increase. (Actually, what we find is the difference between channel utilization of the two systems at the same delay; however, this gives us a measure of traffic gain.)

If we set $T_m(\rho_1) = T_c(\rho_2)$ we get

$$\bar{n}_h \left[\frac{N_{ch}}{\mu C} + \frac{P_{w_1}}{\mu C(1 - \rho_1)} \right] = \bar{n}_h \left[\frac{N_{ch}}{\mu C} + \frac{P_{w_2}}{\mu C(1 - \rho_2)} \right] - (\bar{n}_h - 1)(1 - P_{w_2})(1 - \alpha) \frac{N_{ch}}{\mu C} \qquad (2.14)$$

where P_{w_i} is the waiting probability at a node at which the utilization factor is ρ_i (recall that we are assuming channels are noiseless).

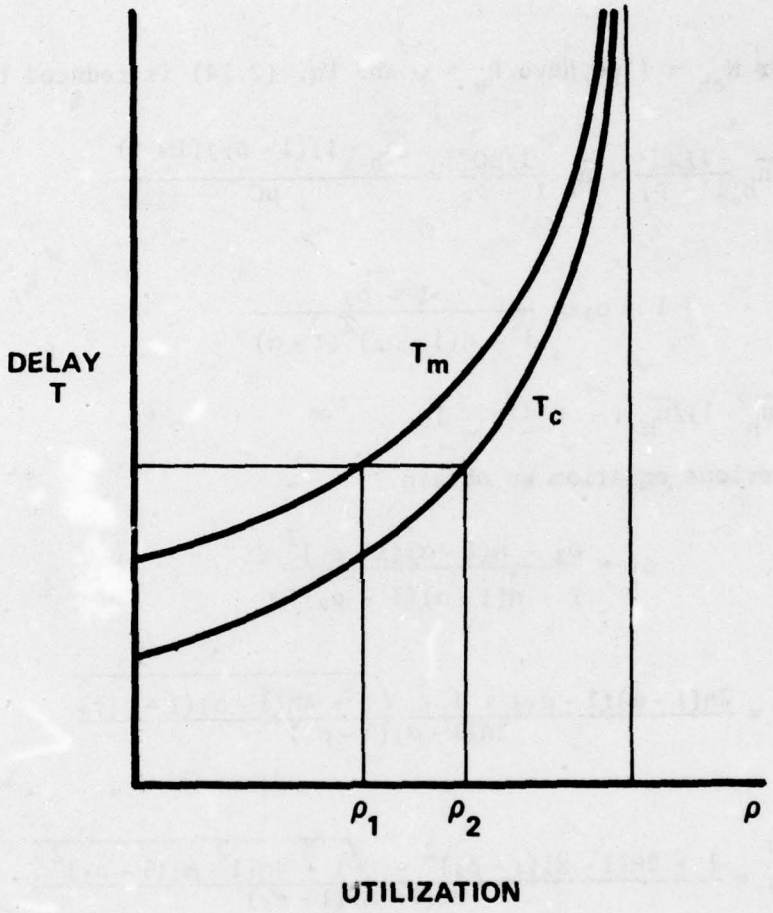


Fig. 2.4 Sketch of Traffic Gain in the Cut-Through System.

For $N_{ch} > 1$, P_w is a complicated expression of ρ and N_{ch} (see Appendix B.1), and it is not possible to find ρ_2 in terms of ρ_1 explicitly. For this reason we solve Eq. (2.14) analytically only for the case where $N_{ch} = 1$. For higher values of N_{ch} numerical techniques are necessary.

For $N_{ch} = 1$ we have $P_w = \rho$ and Eq. (2.14) is reduced to

$$\bar{n}_h \frac{1/\mu C}{1 - \rho_1} = \bar{n}_h \frac{1/\mu C}{1 - \rho_2} - \frac{(\bar{n}_h - 1)(1 - \rho_2)(1 - \alpha)}{\mu C}$$

or

$$1 - \rho_1 = \frac{1 - \rho_2}{1 - \eta(1 - \rho_2)^2(1 - \alpha)}$$

where $\eta = (\bar{n}_h - 1)/\bar{n}_h$, $0 \leq \eta \leq 1$.

from the previous equation we obtain

$$\rho_1 = \frac{\rho_2 - \eta(1 - \alpha)(1 - \rho_2)^2}{1 - \eta(1 - \alpha)(1 - \rho_2)^2} \tag{2.15a}$$

$$\rho_2 = \frac{2\eta(1 - \alpha)(1 - \rho_1) + 1 - \sqrt{1 + 4\eta(1 - \alpha)(1 - \rho_1)^2}}{2\eta(1 - \alpha)(1 - \rho_1)} \tag{2.15b}$$

and

$$\rho_2 - \rho_1 = \frac{1 + 2\eta(1 - \alpha)(1 - \rho_1)^2 - \sqrt{1 + 4\eta(1 - \alpha)(1 - \rho_1)^2}}{2\eta(1 - \alpha)(1 - \rho_1)} \tag{2.16}$$

When $\bar{n}_h = 1$ ($\eta = 0$), then $\rho_2 = \rho_1$; i.e., there is no gain. This is the case for a fully connected net. For $\bar{n}_h \uparrow \infty$ ($\eta = 1$) we have

$$\rho_2 - \rho_1 = \frac{1 + 2(1 - \alpha)(1 - \rho_1)^2 - \sqrt{1 + 4(1 - \alpha)(1 - \rho_1)^2}}{2(1 - \alpha)(1 - \rho_1)} \tag{2.17}$$

Notice that when $\bar{n}_h \to \infty$, T_m and T_c both grow to infinity; however in the limit there is still a gain in using the cut-through system.

2.5 Some Performance Curves and Simulation Results

Figure 2.5 shows the number of cuts as a function of ρ, the channel utilization. The network configuration and traffic pattern is shown in Fig. 2.6. For a single channel system $\bar{n}_c$ is a linear function of ρ. As the number of channels increases, the curve tends to take a step shape and in the limit it becomes a step function. This figure also shows some simulation results for $N_{ch} = 1$, 2 and 4. The statistics are from the model shown in Fig. 2.6. It is a tandem queue in which, except for the first node and the last ones, the traffic goes through three hops (2 intermediate nodes). The reason for selecting this topology was to create a uniform traffic. Although we did not use the independence assumption in the simulation and the message lengths were constant, the match between the simulated and the analytic results is very encouraging.

Figures 2.7-a through 2.7-f show the normalized delay curves ($T_m\mu C$ and $T_c\mu C$) as a function of useful channel traffic. Notice that throughput is not necessarily equal to the effective channel traffic. They are related to each other according to Eqs. (2.11) and (2.13). The curves are for a number of hops equal to 3 and different values of $\alpha(t_h/t_0)$ and P_e (channel error rate). The effect of header length can be seen by comparing Figs. 2.7-a with 2.7-b, 2.7-c with 2.7-d or 2.7-e with 2.7-f. It is seen that larger header lengths degrade performance of the cut-through system. The effect of channel error rate can also be

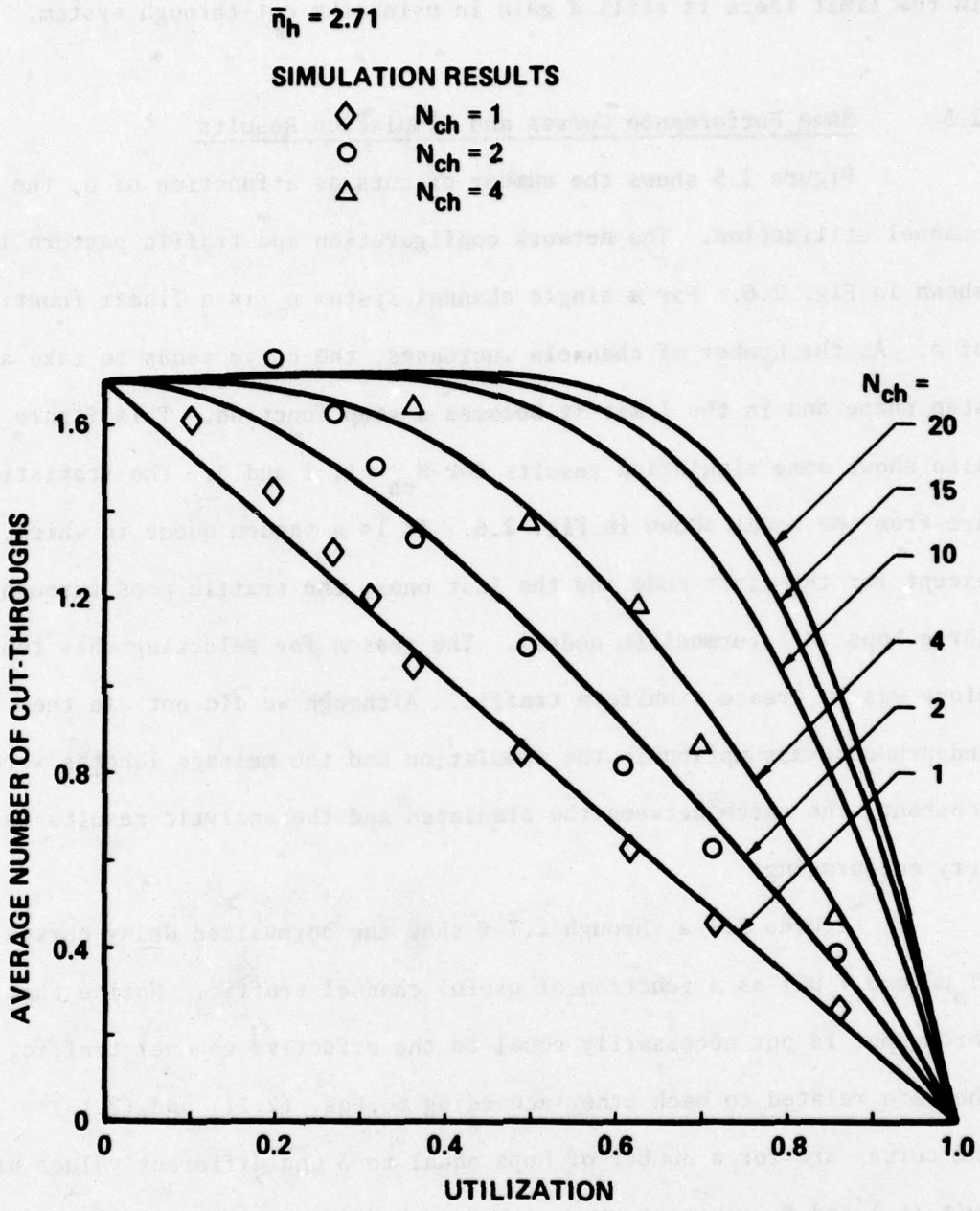
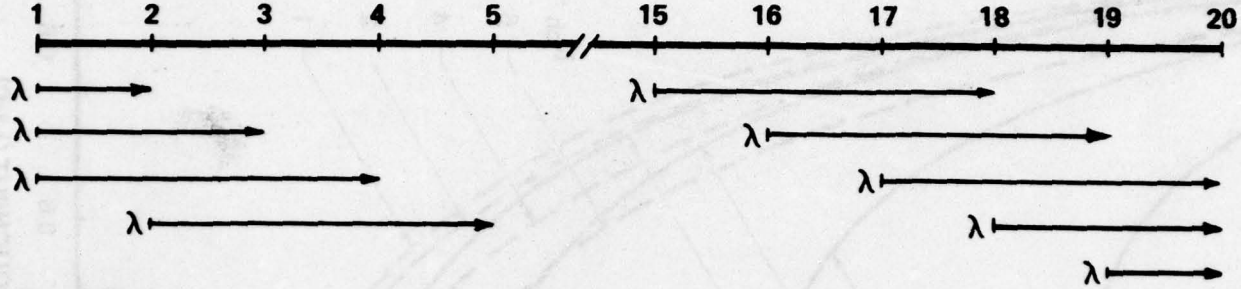


Fig. 2.5 Average Number of Cut-Throughs.



$\bar{n}_h = 2.71$

$\frac{1}{\mu C} = 20$ m sec.

$a = 0.1$

Fig. 2.6 The Simulated Tandem Queues.

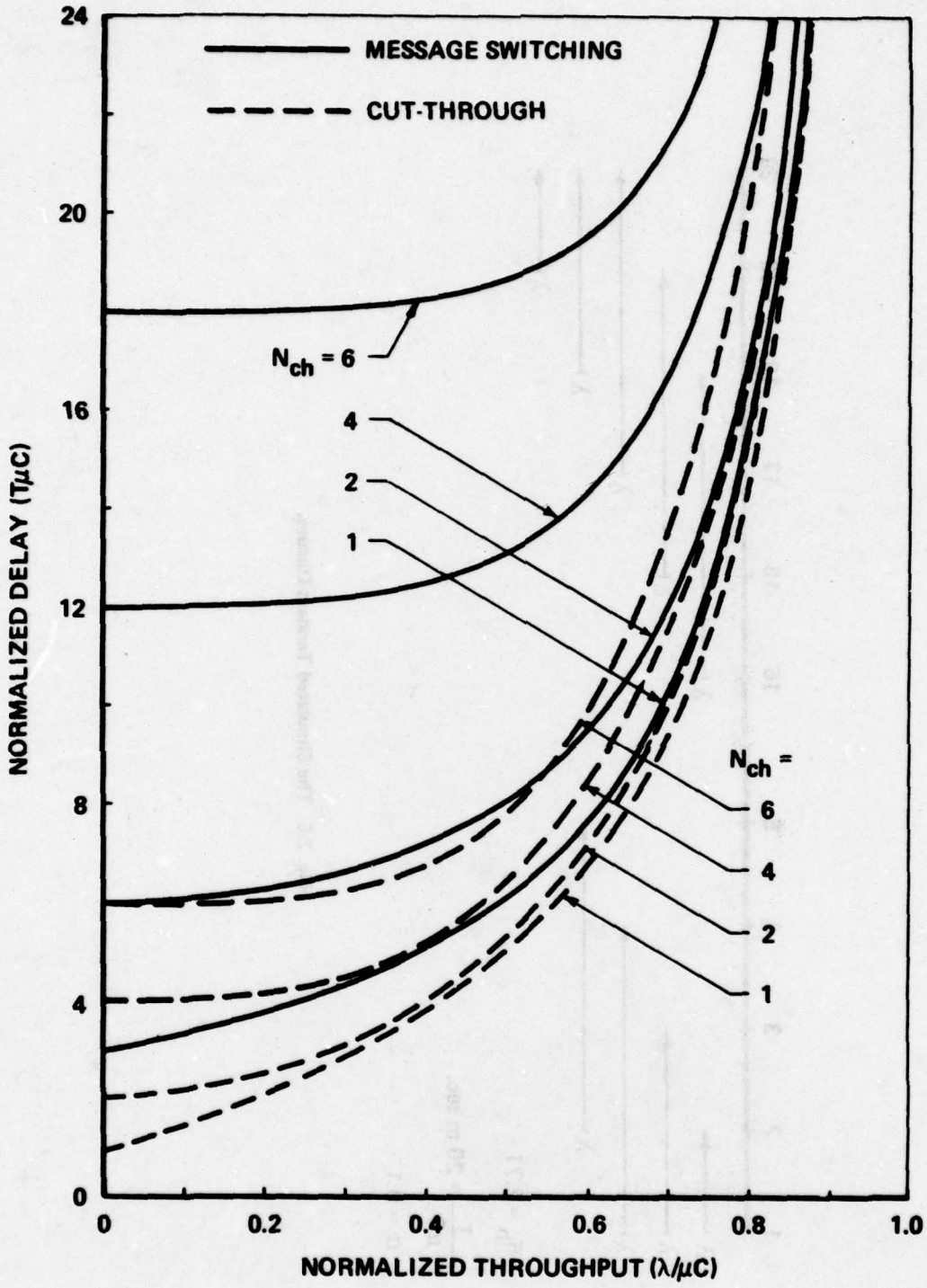


Fig. 2.7-a Delay Vs. Throughput
$\bar{n}_h = 3$, $P_e = 0.0, \alpha = 0.0$

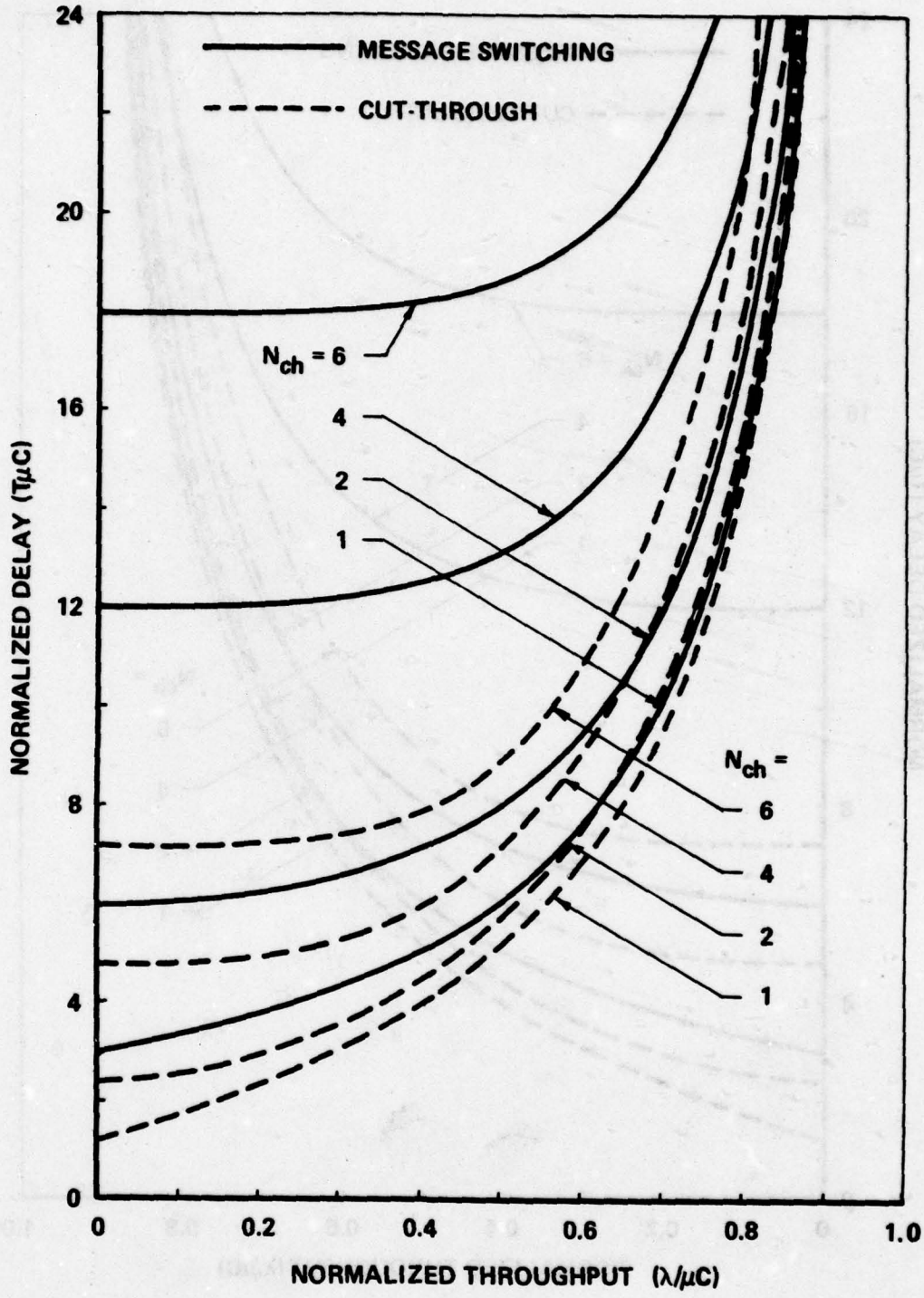


Fig. 2.7-b Delay Vs. Throughput

$\bar{n}_h = 3$, $P_e = 0.0$, $a = .1$

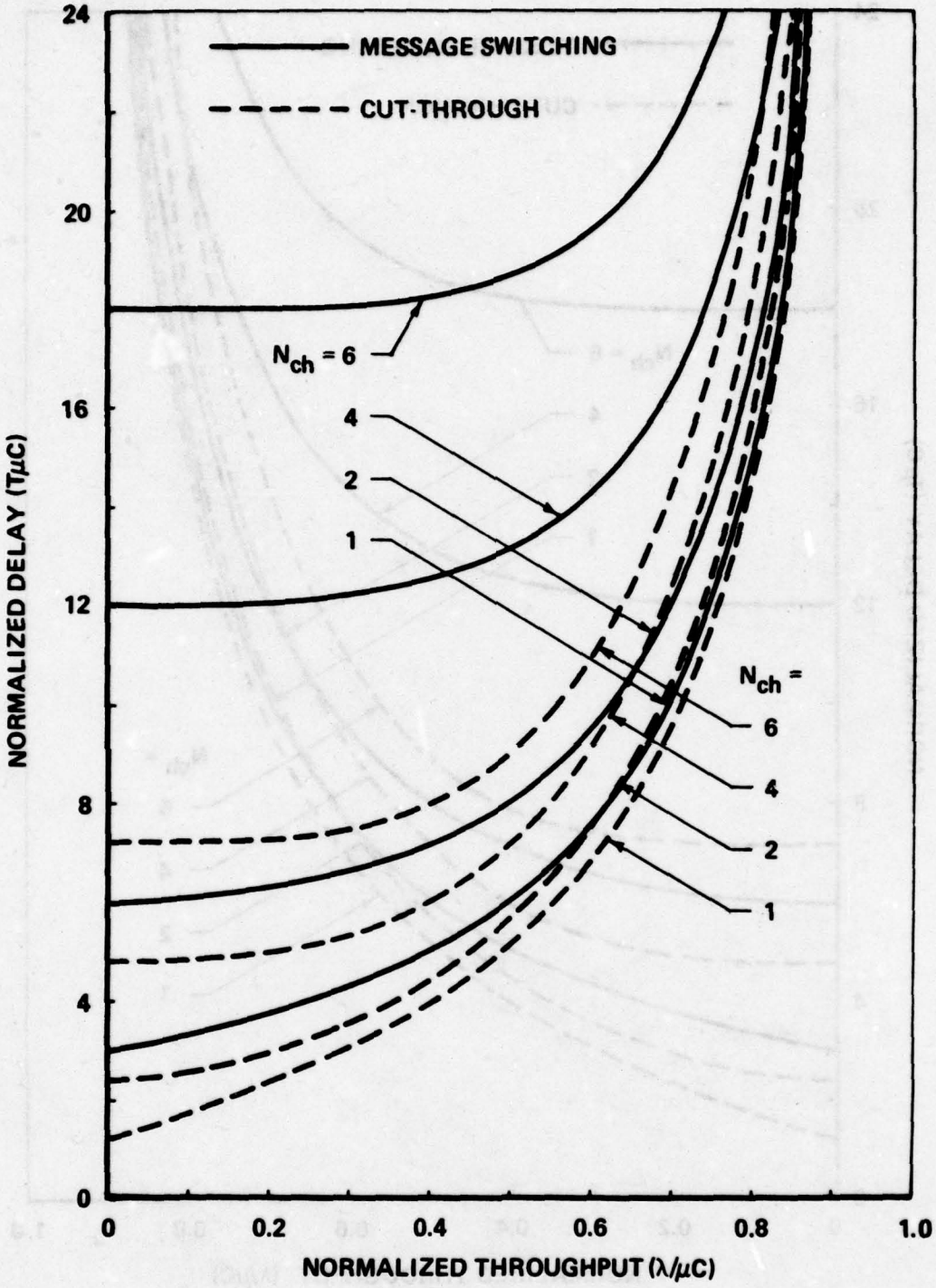


Fig. 2.7-c Delay Vs. Throughput

$\bar{n}_h = 3$, $P_e = 0.001$, $\alpha = 0.0$

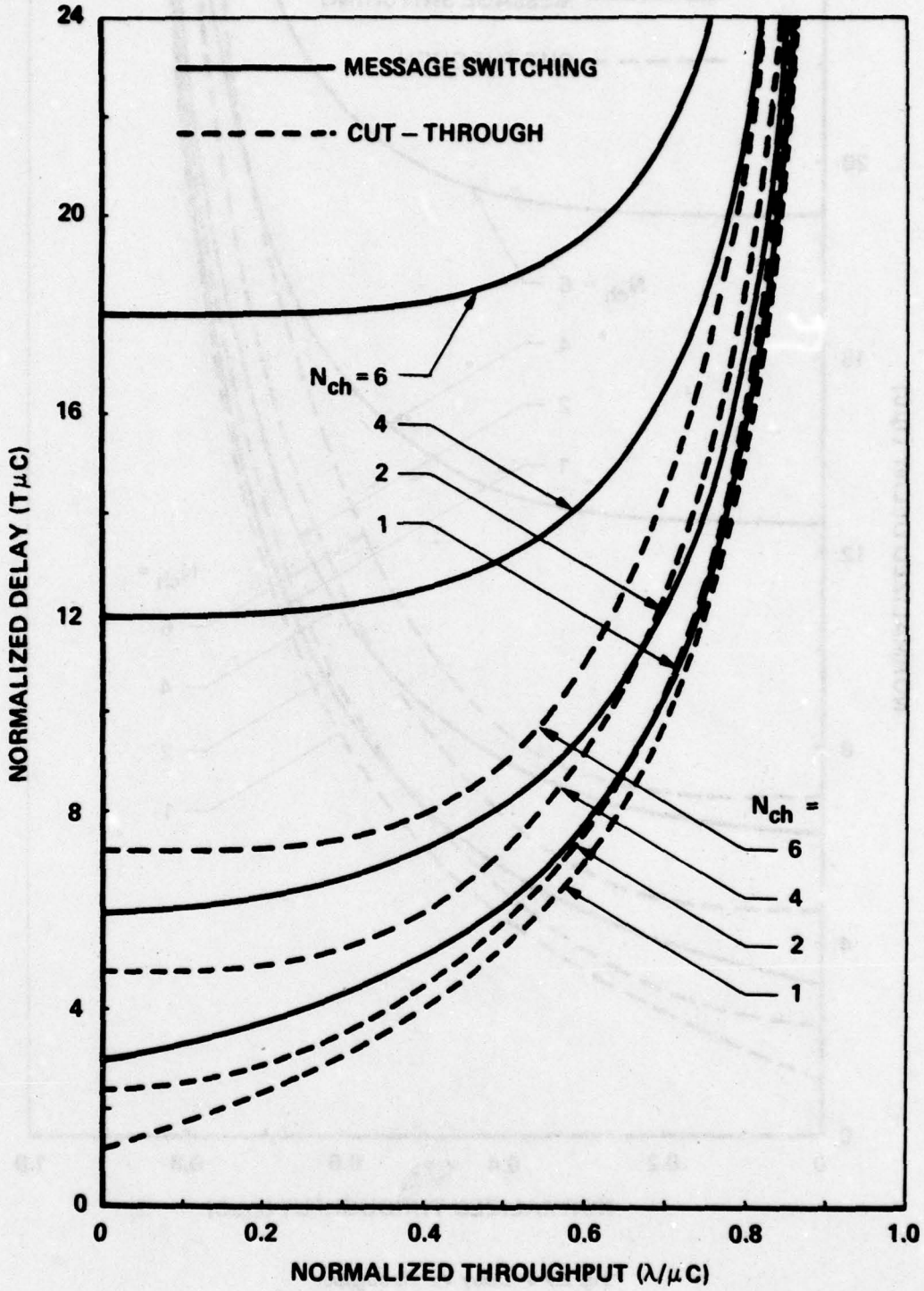


Fig. 2-7-d Delay Vs. Throughput
$\bar{n}_h = 3$, $P_e = 0.001$, $\alpha = 0.1$

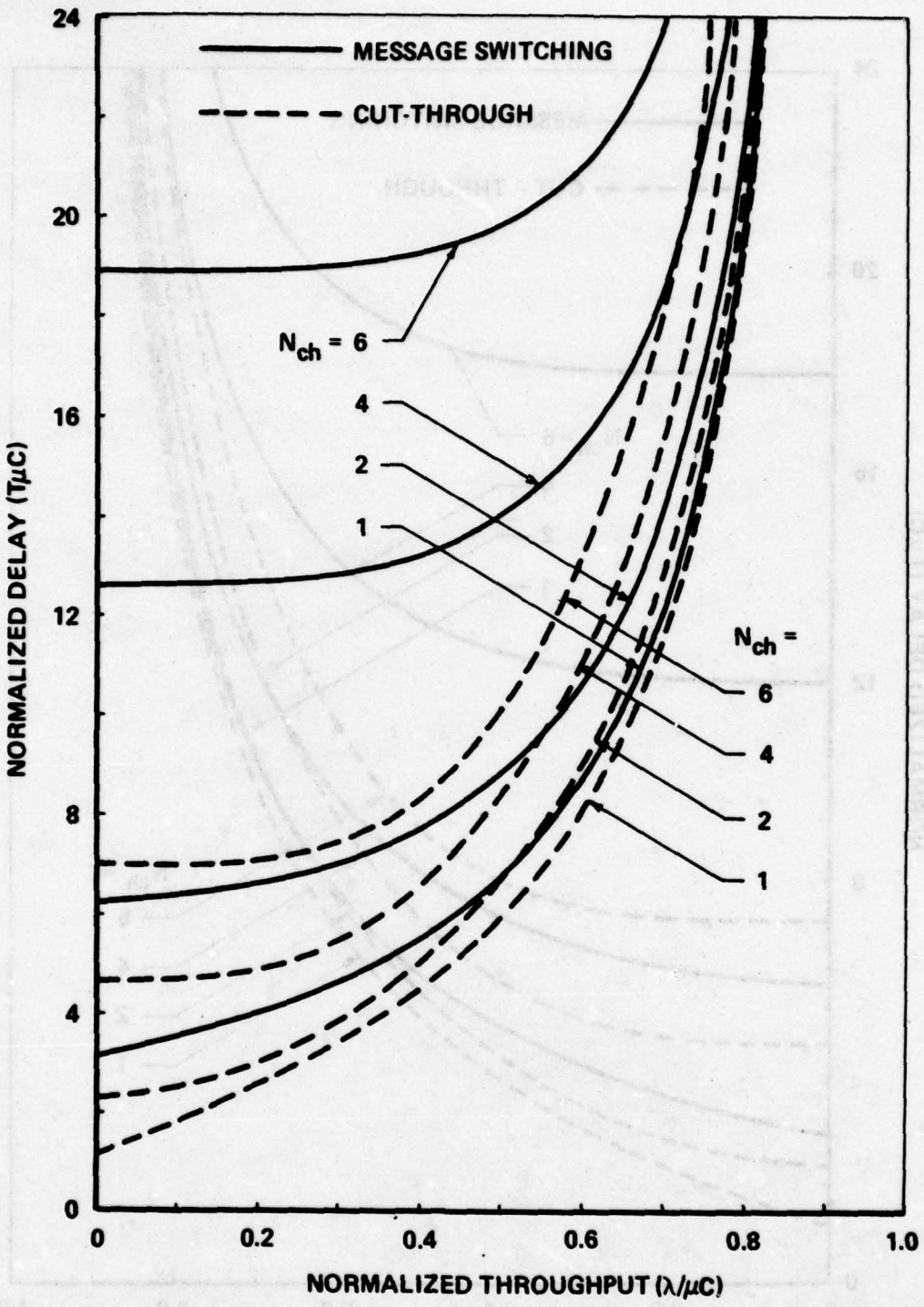


Fig. 2.7-e Delay Vs. Throughput

$\bar{n}_h = 3$, $P_e = 0.05$, $\alpha = 0.0$

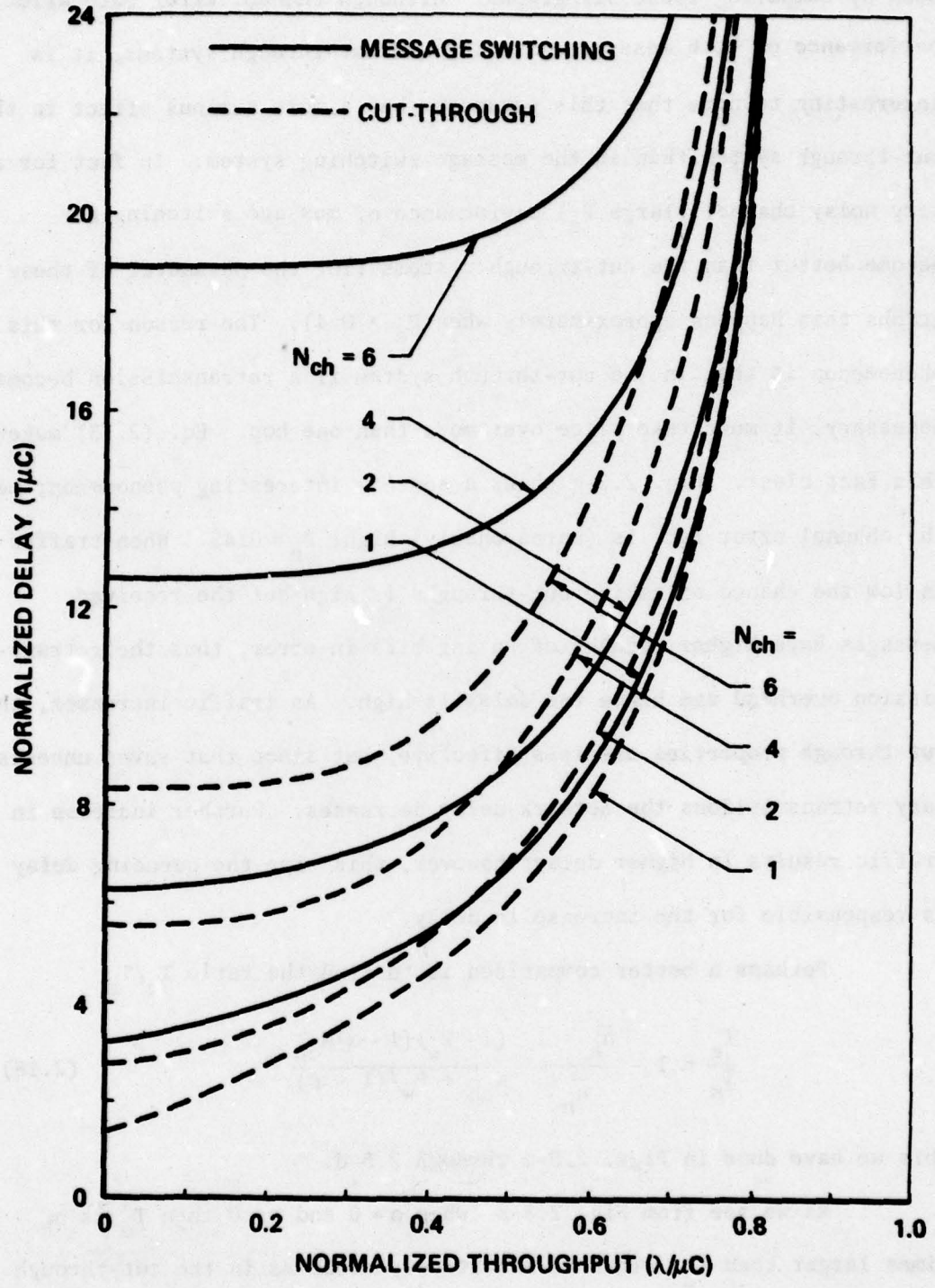


Fig. 2.7-f Delay Vs. Throughput

$\bar{n}_h = 3$, $P_e = .05$, $\alpha = 0.1$

seen by comparing these six graphs. Although channel error rate affects performance of both message-switching and cut-through systems, it is interesting to note that this parameter has a more serious effect in the cut-through system than in the message switching system. In fact for a very noisy channel (large P_e) performance of message switching may become better than the cut-through systems (for the parameter of these graphs this happens approximately when $P_e > 0.4$). The reason for this phenomenon is that in the cut-through system if a retransmission becomes necessary, it must take place over more than one hop. Eq. (2.13) makes this fact clear. Fig. 2.7-g shows a somehow interesting phenomenon; here the channel error rate is (unreasonably) high, $P_e = 0.45$. When traffic is low the chance of making cut-throughs is high but the received messages have higher chances of having bits in error, thus the retransmission overhead and hence the delay is high. As traffic increases, the cut-through properties are less effective, but since that saves unnecessary retransmissions the network delay decreases. Further increase in traffic results in higher delay; however, this time the queueing delay is responsible for the increase in delay.

Perhaps a better comparison is to find the ratio T_c/T_m

$$\frac{T_c}{T_m} = 1 - \frac{\bar{n}_h - 1}{\bar{n}_h} \frac{(1 - P_w)(1 - \alpha)N_{ch}}{N_{ch} + P_w/(1 - \rho)} \qquad (2.18)$$

This we have done in Figs. 2.8-a through 2.8-d.

As we see from Fig. 2.8-a when $\alpha = 0$ and $\rho = 0$ then T_m is $\bar{n}_h$ times larger than T_c. This is intuitively clear as in the cut-through system, messages can snake through all of the intermediate nodes without experiencing any delay (recall that $\alpha = 0$); however, in message

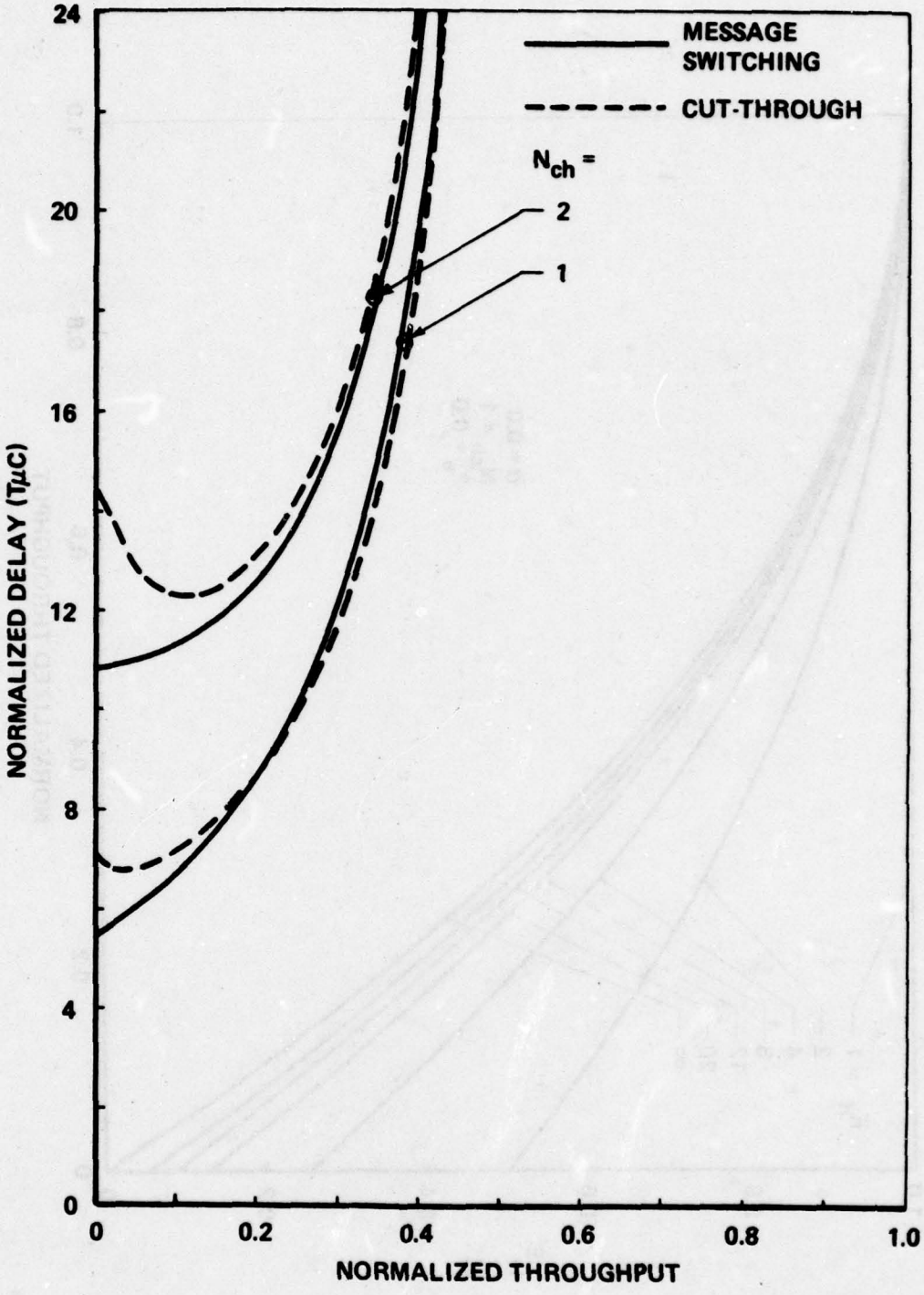


Fig. 2.7-g Delay Vs. Throughput

$\bar{n}_h = 3$, $P_e = 0.45$, $\alpha = 0.1$

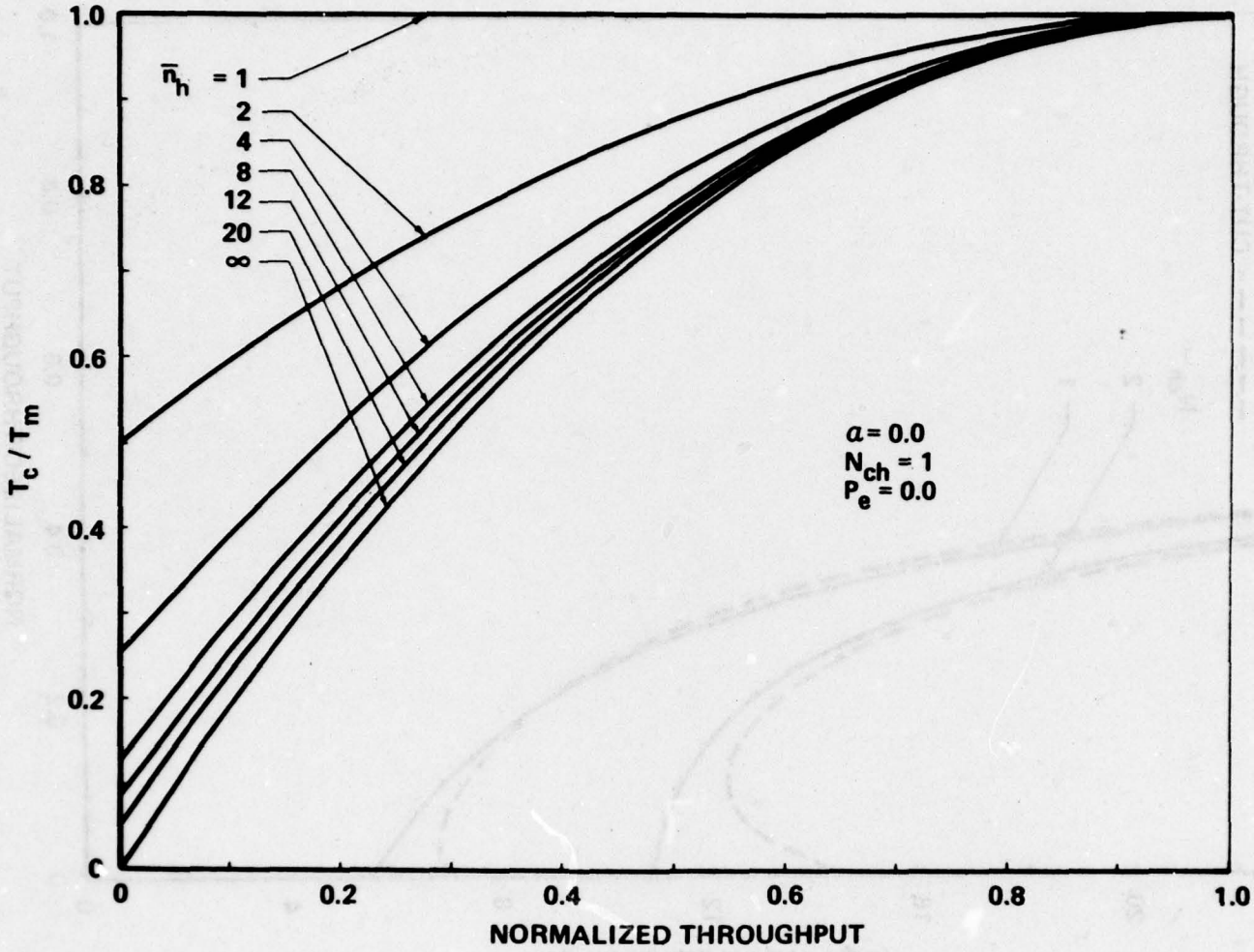


Fig. 2.8-a Ratio of the Delay in the Two Switching Systems.

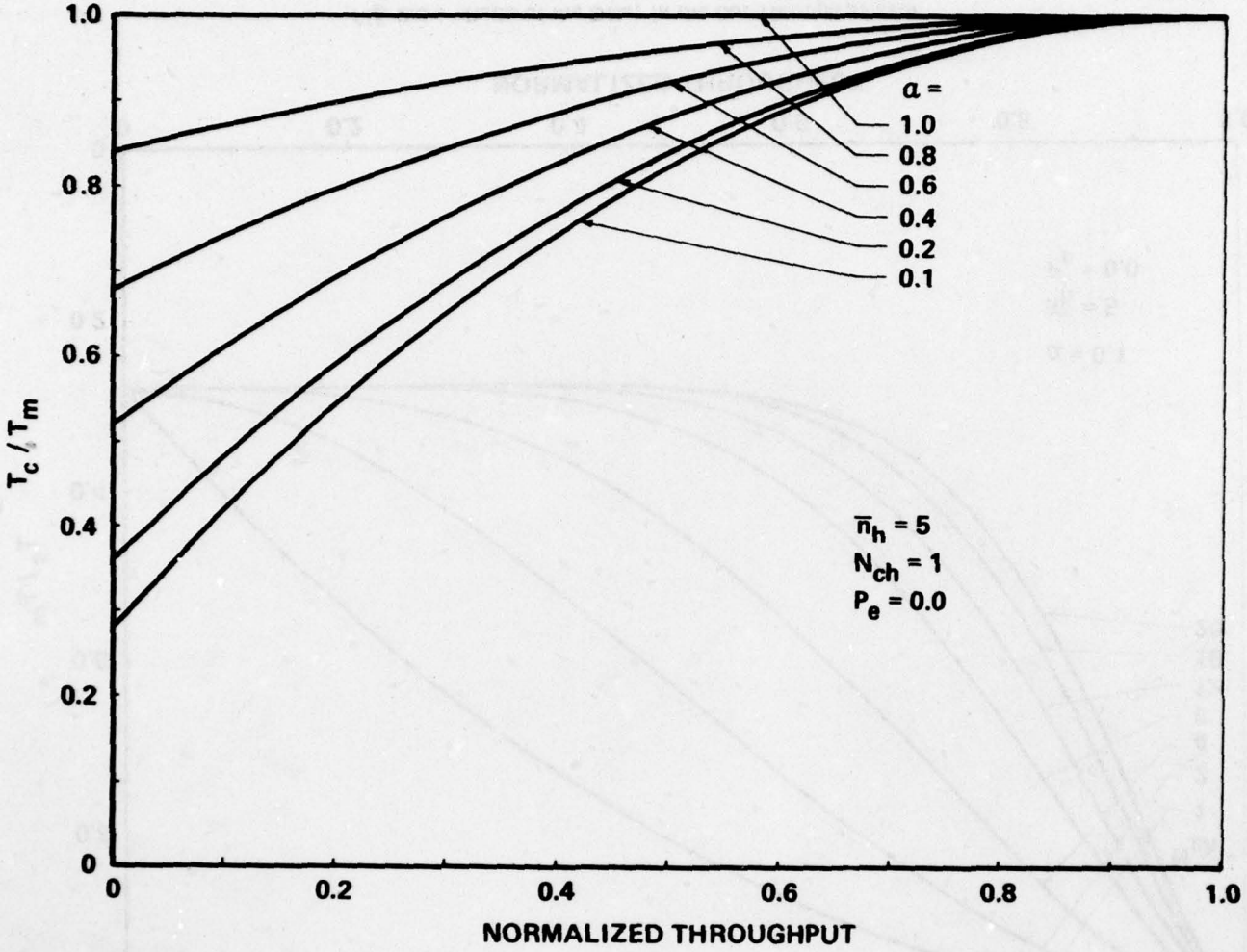


Fig. 2.8-b Ratio of the Delay in the Two Switching Systems.

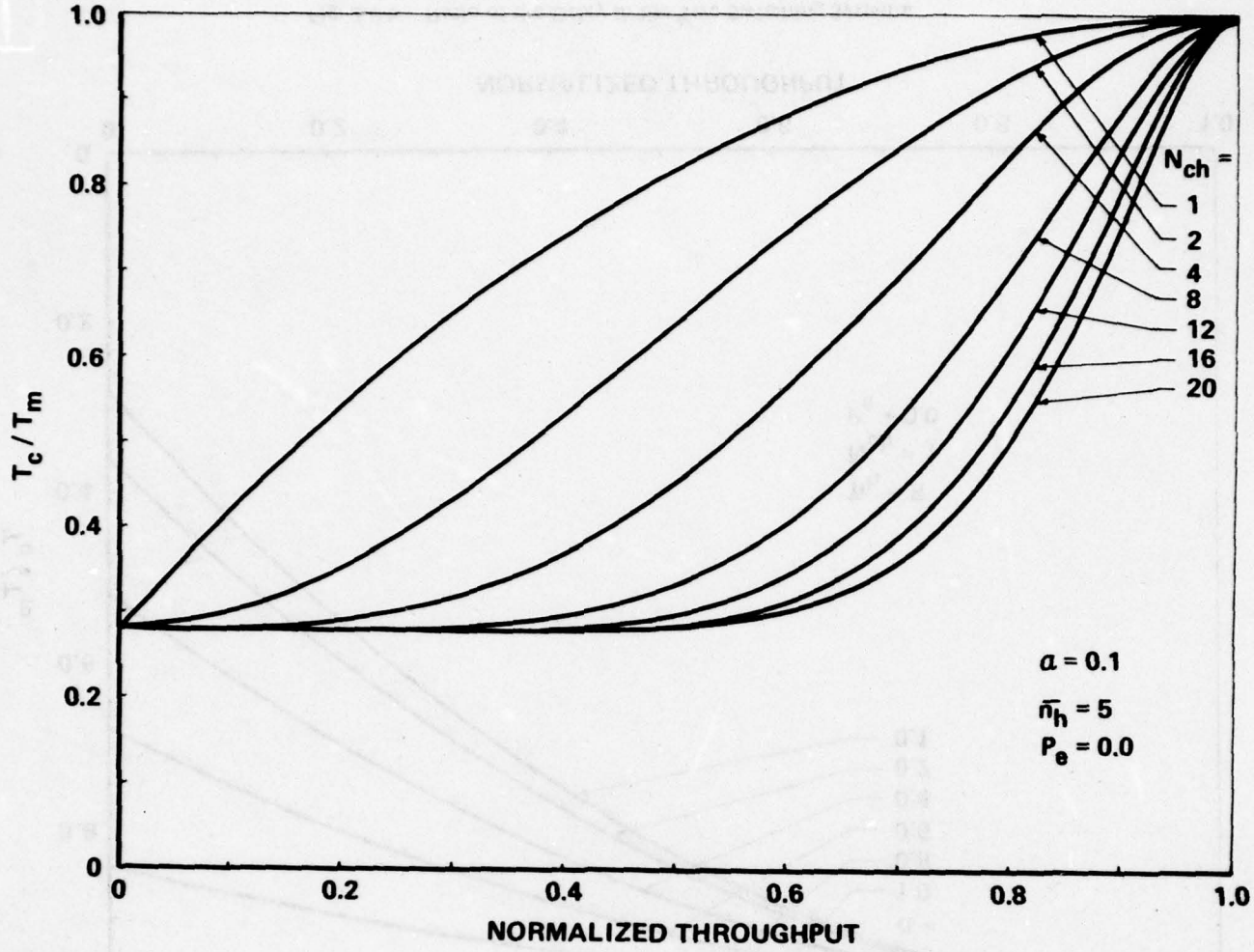


Fig. 2.8-c Ratio of the Delay in the Cut-Through System to the Delay in the Message Switching System.

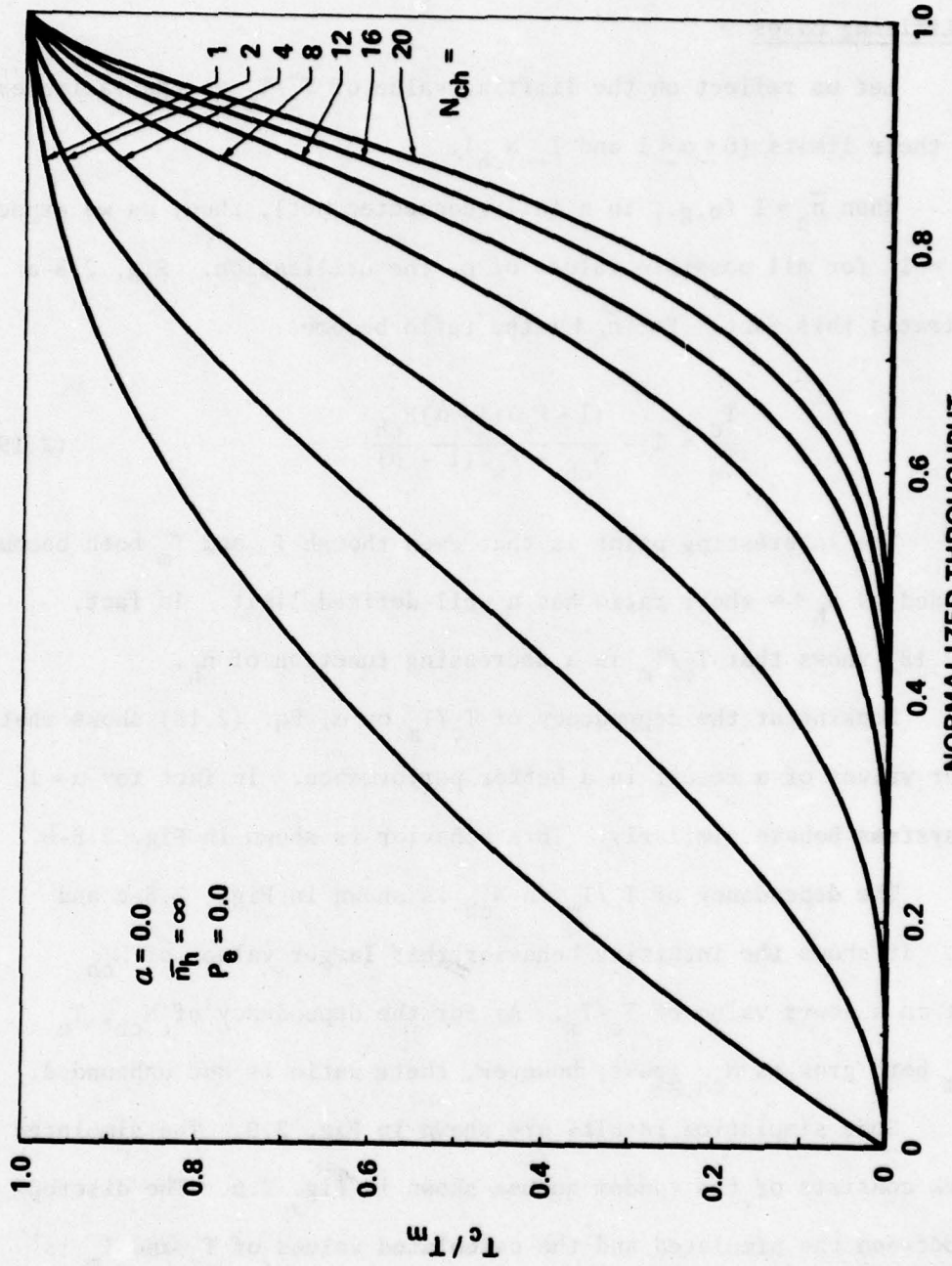


Fig. 2.8-d Ratio of the Delay in the Two Switching Systems.

switching the delay incurred at each intermediate node is at least t_0, the transmission time of a message over a single channel.

Some Limiting Cases

Let us reflect on the limiting value of T_c/T_m as the variables reach their limits ($0 \leq \alpha \leq 1$ and $1 \leq N_{ch}$).

When $\bar{n}_h = 1$ (e.g., in a fully connected net), then, as we expect, $T_c/T_m = 1$ for all possible values of ρ, the utilization. Fig. 2.8-a illustrates this fact. For $\bar{n}_h \uparrow \infty$ the ratio becomes

$$\frac{T_c}{T_m} = 1 - \frac{(1 - P_w)(1 - \alpha)N_{ch}}{N_{ch} + P_w/(1 - \rho)} \tag{2.19}$$

The interesting point is that even though T_c and T_m both become unbounded as $\bar{n}_h \uparrow \infty$ their ratio has a well-defined limit. In fact, Eq. (2.18) shows that T_c/T_m is a decreasing function of $\bar{n}_h$.

Looking at the dependency of T_c/T_m on α, Eq. (2.18) shows that smaller values of α result in a better performance. In fact for $\alpha = 1$, both systems behave similarly. This behavior is shown in Fig. 2.8-b.

The dependency of T_c/T_m on N_{ch} is shown in Figs. 2.8-c and 2.8-d. It shows the intuitive behavior that larger values of N_{ch} result in a lower value of T_c/T_m. As for the dependency of N_{ch}, T_c and T_m both grow as N_{ch} grows; however, their ratio is not unbounded.

Some simulation results are shown in Fig. 2.9. The simulated network consists of the tandem queues shown in Fig. 2.6. The discrepancy between the simulated and the calculated values of T_c and T_m is the result of the topology of the network and of the message length distribution. The analytic results are for exponentially distributed

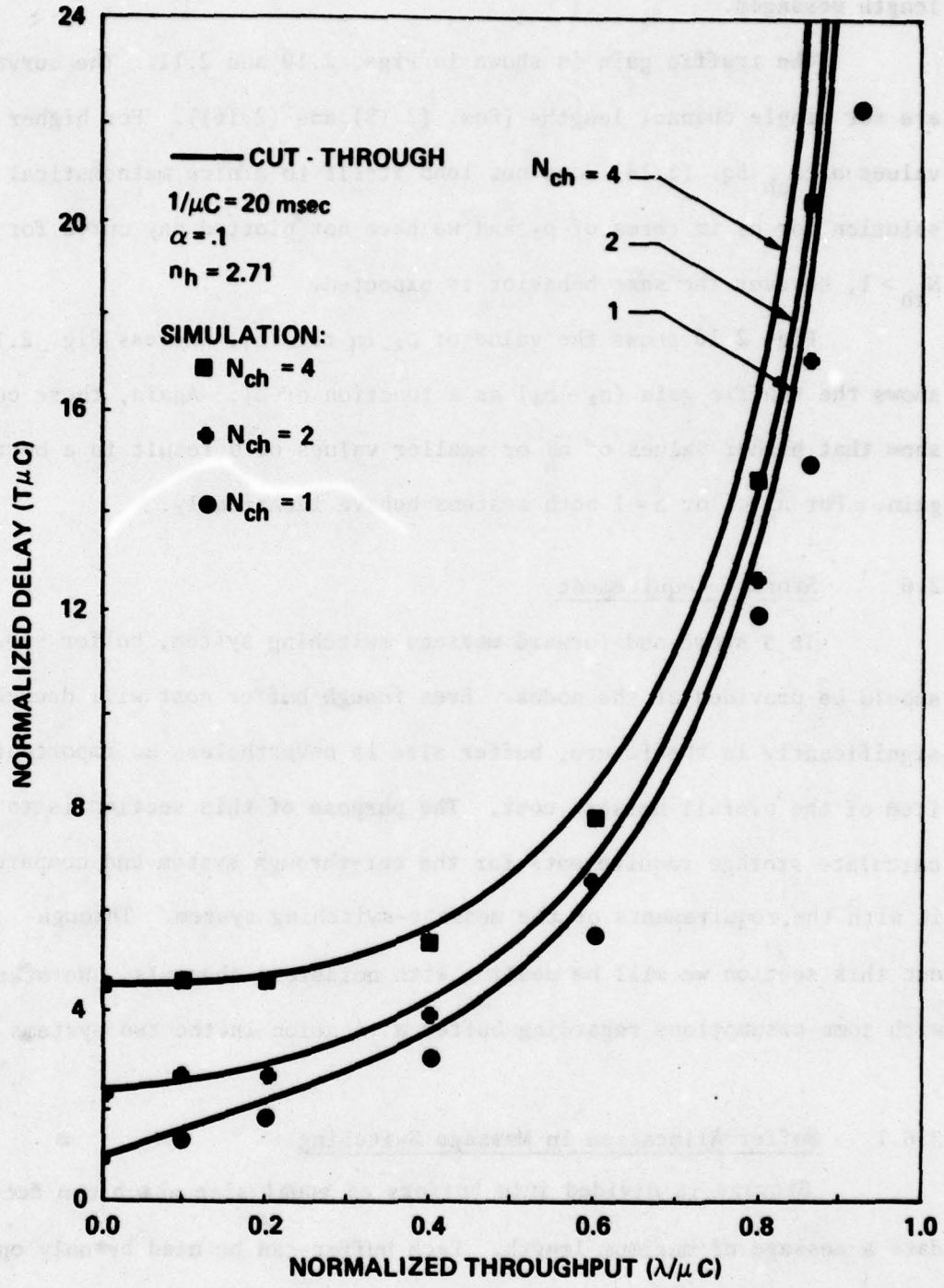


Fig. 2-9 Some Simulation Results.

message lengths, whereas in the simulation program we have used fixed length messages.

The traffic gain is shown in Figs. 2.10 and 2.11. The curves are for single channel lengths (Eqs. (2.15) and (2.16)). For higher values of N_{ch} Eq. (2.14) does not lend itself to a nice mathematical solution for ρ_2 in terms of ρ_1 and we have not plotted any curve for $N_{ch} > 1$; however the same behavior is expected.

Fig. 2.10 shows the value of ρ_2 in term ρ_1, whereas Fig. 2.11 shows the traffic gain $(\rho_2 - \rho_1)$ as a function of ρ_1. Again, these curves show that higher values of $\bar{n}_h$ or smaller values of α result in a better gain. For $\bar{n}_h = 1$ or $\alpha = 1$ both systems behave identically.

2.6 Storage Requirement

In a store-and-forward message-switching system, buffer storage should be provided at the nodes. Even though buffer cost will decrease significantly in the future, buffer size is nevertheless an important item of the overall network cost. The purpose of this section is to calculate storage requirements for the cut-through system and compare it with the requirements of the message-switching system. Throughout this section we will be dealing with noiseless channels. We start with some assumptions regarding buffer allocation in the two systems.

2.6.1 Buffer Allocation in Message Switching

Storage is divided into buffers of equal size which can accomodate a message of maximum length. Each buffer can be used by only one message at a time, even though it does not fill the entire buffer. Within a node, a message is never allocated more than one buffer. With

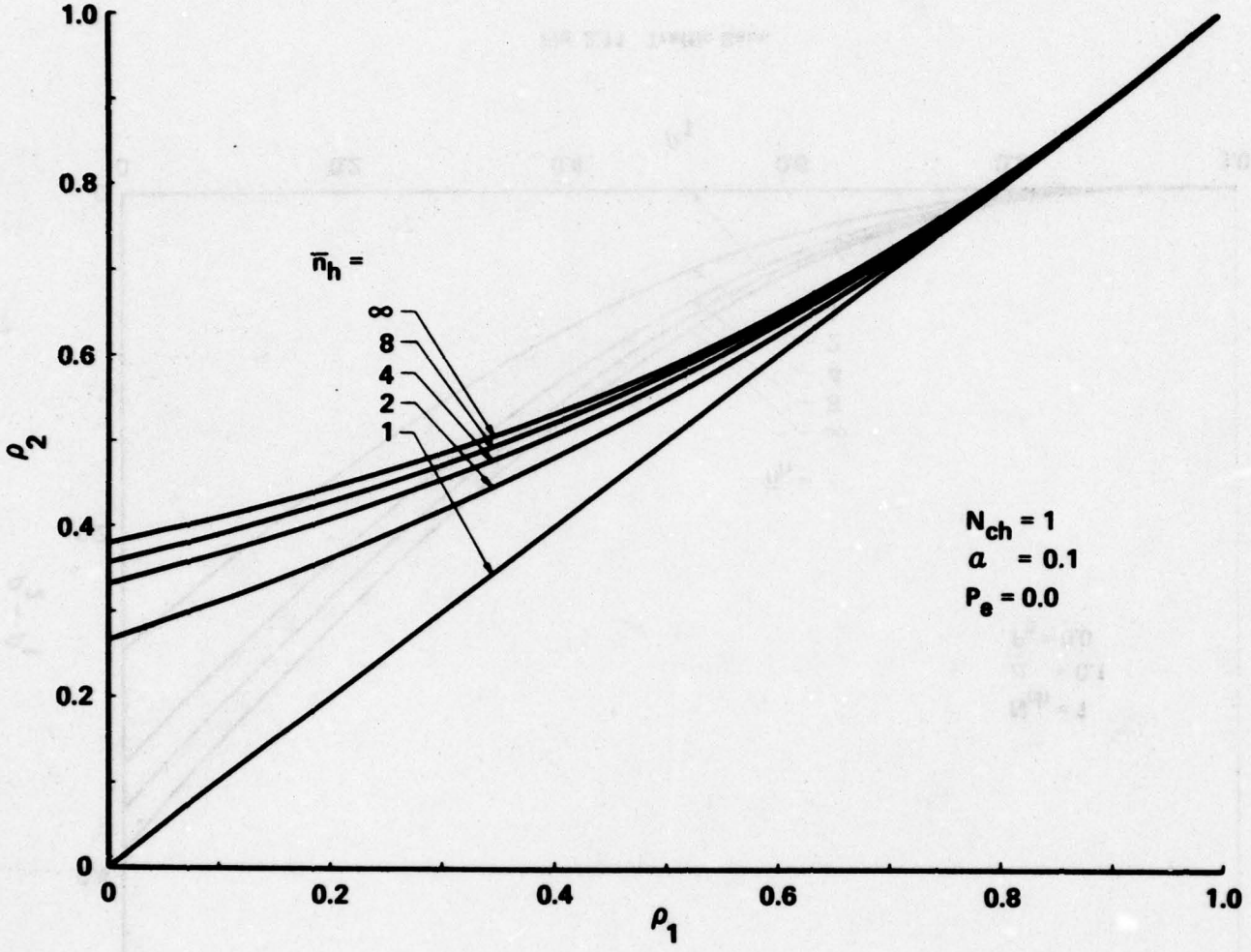


Fig. 2.10 Channel Utilization in the Two Switching Systems for Equal Delay.

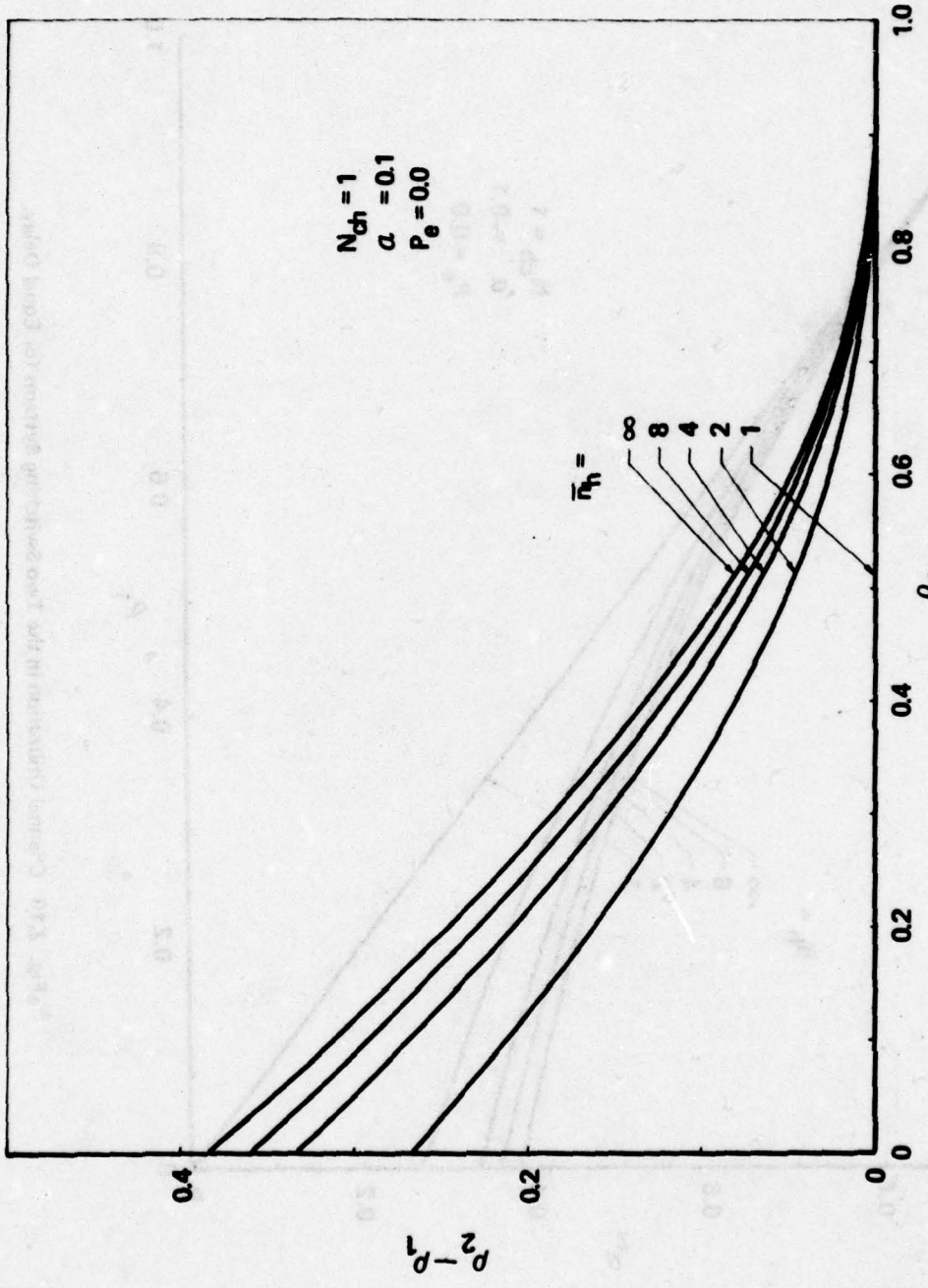


Fig. 2.11 Traffic Gain.

this buffer management, one might think that the assumption of exponentially distributed message length should be replaced with a truncated exponential distribution. However, measurements on existing networks, especially on the ARPANET [KLEI 74B], show that the average message length is relatively small (250 bits in the ARPANET). With a reasonably large buffer size, and considering the fact that the average message length is much smaller than the buffer size, and that large messages occur very infrequently, we are justified in keeping our exponential message length assumption. Regarding the messages which are being transmitted, we assume that when the first bit of a message arrives in a node, a full buffer is allocated to the message. This means that all messages in transmission use two buffers simultaneously. We also assume that acknowledgement is instantaneous. This means at the moment a message is completely received at a node, its source buffer is relinquished.

2.6.2 Buffer Allocation in Cut-Through Switching

There are two types of buffers in this system, message buffers and header buffers (with lengths ℓ_m and ℓ_h). A header buffer is used by a message at a node which it is cutting through; it is allocated when the first bit of a message arrives at a node. After the reception of the header, if it is recognized that the selected outgoing channel is free, no further storage is allocated to this message and it starts being transmitted out. If, however, the outgoing channel is busy, then the message must be completely received in the node and (the remainder of) a message buffer is allocated to it. (This happens at a final node, i.e., the destination node or any intermediate node where a cut is not

possible.) We notice that a message may be using a number of header buffers simultaneously, but it can use at most two message buffers at one time. Regarding the acknowledgement, we assume that a copy of a message is kept at the source node, or at any intermediate node in which it is blocked (i.e., can not make a cut) until it is completely received at a succeeding final node. If this final node is the destination node then, after completion of the transmission, both message buffers are released simultaneously. If, however, this final node is not the destination node then a copy of the message is kept in this final node and the other message buffer is released. Notice that after the reception of the first acknowledgement the copy of the message at the source node is dropped and its buffer is released.

Let

$\bar{S}_m^{(p)}$ = E[total amount of storage required at a path in message switching]

$\bar{S}_m$ = E[total amount of storage required at a node in message switching]

and let $\bar{S}_c^{(p)}$ and $\bar{S}_c$ be the same quantities for the cut-through system. As before, we use ℓ_m and ℓ_h for the message buffer and header buffer size. We also assume that

$$\alpha = \ell_h/\ell_m = \text{(avg. header length)/(avg. msg. length)}$$

2.6.3 Storage Requirement for Message Switching

At any instant a message which is being transmitted occupies two buffers. The other messages occupy only one buffer, so we have

$$\bar{S}_m^{(p)} = \{\bar{N}_m^{(p)} + \bar{n}_h(N_{ch}\rho)\}\ell_m \qquad (2.20)$$

where

$\bar{N}_m^{(p)}$ = E[number of messages on a path for message switching] = $T_m \lambda$

and $(N_{ch}\rho)$ is the average number of messages which are being transmitted at each node. From Eq. (2.20) we get

$$\bar{S}_m = \frac{\bar{S}_m^{(p)}}{\bar{n}_h}$$

$$\bar{S}_m = (\bar{N}_m + N_{ch}\rho)\ell_m \qquad (2.21)$$

where $\bar{N}_m$ is the average number of messages at each node which are on the same path that we are considering.

2.6.4 Storage Requirement for Cut-Through Switching

Let

δ = ceiling{$1/\alpha$}

= ceiling{(msg length)/(header length)}

(ceiling{x} $\triangleq$ smallest integer larger than or equal to x). Consider a message which is being transmitted along a path. Every time that the message makes a cut through a node, it is allocated a header buffer, i.e., the total amount of storage allocated to it is increased by one header buffer. This process continues until the number of the consecutive cuts exceeds δ, after which, for each new header buffer that the message is allocated, it releases one. Fig. 2.12-a shows this phenomenon. The solid lines show the amount of storage allocated to the message while it is being transmitted along the path, Fig. 2.12-b. At time t_1 the message starts transmission from node 1, the source node; because we have assumed propagation delay is negligible, at the same time a

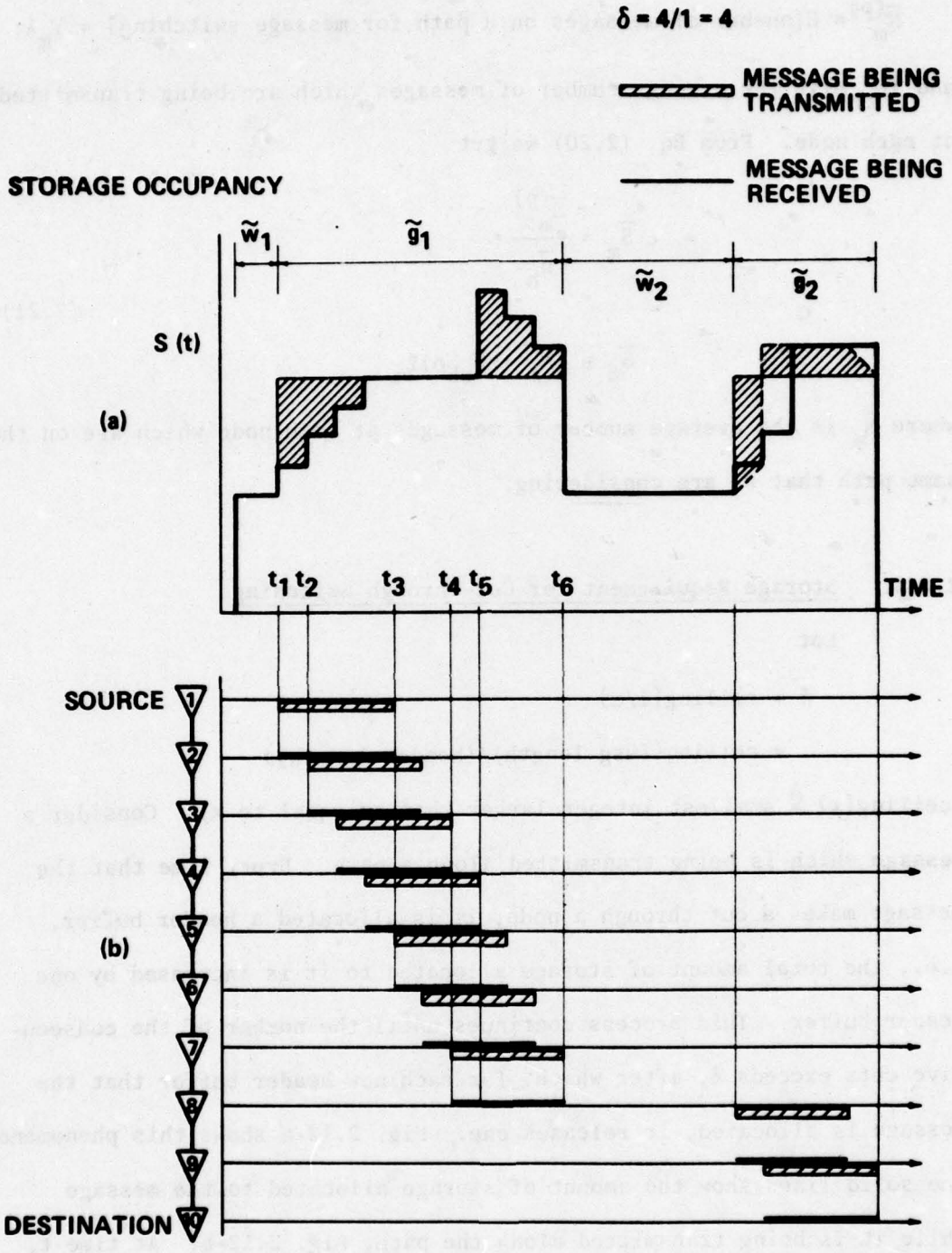


Fig. 2.12. Storage Occupancy in the Cut-Through System.

header buffer is allocated to the message at node 2 (because it can make a cut through this node). After t_h seconds the header is received completely at node 2, at which time it starts transmission towards node 3. By assumption, because the message can make a cut through node 3, another header buffer is allocated to it instantaneously. This process continues and the amount of the allocated storage increases up to node 5; however, from node 6 on (at time t_3), the tail of the message starts leaving nodes 2, 3, ..., and for each new header buffer it occupies, it releases one; hence the total amount of storage allocated remains unchanged. At time t_4 the message arrives in node 8 and a header buffer is allocated to it (so, there is no change in the total amount of buffers allocated to this message on the path). But, after t_h seconds (at time t_5) when the header is received, it is recognized that the selected outgoing link is busy. So, the message is blocked and (the remainder of) a message buffer is allocated to it. As the tail of the message leaves the intermediate nodes, header buffers are released. When the message is received in full at node 8 (at time t_6), the copy of the message present at node 1 (the source node) is dropped and its buffer is released. The right part of Fig. 2.12-a shows the situation where the number of consecutive cuts is less than δ.

We define the storage occupancy function s(t) as follows:

s(t) = amount of storage allocated to a message at time t.

If we approximate the step function s(t), drawn in solid lines in Fig. 2.12-a, by the broken lines in the same figure, we can observe a very useful property. The shaded areas in this figure are equal. The interesting fact is that when the number of consecutive cuts is larger

than δ, there is no approximation involved (left part of Fig. 2.12-a). We use this property in calculating $\bar{S}_c^{(p)}$. To do so, we introduce additional notations $\tilde{g}_i$ and $\tilde{w}_i$ as follows:

$\tilde{g}_i$ = length of the i^{th} transmission period of a message.

$\tilde{w}_i$ = length of the i^{th} waiting period of a message.

Using the above observation, we have the following identities:

$$\int_{t_{\tilde{g}_i}}^{t_{\tilde{g}_i}+\tilde{g}_i} s(t)\, dt = 2\tilde{g}_i \ell_m$$

$$\int_{t_{\tilde{w}_i}}^{t_{\tilde{w}_i}+\tilde{w}_i} s(t)\, dt = \tilde{w}_i \ell_m$$

where $t_{\tilde{g}_i}$ is the time that the i^{th} transmission starts and $t_{\tilde{w}_i}$ is the time that the i^{th} waiting period starts.

Let W_c be the expected total amount of time that a packet spends waiting and T_c, as defined before, be the expected total network delay (waiting time plus transmission time). We have the following relationships:

$$W_c = E\left[\sum_i \tilde{w}_i\right]$$

$$T_c = E\left[\sum_i (\tilde{g}_i + \tilde{w}_i)\right]$$

and

Average Storage Requirement of one message =

$$\frac{E\left[\sum_i (2\tilde{g}_i \ell_m + \tilde{w}_i \ell_m)\right]}{E\left[\sum_i (\tilde{g}_i + \tilde{w}_i)\right]}$$

which can be reduced to

$$\frac{T_c \ell_m + (T_c - W_c)\ell_m}{T_c}$$

There are λT_c messages on a path, so we have

$$\bar{S}_c^{(p)} = \lambda T_c \frac{T_c \ell_m + (T_c - W_c)\ell_m}{T_c}$$

or

$$\bar{S}_c^{(p)} = \lambda[T_c + (T_c - W_c)]\ell_m \tag{2.22}$$

where T_c is given by Eq. (2.7). Our goal is to find the quantity $(T_c - W_c)$.

Let

$$T_0 = T_c - W_c = E[\text{total time that any part of a message is being transmitted on any channel}]$$

and

$$\bar{n}_b = E[\text{number of times that a packet does not enter a free node}]$$
$$= (\bar{n}_h - 1)P_w$$

Then we have

Proposition 2.1

The value of T_0 is given by

$$T_0 = t_0 + (\bar{n}_h - 1)t_h + (t_0 - t_h)\bar{n}_b \tag{2.23}$$

Proof

Each time that the message makes a cut, it spends t_h seconds for transmission (rather than t_0 seconds). If the message can cut through all of the intermediate nodes, then its total transmission time will be

$$t_0 + (\bar{n}_h - 1)t_h$$

However, at each node where the message is blocked, instead of t_h, t_0 seconds are spent for transmission. On the average $\bar{n}_b$ times a packet is blocked, causing $(t_0 - t_h)\bar{n}_b$ seconds extra transmission time. Adding the two terms, we get Eq. (2.23).

Q.E.D.

After some algebra, and recalling that $\alpha = t_h/t_0$, we reduce Eq. (2.23) to

$$T_0 = [1 + (n_h - 1)\alpha + (1 - \alpha)(\bar{n}_h - 1)P_w]t_0 \qquad (2.24)$$

for the special cases we have

$$T_0 = \bar{n}_h t_0 \qquad \text{when } P_w = 1 \qquad (2.25)$$

$$T_0 = t_0 + (\bar{n}_h - 1)t_h \qquad \text{when } P_w = 0 \qquad (2.26)$$

In Eq. (2.25) T_0 is the total transmission time when no cut is possible (i.e., in message switching), while in Eq. (2.26) T_0 is the transmission time when all of the intermediate nodes are cut through.

Combining Eqs. (2.22) and (2.24) and replacing T_c by its value from Eq. (2.7), we get

$$\bar{S}_c^{(p)} = \{\bar{n}_h\bar{N}_m + N_{ch}\rho[\bar{n}_h - 2(\bar{n}_h - 1)(1 - \alpha)(1 - P_w)]\}\ell_m \qquad (2.27)$$

where $\bar{N}_m$ is the average number of messages at a node in a message switching system. For the average buffer requirement at each node we have the following expression

$$\bar{S}_c = \bar{S}_c^{(p)}/\bar{n}_h$$

$$= \left\{ \bar{N}_m + N_{ch}\rho\left[1 - 2\,\frac{\bar{n}_h - 1}{\bar{n}_h}\,(1-\alpha)(1-P_w)\right] \right\} \ell_m \qquad (2.28)$$

Note that if $P_w = 1$, then $\bar{S}_c^{(p)} = \bar{S}_m^{(p)}$.

To compare storage requirements of the two switching methods we study the behavior of the ratio $\theta = \bar{S}_c/\bar{S}_m$

$$\theta = \frac{\bar{N}_m + N_{ch}\rho\left[1 - 2\,\frac{\bar{n}_h - 1}{\bar{n}_h}\,(1-\alpha)(1-P_w)\right]}{\bar{N}_m + N_{ch}\rho} \qquad (2.29)$$

From Eq. (2.29) it is clear that $\bar{S}_c \leq \bar{S}_m$, and equality holds only when $P_w = 1$ or $\alpha = 1$. This shows that the performance of the cut-through system, in terms of storage requirement, is always better than the message-switching system.

The ratio θ is a decreasing function of N_{ch}, the number of channels. For $N_{ch} \uparrow \infty$ we have the limiting value of θ given below:

$$\lim_{N_{ch}\uparrow\infty}(\theta) = 1 - \frac{\bar{n}_h - 1}{\bar{n}_h}\,(1-\alpha) = \frac{1 + (\bar{n}_h - 1)\alpha}{\bar{n}_h} \qquad (2.30)$$

which is independent of the utilization factor ρ. Fig. 2.13 shows the behavior of θ as a function of N_{ch} for different values of ρ. Notice that the true limiting value should be

$$\frac{2 + (\bar{n}_h - 1)\alpha}{2\bar{n}_h} = \frac{1 + 0.5(\bar{n}_h - 1)\alpha}{\bar{n}_h}$$

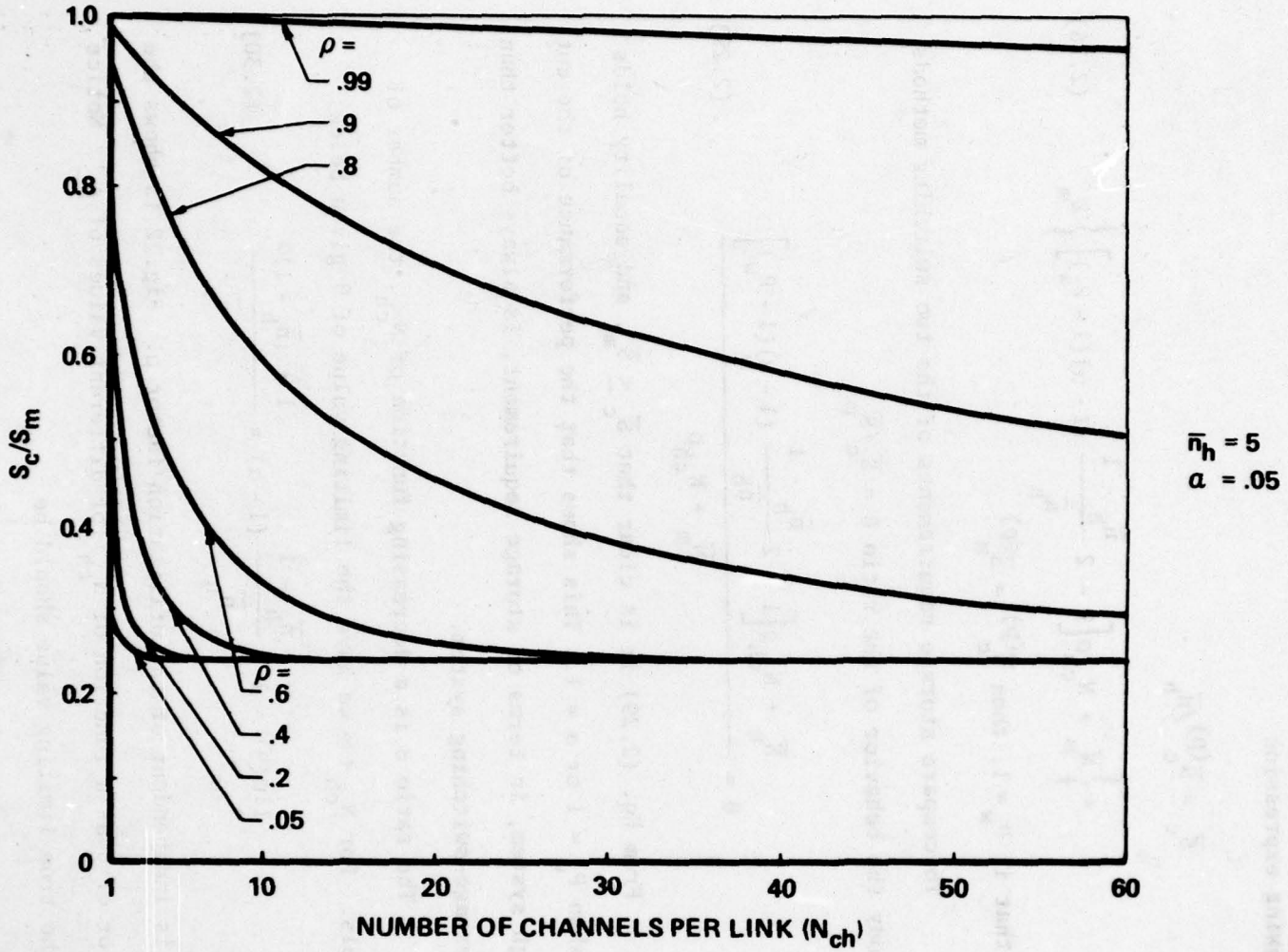


Fig. 2.13 Comparision of Storage Requirement in the Cut-Through System and the Message Switching System.

which can be found by considering the fact that in the limit ($N_{ch} \uparrow \infty$) there is no waiting message in the system ($P_w = 0$). In the case of cut-through switching, all of the messages are cutting through; and in the case of message switching, all of the messages in the system are in transmission. The discrepancy is the result of our approximation in calculating $\overline{s}_c^{(p)}$ (see Figs. 2.12-a and b).

The limiting value found by Eq. (2.30) is 0.24, which should be compared with the true value of 0.22.

2.7 A Design Problem

So far we have been dealing with the analysis of the cut-through switching technique. Our study has shown that this technique is, in fact, a satisfactory solution to the switching problem in communication nets. It takes advantage of the good properties of message switching by simultaneously using multiple links along a path. It exhibits nice properties of circuit switching by not incurring unnecessary intermediate nodal delay for unnecessary store-and-forward.

In this section we look at a design problem for this switching technique. Several other formulations of design problems for packet-switching systems can be found in the literature (e.g., [KLEI 64], [GERL 73A] and [KAMO 76B]; along with these formulations, solutions have been proposed.

The selection of appropriate design criteria for communication nets, by its nature, is an involved task. The difficulty stems from the multiplicity of parameters, hence numerous choices, at the disposal of a network designer. It is not our intention to deal with the design

problem thoroughly since the literature is already rich with solutions for different versions of this problem. We assume that certain parameters, which are usually considered as design parameters, are selected. In particular, we assume that topology, channel capacity and routing matrix are already known to us. The basic design variable for us will be the number of channels per link. In Fig. 2.7 we have already shown the effect of this parameter on the network delay (which so far has been the main measure of performance for us). The significance of the number of channels becomes clearer when we talk about our other performance criteria.

For performance we consider three measures: network delay, throughput and the probability that a message can cut through all of the intermediate nodes along its path. The significance of the first two measures is clear; however, since the last criterion is somehow unusual, we choose to discuss it further.

We have noticed that in some situations the cut-through system resembles a circuit-switched system. This specific property is desirable for certain applications, for example, real-time systems or those applications which require permanent "physical" connections between remote stations. If a message can cut through all of the intermediate nodes on a path, then we can logically assume that there is a "physical" circuit between its source and destination nodes. This is the reason for the significance of the last measure. The probability of this event happening is easy to find. In fact we have

$$P_{nb} = \Pr[\text{no blocking on a path} \mid \text{path length} = n_h]$$
$$= \Pr[\text{making a cut through all of the intermediate nodes} \mid \text{path length} = n_h]$$
$$= (1 - P_w)^{n_h - 1} \tag{2.31}$$

where, as before, P_w is the probability that a message cannot make a cut through a node.

With these preliminaries, we now state our design problem:

Given: Network topology, channel capacity, routing matrix

Maximize: Throughput

with respect to the number of channels per link

Subject to: $\hat{T}_c \leq \hat{T}_{max}$ Delay constraint ($\hat{T}_c = T_c \mu C$)

$P_{nb} \geq \varepsilon$ Blocking probability constraint.

Notes:

(1) We have assumed that by specifying the topology we also specify the path length under consideration, hence we can use Eq. (2.31) for P_{nb}. The average probability of no blocking for the entire network can only be specified when we know the exact topology and routing matrix (a simple calculation shows that for the entire network, $P_{nb} = N[(1 - P_w)]/(1 - P_w)$, $N[\cdot]$ being the generating function of path lengths).

(2) As pointed out in previous sections, we assume noiseless channels, hence we can use Eq. (2.7) to find $\hat{T}_c$.

(3) The above formulation is by no means a unique formulation. In fact any one of the performance measures (ρ, $\hat{T}_c$ or P_{nb}) can be selected as the optimizing measure and the other two as

constraints; however in all of these cases, the results are essentially the same.

The design problem stated above can be solved by mathematical programming. However, reflection on the expression for P_w, Eq. (B.3), shows that all of the performance measures are complicated functions of the number of channels, N_{ch}. For this reason an exact solution is, if not impossible, extremely difficult to find; hence we will present a heuristic solution to this problem.

It is clear that for a given throughput, a smaller number of channels per link (N_{ch}) results in a lower delay; however, it also results in a lower P_{nb} (Figs. 2.14-a and 2.14-b). Therefore the number of channels has opposing effects on these two measures. The following algorithm tries to find the smallest number of channels per link which satisfies both constraints.

Algorithm 2.1

1 - Let $m = \text{floor}\ \{\hat{T}_{max}\}$

(floor $\{x\}$ = largest integer smaller or equal to x)

2 - Find ρ_1 such that $\hat{T}_c(m, \rho_1) = \hat{T}_{max}$

3 - Find $P_{nb}(m, \rho_1)$.

If $P_{nb}(m, \rho_1) < \varepsilon$ go to step 4

If $P_{nb}(m, \rho_1) > \varepsilon$ go to step 5

If $P_{nb}(m, \rho_1) = \varepsilon$ set $\rho = \rho_1$, go to step 6

4 - $(P_{nb}(m, \rho_1) < \varepsilon)$

Find ρ_2 such that $P_{nb}(m, \rho_2) \approx \varepsilon$. Set $\rho = \rho_2$, go to step 6.

5 - ($P_{nb}(m, \rho_1) > \varepsilon$)

If $m = 1$ set $\rho = \rho_1$, go to step 6.

If $m > 1$ set $m = m - 1$, go to step 2.

6 - Set $N_{ch} = m$. ρ has already been determined. From $\rho = \lambda/\mu C$ we can find either λ or μC by knowing the other.

Discussion:

We have used $\hat{T}_c(m, \rho)$ to indicate the normalized network delay when the number of channels is m and the utilization factor is ρ. The same notation holds for $P_{nb}(m, \rho)$.

In step 1, m is the largest number of channels which may be used. Any number greater than this will violate the delay constraint. ρ_1 in step 2 is the maximum utilization achievable when the number of channels is m. In step 4, ρ_1 is not achievable; however, m is the best choice. On the other hand, ρ_1 in step 5 is achievable but m may not be optimal (i.e., it may be possible to use a smaller m).

Algorithm 2.1 is still valid for the case when we have some restrictions on the choice of the number of channels, e.g., when the number of channels should be chosen from a finite set of numbers, $\{N_{ch_1}, N_{ch_2}, \ldots, N_{ch_K}\}$. If that is the case, only steps 1 and 5 have to be changed such that the next smaller channel number is selected.

Consider the following helpful example. For a given network we have $n_h = 3$, $P_e = 0$ and $\alpha = 0$. We also have the following choices for the number of channels per link: 1, 2, 3, 4, 8, 10, 15 and 20. The constraints are $\hat{T}_c < 10.5$ (normalized delay) and $P_{nb} > 0.85$. Figs. 2.14-a and 2.14-b show the delay and P_{nb} curves for this network. The curves are plotted for different numbers of channels per link. We carry out

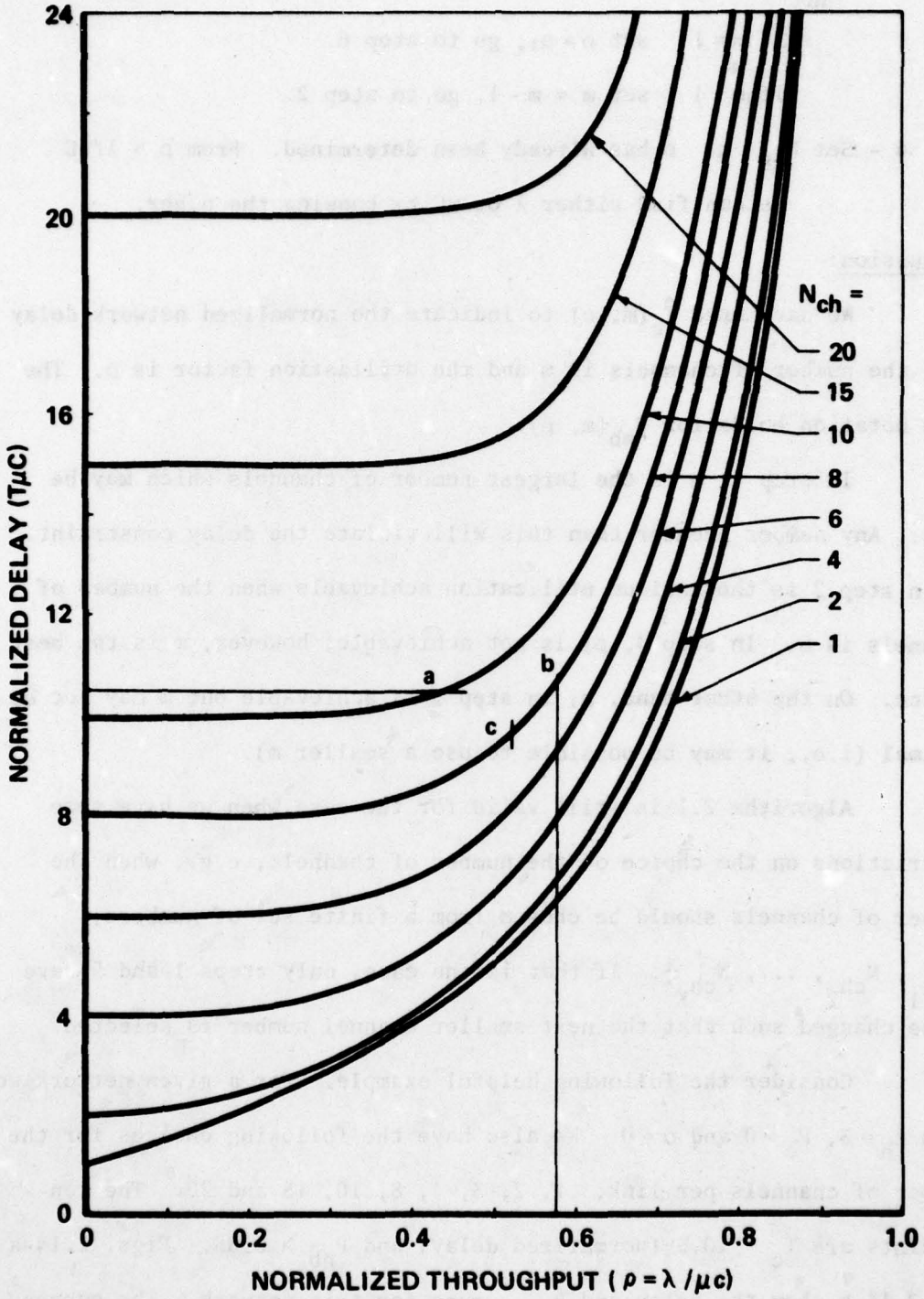


Fig. 2.14-a Delay Vs. Throughput for the Cut-Through System.

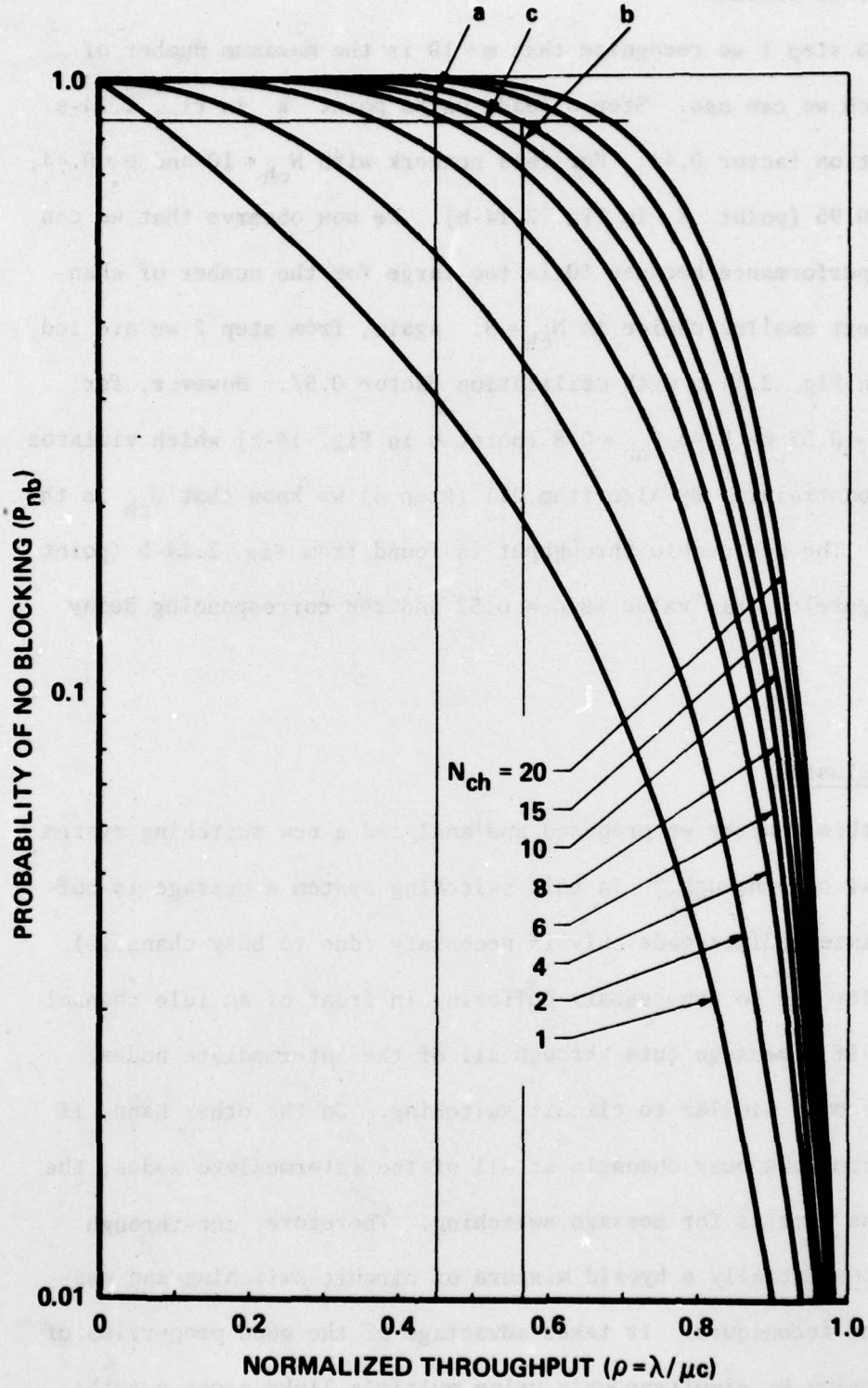


Fig. 2.14-b Probability of No Blocking as a Function of Utilization.

algorithm 2.1 as follows.

From step 1 we recognize that m = 10 is the maximum number of channels which we can use. Step 2 leads us to point a in Fig. 2.14-a with utilization factor 0.44. For this network with $N_{ch} = 10$ and $\rho = 0.44$, P_{nb} will be 0.96 (point a in Fig. 2.14-b). We now observe that we can improve the performance because 10 is too large for the number of channels. The next smaller choice is $N_{ch} = 8$. Again, from step 2 we are led to point b in Fig. 2.14-a with utilization factor 0.57. However, for $N_{ch} = 8$ and $\rho = 0.57$ we have $P_{nb} = 0.8$ (point b in Fig. 14-b) which violates the second constraint. By Algorithm 2.1 (step 4) we know that N_{ch} is the best choice. The achievable throughput is found from Fig. 2.14-b (point c on this figure). This value is $\rho = 0.52$ and the corresponding delay is 9.

2.8 Conclusion

In this chapter we proposed and analyzed a new switching system named "virtual cut-through." In this switching system a message is buffered in an intermediate node only if necessary (due to busy channels), hence the delay due to unnecessary buffering in front of an idle channel is avoided. If a message cuts through all of the intermediate nodes, the system becomes similar to circuit switching. On the other hand, if a message encounters busy channels at all of the intermediate nodes, the outcome is the same as for message switching. Therefore, cut-through switching is essentially a hybrid mixture of circuit-switching and message-switching techniques. It takes advantage of the good properties of message switching by simultaneously using multiple links along a path;

meanwhile, it exhibits desirable properties of circuit switching, by not incurring unnecessary intermediate node delay for store-and-forward.

Our analysis showed that cut-through switching is superior to message switching from at least three points of view: network delay, traffic gain and buffer storage requirement. In general, at the same traffic level, the network delay in the cut-through switching system is less than in message switching. Our analysis in Section 2.3 showed that only when the channel error rate is too high may the message switching delay become less than the cut-through system delay. In Section 2.4 we showed that at the same network delay, cut-through switching can carry more throughput than message switching, and in Section 2.6 we demonstrated that the storage requirement in cut-through switching is less than in message switching. When a message can cut through all of the intermediate nodes, the network path becomes similar to a physical connection between two remote stations. This property is desirable for certain applications. In Section 2.7 we developed an algorithm which can maximize the throughput while guaranteeing certain probability that messages can cut through all of the intermediate nodes on a path.

Our study in this chapter showed that cut-through switching is a good solution to the switching problem in a communication network. In Chapter 3 we will show that in many cases the performance of cut-through switching is even better than that of circuit switching.

We did not study the physical implementation of this system, and we also did not discuss the protocol design for this switching technique. These areas require further research.

CHAPTER 3

A TRADEOFF STUDY OF SWITCHING SYSTEMS

The previous chapter demonstrated that depending on the traffic of a data communication network, considerable delay and storage reduction can be achieved by using the virtual cut-through switching (CTS) technique. Throughout that chapter we compared performance of CTS with message switching (MS) and our study showed that in terms of network delay, CTS never performs worse than MS. In this chapter we are interested in extending the scope of our comparison to include circuit switching (CS).

A tradeoff study of switching systems involves consideration of a wide range of issues; however, in most cases cost and delay are by far the two major criteria. Network cost consists of installation and maintenance cost (from the owner's point of view) and service cost (from the users' point of view). Consideration of cost requires either specializing the study toward a specific system or strong assumptions regarding the physical structure and hardware costs of the switching nodes and the communication media. With the rapid advances of today's technology, with a trend toward lower cost and better performance, any assumption on (even relative) cost of hardware may soon become obsolete. For the above reasons we choose not to base our comparison study on cost and consider the network delay as the only comparison criterion.

In the last chapter, analytic models were developed to predict the throughput-delay profile of MS and CTS. In this chapter we first

formulate a network model for delay in a circuit-switched system; however, no simple solution to the transform equation which results from this model has been found. Next, we generalize the independence assumption [KLEI 64] and develop an analytic model for the delay of a communication path between two users in a CS network. A somehow similar model was formulated by Port, et al. [PORT 71]. Although the model is based on a weakened assumption, it shows rather interesting behavior for CS. Having thus developed expressions for delay in CS, we compare the performance of the three switching systems.

Before giving the analysis, we review the literature.

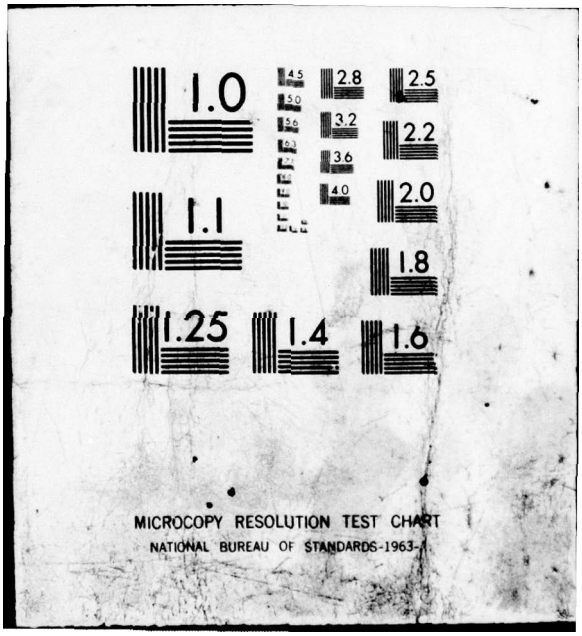
3.1 History and Related Work

With the birth of the first practical packet-switching network, the ARPANET [HEAR 70], [KLEI 70], [FRAN 70], [CARR 70] and [ROBE 70] in 1969, we witnessed an explosion of activities in the field of computer and data communication networks [KLEI 76B]. Numerous symposia and conferences have been devoted to this field and the interest is still growing. In fact, packet switching was an answer to many unsolved problems of data communications and it is not surprising to have witnessed an over-zealous enthusiastic claim in its favor during the early years after its birth. However, there was already a large investment in telephone networks, a technology based on circuit switching. After a period of silence, those in favor of circuit switching started challenging the efficiency of packet switching. The main argument in favor of CS was based on the intuitive fact that if there is a need for transmitting a long, continuous stream of data, then a leased line (or circuit switch)

AD-A077 404 CALIFORNIA UNIV LOS ANGELES DEPT OF COMPUTER SCIENCE F/G 17/2.1
ADVANCED TELEPROCESSING SYSTEMS.(U)
JUN 78 L KLEINROCK MDA903-77-C-0272
UNCLASSIFIED NL

2 OF 4
AD
A077404




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

communication makes good sense. Circuit-switched and telephone networks were well-studied systems and there was an enormous body of literature available for different aspects of their operations ([WILK 56], [SYSK 60] and the references therein); however, in all studies these systems are viewed as loss systems. This means, once a request for service (in this case a call) finds no available trunk channel, it is dropped from the system and is lost. However, a computer communication network is basically a delay system in which the two parties of a communication can wait, if necessary, for a connection to be established between them. In order to compare the performance of the two switching techniques on a concrete basis, there was a need to develop a delay model for CS. This was done by devising a signalling mechanism for CS, operating on a delay basis. Based on this model, some studies have been conducted in an attempt to compare these switching techniques [PORT 71], [CLOS 72A,B], [CLOW 73], [ITOH 73], [MIYA 75], [ROSN 76] and [KUMM 76A,B]. In [PORT 71] it was shown that the "signalling time" (which is different from total network delay) in a CS network for all the required tandem circuits on a path to simultaneously become free is often shorter than the waiting time in a series of high-speed tandem channels in a message-switched network. This paper was devoted to the issue of signalling delay and was one of the earliest attempts to challenge the superiority of packet switching on a quantitative basis. Later reports [CLOS 72A,B] are extensions of this work. In [ITOH 73] a detailed comparison based on the network delay between the switching schemes is made. This work confirmed the intuitive understanding that at higher traffic rates store-and-forward switching results in a better delay performance, whereas for

longer message lengths, CS is superior to the other switching technique. In [KUMM 76A,B] a comprehensive comparison study was done from two points of view: delay performance and usage cost. This study again confirmed the previous results.

In passing, we should point out that almost all of the previous works (except [PORT 71] have neglected to include the effect of channel reservation in their analytical modeling of CS. The fact is, channel reservation degrades the performance of CS severely in the sense that networks under CS operation saturate very fast. As we shall see shortly, an exact analysis of this phenomena is extremely difficult; however, by accepting some simplifying assumptions (which were first introduced in [PORT 71]) we are able to develop a more accurate analytic model than has been used in the studies so far.

The aim of the present study is to give a reasonably realistic and quantitative performance of the three switching systems, CTS, MS and CS. As we did in the last chapter, we do not differentiate between packet switching and message switching; we consider both methods as members of the larger class of store-and-forward switching systems.

3.2 A Delay Model for Circuit Switching

We use the following model for a circuit-switching system which operates as a "delay" system. This model is similar to the models used in the work referred to in the previous section.

In this model a message waits (rather than being lost) at the source node and sends a signal toward its destination (request-for-connection). While travelling node by node towards the destination node,

the signal reserves channels along the path. If at any intermediate node it cannot find a free channel, while holding the channels it has reserved so far, it waits for a channel to become free, at which time the signal reserves it and goes to the next node to repeat the same process. By the time the signal reaches the destination node, a path has been reserved between the source and the destination nodes. Figure 3.1 shows the structure of a node in a communication network (Fig. 3.2) in which the nodes are connected via two sets of channels, message channels and signalling channels. A message from outside the network arrives in box (or queue) A and sends a (request-for-connection) signal to establish a path between this node (S) and its destination node (D). Box B is a queue in which signals wait to reserve one of the message channels between nodes S and j (service facility C). Having reserved a channel, the signal joins queue D, the signalling queue, and waits for transmission over the signalling channels E to join node j. At node j the signal goes through the same process until it arrives at the destination node D. When the signal reaches its destination, its originating message is notified (through a reverse signalling process called request-for-transmission), at which time the message can start transmission. Note that (only) during the message transmission are all of the channels on the path used simultaneously, hence this last transmission is similar to a one-hop transmission. After the message transmission is completed, the reserved channels are released. We assume that no further signalling is required for releasing the reserved channels.

The above model is usually referred to as a "forward reservation-common signalling channel." Reservation of message channels can be done

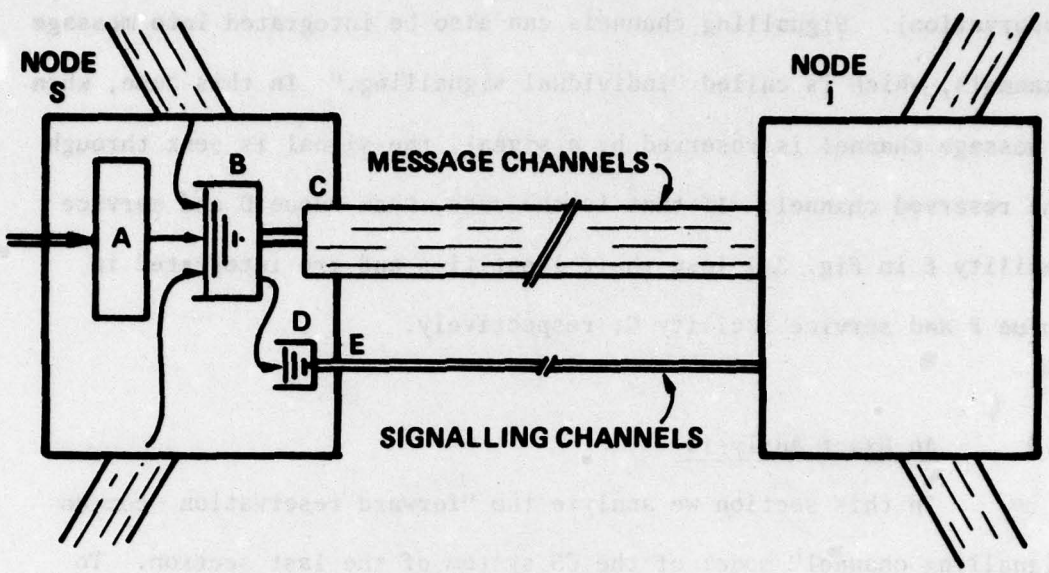


Figure 3.1 Structure of a Node.

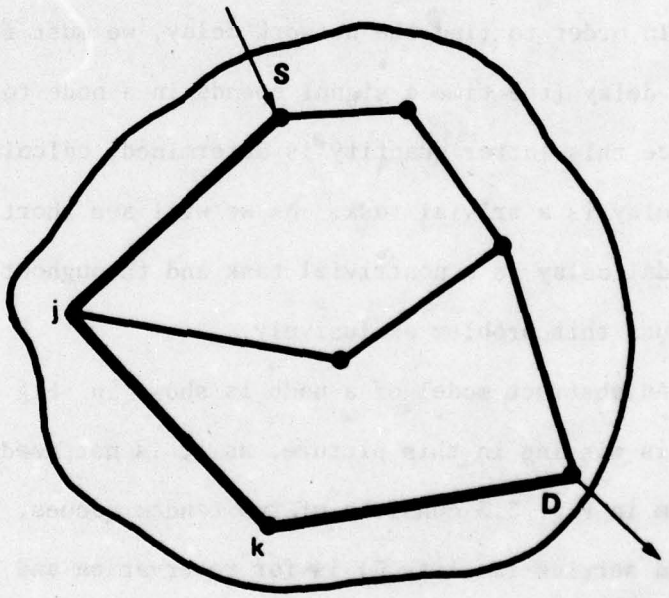


Figure 3.2 A Communication Network.

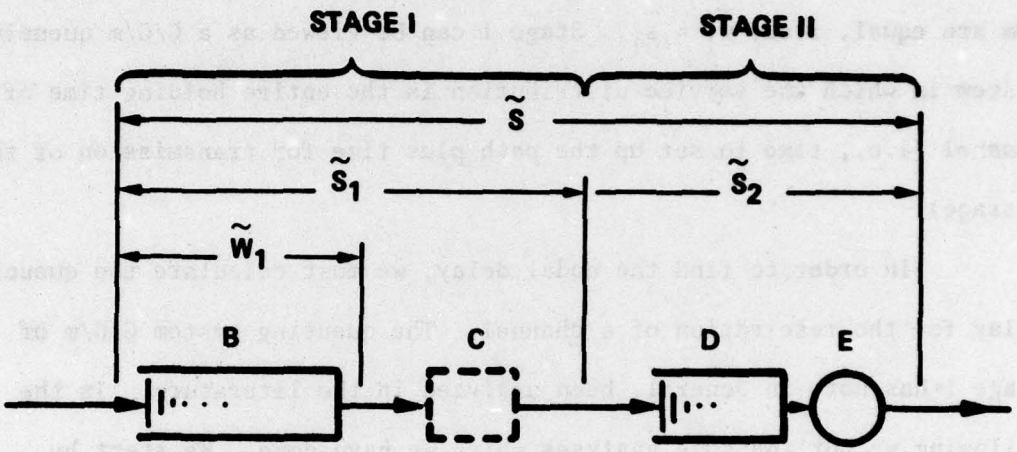
while the signal is coming back from the destination node (backward reservation). Signalling channels can also be integrated into message channels, which is called "individual signalling." In this case, when a message channel is reserved by a signal, the signal is sent through the reserved channel. If that is the case, then queue D and service facility E in Fig. 3.1 lose their identities and are integrated in queue B and service facility C, respectively.

3.3 An Exact Analysis

In this section we analyze the "forward reservation - common signalling channel" model of the CS system of the last section. To simplify the analysis we disregard the backward signalling process, i.e., we assume that when a path for a request is set up, the requesting message starts transmission immediately.

In order to find the network delay, we must first calculate the nodal delay (the time a signal spends in a node to reserve a channel.) Once this latter quantity is determined, calculation of the network delay is a trivial task. As we will see shortly, determination of the nodal delay is a nontrivial task and throughout this section we will discuss this problem exclusively.

An abstract model of a node is shown in Fig. 3.3. The waiting buffer A is missing in this picture, as it is not needed in our analysis. The system in Fig. 3.3 consists of two tandem queues. Stage I (waiting room B and service facility C) is for reservation and stage II (waiting room D and server E) for signalling. Service facility C is a fictitious server in the sense that when a signal reaches the head of queue B and



B – WAITING ROOM TO RESERVE A FREE CHANNEL
C – FICTITIOUS SERVICE FACILITY (MESSAGE CHANNELS)
D – WAITING ROOM TO TRANSMIT SIGNALS
E – SIGNAL CHANNELS

Figure 3.3 Abstract Structure of a Node.

reserves a channel, it is no longer delayed in server C and joins queue D directly. However, the channel reserved by this signal is held and cannot be used by others while it is establishing the rest of the path and during the transmission of its originating message. So, the waiting time $\tilde{w}_1$ and the total system time $\tilde{s}_1$ (waiting plus service) of this system are equal, i.e., $\tilde{w}_1 = \tilde{s}_1$. Stage I can be viewed as a G/G/m queueing system in which the service distribution is the entire holding time of a channel (i.e., time to set up the path plus time for transmission of the message).

In order to find the nodal delay, we must calculate the queueing delay for the reservation of a channel. The queueing system G/G/m of stage I has not, in general, been analyzed in the literature. In the following we outline some analyses which we have done. We start by making the following assumption.

Assumption 3.1

The queueing process at each node is independent of the other nodes and within each node the stochastic processes of stage I and stage II are independent of each other and the network is balanced (Definition 2.1).

With this (compound) assumption, all nodes behave identically and independently.

We now define some notation in addition to that of Chapter 2. In the following, a starred function denotes the Laplace transform of a distribution function. Each random variable is related to the Laplace transform of its distribution function by "<=>" and to its average value

by "⟺". A letter with a bar over the top indicates the average of a random variable.

$S_1^*(\cdot) \Longleftrightarrow \tilde{s}_1 \longleftrightarrow \bar{s}_1$	system time at stage I (reservation queue, Fig. 3.3)
$W_1^*(\cdot) \Longleftrightarrow \tilde{w}_1 \longleftrightarrow \bar{w}_1$	waiting time at stage I
$S_2^*(\cdot) \Longleftrightarrow \tilde{s}_2 \longleftrightarrow \bar{s}_2$	system time at stage II (signal transmission queue, Fig. 3.3)
$S^*(\cdot) \Longleftrightarrow \tilde{s} \longleftrightarrow \bar{s}$	total nodal delay. Notice $\tilde{s} = \tilde{s}_1 + \tilde{s}_2$, and by the assumption stated above $S^*(s) = W_1^*(s)S_2^*(s)$
$H^*(\cdot) \Longleftrightarrow \tilde{h} \longleftrightarrow \bar{h}$	holding time of a channel
$B_m^*(\cdot) \Longleftrightarrow \tilde{t}_0 \longleftrightarrow t_0$	transmission time of a message on a message channel
$B_h^*(\cdot) \Longleftrightarrow \tilde{t}_h \longleftrightarrow t_h$	transmission time of a signal on a signal channel
$N(z) \Longleftrightarrow \tilde{n}_h \longleftrightarrow \bar{n}_h$	number of nodes (or hops) on a path
$\tilde{n}_h^r \longleftrightarrow \bar{n}_h^r$	remaining number of nodes on a path from a randomly chosen node.

Consider a randomly chosen node on the path between two communicating source and destination nodes (say node j on the path between nodes S and D in Fig 3.2); by definition, there are $\tilde{n}_h^r$ nodes remaining on the path between node j and the destination node D. We wish to study the holding time of the channel between nodes j and k (the next node on the path between node j and D). After a channel is reserved between nodes j and k , the signal is transmitted to node k (taking $\tilde{s}_2$ units of time),

then it reserves the remaining $(\tilde{n}_h^r - 1)$ channels on the path, at which time the message can begin transmission (taking $\tilde{t}_0$ units of time; recall that the backward signalling is disregarded). The channel between nodes j and k is in use during all of the above process. Based on what we said, we have the following expression for the holding time h:

$$\tilde{h} = \tilde{s}_2 + (\tilde{n}_h^r - 1)\tilde{s} + \tilde{t}_0 \tag{3.1}$$

Using the residual life argument [KLEI 75], we find the distribution of $\tilde{n}_h^r$ as

$$\Pr\left[\tilde{n}_h^r = n\right] = \frac{1}{\bar{n}_h} \sum_{k \geq n} \Pr[\tilde{n}_h = k] \tag{3.2}$$

By the virtue of the assumptions stated before, we can find the conditional Laplace transform for the distribution of $\tilde{h}$ as follows:

$$H^*(s \mid \tilde{n}_h^r = n) = S^*_2(s)[S^*(s)]^{n-1}B^*_m(s) \tag{3.3}$$

Using Eq. (3.2) to remove the condition on $\tilde{n}_h^r$, we have

$$H^*(s) = \sum_{n \geq 1} H^*(s \mid \tilde{n}_h^r = n)\Pr[\tilde{n}_h^r = n]$$

$$= \sum_{n \geq 1} S^*_2(s)[S^*(s)]^{n-1}B^*_m(s) \frac{1}{\bar{n}_h} \sum_{k \geq n} \Pr[\tilde{n}_h = k]$$

and after some algebra we get

$$H^*(s) = \frac{B^*_m(s)S^*_2(s)}{\bar{n}_h[1 - S^*(s)]} \{1 - N[S^*(s)]\}$$

$$= \frac{B^*_m(s)S^*_2(s)}{\bar{n}_h[1 - S^*_1(s)S^*_2(s)]} \{1 - N[S^*_1(s)S^*_2(s)]\} \tag{3.4}$$

We can now find the average holding time as follows

$$\bar{h} = \frac{dH^*(s)}{ds}\Bigg|_{s=0}$$

or

$$\bar{h} = t_0 + \bar{s}_2 + \frac{\overline{n_h^2} - \bar{n}_h}{2\bar{n}_h}\bar{s} \tag{3.5}$$

where $\overline{n_h^2}$ is the second moment of the number of hops on a path. For the special case where all of the messages go through the same number of hops, Eq. (3.5) is reduced to

$$\bar{h} = t_0 + \bar{s}_2 + \frac{\bar{n}_h - 1}{2}\bar{s} \tag{3.6}$$

However, $\bar{s} = \bar{s}_1 + \bar{s}_2 = \bar{w}_1 + \bar{s}_2$, so we have

$$\bar{h} = t_0 + \frac{\bar{n}_h + 1}{2}\bar{s}_2 + \frac{\bar{n}_h - 1}{2}\bar{w}_1 \tag{3.7}$$

Eqs. (3.4) through (3.7) give us relationships in the G/G/m queueing system of the reservation operation.

In Eq. (3.4), $B_m^*(\cdot)$ is known and $N(\cdot)$ can be calculated from the routing matrix (if the routing is deterministic) and the topology of the network; morever, $S_2^*(s)$ can be found by a reasonable assumption for the operation of the signalling channels. Hence Eq. (3.4) gives us a relationship between the service time (or the holding time) $\tilde{h}$ and the waiting time $\tilde{w}_1$ of the G/G/m queueing system under study. When the number of hops for all paths is the same, Eq. (3.7) gives a relationship between the average values of these quantities. In order to find $\bar{h}$ and $\bar{w}_1$ explicitly, we need another relationship between these two variables. So far, we have not been successful in finding such a relationship.

To verify the accuracy of Eq. (3.7), we simulated the system. The topology of the simulated network is shown in Fig. 3.4. This is essentially similar to the topology which we used for the simulation of Chapter 2 (Fig. 2.6), with the difference that there are 30 nodes on the path. Message trunk capacity is 50 kbps which is divided into a fixed number of channels (1, 2 and 4 in the simulation experiments). There is only one signal channel between two adjacent nodes with 8 kbps capacity and the input processes from the external sources are exponential. Messages and signals are of constant lengths; 1000 bits and 100 bits, respectively. We used values of $\bar{s}_2$ and $\bar{w}_1$ from the simulation and used Eq. (3.7) to calculate the average holding time. This calculated value was compared with the average holding time obtained from the simulation. Fig. 3.5 shows the result of this comparison. In this figure when a curve does not continue to the right beyond a certain value of λ, the input rate, it indicates that the network is saturated. The useful utilization of links ($\lambda/\mu C$) is also shown. It is seen that the network saturates at relatively low input rates, a result of channel reservation. It can also be seen that the network capacity increases as we increase the number of message channels (with the same trunk capacity). In a later section of this chapter we will show this phenomenon more explicitly. Fig. 3.5 shows that the values of $\bar{h}$ and $\bar{w}_1$ obtained from the simulation fit Eq. (3.7) very closely. This match is very encouraging; however, a complete analytic treatment is still lacking.

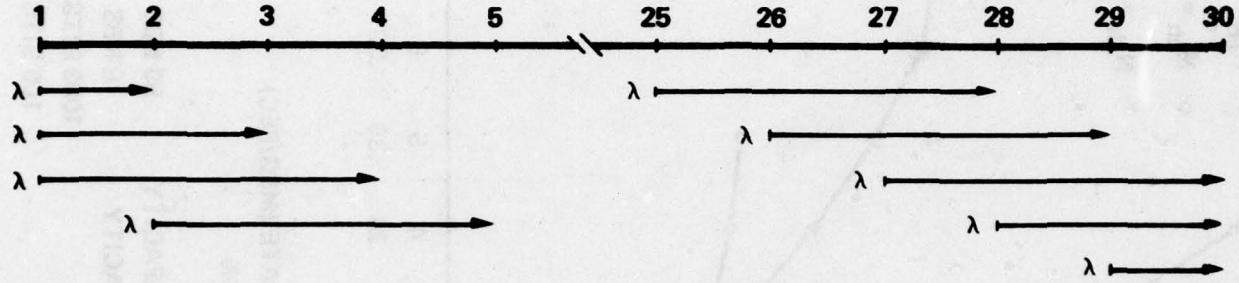
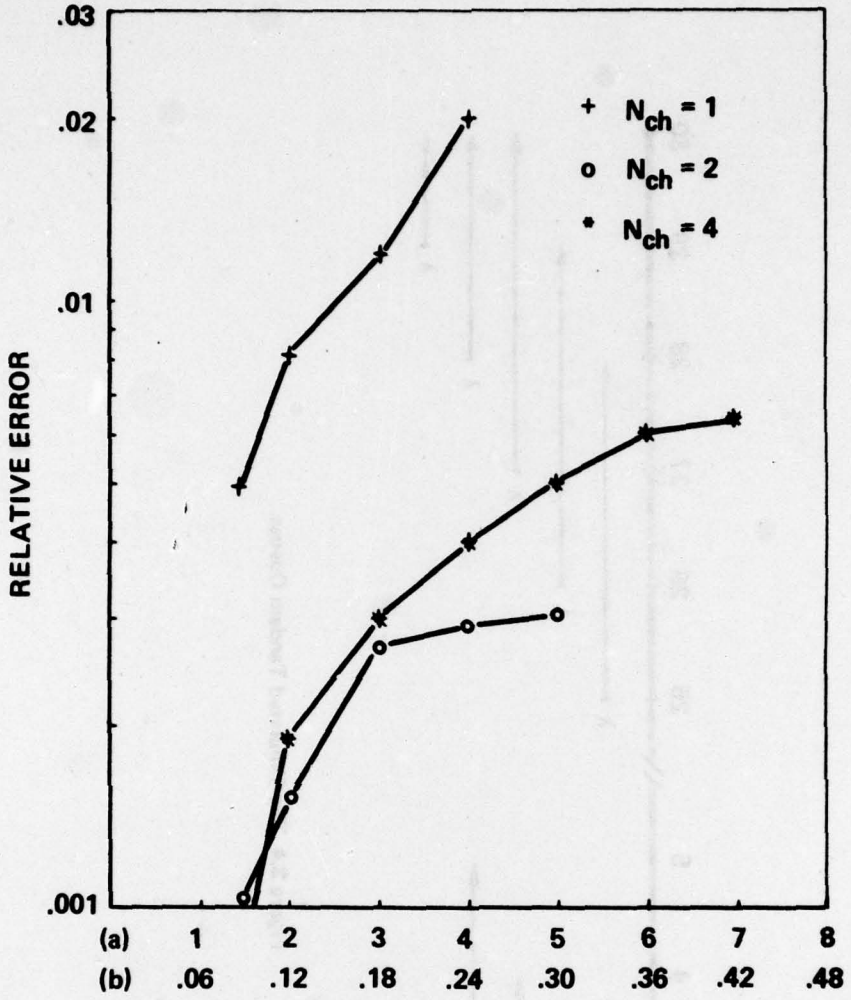


Figure 3.4 The Simulated Tandem Queues.



(a) EXTERNAL INPUT RATE (MSG/SEC)
(b) USEFUL UTILIZATION

MESSAGE CHANNEL CAPACITY	50 KBS
SIGNAL CHANNEL CAPACITY	8 KBS
MESSAGE LENGTH	1000 BITS
SIGNAL LENGTH	100 BITS

Figure 3.5 Accuracy of Analytic Result.

3.4 Analysis of a Path Model for the Circuit-Switching System

The solution of the network model of the previous section required treatment of a multiple server queueing system with non-Poisson input process and non-exponential service time; a system that, to date, has not in general been solved. In order to develop a model which is analytically tractable we make some further simplifying assumptions. To begin with, instead of a network, we consider a path of a communication network (Fig. 3.6-a), and consider the traffic between the source node S and the destination node D. The path consists of n_h hops and the communication links between adjacent nodes are divided into N_{ch} channels. For simplicity, we study a network that operates under the "forward reservation - individual signalling" mode. This means, a signal, after reserving a channel at a node, say i, uses the same channel for transmission to the next node (i + 1). While holding the channels reserved between the source node S and node i + 1, the signal waits at node i + 1 for a channel to become free; at this time the signal is transmitted to node i + 2 and the process continues similarly. When the signal reaches the destination node D a complete path between node S and D is set up. At this time a signal (request-for-transmission) is sent back to node S through the reserved channels (hence no queueing is involved)*. Upon reception of the "request-for-transmission" signal, the message is transmitted and after completion of the transmission, the reserved channels are released immediately. The fact that channels are reserved from left to right (i.e., there is an ordering in the reservation process)

*The RFT signal was not used in the model of the previous section.

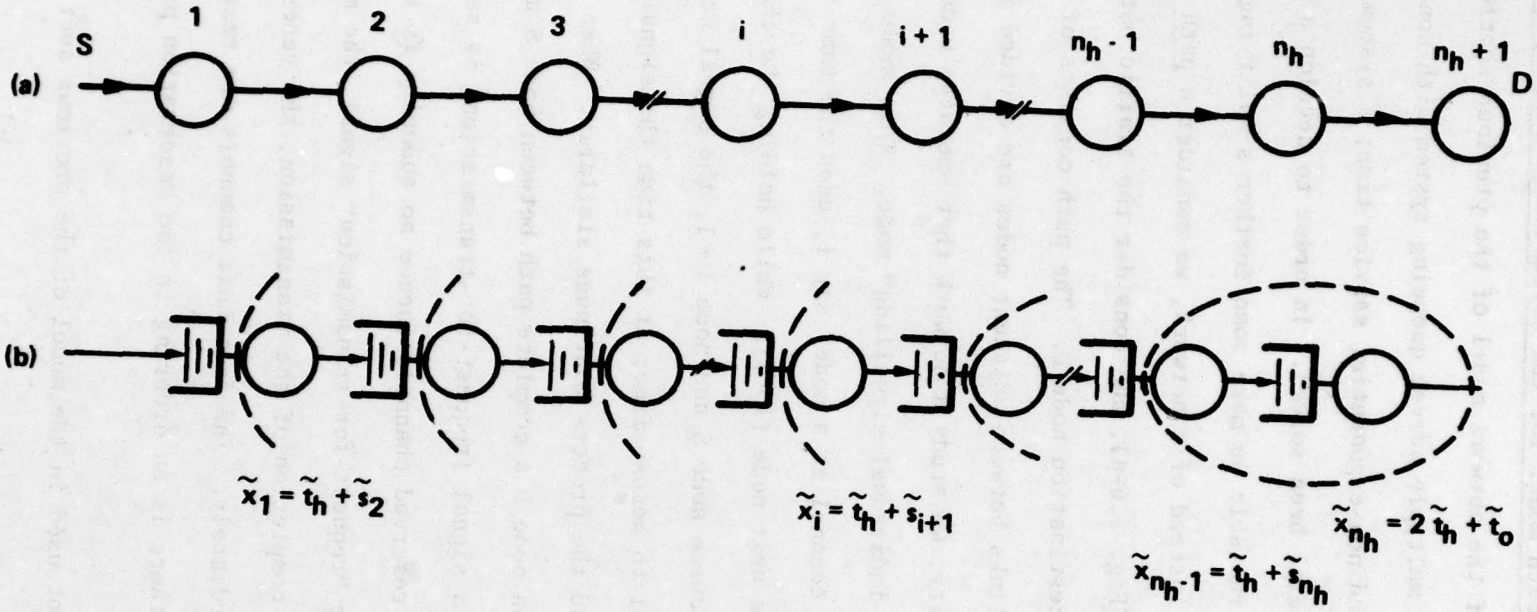


Figure 3.6 Tandem Queue Model of a Communication Path.

prevents the occurrence of deadlock [COFF 71]. Fig. 3.6-b shows the tandem queue model of the communication path (Fig. 3.6-a). In this figure each trunk is abstracted as an N_{ch} server, FIFO queueing system. Because of the reservation process, the service time of each queue is affected by waiting times in the succeeding queues. Let

$\tilde{x}_i \longleftrightarrow \bar{x}_i$ be the service time at the ith queue $\quad 1 \leq i \leq n_h$

and

$\tilde{s}_i \longleftrightarrow \bar{s}_i$ be the total delay at the i^{th} node $\quad 1 \leq i \leq n_h$

As before, t_0 and t_h are the transmission times of a message and a signal, respectively. The service time in the right-most queue (n_h^{th} queue) is

$$\tilde{x}_{n_h} = \tilde{t}_h + \tilde{t}_h + \tilde{t}_0 = 2\tilde{t}_h + \tilde{t}_0$$

The first $\tilde{t}_h$ corresponds to the "request-for-connection" signal from node n_h to node $n_h + 1$, the destination node. After this a "request-for-transmission" signal originates from node $n_h + 1$ and is sent to node 1. Because a complete path is already set up, this transmission takes only $\tilde{t}_h$ seconds. After reception of this signal, the message is transmitted, which takes $\tilde{t}_0$ seconds. The service time at queue $n_h - 1$ is affected by the system time (waiting plus service time) of the n_h^{th} queue, so we have

$$\tilde{x}_{n_h-1} = \tilde{t}_h + \tilde{s}_{n_h}$$

In general, we have the following expression for the service time at the i^{th} queue:

$$\tilde{x}_i = \tilde{t}_h + \tilde{s}_{i+1} \qquad 1 \leq i < n_h \tag{3.8-a}$$

$$\tilde{x}_{n_h} = 2\tilde{t}_h + \tilde{t}_0 \tag{3.8-b}$$

and for the average values we have

$$\bar{x}_i = t_h + \bar{s}_{i+1} \qquad 1 \leq i < n_h \tag{3.9-a}$$

$$\bar{x}_{n_h} = 2t_h + t_0 \tag{3.9-b}$$

In order to simplify the analysis we accept a generalized version of the independence assumption [KLEI 64].

Assumption 3.2

The distribution of service time at each of the queueing systems shown in Fig. 3.6-b is negative exponential with the average values determined by Eqs. (3.9), and is stochastically independent of the service time of its succeeding nodes.

Assuming that the input process of messages to the first node is Poisson, by virtue of Assumption 3.2, the input process to all of the queues on the path will be Poisson and the queueing system at each node can be treated as an $M/M/N_{ch}$ queue, N_{ch} being the number of servers (Appendix B.1).

To find the path delay (the time between the arrival of a message at the first node until the completion of its transmission), one must start from the last queue (n_h^{th}) and iteratively proceed to the first queue; the system time of queue 1, $\tilde{s}_1$, is the total path delay.

Remarks

(1) It has been shown [CLOS 72A], [KUMM 76A, B] that in the "common channel" signalling method, a correct signalling channel capacity should be dynamically adjusted to the traffic in the network. Because the emphasis of the present work is to study the effect of channel holding time on the overall performance, we based our study on an "individual signalling channel" model, hence we are not faced with this optimization problem.

(2) Perhaps it would be more accurate to make some simplifying assumptions and use the network model of Section 3.3; however, as we are doing a comparison study, by using the same path model for the other switching systems (CTS and MS), any inaccuracy which is the result of the topology will be common to all three systems.

(3) We should point out that the way we have treated the holding time distribution is not accurate in a narrow sense. Perhaps some other distributions (e.g., Erlangian) for the service time would be a better choice; however, any distribution except negative exponential would complicate the solution. As we shall see shortly, even this simple model clearly shows the effect of channel reservation and holding time. In all of the previous reports, except in [PORT 71], the queueing system at each node is considered as an M/M/m system in which the average service time is the same as the average transmission time of a message, an assumption which is far from realistic (unless the model is for a fully connected network); our model corrects that defect.

3.5 Discussion of Results

Based on the model developed in the previous section we present a numerical comparison of the network delay for the three switching systems: CTS, MS and CS. The network model is the one shown in Fig. 3.6-a and our comparison is based on network delay only, i.e., we will disregard the access delay to the network, as this component of delay is common to all three systems. A total trunk capacity of C = 50 kbps is used and for CS the trunk is subdivided into N_{ch} channels (hence for CTS and MS we consider single channel links only). The average header (or signal) length is assumed to be 100 bits and the average message length ($1/\mu$) will be specified for each case. Regarding the distribution of the random variables involved and the input process, we accept the assumptions of Section 2.1 and also assumption 3.2. Lastly, we also assume that the channels are noiseless. The useful utilization, which for simplicity will also be referred to as utilization, denoted by $\rho = \lambda/\mu C$ (λ being the traffic rate on a trunk), is the fraction of the total trunk capacity used by the useful information (i.e., the message). This is usually different from (and less than) the effective utilization ρ_e, which is the utilization caused by the useful information, the header (or signal) and the holding time of the reserved channel. In our comparison study, we study the effect of four parameters: input rate; average message length, $1/\mu$; path length, n_h; and the number of channels per line, N_{ch} (this last parameter is only important for CS).

We first consider the network delay for a path of average length, $n_h = 4$, Figure 3.7-a. The average message length (useful information) is 1500 bits. This figure shows the normalized path

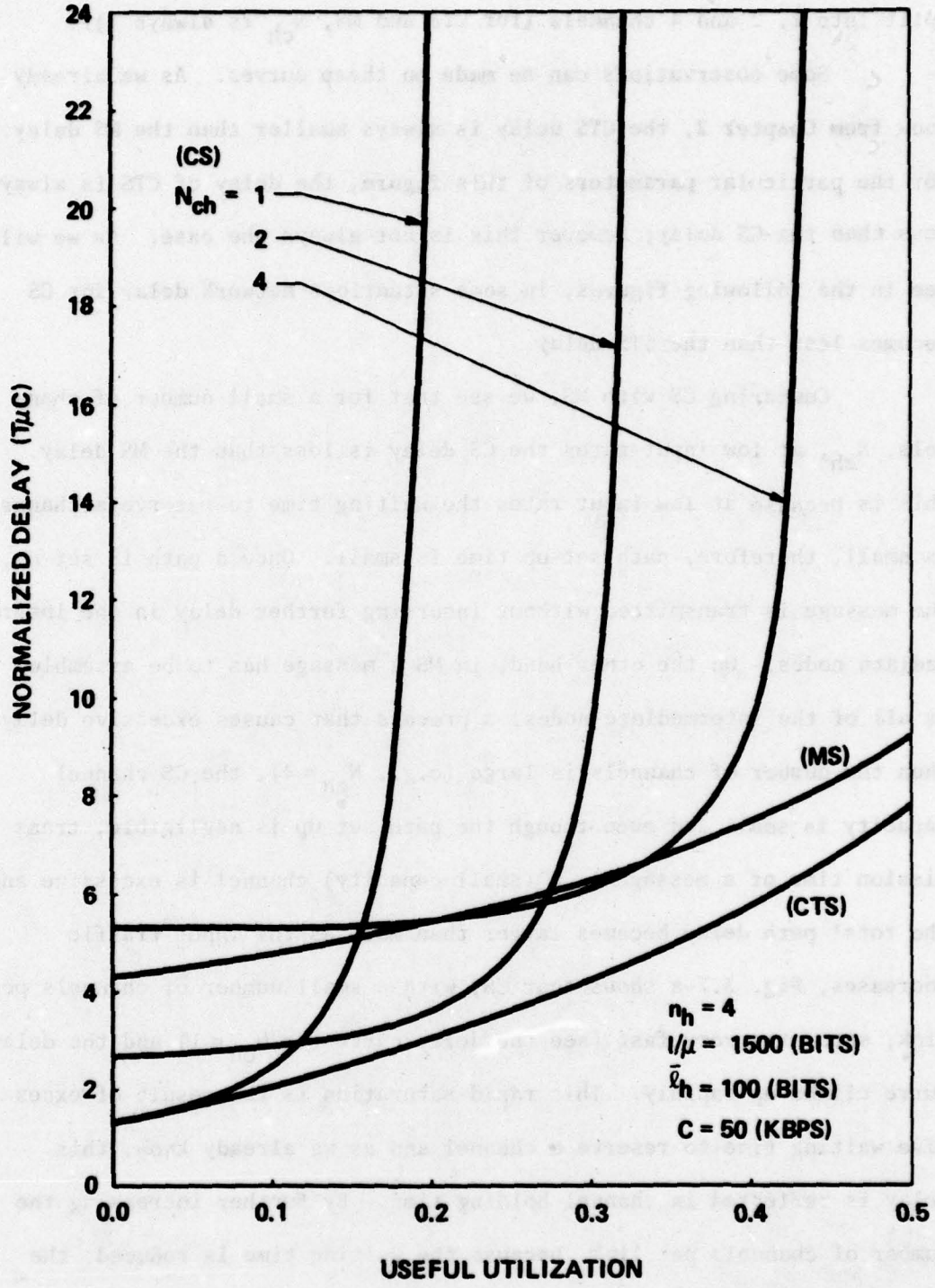


Figure 3.7-a Comparison of Switching Systems.

delay (= delay/t_0 = delay μC) for CTS, MS and CS when the trunk is split into 1, 2 and 4 channels (for CTS and MS, N_{ch} is always 1).

Some observations can be made on these curves. As we already know from Chapter 2, the CTS delay is always smaller than the MS delay. For the particular parameters of this figure, the delay of CTS is always less than the CS delay; however this is not always the case. As we will see in the following figures, in some situations network delay for CS becomes less than the CTS delay.

Comparing CS with MS, we see that for a small number of channels, N_{ch}, at low input rates the CS delay is less than the MS delay. This is because at low input rates the waiting time to reserve a channel is small, therefore, path set-up time is small. Once a path is set up, the message is transmitted without incurring further delay in the intermediate nodes. On the other hand, in MS a message has to be assembled at all of the intermediate nodes, a process that causes excessive delay. When the number of channels is large (e.g., $N_{ch} = 4$), the CS channel capacity is small and even though the path set up is negligible, transmission time of a message on a (small capacity) channel is excessive and the total path delay becomes larger than MS. As the input traffic increases, Fig. 3.7-a shows that CS, with a small number of channels per link, saturates very fast (see the delay curve for $N_{ch} = 1$) and the delay curve climbs up rapidly. This rapid saturation is the result of excessive waiting time to reserve a channel and as we already know, this delay is reflected in channel holding time. By further increasing the number of channels per link, because the waiting time is reduced, the network does not saturate so fast (see the CS delay curve for $N_{ch} = 4$),

even though the transmission time increases. Comparing this curve with the delay curve of MS we observe an interesting behavior. For very small input rates, the MS delay is less than the CS delay; however, as the input rate increases, there is a cross-over between the two delay curves, and for a short interval the CS delay becomes less than the MS delay. For a further increase of input rate, there is a sharp increase in the CS delay and MS again becomes better than CS (from the delay point of view). The cross-over occurs because for a certain range of input rates the combined waiting time and reassembly delay at the intermediate nodes for MS becomes more than the path set-up delay and message transmission time for CS. We should notice that the interval at which CS becomes better than MS depends very much on the parameters of the system. For example, for a small average message length ($1/\mu$ = 1000 bits; Fig. 3.7-b), there is no cross-over point for the MS and CS delay curves (for $N_{ch} = 4$). (In Fig. 3.7-b cross-over occurs at larger values of N_{ch}.) For comparison, we have also presented delay curves for the average message length of 2000 bits in Fig. 3.7-c and a longer path length, $n_h = 8$, in Fig. 3.8. Comparison of Fig. 3.7-a with Fig. 3.8 shows that for long path lengths, CS is more advantageous than MS in a larger region. This is because in MS it is necessary to assemble the message in all of the intermediate nodes and so the network delay becomes large, whereas in CS, once a path is set up, there is no further delay due to intermediate nodes.

Before going further, we find it useful to point out an unusual behavior of CS. Consider two identical queueing systems that differ only in the number of servers; the total service capacity of both systems are

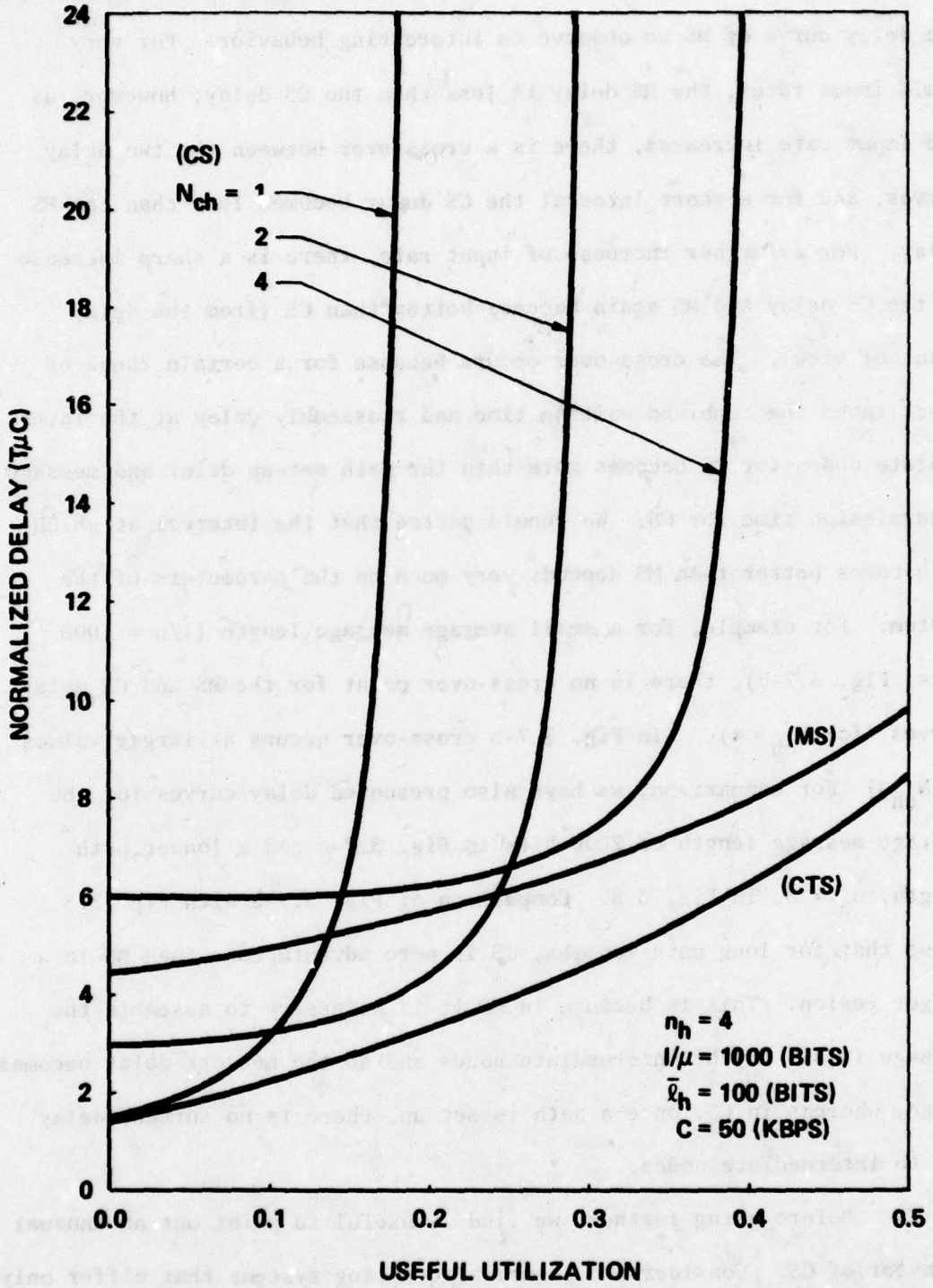


Figure 3.7-b Comparison of Switching Systems.

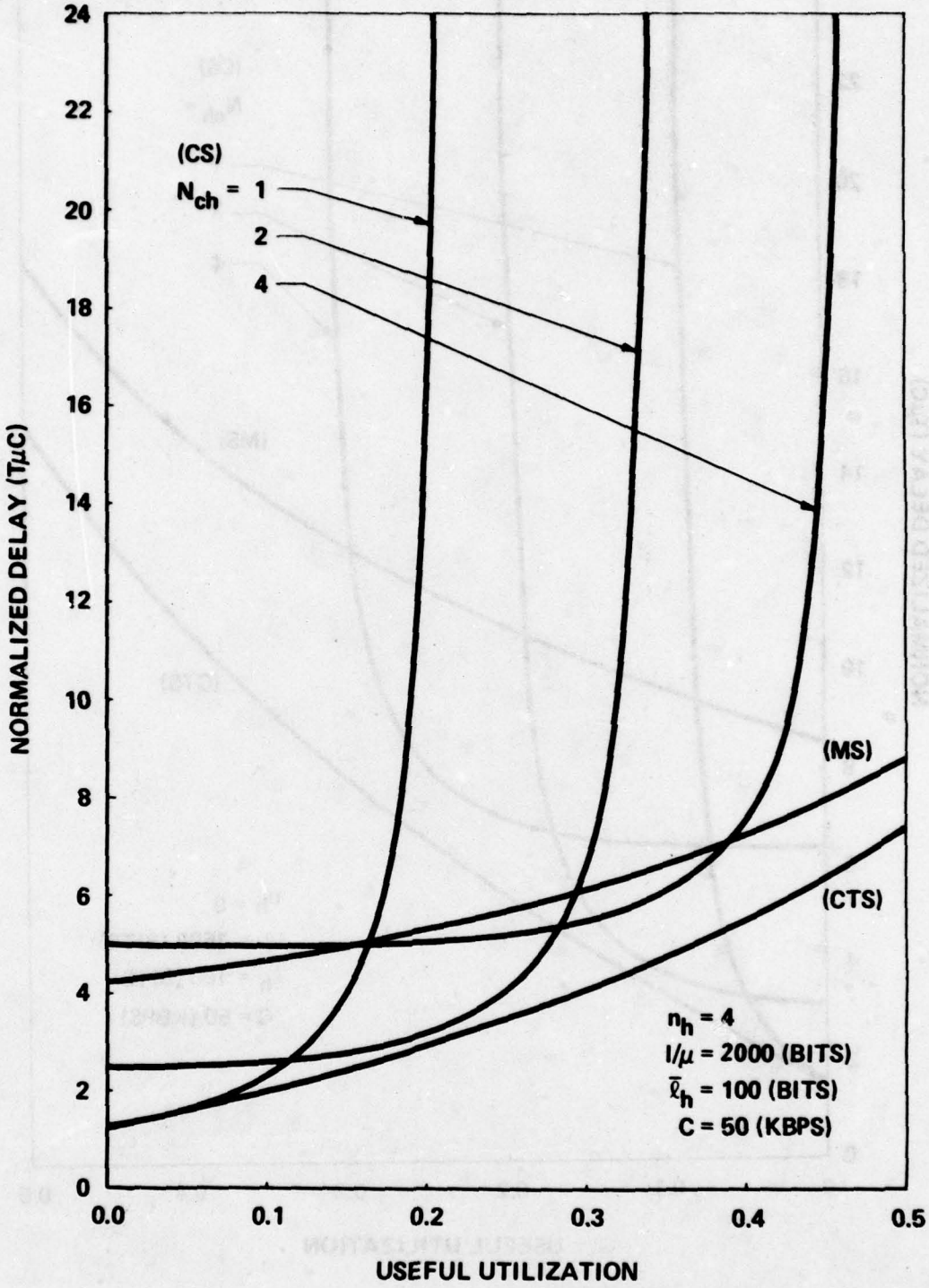


Figure 3.7-c Comparison of Switching Systems.

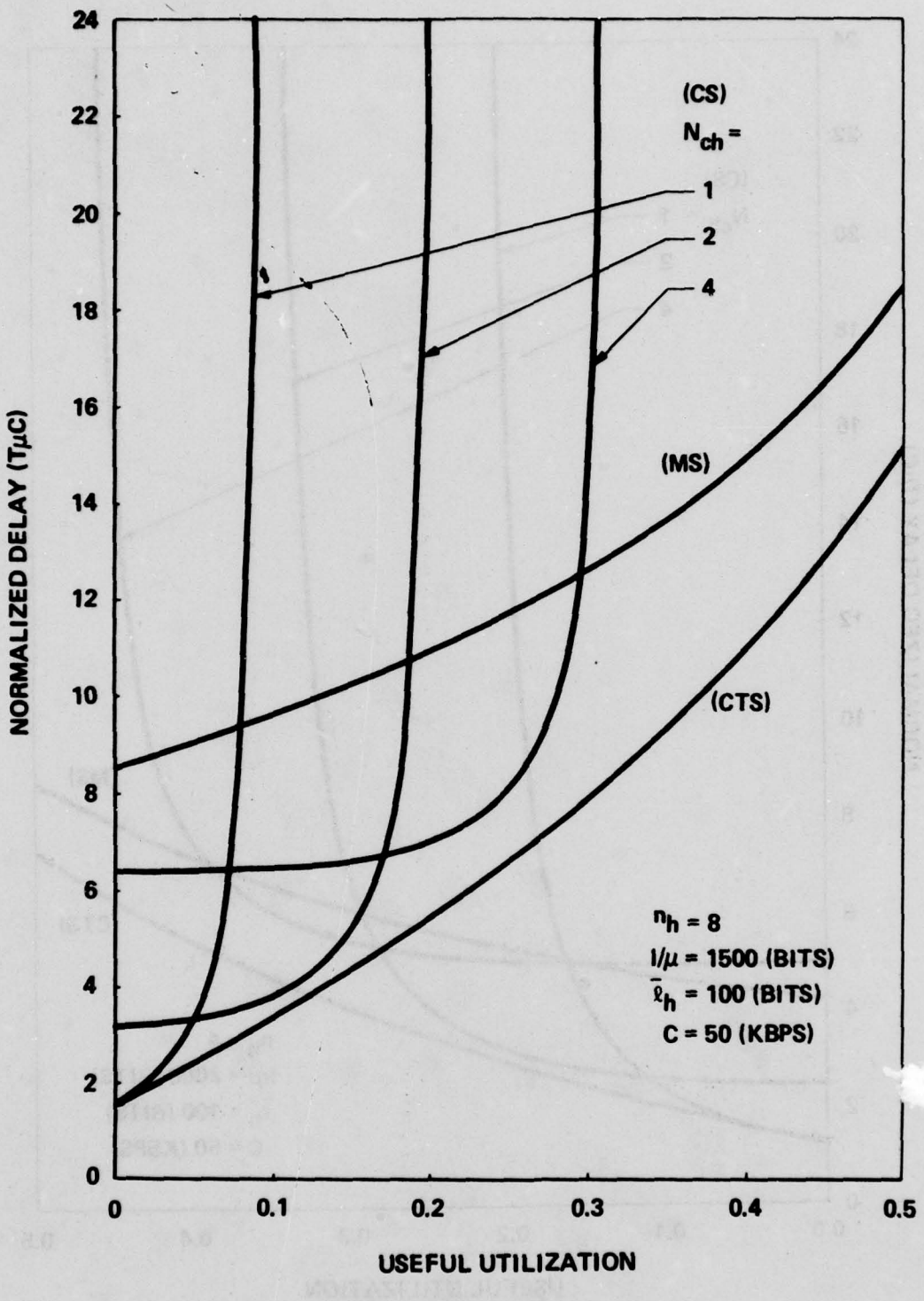


Figure 3.8 Comparison of Switching Systems.

the same. The waiting time (time from the arrival of a request until the request starts service) in the system with a larger number of servers is always smaller than the other one; however, the total delay (time between the instant of arrival until the instant of departure from the system) for the system with the smaller number of servers is always less than the other [KLEI 74A]. The delay curves for CS do not show this behavior. In Fig. 3.7-a, as the number of channels per link is increased (the total capacity is, however, kept constant), for small input rates, the system delay for a smaller number of channels is less than the delay for a larger number of channels; however, as the input rate is increased, the delay in the system with the smaller number of channels becomes unbounded, whereas the systems with the larger number of channels stay unsaturated. This behavior is the result of the peculiar interaction of demands for service from the system.

Figs. 3.9-a and 3.9-b show the effect of message length on network delay. In these figures the useful utilization is kept constant ($\rho = 0.3$), hence the input rate varies as the message length changes (the header, or signal, length is kept constant: 100 bits). For too small message lengths ($1/\mu < 100$ bits), the network delay for all three switching systems is unbounded, a consequence of the high input rate (to keep the useful utilization constant). As the message length increases, the network delay first decreases and then grows again. As before, we observe that the CTS delay is always less than the MS delay. For CS, after a sharp decrease (which indicates a rapid recovery from saturation) the delay escalates; however the rate of increase for the delay in CS is less than the rate for the other switching systems and

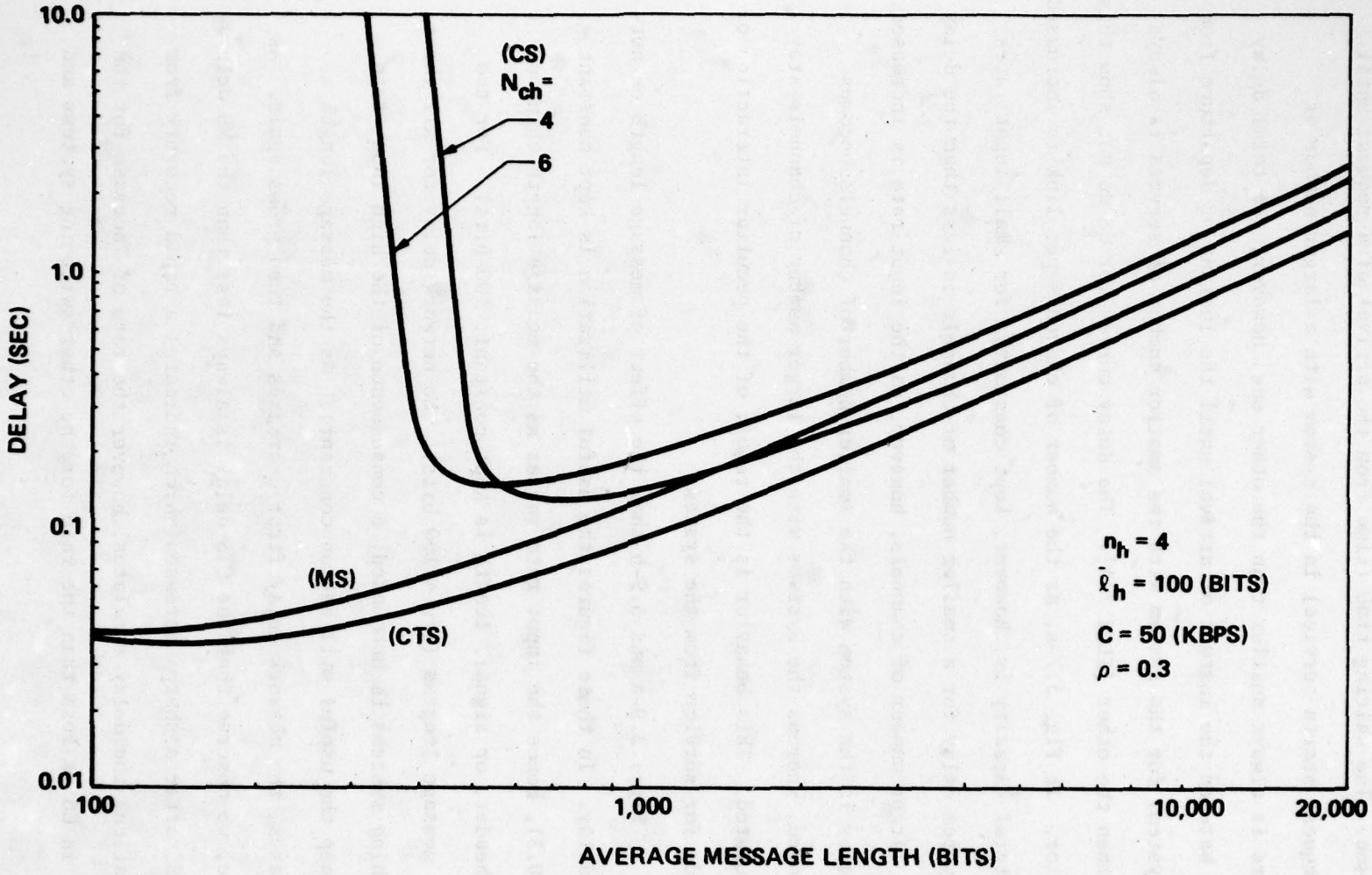


Figure 3.9-a Comparison of Switching Systems.

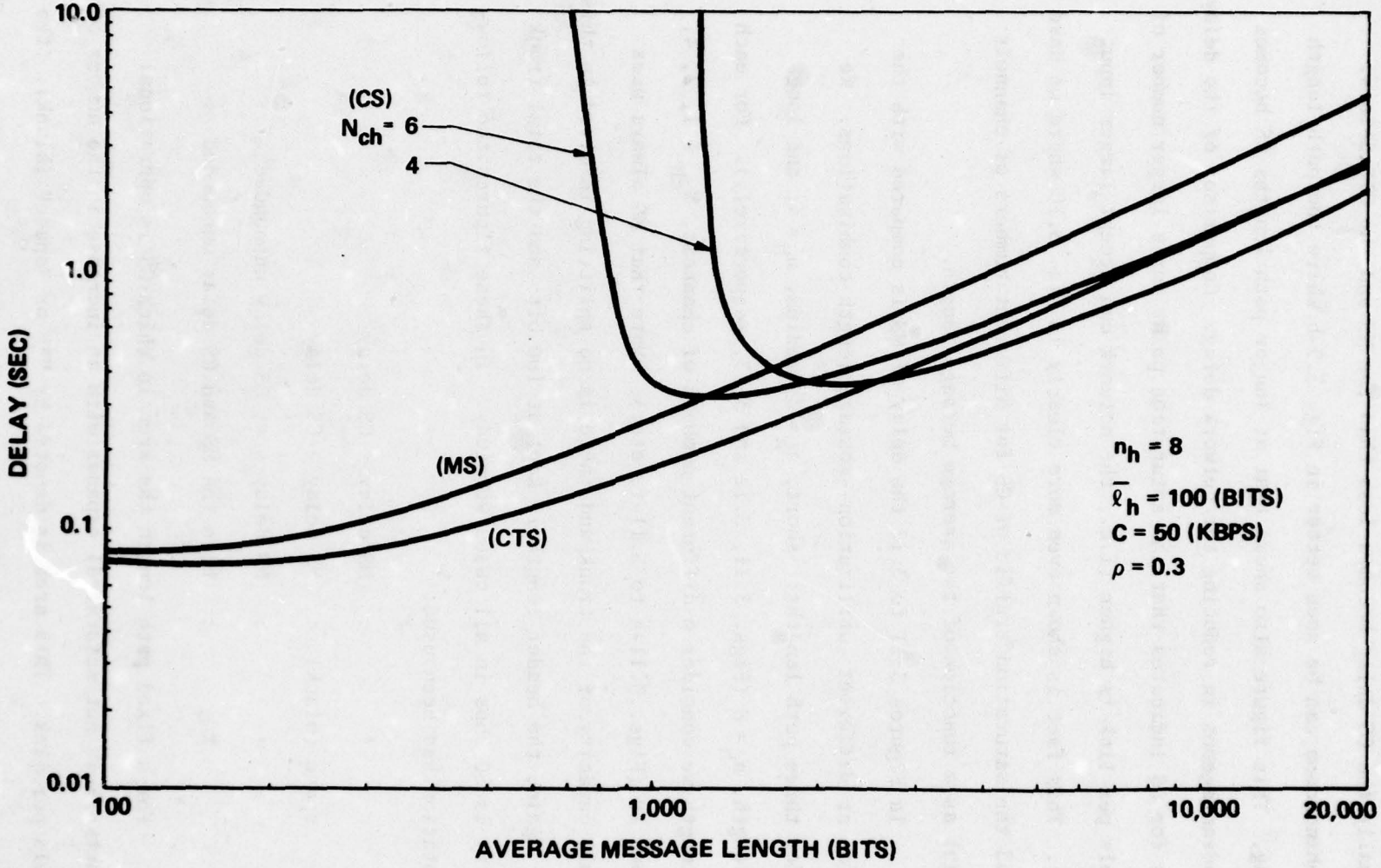


Figure 3.9-b Comparison of Switching Systems.

eventually the CS delay becomes less than the MS and the CTS delays. This phenomenon can be seen better in Fig. 3.9-b where the path length is long. This figure also shows that at longer path lengths CS becomes more advantageous in reducing the network delay. Comparison of the delay curves for CS indicates that the saturation point for a larger number of channels per link is higher (i.e., the network can accept larger input rates). This fact is shown even more clearly in Fig. 3.10 where we have plotted the saturation traffic of CS for different numbers of channels per link as a function of the average message length.

In Figures 3.11 to 3.13 the delay of MS is compared with the CS delay at different utilization--message length combinations. We consider three path lengths: short, $n_h = 2$; medium, $n_h = 4$; and long path length, $n_h = 8$ (Figs. 3.11, 3.12 and 3.13, respectively). For each path length we consider 6 different numbers of channels, N_{ch} = 1, 2, 4, 6, 8 and 10 (Figs. 3.11-a to 3.11-f, etc.). Note that MS always uses the full capacity of the trunk and there is no splitting involved in this case. Again, the header length is kept at 100 bits and the total trunk capacity is 50 kbps in all cases we study. In these figures the following notation has been used:

*	:	MS delay < CS delay
space (blank)	:	MS delay > CS delay
•	:	MS delay < ∞; CS delay unbounded
X	:	both the MS and CS delay unbounded

For a fixed path length the area in which CS is operational (the network is not saturated) expands with an increase in the number of channels per link. This area is denoted by "*" or "space" (blank), (the

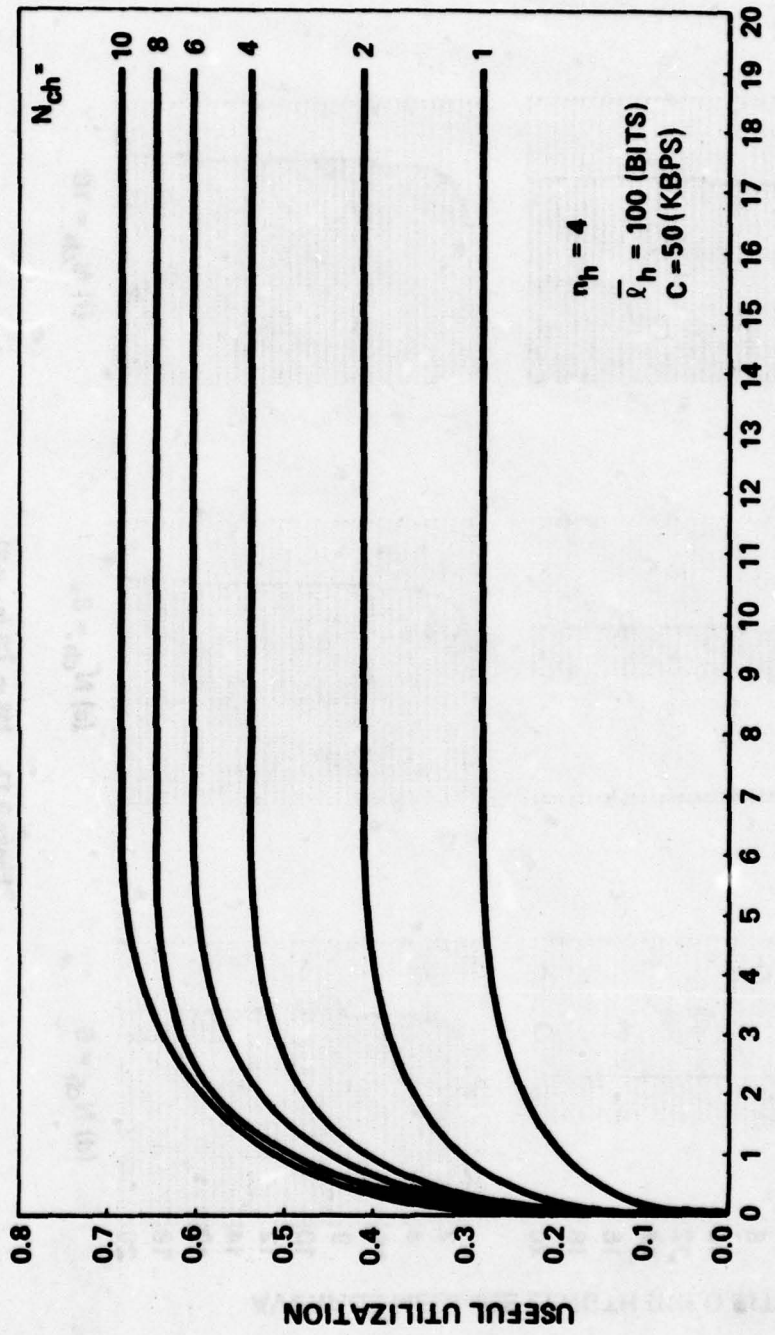


Figure 3.10 Saturation Point of Circuit Switching.

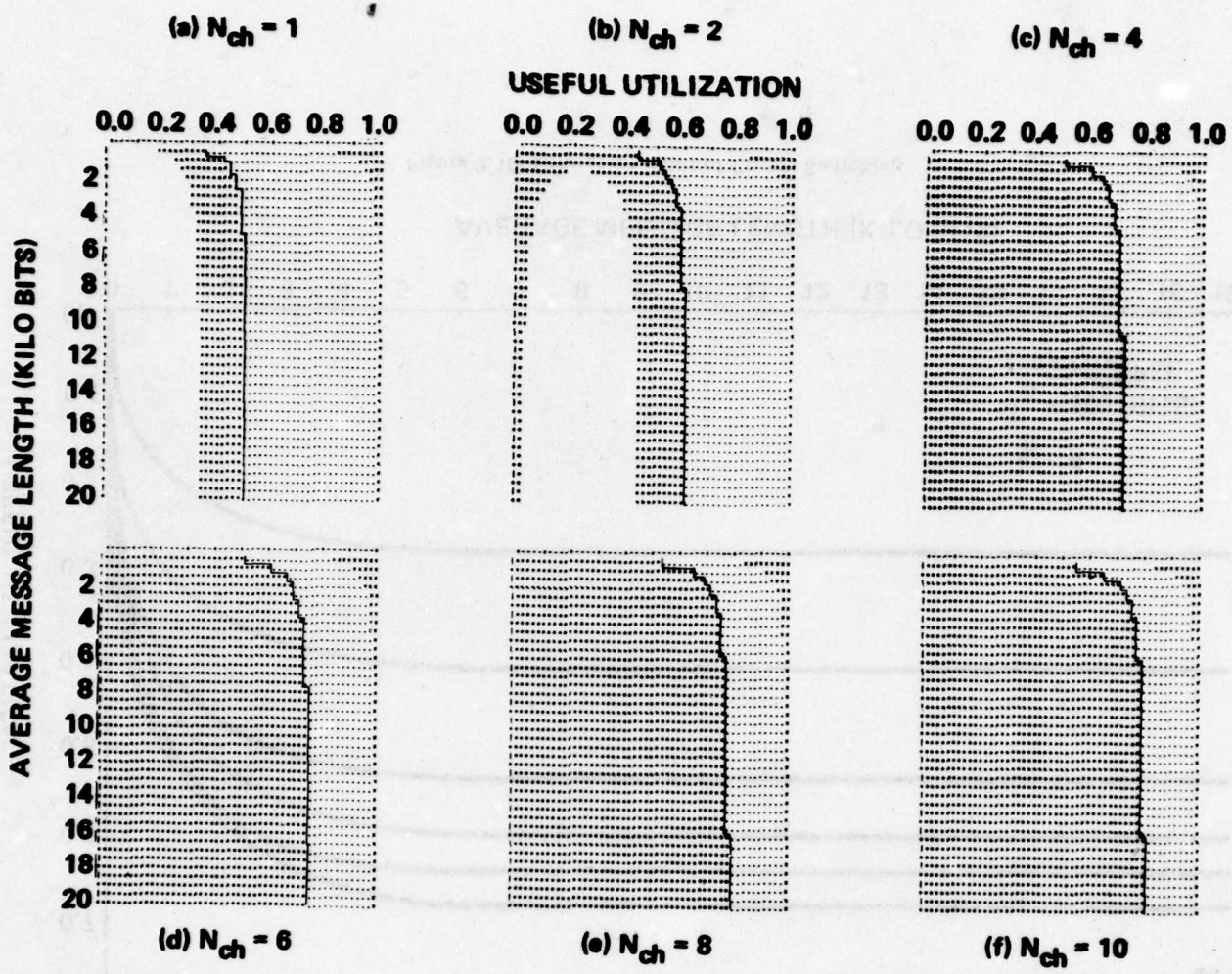


Figure 3.11 MS vs. CS (n_h = 2).

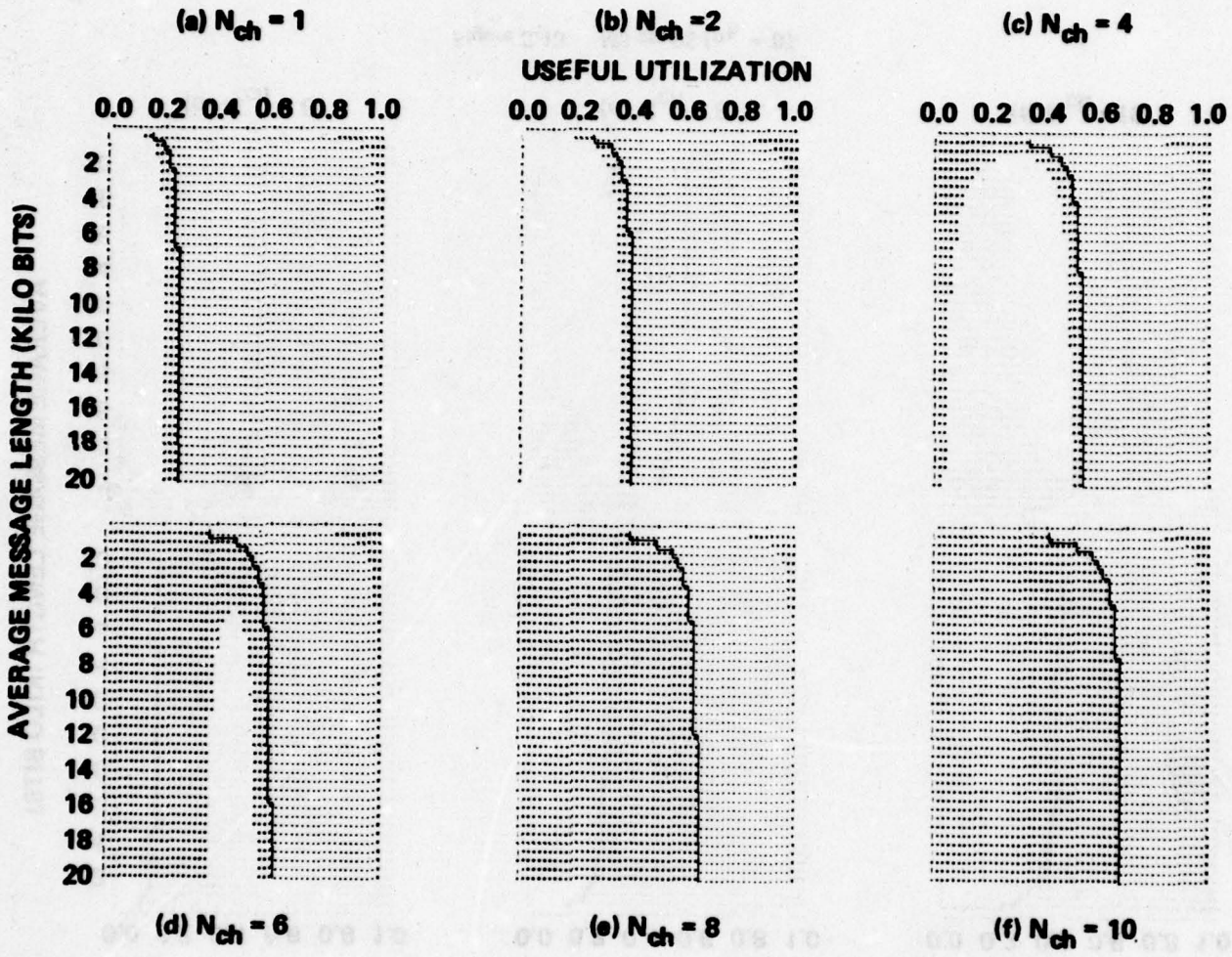


Figure 3.12 MS vs. CS (n_h = 4).

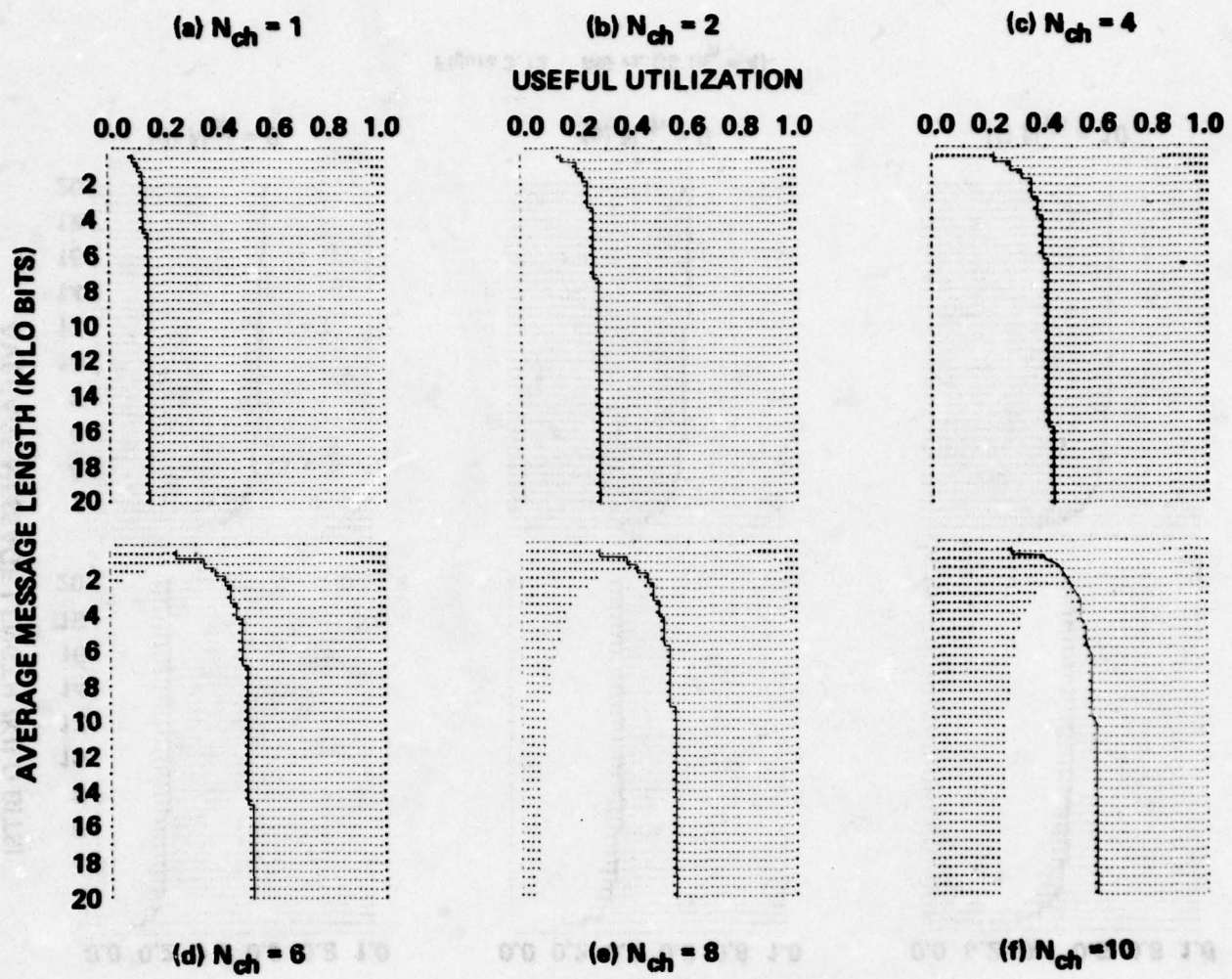


Figure 3.13 MS vs. CS (n_h = 8).

region designated by "." or "X" is where the CS delay is unbounded). For a given number of channels, as the path length increases, the area where CS is operational shrinks. This effect can be seen by comparing, for example, Figs. 3.11-f, 3.12-f and 3.13-f with each other. For short path lengths a large number of channels is not advantageous to CS (Fig. 3.11), although a small number of channels shrinks the operational region of CS.

To complete our study, we have done the same type of comparison between CTS and CS in Fig. 3.14 for a long path length, $n_h = 8$. As we pointed out earlier, CS manifests its advantage when the path length is long; for shorter path lengths ($n_h < 8$), CS is hardly better than CTS. Figure 3.14 is very similar to its counterpart figure, Fig. 3.13; however the area more favorable to CS has shrunk. This is due to the fact that the CTS delay is always less than the MS delay.

Table 3.1 shows the results of our comparison study. In this table we have chosen two intervals for the message length ($500 \leq 1/\mu \leq 2000$ and $2000 < 1/\mu$) and three intervals (small, medium and large) for the other three parameters (ρ, N_{ch} and n_h). In each box the entry in the upper left corner is the result of the comparison between CS and MS and the one in the lower right corner relates to CS and CTS. If in an interval the MS delay is less than the CS delay, then the CTS delay is also less than the CS delay and we have indicated only "MS" for such an interval. In some cases we have indicated two switching systems (e.g., CS and MS). This indicates that the selected range is too large and a finer interval should be specified to determine a firm advantage between the two systems.

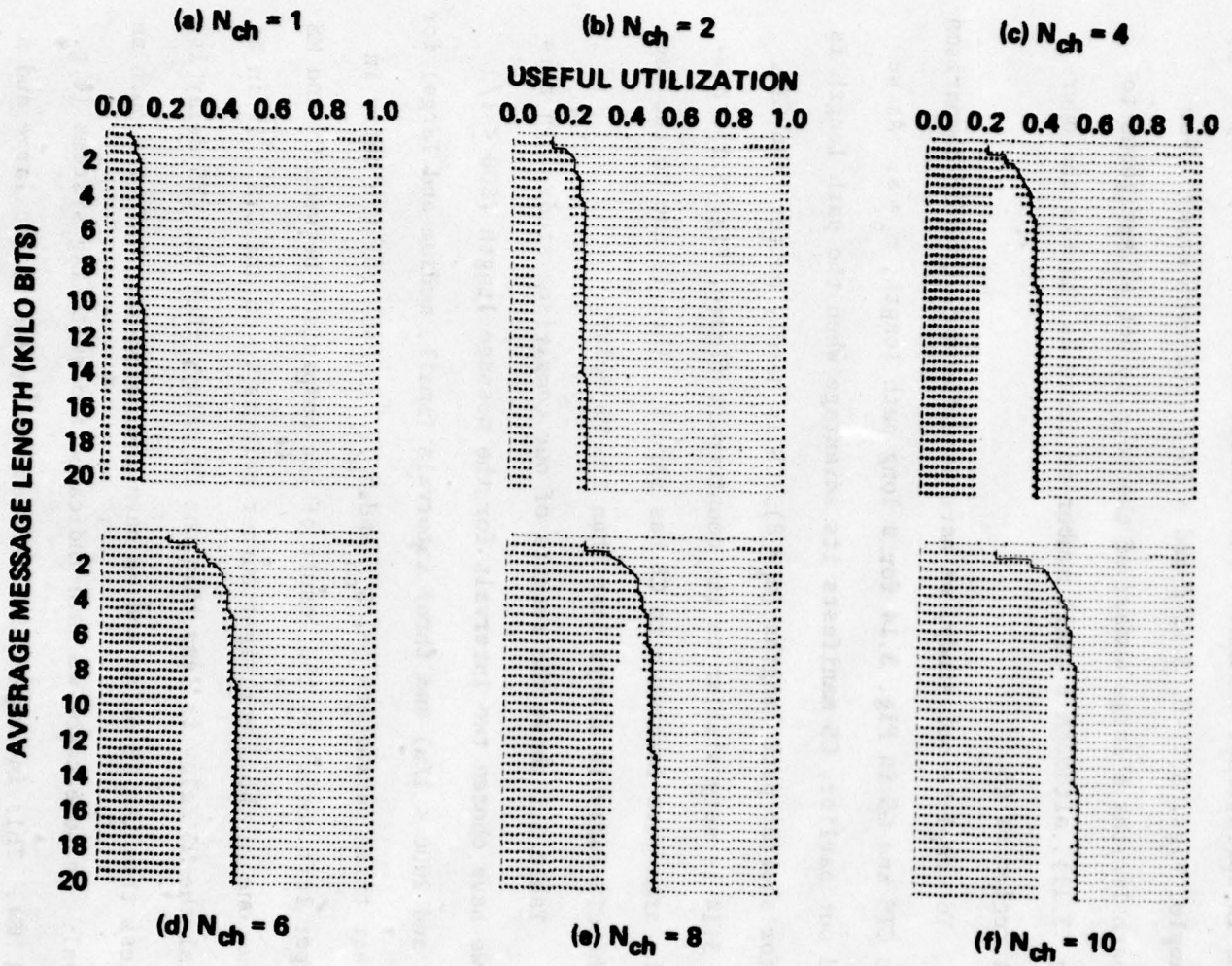


Figure 3.14 CTS vs. CS (n_h = 8).

			SMALL $n_h = 2$		MEDIUM $n_h = 4$		LARGE $n_h = 8$	
			$500 \leq \ell_m \leq 2000$	$2000 < \ell_m$	$500 \leq \ell_m \leq 2000$	$2000 < \ell_m$	$500 \leq \ell_m \leq 2000$	$2000 < \ell_m$
SMALL	S	$N_{ch} = 1$	CS / CTS	CS / CS-CTS	CS / CTS	CS / CS-CTS	CS / CTS	CS / CS-CTS
$\rho \leq .1$	M	$2 \leq N_{ch} \leq 4$	MS	MS	MS	MS	CS-MS / CTS	CS-MS / CS-CTS
	L	$4 < N_{ch} \leq 20$	MS	MS	MS	MS	MS	MS
MEDIUM	S	$N_{ch} = 1$	CS / CTS	CS-MS / CS-CTS	CS-MS / CTS	CS / CTS	MS	MS
$1 \leq \rho \leq .4$	M	$2 \leq N_{ch} \leq 4$	MS	CS-MS / CTS	CS-MS / CTS	CS	CS / CTS	CS
	L	$4 < N_{ch} \leq 20$	MS	MS	MS	MS	MS	CS
LARGE	S	$N_{ch} = 1$	MS	MS	MS	MS	MS	MS
$\rho > .4$	M	$2 \leq N_{ch} \leq 4$	MS	MS	MS	MS	MS	MS
	L	$4 < N_{ch} \leq 20$	MS	MS	MS	MS	MS	MS

Table 3.1. Comparison of Switching Techniques

Summarizing our observations, the following remarks can be made:

(1) The reservation operation in CS causes a substantial decrease in the network capacity. For this reason CS is not a good choice when the traffic is high (Table 3.1).

(2) For large messages and large path lengths, if the utilization is not too high, CS usually outperforms the other two switching systems (Table 3.1).

(3) When the capacity of the trunk is kept fixed, proper selection of the number of channels per trunk is very critical for the proper operation of the CS system. In fact, this number has an opposing effect on the performance of the network. When the number of channels is small, then the network saturates quickly; on the other hand, a large number of channels results in a large delay. We deal with the problem of optimal number of channels in the next section.

(4) Based on our study, the decision as to which form of switching to use should be based on a careful study of the parameters of the network and no general quantitative statement in this regard can be made.

If the data stream is intermittent, i.e., if a communication session consists of an alternating sequence of message blocks and idle periods, depending on the duration of the idle periods and the message lengths, holding the connection during the idle periods may be to the advantage of CS. We have not studied such cases: however, the analytic model developed in the section can easily be modified to encompass

this situation.

3.6 Optimal Number of Channels in Circuit Switching

With the growth of telecommunication networks and remote data processing facilities there has been an increased diversity in the characteristics of the traffic requirement for these communication systems. Our study in the previous section indicates that no single switching system can provide the best performance for all types of traffic requirements and, in fact, the best choice for the switching technique is an integrated system of different techniques. Integration of switching systems has recently become a subject of interest ([FISC 76], [GERL 78], [PORT 76] and the references therein). In [PORT 76], the feasibility of such an integration was studied; based on that study a network architecture was proposed. (In fact, the virtual cut-through switching that we proposed in Chapter 2 was motivated by the idea of integrating circuit switching and packet switching.)

In this section we assume that such an integration is feasible and that we have the hardware facility to integrate both MS and CS in a single network such that, depending on the users' demands and network load, a proper switching mode can be selected to provide the best grade of service (we consider only MS and CS). We address the following question: given the trunk capacity and the path length, which switching technique (MS or CS) should be used to minimize the delay, and if the CS system is to be used, what is the best choice for N_{ch} (number of channels per link)?

For the solution we consider two cases: dynamic and static

trunk splitting. In the first case, the number of channels per link can be dynamically adjusted according to the load pattern in the network in order to minimize the network delay. In the static case, however, once the value of N_{ch} is chosen, it cannot be changed.

Case I: Dynamic Trunk Splitting

Dynamic trunk splitting can be done by synchronous time division multiplexing techniques (STDM); we do not address the implementation issues. We assume that the number of channels per link can be adjusted in order to minimize the network delay. If it turns out that MS results in a smaller network delay, then the total trunk capacity is dedicated to message switching. The result of such an optimal channel splitting and switching system selection is shown in Figs. 3.15-a through 3.15-c where we show results for three different path lengths. For an "average message length - useful utilization" combination, if CS results in a smaller delay, the optimum number of channels (modulo 10) is printed. If CS does not give a smaller delay than MS, then that point is left blank. The areas designated by "X", as in the previous figures, are where both MS and CS delays are unbounded. Fig. 3.15 shows, for a given useful utilization, that CS is preferable for larger message lengths and MS is advantageous for shorter message lengths. It should be noted, for some values of the utilization (for example $\rho > 0.6$) that MS is the only choice, as the CS delay is unbounded for these values of trunk utilization. On the other hand, for a given average message length, smaller utilizations favor CS; for large utilizations MS is a better choice for switching.

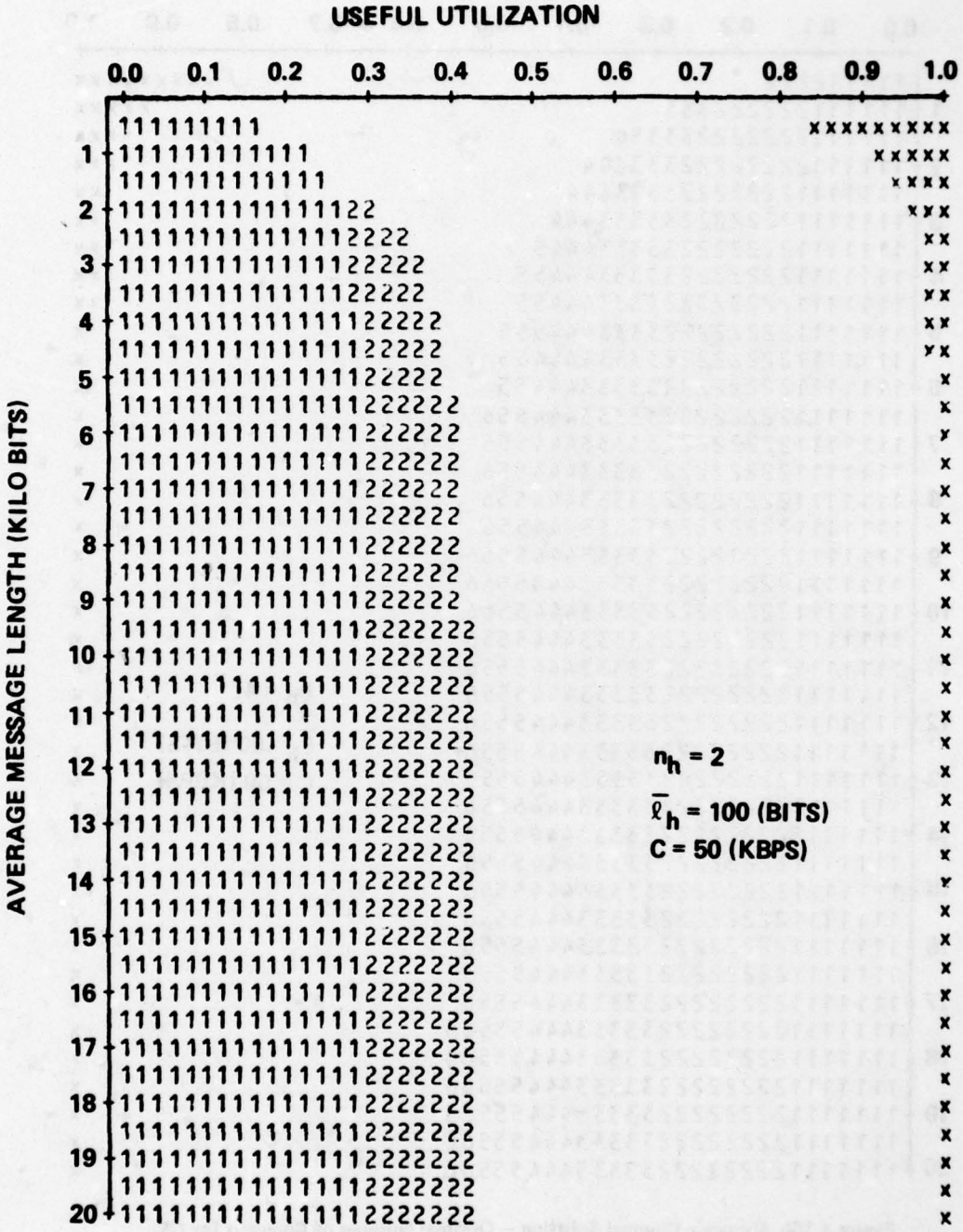


Figure 3.15-a Dynamic Channel Splitting – Optimal Number of Channels for CS.

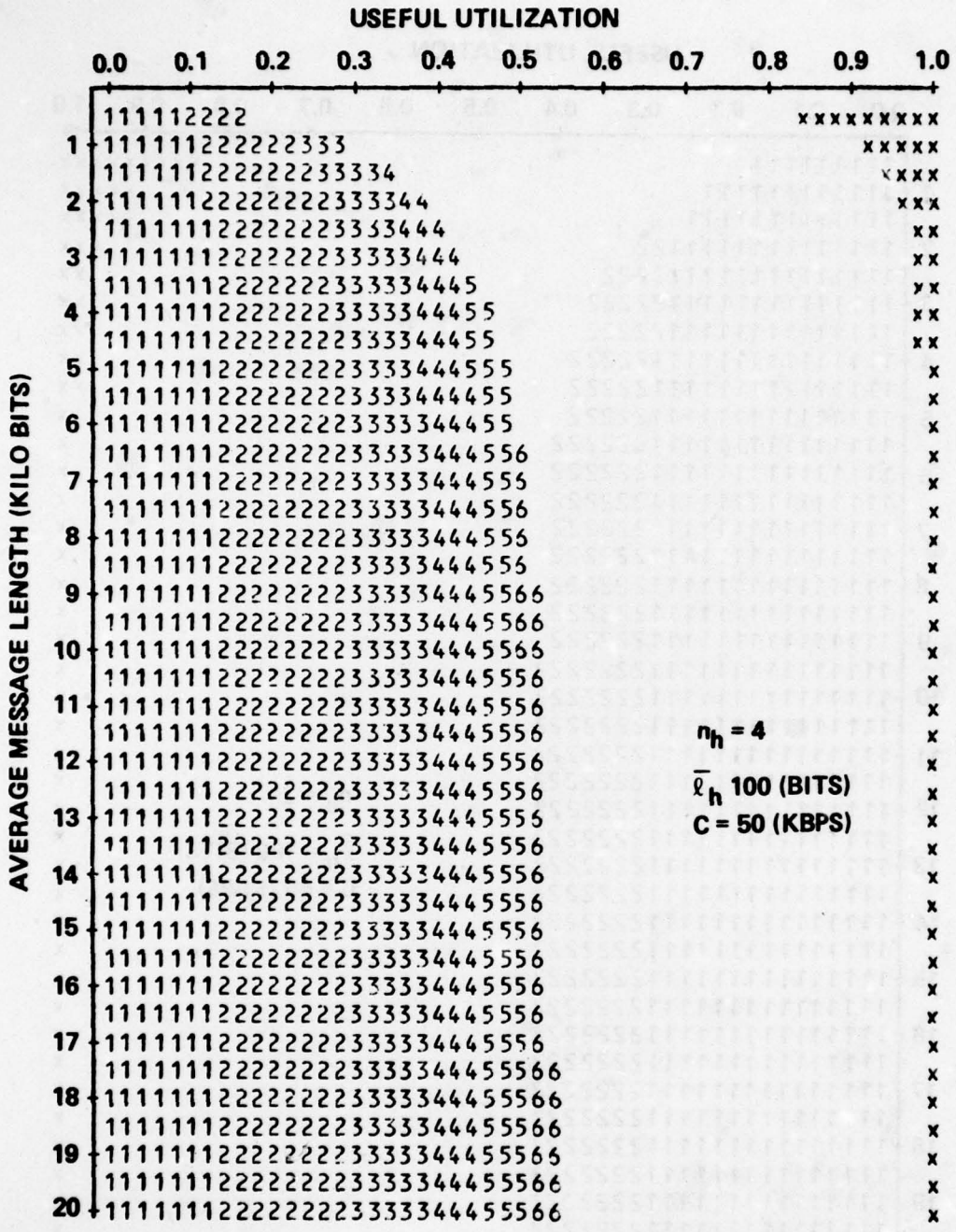


Figure 3.15b Dynamic Channel Splitting – Optimal Number of Channels for CS.

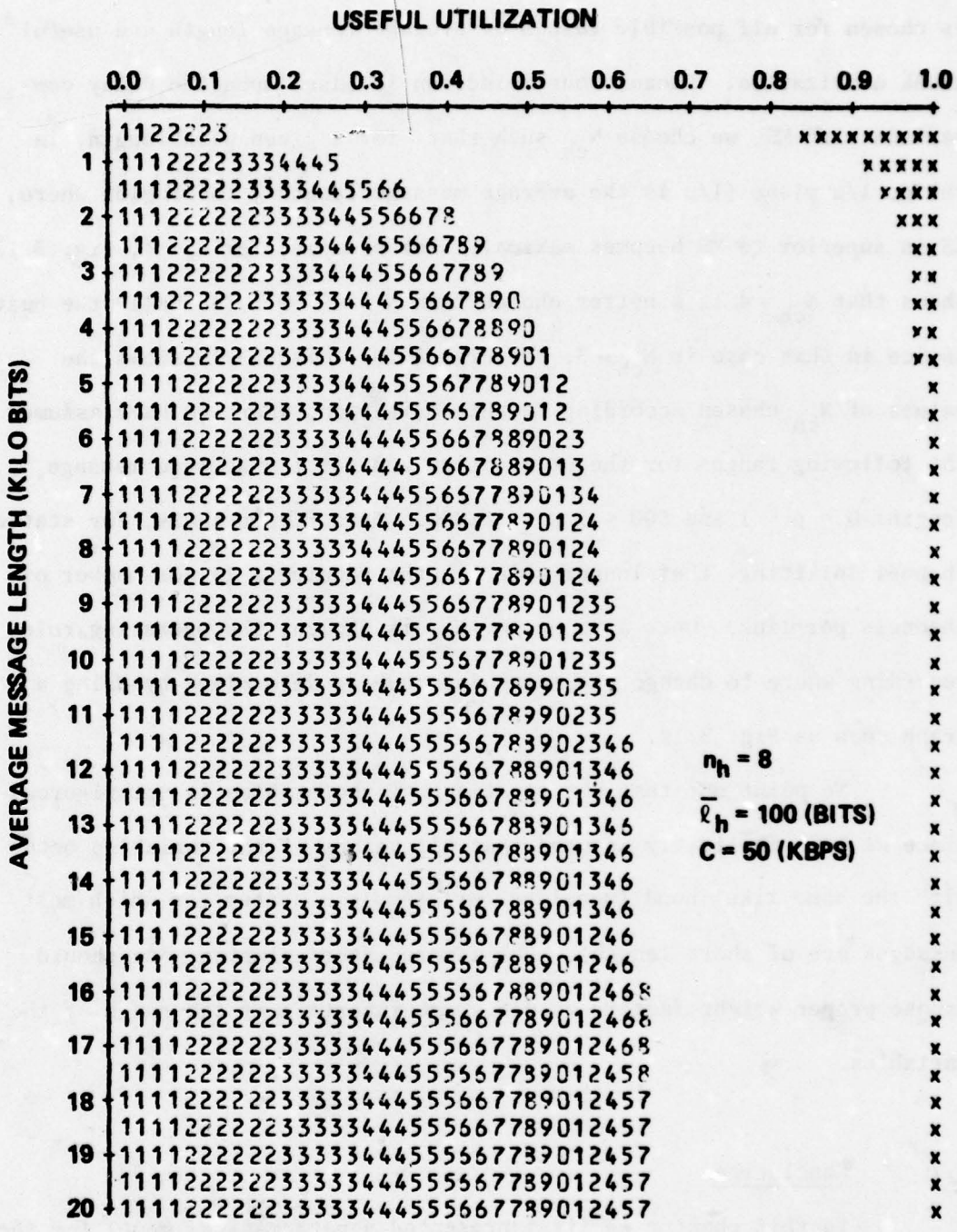


Figure 3.15-c Dynamic Channel Splitting – Optimal Number of Channels for CS.

Case II: Static Trunk Splitting:

In static trunk splitting, the number of channels per link is chosen for all possible values of average message length and useful trunk utilization. Because our selection is based upon the delay comparison with MS, we choose N_{ch} such that, for a given path length, in the ρ, $1/\mu$ plane ($1/\mu$ is the average message length), the region where CS is superior to MS becomes maximal. For example, for $n_h = 4$, Fig. 3.12 shows that $N_{ch} = 4$ is a better choice than $N_{ch} = 2$ or 1 (actually the best choice in this case is $N_{ch} = 3$, see Fig. 3.16). Fig. 3.16 shows the values of N_{ch} chosen according to this criterion, where we have assumed the following ranges for the utilization and for the average message length: $0 \leq \rho \leq 1$ and $500 < 1/\mu \leq 20{,}000$. This figure shows, for static channel splitting, that longer path lengths require a larger number of channels per link. Once a value for N_{ch} is chosen, the operating rule regarding where to change the switching mode is determined by using a graph such as Fig. 3.12.

We point out that our treatment of the problem is not rigorous since we have implicitly assumed that all values of the variables occur with the same likelihood (consider for example a system for which most messages are of short length). For a more exact solution, one should impose proper weight factors on different intervals of the range of the variables.

3.7 Conclusion

In this chapter we first presented a mathematical model for the delay performance of circuit switching which incorporated the effect of

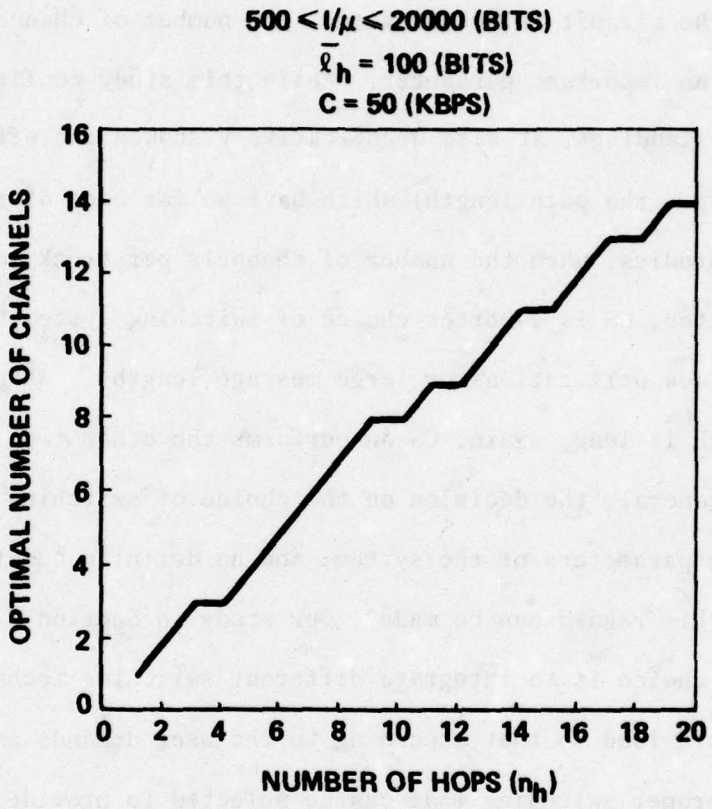


Figure 3.16 Static Channel Splitting – Optimal Number of Channels for CS.

reservations. By using this model we carried out a comparison study between the delay performance of three switching systems: circuit switching, message switching and cut-through switching. Our study showed that the boundary between the areas of relative effectiveness of these switching techniques depends greatly on the network topology (which was reflected in the path length), the message length and the useful utilization. For the circuit-switched system, the number of channels per trunk is also an important parameter. While this study confirmed the previous understandings, it also quantitatively showed the effect of some parameters (e.g., the path length) which have so far been disregarded. Based on our studies, when the number of channels per trunk in CS is properly selected, CS is a better choice of switching system than MS and/or CTS at low utilizations or large message lengths. When the communication path is long, again, CS outperforms the other two switching systems. In general, the decision on the choice of switching method depends on the parameters of the system, and no definite quantitative statement in this regard can be made. Our study in Section 3.6 showed that the best choice is to integrate different switching techniques in a single network load so that according to the user demands and the network load, a proper switching mode can be selected to provide the best grade of service.

CHAPTER 4

STATIC FLOW CONTROL IN STORE-AND-FORWARD COMPUTER NETWORKS

The last two chapters were concerned with the switching problems in computer-based communication networks. In this chapter and the next we focus on the analysis of flow control techniques in such networks.

A computer network is a collection of resources to be used by competing users. Resources of such networks can be categorized in two classes [POUZ 76B]:

- Basic: buffers, transmission bandwidth, processor time
- Incidental: name space, table entries, logical channels, etc.

"Users" are the messages to be transmitted. The collection of resources has a limited capacity which causes conflicts to occur among the users of the system. These conflicts may result in a degradation of performance of the system to the point that the system becomes clogged and the throughput goes to zero. This behavior is typical of "contention" systems which are described by Agnew [AGNE 74] as follows:

> "Congestion-prone systems include queues, where long waiting times accompany nearly maximal throughput, and 'contention' systems where the throughput goes to zero and the waiting time to infinity for large enough loads. In a queue-like system congestion is typically the result of uneven or uncertain behavior by users and servers. While this behavior is also present in a contention system, additional user-user and user-server interaction help to cause the declining throughput..."

Networks cannot afford to accept all the traffic that is offered without control; there must be rules which govern the acceptance of

traffic from outside and coordinate the flow inside the network. These rules are commonly known as flow control procedures. A more precise definition of flow control is given by Pouzin [POUZ 76B]:

Definition 4.1

> Flow control is the set of mechanisms whereby a flow of data can be maintained within limits compatible with the amount of available resources.

In order to keep the network traffic within desirable limits, flow control, among other things, should be equipped with mechanisms to throttle some resources. The main throttling tools are [POUZ 76B]: stop-and-go, where a sender continues transmission until it is signalled not to; credit, where the sender receives an indication giving the amount of data it can transmit; rate, where the sender transmits traffic at a predetermined rate; delay, where the sender's transmissions are slowed down by excessive network delay; and class, where the messages carry with them a class indicator and in case of congestion certain classes are denied entry. These tools have been identified as the "basic throttling tools" as opposed to the "incidental throttling tools" which are imposed on incidental resources [POUZ 76B].

Existing control methods in store-and-forward communication nets can be classified as local control and global control. Local control is applied by a communication processor within the subnet on the basis of its own as well as its immediate neighbors' traffic data and resource utilization. This type of control is a direct consequence of limited buffer space in each node and is basically useful for prevention

of local congestion. If the users who chiefly contribute to an overload are distant from the point of congestion, local control methods require that congestion measures spread over long distances before effective measures are taken [DAVI 71]. Thus local control is not by itself sufficient to prevent congestion, and global control is necessary in order to further stop the input to the network well before the network is loaded to saturation. This control can be accomplished by limiting the number of packets which are simultaneously in the network. Existing methods of global flow control are: isarithmic flow control [DAVI 71] used in the NPL network, in which the total number of credits in a network are controlled and kept constant, and end-to-end flow control [MCQU 72] used in the ARPANET, where, basically, the total number of credits between two users are limited.

Most end-to-end flow control mechanisms use a variant of the credit throttling technique and are usually described in terms of window size [CERF 74], [BELS 75], where the number of unacknowledged messages (or packets) are limited to the "window size." A variant of this mechanism was originally used in the French Cyclades network [ZIMM 75] and the ARPANET VDH (Very Distant Host) connections [BBN 73].

End-to-end flow control is accomplished through inter-process communication protocols and any attempt to quantitatively study the former should start with the development of an analytic model for the latter. Several authors have discussed techniques for flow control at different levels [CARR 70], [DAVI 71], [KAHN 71], [WALD 72], [OPDE 74], [BELS 75], [CROW 75], [DANT 75A,B], [ZIMM 75], [KLEI 76B] and [POUZ 76B]. However, due to the complex multi-variate environment of flow control,

quantitative and analytic results have been lacking. Except for simulation studies of flow control [PRIC 73], [GIES 76] and [LELA 76], in most analytic works the authors have focused on the influence of the individual tools of flow control on the overall performance [PENN 75], [SUNS 75], [CHAT 76], [CHOU 76], [FAYO 77] but in general, due to the complexity of the problem, the interaction of these tools has not been analytically studied.

The purpose of this chapter is to develop analytic models for end-to-end communication protocols and to study the window mechanism for flow control in store-and-forward computer based communication networks. We start with a very simple, deterministic model in which there are no stochastic fluctuations in the load on the system. Then, step-by-step, we improve this basic model and develop a static flow control model in which the parameters of the system are not dynamically adjusted to the stochastic fluctuations of the system load. We will use the static flow control results in the next chapter to develop a dynamic flow control system.

4.1 The Model

As in the previous chapters, we continue to assume that messages consist of single packets; hence we do not differentiate between message switching and packet switching. Because the present study is concerned with the analysis of end-to-end protocols, we consider a communication session between a source and destination (say nodes A and B) in a communication network. The environment under consideration is shown in Fig. 4.1. In this figure the traffic controller (TC) in the source node

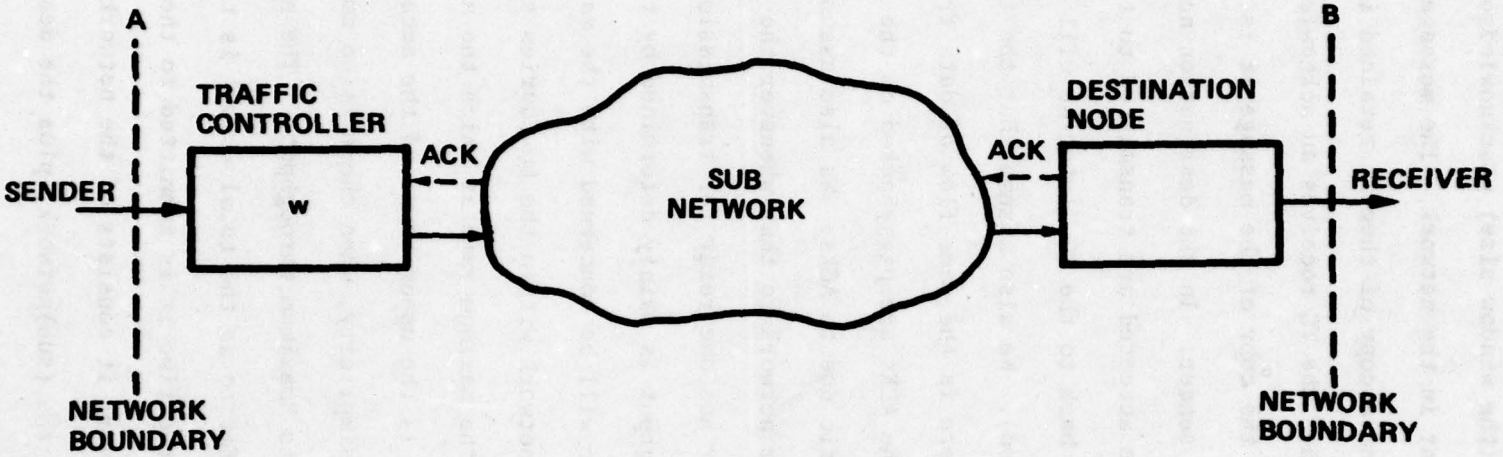


Figure 4.1 - Structure of a Network.

is responsible for regulating the traffic from node A to node B. It allows a maximum of w (the window size) unacknowledged messages from the sender to be present in the network. The messages are sent toward the destination node and a copy of them is retained in the buffer space provided by the TC. When the TC receives an acknowledgement (ACK) for a message, it discards the copy of the message it is storing and accepts a new message from the sender. In the destination node messages which are received and can be accepted are transmitted to the receiver and ACKs for them are sent back to the TC (later we will explain why a message may not be accepted). We also assume that the traffic on the path is symmetric (i.e., there is the same flow of data from the receiver to the sender) and that the ACKs are piggybacked on the messages, hence there is no extra traffic due to ACKs. We also assume that the sender is fast compared to the network so that whenever the TC can accept a new message, the sender has one ready for transmission. With these assumptions, the throughput is mainly determined by the network and throughout our study we will be concerned with the maximum throughput-delay behavior of the network within the boundaries shown in Fig. 4.1. Maximum throughput is the maximum rate at which the network can accept new messages (hence it is the upper bound of the acceptable input rate to the network). For simplicity, when there is no ambiguity, we use "throughput" to indicate "maximum throughput." The end-to-end delay (which we will also refer to as the total delay) is the delay a message experiences from the time it is submitted to the TC until it is delivered to the receiver; it consists of the network delay, the delay incurred in the TC and the (sub)network, plus the destination node delay

(note that the end-to-end delay does not include the admission delay to the system). Another delay measure which is important in our study is the acknowledgement delay (T_a) which is the period from the time a message is delivered to the (sub)network until its ACK is received by the TC. We point out that the ACK delay is equal to the round-trip delay only if the message transmission is successful. If a message is lost in the network (due to a transmission error and/or rejection at the destination node; see Section 4.4.1) no ACK for the message is received (hence T_a, the acknowledgement delay, is infinite).

We have not specified the timeout and certain other mechanisms of the system. We elaborate on these operations as the need arises in later sections. In a store-and-forward computer network, since packet sequences may be received out of order, the destination node must be equipped with a mechanism to deliver packets or messages in correct order. This mechanism can be implemented in different ways: the destination node may reject out-of-order packets or messages; or it may store the out-of-order packets and messages, but deliver them in the original order to the receiver. The first solution results in a waste of network channel bandwidth (because of retransmissions), whereas the second solution results in an excessive buffer storage requirement at the destination node. The sequencing problem is more important for the case of multi-packet messages. Because of our assumption of single packet messages, the sequencing problem is not as critical in our study and we will ignore this issue. In [SUNS 75] the effect of sequencing on the network performance has been partially investigated. Lastly, due to the single packet message assumption, there will not be any case of reassembly

deadlock in the system we study and, in rare situations, only store-and-forward deadlock may occur [KAHN 71], [OPDE 74]; therefore, we will ignore the deadlock problem as well.

4.2 A Fluid Approximation

We begin our study with the analysis of a purely deterministic model. We assume that the random variables involved are deterministic, hence this model represents a fluid approximation of the original system [NEWE 68], [KLEI 76B]. We also assume that the network is completely reliable, which means that the channels are noiseless and for every message delivered to the (sub)network by the TC, an ACK is received (with probability one). We use the following notation:

T_a	ACK delay (sec)
$1/\mu$	Message length (bits)
C_1	Network channel capacity (bps)
X_1	Message transmission time on a channel ($= 1/\mu C_1$) (sec)
γ_1	Maximum discharge rate of messages ($= \mu C_1$) (msg/sec)
λ^*	Maximum throughput (or acceptance rate) of the network (msg/sec)
λ	Input rate of messages to the network (msg/sec)
w	Window size (msg)

Consider Fig. 4.2; if at time t_0 a message is delivered to the network, its ACK will be received at time $t_1 (= t_0 + T_a)$. Because every T_a seconds one message is accepted by the network, for a window size equal to 1 ($w = 1$), the maximum throughput becomes $1/T_a$ and in general for a window size of w the maximum throughput is w/T_a. Fig. 4.3 shows a sketch of the maximum throughput as a function of the window size.

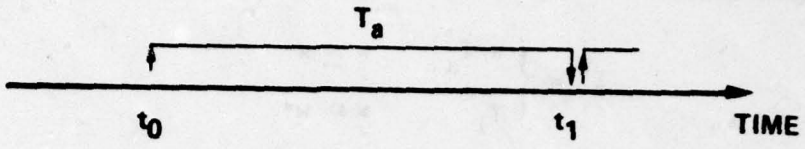


Figure 4.2 - Transmission Interval in a Network When Window Size is 1.

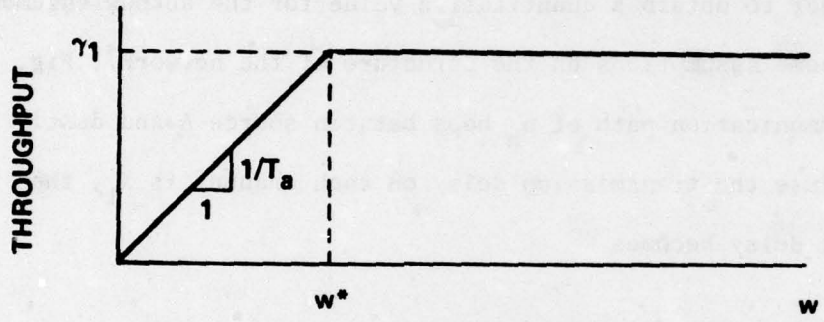


Figure 4.3 - Diagram of Network Throughput as a Function of Window Size (Deterministic Model)

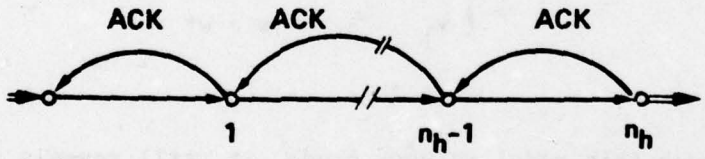


Figure 4.4 - A Communication Path

Because the throughput cannot exceed γ_1, the maximum discharge rate, the curve levels off when the window reaches a critical size (w*). Summing up, we have the following relationship between the maximum throughput and the window size:

$$\lambda^* = \begin{cases} w/T_a & w \leq w^* \\ \gamma_1 & w > w^* \end{cases} \qquad (4.1)$$

where $w^* = T_a\gamma_1$. Note that λ^* is the maximum throughput that the network can carry; the input rate (λ) is bounded above by λ^*.

In order to obtain a quantitative value for the acknowledgement delay we make some assumptions on the structure of the network. Fig. 4.4 shows a communication path of n_h hops between source A and destination B. Because the transmission delay on each channel is X_1, the acknowledgement delay becomes

$$T_a = 2n_hX_1 = 2n_h/\gamma_1$$

and Eq. (4.1) reduces to

$$\lambda^* = \begin{cases} w\gamma_1/2n_h & w \leq w^* \\ \gamma_1 & w > w^* \end{cases} \qquad (4.2)$$

where $w^* = 2n_h$.

Although this model is very crude, it still reveals some interesting characteristics of the system. In particular it shows that the traffic and the throughput of the system can be controlled by the window mechanism. Furthermore, it is not beneficial to increase the window size beyond a certain value. Considering the fact that a large window size means a large number of buffers in the traffic controller and that the buffer storage is a cost item in the overall network cost, the

uselessness of a large window size becomes clear. Regarding the best window size, w*, Fig. 4.3 shows w should be chosen such that there is one message on each channel (or node); in other words, one should try to fill the pipe of channels along the path.

4.3 A Simplified Stochastic Analysis

We improve the accuracy of the model developed in the last section by permitting some fluctuations in the arrival process and message lengths. We accept the following assumptions:

Assumption 4.1

1. External Poisson arrivals to the traffic controller (with rate λ)
2. Exponentially distributed message lengths (with mean $1/\mu$)

Although external traffic is accepted to the network by the traffic controller's demand and the data stream is not continuous, we assume that the overall process of the external message arrival to the TC is Poisson. Section 2.6.1 contains a detailed justification of the second assumption.

Except for the above, we continue to accept the assumptions made in the last section. In particular we assume that the network is reliable, that the channels are noiseless and that the destination node accepts all of the received messages; hence no retransmission is necessary (more about retransmission in Section 4.4).

When the flow in the network is stabilized, the input rate to all of the nodes (Fig. 4.4) becomes identical. By accepting the independence assumption of Kleinrock [KLEI 64], we are in a position to find

the acknowledgement delay (which is the same as the round-trip delay) and the throughput. By virtue of the independence assumption each channel on the path behaves like an M/M/1 queueing system. There are $2n_h$ channels on a round-trip, therefore

$$T_a = \frac{2n_h}{\gamma_1 - \lambda} \qquad (\gamma_1 = \mu C_1) \tag{4.3}$$

where $1/(\gamma_1 - \lambda)$ is the average system time of an M/M/1 system with service rate γ_1 and input rate λ. On the average each T_a seconds a maximum of w messages can be accepted to the system; therefore, the maximum input rate becomes

$$\lambda^* = \frac{w}{T_a^*} \tag{4.4}$$

For an input rate $\lambda = \lambda^*$, we can solve Eqs. (4.3) and (4.4) for λ^* to get

$$\lambda^* = \frac{w}{w + 2n_h}\ \gamma_1 \tag{4.5}$$

and also

$$T_a^* = \frac{w + 2n_h}{\gamma_1} \tag{4.6}$$

The network delay (which is the delay of the one-way trip from the TC to the destination node) is half of the round-trip delay, so

$$T^* = \frac{w + 2n_h}{2\gamma_1} \qquad \gamma_1 > 0 \quad \text{and} \quad w > 0$$

When the window size (w) and/or channel transmission rate (γ_1) is zero there is no traffic flowing through the network and we define the delay to be zero; therefore we have

$$T^* = \begin{cases} \dfrac{w + 2n_h}{2\gamma_1} & \gamma_1 > 0 \quad \text{and} \quad w > 0 \\ 0 & \gamma_1 = 0 \quad \text{or} \quad w = 0 \end{cases} \tag{4.7}$$

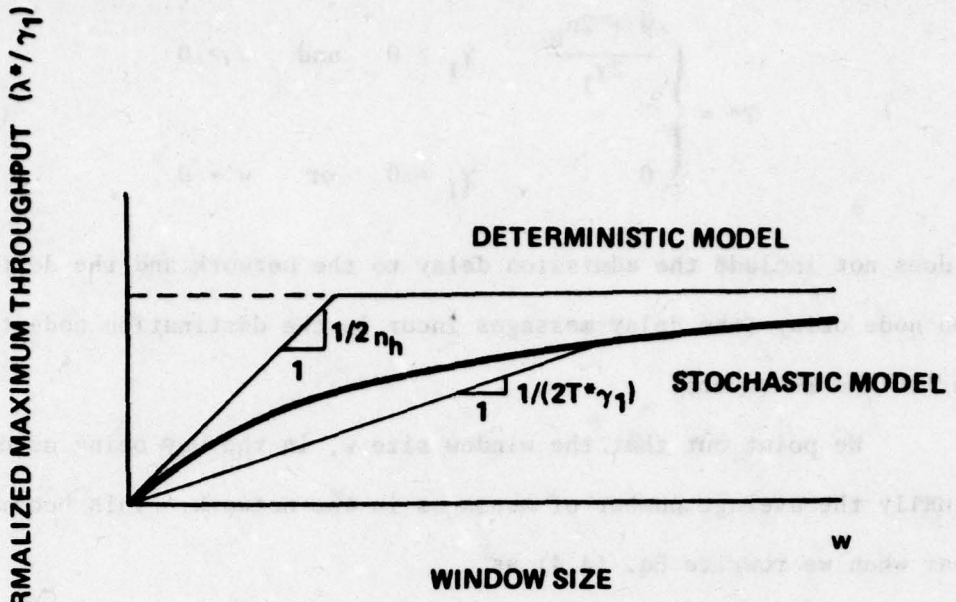
T^* does not include the admission delay to the network and the destination node delay (the delay messages incur in the destination node to be sent to the receiver).

We point out that the window size w, in the way being used, is actually the average number of messages in the network. This becomes clear when we rewrite Eq. (4.4) as

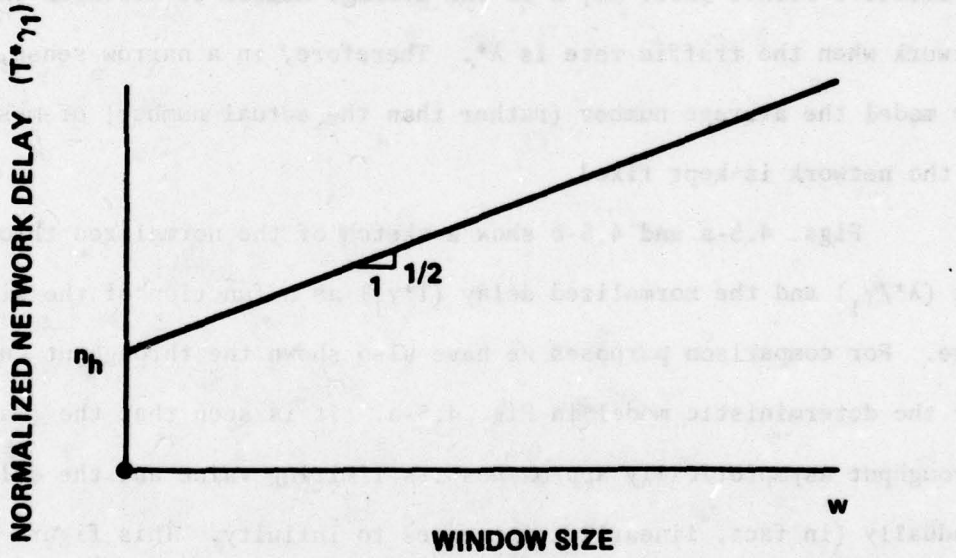
$$w = \lambda^* T_a^*$$

By Little's result [LITT 61] w is the average number of messages in the network when the traffic rate is λ^*. Therefore, in a narrow sense, in our model the average number (rather than the actual number) of messages in the network is kept fixed.

Figs. 4.5-a and 4.5-b show a sketch of the normalized throughput (λ^*/γ_1) and the normalized delay ($T^*\gamma_1$) as a function of the window size. For comparison purposes we have also shown the throughput curve for the deterministic model in Fig. 4.5-a. It is seen that the (maximum) throughput asymptotically approaches its limiting value and the delay gradually (in fact, linearly) increases to infinity. This figure depicts the effectiveness of the window mechanism in controlling the traffic more explicitly. Because the number of unacknowledged messages is limited to w, when the network delay increases (because of stochastic fluctuation), the input rate is decreased (Eq. 4.4) to keep the total



(a) Maximum Throughput



(b) Network Delay

Figure 4.5 - Sketch of Maximum Throughput and Delay as a Function of Window Size

number of messages in the system constant. This fact becomes more clear if we consider the maximum allowable window size for a given delay. From the analysis of Section 4.2 we know

$$w_{max} = \frac{T_a^*}{\bar{X}_1}$$

$\bar{X}_1$ being the average transmission time of a message on a channel when there is no queueing delay. Using the value of T_a^* as given by Eq. (4.6) and considering the fact that $\bar{X}_1 = 1/\mu C_1 = 1/\gamma_1$, we have

$$w_{max} = w + 2n_h$$

This last relationship indicates that the window size is always less than the maximum allowable window, hence the system never "saturates," i.e., channel utilization never reaches 1 (of course, as long as $w < \infty$).

The expression for the network delay, Eq. (4.7), can be rewritten as

$$T^* = \frac{n_h}{\gamma_1} + \frac{w}{2\gamma_1} = n_h\bar{X}_1 + \frac{w}{2}\bar{X}_1$$

The first term, $n_h\bar{X}_1$, represents the transmission delay and the second term can be viewed as the queueing delay caused by the background traffic. Note that w is the total number of messages on the path, hence w/2 is the number of messages that are being sent from node A to node B. We can also rewrite Eq. (4.5) as follows:

$$\lambda^* = \frac{\ell^*}{\ell^* + 2}\ \gamma_1$$

where $\ell^* = w/n_h$ is a measure of the load in a node (more about that in Chapter 6). We note that if w and n_h both vary such that the ratio w/n_h

does not change, the throughput of the network does not vary either; however, the network delay changes. This fact shows that from the throughput point of view, all networks with the same ℓ^* are identical.

λ^* and T^* given by Eqs. (4.5) and (4.7) are the maximum throughput and the corresponding network delay that the network can carry when the window size is w. For an input rate of $\lambda \leq \lambda^*$ the network delay becomes

$$T = \frac{n_h}{\gamma_1 - \lambda} \qquad 0 \leq \lambda \leq \lambda^* \tag{4.8}$$

If the input rate is larger than λ^*, then the extra traffic is not admitted to the net. Fig. 4.6-a shows a sketch of the maximum throughput as a function of w, and Fig. 4.6-b shows the throughput-delay behavior of the system. If the window size is set to w_1, then the maximum throughput is λ_1^*. When the input rate is $\lambda < \lambda_1^*$ (therefore, not all of the window buffers are used), the network delay is determined by Eq. (4.8). For larger window sizes the maximum input rate increases (w^* and w_2 in Figure 4.6-a), and the maximum network delay increases as well (Fig. 4.6-b).

The question which now arises is, what is the best choice for the window size. The deterministic model of Section 4.2 indicated that a window size of $w > 2n_h$ gives the best throughput, and that the best choice for w is $2n_h$. In order to answer this question for the stochastic model of this section, we should first define our measure of "goodness." If we favor high throughput, then a large window size is the best, but this results in a large delay as well (Figs. 4.5-a and 4.5-b). On the other hand, if delay is our measure of performance, then a window of

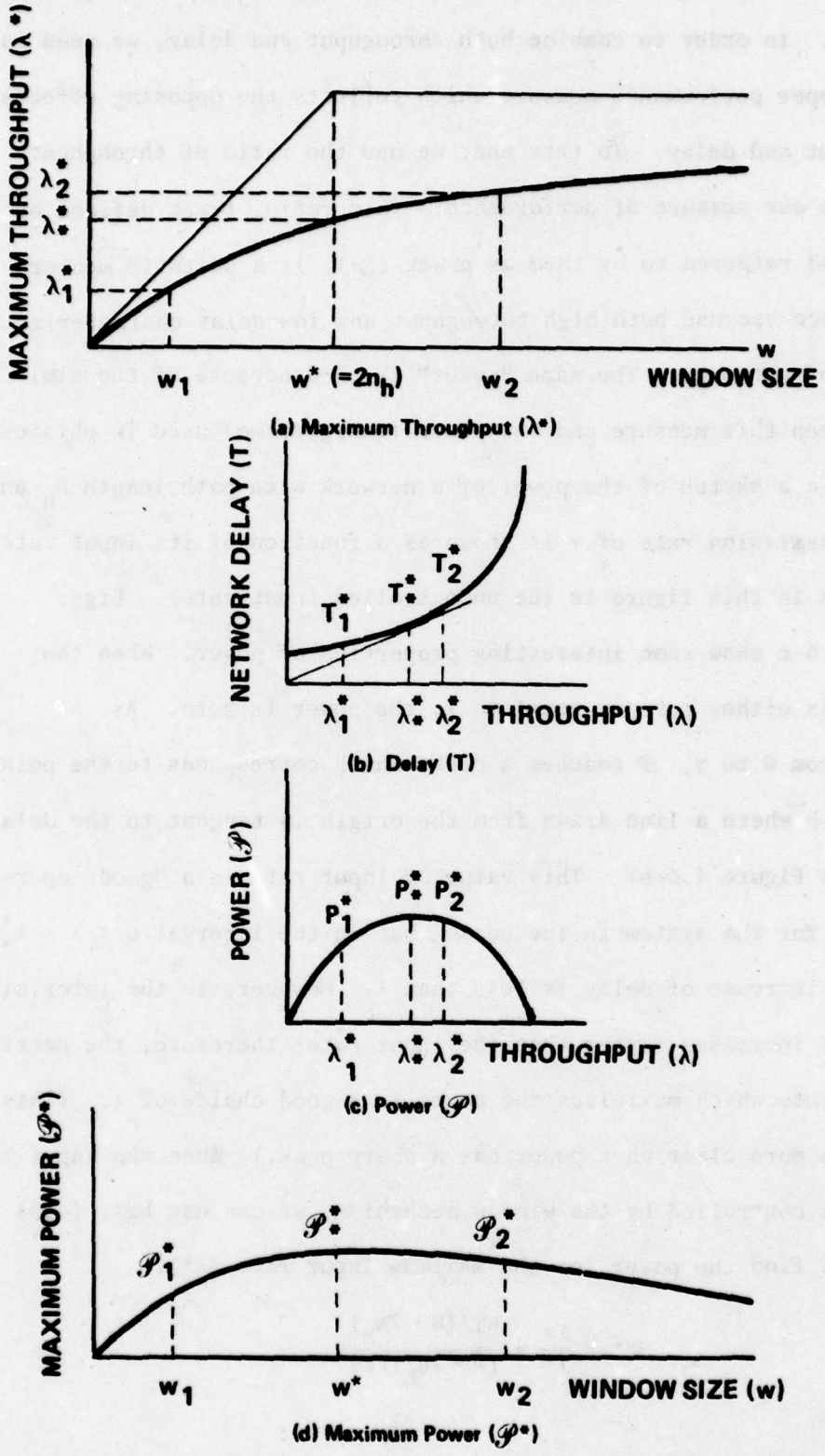


Figure 4.6 - Sketch of Throughput, Delay and Power of a Network.

size 0 is the best choice (the throughput for this window size is, however, zero). In order to combine both throughput and delay, we need to define a proper performance measure which reflects the opposing effects of throughput and delay. To this end, we use the ratio of throughput and delay as our measure of performance. This ratio, first defined by [GEIS 76] and referred to by them as power ($\mathscr{P}$), is a suitable measure of performance because both high throughput and low delay characterize efficiency of networks. The name "power" is used because of the similarity between this measure and the power (energy/time) used in physics. In Fig. 4.6-c a sketch of the power of a network with path length n_h and channel transmission rate of γ is shown as a function of its input rate. (Note that λ in this figure is the uncontrolled input rate). Figs. 4.6-b and 4.6-c show some interesting properties of power. When the input rate is either zero or equal to γ, the power is zero. As λ increases from 0 to γ, $\mathscr{P}$ reaches a peak, which corresponds to the point in Fig. 4.6-b where a line drawn from the origin is tangent to the delay curve (λ^*_* is Figure 4.6-b). This value of input rate is a "good" operating point for the system in the sense that in the interval $0 \leq \lambda < \lambda^*_*$ the rate of increase of delay is less than λ. However, in the interval of $\lambda^* < \lambda$, T increases faster than the input rate; therefore, the particular input rate which maximizes the power is a good choice of λ. (This fact becomes more clear when power has a sharp peak.) When the input to a network is controlled by the window mechanism, we can use Eqs. (4.5) and (4.7) to find the power for the maximum input rate (λ^*).

$$\mathscr{P}^* \triangleq \frac{\lambda^*}{T^*} = \frac{w\gamma/(w + 2n_h)}{(w + 2n_h)/2\gamma}$$

or

$$\mathcal{P}^* = \frac{2\gamma^2 w}{(w + 2n_h)^2}$$

Fig. 4.6-d shows a sketch of $\mathcal{P}^*$ as a function of window size. It is seen that at zero window size the power is also zero. As w increases, the power reaches a peak and then decreases; in the limit, when $w \uparrow \infty$, $\mathcal{P}^*$ goes to zero. It is easy to show that $\mathcal{P}^*$ reaches its peak when w is equal to

$$w^* = 2n_h$$

Therefore, similar to the deterministic case, the best window size (when power is the measure of performance) is twice the number of hops.

In the opening remarks of this chapter we pointed out that a computer network is essentially a contention system in which the throughput goes to zero as the load of the system increases. The reason why the models which we have used so far don't reveal this phenomonen is because we have assumed the network is reliable and messages are not lost in the network. That is the principal deficiency with these models. Although the model which we have used in this section does not show this throughput degradation behavior, it is very useful in that it provides us with a closed form solution for the network throughput and delay. As we will see in Section 4.4, for a more improved model which shows the contention behavior we cannot find these quantities explicitly. In a later chapter we use these closed form solutions to develop a network algebra for a systematic study of series and parallel interconnections of networks.

4.4 A Generalized Stochastic Model

To improve the model of Section 4.3, we make futher assumptions regarding the operation of different parts of a network. More specifically we elaborate on the following issues:

- The acknowledgement and timeout strategy
- The (sub)network delay distribution
- The destination node structure

4.4.1 The Acknowledgement and Timeout Strategy

So far, we have assumed that once the TC delivers a message to the (sub)network, the message is guaranteed to be received and accepted by the destination node buffer and is acknowledged for correct reception. This assumption implies that a message has to be transmitted only once, hence the input rate of external traffic to the TC is equal to the output rate of messages from it. In real life there are always chances that the acknowledgement for a message does not return. This happens for a variety of reasons, some of which we list below:

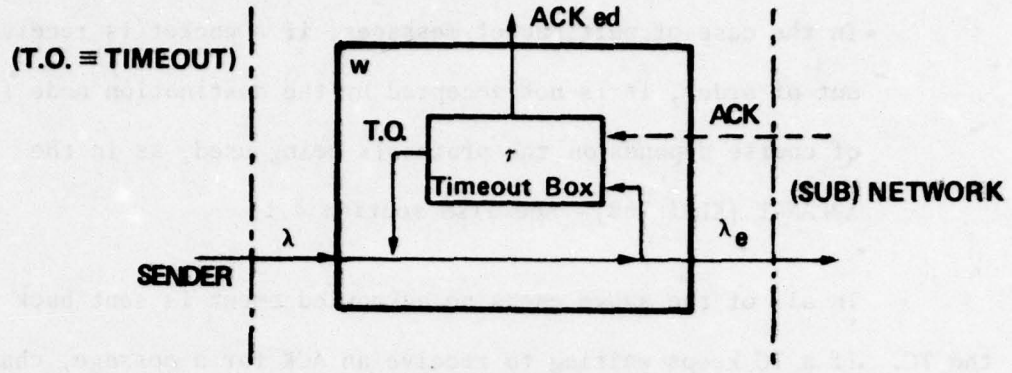
- Network channels are not noiseless and a message may arrive in the destination node with some bits in error. Such messages are not accepted by the destination node.

- Communication processors in a network have finite buffer storage for store-and-forward (in the intermediate nodes) and reassembly (in the destination node). When the buffer storage is full, no further messages can be accepted.

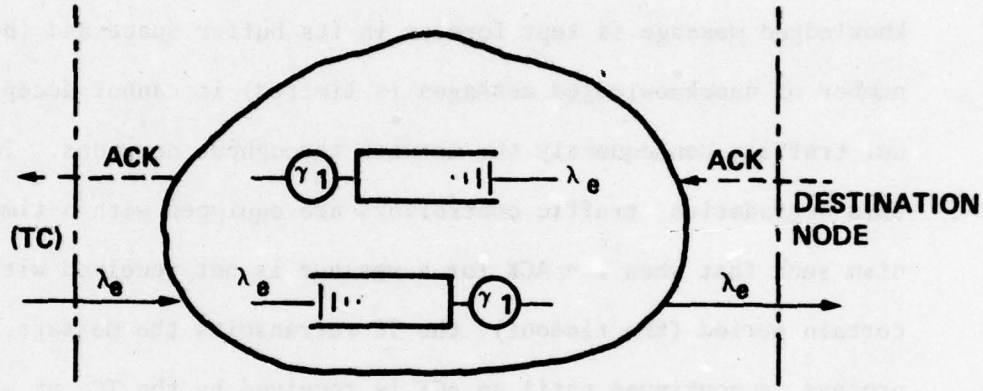
- In the case of multipacket messages, if a packet is received out of order, it is not accepted by the destination node (this of course depends on the protocols being used, as in the ARPANET [KLEI 76B] - See also section 4.1).

In all of the above cases no acknowledgement is sent back to the TC. If a TC keeps waiting to receive an ACK for a message, chances are that it will never receive such an ACK and the copy of the unacknowledged message is kept forever in its buffer space and (because the number of unacknowledged messages is limited) it cannot accept new external traffic; consequently the network throughput degrades. To prevent this degradation, traffic controllers are equipped with a timeout mechanism such that when the ACK for a message is not received within a certain period (the timeout), the TC retransmits the message. This process is continued until an ACK is received by the TC, at which time the copy of the message is dropped from the traffic controller's buffer and a new message can be accepted. Fig. 4.7-a shows the structure of the TC. After delivering a message to the (sub)network, the TC stores a copy of the message in its timeout box. If an ACK for this message is received within τ seconds (the timeout period), the message is dropped; else the message is retransmitted. As shown in this figure, the delivery of messages from the TC to the (sub)network is instantaneous and there is no queueing delay involved. Because of retransmission, the acceptance rate of messages from outside (λ) is different from (and less than) λ_e, the message delivery rate to the (sub)network.

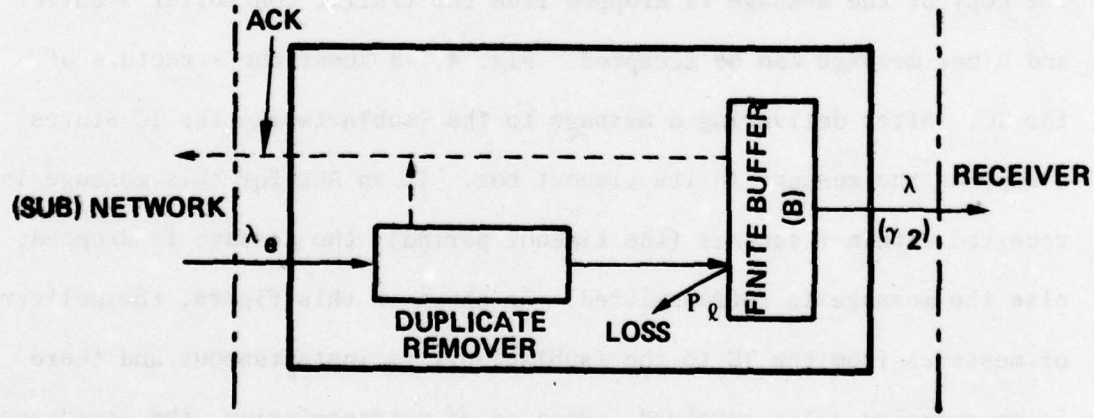
We should point out that if an ACK does not arrive during the timeout period, that does not imply that it will never be received.



(a) Traffic Controller (TC)



(b) (Sub) Network



(c) Destination Node

Figure 4.7 - Structure of Network Components.

Also, due to the randomness of the round-trip delay, ACK's for different (re)transmissions of a message may return out of order. Consider the following cases:

I. After the first timeout the TC retransmits the message, and while it is waiting for the second acknowledgement the ACK for the first transmission comes back. At this time the copy of the message should be dropped and a new message can be accepted.

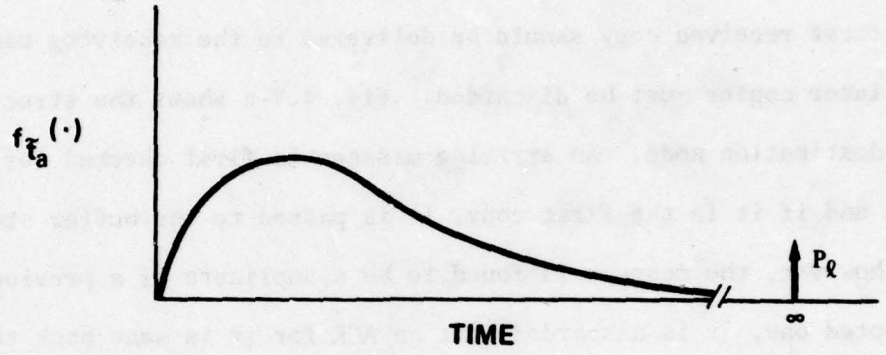
II. The ACK for a second retransmission may arrive before the ACK for the first one. Again, at this time the TC can accept a new message to the network.

In fact, reception of an ACK for a message signifies its correct delivery to the destination node. To simplify the analysis we require that the ACK for the last (re)transmitted message should be received so that the TC can accept a new message to the network. This means, once an ACK is not received within the timeout period, we assume the transmission has failed and any ACK received after the timeout period is ignored. As a consequence of this assumption, in the first case given above the ACK is ignored, but in the second case it is considered as an indication of the correct delivery of the message. The reason for this simplifying assumption is that modeling of the exact operation is extremely difficult and the analysis is unnecessarily complicated. One way of modeling the exact operation is to tag messages with a class indicator. Newly arrived messages are tagged as Class 1, and after each time out the class indicator is increased by 1. The resulting model can be analyzed as a network of queues with different classes of customers [BASK 75];

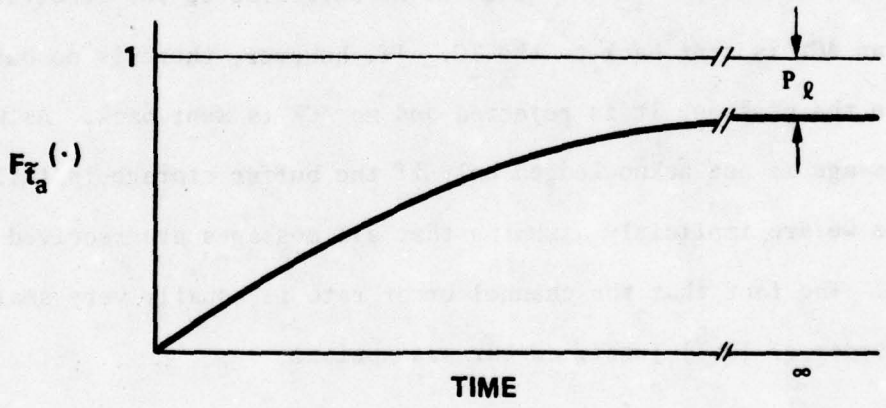
however, the analysis is fairly complicated and for this reason we accept the simplified assumption stated above in this chapter.

4.4.2 The (Sub)network Delay Distribution

A detailed specification of the structure of the (sub)network is not the object of our study; we view the (sub)network as a source of a random delay which we choose to represent by a proper distribution function. In [WONG 77] it is shown that the end-to-end delay in a Markovian network of queues is the sum of independent and exponentially distributed random variables. Simulation and measurement studies of existing networks [COLE 71], [KLEI 74B], [FORG 75], [KLEI 76B] show that a multistage Erlangian distribution is a good approximation for the round-trip delay distribution. Ideally, an Erlangian distribution of degree larger than 10 is the best choice; however, any degree larger than 2 makes the derivations and the integrations extremely complicated. Considering the fact that the protocol performance is relatively insensitive to the degree of the distribution ([SUNS 75], also our analysis with degrees = 1 and 2), we choose to use an Erlangian distribution of the second degree (concatenation of two exponential distributions) for the round-trip delay. We further assume that one one-way delay is similar to the system delay of a simple M/M/1 queueing system. Fig. 4.7-b shows the structure of the (sub)network used in our model. Because messages which contain errors, or which cannot be accepted at the destination buffer, are not acknowledged, the acknowledgement delay may be infinite. We present this fact in our model by including an impulse (of size P_ℓ, the probability of loss) in the ACK delay distribution at time equal to infinity. Fig. 4.8 shows a sketch of the density and the



(a) Density Function



(b) Distribution Function

Figure 4.8 - Diagram of the Density and Distribution Functions of Acknowledgement Delay.

distribution function of acknowledgement delay.

4.4.3 The Destination Node Structure

In Section 4.4.1 we assumed that the TC must receive an ACK for the last (re)transmitted message. This implies that the destination node may receive multiple copies of the same message. Clearly, only the first received copy should be delivered to the receiving user and the later copies must be discarded. Fig. 4.7-c shows the structure of the destination node. An arriving message is first checked for duplication and if it is the first copy, it is passed to the buffer storage. If, however, the message is found to be a duplicate of a previously accepted one, it is discarded, but an ACK for it is sent back to the sending TC. If there is space available at the buffer, the message is admitted and is put in the queue to be delivered to the receiving user and an ACK is sent back to the TC. If, however, there is no buffer to store the message, it is rejected and no ACK is sent back. As we note, a message is not acknowledged only if the buffer storage is full. This means we are implicitly assuming that all messages are received error free. The fact that the channel error rate is usually very small (on the order of 10^{-5}) justifies our assumption.

4.4.4 Analysis

We summarize the assumptions stated above as follows:

Assumptions 4.2

1. Poisson arrival of the messages to the (sub) network.
2. Exponential message length distribution (with mean $1/\mu$)
3. ACK for the last (re)transmitted message must be received.

4. Round-trip delay has an Erlangian distribution of degree two (summation of two exponential distributions) consisting of concatenation of two system delay distributions of an M/M/1 queue:

$$F_{\tilde{t}_r}(t) = 1 - e^{-(\gamma_1-\lambda_e)t} - (\gamma_1 - \lambda_e)te^{-(\gamma_1-\lambda_e)t}$$

where $\tilde{t}_r$ is a random variable representing the round-trip delay, λ_e is the (sub)network traffic rate and $\gamma_1 = \mu C_1$, C_1 being the network channel capacity.

5. Destination node has a finite number of buffers.

6. At the destination node only the first successful copy of a message is accepted; however, ACKs for later copies are sent back.

7. Negligible channel error rate.

8. No message sequencing (see Section 4.1)

We start our analysis by deriving the ACK delay distribution. Based on Assumption 4.2-4 the ACK delay distribution for successful transmission is

$$F_{\tilde{t}_a}(t \mid \text{no loss}) = 1 - e^{-(\gamma_1-\lambda_e)t} - (\gamma_1 - \lambda_e)te^{-(\gamma_1-\lambda_e)t}$$

where $\gamma_1 = \mu C_1$ and C_1 is the network channel capacity, λ_e is the effective network traffic (which includes retransmissions) and $\tilde{t}_a$ is a random variable representing the acknowledgement delay. We also have

$$F_{\tilde{t}_a}(t \mid \text{loss}) = 0$$

Therefore, the distribution of the acknowledgement delay becomes

$$F_{\tilde{t}_a}(t) = (1 - P_\ell)\left[1 - e^{-(\gamma_1-\lambda_e)t} - (\gamma_1 - \lambda_e)te^{-(\gamma_1-\lambda_e)t}\right] \quad (4.9)$$

and its density function will be

$$f_{\tilde{t}_a}(t) = (1 - P_\ell)(\gamma_1 - \lambda_e)^2 te^{-(\gamma_1-\lambda_e)t} \quad (4.10)$$

where P_ℓ is the probability that an ACK is not received.

Next we calculate the retranmission probability. Retransmission is necessary when either the message is lost at the destination buffer, or the transmission is successful but the acknowledgement delay is greater than the timeout.

$$\begin{aligned} P_r &= \Pr[\text{retransmission}] \\ &= P_\ell + (1 - P_\ell)\ \Pr[\tilde{t}_a > \tau \mid \text{no loss}] \end{aligned}$$

or

$$P_r = P_\ell + (1 - P_\ell)[1 + (\gamma_1 - \lambda_e)]e^{-(\gamma_1-\lambda_e)\tau} \quad (4.11)$$

Because of retransmissions, the effective network traffic is larger than λ, the input rate of messages to the network. They are related to each other as follows:

$$\lambda_e = \lambda(1 - P_r) + (\lambda + \lambda_e)P_r$$

or

$$\lambda_e = \frac{1}{1 - P_r}\lambda \quad (4.12)$$

We now consider the length of time a message occupies a buffer in the TC. We use a renewal theory argument to find this quantity.

Let

$$T_1 \triangleq E[\tilde{t}_a \mid \tilde{t}_a \leq \tau]$$

Then

$$T_{oc} \triangleq E[\text{time of occupancy}]$$
$$= T_1(1 - P_r) + (\tau + T_{oc})P_r$$

Solving this last equation for T_{oc}, we get

$$T_{oc} = T_1 + \frac{P_r}{1 - P_r}\tau$$

Calculation of T_1 is fairly easy. By definition, we have

$$T_1 = E[\tilde{t}_a \mid \tilde{t}_a \leq \tau] = \int t \, dF_{\tilde{t}_a|\tilde{t}_a \leq \tau}(t)$$

where

$$F_{\tilde{t}_a|\tilde{t}_a \leq \tau}(t) \triangleq Pr[\tilde{t}_a \leq t \mid \tilde{t}_a \leq \tau]$$

but

$$F_{\tilde{t}_a|\tilde{t}_a \leq \tau}(t) = \begin{cases} \dfrac{Pr[t_a \leq t]}{Pr[t_a \leq \tau]} & t \leq \tau \\ 0 & t > \tau \end{cases}$$

Using the above relationships and after some algebra we get

$$T_1 = \frac{2}{\gamma_1 - \lambda_e} - \frac{(\gamma_1 - \lambda_e)\tau^2 e^{-(\gamma_1 - \lambda_e)\tau}}{1 - [1 + (\gamma_1 - \lambda_e)\tau]e^{-(\gamma_1 - \lambda_e)\tau}}$$

and finally we find the value of T_{oc} as follows:

$$T_{oc} = \frac{2}{\gamma_1 - \lambda_e} + \frac{P_\ell + (1 - P_\ell)e^{-(\gamma_1 - \lambda_e)\tau}}{(1 - P_\ell)\{1 - [1 + (\gamma_1 - \lambda_e)\tau]e^{-(\gamma_1 - \lambda_e)\tau}\}}\tau \quad (4.13)$$

Summing the relationships we have derived so far, we have

$$P_r = P_\ell + (1 - P_\ell)(1 + \Lambda\tau)e^{-\Lambda\tau} \tag{4.11}$$

$$\lambda_e = \frac{1}{1 - P_r}\lambda = \frac{\lambda}{(1 - P_\ell)[1 - (1 + \Lambda\tau)e^{-\Lambda\tau}]} \tag{4.12}$$

$$T_{oc} = \frac{2}{\Lambda} + \frac{P_\ell + (1 - P_\ell)e^{-\Lambda\tau}}{(1 + P_\ell)[1 - (1 + \Lambda\tau)e^{-\Lambda\tau}]}\tau \tag{4.13}$$

where $\Lambda \triangleq \gamma_1 - \lambda_e$

Because a maximum of w (the window size) messages are accepted to the network every T_{oc} seconds, the maximum input rate will be

$$\lambda^* = \frac{w}{T_{oc}} \tag{4.14}$$

When the input rate is $\lambda = \lambda^*$, from Eq. (4.12) we have

$$\lambda_e = \frac{1}{1 - P_r}\frac{w}{T_{oc}} \tag{4.15}$$

We can use the values of P_r and T_{oc} from Eqs. (4.11) and (4.13) in Eq. (4.15) to get

$$\lambda_e = \frac{w\Lambda}{2(1 - P_\ell)[1 - (1 + \Lambda\tau)e^{-\Lambda\tau}] + [P_\ell + (1 - P_\ell)e^{-\Lambda\tau}]\Lambda\tau}$$

or in general

$$\lambda_e = G_1(\tau, w, \gamma_1, P_\ell, \lambda_e) \tag{4.16}$$

The unknowns in the above equation are λ_e and P_ℓ, the probability that an ACK is not returned. Because the channels are assumed

to be noiseless, P_ℓ is the probability that a message is rejected from the destination buffer. It follows from our previous assumptions that the input process of messages to the destination node is Poisson. However, because the duplicate messages are not stored in the buffer (but are acknowledged, see Fig. 4.7-c), the counting process of message arrivals to the buffer storage is no longer Poisson. In order to simplify the derivation of the rejection probability we make one more assumption regarding this process.

Assumption 4.2 (continued)

9. The counting process of message arrivals to the destination buffer is Poisson.

By virtue of Assumption 4.2-2, the destination buffer behaves like an M/M/1/B queueing system where B is the buffer size at the destination node. To calculate the loss probability P_ℓ, we notice that at equilibrium the input rate of messages to the network (λ) is equal to the output rate of messages from it. Hence the total arrival rate of messages to the buffer (the accepted messages plus the rejected ones) will be

$$\lambda_d = \frac{\lambda}{(1 - P_\ell)}$$

and by Eq. (4.12) we have

$$\lambda_d = \frac{1 - P_r}{1 - P_\ell} \lambda_e$$

and we have [KLEI 75]

$$P_\ell = \frac{\gamma_2 - \lambda_d}{\gamma_2^{B+1} - \lambda_d^{B+1}} \lambda_d^B \qquad (4.17)$$

where $\gamma_2 = \mu C_2$ and C_2 is the capacity of the output channel.

The above equation may be written as

$$P_\ell = G_2(\gamma_2, B, P_\ell, \lambda_e) \qquad (4.18)$$

The analytic results presented so far depend upon the solution of the two equations, (4.16) and (4.18). For a given value of the parameters (τ, w, γ_1, γ_2, and B) the above system of equations can be solved through an iterative procedure [HILD 66] and [ORTE 70].

Having found the network traffic λ_e, we can use Eq. (4.12) to find the maximum input rate (or the maximum throughput) of the network. The total end-to-end delay consists of two components: the network delay and the destination buffer delay. In calculating the network delay we should notice that only the delay of the first successful message is of importance to us; further retransmissions only effect the network traffic (hence have an indirect effect on the delay). The average network delay can be found through a renewal theory argument [COX 62] as follows:

$$T = \frac{1}{\gamma_1 - \lambda_e}(1 - P_\ell) + (\tau + T)P_\ell$$

In the above equation $1/(\gamma_1 - \lambda_e)$ is the one-way network delay (from the TC to the destination node) if no retransmission (due to rejection) is required. If a message is rejected, after τ seconds it is retransmitted (the transmission process renews). The second term corresponds to the

delay when retransmission (due to rejection) is necessary. Solving the above equation for T, we get

$$T = \frac{1}{\gamma_1 - \lambda_e} + \frac{P_\ell}{1 - P_\ell}\tau \qquad (4.19)$$

The other component of the total delay is the destination buffer delay. This is equivalent to the system delay of the corresponding M/M/1/B queue [KLEI 75]

$$T_{dst} = \frac{1/\mu C_2}{1 - \rho_d} \frac{1 - (1 + B)\rho_d^B + B\rho_d^{B+1}}{1 - \rho_d^B} \qquad (4.20)$$

where $\rho_d = \lambda_d/\mu C_2$ is the utilization factor of the destination buffer and C_2 is the output channel capacity.

Finally we have

$$T_{ee} = T + T_{dst} \qquad (4.21)$$

where T_{ee} is the end-to-end delay.

The analysis of this section has been based on the equilibrium of the system for a selected set of parameters (τ, w, μ, C_1, C_2, B) which are not dynamically adjusted to the stochastic fluctuations in the load of the system. We will refer to the mechanism analyzed above as "static flow control."

4.4.5 Numerical Results

In this section we present numerical examples for static flow control and study the effect of different parameters (w, τ and B) on the throughput-delay performance of a network. For a given set of parameters we can find the maximum throughput (λ^*) and the corresponding (maximum)

total delay (T^*_{ee}). Whenever there is no ambiguity, we will refer to these quantities simply by throughput and delay. Throughout our presentation we consider the normalized values of certain quantities; the normalization constant is $\overline{X}_1 = 1/(\mu C_1)$, the average transmission time of a message on a network channel. It also turns out that only the ratio C_1/C_2 is of importance to us, which we will use in this section.

We start by studying the effect of the timeout period on the network throughput. Fig. 4.9 shows the normalized throughput ($\lambda^*/\mu C_1$) as a function of the normalized timeout ($\tau/\overline{X}_1$) for different window sizes when the destination buffer size (B) is 10 and $C_1/C_2 = 1$. It can be seen that when the timeout is very small the throughput decreases to zero. The reason for this degradation is as follows: when the timeout is small, the TC retransmits messages very fast, hence the (sub)network channel utilization ($\lambda_e/\mu C_1$) approaches 1 and the network delay becomes unbounded; because the number of unacknowledged messages is limited, no new message is admitted to the network and the throughput degrades to zero (Eq. 4.14). It is interesting to note that in order for the system to remain stable, the normalized timeout should be larger than the window size. This fact is intuitively clear in the deterministic case; because the transmission of w (w is the window size) messages takes wX_1 seconds (X_1 being the transmission time of one message), a timeout of $\tau < wX_1$ causes retransmission of a message to start before transmission of the w messages can possibly be completed; as a result, the network becomes saturated. We can establish this fact analytically. For stability we require

$$\lambda_e < \gamma_1$$

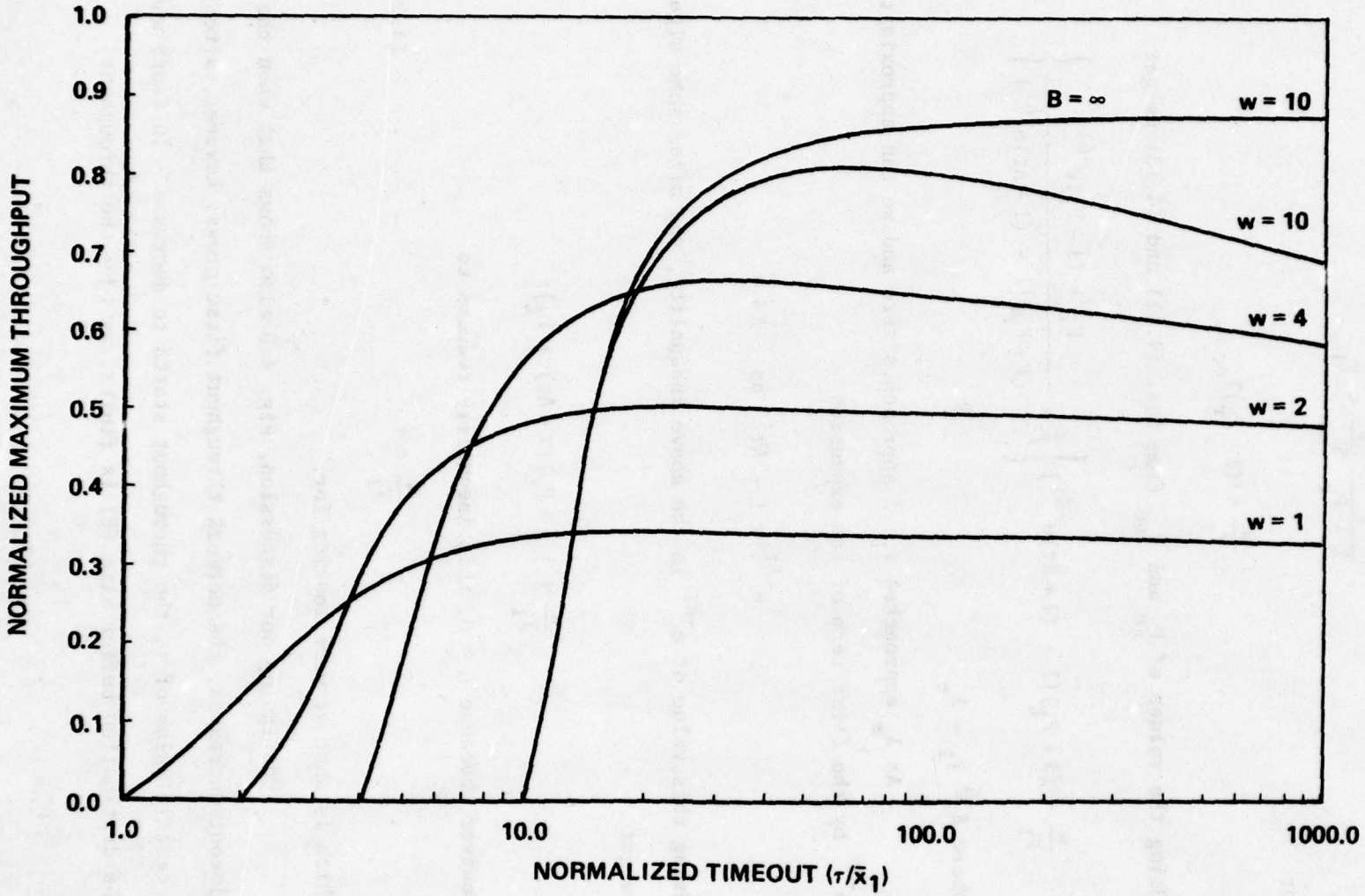


Figure 4.9 - Dependency of Network Throughput on Timeout ($B = 10$; $C_1/C_2 = 1.0$).

Using the value of λ_e from Eq. (4.15), the above inequality becomes

$$\frac{1}{1 - P_r} \frac{w}{T_{oc}} < \gamma_1$$

or

$$\frac{w}{\gamma_1} < (1 - P_r)T_{oc}$$

Using the values of P_r and T_{oc} from Eqs. (4.11) and (4.13) we get

$$\frac{w}{\gamma_1} < (1 - P_\ell)[1 - (1 + \Lambda\tau)e^{-\Lambda\tau}] \left\{ \frac{2}{\Lambda} + \frac{P_\ell + (1 - P_\ell)e^{-\Lambda\tau}}{(1 - P_\ell)[1 - (1 + \Lambda\tau)e^{-\Lambda\tau}]} \right\}$$

where $\Lambda \triangleq \gamma_1 - \lambda_e$

As λ_e approaches γ_1, Λ approaches zero and we can approximate $e^{-\Lambda\tau}$ by the first term of its expansion

$$e^{-\Lambda\tau} \simeq 1 - \Lambda\tau \quad \text{as} \quad \Lambda \downarrow 0$$

Using this value of $e^{-\Lambda\tau}$ in the above inequality, and after some algebra we get

$$\frac{w}{\gamma_1} < [(1 - P_\ell)(1 + \Lambda\tau) + P_\ell]\tau$$

However, because $\Lambda \simeq 0$, this inequality reduces to

$$\frac{w}{\gamma_1} < \tau \tag{4.22}$$

which is what we were looking for.

Continuing our discussion, Fig. 4.9 also shows that when the timeout increases, the network throughput first grows; however, after a certain value of τ, the throughput starts to decrease. In fact, when the destination buffer size (B) is finite, as $\tau \uparrow \infty$ the throughput

degrades to zero, i.e.,

$$\lim_{\tau \uparrow \infty} \lambda^* = 0$$

This fact is easily shown by using Eqs. (4.14) and (4.15)

$$\lim_{\tau \uparrow \infty} \lambda^* = \lim_{\tau \uparrow \infty} \frac{w}{T_{oc}} = \lim_{\tau \uparrow \infty} \frac{w}{\frac{2}{\Lambda} + \frac{P_\ell + (1 - P_\ell)e^{-\Lambda\tau}}{(1 - P_\ell)[1 - (1 + \Lambda\tau)e^{-\Lambda\tau}]}\tau} = 0$$

Fig. 4.9 shows that this throughput degradation is more severe for larger window sizes. The reason for this degradation is that for a large timeout the TC waits for a long time for an ACK to come back; however, because of the finite destination buffer size, there is a non-zero probability that an ACK never comes back, hence the buffer of the traffic controller becomes full of unacknowledged messages, and no further external traffic is accepted. We should point out that this degradation is not present if the destination buffer size is infinite; the throughput curve in Fig. 4.9 for $w = 10$ and the destination buffer size of infinity explicitly shows this fact. Fig. 4.9 shows that for a fixed window size, there is a value of timeout that optimizes (maximizes) the throughput. We will designate this optimal value by τ_t^* (as opposed to τ_d^*, which is the optimal value of timeout with respect to the end-to-end delay; see below). This figure shows that when the window size is small, the throughput curve, after an initial increase, remains quite flat, and only for large window sizes is the maximal throughput sensitive to timeout.

The effect of the timeout period on delay is shown in Figs. 4.10-a, 4.10-b and 4.10-c. Fig. 4.10-a shows the normalized (maximum)

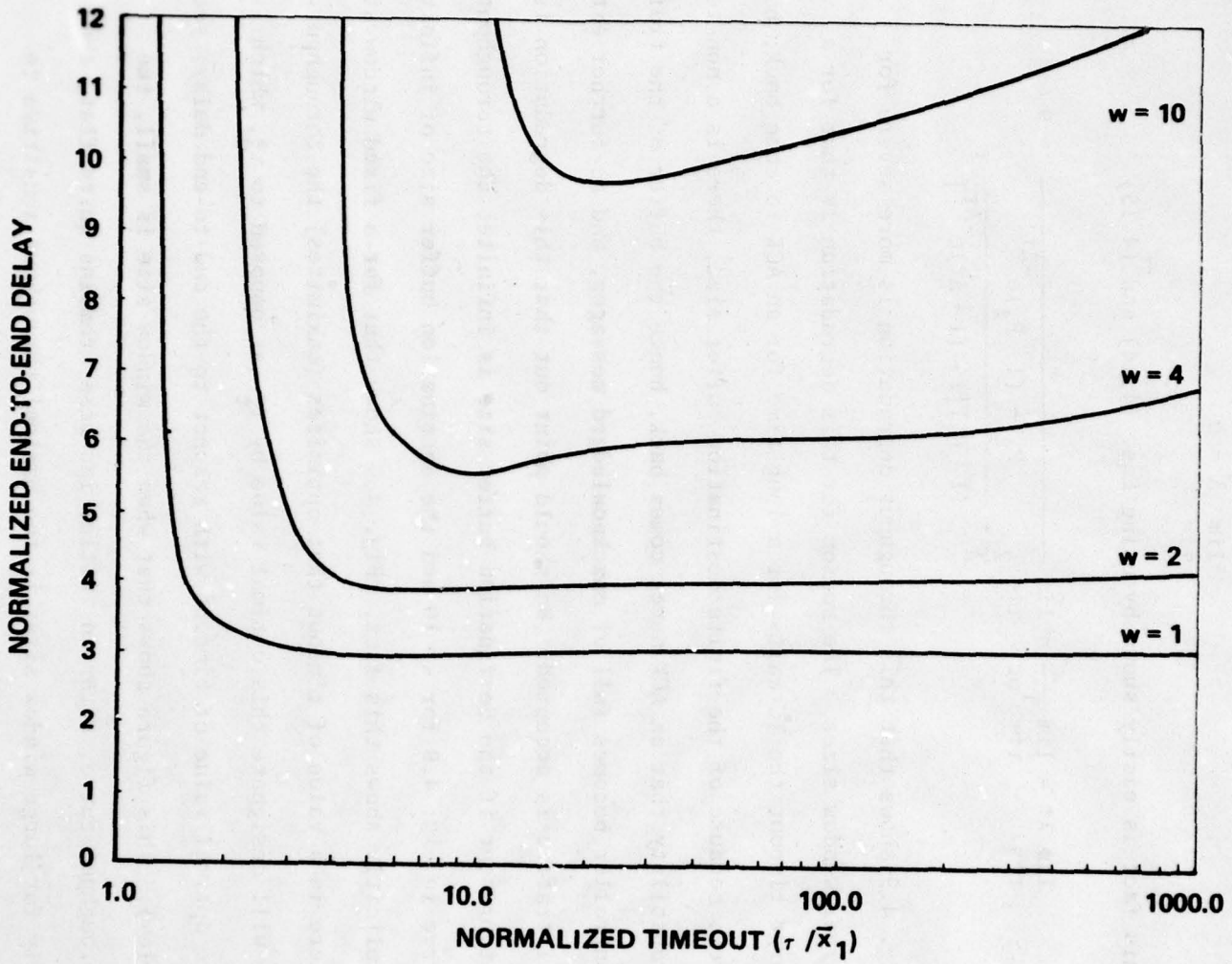


Figure 4.10-a End-To-End Delay as a Function of Timeout (B = 10; C_1/C_2 = 1.0).

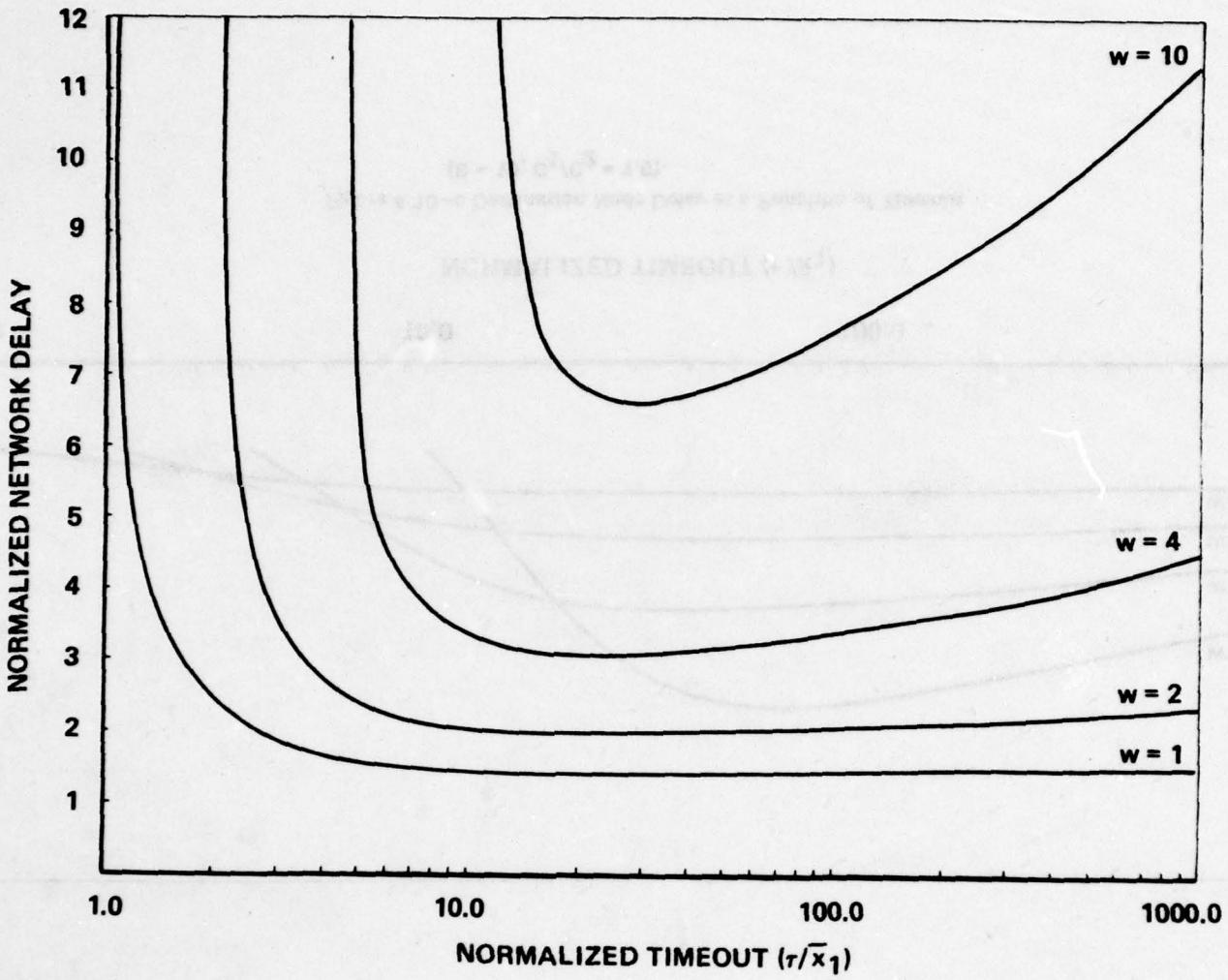


Figure 4.10-b Network Delay as a Function of Timeout
($B = 10$; $C_1/C_2 = 1.0$).

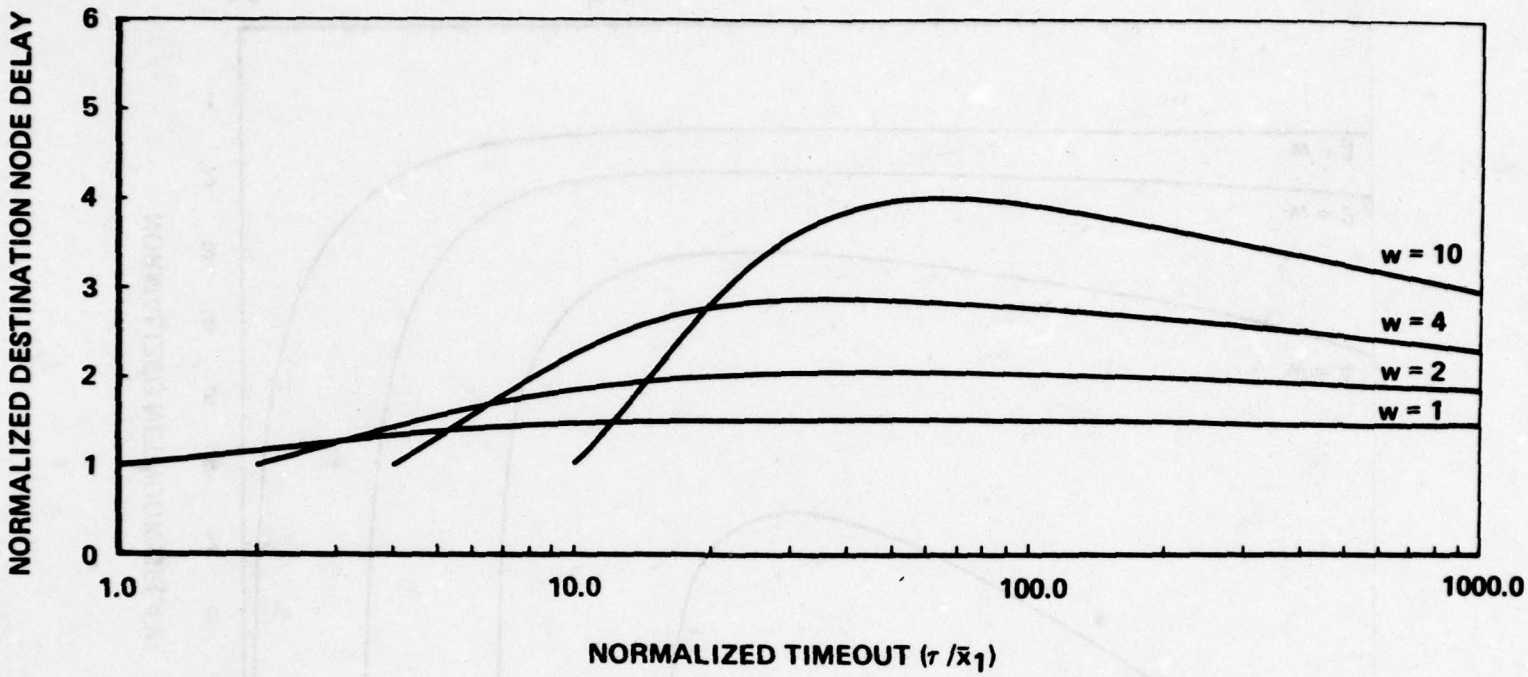


Figure 4.10 –c Destination Node Delay as a Function of Timeout (B = 10; C_1/C_2 = 1.0).

end-to-end delay ($T_{ee}/\overline{X}_1$) as a function of the normalized timeout for different window sizes (end-to-end delay, as defined in Section 4.4.4, is the sum of the network delay and the destination node delay). As in Fig. 4.9, this figure also shows that for a fixed window size, there is a value of timeout which optimizes (minimizes) the end-to-end delay; this optimal timeout (τ_d^*, d for delay) is usually different from τ_t^*. When the timeout is too small, like the throughput, the end-to-end delay becomes unbounded (the minimum allowable timeout is, of course, $w\overline{X}_1$); as the value of the timeout increases, the end-to-end delay first reaches a minimum and then becomes unbounded. As Figs. 4.10-b and 4.10-c show, this unboundedness is due to the network delay.

We now study the effect of the window size on the network performance. In Fig. 4.11 the normalized throughput is shown as a function of the window size. On the curve designated by "optimal throughput," for each window size the optimal value of timeout (τ_t^*) is chosen such that the throughput becomes maximal; whereas on the curve designated by "optimal delay," the optimal value of timeout (τ_d^*) is chosen so that the end-to-end delay becomes minimal. This figure also shows the throughput curves for several constant (normalized) timeouts. For a fixed timeout, the network throughput initially increases as w increases; however at a certain value of window size the throughput curve reaches its maximum and then falls down; finally as w becomes greater than $\tau/\overline{X}_1$ the curve goes to zero, which agrees with our previous result (Eq. 4.22). The throughput curves for constant timeout display the known behavior of a contention system (described in the opening of this chapter); here the window size represents the load on the system. Note that the optimal

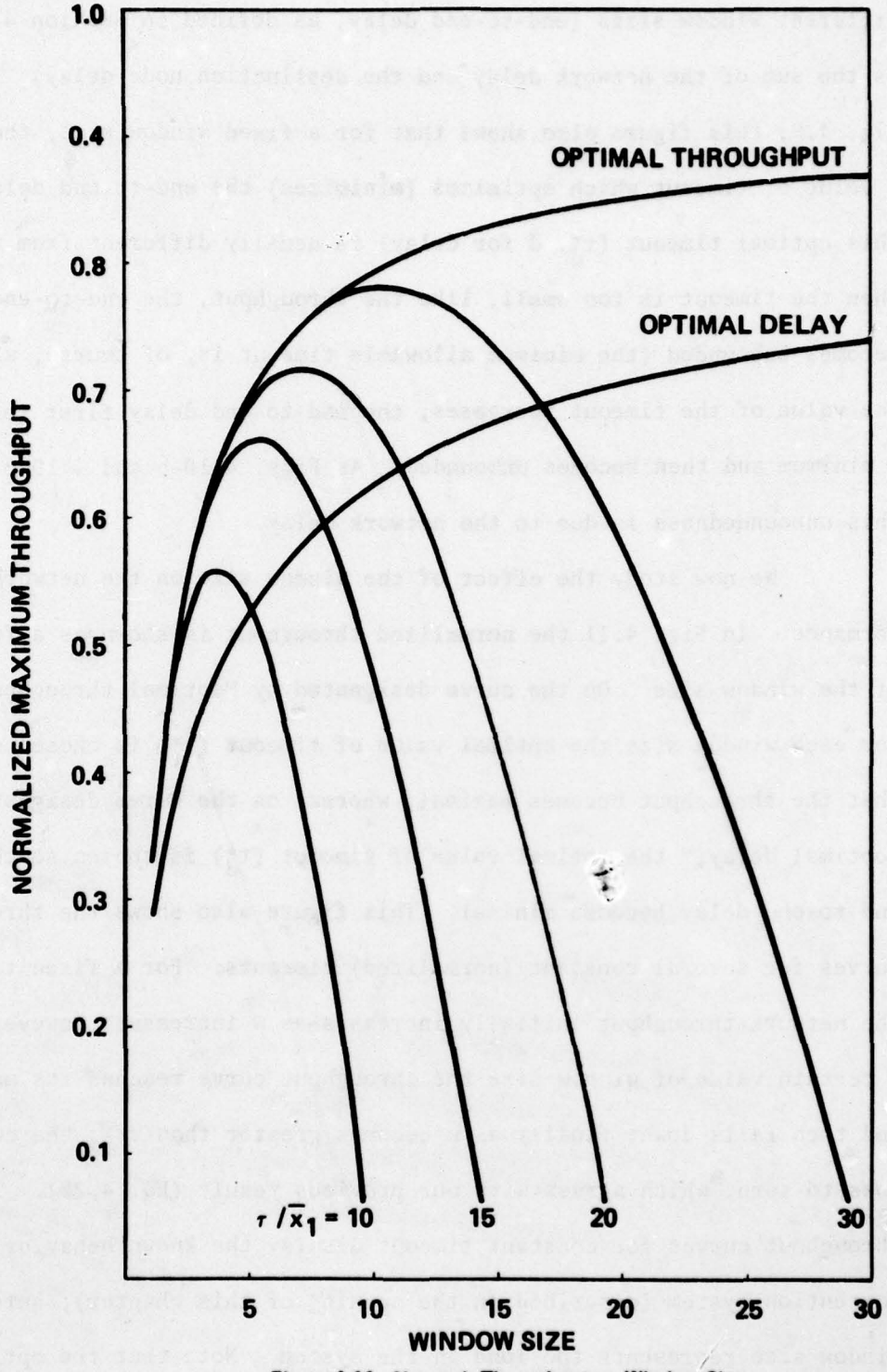


Figure 4.11 - Network Throughput vs. Window Size ($B = 10$; $C_1/C_2 = 1.0$).

throughput curve is the maximum envelope of these contours, and defines the tight upper bound on the throughput-window size performance.

The end-to-end delay as a function of window size is shown in Fig. 4.12, which is the counterpart of Fig. 4.11. In this figure, the contours of constant (normalized) timeout peel off from the delay curve of the optimal throughput, become tangent to the optimal delay curve and finally become unbounded at a value of the window size equal to $\tau/\overline{X}_1$. In analogy to Fig. 4.11, the optimal delay curve in Fig. 4.12 is the minimal envelope of the contours of constant τ's, hence it is the tight lower bound on the delay-window size performance.

To show the effect of the contention more explicitly, we have plotted the network throughput versus the network traffic for (constant) normalized timeouts in Fig. 4.13. For a fixed τ, an increase in the window size results in an increase in the network traffic. Initially the throughput increases as the network traffic grows; however, after a certain value of λ_e, the throughput starts declining, and when the network saturates ($\lambda_e/\mu C_1 = 1$) the throughput decreases to zero. In the same figure we have also shown the contours of constant window size. Note that for a fixed window size, when the timeout increases, throughput first grows and then levels off; however the network traffic decreases continuously. Note also that all w-contours peel off from a common upper bound curve, which is the maximal throughput curve (Fig. 4.11). Note how the window size and timeout vary along this curve; in fact, if any one of these two parameters (w and/or τ) is held fixed and the other is increased, the throughput decreases to zero; only when both w and τ are increased simultaneously (such that for each w the optimal τ_t^* is chosen) will the

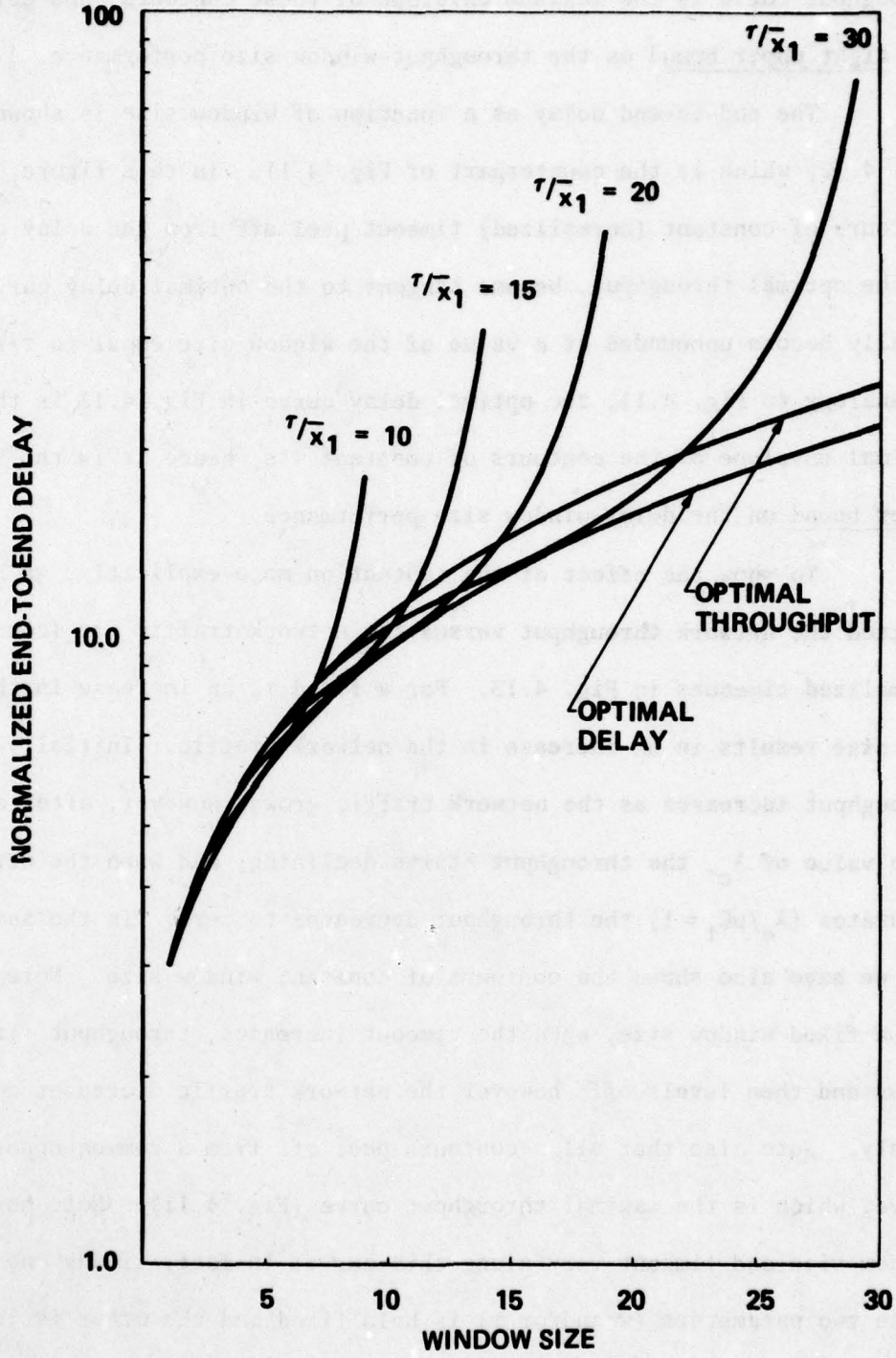


Figure 4.12 - End-To-End Delay v.s. Window Size ($B = 10$; $C_1/C_2 = 1.0$).

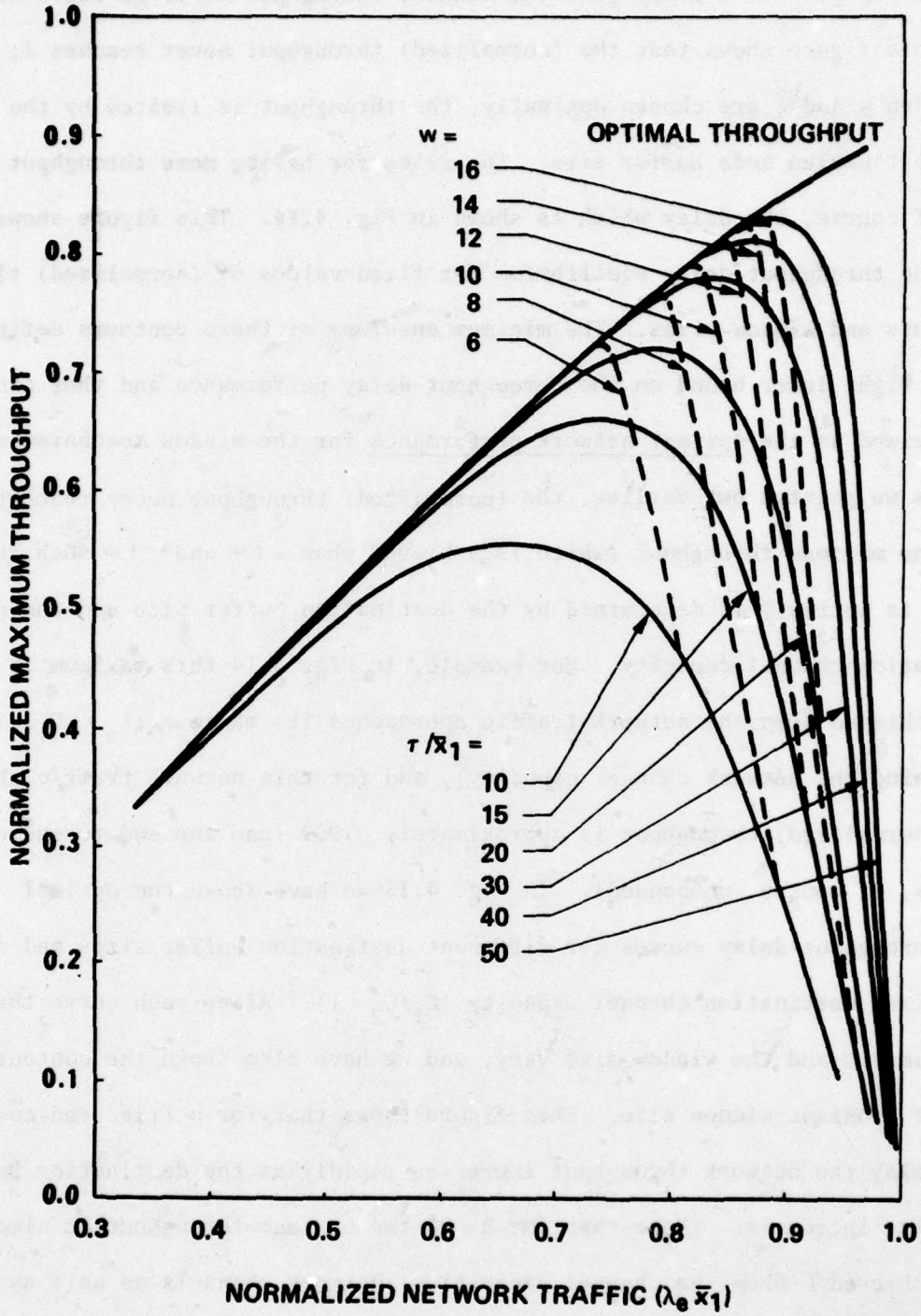


Figure 4.13 - Throughput v.s. Network Traffic
(B = 10; C_1/C_2 = 1.0).

throughput not degrade (see the maximal throughput curve in Fig. 4.13). This figure shows that the (normalized) throughput never reaches 1; even when w and τ are chosen optimally, the throughput is limited by the destination node buffer size. The price for having more throughput is, of course, the delay which is shown in Fig. 4.14. This figure shows the throughput-delay equilibrium for fixed values of (normalized) time-outs and window sizes. The minimum envelope of these contours defines a tight lower bound on the throughput-delay performance and thus can be veiwed as the optimal network performance for the window mechanism model. As we pointed out earlier, the (normalized) throughput never reaches 1; the maximum throughput (which is achieved when $w \uparrow \infty$ and $\tau \uparrow \infty$ such that τ is optimal) is determined by the destination buffer size and the destination channel capacity. For example, in Fig. 4.14 this maximum is achieved when the network traffic approaches its maximum ($\lambda_e = \mu C_1$, C_1 being the network channel capacity), and for this network traffic the (normalized) throughput is approximately 0.909 (and the end-to-end delay is, of course, unbounded). In Fig. 4.15 we have shown the optimal throughput-delay curves for different destination buffer sizes and a fixed destination channel capacity ($C_1/C_2 = 1$). Along each curve the timeout and the window size vary, and we have also shown the contours of constant window size. This figure shows that for a fixed end-to-end delay the network throughput increases rapidly as the destination buffer size increases. (Note that for B = 20 the maximum throughput is almost achieved.) When the channel capacities (network channels as well as the destination node channel) are known, this figure can be used as a design tool to determine a proper destination buffer size. For example, if we

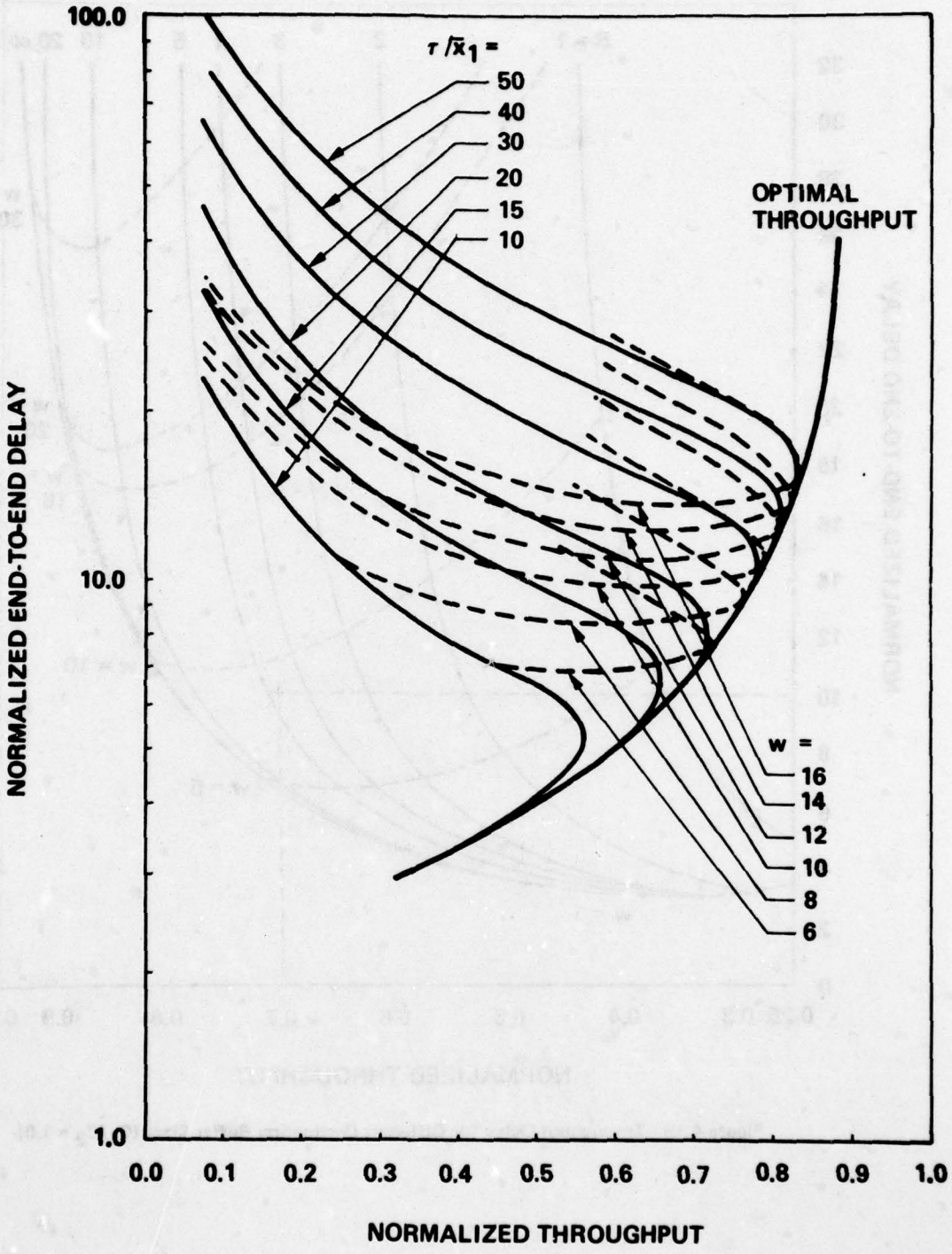


Figure 4.14 - Throughput-Delay Tradeoff
($B = 10$; $C_1/C_2 = 1.0$).

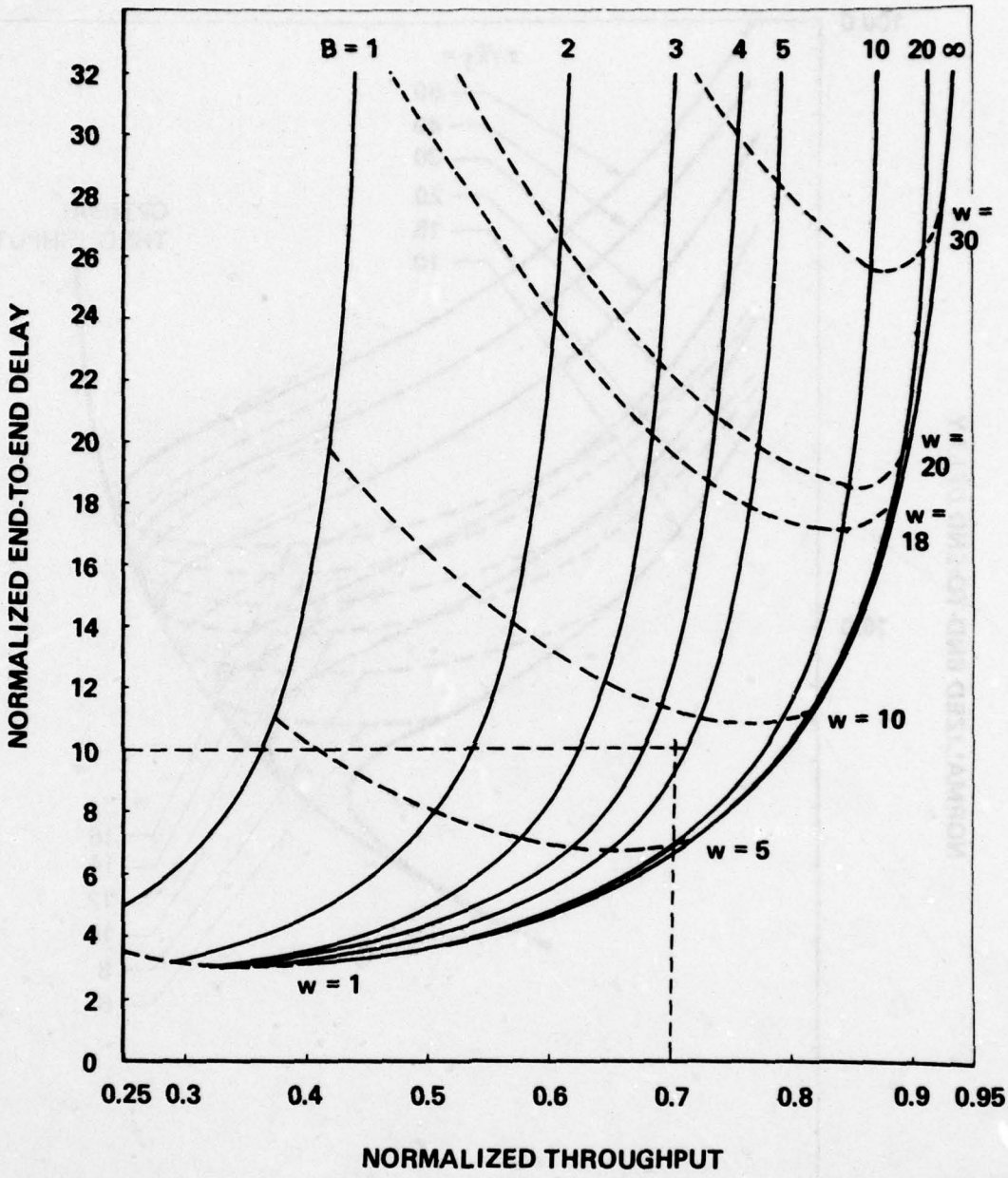


Figure 4.15 - Throughput-Delay for Different Destination Buffer Sizes ($C_1/C_2 = 1.0$).

set a minimum level of throughput, say $\lambda/\mu C_1 > 0.7$, and a maximum end-to-end delay, say $T_{ee}/\bar{X}_1 < 10$, then this figure shows that the destination buffer size should be at least 5 (the point specified by the intersection of the light dashed lines in Fig. 4.15). The operating rule is to use a window size of 5, and the optimal timeout should be determined using a curve like the one in Fig. 4.9. Note that a destination buffer size of $B > 5$ should also satisfy our requirements; however, we have chosen the minimum buffer size to minimize the overall network cost.

When the destination buffer size and the capacities of channels are known, then it is important to know what (w,τ) combination to use in order to provide a certain grade of service. In Fig. 4.16 we have plotted design contours of constant (normalized) throughput (the heavy solid curves), and (normalized) end-to-end delay (the heavy dashed curves); we will refer to these curves as λ^*-contours and Γ^*-contours respectively. This figure displays most of the information contained in the previous figures; in particular we observe the following characteristics:

1. The loci of minimum (normalized) timeouts is the lowest λ^*-contour (which coincides with the highest T^*-contour) designated by "$w = \tau/\bar{X}_1$" in Fig. 4.16. Along this curve the throughput is zero and the delay is unbounded (infinite).

2. A line of constant window size (a line perpendicular to the window size axis) may intersect a specified λ^*-contour (say $\lambda^* = \lambda_1^*$) at (a) two points; (b) one point (tangent to the $\lambda^* (= \lambda_1^*)$-contour); and (c) no point. In case (a) the two values of timeout give the same throughput (but the delays at these two points are different from each

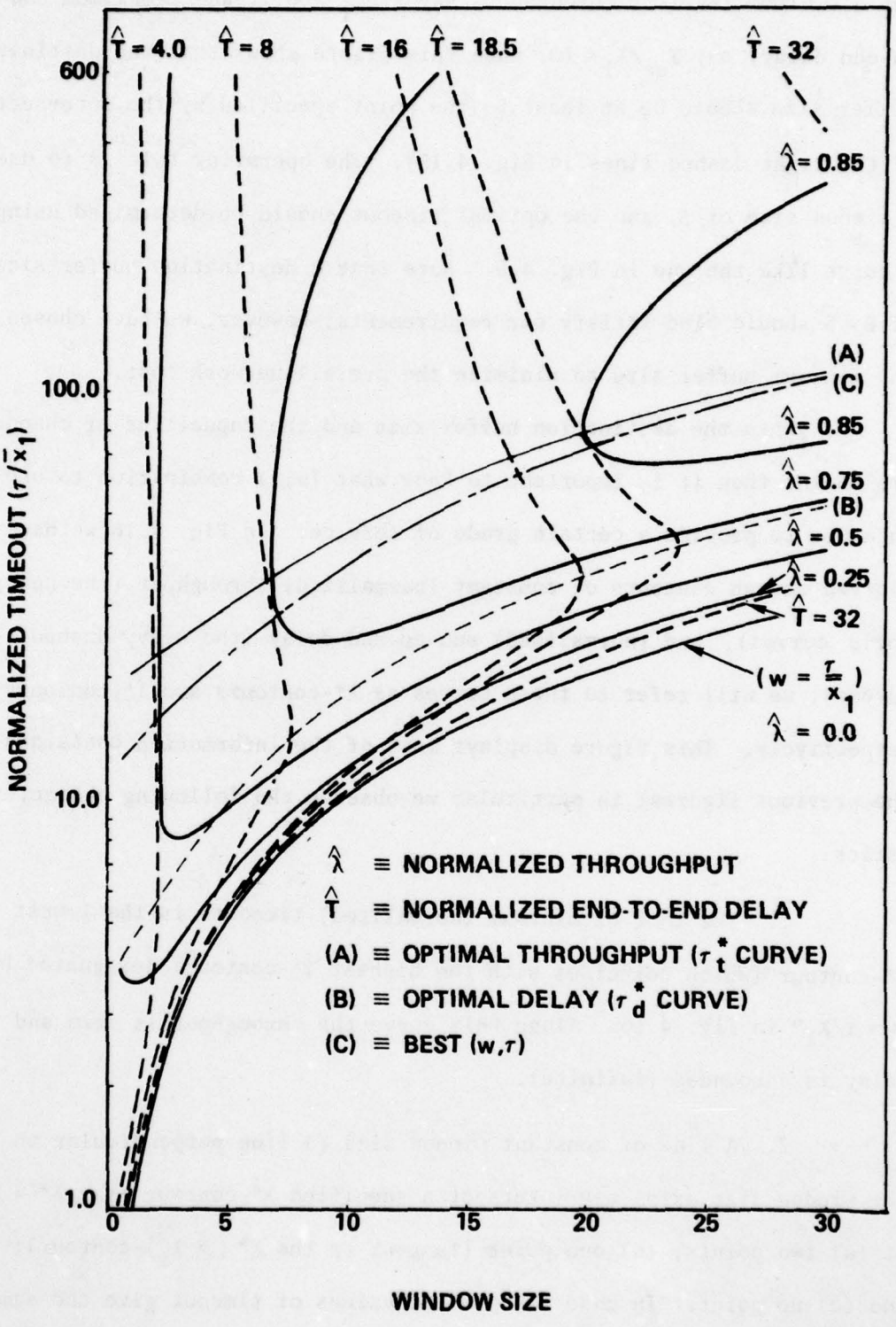


Figure 4.16 - Contours of Constant Throughputs and End-To-End Delays (B = 10; C_1/C_2 = 1.0).

other). In case (b) the value of timeout at the tangent point is τ_t^*, the optimal timeout for maximal throughput (compare with Fig. 4.9). We have indicated the loci of these points by curve A in this figure (which we will refer to as the τ_t^* - curve). If the line of constant window size does not intersect the $\lambda^*(=\lambda_1^*)$-contour (case (c)), that indicates that a throughput of λ_1^* cannot be realized by this window size.

3. The above discussion can be carried out for the T*-contours; the loci of points where the constant w lines are tangent to T*-contours (the τ_d^*-curve) is designated by curve B.

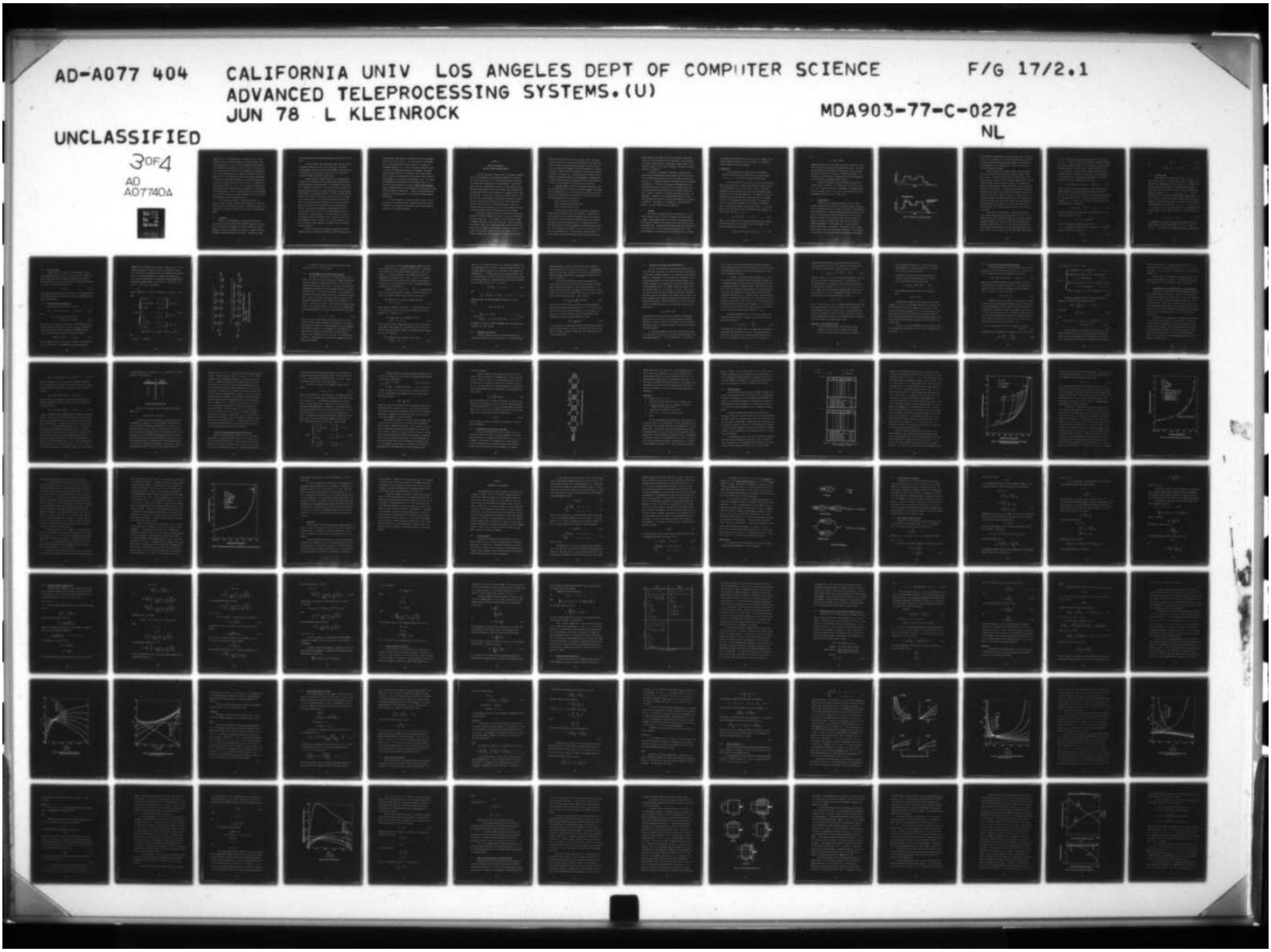
4. We can use Fig. 4.16 to find the maximum achievable throughput (and the corresponding window size) for a fixed timeout. The λ^* - contour which is tangent to the constant τ line (a line perpendicular to the τ axis with an intercept equal to the specified timeout) determines the maximum throughput, and the value of the window size at the tangent point is the w which provides this throughput. It is interesting to note that this value of the throughput is the point where the τ-constant line intersects the envelope of the throughput curves in Fig. 4.9 (we have not drawn this curve).

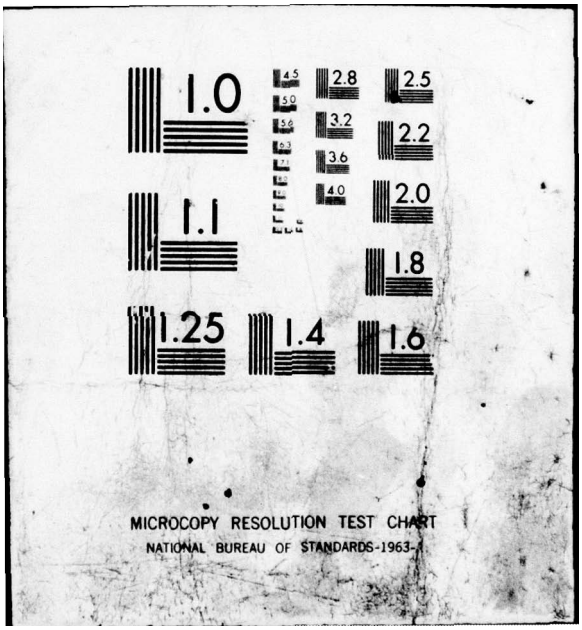
5. Fig. 4.16 can be used to find the best (w,τ) combination which guarantees a certain grade of service. Assume that for a given network (for which the parameters μ, C_1, C_2 and B are known) we are required to provide at least a minimum throughput (say $\lambda^*/\mu C_1 \geq 0.85$). In order to find the (w,τ) which results in the minimum end-to-end delay, we trace the $\lambda^*(=0.85)$-countour and find the T*-contour that is tangent to it. (In Fig. 4.16 the T-contour of 18.5 is such a contour, which is

AD-A077 404 CALIFORNIA UNIV LOS ANGELES DEPT OF COMPUTER SCIENCE F/G 17/2.1
ADVANCED TELEPROCESSING SYSTEMS.(U)
JUN 78 L KLEINROCK MDA903-77-C-0272
NL

UNCLASSIFIED

3 OF 4
AD
A077404




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
7.1
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

tangent to the $\lambda^*(=0.85)$-contour at $w=21$ and $\tau/\overline{X}_1 = 73.$) Any (w,τ) other than (21,73) results in a higher delay, hence this pair is the best operating point. The loci of these tangent points (or the best (w,τ) combinations) is shown in Fig. 4.16 as curve C. Note that we can use a smaller w and still have the same throughput (e.g., intersection of $\lambda^*(=0.85)$-contour and curve A); however, the delay at this point is larger than 18.5. Considering the fact that the window size represents the buffer size of the traffic controller, this latter operating point (with a smaller window size) may be a better choice, as it results in a lower network (buffer) cost. Fig. 4.16 can be used in an analogous way in situations where the end-to-end delay is required to be no more than a given maximum value. Similar to the previous case, the best operating (w,τ) value (which results in a maximum throughput) is found by locating the intersection of the T*-contour and curve C.

This concludes our analysis of static flow control. In Chapter 5 we will introduce a dynamic control which uses the properties of this static control and which is capable of adjusting the window size according to the stochastic fluctuations of the load in the destination buffer.

4.5 Conclusion

Based on a window mechanism, we developed three models for static flow control in store-and-forward computer communication networks. In all of the models the assumption was that the sending user is fast compared to the network and always has a message to transmit; therefore, throughput is mainly determined by the network. Throughout our study

we were concerned with the "maximum" throughput-delay behavior of a network.

We first studied a fluid approximation model in which there is no stochastic fluctuation in the load on the system. This model, although simple, shows that the traffic and throughput of a network can be controlled by a window mechanism and that it is not beneficial to increase the window size beyond a certain value.

We then developed a simple stochastic model by improving the accuracy of the deterministic model by permitting some fluctuations in the arrival process and message lengths. By assuming the arrival process of messages to the network to be Poisson, and message lengths to be exponentially distributed, closed form expressions for maximum throughput and delay of a network were derived. In order to determine the best window size to be used, we used power (defined as the ratio of throughput and delay [GIES 76]) as the measure of performance; for this measure, it was found that the best window size for this simple stochastic model is the same as the one determined by the deterministic model. Because of some simplifying assumptions regarding network reliability, this model does not reveal the contention phenomenon (where the throughput of a network degrades as its traffic increases); however, it is very useful in that it provides us with a closed form solution for the throughput and delay. In Chapter 6 we use these closed form solutions to develop a network algebra for systematic studies of interconnections of networks.

Finally, by making further assumptions regarding the acknowledgement and timeout strategy, the (sub)network delay distribution and

the destination node structure, we developed a generalized stochastic model for static flow control. It was shown that for a fixed window size, if the timeout period is either too large or too small, the throughput degrades to zero. In fact, for a given window size, there is a value for timeout which optimizes the throughput. This model explicitly shows the effect of contention in that, when the timeout period is fixed, as the window size (or the network traffic) increases, the throughput first increases and then degrades to zero. By studying the throughput-delay performance of networks (when traffic is controlled by window mechanism), we found the optimal network performance curve; the minimum envelope of the throughput-delay equilibrium for fixed values of timeouts and window sizes. A number of design problems were proposed and the optimal solution to each was given through graphical presentation.

In the next chapter, by using the results of static flow control analysis, we develop and study a dynamic flow mechanism in which the window size and timeout are dynamically adjusted to the stochastic fluctuations in the load on the system.

CHAPTER 5

DYNAMIC FLOW CONTROL IN STORE-AND-FORWARD COMPUTER NETWORKS

In static flow control (studied in the last chapter), parameters of the control mechanism are not dynamically adjusted to the network load. Due to finite buffer size at the destination node, messages may be rejected at this node and have to be retransmitted. The extra network traffic due to retransmissions results in a high round-trip delay, therefore causing frequent occurrences of timeout, and consequently more retransmissions; thus we see the potential for a dangerous positive feedback situation. We notice that if there were a further control to "slow down" the input of new messages to the network by shrinking the window size when the destination buffer begins to fill, then fewer messages would be rejected (or lost) at the destination node, and thus fewer retransmissions would be necessary. This is the idea behind the dynamic flow control mechanism which we develop and study in this chapter.

We consider a mechanism located at the destination node, which supervises the buffer occupancy at this node; let us call this mechanism a traffic director (TD). At the disposal of the TD there are a number of different window sizes from which it can choose. When the occupancy of the buffer is high, the TD signals the traffic controller (TC) at the source node (via a special control packet) to change its window size to a smaller one; when the occupancy is low, the TD notifies the TC to enlarge the window size. This notification is done by control packets

which are sent from the TD to the TC of the source node. The window size used by the TC at the source node determines the input rate to the network; therefore, the above mechanism dynamically controls the input flow to the network (the assumption is that the sending user always has a message to transmit, hence the throughput is determined by the network constraints, see Section 4.1). To prevent excessive overhead, control packets are piggybacked (similar to acknowledgements) on messages which are being transmitted from the destination node to the source node; only if there is no such message is a control packet sent by itself.

A decision table is stored at the destination node which specifies the best window sizes to be used for different states of the network. The TD, by knowing the state of the network, can therefore use the decision table to decide upon the best window size it should send to the TC. The questions which arise at this point are:

How is a network state specified?

What is meant by the best window size?

How is the decision table set up?

In order to answer these questions we develop a mathematical model of the system based on Markov decision theory. We model the destination node as a finite waiting room queueing system with K possible input rates (K is also the number of different window sizes; see below), for which we want to find an optimal policy for choosing input rates such that the long run average throughput of the buffered system is maximal, and the average delay is minimal. (As we will see, delay is not due only to the destination node.) After defining a reward criteria, we derive a maximal reward rate stationary operating policy for the finite

queue; therefore, the problem we pose will be reduced to the optimal control of a finite queue with variable input rates. As we see below, due to special characteristics of the system, an exact solution to the mathematical model is very laborious; therefore, we develop a heuristic solution to the problem.

In Section 5.1 the model is formulated as a discrete-time, multiple-state variable Markov decision process, which we will refer to as a $\hat{T}_r$-cycle-delay decision process ($\hat{T}_r$ is the round-trip delay, see Section 5.1.1). A heuristic solution to this Markov process is presented in Section 5.2. In Section 5.3 we discuss the implementation of the heuristic optimal policy, and in Section 5.4 we further elaborate on a 1-cycle-delay decision process which we use to generate the decision table. Finally, in Section 5.5 we present some numerical results. For completeness, a summary of results from Markov decision theory and a brief description of the policy-iteration method [HOWA 60] are presented in Appendix C. The material in this appendix is taken from Howard [HOWA 60, 71], Jewell [JEWE 63] and Ross [ROSS 70].

5.1 The Model

We consider the destination node of a network (Fig. 4.1) as a buffer storage of size B in front of a transmission line with capacity C_2 (bps). Messages which arrive at the storage (after passing through the network) are buffered (if necessary) and are then transmitted out to the receiving user. We assume that the message lengths are exponentially distributed and let $1/\mu$ be the average message length (bits). With these assumptions, the transmission times of messages out of the buffer

are exponentially distributed with a rate of $\gamma_2 = \mu C_2$. Messages, after passing through the network, arrive at the buffer. For the arrival process we make the following assumption:

Assumption 5.1

The counting process of message arrival to the buffer is assumed to be Poisson with an arrival rate $\lambda \in \Lambda$ (msg/sec), where $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_K\}$ is the set of possible arrival rates.

For reasons which will be clear shortly we study a discrete-time model with very small time increments, and observe the system at miniscule time intervals δ (e.g., $\delta = 4 \times 10^{-3}$ sec = 4 msec). The discrete approximation to the continuous case causes some inconvenience when we describe the state transition probabilities. We assume that at most one departure and one arrival may occur at each time slice (of δ seconds). This means, at the beginning of each time interval a new message may arrive in the buffer with probability $\lambda\delta$ (the result of a Bernoulli trial); thus the average number of arrivals per second is λ. The transmission times of messages are chosen independently from a geometric distribution such that, for $\hat{\gamma}_2 = \gamma_2 \delta < 1$

$$s_n = \hat{\gamma}_2(1 - \hat{\gamma}_2)^{n-1} \qquad n = 1, 2, \ldots \tag{5.1}$$

where s_n is the probability that a message transmission time is exactly n time units long (i.e., that its service time is $n\delta$ seconds). The average transmission time is therefore

$$\hat{X}_2 = \frac{1}{\hat{\gamma}_2} \text{ time units} = \frac{\delta}{\hat{\gamma}_2} \text{ sec} = \frac{1}{\gamma_2} \text{ sec} \tag{5.2}$$

or

$$\hat{X}_2 = \frac{1}{\mu C_2} \text{ seconds}$$

Regarding the order in which the events take place at the end of a time slice, we assume that departures take place before arrivals. This means that a message which has completed its transmission leaves the system, and then a message which has arrived is allowed into the buffer. (If the buffer is full and there is no departure, then the arriving message is lost and must be retransmitted from the source.)

For notational purposes for the discrete time version we use variables under a circumflex (^); in particular, we let $\hat{\lambda}_k = \lambda_k \delta$, $\hat{\lambda} = \lambda\delta$, and $\gamma_2 = \gamma_2\delta$. The round-trip delay is designated by the random variable $\tilde{t}_r$ with an average T_r and we let $\hat{T}_r = T_r/\delta$. For the one-way delay, the corresponding notation will be $\tilde{t}$, T and $\hat{T} = T/\delta$.

5.1.1 The State Space

The decision on the appropriate window size by the traffic director (TD) (hence on the input rate to the buffer storage; see below), should be based basically on the occupancy of the buffer. Having decided on the new window size, through the control signalling process described earlier, the TD notifies the traffic controller (TC) of this decision. When the TC receives this notification (after the one-way delay of $\tilde{t}$ seconds), it adjusts the window and the input rate to the network is changed corresponding to the new window size. We assume that this change of input rate is instantaneous. The diagram in Fig. 5.1 shows the evolution in time of the input rate to the network. In this figure, the solid line is the input rate we use in our analysis and the dashed

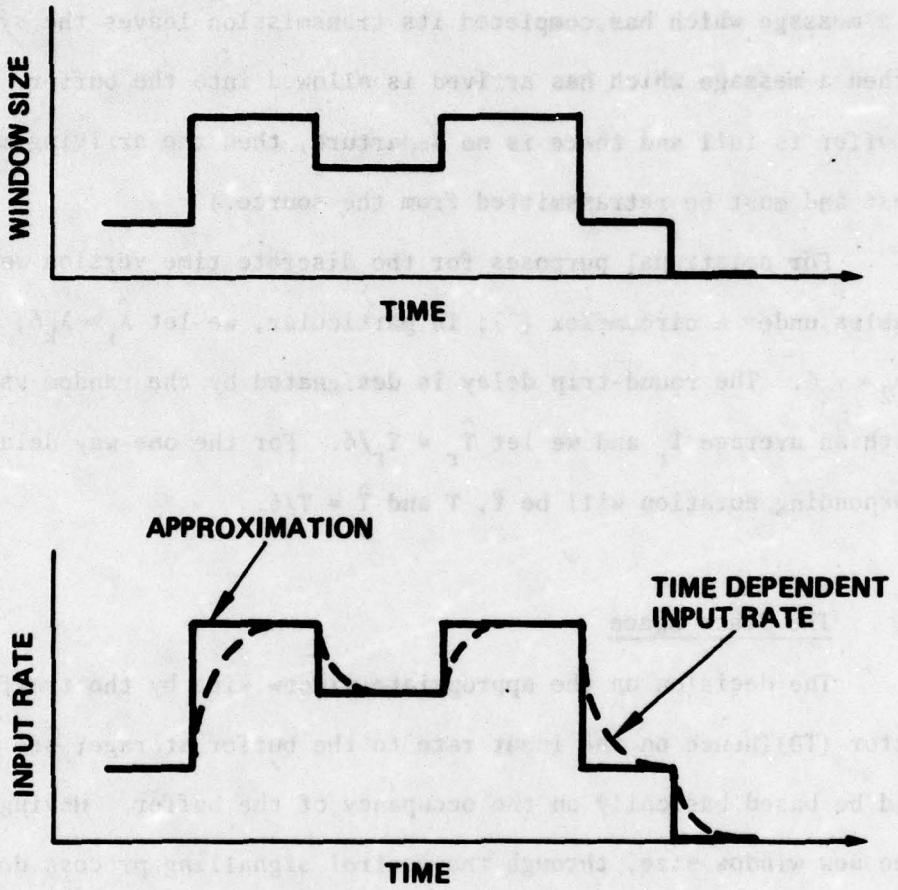


Figure 5.1 - Evolution in Time of Input Rate to Network.

curve represents an approximation to the actual input rate as a function of time. It is possible to use the dashed curve and analyze the system as a queue with time-varying arrival rates [NEWE 68, 71]; however, due to the complexity of the resulting model, we choose to use the input rate with instantaneous changes.

After the input rate is changed, messages which are admitted with the new rate reach the destination node after $\tilde{t}$ seconds. Therefore, there is a (random) gap from the time the TD decides upon a new input rate until the result of this input rate becomes effective at the destination node. When this time gap is random, the resulting model becomes extremely difficult to analyze and in fact, no such analysis has been done in the literature. To render the model amenable to analysis, we further assume that the time gap between the moment that a decision is made until the time it becomes effective is a constant interval of duration T_r seconds, the average round-trip delay. Note that due to the fact that the input rate to the network varies, the average round-trip delay fluctuates as well; however, we will take T_r as the average of round-trip delays due to different possible choices of window sizes (or input rates).

With the above considerations, we note that the decision at time t becomes effective at the destination node at time $t + T_r$. Considering the fact that a decision on window size is basically based on the buffer occupancy, the decision at time t should be based on the occupancy of the buffer at time $t + T_r$. Thus the state description at time t should contain information, not only regarding the buffer occupancy at time t, but also the input rates to the buffer in the interval

$(t, t + T_r)$. (These input rates have been decided upon in the interval $(t - T_r, t)$.) In order to include this information in the state variable, we consider a discrete time model in which the system is observed every δ seconds, and when a decision is made it becomes effective after $\hat{T}_r$ $(= T_r/\delta)$ time units. In the discrete time model we designate time by $\hat{t}$; therefore time t (in seconds) is identical to $\hat{t}$ $(= t/\delta)$ measured in a time unit equal to δ. With the above considerations, the state of the system at time $\hat{t}$ is represented by $X(\hat{t})$, where

$$X(\hat{t}) = \{n(\hat{t}),\ \hat{\lambda}(\hat{t},1),\ \hat{\lambda}(\hat{t},2),\ \ldots,\ \hat{\lambda}(\hat{t},\hat{T}_r - 1)\} \qquad (5.3)$$

where $0 \leq n(\hat{t}) \leq B$ is the buffer occupancy at time $\hat{t}$ and $\hat{\lambda}(\hat{t},k)$ (measured in msg/δsec) is the input rate to the buffer at time $(\hat{t} + k)$, $1 \leq k < \hat{T}_r$. Note that $\hat{\lambda}(\hat{t},k)$ is the result of the decision which was made $\hat{T}_r - k$ time units earlier than $\hat{t}$. We may also label the states by the integers 0, 1, ..., N, where (N + 1) is the size of the state space; in equilibrium a state i is defined as

$$i = \{ n_i,\ \hat{\lambda}_i^1,\ \hat{\lambda}_i^2,\ \ldots,\ \hat{\lambda}_i^{\hat{T}_r - 1} \} \qquad 0 \leq i \leq N \qquad (5.4)$$

The set of all states in the state space will be denoted by S. The cardinality of the set S, $|S|$, is

$$|S| = N + 1 = (B+1) \times K^{\hat{T}_r - 1}$$

where K is the number of different window sizes.

As a notational convention we use the symbol "<=>", to connect a state, say $X(\hat{t})$ to its label, i.e.,

$$X(\hat{t}) \iff i \qquad (5.5)$$

if

$$n(\hat{t}) = n_i \tag{5.6-a}$$

and

$$\hat{\lambda}(\hat{t},k) = \hat{\lambda}_i^k \qquad 1 \leq k < \hat{T}_r \tag{5.6-b}$$

5.1.2 The Action Space

We consider a finite set of window sizes $W = \{w_1, w_2, \ldots, w_K\}$ at the disposal of the TD. From our earlier analysis in Chapter 4, we know that for a given network and for a window size w, a timeout τ_t^* can be chosen such that the throughput of the network becomes maximal. Hence, when the TD decides upon a window size, it also determines the appropriate timeout $\tau \in \{\tau_1, \tau_2, \ldots, \tau_K\}$ (or $\{\hat{\tau}_1, \hat{\tau}_2, \ldots, \hat{\tau}_K\}$) and notifies the TC in the source node (in the manner described above). The set W is the set of actions in our model; however, because each (w_k,τ_k) determines an input rate (see Chapter 4), we might as well assume that the corresponding set of input rates $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_K\}$ (or $\{\hat{\lambda}_1, \hat{\lambda}_2, \ldots, \hat{\lambda}_K\}$, where $\hat{\lambda} = \delta\lambda$) is the set of actions. In the discussion below we will refer to either of these two sets (W or Λ) as the action space, i.e.,

$$A = \Lambda \quad \text{or} \quad A = W \tag{5.7}$$

Corresponding to the above sets of input rates, timeouts and window sizes, there is a set of network delays $T = \{T_1, T_2, \ldots, T_K\}$ or $\{\hat{T}_1, \hat{T}_2, \ldots, \hat{T}_K\}$ where T_k is the network delay when w_k is being used.

5.1.3 The Policy Space

A policy specifies a window size (or equivalently an input rate) which should be sent to the TC. More precisely, by a policy f we mean a decision rule that says, given that the system is in state i, then we should use $\hat{\lambda}_k$ (or w_k) as the input rate to the buffer which will be effective $\hat{T}_r$ time units later, that is,

$$f(i) = k \tag{5.8}$$

We are implicitly using policies that are functions of the present state of the system; policies in this class are known as "stationary policies" and will be denoted by F.

5.1.4 The State Transition Probabilites

Let the system states at time $\hat{t}$ and $\hat{t}+1$ be

$$X(\hat{t}) = \{n(\hat{t}),\ \hat{\lambda}(\hat{t},1),\ \hat{\lambda}(\hat{t},2),\ \ldots,\ \hat{\lambda}(\hat{t},\hat{T}_r-1)\} \tag{5.9}$$

and

$$X(\hat{t}+1) = \{n(\hat{t}+1),\ \hat{\lambda}(\hat{t}+1,1),\ \hat{\lambda}(\hat{t}+1,2),\ \ldots,\ \hat{\lambda}(\hat{t}+1,\hat{T}_r-2),\ \hat{\lambda}(\hat{t}+1,\hat{T}_r-1)\} \tag{5.10}$$

where $\hat{\lambda}(\hat{t},k)$ is the input rate at time $\hat{t}+k$. The input rate that the TD decides at time $\hat{t}$ to be effective at time $\hat{t}+\hat{T}_r$, $\lambda_{f(X(\hat{t}))}$, becomes part of the state vector at time $\hat{t}+1$; therefore the components of state vectors $X(\hat{t})$ and $X(\hat{t}+1)$ are related to each other as follows:

$$\hat{\lambda}(\hat{t}+1,k) = \hat{\lambda}(\hat{t},k+1) \qquad 1 \leq k \leq \hat{T}_r-2 \tag{5.11}$$

The last component of state vector $X(\hat{t}+1)$, $\lambda(\hat{t}+1,\hat{T}_r-1)$, is the decision which is made at time $\hat{t}$. The relationship between the input rate

components of two consecutive state vectors is shown in Fig. 5.2. The number of messages in the buffer at time $\hat{t}+1$ is different from $n(\hat{t})$; this is the result of the possible arrival of a message, with probability $\hat{\lambda}(\hat{t},1)$, or the possible departure of a message, with probability $\hat{\gamma}_2$. With the above considerations, the equilibrium transition probabilities under policy f between states $i = \{n_i, \lambda_i^1, \lambda_i^2, \ldots, \lambda_i^{\hat{T}_r-1}\}$ and $j = \{n_j, \lambda_j^1, \lambda_j^2, \ldots, \lambda_j^{\hat{T}_r-1}\}$ will be as follows:

Let

$$p_{ij}(f) = \Pr[X(\hat{t}+1) \Longleftrightarrow j \mid X(\hat{t}) \Longleftrightarrow i, \text{ and policy } f \in F \text{ is used}] \qquad 0 \leq \hat{t}$$

then

$$p_{ij}(f) = \begin{cases} \hat{\lambda}_i^1(1-\hat{\gamma}_2) & \text{if } n_j = n_i + 1 \\ 1 - \hat{\lambda}_i^1 - \hat{\gamma}_2 + 2\hat{\lambda}_i^1\hat{\gamma}_2 & \text{if } n_j = n_i \\ (1-\hat{\lambda}_i^1)\hat{\gamma}_2 & \text{if } n_j = n_i - 1 \end{cases} \text{ if } 0 < n_i < B$$

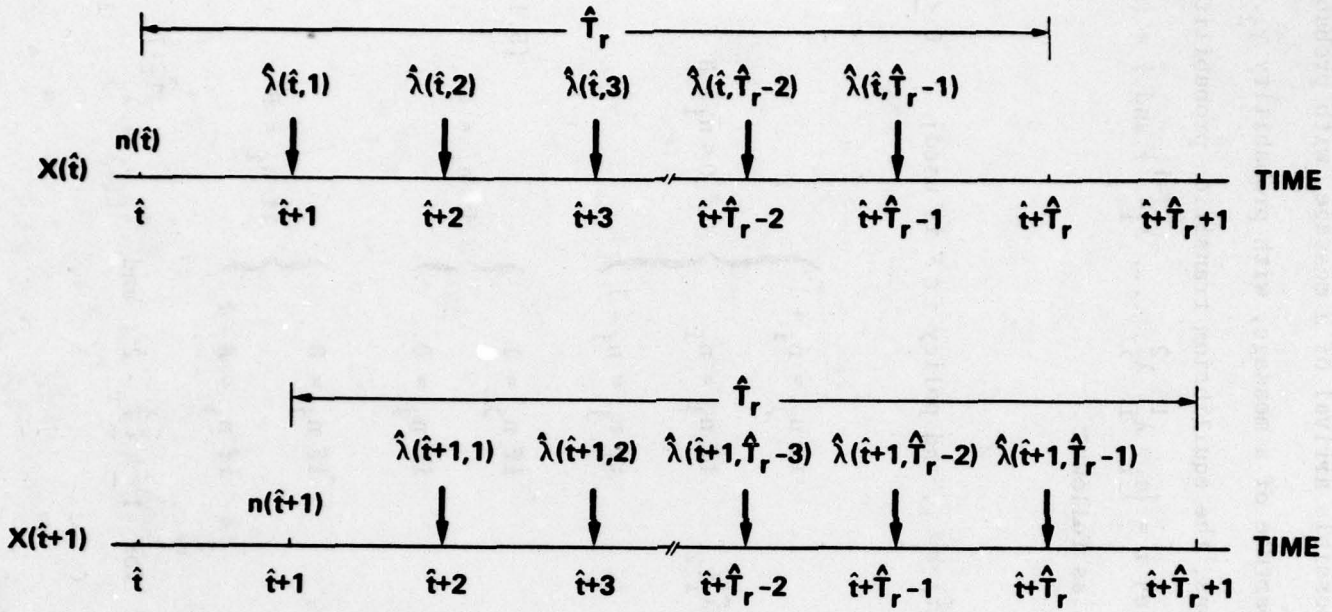
$$p_{ij}(f) = \begin{cases} \hat{\lambda}_i^1 & \text{if } n_j = 1 \\ 1 - \hat{\lambda}_i^1 & \text{if } n_j = 0 \end{cases} \text{ if } n_i = 0 \qquad (5.12\text{-a})$$

$$p_{ij}(f) = \begin{cases} 1 - \hat{\gamma}_2 + \hat{\lambda}_i^1\hat{\gamma}_2 & \text{if } n_j = B \\ (1-\hat{\lambda}_i^1)\hat{\gamma}_2 & \text{if } n_j = B-1 \end{cases} \text{ if } n_i = B$$

$$\text{if } \hat{\lambda}_j^k = \hat{\lambda}_i^{k+1} \text{ for } 1 \leq k \leq \hat{T}_r - 2 \text{ and } \hat{\lambda}_{f(i)} = \hat{\lambda}_j^{\hat{T}_r-1}$$

and

$$p_{ij}(f) = 0 \qquad \text{otherwise} \qquad (5.12\text{-b})$$



$\hat{\lambda}(\hat{t},k) = \hat{\lambda}(\hat{t}+k)$ INPUT RATE AT TIME $\hat{t}+k$

$f(X(\hat{t})) = \hat{\lambda}(\hat{t}+\hat{T}_r)$

Figure 5.2 - State Vectors at Two Consecutive Time Cycles.

We refer to the matrix of transition probabilities under policy f by $P(f)$, i.e., $P(f) = \{p_{ij}(f)\}$.

5.1.5 The Performance Criteria and the Reward Function

The objective of our analysis is to find an optimal policy which, in the long run, maximizes the average throughput and minimizes the average delay of the system; this objective is reflected in our formulation through the reward criteria. Here again we are faced with incorporating the opposing effects of throughput and delay in a single objective function. In Section 4.3, faced with a similar problem, we used power (the ratio of throughput and delay) as the measure of performance. Power is a good (but not the only) measure in the sense that both high throughput and low delay characterize the efficiency of a system. In this section, however, we choose another performance measure and use it in our decision problem. Let λ and T be the throughput and the corresponding delay of the system; then the reward for this system is

$$R = \alpha\lambda - T \qquad \alpha \geq 0 \tag{5.13}$$

Similar to power, a high throughput and low delay result in a large value for R, which reflects a measure of the efficiency of the network. The coefficient α reflects our relative preference for large throughput with respect to low delay. $\alpha = 0$ corresponds to a situation where we cannot accept any delay (we will see shortly that this value of α results in zero throughput and hence, zero delay). The other extreme is when $\alpha \uparrow \infty$, which reflects the case in which throughput is the only measure of performance.

We now need to find the expected immediate reward (the reward to be expected in the next transmission) for each state. In Appendix C we have used $R_i(f)$ for the expected immediate reward for state i under policy f. As we will see shortly, in the problem at hand $R_i(f)$ is independent of the decision made in state i (in fact, it is dependent on the decision made $\hat{T}_r$ time units earlier); however, for notational consistency, we use the symbol $R_i(f)$ to refer to the expected immediate reward for state i under policy f.

Consider state $i = \{n_i, \hat{\lambda}_i^1, \hat{\lambda}_i^2, \ldots, \hat{\lambda}_i^{\hat{T}_r - 1}\}$. A transition from state i to any succeeding state (including i) causes each of the n_i messages in the buffer to suffer one unit of delay, and so we have

$$R_i^o = \text{Reward rate due to buffer occupancy for state i}$$
$$= -n_i \qquad 0 \leq n_i \leq B \tag{5.14}$$

Once a message is admitted to the buffer, it is guaranteed delivery to the receiving user; in the next transition a message may arrive with probability $\hat{\lambda}_i^1$, therefore if $n_i < B$

$$R_i^t = \text{Reward rate due to throughput for state i}$$
$$= \alpha\hat{\lambda}_i^1 \qquad \text{if} \quad n_i < B \tag{5.15}$$

On the other hand, when a message arrives with rate $\hat{\lambda}_i^1$, it also brings in an amount of delay equal to $\hat{T}_i^1$ (corresponding to $\hat{\lambda}_i^1$) which it has suffered in passing through the network. The probability that a message arrives is $\hat{\lambda}_i^1$; therefore

$$R_i^d = \text{Reward rate due to network delay for state i}$$
$$= -\hat{\lambda}_i^1 \hat{T}_i^1 \qquad \text{if } n_i < B \tag{5.16}$$

When $n_i = B$, i.e., when the buffer is full, a message is accepted in the next transition only if a position in the buffer becomes available, i.e., a message completes transmission out of the buffer (this happens with probability $\hat{\gamma}_2$). If a buffer position does not become available, then an arriving message is rejected. Because the rejected message must wait $\hat{\tau}_i^1$ time units before it is retransmitted ($\hat{\tau}_i^1$ is the timeout corresponding to the input rate $\hat{\lambda}_i^1$), when $n_i = B$ we have

$$R_i^t = \alpha\hat{\lambda}_i^1\hat{\gamma}_2 \qquad \text{if} \quad n_i = B \tag{5.17}$$

and

$$R_i^d = -\hat{\lambda}_i^1[\hat{\gamma}_2\hat{T}_i^1 + (1-\hat{\gamma}_2)\hat{\tau}_i^1] \qquad \text{if} \quad n_i = B \tag{5.18}$$

Summing the above, the expected immediate reward rate for state i becomes

$$R_i(f) = R_i^o + R_i^d + R_i^t$$

or

$$R_i(f) = \begin{cases} \alpha\hat{\lambda}_i^1 - (n_i + \hat{\lambda}_i^1\hat{T}_i^1) & n_i < B \\ \alpha\hat{\lambda}_i^1\hat{\gamma}_2 - [n_i + \hat{\lambda}_i^1\hat{\gamma}_2\hat{T}_i^1 + \hat{\lambda}_i^1(1-\hat{\gamma}_2)\hat{\tau}_i^1] & \text{if} \quad n_i = B \end{cases} \tag{5.19}$$

We designate the vector of expected immediate reward rates under policy f by $R(f)$, i.e., $R(f) = \{R_i(f)\}$.

5.1.6 Statement of the Problem

The above formulation specifies a Markov decision process (chain) that represents a system in which there is a constant time gap

between the instant a decision is made until the time the decision becomes effective; we will refer to this process as a $\hat{T}_r$-cycle-delay Markov decision process. Processes similar to this are encountered in inventory problems, where there is a lead-time for an order to arrive [MILL 74], [VEIN 66].

The set of stationary policies in our decision process is actually a subset of the set of all policies (see Appendix C for examples of other possible policies). Under a stationary policy f, the average reward per unit time is designated by the vector $g(f) = \{g_i(f)\}$, where $g_i(f)$ is the long run expected reward per unit time given that the process initially starts in state i. $g(f)$ is given by

$$g(f) = \lim_{n\uparrow\infty} \frac{1}{n+1} \sum_{t=0}^{n} [P(f)]^t R(f) \qquad (5.20)$$

where $[P(f)]^t$ is the t^{th} power of $P(f)$ and is the t-step probability transition matrix under the stationary policy f. If f is a policy such that $P(f)$ has one chain, then all components of $g(f)$ are equal. Our objective is to find a (stationary) policy f* which maximizes each component of the expected average reward per unit time,

$$g(f^*) = \max_{f\in F} \{g(f)\} \qquad (5.21)$$

It can be shown for a bounded reward function, when the action space and the state space are finite, that such a stationary policy always exists [BLAC 62], [JEWE 63], [ROSS 70], [SOBE 74]. These conditions are met in the problem at hand.

5.2 Solution to the Problem: The Look-Ahead Policy

The $\hat{T}_r$-cycle-delay Markov decision process formulated in the last section can, in principle, be solved for an optimal stationary policy by using any numerical technique, e.g., the policy-iteration method [HOWA 60]. Unfortunately, for any realistic set of parameters, the size of the problem (especially the state space) becomes too large; it is extremely laborious, if not impossible, to solve such a large problem by any conventional means. As an example, consider a system in which the round-trip delay, $\hat{T}_r$, is 0.2 seconds. If we use $\delta = 4$ msec as our time unit, the round-trip delay $\hat{T}_r$ becomes 50 time units. Therefore, each state vector has 50 components, one of which (the first one) is the buffer occupancy, and the remaining 49 elements are the input rates for the next 49 time cycles. Assume the buffer size is 10 ($B = 10$) and that there are 5 different window sizes to use; therefore, the size of the state space is

$$|S| = 11 \times 5^{49} \simeq 10^{35}$$

Considering the fact that each cycle of the policy-iteration method requires the solution of a set of $|S|$ linear simultaneous equations (Appendix C), a direct solution to this problem is out of the question, hence we must resort to a heuristic solution. To this end we consider the following points:

1. The major difficulty arises from a long round-trip delay. For a round trip of 2 units ($\hat{T}_r = 2$), the size of the state space for the above example becomes $|S| = 11 \times 5 = 55$ and when $\hat{T}_r = 1$, it is only 11.

($\hat{T}_r = 1$ results in a 1-cycle-delay Markov decision process in which a decision becomes effective in the next transition; this is the case with an ordinary Markov decision process and we make extensive use of this case in our heuristic solution.)

2. Consider a 1-cycle-delay decision process ($\hat{T}_r = 1$). For this system the only information necessary to determine a window size is the buffer occupancy; in fact, in this case the state of the system is represented simply by the buffer occupancy. When there is a lead-time of $\hat{T}_r > 1$ for a decision to become effective, the process makes use of the input rate components of a state vector to predict the future occupancy of the buffer at $(\hat{T}_r - 1)$ time units later. Based on this prediction, it then decides on a window size (or an input rate) to become effective at $\hat{T}_r$ time units later. In fact, if at time $\hat{t}$ we provide the decision process with the future buffer occupancy at, say k, time units later ($k < \hat{T}_r$), then from the decision process point of view, the lead-time for a decision to become effective reduces to $\hat{T}_r - k$.

The above considerations lead us to develop a heuristic solution for the decision process. Consider state i of a $\hat{T}_r$-cycle-delay Markov decision process,

$$i = \{n_i, \hat{\lambda}_i^1, \hat{\lambda}_i^2, \ldots, \hat{\lambda}_i^{\hat{T}_r - 1}\} \tag{5.22}$$

The component $\hat{\lambda}_i^k$ in the state vector is the input rate to the buffer at k time units after the system is in state i. By knowing the future input rates to the buffer, we can find the expected occupancy of the buffer at $\hat{T}_\ell$ time units later ($0 \leq \hat{T}_\ell \leq \hat{T}_r$); we refer to $\hat{T}_\ell$ as the

look-ahead time and designate the expected buffer occupancy after $\hat{T}_\ell$ transitions from state i by $\bar{n}_i\,(+\hat{T}_\ell)$. When the set of policies and state transition probabilities are properly defined, the vector

$$i' = \{\bar{n}_i(+\hat{T}_\ell),\ \hat{\lambda}^{\hat{T}_\ell+1},\ \hat{\lambda}^{\hat{T}_\ell+2},\ \ldots,\ \hat{\lambda}^{\hat{T}_r-1}\} \tag{5.23}$$

specifies a state of a $(\hat{T}_r - \hat{T}_\ell)$-cycle-delay decision process. We designate the set of states of this process by S'. For this process, a (stationary) policy f' is defined as a function from the state space S' to the action space A defined by Eq. (5.7), i.e., $f':S'\rightarrow A$. If $|S'|$ is not too large, we can find an optimal stationary policy f'^* for this process, and use f'^* to find a (sub)optimal stationary policy f_s^* for the original $\hat{T}_r$-cycle-delay decision process, so that

$$f_s^*(i) = f'^*(i') \tag{5.24}$$

where i' is given by Eq. (5.23) The expected reward rate (and the long run average throughput) for policy f_s^* is very close to the optimal expected reward rate for the original process. (The numerical results in Section 5.5 support this claim.) We refer to policy f_s^* as the optimal $\hat{T}_\ell$-look-ahead policy, where $\hat{T}_\ell$ is the look-ahead time. We summarize the above procedure in the algorithm given below.

Algorithm 5.1: The Look-Ahead Policy

1. For the $\hat{T}_r$-cycle-delay Markov decision process, find a look-ahead interval $0 \leq \hat{T}_\ell < \hat{T}_r$, such that the resulting $(\hat{T}_r - \hat{T}_\ell)$-cycle-delay Markov decision process is solvable.

2. Use any appropriate technique (e.g., the policy-iteration method, Appendix C) to find an optimal policy f'^* for the $(\hat{T}_r - \hat{T}_\ell)$-cycle-delay decision process.

3. For each state i of the $\hat{T}_r$-cycle-delay decision process find $\bar{n}_i \ (+ \hat{T}_\ell)$ and the corresponding state i' for the $(\hat{T}_r - \hat{T}_\ell)$-cycle-delay decision process, where

$$i' = \{\bar{n}_i (+ \hat{T}_\ell),\ \lambda_i^{\hat{T}_\ell+1},\ \lambda_i^{\hat{T}_\ell+2},\ \ldots,\ \lambda_i^{\hat{T}_r-1}\}$$

4. The (sub)optimal decision at state i is given by $f_s^*(i)$ where

$$f_s^*(i) = f'^*(i')$$

The problem which remains is that the expected future buffer occupancy, $\bar{n}_i \ (+ \hat{T}_\ell)$, is a real number, whereas for the solution of the decision problem we require the occupancy to be an integer. We can bypass this problem by rounding $\bar{n}_i \ (+ \hat{T}_\ell)$ to its nearest integer. The above heuristic solution plays a crucial role in making possible the use of the policy-iteration method to solve optimization problems involving long round-trip delays and a large number of possible window sizes to choose from.

In order to use the above algorithm for the (sub)optimal selection of window size, we must find $\bar{n}_i \ (+ \hat{T}_\ell)$. This is the subject of the next section.

5.2.1 Calculation of the Expected Buffer Occupancy

For notational simplicity in the derivations below, instead of referring to a state of the system by its label i (e.g., Eq. (5.22)), we choose to specify the state of the system at time $\hat{t}$ by $X(\hat{t})$,

$$X(\hat{t}) = \{n(\hat{t}),\ \hat{\lambda}(\hat{t},1),\ \hat{\lambda}(\hat{t},2),\ \ldots,\ \hat{\lambda}(\hat{t},\hat{T}_r-1)\} \tag{5.25}$$

where $n(\hat{t})$ is the buffer occupancy at time $\hat{t}$ and $\hat{\lambda}(\hat{t},k)$ $(= \hat{\lambda}(\hat{t}+k))$ is the input rate to the buffer at time $(\hat{t}+k)$. The expected buffer occupancy at time $(\hat{t}+k)$ will be designated by $\overline{n}(\hat{t}+k)$; therefore, if $X(\hat{t}) \Longleftrightarrow i$ (see Eqs. (5.5) and (5.6)), then

$$\overline{n}(\hat{t}+k) = \overline{n}_i(+k) \tag{5.26}$$

Consider the state of the system at time $\hat{t}$ given in Eq. (5.25). In order to find the expected buffer occupancy at time $\hat{t}+\hat{T}_\ell$ (where $\hat{T}_\ell$ is the look-ahead time), we need to know the equilibrium probability distribution of the number of messages in the buffer at times $\hat{t}+1$, $\hat{t}+2$, ..., $\hat{t}+\hat{T}_\ell$. This can easily be done by considering the buffer system as a (discrete time) birth-death process [KLEI 75] in which the death rate is $\hat{\gamma}_2$, the birth rate at time $\hat{t}+k$ is $\hat{\lambda}(\hat{t}+k)$, and at most one birth and/or one death can occur at each time unit. Let

$$P_j(\hat{t}+k) = \Pr[\text{at time } \hat{t}+k \text{ there are } j \text{ messages in the buffer}] \qquad 0 \le k < \hat{T}_r \tag{5.27}$$

Because there are $n(\hat{t})$ messages in the buffer at time $\hat{t}$, then trivially

$$P_j(\hat{t}) = \begin{cases} 0 & j \neq n(\hat{t}) \\ 1 & j = n(\hat{t}) \end{cases} \tag{5.28}$$

Let $1 \leq k \leq \hat{T}_\ell$ and $\hat{t}' = \hat{t} + k - 1$, then

$$P_j(\hat{t}+k) = \begin{cases} [1-\hat{\lambda}(\hat{t},k)]P_0(\hat{t}') + [1-\hat{\lambda}(\hat{t},k)]\hat{\gamma}_2 P_1(\hat{t}') & \text{if } j = 0 \\ \hat{\lambda}(\hat{t},k)P_0(\hat{t}') + \{[1-\hat{\lambda}(\hat{t}+k)](1-\hat{\gamma}_2) + \hat{\lambda}(\hat{t}+k)\hat{\gamma}_2\}P_1(\hat{t}') \\ \qquad + [1-\hat{\lambda}(\hat{t},k)]\hat{\gamma}_2 P_2(\hat{t}') & \text{if } j = 1 \\ \hat{\lambda}(\hat{t},k)(1-\hat{\gamma}_2)P_{j-1}(\hat{t}') + \{[1-\hat{\lambda}(\hat{t}+k)](1-\hat{\gamma}_2) + \hat{\lambda}(\hat{t}+k)\hat{\gamma}_2\}P_j(\hat{t}') \\ \qquad + [1-\hat{\lambda}(\hat{t}+k)]\hat{\gamma}_2 P_{j+1}(\hat{t}') & \text{if } 1 < j < B \\ \hat{\lambda}(\hat{t},k)(1-\hat{\gamma}_2)P_{B-1}(\hat{t}') + [1-\hat{\gamma}_2 + \hat{\lambda}(\hat{t}+k)\hat{\gamma}_2]P_B(\hat{t}') & \text{if } j = B \end{cases} \tag{5.29}$$

Using these occupancy probabilities, the average number of messages in the buffer can be calculated as follows:

$$\bar{n}(\hat{t}+k) = \sum_{j=1}^{B} P_j(\hat{t}+k)\,j \qquad 0 \leq k \leq \hat{T}_\ell$$

and we have

$$\bar{n}(\hat{t}) = n(\hat{t}) \qquad k = 0 \tag{5.30}$$

and

$$\bar{n}(\hat{t}+k) = \bar{n}(\hat{t}+k-1) + \hat{\lambda}(\hat{t},k)[1-(1-\hat{\gamma}_2)P_B(\hat{t}+k-1)] - \hat{\gamma}_2[1-P_0(\hat{t}+k-1)] \qquad 0 < k \leq \hat{T}_\ell \tag{5.31}$$

The first bracket on the right-hand side of Eq. (5.31) is the probability that at time $(\hat{t}+k)$ a message can be accepted into the buffer. At this time a message arrives with probability $\hat{\lambda}(\hat{t},k)$; therefore, the first term is the increase in the average buffer occupancy at

time $(\hat{t} + k)$ with respect to the average occupancy at time $(\hat{t} + k - 1)$. The second bracket is the probability that at time $(\hat{t} + k - 1)$ the buffer is not empty. If that is the case, a message may leave the buffer with probability $\hat{\gamma}_2$.

We can use Eqs. (5.28) through (5.31) iteratively to find the expected buffer occupancy at time $\hat{t} + \hat{T}_\ell$. Once $\overline{n}(\hat{t} + \hat{T}_\ell)$ is calculated, it is rounded to its closest integer and is used in Algorithm 5.1.

5.3 Implementation of the Look-Ahead Policy - The Decision Table

We have so far answered two of the three questions we addressed in the opening remarks of this chapter. The state of the system (network and the destination buffer) was defined as a vector, the components of which are the buffer occupancy and the future input rates to the destination buffer. After defining a reward function, we gave a heuristic solution for the optimal policy to select the best window size for each state. In this section we address the third question and describe the structure of the decision table, how it is set up, and how it is used by the TD.

In Section 5.2 we discovered how a (sub)optimal stationary policy for the $\hat{T}_r$-cycle-delay Markov decision process can be found heuristically by using the optimal policy for the corresponding $(\hat{T}_r - \hat{T}_\ell)$-cycle-delay decision process. Once an optimal policy for this smaller size decision process is found, we can set up a decision table which has one entry for each state of the $(\hat{T}_r - \hat{T}_\ell)$-cycle-delay decision process. For each state, the corresponding (optimal) window size is listed in the table; therefore

$$\text{Size of the decision table} = (B+1)K^{(\hat{T}_r-\hat{T}_\ell-1)} \tag{5.32}$$

where B is the buffer size, K is the number of different window sizes, $\hat{T}_r$ is the round-trip delay, and $\hat{T}_\ell$ is the look-ahead time. The TD always stores the current state vector of the system; at time $\hat{t}$ this vector is as follows:

$$X(\hat{t}) = \{n(\hat{t}),\ \hat{\lambda}(\hat{t},1),\ \hat{\lambda}(\hat{t},2),\ \ldots,\ \hat{\lambda}(\hat{t},\hat{T}_r-1)\}$$

Each time the TD wants to decide on a window size, it calculates the expected occupancy of the buffer at time $\hat{t}+\hat{T}_\ell$, $\overline{n}(\hat{t}+\hat{T}_\ell)$, and sets up the following vector:

$$X'(\hat{t}) = \{\overline{n}(\hat{t}+\hat{T}_\ell),\ \hat{\lambda}(\hat{t},\hat{T}_\ell+1),\ \ldots,\ \hat{\lambda}(\hat{t},\hat{T}_r-1)\} \tag{5.33}$$

(In fact, the expected buffer occupancy should be rounded to its nearest integer.) $X'(\hat{t})$ is a state vector for the $(\hat{T}_r-\hat{T}_\ell)$-cycle-delay decision process, and the window size for $X'(\hat{t})$ in the decision table is the one that the TD should use. In Figure 5.3 we have shown the decision table when $\hat{T}_\ell = \hat{T}_r - 1$, and the destination buffer size is 6. In this case the decision table specifies an optimal policy for a 1-cycle-delay decision process in which a decision becomes effective on the next transition (which is the case for an ordinary Markov decision process); therefore, the state of the process is simply the buffer occupancy. A decision is an optimal window size for a state. Note that when the buffer occupancy is 5 or 6, the optimal window size is 0, i.e., for the particular value of α which is used for this table, the system is closed

to further input when the occupancy is 5 or 6. (See Table 5.1 for other examples of a decision table.)

State (buffer occupancy)	Decision (window size)
0	4
1	3
2	2
3	1
4	1
5	0
6	0

Figure 5.3 Example of a Decision Table

For Fig. 5.3, $X'(\hat{t})$ given by Eq. (5.33) reduces to a single number, i.e.,

$$X'(\hat{t}) = \bar{n}(\hat{t} + \hat{T}_\ell) = \bar{n}(\hat{t} + \hat{T}_r - 1)$$

The choice of the look-ahead time, $\hat{T}_\ell$, should be based on two considerations: optimality of the decisions made by the TD and feasibility of solving the reduced decision process. For a small $\hat{T}_\ell$, the decisions made by the TD are closer to the optimal (note that for $\hat{T}_\ell = 0$, the TD's decisions are optimal); however, as we pointed out earlier, the resulting $(\hat{T}_r - \hat{T}_\ell)$-cycle-delay decision process is too large to solve, and also the size of the decision table becomes too large, see Eq. (5.32). On the other hand, when $\hat{T}_\ell$ is large the result of the decision made by the TD may be far from optimal; nevertheless, a large $\hat{T}_\ell$ also results in a small decision process to solve and a small decision table. The

largest value of $\hat{T}_\ell$ is $\hat{T}_r - 1$, for which the state of the reduced decision process is simply the buffer occupancy. For $\hat{T}_\ell = \hat{T}_r - 2$, $X'(\hat{t})$ has two components which corresponds to the state of a 2-cycle-delay decision process. Although this system is not too large to solve (the size of state space for B = 10 and K = 5 is 55), a problem which arises is that the resulting state transition probability matrix (Eq. (5.12)) may have more than one irreducible chain; therefore, the solution to the decision process becomes very complex and time consuming. (The policy iteration algorithm described in Appendix C is for single chain Markov processes; for multi-chain processes some modifications to the algorithm should be made, see [HOWA 60, 71].) For values of $\hat{T}_\ell$ less than $\hat{T}_r - 2$, not only does that state space become large, but also we face the problem of multi-chain systems. For these reasons, $\hat{T}_\ell = \hat{T}_r - 1$ is a convenient choice, and we have used this value of $\hat{T}_\ell$ in our numerical examples. To support this choice of $\hat{T}_\ell$, we also used $\hat{T}_\ell = \hat{T}_r - 2$ and found that the performance of the network (in terms of throughput and delay) was almost identical to the case when $\hat{T}_\ell = \hat{T}_r - 1$. In the next section, we elaborate on Markov decision processes with a lead time equal to one (i.e., a decision becomes effective on the next transition).

5.4 1-Cycle-Delay and Continuous Time Decision Processes

When the look-ahead time $\hat{T}_\ell$ is equal to $(\hat{T}_r - 1)$, the decision table becomes an optimal (stationary) policy of a 1-cycle-delay (discrete time) Markov decision process. In this process decisions become effective in the next state transition, which is the case for an ordinary discrete time Markov decision process. The general formulation of

$\hat{T}_r$-cycle-delay processes we presented in Section 5.1 also applies to a discrete time Markov decision process; however, because we have used $\hat{T}_\ell = \hat{T}_r - 1$ in our numerical examples (Section 5.5), we find it useful to elaborate briefly on this 1-cycle-delay process.

The state of the process is the number of messages in the buffer

$$S = \{0, 1, \ldots, B\} \tag{5.34}$$

where B is the buffer size. The action space A, as defined in Section 5.1.2, is the set of window sizes W (or equivalently input rates Λ) that can be used (Eq. 5.7). Corresponding to each window size, say w_k, there is a network delay $\hat{T}_k$ and a timeout period $\hat{\tau}_k$ (we will shortly describe how these quantities are calculated). A (stationary) policy f is a function from the state space S to the action space A (i.e., $f: S \to A$) such that if $f(i) = k$, $i \in S$, we use w_k (or $\hat{\lambda}_k$) when the system is in state i. The transition probabilites are basically similar to Eq. (5.12) We use $p_{ij}(f)$ to indicate the transition probability from state i to state j when (stationary) policy f is used. If $f(i) = k$, then

$$p_{ij}(f) = \begin{cases} \hat{\lambda}_k(1-\hat{\gamma}_2) & j = i+1 \\ 1 - \hat{\lambda}_k - \hat{\gamma}_2 + 2\hat{\lambda}_k\hat{\gamma}_2 & j = i \\ (1-\hat{\lambda}_k)\hat{\gamma}_2 & j = i-1 \end{cases} \quad 0 < i < B$$

$$\begin{cases} \hat{\lambda}_k & j = 1 \\ (1-\hat{\lambda}_k) & j = 0 \end{cases} \quad i = 0 \tag{5.35}$$

$$\begin{cases} 1 - \hat{\gamma}_2 + \hat{\lambda}_k\hat{\gamma}_2 & j = B \\ (1-\hat{\lambda}_k)\hat{\gamma}_2 & j = B-1 \end{cases} \quad i = B$$

We denote the matrix of transition probabilites under policy f by $P(f)$. As in Section 5.1.5, we can find the expected immediate reward rate at state i under policy f.

$$R_i(f) = \begin{cases} \alpha\hat{\lambda}_k - (i + \hat{\lambda}_k\hat{T}_k) & \text{if } i < B \text{ and } f(i) = k \\ \alpha\hat{\lambda}_k\hat{\gamma}_2 - [i + \hat{\lambda}_k\hat{\gamma}_2\hat{T}_k + \hat{\lambda}_k(1-\hat{\gamma}_2)\hat{\tau}_k] & \text{if } i = B \text{ and } f(i) = k \end{cases}$$

We refer to $\{R_i(f)\}$ by $R(f)$. Our objective is to find a stationary policy f* which maximizes the components of the $(B+1)$-vector of the expected reward rates

$$g(f^*) = \max_{f \in F} g(f)$$

where $g(f) = \{g_i(f)\}$ is the vector of long run expected rewards per unit time when policy f is used (Eq. (5.20)). It can be shown that such an optimal policy always exists; furthermore, when $P(f)$ has one irreducible chain, then all components of $g(f)$ are equal and we can use the algorithm given in Appendix C to find this policy.

When the time increment δ is very small, decisions become effective very fast; in the limit as $\delta \to 0$, the above discrete time Markov process becomes a continuous time process. In the continuous time mode the rate of transitions (rather than the probability of a transition) out of a state is a function of the action taken in that state. Because in this process decisions become effective instantaneously, the optimal reward rate for the continuous time process is the best performance we can expect from the system, and the throughput for the optimal policy f* is the highest that can be expected. Therefore, we will compare the performance of our look-ahead policy with the performance of this

continuous time model.

Consider a continuous time Markov decision process in which actions are taken according to a stationary policy f; in this case the policy f specifies the input rate to the destination buffer; therefore the input rates to the buffer are functions of the states as shown in Fig. 5.4. For this system we can find the distribution of the equilibrium occupancy probabilities and, in particular, the average input rate to the system

$$\lambda_{in}(f) = \sum_{i=0}^{B-1} P_i(f)\lambda_{f(i)} \tag{5.36}$$

Because each message that is accepted to the buffer is guaranteed to be delivered to the receiving user, $\lambda_{in}(f)$ is also the long run throughput of the system under policy f. If $\lambda_{f(B)}$ (the decided input rate when the buffer is full) is not zero, then $P_B(f)$ is the probability of loss when window sizes (or input rates) are selected under policy f

$$P_\ell = P_B(f) \tag{5.37}$$

As we see shortly, in order to find the set of input rates Λ, we must know this probability.

5.4.1 A Procedure to Find the Set of Input Rates

So far we have assumed that for a set of window sizes W, we can use the results of our static flow control (Chapter 4) to find the optimal input rates (or throughputs) of the network $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_K\}$. In Chapter 4 we found that if the window size w, channel capacity γ_1, and the loss probability P_ℓ of a network are known, then an optimal timeout (τ_t^*) can be found such that the throughput of the network

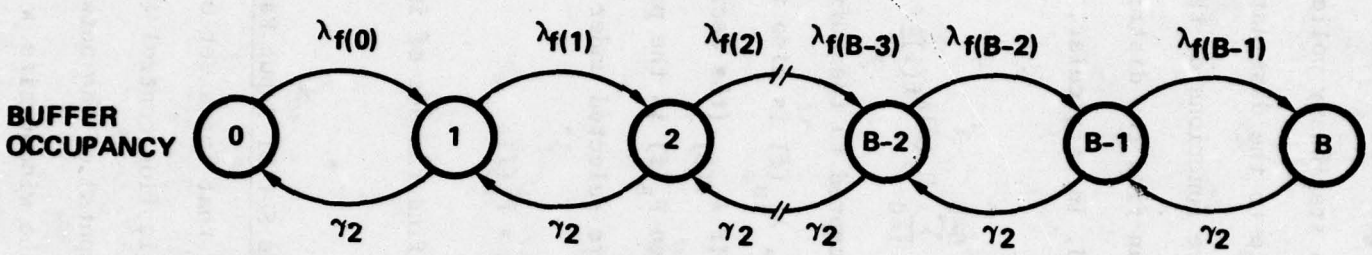


Figure 5.4 - State-Transition-Rate Diagram for the Destination Buffer.

becomes maximal (Eqs. (4.14) and (4.16)). The problem which we face here is that the loss probability, P_ℓ, is itself a function of the (optimal) policy that we use to select window sizes (Eq. 5.37)). On the other hand, the optimal policy is determined by using the set of input rates Λ; therefore, evaluation of input rates $\{\lambda_k\}$ depends on the loss probability P_ℓ, which is itself a function of the input rates. To remove this recurrence, we can use a linear iteration method given in Algorithm 5.2.

Algorithm 5.2

1. Initialize P_ℓ (e.g., $P_\ell = 0$).
2. Find the set of input rates for the set of window sizes W.
3. Use the set of input rates, Λ, in the policy-iteration method to find an optimal policy f*.
4. Find the new loss probability, $P'_\ell = P_B(f^*)$.
5. If $|P_\ell - P'_\ell| > \epsilon$ set $P_\ell = P'_\ell$ and go to step 2.
6. Stop.

We start the procedure by initializing the loss probability P_ℓ (an initial value of 0 is a good choice). With this probability of loss we can find the set of optimal input rates Λ corresponding to the set of window sizes W. Having found Λ (and the corresponding network delays and timeout periods), we can use the policy-iteration method to find an optimal policy f* which maximizes the reward rate, step 3. Corresponding to this policy, there is a loss probability P'_ℓ, Eq. (5.37). If P'_ℓ and P_ℓ are not close enough to each other, we repeat steps 2 to 4 using the new loss probability P'_ℓ instead of P_ℓ. When the difference between P'_ℓ

and P_ℓ is small, policy f* found in Step 3 is the "optimal" policy (within the tolerance ε) that we should use in our decision table. In all of our numerical examples Algorithm 5.2 converged in less than six iterations; however, this algorithm is not always guaranteed to terminate - in a narrow sense it should be referred to as a procedure.

5.5 Numerical Results

In this section we present numerical examples for dynamic flow control, study the optimality of the look-ahead policy, and investigate the effect of the parameter α on the throughput-delay performance of a network. As in Chapter 4, throughout our presentation we consider the normalized value of certain parameters; the normalization constant is $\bar{X}_1 = 1/(\mu C_1)$, the average transmission time of a message on a network channel.

We begin by studying the throughput-delay performance of a continuous time model in which the effect of a decision is instantaneous. This means, based on the occupancy of the buffer, when a new window size (or input rate) is decided upon, it becomes effective instantaneously in the next state transition. Therefore, in this hypothetical system there is no time gap from the moment that a decision is made until it becomes effective. Hence, performance of the continuous time model is the best that can be achieved.

As we pointed out earlier, the value of α reflects our relative preference for high throughput over low delay. Table 5.1 shows the effect of parameter α on the throughput-delay performance of the system. In this table the destination buffer size (B) is 10, the channel

$B = 10$ $\qquad C_1 = C_2 = 50$ kbps

$W = \{0, 1, 2, \ldots, 20\}$ $\qquad 1/\mu = 1000$ bits

$\alpha = 0.4$

STATE	PROB	WINDOW
0	0.310	6
1	0.232	5
2	0.176	5
3	0.128	4
4	0.078	3
5	0.047	3
6	0.028	2
7	0.014	1
8	0.001	0
9	0.00	0
10	0.00	0

Average window:	4.77
Average buffer occupancy:	1.78
Average throughput:	34.51
Average buffer delay:	0.05
Average network delay:	0.07
Average total delay	0.12

$\alpha = 0.6$

STATE	PROB	WINDOW
0	0.248	8
1	0.199	8
2	0.159	7
3	0.123	6
4	0.092	5
5	0.066	5
6	0.047	4
7	0.031	3
8	0.019	2
9	0.009	2
10	0.004	0

Average window:	6.66
Average buffer occupancy:	2.38
Average throughput:	37.59
Average buffer delay	0.06
Average network delay:	0.09
Average total delay:	0.15

Table 5.1. Example of optimal window selection for a continuous time model.

capacities of the network and destination node (C_1 and C_2) are both 50 kbps, the average message length is 1000 bits, and the set of allowable window sizes is $W = \{0, 1, 2, \ldots, 20\}$. This table shows the optimal policies and their corresponding statistics for $\alpha = 0.4$ and $\alpha = 0.6$. It is seen that for $\alpha = 0.6$, larger window sizes are used (and a higher throughput results); for example, when the buffer is empty (state 0) for $\alpha = 0.4$, $w = 6$ is the optimal decision, whereas when $\alpha = 0.6$, $w = 8$ becomes the best decision. Although the delay (and throughput) due to $w = 8$ is larger than $w = 6$, a window size of 8 becomes the optimal decision because $\alpha = 0.6$ puts more weight on high throughput than $\alpha = 0.4$. It is seen that not all of the allowable window sizes are used in either of the above cases; therefore, not only can the parameter α be used to control the throughput, but the maximum window size can be controlled also by a proper selection of α. Considering the fact that the maximum window size which is used reflects the number of buffers needed at the source node, the importance of this parameter becomes clear. Furthermore, the number of buffers required at the destination node is also determined by α (for example, for $\alpha = 0.4$ a buffer size of $B = 8$ is sufficient); therefore, this parameter can also be used to control the utilization of the destination node buffer.

Figure 5.5 shows the throughput-delay (end-to-end or total delay) performance of the continuous time model for different buffer sizes for the destination node. The set of allowable window sizes and other parameters of the system are the same as in Table 5.1. For a fixed B, the average throughput and delay increase as the parameter α increases. For a value of α an optimal decision is determined such that

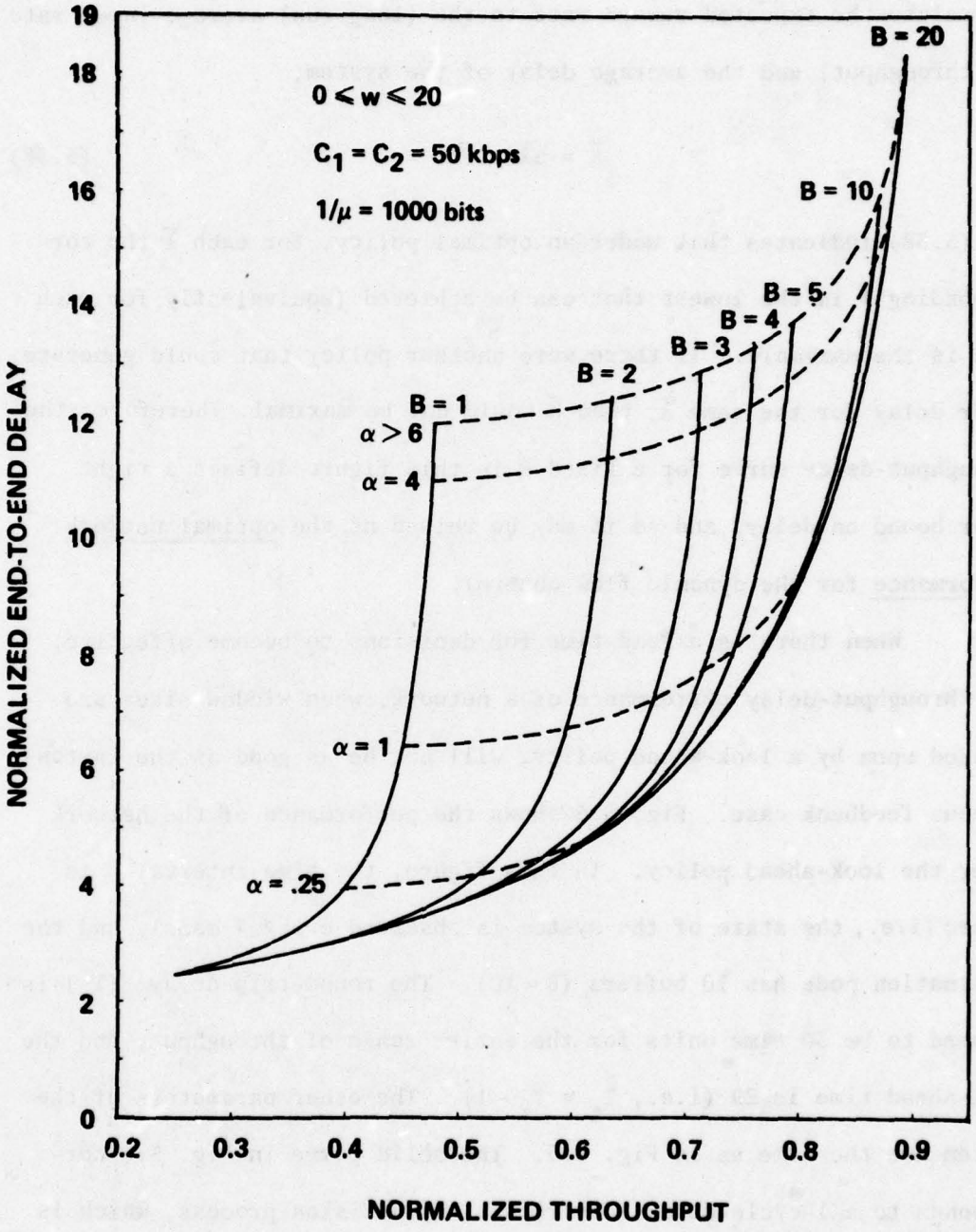


Figure 5.5 - Throughput-Delay Performance of Continuous Time Model For Different Destination Buffer Sizes.

the expected (long run) reward rate becomes maximal. From Eq. (5.13) we can relate the expected reward rate to the (long run) average input rate (or throughput) and the average delay of the system,

$$\overline{R} = \alpha\overline{\lambda} - \overline{T} \tag{5.38}$$

Eq. (5.38) indicates that under an optimal policy, for each $\overline{\lambda}$ the corresponding $\overline{T}$ is the lowest that can be achieved (equivalently for each $\overline{T}$, $\overline{\lambda}$ is the maximal). If there were another policy that could generate lower delay for the same $\overline{\lambda}$, then $\overline{R}$ would not be maximal. Therefore, the throughput-delay curve for a fixed B in this figure defines a tight lower bound on delay, and so it may be veiwed as the optimal network performance for the dynamic flow control.

When there is a lead-time for decisions to become effective, the throughput-delay performance of a network, when window sizes are decided upon by a look-ahead policy, will not be as good as the instantaneous feedback case. Fig. 5.6 shows the performance of the network under the look-ahead policy. In this figure, the time interval δ is 4 msec (i.e., the state of the system is observed every 4 msec), and the destination node has 10 buffers (B = 10). The round-trip delay $(\hat{T}_r)$ is assumed to be 30 time units for the entire range of throughput, and the look-ahead time is 29 (i.e., $\hat{T}_\ell = \hat{T}_r - 1$). The other parameters of the system are the same as in Fig. 5.5. The solid curve in Fig. 5.6 corresponds to a 1-cycle-delay (discrete time) decision process, which is the counterpart of the continuous time model; therefore, this curve represents the best performance that can be achieved in the discrete time model from a look-ahead policy. In order to investigate its performance,

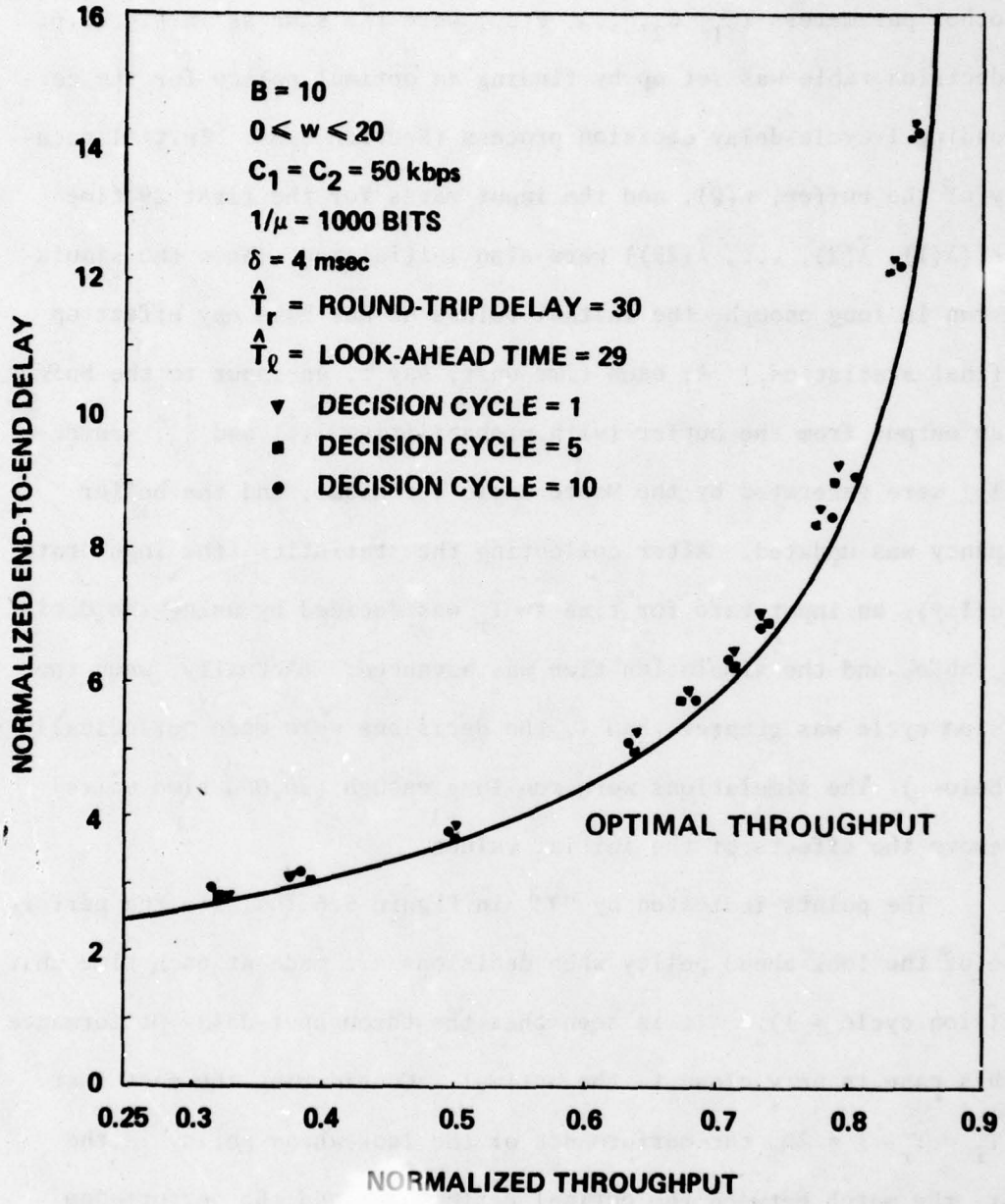


Figure 5.6 - Throughput-Delay Performance of Look-Ahead Policy.

we tested the look-ahead policy by simple simulation experiments.

In the simulation experiments we used $\hat{T}_r = 30$ and $\hat{T}_\ell = \hat{T}_r - 1 = 29$. The other parameters ($C_1$, C_2, ..., etc.) were the same as in Fig. 5.6. The decision table was set up by finding an optimal policy for the corresponding 1-cycle-delay decision process (Section 5.3). Initial occupancy of the buffer, $n(0)$, and the input rates for the first 29 time units ($\hat{\lambda}(1)$, $\hat{\lambda}(2)$, ..., $\hat{\lambda}(29)$) were also initialized. (When the simulation run is long enough, the initial values do not have any effect on the final statistics.) At each time unit, say $\hat{t}$, an input to the buffer and an output from the buffer (with probabilities $\hat{\lambda}(\hat{t})$ and $\hat{\gamma}_2$, respectively) were generated by the Monte Carlo technique, and the buffer occupancy was updated. After collecting the statistics (the input rate and delay), an input rate for time $\hat{t} + \hat{T}_r$ was decided by using the decision table, and the simulation time was advanced. (Actually, when the decision cycle was greater than 1, the decisions were made periodically; see below.) The simulations were run long enough (20,000 time units) to remove the effects of the initial values.

The points indicated by "▼" in Figure 5.6 indicate the performance of the look-ahead policy when decisions are made at each time unit (decision cycle = 1). It is seen that the throughput-delay performance in this case is very close to the optimal. Considering the fact that for $\hat{T}_\ell = \hat{T}_r - 1 = 29$, the performance of the look-ahead policy is the worst, the match between the optimal performance and the performance due to the look-ahead policy is very encouraging.

We pointed out earlier that control packets to carry decisions to the source node are piggybacked on messages going from the destination

node to the source node, and if there is no such message, control packets are transmitted by themselves. Therefore, signalling of control packets generates overhead traffic. For this reason it is not a good policy to decide on a new window size at each time unit and it is better to make decisions periodically. In Fig. 5.6 we have shown the performance of the look-ahead policy when decisions are made every 5 time units ("■"), or every 10 time units ("●"). (When the decision cycle is, say 5, the decision made at time $\hat{t}$ is used as the decision for times $\hat{t}+1$ through $\hat{t}+4$; at time $\hat{t}+5$ a new decision is made.) For decision cycles larger than 10 the throughput-delay curves differed from the optimal (the solid curve) by too much. Therefore, as long as the decision cycle is below a certain limit, the throughput-delay performance of the periodic look-ahead policy is not sensitive to the exact value of the decision cycle. This property is important, as a periodic decision reduces the overhead due to signalling of control packets.

So far we have assumed that the round-trip delay, $\hat{T}_r$, is fixed for the entire range of throughput; in reality $\hat{T}_r$ is larger for higher throughputs. To investigate the sensitivity of the throughput-delay performance to round-trip delay, we used the simulation program described earlier, but changed the round-trip delay as the throughput increased (the look-ahead time was also changed such that $\hat{T}_\ell = \hat{T}_r - 1$). The points indicated by "●" in Fig. 5.7 correspond to the performance of the system when $\hat{T}_r$ varies. The solid curve in this figure is the throughput-delay curve for the 1-cycle-delay decision process for which the round-trip delay is fixed at $\hat{T}_r = 30$. This figure indicates that the performance of the look-ahead policy is not sensitive to a variation in $\hat{T}_r$, and if a

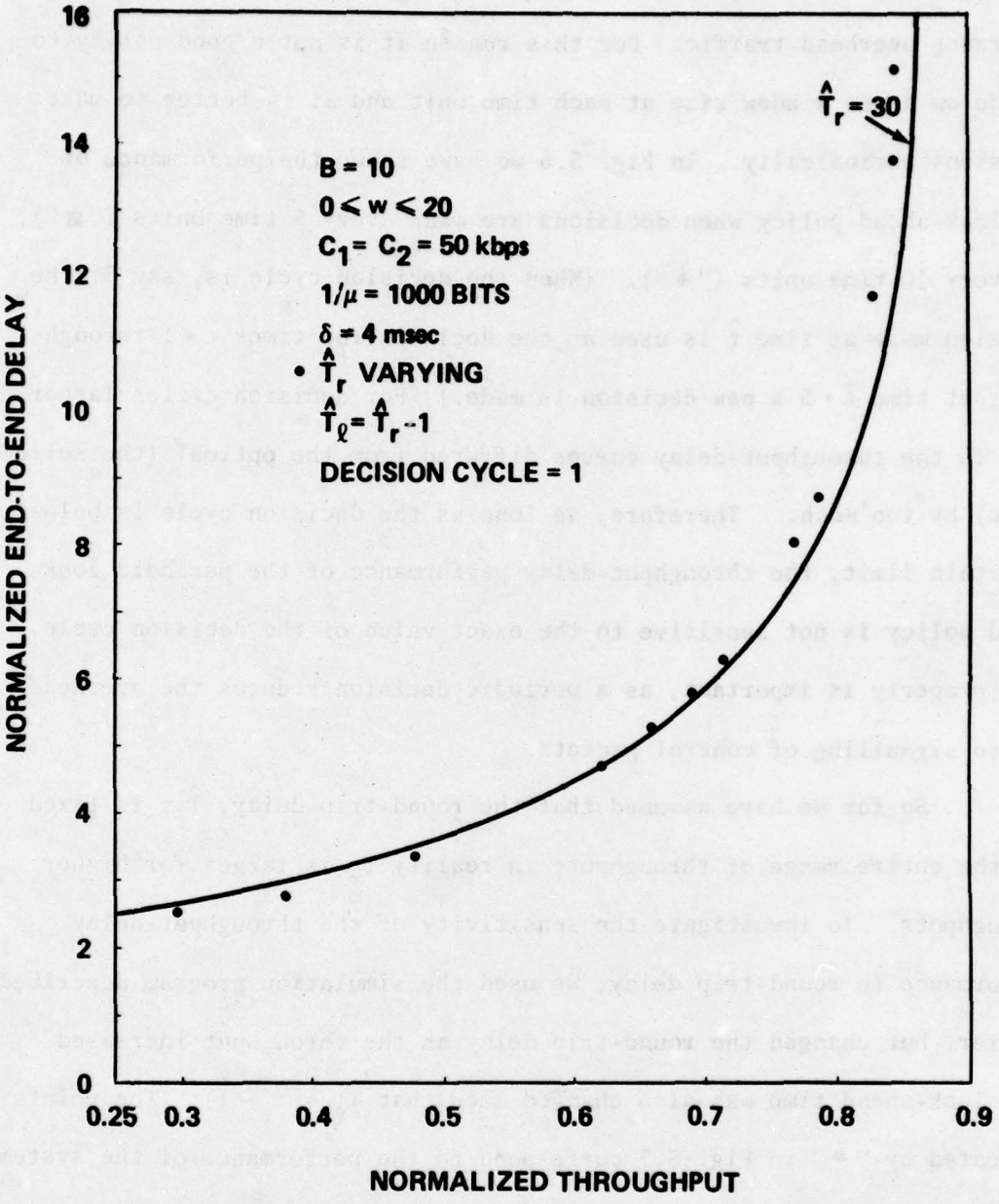


Figure 5.7 - Throughput-Delay Performance of Look-Ahead Policy for Varying Round-Trip Delay.

proper average round-trip delay is used, the performance is close to the optimal.

In order to investigate the sensitivity of the throughput-delay performance on the look-ahead time ($\hat{T}_\ell$), we simulated a network with parameters shown in Fig. 5.6, but we let $\hat{T}_\ell = \hat{T}_r - 2 = 28$. The performance in this case was almost identical to the case when $\hat{T}_\ell = 29$. To use a look-ahead time smaller than $\hat{T}_r - 2$, we must find an optimal policy for the corresponding n-cycle delay Markov decision process, where $n > 2$. As pointed out earlier, the solution is fairly complicated and time consuming because the corresponding Markov processes may have more than one irreducible chain. Considering the fact that a maximum look-ahead time (i.e., $\hat{T}_\ell = \hat{T}_r - 1$) results in a performance very close to the optimal (Fig. 5.6), $\hat{T}_r - 1$ is a convenient and good choice for the look-ahead time ($\hat{T}_\ell$).

5.6 Conclusion

In this chapter we developed and analyzed a dynamic decision policy to select the window size to control the flow in store-and-forward computer networks. The dynamic flow control developed here is essentially an extension of the static flow control studied in the previous chapter.

Based on the Markov decision process, the decision problem was formulated, and a heuristic solution to an optimal policy was presented as an alternate to the laborious exact solution. The formulation and the heuristic solution is general in the sense that it can be applied to any decision process in which there is a lead-time for a decision to

become effective, a common situation in inventory systems. We introduced a parameter α which reflects our preference for throughput over delay. This parameter can be considered as a further control tool to limit the throughput, the delay, the average and/or the maximum window size, and also to limit the destination buffer utilization (Table 5.1).

The numerical examples of Section 5.5 indicate that the performance of a network when window sizes are selected according to a sub-optimal policy determined by the heuristic solution is very close to the performance of an optimal policy, Fig. 5.6. This figure further indicates that the performance of a periodic decision is very close to the optimal so long as the decision period is chosen properly. This property is of significant importance in practical application, as a periodic decision reduces the overhead due to the signalling of control packets. Lastly, comparison of Fig. 5.6 and Fig. 4.15 shows that the throughput-delay of a network is better than the performance under a static control when the window sizes are chosen dynamically. Therefore, dynamic flow control can further enhance the traffic handling capability of computer networks.

CHAPTER 6

INTERCONNECTION OF NETWORKS

The interconnection of computer networks has recently become a subject of interest. The idea is to make the computing power and resources of a network available to users and computers attached to other networks [CERF 74], [SUNS 75]. In this chapter we study some issues related to the interconnection of computer networks, otherwise known as internetting. As in previous chapters, we aim to develop analytic tools for the performance study and design of computer networks. In many cases, the models which we use are rather simplistic; nevertheless, they help us understand the implications of real system behavior. We begin by establishing a network algebra as an aid in the systematic study of computer network interconnection; we then look at some design problems for allocation of resources among a number of parallel-connected networks. Finally, we discuss buffer allocation strategies in nodes where a number of networks merge together.

6.1 A Network Algebra

This section is concerned with the performance analysis of systems composed of series or parallel interconnections of a number of computer nets. In particular, in analogy with the basic laws of electrical circuits (Kirchoff's Laws), we develop a network algebra for the systematic study of computer networks, and characterize the equivalent network (defined below) for such interconnected nets.

For the purpose of the present study we need closed form expressions for the maximum throughput and delay of a network in terms of its system parameters (window size, channel capacities, etc.). Such expressions could not be found for the model developed in Section 4.4, hence we use the (less realistic) results of our study in Section 4.3. According to that analysis we have the following expressions for the maximum network throughput and delay:

$$\lambda^* = \frac{w}{w + 2n_h}\gamma \tag{4.5}$$

$$T^* = \begin{cases} \dfrac{w + 2n_h}{2\gamma} & w > 0 \quad \text{and} \quad \gamma > 0 \\ 0 & w = 0 \quad \text{or} \quad \gamma = 0 \end{cases} \tag{4.7}$$

where w is the window size, n_h is the path length (number of hops) and $\gamma (= \mu C)$ is the maximum transmission rate of the channels. (Note that when $w = 0$ or $\gamma = 0$ there is no traffic, hence the delay is defined to be zero.) The above equations can be written in the following form

$$\lambda^* = \frac{\ell^*}{\ell^* + 2}\gamma \tag{6.1}$$

$$T^* = \begin{cases} \dfrac{\ell^* + 2}{2\gamma} n_h & \ell^* > 0 \quad \text{and} \quad \gamma > 0 \\ 0 & \ell^* = 0 \quad \text{or} \quad \gamma = 0 \end{cases} \tag{6.2}$$

where $\ell^* = w/n_h$

The parameter ℓ^*, as we will see shortly, plays an important role in our study and we find it useful to reflect on its physical interpretations. The window size w, as we have used in our analysis, is the average number of messages in the network (more precisely, the average

number of messages on the network path under our study). There are n_h hops (or nodes) on the path (hence there are $2n_h$ hops on a round trip); therefore, when the network carries a traffic determined by Eq. (6.1), $\ell^* = w/n_h$ will be the average number of messages in a node, of which $\ell^*/2$ are being transmitted to the destination and the other half are carrying acknowledgements to the source node (recall that we have assumed acknowledgements are piggybacked on messages sent from the destination node to the source node). In queueing theory the average number of customers in a system may be considered to be the system load [AGNE 74]; therefore, $\ell^* = w/n_h$ is the load (or the load factor) of a node (and $\ell^*/2$ is the load of a channel) on the path due to the maximum traffic from our source to destination. Note that through the window control, the load of a node is limited to w/n_h and Eq. (6.1) determines the traffic which generates this load, hence λ^* is the throughput capacity of the network. If, due to some limitations (which we discuss shortly), the throughput (λ) is less than λ^* given by Eq. (6.1), then the load on the nodes is simply

$$\ell = \frac{2\lambda}{\gamma - \lambda} \tag{6.3}$$

On the other hand, if the load of the nodes is known (say $\ell \leq \ell^*$), then the throughput and delay of the network will be

$$\lambda = \frac{\ell}{\ell + 2}\ \gamma \qquad \text{for } 0 \leq \ell \leq \ell^* \tag{6.4}$$

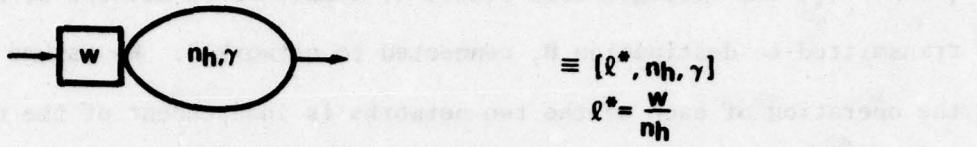
$$T = \begin{cases} \dfrac{\ell + 2}{2\gamma}\, n_h & 0 < \ell \leq \ell^* \text{ and } \gamma > 0 \\ 0 & \ell = 0 \quad \text{or} \quad \gamma = 0 \end{cases} \tag{6.5}$$

In the sequel we will refer to $\ell^* = w/n_h$ as the maximum load factor; to λ^* as the traffic handling capacity; and to T^* as the maximum delay of a network. (When there is no ambiguity we refer to λ^* and T^* simply by throughput and delay.) Note that the load factor (ℓ) is different from the utilization factor ρ ($= \lambda/\mu C$, where λ is the traffic rate and μC is the maximum transmission rate of channels) defined as the fraction of the channel capacity used for transmission.

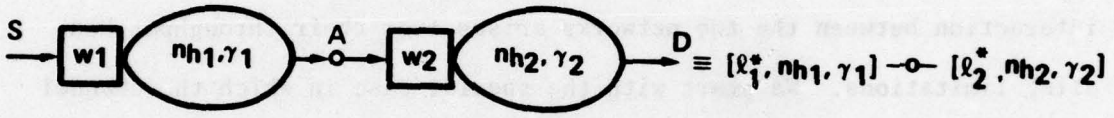
Eqs. (6.1) and (6.2) show that a network (for the purpose of studying its throughput and delay) can be totally characterized by three parameters: γ ($= \mu C$), the transmission rate of the channels; n_h, the number of hops messages should go through; and ℓ^* ($= w/n_h$), the maximum load factor of nodes. Symbolically we designate a network by a triple $[\ell^*, n_h, \gamma]$ (Fig. 6.1-a). In this figure the box with a "w" inside indicates a TC with window size w in front of a network with path length "n_h" and channel transmission rate γ. In what follows we will find a network $[\ell_e^*, n_{h_e}, \gamma_e]$ equivalent to a system which is composed of series and/or parallel connections of networks. The symbols "$-o-$" and "$-=-$" will be used to indicate series and parallel connection of two networks, respectively (Figs. 6.1-b and 6.1-c). Throughout this section we assume that $w > 0$ and $\gamma > 0$. We start by defining equivalent networks.

Definition 6.1

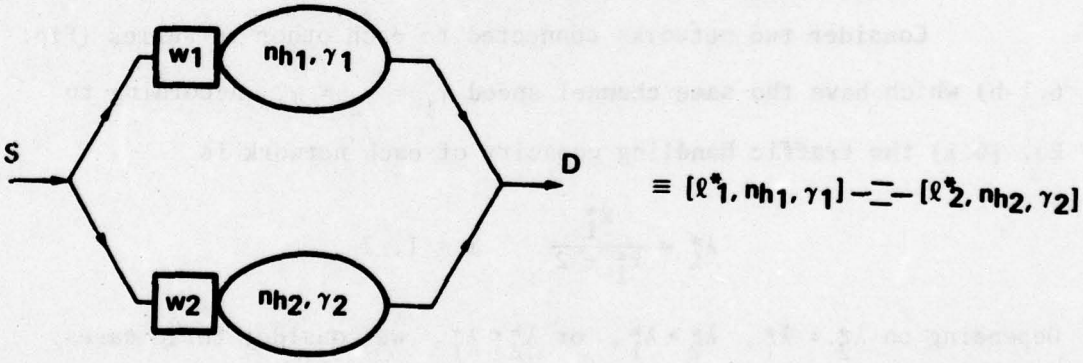
Two networks are said to be equivalent if the traffic handling capacity and the maximum delay of both are identical.



(a) A Network



(b) Series Connection



(c) Parallel Connection

Figure 6.1 Short Hand Notation.

6.1.1 Series Connection of Networks

Consider Fig. 6.1-b in which two networks are connected in series (the connecting node A is usually referred to as the GATEWAY [CERF 74]), and messages from source S, connected to network 1, are transmitted to destination D, connected to network 2. We assume that the operation of each of the two networks is independent of the other, which means that when a message from network 1 is delivered to network 2 (via the GATEWAY), the ACK for this message is sent back to the TC of network 1; a copy of this message is kept by the TC of the second network until the ACK is received from the destination node. The only interaction between the two networks arises from their throughput handling limitations. We start with the special case in which the channel transmission rates of both networks are equal.

6.1.1.1 Equal Channel Transmission Rates

Consider two networks connected to each other in series (Fig. 6.1-b) which have the same channel speed $\gamma_1 = \gamma_2 = \gamma$. According to Eq. (6.1) the traffic handling capacity of each network is

$$\lambda_i^* = \frac{\ell_i^*}{\ell_i^* + 2} \qquad i = 1, 2$$

Depending on $\lambda_2^* = \lambda_1^*$, $\lambda_2^* > \lambda_1^*$, or $\lambda_2^* < \lambda_1^*$, we consider three cases.

Case I: $\lambda_2^* = \lambda_1^*$

In this case, the maximum throughput of the system is λ_e^*, where

$$\lambda_e^* = \lambda_1^* = \frac{\ell_1^*}{\ell_1^* + 2}\gamma$$

$$= \lambda_2^* = \frac{\ell_2^*}{\ell_2^* + 2}\gamma \qquad (6.6)$$

From Eq. (6.6) we find

$$\ell_1^* = \ell_2^*$$

i.e., the maximum load factor of the two networks is identical. The maximum total delay is the sum of the delays in the two networks,

$$T_e^* = T_1^* + T_2^*$$

$$= \frac{\ell_1^* + 2}{2\gamma} n_{h_1} + \frac{\ell_2^* + 2}{2\gamma} n_{h_2}$$

or

$$T_e^* = \frac{\ell_1^* + 2}{2\gamma} [n_{h_1} + n_{h_2}]$$

$$= \frac{\ell_2^* + 2}{2\gamma} [n_{h_1} + n_{h_2}] \qquad (6.7)$$

Comparing Eqs. (6.6) with (6.1), and Eqs. (6.7) with (6.2), we note that when $\ell_1^* = \ell_2^*$, then the two networks connected in series are equivalent to a network with the maximum load factor of

$$\ell^* = \ell_1^* = \ell_2^*$$

and the equivalent number of hops is the sum of the number of hops of the two networks. Thus we have

$$[\ell^*, n_{h_1}, \gamma] \multimap [\ell^*, n_{h_2}, \gamma] \Rightarrow [\ell^*, n_{h_1} + n_{h_2}, \gamma]$$

The equivalent window size is

$$w_e = \frac{n_{h_1} + n_{h_2}}{n_{h_1}} w_1 = \frac{n_{h_1} + n_{h_2}}{n_{h_2}} w_2$$

The throughput handling capacity and the maximum delay of the equivalent network is given by Eqs. (6.6) and (6.7).

Case II: $\lambda_2^* > \lambda_1^*$

In this case network 1 is the bottleneck and its traffic handling capacity determines the system throughput.

$$\lambda_e^* = \lambda_1^*$$

$$= \frac{\ell_1^*}{\ell_1^* + 2} \gamma$$

Note that when $\gamma_1 = \gamma_2$, then $\lambda_1^* < \lambda_2^*$ is equivalent to $\ell_1^* < \ell_2^*$, i.e., the network with the smaller maximum load factor is the bottleneck. The delay of network 2 can be calculated by considering the fact that for a traffic equal to λ_1^*, the load at the nodes of network 2 is ℓ_1^* (notice that $\lambda_e^* = \lambda_1^*$). From Eq. (6.5) we have

$$T_2 = \frac{\ell_1^* + 2}{2\gamma} n_{h_2}$$

Hence the total delay becomes

$$T_e^* = T_1^* + T_2$$

$$= \frac{\ell_1^* + 2}{2\gamma} n_{h_1} + \frac{\ell_1^* + 2}{2\gamma} n_{h_2}$$

$$= \frac{\ell_1^* + 2}{2\gamma} (n_{h_1} + n_{h_2})$$

Therefore, when $\lambda_2^* > \lambda_1^*$ we have

$$[\ell_1^*, n_{h_1}, \gamma] \multimap [\ell_2^*, n_{h_2}, \gamma] \Rightarrow [\ell_1^*, n_{h_1} + n_{h_2}, \gamma]$$

The equivalent window size will become

$$w_e = \frac{n_{h_1} + n_{h_2}}{n_{h_1}} w_1$$

Case III: $\lambda_2^* < \lambda_1^*$

In this case, contrary to case II, network 2 is the bottleneck and its traffic handling capacity determines the system throughput. The expressions for the different quantities of interest are exactly like the ones for Case II, only indices 1 and 2 should be interchanged.

We can generalize the above results for the case where more than two networks are involved.

If $\gamma_i = \gamma$ for $i = 1, 2, \ldots, N$, then

$$\sum_{i=1}^{N} (-o-) \; [\ell_i^*, n_{h_i}, \gamma] \Rightarrow [\ell_{i*}^*, \sum_{1}^{N} n_{h_i}, \gamma]$$

where i* is such that

$$\lambda_{i*}^* = \min_{1 \le i \le N} \{\lambda_i^*\}$$

The equivalent window size will be

$$w_e = \frac{\sum_i n_{h_i}}{n_{h_{i*}}} w_{i*}$$

The maximum throughput and delay of the system will be

$$\lambda_e^* = \lambda_{i*}^*$$

and

$$T_e^* = \frac{\ell_{i*}^* + 2}{2\gamma} \left[\sum_i n_{h_i}\right]$$

6.1.1.2 Non-Equal Channel Transmission Rates

When the channel transmission rates of networks are not identical, the analysis becomes a bit more complex as we will see shortly. As in Section 6.1.1.1, we start with two networks connected in series and consider three cases: $\lambda_2^* = \lambda_1^*$, $\lambda_2^* > \lambda_1^*$ and $\lambda_2^* < \lambda_1^*$.

Case I: $\lambda_2^* = \lambda_1^*$

When the traffic handling capacity of the two networks is equal we have

$$\frac{\ell_1^*}{\ell_1^* + 2} \gamma_1 = \frac{\ell_2^*}{\ell_2^* + 2} \gamma_2$$

Solving the above equation for ℓ_1^* we get

$$\ell_1^* = 2 \frac{\ell_2^* \gamma_2}{\ell_2^* (\gamma_1 - \gamma_2) + 2\gamma_1}$$

Therefore, the maximum load factors of the two networks are related to each other by the following relationship:

$$\ell_i^* = 2 \frac{\ell_j^* \gamma_j}{\ell_j^* (\gamma_i - \gamma_j) + 2\gamma_i} \qquad i,j = 1,2 \quad \text{and} \quad i \neq j$$

The throughput of the system will be

$$\lambda_e^* = \lambda_1^* = \frac{\ell_1^*}{\ell_1^* + 2} \gamma_1$$

$$= \lambda_2^* = \frac{\ell_2^*}{\ell_2^* + 2} \gamma_2$$

The maximum system delay is the sum of the delay in the two networks:

$$T_e^* = T_1^* + T_2^*$$

$$= \frac{\ell_1^* + 2}{2\gamma_1} n_{h_1} + \frac{\ell_2^* + 2}{2\gamma_2} n_{h_2}$$

or

$$T_e^* = \frac{\ell_1^* + 2}{2\gamma_1} \left[1 + \frac{2}{\ell_1^* \left(\frac{\gamma_2}{\gamma_1} - 1 \right) + 2 \frac{\gamma_2}{\gamma_1}} \frac{n_{h_2}}{n_{h_1}} \right] n_{h_1}$$

$$= \frac{\ell_2^* + 2}{2\gamma_2} \left[1 + \frac{2}{\ell_2^* \left(\frac{\gamma_1}{\gamma_2} - 1 \right) + 2 \frac{\gamma_1}{\gamma_2}} \frac{n_{h_1}}{n_{h_2}} \right] n_{h_2}$$

Therefore, for $\lambda_2^* = \lambda_1^*$ we have

$$[\ell_1^*, n_{h_1}, \gamma_1] \multimap [\ell_2^*, n_{h_2}, \gamma_2] \Rightarrow [\ell_e^*, n_{h_e}, \gamma_e]$$

where

$$\gamma_e = \gamma_1$$

$$\ell_e^* = \ell_1^*$$

$$n_{h_e} = \left[1 + \frac{2}{\ell_1^* \left(\frac{\gamma_2}{\gamma_1} - 1 \right) + 2 \frac{\gamma_2}{\gamma_1}} \frac{n_{h_2}}{n_{h_1}} \right] n_{h_1}$$

and the equivalent window size will be

$$w_e = \ell_e^* n_{h_e} = \left[1 + \frac{2}{\ell_1^* \left(\frac{\gamma_2}{\gamma_1} - 1 \right) + 2 \frac{\gamma_2}{\gamma_1}} \frac{n_{h_2}}{n_{h_1}} \right] w_1$$

We can also replace the indices 1 and 2 in the above equations to get an equivalent network, i.e.,

$$\gamma_e = \gamma_2$$

$$\ell_e^* = \ell_2^*$$

$$n_{h_e} = \left[1 + \frac{2}{\ell_2^* \left(\frac{\gamma_1}{\gamma_2} - 1 \right) + 2 \frac{\gamma_1}{\gamma_2}} \frac{n_{h_1}}{n_{h_2}} \right] n_{h_2}$$

and the equivalent window size will be

$$w_e = \ell_e^* n_{h_e} = \left[1 + \frac{2}{\ell_2^* \left(\frac{\gamma_1}{\gamma_2} - 1 \right) + 2 \frac{\gamma_1}{\gamma_2}} \frac{n_{h_1}}{n_{h_2}} \right] w_2$$

Case II: $\lambda_2^* > \lambda_1^*$

In this case network 1 is the bottleneck, hence we have

$$\lambda_e^* = \lambda_1^* = \frac{\ell_1^*}{\ell_1^* + 2} \gamma_1 \tag{6.8}$$

When network 2 carries a traffic equal to λ_1^* the load at its nodes becomes

$$\ell_2 = \frac{2 \ell_1^* \gamma_1}{\ell_1^* (\gamma_2 - \gamma_1) + 2\gamma_2} \tag{6.9}$$

and it is easy to demonstrate that $\lambda_2^* > \lambda_1^*$ is equivalent to $\ell_2^* > \ell_2$. For a load of ℓ_2 the delay at network 2 becomes

$$T_2 = \frac{\ell_2 + 2}{2\gamma_2} n_{h_2}$$

Using the value of ℓ_2 from Eq. (6.9), and after some algebra, we get

$$T_2 = \frac{\ell_1^* + 2}{2\gamma_1} \frac{2}{\ell_1^* \left(\frac{\gamma_2}{\gamma_1} - 1 \right) + 2 \frac{\gamma_2}{\gamma_1}} n_{h2}$$

The total system delay is therefore

$$T_e^* = T_1^* + T_2$$

$$= \frac{\ell_1^* + 2}{2\gamma_1} \left[1 + \frac{2}{\ell_1^* \left(\frac{\gamma_2}{\gamma_1} - 1 \right) + 2 \frac{\gamma_2}{\gamma_1}} \frac{n_{h_2}}{n_{h_1}} \right] n_{h_1} \qquad (6.10)$$

Comparing Eqs. (6.8) with (6.1), and (6.10) with (6.2), we have, if $\lambda_2^* > \lambda_1^*$, then

$$[\ell_1^*, n_{h_1}, \gamma_1] \multimap [\ell_2^*, n_{h_2}, \gamma_2] \Rightarrow [\ell_1^*, n_{h_e}, \gamma_1]$$

where

$$n_{h_e} = \left[1 + \frac{2}{\ell_1^* \left(\frac{\gamma_2}{\gamma_1} - 1 \right) + 2 \frac{\gamma_2}{\gamma_1}} \frac{n_{h_2}}{n_{h_1}} \right] n_{h_1}$$

and the equivalent window size will be

$$w_e = \left[1 + \frac{2}{\ell_1^* \left(\frac{\gamma_2}{\gamma_1} - 1 \right) + 2 \frac{\gamma_2}{\gamma_1}} \frac{n_{h_2}}{n_{h_1}} \right] w_1$$

Case III: $\lambda_2^* < \lambda_1^*$

In this case network 2 is the bottleneck for the throughput. The analysis is similar to Case II, only indices 1 and 2 should be interchanged.

In general, when several networks are connected in series, the bottleneck for the throughput is the network with the smallest traffic handling capacity, and we have

$$\sum_{i=1}^{N} (\multimap) \; [\ell_i^*, n_{h_i}, \gamma_i] \Rightarrow [\ell_e^*, n_{h_e}, \gamma_e]$$

Let i^* be such that

$$\lambda^*_{i^*} = \min_{1 \le i \le N} \{\lambda^*_i\}$$

then

$$\gamma_e = \gamma_{i^*}$$

$$\ell^*_e = \ell^*_{i^*}$$

$$n_{h_e} = \alpha_{i^*} n_{h_{i^*}}$$

where

$$\alpha_{i^*} = \sum_{i=1}^{N} \left[\frac{2}{\ell^*_{i^*} \left(\frac{\gamma_i}{\gamma_{i^*}} - 1 \right) + 2 \frac{\gamma_i}{\gamma_{i^*}}} \frac{n_{h_i}}{n_{h_{i^*}}} \right]$$

The equivalent window size, the throughput and the delay will be

$$w_e = \alpha_{i^*} w_{i^*}$$

$$\lambda^*_e = \lambda^*_{i^*}$$

$$T^*_e = \frac{\ell^*_{i^*} + 2}{2\gamma_{i^*}} \alpha_{i^*} n_{h_{i^*}}$$

If $\gamma_i = \gamma$ for all i, we get the same results as in Section 6.1.1.1.

6.1.2 Parallel Connection of Networks

As in the analysis of series connections, we assume that the operation of each of the parallel-connected networks is independent of the other. Parallel connection is similar to the case in which a user has a number of networks at his disposal to send his messages to the receiver. Note that we assume the networks do not share common windows (we will

consider this case in the following sections). Unless the maximum load factors of the networks are equal, the analysis becomes very complicated and no simple expression for the equivalent network can be derived; therefore, in what follows we will assume $\ell_i^* = \ell^*$ for all i.

Consider N networks connected in parallel. When each network carries its maximum traffic, the total traffic carried by the system becomes the sum of the traffic handling capacities of each network, hence we have

$$\lambda_e^* = \sum_{i=1}^{N} \lambda_i^*$$

$$= \sum_{i=1}^{N} \frac{\ell_i^*}{\ell_i^* + 2} \gamma_i$$

or

$$\lambda_e^* = \frac{\ell^*}{\ell^* + 2} \sum_i \gamma_i \qquad (6.11)$$

The system delay can be calculated by considering the fact that a fraction λ_i^*/λ_e^* of the total traffic λ_e^* that is sent through network i experiences a delay of T_i^*, hence the average system delay becomes

$$T_e^* = \sum_{i=1}^{N} \frac{\lambda_i^*}{\lambda_e^*} T_i^*$$

Using the values of λ_i^* and T_i^* from Eqs. (6.1) and (6.2), and considering the fact that $\ell_i^* = \ell^*$ for all i, we have

$$T_e^* = \frac{\ell^* + 2}{2 \sum_{i=1}^{N} \gamma_i} \sum_i n_{h_i} \qquad (6.12)$$

Eqs. (6.11) and (6.12) indicate that for the equivalent network the maximum channel transmission rate is the sum of the transmission rates

of the networks, and the equivalent number of hops is also the sum of the number of hops in each network.

Summing up, we have the following:

If $$\ell_i^* = \ell^* \qquad i = 1, 2, \ldots, N$$

Then $$\sum_{i=1}^{N} (-=-) \; [\ell^*, n_{h_i}, \gamma_i] \Rightarrow [\ell^*, \sum_i n_{h_i}, \sum_i \gamma_i]$$

The equivalent window size will be

$$w_e = \frac{\sum_i n_{h_i}}{n_{h_j}} w_j \qquad \text{for any } j$$

The traffic handling capacity and the delay of the equivalent network can be found by using Eqs. (6.11) and (6.12).

Table 6.1 shows a summary of our results for the series and/or parallel connection of networks when $w_i > 0$ and $\gamma_i > 0$. For the series connection, the network with the lowest maximum traffic handling capacity is the bottleneck of the system. In the special case where the channel transmission rates of all networks are identical, the number of hops of the equivalent network becomes the same as the sum of the number of hops of the networks. For the parallel connection we have shown only the results for the special case where the maximum load factors of all of the networks are identical; the results for the general case are too complex to be useful.

6.2 Optimal Allocation of Resources

In the previous section we were concerned with the analysis of systems composed of a number of networks connected to each other in

	SERIES —●—	PARALLEL —=—
SPECIAL CASE	$\gamma_i = \gamma_j = \gamma \Rightarrow \quad 1 \le i, j \le N$ $\sum_{i=1}^{N} (-\circ-) \; [\ell_i^*, n_{h_i}, \gamma] \Rightarrow [\ell_e^*, n_{h_e}, \gamma_e]$ Let i^* be such that $\lambda_{i^*}^* = \min_i \{\lambda_i^*\}$ $(= \ell_{i^*}^* = \min_i \{\ell_i^*\})$ $\gamma_e = \gamma$ $\ell_e^* = \ell_{i^*}^*$ $n_{h_e} = \sum n_{h_i}$ $w_e = \frac{\sum n_{h_i}}{n_{h_{i^*}}} w_{i^*}$ $\lambda_e^* = \lambda_{i^*}^*$ $T_e^* = \frac{\ell_{i^*}^* + 2}{2\gamma} \sum n_{h_i}$	$\ell_i^* = \ell_j^* = \ell^* \quad 1 \le i, j \le N$ $\sum_{i=1}^{N} (-=-) \; [\ell^*, n_{h_j}, \gamma_i] \Rightarrow [\ell_e^*, n_{h_e}, \gamma_e]$ $\gamma_e = \sum \gamma_i$ $\ell_e^* = \ell^*$ $n_{h_e} = \sum n_{h_i}$ $w_e = \frac{\sum n_{h_i}}{n_{h_j}} w_j$ for any j $\lambda_e^* = \frac{\ell^*}{\ell^* + 2} \sum \gamma_i$ $T_e^* = \frac{\ell^* + 2}{2 \sum \gamma_i} \sum n_{h_i}$
GENERAL CASE	$\sum (-\circ-) \; [\ell_i^*, n_{h_i}, \gamma_i] \Rightarrow [\ell_e^*, n_{h_e}, \gamma_e]$ Let i^* be such that $\lambda_{i^*}^* = \min_i \{\lambda_i^*\}$ and $\alpha_{i^*} \triangleq \sum_{i=1}^{N} \frac{2}{\ell_i^* \left(\frac{\gamma_i}{\gamma_{i^*}} - 1\right) + 2\frac{\gamma_i}{\gamma_{i^*}}} \frac{n_{h_i}}{n_{h_{i^*}}}$ $\gamma_e = \gamma_{i^*}$ $\ell_e^* = \ell_{i^*}^*$ $n_{h_e} = \alpha_{i^*} n_{h_{i^*}}$ $w_e = \alpha_{i^*} w_{i^*}$ $\lambda_e^* = \lambda_{i^*}^*$ $T_e^* = \frac{\ell_{i^*}^* + 2}{2\gamma_{i^*}} \alpha_{i^*} n_{h_{i^*}}$	

TABLE 6.1. A NETWORK ALGEBRA

$\gamma_i > 0, \quad w_i > 0$

series and/or parallel, and we characterized equivalent networks to such interconnected nets. In this section we look at some design problems for parallel networks. Throughout our study of flow control, we have focused on a communication path of a network, and by using the number of hops and capacity of channels along a path, we could find its traffic handling capacity as a function of the window size (our flow control variable). Our distinction between a network and a path has been very loose; for example, the analysis of parallel networks (Section 6.1.2) can be applied to two separate networks connected in parallel, or to two disjoint paths (between a source and destination) with independent window controls in the same network. The design problem which we are going to study is particularly useful when we take the latter point of view (i.e., two paths of the same network). It is, however, important to realize that our study is based on the assumption that the paths are stochastically independent of each other.

The design problems which we address in this section are concerned with the optimal allocation of (limited) resources among a number of parallel communication paths. The resources we consider are window buffers and transmission capacity. The optimization can be done with respect to throughput (maximize the throughput), the delay (minimize the delay), or some throughput-delay combination. In some special cases the optimal allocation with respect to throughput also results in the minimal delay; however, in most cases, the maximum throughput does not occur at minimum delay. For example, the resources can be allocated so that the delay becomes zero; however, this allocation also results in zero throughput. When the optimization is done with respect to

throughput-delay, a proper objective function should be defined to incorporate the opposing effects of throughput and delay (in Chapter 5 we used such a function in our optimal decision problem). In the discussion below we first carry out the optimization with respect to maximal throughput. Later we consider the case when delay is also a measure of performance. We start with the general case of allocating both window buffers and channel capacity, and then we consider the special cases of allocating either window buffers or channel capacity.

6.2.1 Optimal Allocation of Window Buffers and Channel Capacities

Consider Fig. 6.1-c in which two paths of lengths n_{h_1} and n_{h_2} (number of hops) can be used to send messages from source S to destination D. Assume that we have a pool of w buffers to use in the traffic controllers of the two paths, and a total channel capacity of C (or a transmission rate of $\gamma = 1/\mu C$, $1/\mu$ being the average message length) to allocate to the two paths. The question is: "what is the optimal allocation of these resources to the paths, in order to optimize the total traffic handling capacity between source S and destination D?" The problem can be formally stated as follows:

Given: path lengths n_{h_1} and n_{h_2}

Optimize: the traffic handling capacity

With respect to: window sizes (w_1 and w_2) and channel capacities (C_1 and C_2)

Such that:

$$w_1 + w_2 = w \tag{6.13}$$

and

$$C_1 + C_2 = C \quad \text{(or equivalently } \gamma_1 + \gamma_2 = \gamma\text{)} \tag{6.14}$$

For a given choice of w_1, w_2, γ_1, and γ_2 (such that Eqs. (6.13) and (6.14) are satisfied) we can use Eq. (4.5) to find the traffic handling of each path. The maximum total throughput of the two paths when they operate in parallel (as discussed in Section 6.1.2) is the sum of the traffic handling capacities of the paths; therefore, the optimization problem can be stated equivalently as

Maximize:

$$\lambda^* = \frac{w_1}{w_1 + 2n_{h_1}} \gamma_1 + \frac{w_2}{w_2 + 2n_{h_2}} \gamma_2 \tag{6.15}$$

With respect to: w_1, w_2, γ_1 and γ_2

Such that: (6.13) and (6.14) are satisfied

We can use any conventional method of nonlinear optimization [LUEN 73] to solve the above problem. The simplest approach is to use Eqs. (6.13) and (6.14) to find w_2 and γ_2 in terms of w_1 and γ_1, and use these values in Eq. (6.15) to get

$$\lambda^* = \frac{w_1}{w_1 + 2n_{h_1}} \gamma_1 + \frac{w - w_1}{(w - w_1) + 2n_{h_2}} (\gamma - \gamma_1) \tag{6.16}$$

We now can differentiate λ^* with respect to w_1 and γ_2 and set the derivatives equal to zero.

$$\frac{\partial \lambda^*}{\partial w_1} = 0$$

$$\frac{\partial \lambda^*}{\partial \gamma_1} = 0$$

The formal solution to this set of nonlinear equations is

$$w_1 = \frac{n_{h_1}}{n_{h_1} + n_{h_2}} w \qquad (6.17\text{-a})$$

and

$$\gamma_1 = \frac{n_{h_1}}{n_{h_1} + n_{h_2}} \gamma \qquad (6.18\text{-a})$$

and from constraints (6.13) and (6.14) we get

$$w_2 = \frac{n_{h_2}}{n_{h_1} + n_{h_2}} w \qquad (6.17\text{-b})$$

and

$$\gamma_2 = \frac{n_{h_2}}{n_{h_1} + n_{h_2}} \gamma \qquad (6.18\text{-b})$$

The above values for w_i's and γ_i's are not, however, the optimal solution! They correspond to the inflection point of the λ^* surface specified by Eq. (6.16). In fact, the optimal solution is located at a boundary point of the λ^* surface, which is obtained by allocating the network resources to only <u>one</u> of the paths (note that the variables w_1 and γ_1 are bounded below by zero, and bounded above by w and γ, respectively). This is shown in the following theorem.

<u>Theorem 6.1</u>

The maximum traffic handling capacity of two paths in parallel is obtained by allocating all resources (window buffers and channel capacity) to the path with shorter length.

Proof

Consider two paths in parallel with lengths n_{h_1} and n_{h_2}, where

$$n_{h_1} \leq n_{h_2}$$

If we allocate all the resources to path 1, then the traffic handling capacity becomes

$$\frac{w}{w + 2n_{h_1}} \gamma$$

In order to establish the proof we must show that

$$\frac{w}{w + 2n_{h_1}} \gamma \geq \frac{w_1}{w_1 + 2n_{h_1}} \gamma_1 + \frac{w - w_1}{w - w_1 + 2n_{h_2}} (\gamma - \gamma_1) \qquad (6.19)$$

for all $0 \leq w_1 \leq w$ and $0 \leq \gamma_1 \leq \gamma$

We can rewrite the inequality (6.19) as

$$\left[\frac{w}{w + 2n_{h_1}} - \frac{w - w_1}{w - w_1 + 2n_{h_2}}\right] \gamma \geq \left[\frac{w_1}{w_1 + 2n_{h_1}} - \frac{w - w_1}{w - w_1 + 2n_{h_2}}\right] \gamma_1 \qquad (6.20)$$

Because $n_{h_1} \leq n_{h_2}$ we have

$$\frac{w}{w + 2n_{h_1}} - \frac{w - w_1}{w - w_1 + 2n_{h_2}} \geq 0 \quad \text{for all} \quad 0 \leq w_1 \leq w \qquad (6.21)$$

The condition $0 \leq w_1 \leq w$ implies that

$$\frac{w}{w + 2n_{h_1}} \geq \frac{w_1}{w_1 + 2n_{h_1}}$$

Hence the content of the brackets on the left-hand side of Eq. (6.20) is non-negative and larger than the content of the right-hand side bracket.

The fact that $\gamma_1 \leq \gamma$ establishes the proof.

Q.E.D.

The traffic handling capacity of two paths in parallel for different resource allocations are shown in Figs. 6.2-a and 6.2-b. In these figures the quantities are normalized as follows: The path lengths (n_{h_1} and n_{h_2}) and the window sizes (w_1 and w_2) are normalized by w, the total number of window buffers; the channel transmission rates (γ_1 and γ_2) and the throughput (λ^*) are normalized by γ, the total transmission rate. These normalizations simplify the presentation.

The throughput as a function of window allocation is shown in Fig. 6.2-a (note that $w_2/w = 1 - w_1/w$) where we have plotted the curves for several constant γ_1/γ. For (normalized) path lengths of $n_{h_1}/w = 0.1$ and $n_{h_2}/w = 0.4$, Eqs. (6.17) and (6.18) give $w_1/w = 0.2$, $w_2/w = 0.8$, $\gamma_1/\gamma = 0.2$ and $\gamma_2/\gamma = 0.8$. $w_1/w = 0.2$ in Fig. 6-2.a is the point where all of the constant γ_1/γ curves meet. Note that at this point the throughput is maximum only for the curve $\gamma_1/\gamma = 0.2$. Fig. 6.2-a shows that the optimal allocation (point A) is where all resources are dedicated to path 1, which has a shorter length.

Fig. 6.2-b shows the (normalized) traffic handling capacity of these paths as a function of the (normalized) channel transmission rate of path 1 for different values of w_1/w (the normalized window buffer allocated to path 1). This figure shows also that the maximum throughput is achieved at point B by allocating all the resources to the shorter path. Note that when window buffers are allocated according to Eq. (6.17), then the throughput becomes insensitive to the capacity

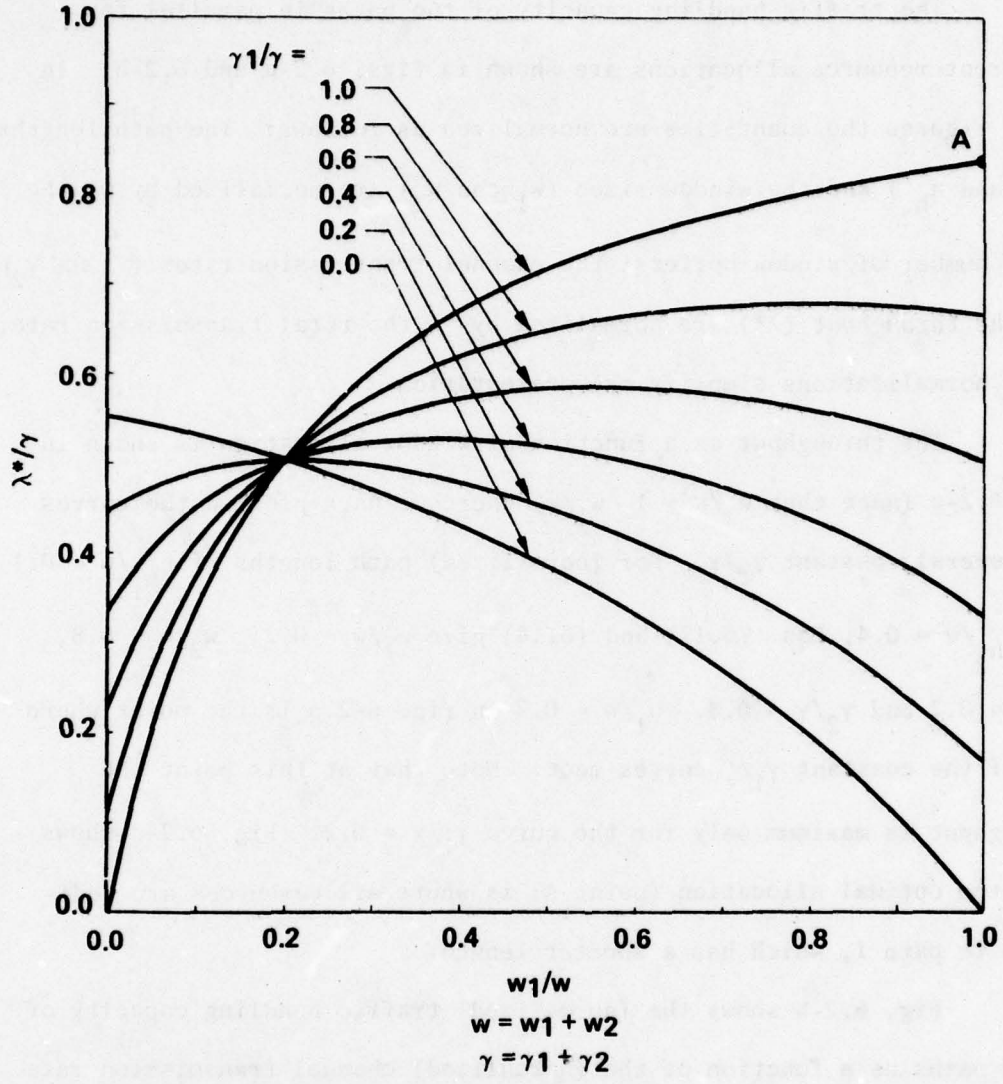


Figure 6.2-a. Traffic Handling Capacity vs Window Buffer Allocation ($n_{h_1}/w = 0.1$, $n_{h_2}/w = 0.4$).

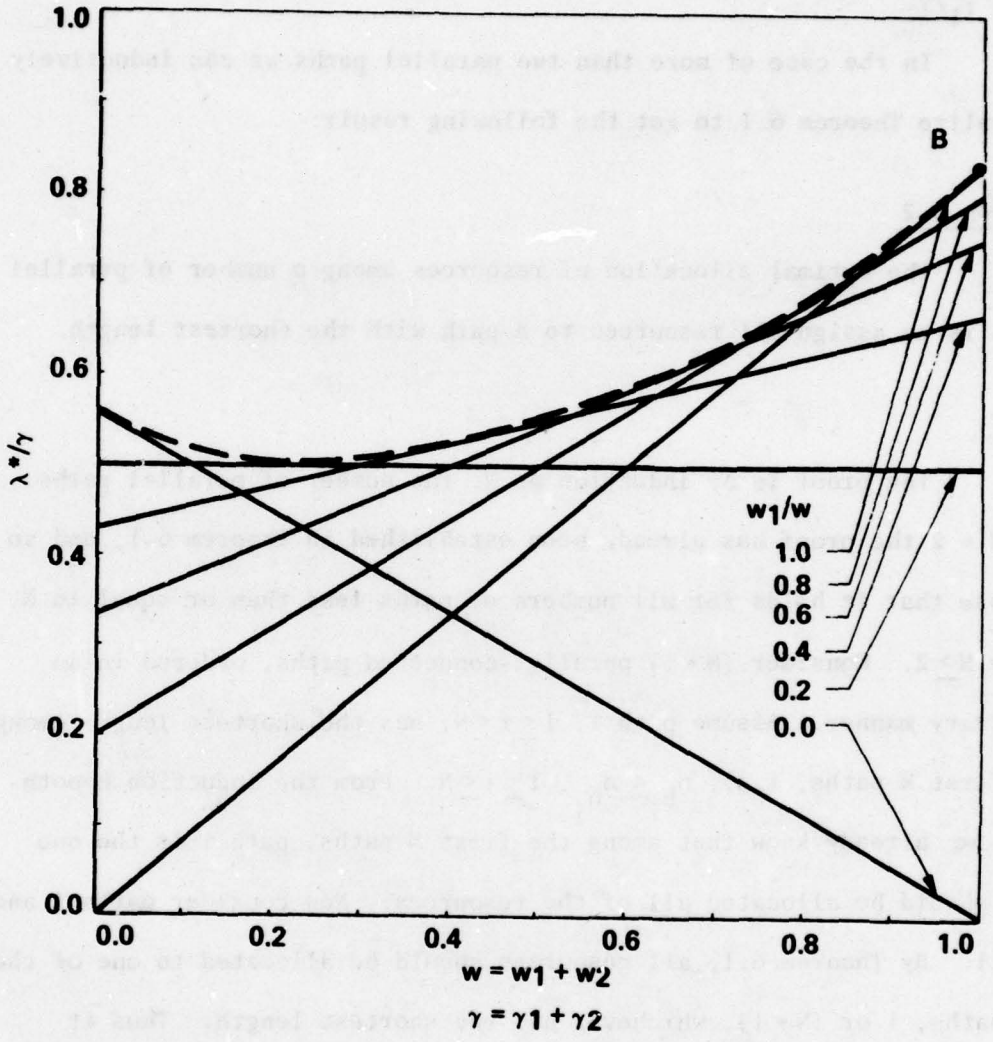


Figure 6.2-b. Traffic Handling Capacity vs Capacity Allocation ($n_{h_1}/w = 0.1$, $n_{h_2}/w = 0.4$).

allocation (the line of $w_1/w = 0.2$ in Fig. 6.2-b). All throughput lines for fixed-window buffer allocations are tangent to a common envelope which specifies the upper bound on the throughput achievable for a fixed γ_1/γ.

In the case of more than two parallel paths we can inductively generalize Theorem 6.1 to get the following result:

Theorem 6.2

The optimal allocation of resources among a number of parallel paths is to assign all resources to a path with the shortest length.

Proof:

The proof is by induction on N, the number of parallel paths. For N = 2 the proof has already been established in Theorem 6.1, and so suppose that it holds for all numbers of paths less than or equal to N, where $N \geq 2$. Consider (N + 1) parallel-connected paths, ordered in an arbitrary manner. Assume path i, $1 \leq i \leq N$, has the shortest length among the first N paths, i.e., $n_{h_i} \leq n_{h_j}$, $1 \leq j \leq N$. From the induction hypothesis we already know that among the first N paths, path i is the one that should be allocated all of the resources. Now consider paths i and (N + 1). By Theorem 6.1, all resources should be allocated to one of the two paths, i or (N + 1), whichever has the shortest length. Thus it follows that for any number of parallel-connected paths, all resources should be allocated to the one with the shortest length.

Q.E.D.

6.2.2 Optimal Window Buffer Allocation

If the channel capacities of the number of paths operating in parallel are known, then the problem is how to allocate a pool of window buffers to these paths in order to maximize the traffic handling capacity. For any window allocation w_1 and w_2, the throughput is given by Eq. (6.15). Because the condition (6.13) should also be satisfied, the optimization problem reduces to

Maximize:

$$\lambda^* = \frac{w_1}{w_1 + 2n_{h_1}} \gamma_1 + \frac{(w - w_1)}{(w - w_1) + 2n_{h_2}} \gamma_2$$

With respect to: w_1

Such that: $0 \leq w_1 \leq w$

This optimization problem can be easily solved and results in the following value for w_1

$$\frac{w_1}{w} = \left[1 + 2 \frac{n_{h_1} + n_{h_2}}{w} \right] \frac{\sqrt{\gamma_1 n_{h_1}}}{\sqrt{\gamma_1 n_{h_1}} + \sqrt{\gamma_2 n_{h_2}}} - \frac{2n_{h_1}}{w} \qquad (6.22)$$

If the value of w_1 found above is not within the limit $0 \leq w_1 \leq w$, then it should be set to 0 or w. It is easy to show that in order for $0 \leq w_i \leq w$ $(i = 1,2)$, we must have

$$4 \frac{n_{h_i} n_{h_j}}{(w + 2n_{h_j})^2} \gamma_j \leq \gamma_i \leq \frac{(w + 2n_{h_j})^2}{4n_{h_i} n_{h_j}} \qquad i,j = 1,2 \ldots \quad i \neq j$$

Note that the optimal allocation depends on the channel capacity and the path length, and in general the path with the larger channel

capacity and/or the smaller length is allocated a larger portion of the window size. Figs. 6.2-a and 6.2-b show several examples of optimal allocations. In Fig. 6.2-a, where we have plotted the total throughput as a function of the window allocated to path 1 for different channel capacities, the maximum point of each curve corresponds to the optimal window allocation. Note that when $\gamma_1/(\gamma_1 + \gamma_2)$ is zero or one, the optimal allocation is located at a boundary point. When

$$\frac{\gamma_i}{\gamma_1 + \gamma_2} = \frac{n_{h_i}}{n_{h_1} + n_{h_2}} \qquad i = 1,2$$

then the optimal window allocation becomes

$$w_i = \frac{n_{h_i}}{n_{h_1} + n_{h_2}} w \qquad i = 1,2$$

This means that when the channel capacities are proportional to path lengths, then the best allocation is also the one which assigns the window buffers proportional to their path lengths. In Fig. 6.2-a, the curve for $\gamma_1/(\gamma_1 + \gamma_2) = 0.2$ is such a curve, and the best window allocation is $w_1/w = 0.2$. In Fig. 6.2-b we can use the optimal envelope to find the best window allocation. For a given $\gamma_1/(\gamma_1 + \gamma_2)$, the line of constant w_1/w which is tangent to the envelope at the point with the given abscissa corresponds to the optimal allocation.

6.2.3 Optimal Capacity Allocation

When the window sizes of the paths are known, we can take the same approach as we did in the previous sections to find the optimal allocation of a fixed capacity among a number of parallel paths. The

optimization problem becomes

Maximize:

$$\lambda^* = \frac{w_1}{w_1 + n_{h_1}} \gamma_1 + \frac{w_2}{w_2 + 2n_{h_2}} \gamma_2 \tag{6.23}$$

With respect to: γ_1 and γ_2

Such that: $\gamma_1 + \gamma_2 = \gamma$

The optimization in this case is fairly simple, as demonstrated by the following theorem:

Theorem 6.3

If the window sizes of two paths in parallel are known, then the optimal allocation of channel capacity to the two paths is to assign all of the capacity to the path with the larger maximum load factor ($\ell_i^* = w_i/n_{h_i}$). If the maximum load factors of the two paths are identical, then the maximal throughput is not sensitive to the capacity of each individual channel.

Proof:

We can find γ_2 in terms of γ_1, and using Eq. (6.23), we get

$$\lambda^* = \left(\frac{w_1}{w_1 + 2n_{h_1}} - \frac{w_2}{w_2 + 2n_{h_2}} \right) \gamma_1 + \frac{w_2}{w_2 + 2n_{h_2}} \gamma \qquad 0 \leq \gamma_1 \leq \gamma \tag{6.24}$$

If the coefficient of γ_1 in Eq. (6.24) is positive, then λ^* becomes maximum when $\gamma_1 = \gamma$; if it is negative then $\gamma_1 = 0$ results in the optimal λ^*; and finally, if the coefficient is zero, then the throughput is independent of capacity allocation. In order for $\gamma_1 = \gamma$

to be optimal, the following condition should be satisfied:

$$\frac{w_1}{w_1 + 2n_{h_1}} - \frac{w_2}{w_2 + 2n_{h_2}} > 0$$

The above inequality can be reduced to

$$\ell_1^* = \frac{w_1}{n_{h_1}} > \frac{w_2}{n_{h_2}} = \ell_2^*$$

Similarly, for $\gamma_1 = 0$ to be optimal we should have

$$\ell_1^* = \frac{w_1}{n_{h_1}} < \frac{w_2}{n_{h_2}} = \ell_2^*$$

and if

$$\frac{w_1}{n_{h_1}} = \frac{w_2}{n_{h_2}}$$

then any allocation results in the same throughput of

$$\lambda^* = \frac{w_1}{w_1 + 2n_{h_1}} \gamma = \frac{w_2}{w_2 + 2n_{h_2}} \gamma$$

Q.E.D.

We use Figs. 6.2-a and 6.2-b to show some examples of optimal capacity allocation. In these figures the number of hops of the two parallel paths are such that $n_{h_1}/(w_1 + w_2) = 0.1$ and $n_{h_2}/(w_1 + w_2) = 0.4$. For these values of n_{h_1} and n_{h_2}, in order for the maximum load factors to be identical ($w/n_{h_1} = w/n_{h_2}$) we require

$$\frac{w_1}{w_1 + w_2} = 0.2 \quad \text{and} \quad \frac{w_2}{w_1 + w_2} = 0.8$$

If $w_1/(w_1 + w_2) < 0.2$, then the best performance (maximum throughput) is obtained when $\gamma_1 = 0$ (or $\gamma_2 = \gamma$). On the other hand, if w_1 and w_2 are such that $w/(w_1 + w_2) > 0.2$ (or $\ell_1^* > \ell_2^*$), then $\gamma_1 = \gamma$ gives the best performance. Note that if $w_1/(w_1 + w_2) = 0.2$ (or $\ell_1^* = \ell_2^*$) then all of the curves for different values of $\gamma_1/(\gamma_1 + \gamma_2)$ result in the same throughput.

In Fig. 6.2-b the total traffic handling capacity for different window sizes has been shown as a function of (relative) channel capacity of path 1. Note that if $\ell_1^* < \ell_2^*$, then the slope of the throughput line is negative and the maximum throughput is achieved when $\gamma_1/\gamma = 0$; for $\ell_1^* = \ell_2^*$ the throughput line becomes parallel to the γ_1/γ axis and the throughput is insensitive to capacity allocation; if $\ell_1^* > \ell_2^*$, the slope is positive and $\gamma_1 = \gamma$ results in the best performance.

Theorem 6.3 can be generalized to the case of more than two paths in parallel.

Theorem 6.4

If the window sizes of $N (\geq 2)$ paths in parallel are known, then the optimal allocation (with respect to throughput) of a total channel capacity to the paths is to assign all of the capacity to the path with the largest maximum load factor ($\ell_i^* = w_i/n_{h_i}$).

Proof:

We prove this theorem by contradiction. Assume there is a capacity allocation, claimed to be optimal, which does not agree with the theorem. We show there is at least one allocation which gives a better performance. Consider paths i and j where

$$\ell_i^* = \frac{w_i}{n_{h_i}} > \frac{w_j}{n_{h_j}} = \ell_j^*$$

and the allocated capacities of these paths are such that

$$C_i \neq 0 \quad \text{and} \quad C_j \neq 0 \qquad (\text{or } \gamma_i \neq 0 \quad \text{and} \quad \gamma_j \neq 0).$$

The contribution of these two paths to the total throughput is

$$\frac{w_i}{w_i + 2n_{h_i}} \gamma_i + \frac{w_j}{w_j + 2n_{h_j}} \gamma_j$$

By Theorem 6.3 we know that if we assign the capacity $C_i + C_j$ to path i, then the throughput will be better than given above, i.e.,

$$\frac{w_i}{w_i + n_{h_i}} (\gamma_i + \gamma_j) > \frac{w_i}{w_i + 2n_{h_i}} \gamma_i + \frac{w_j}{w_j + 2n_{h_j}} \gamma_j$$

Therefore, there is an allocation different from the given one, which results in a higher traffic handling capacity. This contradicts the claim that the given allocation is optimal.

Q.E.D.

6.2.4 Delay Considerations

Before discussing the optimal resource allocation when delay is also a measure of performance, we find it useful to reflect on the characteristics of the (maximum) delay T*.

The maximum delay of a path (which occurs when the throughput is maximal, λ^*) when the window size is w and its channel transmission rate is γ was found in Section 4.3 to be

$$T^* = \begin{cases} \dfrac{w + 2n_h}{2\gamma} & w > 0 \quad \text{and} \quad \gamma > 0 \\ 0 & w = 0 \quad \text{and} \quad \gamma = 0 \end{cases} \tag{4.7}$$

The discontinuity of T^* when either w or γ is zero causes some problems which are more severe (from the analysis point of view) when the channel capacity (or γ) is decreased to zero. In a queueing system (e.g., an M/M/1 system), when the input rate is zero, although there is no traffic to the system, the average system time is equal to the average service time. The physical interpretation of this result is that, when there is no traffic, if a "customer" arrives in the system, its delay is the average service time. Our point of view in analysis of networks is different from above, in the sense that we consider the delay of messages that actually enter the system. When w or γ is zero, because there is no traffic flowing through the system, we define the delay to be zero. In Figs. 6.3-a and -b we sketch the maximal delay and throughput as a function of channel transmission rate. When γ decreases to zero, in order to get a throughput of λ^* , we incur a large delay. For a very small (but non zero) capacity, the delay is unbounded; however, when γ reaches zero, we define the delay to be zero. The same discontinuity is observed when the window size is decreased to zero but γ is held fixed. Figs. 6.3-c and -d show the dependency of T^* and λ^* on w. Note that the discontinuity of T^* at the origin due to zero window size is not as large as the discontinuity in Fig. 6.3-a ($\gamma \downarrow 0$).

The effect of the discontinuity of T^* for two paths in parallel can be seen in Fig. 6.4-a. In this figure (which is the counterpart of Fig. 6.2-a) the (normalized) total delay as a function of window size

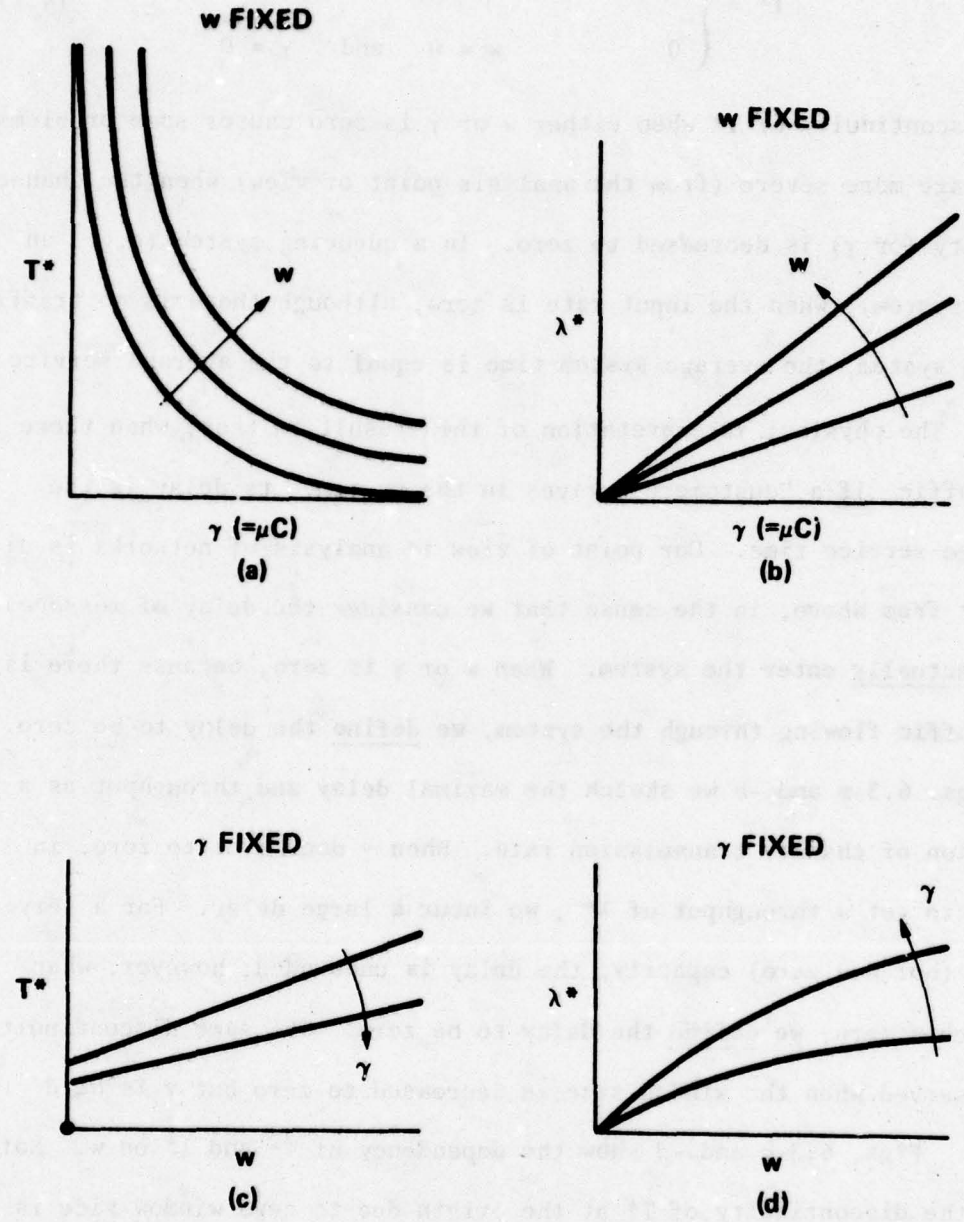
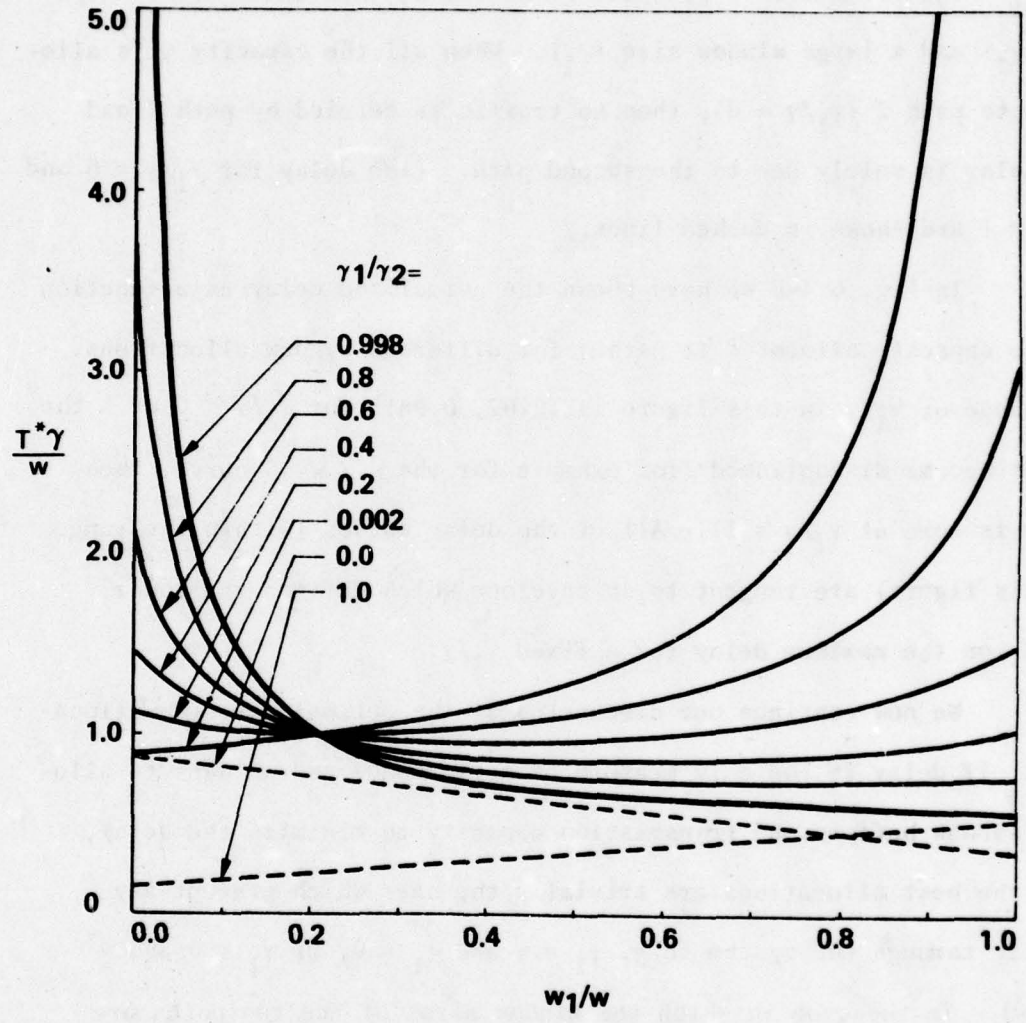


Figure 6.3. Diagram of Maximum Throughput and Delay.



$w = w_1 + w_2$

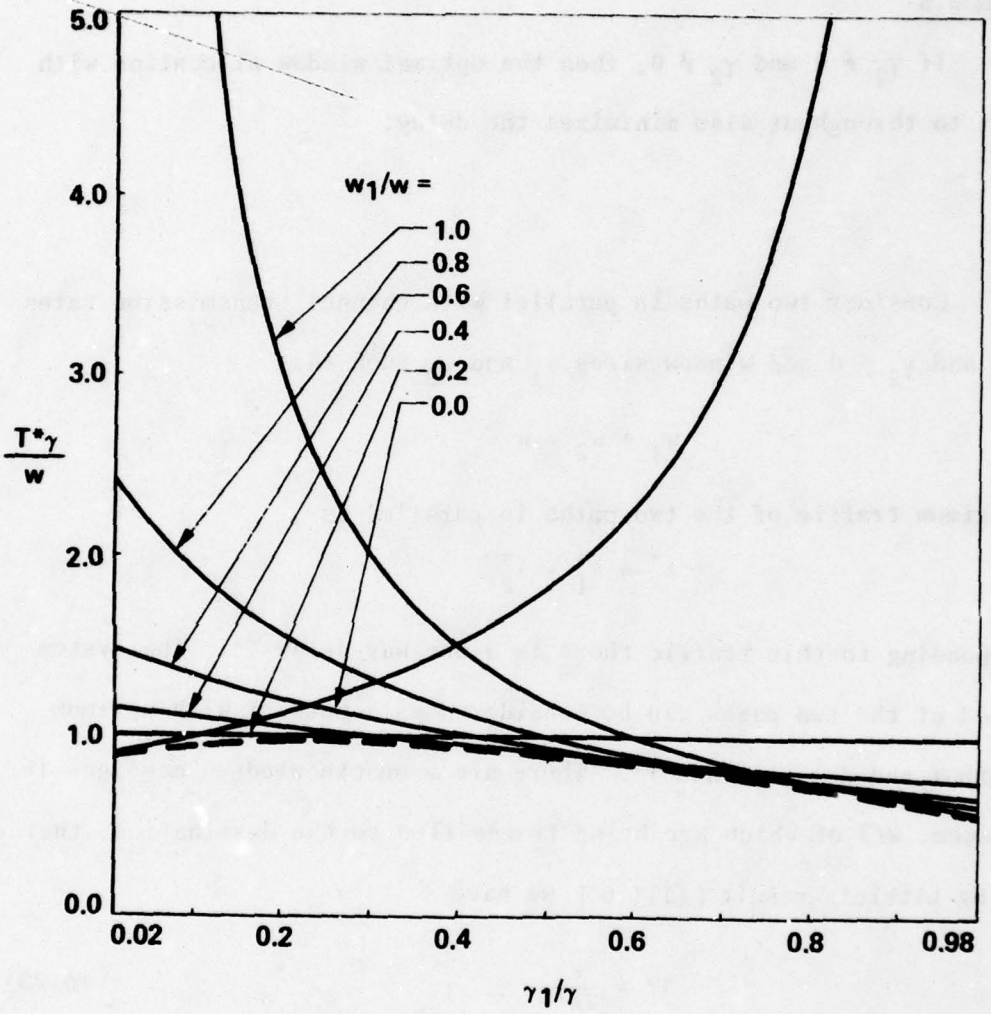
$\gamma = \gamma_1 + \gamma_2$

Figure 6.4-a. Traffic Weighted Delay vs Window Buffer Allocation ($n_{h_1}/w = 0.1$, $n_{h_2}/w = 0.4$).

of path 1 is shown for different capacity allocations. Similar to Fig. 6.2-a the path lengths, window size and channel capacity are normalized. Note how the curves for small $\gamma_1/(\gamma_1 + \gamma_2)$ become unbounded as $w_1/(w_1 + w_2)$ increases to 1, which is the result of small channel capacity (γ_1) and a large window size (w_1). When all the capacity γ is allocated to path 2 ($\gamma_1/\gamma = 0$), then no traffic is carried by path 1 and the delay is solely due to the second path. (The delay for $\gamma_1/\gamma = 0$ and $\gamma_1/\gamma = 1$ are shown in dashed lines.)

In Fig. 6.4-b we have shown the normalized delay as a function of the capacity allocated to path 1 for different window allocations. The range of γ_1/γ in this figure is (0.02, 0.98); for $\gamma_1/\gamma = 0$ or 1 the curves become discontinued (for example for the $w_1/w = 0$ curve, the delay is zero at $\gamma_1/\gamma = 1$). All of the delay curves (within the range of this figure) are tangent to an envelope which defines the lower bound on the maximum delay for a fixed γ_1/γ.

We now continue our discussion of the optimal resource allocation. If delay is the only measure of performance and we want to allocate window buffers and transmission capacity to minimize the delay, then the best allocations are trivially the ones which prevent any traffic through the system (e.g. $\gamma_1 = \gamma$ and $w_1 = 0$, or $\gamma_1 = 0$ and $w_1 = w$). In the case in which the window sizes of the two paths are known, we must allocate all of the capacity to only one of the paths in order to minimize the delay. This fact is shown in Fig. 6.4-a; depending on the value of $w_1/(w_1 + w_2)$ the capacity should be allocated to path 1 or 2. The more interesting case is when the channel capacities of the two paths are known and we want to allocate a total of w windows to the



$w = w_1 + w_2$

$\gamma = \gamma_1 + \gamma_2$

Figure 6.4-b. Traffic Weighted Delay vs Capacity Allocation ($n_{h_1}/w = 0.1$, $n_{h_2}/w = 0.4$).

two paths such that the delay becomes minimal. When $\gamma_1 \neq 0$ and $\gamma_2 \neq 0$, then we have

Theorem 6.5

If $\gamma_1 \neq 0$ and $\gamma_2 \neq 0$, then the optimal window allocation with respect to throughput also minimizes the delay.

Proof

Consider two paths in parallel with channel transmission rates $\gamma_1 \neq 0$ and $\gamma_2 \neq 0$ and window sizes w_1 and w_2 such that

$$w_1 + w_2 = w$$

The maximum traffic of the two paths in parallel is

$$\lambda^* = \lambda_1^* + \lambda_2^*$$

Corresponding to this traffic there is a one-way delay T*. The system composed of the two paths can be considered as a network with maximum throughput and delay λ^* and T*. There are w unacknowledged messages in the system, w/2 of which are being transmitted to the destination, therefore, by Little's result [LITT 61] we have

$$T^* = \frac{w}{2\lambda^*} \tag{6.25}$$

This equation shows that the particular choice of w_i's that maximizes λ also minimizes the delay T*.

Q.E.D.

By virtue of this theorem we can use the results of the optimal window allocation with respect to throughput (Section 6.2.2) for the minimal delay as well; in particular Eq. (6.22) gives the

window allocation which optimizes both the traffic handling capacity and the delay. In Fig. 6.4-a the optimal window allocation for a fixed γ_1 and γ_2 is when the delay curve is minimal (note that this does not apply to the case when either γ_1 or γ_2 is zero). Similar to Fig. 6.2-b, in Fig. 6.4-b the delay curve for the window allocation which is tangent to the lower bound envelope at the point with abscissa $\gamma_1/(\gamma_1 + \gamma_2)$ defines the optimal window allocation.

So far our optimality criteria have been either the maximal throughput or the minimal delay, and except for the special case that we studied in Section 6.2.2 (optimal window allocation for given channel capacities), the solution to the optimization problem was different for each criteria. In order to combine these two performance measures in a single optimization problem, we need to define a proper objective function which incorporates these two criteria. Such functions have already been used in this dissertation. In Chapter 4 we defined the power ($\mathscr{P}$) as the ratio of throughput and delay [GIES 76], and later in Chapter 5 we used a linear combination of throughput and delay ($\alpha\lambda - T$; $\alpha \geq 0$). Because both high throughput and low delay characterize the efficiency of a network, both of these objective functions are good candidates for the objective function.

In the remainder of this section we study the optimal resource allocation in parallel paths where the optimality criterion is both throughput and delay, and we choose to use power as our objective function.

Consider two parallel paths of lengths n_{h_1} and n_{h_2} between a

source and destination. For a window and capacity allocation such that constraints (6.13) and (6.14) are satisfied, the power of the system, when the throughput is equal to the traffic handling capacity, is

$$\mathcal{P}^* = \frac{\lambda^*}{T^*} \tag{6.26}$$

where

$$\lambda^* = \lambda_1^* + \lambda_2^*$$

and

$$T^* = \frac{\lambda_1^*}{\lambda^*} T_1^* + \frac{\lambda_2^*}{\lambda^*} T_2^*$$

The optimization problem becomes

Maximize: $\mathcal{P}^*$

With respect to: w_1, w_2, γ_1 and γ_2

Such that:

$$w_1 + w_2 = w$$

and

$$\gamma_1 + \gamma_2 = \gamma$$

This nonlinear optimization problem can be solved by an appropriate mathematical technique; however, instead of entangling ourselves in the mathematical formulation, we present the optimal allocation through a graphical presentation. In Fig. 6.5 we have shown the power of the two parallel paths (of lengths $n_{h_1}/w = .1$ and $n_{h_2}/w = .4$, w being the total number of window buffers to be allocated) as a function of the window allocated to path 1 for different capacity allocations.

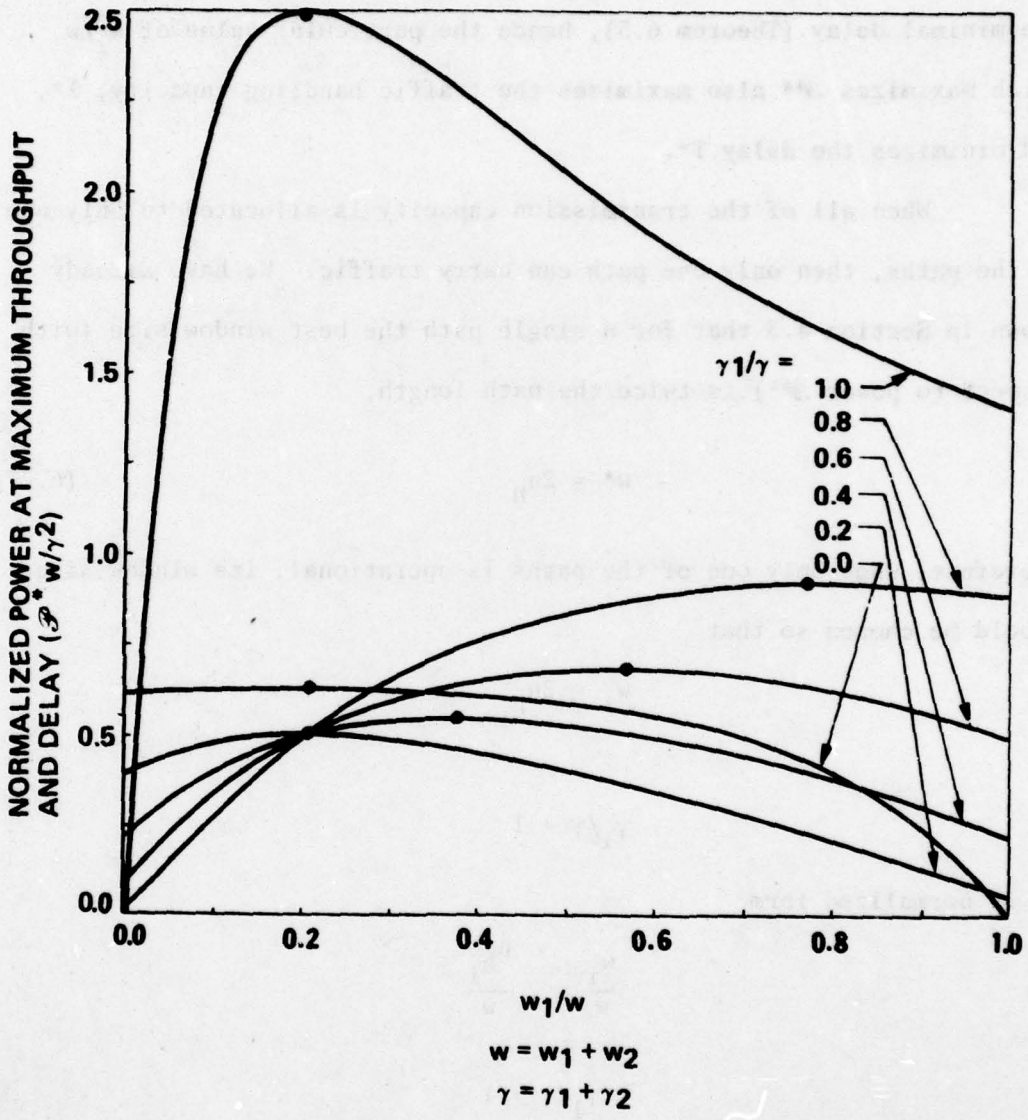


Figure 6.5. Power of Two Parallel Paths.

For $0 < \gamma_1/\gamma < 1$ (or $0 < \gamma_2/\gamma < 1$) there is a value of w_1/w which optimizes the corresponding $\mathscr{P}^*$ curve. We have already established the fact that for these values of γ_i the maximal throughput corresponds to the minimal delay (Theorem 6.5), hence the particular value of w_i/w which maximizes $\mathscr{P}^*$ also maximizes the traffic handling capacity, λ^*, and minimizes the delay T^*.

When all of the transmission capacity is allocated to only one of the paths, then only one path can carry traffic. We have already shown in Section 4.3 that for a single path the best window size (with respect to power $\mathscr{P}^*$) is twice the path length,

$$w^* = 2n_h \tag{6.27}$$

Therefore, when only one of the paths is operational, its window size should be chosen so that

$$w_i = 2n_{h_i}$$

if

$$\gamma_i/\gamma = 1$$

or in normalized form

$$\frac{w_i}{w} = 2\,\frac{n_{h_i}}{w}$$

if

$$\gamma_i/\gamma = 1$$

In Fig. 6.5 the maximum of the $\gamma_1/\gamma = 1$ curve occurs at

$$\frac{w_1}{w} = 2\,\frac{n_{h_1}}{w} = 0.2$$

and for

$$\gamma_1/\gamma = 0$$

the maximum occurs at

$$\frac{w_2}{w} = \frac{n_{h_2}}{w} = 0.8$$

or

$$\frac{w_1}{w} = 1 - \frac{w_2}{w} = 0.2$$

Therefore, when power is the optimality criterion:

i. For an optimal window and capacity allocation all the capacity should be allocated to one of the paths (the shorter) and the window size should be chosen according to Eq. (6.27).

ii. For an optimal capacity allocation (the window sizes are known) all the capacity should be allocated to one of the paths. The decision as to which path should the capacity be allocated is based upon the relative values of w_i's and n_{h_i}'s; see Fig. 6.5.

iii. For an optimal window allocation (the capacities are known) the window buffers should be allocated according to Eq. (6.22). This window allocation results also in maximum traffic handling capacity and minimum delay.

6.3 Buffer Allocation Schemes in a Destination Node

Throughout our study of flow control we have concentrated on a single source-destination data traffic, and have used the window mechanism as the main tool for throttling the traffic between a source and destination. In Section 4.4 and Chapter 5 we demonstrated how the

loss due to the finiteness of the destination buffer node affects, and in a way controls, the traffic. In this section we reflect on the effect of the interference among a number of data streams that merge into a single node, and study how a further control can be imposed on a data traffic by restricting its usage of the available destination buffers.

To make our study more general, we consider the operation of a destination node in isolation from the network. We consider a node in which streams of messages from one or more input channels are buffered and multiplexed into a single outgoing channel. The channels may belong to one network, or they can be considered to be data streams from different networks which merge together. As a result of the finite storage size, some messages may be lost and may need to be retransmitted from their source nodes. In Chapter 4 we found that retransmission of messages affects the useful traffic; therefore, the input rate of messages on the channels is affected by this loss. To simplify the analysis, we assume that the traffic rates on the input channels are not affected by this loss, i.e., the input processes are stochastically independent of the node.

This type of system, generally known as a multiplexing node, has been analyzed by others in one form or another, but almost all of the previous studies were based on the assumption that as far as allocation of buffer storage is concerned, there is no discrimination among messages from different input channels (which we will call different classes of messages). In other words, a pool of buffers is completely shared among different classes of messages. In this section

we compare some other storage sharing and/or allocation schemes.

In the discussion below, we assume that messages are transmitted out (served) according to their order of arrival and that there is no priority involved.

The simplest scheme is Complete Sharing, (CS), in which an arriving message is accepted if any buffer storage is available, and all of the classes of messages are treated uniformly (Fig. 6.6-a). The other extreme of buffer allocation is to partition the storage into fixed blocks of buffers and assign each block to a class, which we will refer to as Complete Partitioning, CP. In this scheme a message is rejected if the storage allocated to its class is full even if the entire storage is not completely used (Fig. 6.6-b). Intuitively, as long as the throughput or the probability of blocking is of concern, CS results in a better performance. This is because as long as there is a buffer available it can be used by any incoming message, regardless of its class. CS results in a higher throughput and utilization; however, it suffers from two drawbacks. The first is, if input rates are asymmetric, the classes with higher input rates dominate the system and consume most of the storage and although the total throughput and utilization of the system is high, the contribution of the low input classes to this throughput may be unfairly low. In other words, the high input classes deny others the use of the system [KAMO 76A]. Another drawback is that, even with uniform input rates, if all of the input rates are scaled up simultaneously by a factor (say η), as η increases to infinity, in contrast to intuition, there is a nonzero probability that any one of the classes of messages will be absent from

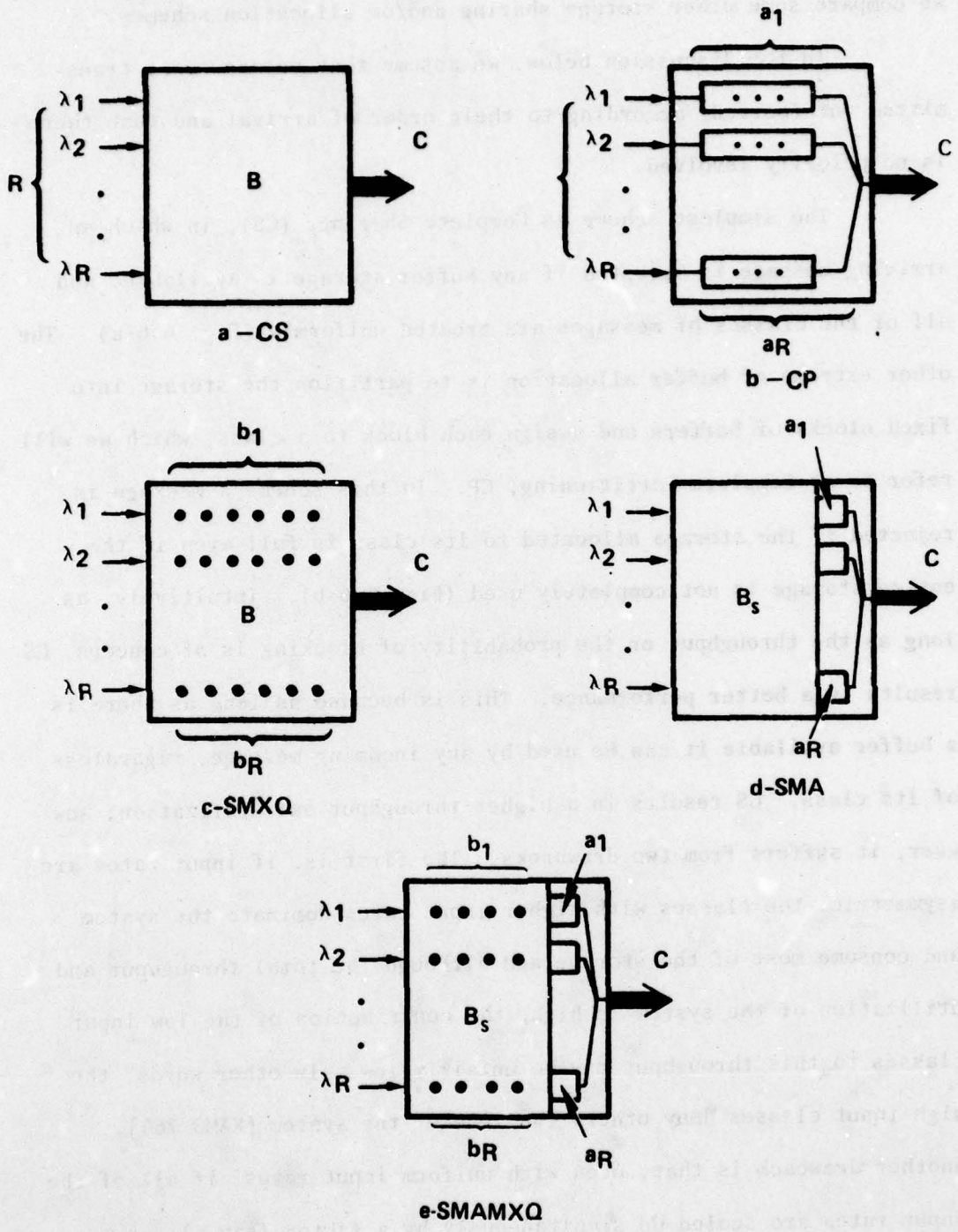


Figure 6.6. Storage Allocation and Sharing Schemes

the storage. This phenomena may not be desirable in certain situations, as we may require that in the case of infinite input rates there should be a nonzero probability that one or more messages of each class be present in the system.

The above considerations lead us to impose some restriction on the contention for the space. The first drawback may be remedied by imposing a limit on the number of buffers that can be occupied by any class simultaneously. The scheme which incorporates this idea will be called Sharing with Maximum Queue, (SMXQ), Fig. 6.6-c. Even with SMXQ we are not guaranteed to have remedied the second drawback, namely, a nonzero probability of being absent from the system at a very high input rate of all classes. In order to remove this deficiency we have to allocate a minimum number of buffers to each class. This leads us to a scheme which has been called Sharing with Minimum Allocation, (SMA). In this scheme, besides the minimum allocated buffers to each class, there is a pool of buffers to be shared by different classes of messages. If this sharing is done without any limitation, the scheme will be simply called SMA (Fig. 6.6-d). If, however, we impose some limitation on the maximum number of messages of each class that can be present in the system, then the scheme will be called Sharing with Minimum Allocation and Maximum Queue, SMAMXQ (Fig. 6.6-e).

Statistical multiplexors have been modeled and analyzed by others, [CHU 72 and the references therein], but the problem we are addressing here has not been studied before. Rich and Schwartz [RICH 75], and Drukey [DRUK 75] made a preliminary study of buffer allocation in a Store-and-Forward (S/F) node. A comprehensive study

of buffer sharing in S/F nodes was carried out by Kamoun and Kleinrock [KAMO 76A and B]. Although most of the terminology used in this paper is from these last two works, the present study is different from theirs since in a S/F node, each class of input stream has a dedicated output channel, whereas in a statistical multiplexor a single output channel is shared by all of the input messages. This introduces major differences in the analysis.

We model the system by a single server (i.e., one channel) queueing system with finite waiting room of size B. The input to the system consists of R independent Poisson streams with rates λ_r, (r = 1, 2, ..., R). In order to analyze the system, we require that all of the message lengths be distributed exponentially with the average length $1/\mu$ bits. Assuming the output channel speed to be C, the service time is exponentially distributed with rate μC. We further assume that any arriving messages which are not accepted leave the system without service and the accepted ones are served on a first-come-first-served basis. In the discussion below, a_r and b_r are the minimum number of buffers allocated and the maximum queue length of class r messages, respectively.

The above allocation schemes are analyzed in Appendix D, where we have presented either explicit expressions or numerical algorithms to calculate different performance measures, namely, probability of blocking, system throughput, etc.

In the remainder of this section we present some performance curves and compare the different sharing schemes. In all of the numerical cases we study, we assume R = 2 (i.e., there are two input classes).

Figures 6.7-a and 6.7-b show the behavior of the (normalized) throughput and the probability of blocking with respect to the allocated storage when the CP scheme is used. In these figures a pool of 10 buffers is shared by two classes of messages. The horizontal axis represents the number of buffers allocated to the first class, a_1 (hence $a_2 = 10 - a_1$). It can be seen that there is an optimal allocation of storage which maximizes the total throughput (or minimizes the blocking probability). The optimal partitioning of storage depends on the values of the input rates. A good approximation to this partitioning is to divide the storage according to relative values of the input rates (i.e., $a_1 = (\lambda_1/(\lambda_1 + \lambda_2))B$ and $a_2 = (\lambda_2/(\lambda_1 + \lambda_2))B$. The performance curve is flatter for larger input rates. This means that deviation from the optimal partitioning does not affect the total throughput severely. Therefore, as can be seen from the curves for ρ_1' and ρ_2' (ρ_r' is the normalized throughput of class r messages), one is able to control the contribution of each class to the output by changing the partitioning of the storage with little degradation of the performance. For example in Fig. 6.7-a, $a_1 = 6$ and $a_2 = 4$ results in an almost equal contribution to the output by each class, and the drop in the maximum throughput (which is achieved at $a_1 = 3$ and $a_2 = 7$) is negligible. Also we have shown the blocking probability and the throughputs for the complete sharing scheme in these figures (the dashed lines). Note that the total throughput of CS (which is about 0.99) is larger than the maximal throughput of the CP scheme. If we desire to provide each of the classes with a minimum throughput, however, the CS scheme fails to guarantee this level of service. As we will see shortly, in CS any one

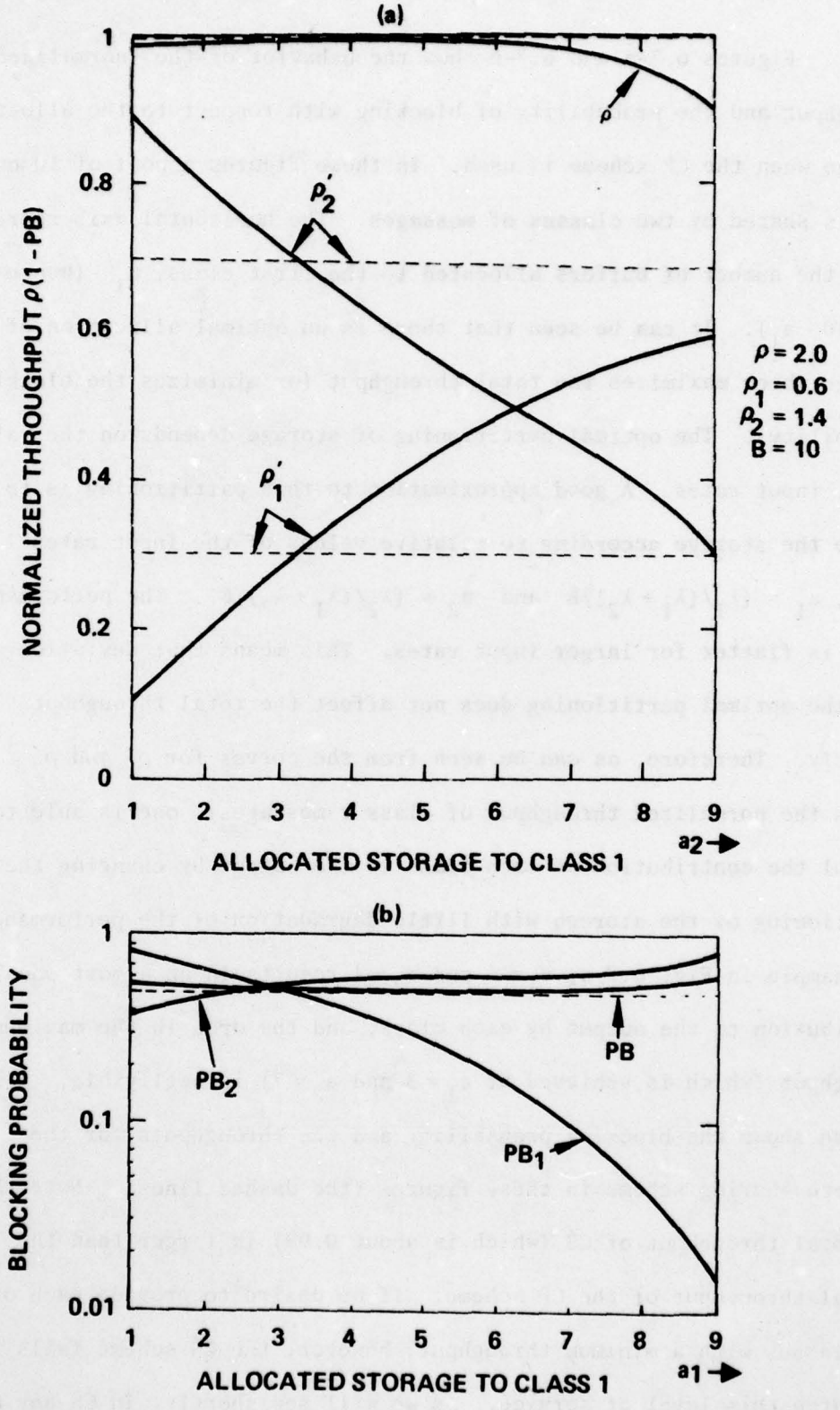


Figure 6.7. Effect of Allocated Storage in the CP Scheme.

of the input streams has the potential of monopolizing the system and denying others the use of the buffer storage.

In Fig. 6.8 a pool of B = 6 buffers is shared according to the SMXQ scheme. Notice that we have the following:

i. If $b_1 = b_2 = 3$ then SMXQ is equivalent to CP.

ii. If $b_1 = b_2 = 4$ then SMXQ is equivalent to SMA.
($a_1 = a_2 = 2$)

iii. If $b_1 = b_2 = 5$ then SMXQ is equivalent to SMA.
($a_1 = a_2 = 1$)

iv. If $b_1 = b_2 = 6$ then SMXQ is equivalent to CS.

The curves show the normalized throughput of each class when λ_1 remains fixed and 0.3 and λ_2 varies. It is seen how ρ_1' degrades when $b_1 = b_2 = 6$ (CS). In fact, as $\lambda_2 \uparrow \infty$, $\rho_1' \downarrow 0$. Fig. 6.8 shows how one can prevent this phenomena by permanently allocating some buffers to class 1 messages. The asymptotic values of ρ_1' and ρ_2' can be calculated according to the analysis given in Appendix D.

Fig. 6.9 shows the probability of being absent from the system as a function of the input rate to the system. In this figure the two input rates are identical (i.e., $\lambda = \lambda_1 + \lambda_2 = 2\lambda_1 = 2\lambda_2$). The horizontal axis represents the total input rate λ. Again, a pool of B = 6 buffers is shared according to the SMXQ scheme as in Fig. 6.8. As λ increases, all curves except the CS curve eventually decrease to zero.

As we pointed out earlier, if the measure of performance is the total throughput or the probability of blocking, then the CS scheme results in a better performance. This fact is shown in Fig. 6.10, in

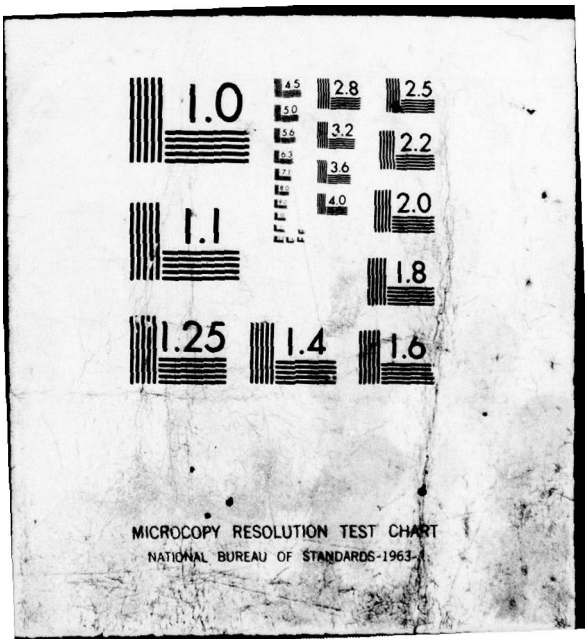
AD-A077 404 CALIFORNIA UNIV LOS ANGELES DEPT OF COMPUTER SCIENCE F/G 17/2.1
ADVANCED TELEPROCESSING SYSTEMS.(U)
JUN 78 L KLEINROCK MDA903-77-C-0272

UNCLASSIFIED NL

4 OF 4
AD
A077404



END
DATE
FILMED
1-80
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

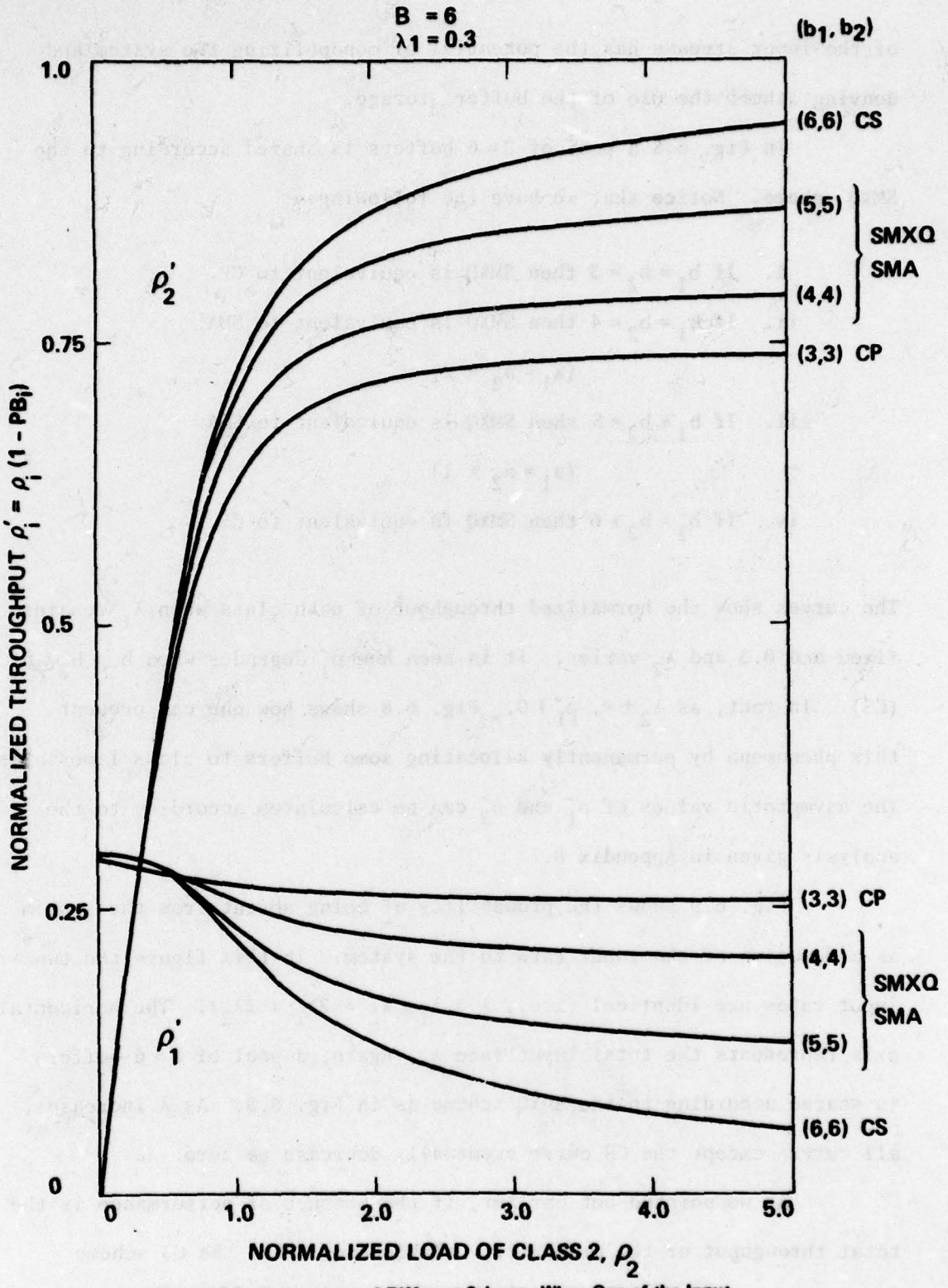


Figure 6.8. Comparison of Different Schemes When One of the Input Rates Increases.

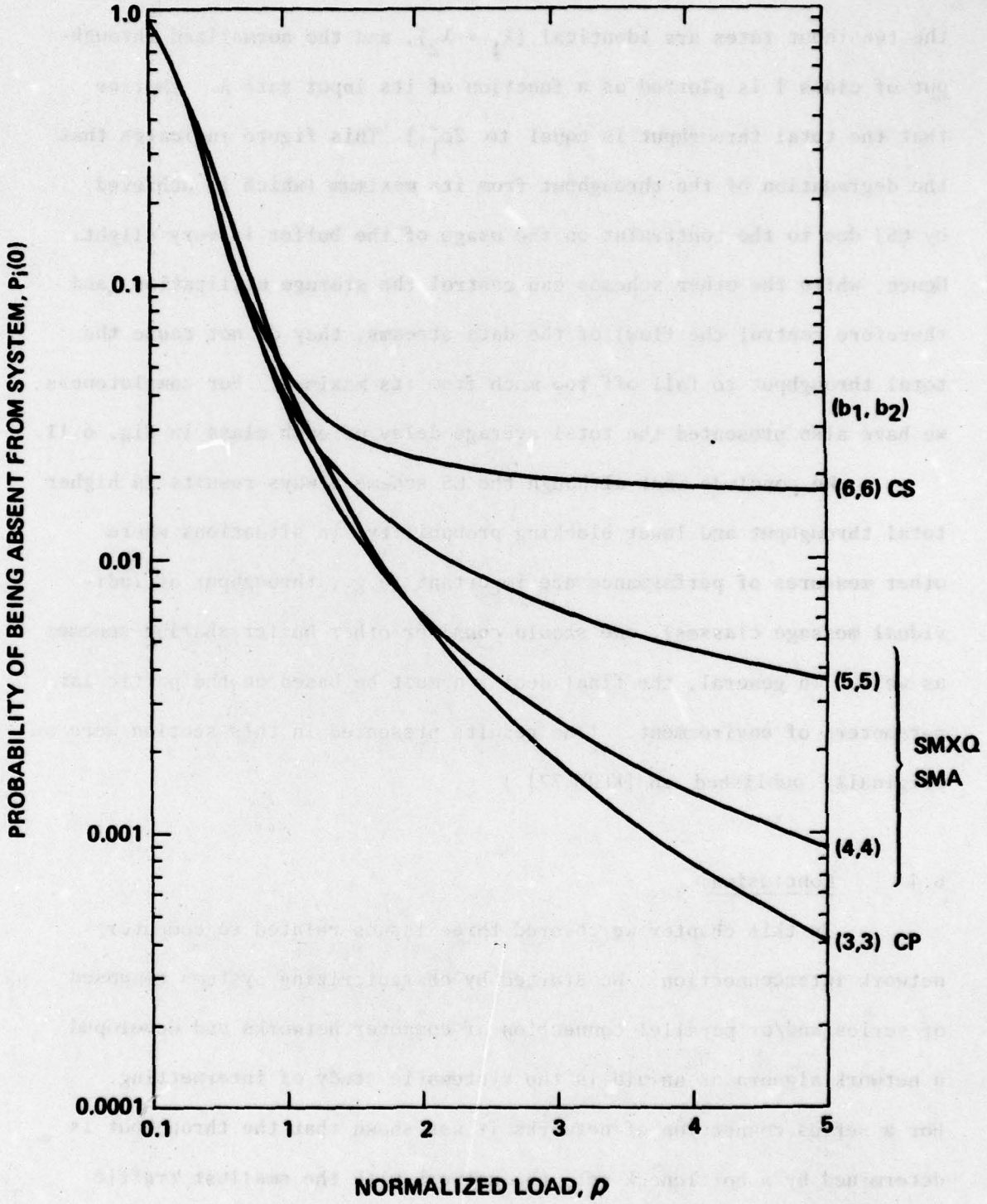


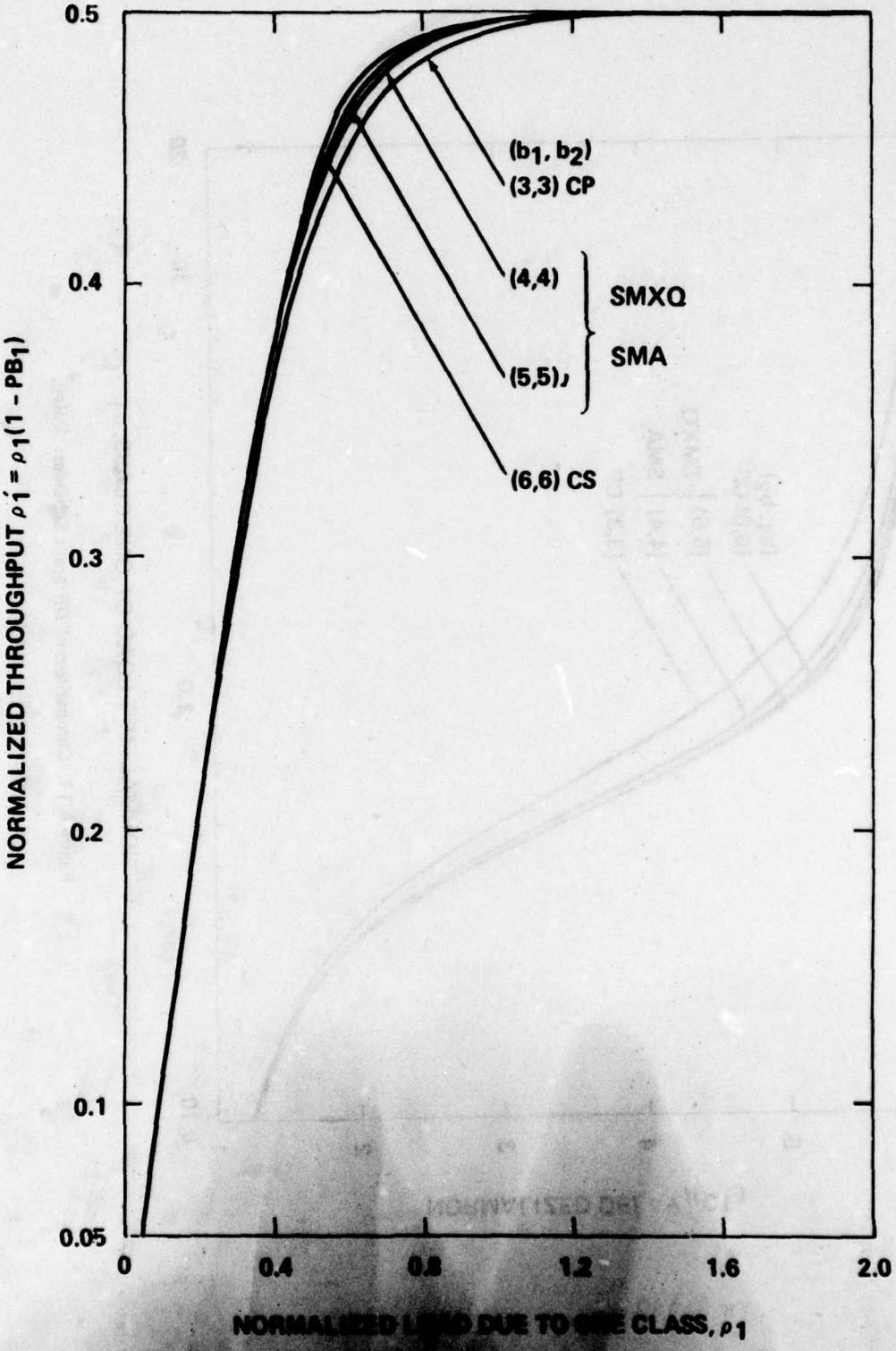
Figure 6.9. Comparison of Schemes: Probability of Being Absent from the System.

which a total of B = 6 buffers are shared as in Fig. 6.8. In this figure the two input rates are identical ($\lambda_1 = \lambda_2$), and the normalized throughput of class 1 is plotted as a function of its input rate λ. (Notice that the total throughput is equal to $2\rho_1'$.) This figure indicates that the degradation of the throughput from its maximum (which is achieved by CS) due to the constraint on the usage of the buffer is very slight. Hence, while the other schemes can control the storage utilization (and therefore control the flow) of the data streams, they do not cause the total throughput to fall off too much from its maximum. For completeness we have also presented the total average delay of each class in Fig. 6.11.

We conclude that although the CS scheme always results in higher total throughput and lower blocking probability, in situations where other measures of performance are important (e.g., throughput of individual message classes), one should consider other buffer sharing schemes as well. In general, the final decision must be based on the particular parameters of environment. (The results presented in this section were originally published in [KERM 77].)

6.4 Conclusion

In this chapter we covered three issues related to computer network interconnection. We started by characterizing systems composed of series and/or parallel connection of computer networks and developed a network algebra as an aid in the systematic study of internetting. For a series connection of networks it was shown that the throughput is determined by a bottleneck net, the network with the smallest traffic handling capacity. In this case, part of the resources of the other


0.5
0.4
0.3
0.2
0.1
0.05
0
0.4
0.8
1.2
1.6
2.0
NORMALIZED THROUGHPUT $\rho'_1 = \rho_1(1 - PB_1)$
NORMALIZED LOAD DUE TO ONE CLASS, ρ_1
(b_1, b_2)
(3,3) CP
(4,4)
SMXQ
SMA
(5,5)
(6,6) CS

Throughput.

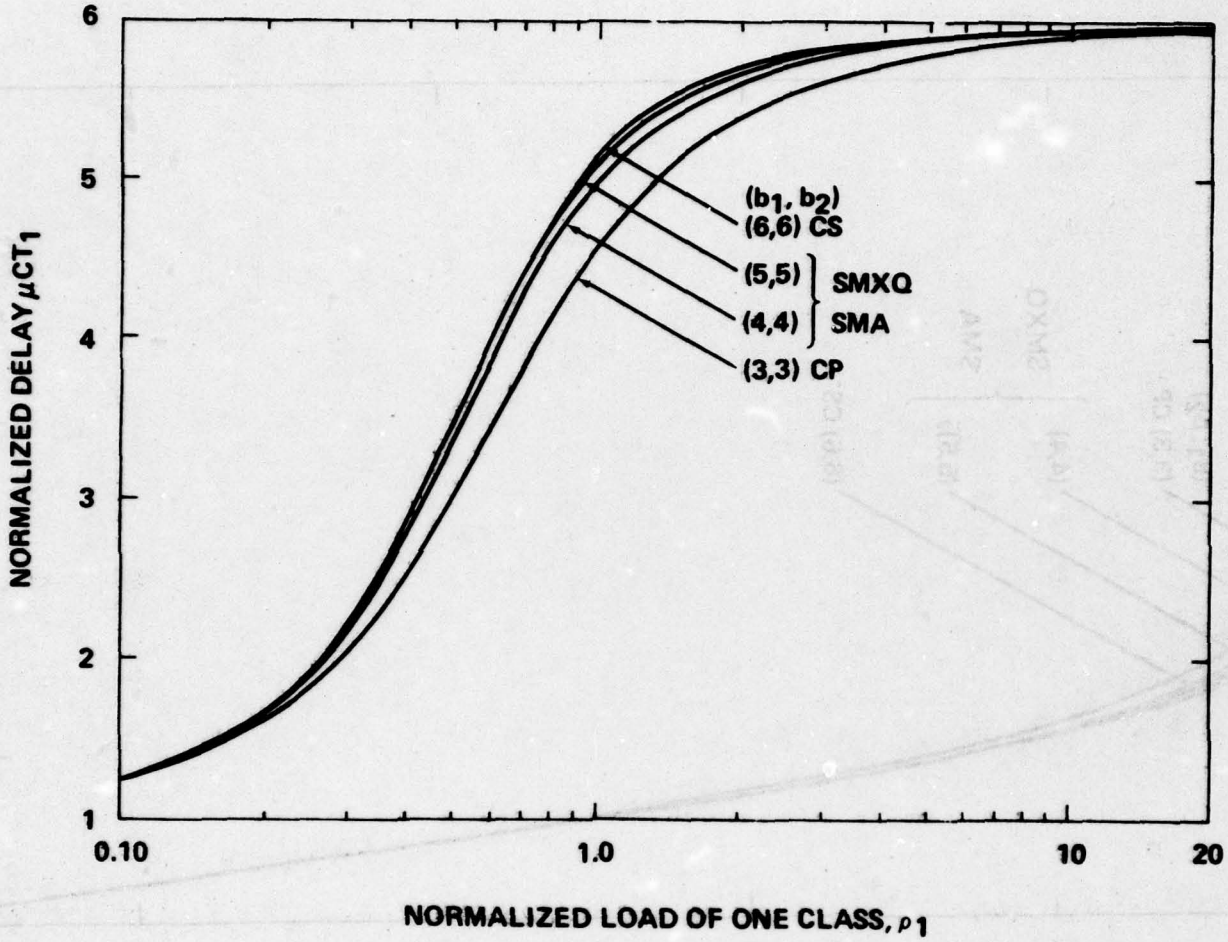


Figure 6.11. Comparison of Different Schemes: Delay.

networks is wasted. The analysis of networks connected in parallel was carried out only for the special case in which the maximum load factors of the connected networks are identical. We showed that a system composed of a parallel connection of such networks is identical to a single network with a path length and channel capacity equal to the sum of the path lengths and the sum of the channel capacities of its components, and its maximum load factor is equal to the maximum load factor of each compenent network.

The second subject we studied was the problem of optimal allocation of limited resources (window buffers and/or transmission capacity) among a number of parallel communication paths. It was shown that the solution to this problem depends on the optimality criteria. We studied this optimization problem with respect to three different criteria: throughput, delay and throughput-delay. For the throughput-delay criteria we chose power (defined as the ratio of throughput and delay) as our measure of performance. It was shown that only for a special case (optimal window allocation between parallel paths when the capacities are known) are the solutions the same with respect to the three performance measures.

The last issue we covered in this chapter was concerned with buffer allocation in a destination node where a number of communication paths or nets merge. This subject was carried out in a more general context of buffer allocation schemes in a multiplexing node. Different allocation schemes were analyzed and their performances were compared with each other. (These schemes were originally proposed in [KAMO 76A and 76B] for a store-and-forward node.) It was demonstrated how further

control on the flow of data can be achieved through different allocation and/or sharing schemes. These controls are supplementary to the static and dynamic flow controls which we studied in the previous chapters.

CHAPTER 7

CONCLUSIONS AND SUGGESTIONS FOR FURTHER RESEARCH

This research was motivated by the need to develop efficient techniques to enhance the traffic handling capability of computer networks. This capability is usually expressed in terms of "throughput" and "delay", which are affected by both the network switching and flow control techniques. This dissertation is addressed to the analytic modeling and performance evaluation of these techniques. Except for some suggestions regarding the physical realization, we have ignored most of the implementation problems of the techniques we have analyzed. The analytic models which we used are, in many cases, rather simplistic; nevertheless, they help in understanding the implications of the real system behavior.

This dissertation can logically be divided into two parts. In the first part (Chapters 2 and 3), we studied switching problems. In the second part (Chapters 4 through 6) the flow control problems in computer networks were discussed.

The main conclusions of this research are as follows:

(1) The hybrid switching technique, virtual cut-through, is a fine solution to the switching problem in computer networks. The network delay for cut-through switching is almost always better than that of message switching; only when the channel error rates are high, does message switching result in a smaller delay.

(2) Our tradeoff studies showed that the selection of a network switching technique should be based on a careful study of the network topology (especially the communication path lengths) and traffic characteristics (message lengths and message arrival rates).

(3) The contention effect in computer networks was explicitly demonstrated in our study of static flow control. It was shown that in order to improve the throughput-delay performance of a network, window size and timeout should be carefully selected.

(4) In Chapter 5 it was shown that the throughput-delay performance of a network can be further improved by dynamically adjusting the window size and certain other network parameters to the stochastic load fluctuations. In that chapter we presented a methodology to implement a dynamic flow control.

(5) In Chapter 6 some issues in interconnection of computer networks were addressed. A network algebra for series and/or parallel connection of networks was developed. It was shown that in series connection of networks, the bottleneck network determines the throughput of the system. The study of parallel-connected networks was carried out only for the special case where all the networks have the same load factor ($\ell^* = w/n_h$).

(6) The solution to the optimal allocation of limited resources (window buffers and channel capacity) among a number of parallel networks depends on the optimality criteria, throughput, delay, and throughput-delay. It was shown that only in a special case was the solution with respect to the three performance measures the same. In many cases the optimal allocation of window buffers and channel capacity

is to assign all the resources to only one of the networks.

(7) Our study of buffer allocation schemes in a node where a number of networks merge together showed that in order to provide a fair grade of service to different streams of data which merge together, some nontrivial sharing and allocation schemes should be implemented.

Throughout the dissertation numerical examples were presented for certain values of different parameters. The models and methodologies are, however, independent of these particular values and may be applied to systems with different parameters.

Extensions for Further Research

While there is an abundant literature on issues related to flow control in computer networks, the analytic and quantitative study of flow control is still in its infancy. In the analytic models which we used for the window mechanism, the assumption was that the sending user always has a message to transmit; furthermore, in our analysis there was no account for the admission delay to the network. These assumptions may be relaxed in order to make the model more realistic. More important, our model is based on the assumption of single packet messages in which the problem of message sequencing was not considered. Further research and a more refined model is necessary to investigate the important problem of message sequencing.

The Markov decision process which was formulated in Chapter 5 appears to have extended applications to decision processes in which there is a lead-time for a decision to become effective, for example, in inventory systems. The identification of other systems for which our formulation is applicable represents a worthwhile investigation.

APPENDIX A

LIST OF OFTEN-USED SYMBOLS

γ_{jk}	Average number of messages entering network per second with origin j and destination k.
λ_i	Average number of messages entering i^{th} channel per second.
λ	Average number of messages entering a channel in a balanced net.
λ^*	Maximum input rate or maximum throughput.(Chap. 4,6)
γ	Total arrival rate of messages from external sources.(Chap.2)
γ	Maximum transmission rate of a channel (=μC)
C	Total capacity of a link.
$1/\mu$	Average message length.
t_0	$=1/\mu C$ = average transmission time of a message over a channel.
t_h	Average transmission time of a header.
α	t_h/t_0 (Chap. 2)
δ	$\lceil 1/\alpha \rceil$ (Chap. 2)
N_{ch}	Number of channels per link.
$\tilde{n}_h$	A random variable representing the number of hops a message goes through.
$p_h^{(n)}$	$= Pr[\tilde{n}_h=n]$
$\bar{n}_h$	Average of $\tilde{n}_h$.
N(z)	z transform of number of hops a message goes through = $E[z^{\tilde{n}_h}]$.
$\tilde{n}_c$	Number of cut-throughs a message experiences ($\tilde{n}_c \leq \tilde{n}_h - 1$).
$\bar{n}_c$	Average number of cut-throughs.
$\bar{n}_c^n$	$E[\tilde{n}_c \mid \tilde{n}_h = n]$

$C_{n,\rho}(z)$	$E[z^{\tilde{n}_c} \mid \tilde{n}_h = n, \rho]$.
$C_{\rho}(z)$	$E[z^{\tilde{n}_c} \mid \rho]$.
T	Network delay.
T_m	Average network delay when message switching is used.
T_c	Average network delay when cut-through switching is used.
T^*	Maximum delay. (Chap. 4, 6)
T_{ee}	End-to-end delay.
$T_a, \hat{T}_a$	Acknowledgement delay.
$T_r, \hat{T}_r$	Round-trip delay.
P_w	Probability of waiting.
P_e	Channel error rate.
P_ℓ	Probability of loss.
P_r	Probability of retransmission.
$\overline{S_m^{(p)}}$	E[total amount of storage required on a path in message switching].
$\overline{S}_m$	E[total amount of storage required at a node].
$\overline{S_c^{(p)}}$	Same as $\overline{S_m^{(p)}}$, but for the cut-through system.
$\overline{S}_c$	Same as $\overline{S}_m$, but for the cut-through system.
Z_{jk}	Average message delay for messages with origin j and destination k.
$\tilde{s}$	Total delay in a node.
$\tilde{w}$	Waiting time in a node.
ρ	The utilization factor of each channel.
ℓ	Load factor.

ℓ^*	Maximum load factor.
w	Window size.
τ	Timeout.

APPENDIX B

THEOREMS AND PROOFS FOR CHAPTER 2

B.1 Multiple Exponential Channels

As we have considered a communication network in which adjacent nodes are connected via multiple channel links, we need the queueing results for a multiple server system. We define ρ as the ratio of the average input data rate to the maximum transmission rate

$$\rho = \frac{\lambda}{\mu C}$$

where C is the total capacity of a link which is divided into N_{ch} channels. The solution for P_k, the probability of having k messages at a node when $\rho < 1$ and there are N_{ch} channels (servers) is given below [KLEI 75].

$$P_k = \begin{cases} P_0 \dfrac{\rho^k N_{ch}^k}{k!} & k \leq N_{ch} \\ P_0 \dfrac{\rho^k (N_{ch})^{N_{ch}}}{N_{ch}!} & k \geq N_{ch} \end{cases} \tag{B.1}$$

where

$$P_0 = \left[\sum_{k=0}^{N_{ch}-1} \frac{(N_{ch}\rho)^k}{k!} + \frac{(N_{ch}\rho)^{N_{ch}}}{(1-\rho)N_{ch}!} \right]^{-1} \tag{B.2}$$

$$P_w = \Pr\left[\geq N_{ch}\right] = \Pr[\text{waiting}] = P_0 \frac{(N_{ch}\rho)^{N_{ch}}}{(1-\rho)N_{ch}!} \tag{B.3}$$

$$T = E[\text{time spent in a node}] = \frac{N_{ch}}{\mu C} + \frac{P_w}{\mu C(1-\rho)} \tag{B.4}$$

$$\bar{N} = \lambda T = N_{ch}\rho + \frac{P_w}{1 - \rho}\rho \qquad (B.5)$$

B.2 Theorem 2.3 and its Proof

The average network delay for the cut-through system ir a network with noisy channels, conditioned on the path length ($\tilde{n}_h = n$), is given by

$$T_c = T_c^{(n)}(n) \qquad (B.6)$$

where

$$T_c^{(j)}(i) = \begin{cases} \displaystyle\sum_{k=0}^{j-1} \Big\{ T_{cc}^{k}(i,j) + T_c^{(j-k-1)}(i)(1 - P_e)^{k+1} \\ \qquad + T_e^{(j)}(j)\left[1 - (1 - P_e)^{k+1}\right] \Big\} P_c^{(j)}(k) & i > j > 0 \\ \dfrac{\displaystyle\sum_{k=0}^{j-1} \left[T_{cc}^{(k)}(i,j) + T_c^{(j-k-1)}(i)(1 - P_e)^{k+1} \right] P_c^{j}(k)}{\displaystyle\sum_{k=0}^{j-1} (1 - P_e)^{k+1} P_c^{(j)}(k)} & i = j > 0 \\ 0 & j = 0 \end{cases} \qquad (B.7)$$

where

$$T_{cc}^{(k)}(i,j) = \begin{cases} E[\tilde{s}] + kt_h = \left[\dfrac{N_{ch}}{\mu C} + \dfrac{P_w}{\mu C(1 - \rho_e)}\right] + kt_h & i = j \\ E[\tilde{s} \mid \tilde{w} > 0] + kt_h = \dfrac{N_{ch}}{\mu C} + \dfrac{1}{\mu C(1 - \rho_e)} + kt_h & i > j \end{cases} \qquad (B.8)$$

and

$$P_c^{(n)}(k) = \begin{cases} (1 - P_w)^k P_w & \text{if} \quad 0 \le k < n - 1 \quad \text{and} \quad n > 1 \\ (1 - P_w)^k & \text{if} \quad k = n - 1 \end{cases} \qquad (B.9)$$

where ρ_e is the effective traffic rate.

Proof:

We start with some notation:

$\tilde{n}_{cc}$: number of continuous cut-throughs

$P_c^{(n)}(k)$ = $Pr[\tilde{n}_{cc} = k \mid \tilde{n}_h = n]$; density function of $\tilde{n}_{cc}$

$T_c^{(j)}(i)$ = E[delay in the cut-through system | path length = i and (i - j) hops have been made through (j hops remain to go)] $1 \leq j \leq i$

$T_{cc}^{(k)}(i,j)$ = E[delay when a message makes k continuous cuts | path length = i and (i - j) hops have been made through (j hops remain to go)] $1 \leq j \leq i$ and $0 \leq k \leq j - 1$.

The density function $P_c^{(n)}(k)$ is given by Eq. (B.9).

We first prove Eq. (B.8). Let us consider a tagged message. If $i > j$, this indicates that after having been transmitted over the first (i - j) hops, the message is now blocked in an intermediate node. This implies that it should incur some queueing delay before it starts transmission again. On each of the k nodes that it cuts through, the message incurs t_h seconds of delay (only for assembling its header). Therefore we have

$$T_{cc}^{(k)}(i,j) = E[\tilde{s} \mid \tilde{w} > 0] + kt_h \qquad i > j$$

On the other hand, $i = j$ implies that the message is at the first node of a path. So, its time of residency in the first node is not conditioned on any event and we have

$$T_{cc}^{k}(i,i) = E[\tilde{s}] + kt_h \qquad i = j$$

This gives us Eq. (B.8).

To prove Eq. (B.7) we use a renewal type argument. On a path consisting of i hops, assume our tagged message has j hops to go. If the message makes k continuous cuts (with probability $P_c^{(j)}(k)$) and is received at a final node (see Section 2.3.2 for the definition of a final node) and has no bits in error, then it will incur $T_{cc}^{k}(i,j)$ seconds delay and will have (j - k - 1) hops to go. Delay for the remaining transmission is $T_c^{(j-k-1)}(i)$. On the other hand, if the message has some bits with error, it must be retransmitted over the remaining j hops; however, for retransmission its delay in the first node will not be a conditional delay, and it is as if the message were at the first node of a j hop path. Putting what we have said in mathematical form we have:

$$T_c^{(j)}(i \mid \tilde{n}_{cc} = k \text{ and no error}) = T_{cc}^{(k)}(i,j) + T_c^{(j-k-1)}(i)$$

$$T_c^{(j)}(i \mid \tilde{n}_{cc} = k \text{ and error}) = T_{cc}^{(k)}(i,j) + T_c^{(j)}(j)$$

so

$$\begin{aligned} T_c^{(j)}(i \mid \tilde{n}_{cc} = k) &= T_c^{(j)}(i \mid \tilde{n}_{cc} = k \text{ and no error}) \times \Pr\left[\begin{matrix}\text{no error on}\\ k+1 \text{ hops}\end{matrix}\right] \\ &\quad + T_c^{(j)}(i \mid \tilde{n}_{cc} = k \text{ and error}) \times \Pr\left[\begin{matrix}\text{at least one error}\\ \text{on } k+1 \text{ hops}\end{matrix}\right] \\ &= \left[T_{cc}^{(k)}(i,j) + T_c^{(j-k-1)}(i)\right]\left[(1-P_e)^{k+1}\right] \\ &\quad + \left[T_{cc}^{(k)}(i,j) + T_c^{(j)}(j)\right]\left[1 - (1-P_e)^{k+1}\right] \end{aligned}$$

Finally we have

$$T_c^{(j)}(i) = \sum_{k=0}^{j-1} T_c^{(j)}(i \mid \tilde{n}_{cc} = k) P_c^{(j)}(k)$$

After some algebra and collection of terms we get Eq. (B.7). Eq. (B.6) is simply an identity.

Q.E.D.

B.5 Theorem 2.4 and its Proof

For the cut-through system in a balanced network with noisy channels the useful traffic is related to the effective channel traffic according to the following relationships (ρ_e is the effective traffic and n is the path length).

$$\rho = \frac{n}{N_t(n)}\,\rho_e \tag{B.10}$$

where

$$N_t(i) = \frac{\sum_{k=0}^{i=1}\left[(k+1) + N_t(i-k-1)(1-P_e)^{k+1}\right]P_c^{(i)}(k)}{\sum_{k=0}^{i-1}(1-P_e)^{k+1}P_c^{(i)}(k)} \qquad i > 0 \tag{B.11}$$

$$N_t(i) = 0 \quad i = 0$$

Proof:

Let

$$N_t(i) \triangleq E[\text{number of times a message is transmitted on a path of length i}]$$

Notice that if channels are noiseless, then there is no retransmission and we have $N_t(i) = i$. Obviously the useful traffic will be related to effective traffic by Eq. (B.10).

To prove Eq. (B.11) we use a renewal type argument:

$$N_t(i \mid \tilde{n}_{cc} = k \text{ and no error}) = k + 1 + N_t(i-k-1)$$

$$N_t(i \mid \tilde{n}_{cc} = k \text{ and error}) = k + 1 + N_t(i)$$

so

$$N_t(i \mid \tilde{n}_{cc} = k) = (k + 1 + N_t(i - k - 1))(1 - P_e)^{k+1}$$

$$+ (k + 1 + N_t(i))\left[1 - (1 - P_e)^{k+1}\right]$$

Removing the condition on the number of continuous cuts we have

$$N_t(i) = \sum_{k=0}^{i-1} N_t(i \mid \tilde{n}_{cc} = k) P_c^{(i)}(k)$$

$$= \sum_{k=0}^{i-1} \left\{ \left[(k + 1) + N_t(i - k - 1)\right] (1 - P_e)^{k+1} \right.$$

$$\left. + \left[(k + 1) + N_t(i)\right] \left[1 - (1 - P_e)^{k+1}\right] \right\} P_c^{(i)}(k)$$

After some algebra and collecting similar terms we get Eq. (B.11).

For $i = 0$, because there is no transmission involved, we have

$$N_t(0) = 0$$

Q.E.D.

When $P_e = 0$ it can easily be shown that $N_t(i) = i$ and $\rho = \rho_e$.

APPENDIX C

SOME RESULTS FROM MARKOV DECISION THEORY

The following results are taken from Jewell [JEWE 63], Howard [HOWA 60, 71] and Ross [ROSS 70].

Consider a stochastic process X(t) which is observed at time points t = 0, 1, ... to be in one of a number of possible states. The set of states S will be labeled by non-negative integers 0, 1, 2, ..., N. Let A be a finite set of actions such that corresponding to each $a \in A$, a transition probability $p_{ij}(a)$, and a reward $r_{ij}(a)$ are uniquely specified. We assume that the rewards are bounded, i.e., $|r_{ij}(a)| < \infty$ for all a, i and j. When process X(t) is in state i at time t and action a is chosen, then

1. The next state of the process is chosen according to the transition probabilities of $p_{ij}(a)$.
2. On transition from state i to j under action a, the process earns a reward $r_{ij}(a)$.

Let a(t) be the action chosen at t; then assumption 1 is equivalent to stating that

$$\Pr[X(t+1) = j \mid X(0), a(0), X(1), a(1), \ldots, X(t) = i, a(t) = a] = p_{ij}(a) \tag{C.1}$$

This shows that both the cost and the transition probabilities are functions of only the last state and the subsequent action.

The actions should be taken according to a rule. We define a policy f to be any rule for choosing an action and $\mathcal{F}$ to be the class of all policies. The action chosen by a policy may, for instance, depend on the history of the process up to that point, or it may be randomized in the sense that it chooses an action with some probability P_a, $a \in A$.

For state i and action a chosen at that state, we define

$$R_i(a) \overset{\Delta}{=} \sum_{j=0}^{N} p_{ij}(a) r_{ij}(a) \qquad i = 0, 1, 2, \ldots, N \qquad (C.2)$$

$R_i(a)$ can be interpreted as the reward to be expected in the next transition out of state i; it will be called the expected immediate reward for state i, when action a is taken.

Suppose action a(t) is chosen by policy f at time t. This in turn specifies the state transition probabilities and reward at time t. Thus, f determines both the evolution in time of the Markov process X(t) and the sequence of rewards it earns. For a policy f, the expected total reward that the process X(t) earns in the next n + 1 time units when it starts in state i is

$$v_i^n(f) = E_f \left[\sum_{t=0}^{n} R_{X(t)}(a(t)) \mid X(0) = i \right] \qquad (C.3)$$

where the expectation is taken with respect to f. We define the average reward per unit time for X(t) which was initially started in state i as

$$g_i(f) = \lim_{n \to \infty} \left\{ \frac{1}{n+1} E_f \left[\sum_{t=0}^{n} R_{X(t)}(a(t)) \mid X(0) = i \right] \right\} \qquad (C.4)$$

where the limit exists, since the rewards are assumed to be bounded. We say policy f* is the average cost optimal over all policies if

$$g_i(f^*) = \max_f \{g_i(f)\} \qquad i \in S,\ f \in \mathcal{F} \tag{C.5}$$

An important sub-class of the class of all policies is the class of stationary policies, where a policy is said to be stationary if it is non-randomized and the action it chooses at time t only depends on the state of the process at time t. Thus, a stationary policy f is a function from state space to action space, i.e., $f(\cdot): S \to A$. It easily follows from Eq. (C.1) that if a stationary policy f is employed, then the sequence of states $\{X(t),\ t = 0, 1, 2, \ldots\}$ forms a Markov chain with transition probabilities $\{p_{ij}\} = \{p_{ij}(f(i))\}$; and it is for this reason that the process is called the Markov Decision Process [ROSS 70]. In the following we deal only with stationary policies, and we refer to this class of policies as F. Because a stationary policy uniquely defines the action for a state, say i, we use $R_i(f)$ and $p_{ij}(f)$ instead of $R_i(f(i))$ and $p_{ij}(f(i))$, respectively. Let $R(f) = \{R_i(f)\}$, $P(f) = \{p_{ij}(f)\}$, and $g(f) = \{g_i(f)\}$, then Eq. (C.4) can be written equivalently as follows.

$$g(f) = \lim_{n \to \infty} \left\{ \frac{1}{n+1} \sum_{t=0}^{n} [P(f)]^t R(f) \right\} \tag{C.6}$$

where $[P(f)]^t$, the t^{th} power of $P(f)$, is the t-step probability state transition matrix under the stationary policy f.

In [HOWA 60] it is shown that for a long time horizon n, the expected total reward of the process under stationary policy f, when it starts from state i is given asymptotically by

$$v_i^n(f) = g_i(f)n + v_i(f) \tag{C.7}$$

where $v_i(f)$ is referred to as the asymptotic intercept of state i. For large n, only the rate $g_i(f)$ is important for evaluation of $v_i^n(f)$.

For each stationary policy f, since S is finite, if X(t) is an irreducible Markov chain then X(t) possesses a unique stationary state probability distribution $\{\pi_i(f)\}$ such that [FELL 50]

$$\Pi_j = \sum_{i=0}^{N} \Pi_i p_{ij}(f) \qquad j = 0, 1, 2, \ldots, N$$

$$\Pi_i \geq 0 \qquad i = 0, 1, 2, \ldots, N \tag{C.8}$$

and

$$\sum_{i=0}^{N} \Pi_i = 1$$

From the ergodic theorem in the theory of Markov processes we have [CHUN 67], [HOWA 60,71]

$$g(f) = g_i(f) = \sum_{j=0}^{N} \Pi_j R_j(f) \tag{C.9}$$

where g(f) is defined as the cost rate or the expected cost rate of the process X(t). For ergodic chains and stationary policies Eq. (C.5) simplifies to

$$g(f^*) = \max_{f \in F} g(f) \tag{C.10}$$

Eq. (C.7) shows that the reward gained by the process is maximal when it starts in any state if the actions are chosen under stationary policy f^*. For a finite state space, if every stationary policy results in an irreducible Markov chain, then such an optimal policy exists [BLAC 62], [JEWE 63], [ROSS 70], [SOBE 73].

Given a stationary policy f, the reward rate of the resulting Markov chain can be determined by the following equations [HOWA 60, 71]

$$g(f) + v_i(f) = R_i(f) + \sum_{j=0}^{N} p_{ij}(f)v_j(f) \qquad i = 0, ., \ldots, N \tag{C.11}$$

The set of (N + 1) linear simultaneous equations are not sufficient to determine the v's and g(f). In fact, except for g(f), only relative values of the v's can be determined. Fortunately, a set of relative values of the v's is sufficient for the purpose of the following iterative method in solving for an optimal policy.

The Policy-Iteration Method [HOWA 60, 71]

Based on Eq. (C.11), an iterative method to find an optimal stationary policy has been developed in [HOWA 60]. In the following we describe the algorithm, called the policy-iteration method, for the special case that for all policies $f \in F$, the probability state transition matrix $P(f)$ results in a single chain Markov process. For multichain systems, some modifications to the algorithm are necessary, see [HOWA 60, 71].

The basic iteration cycle in the policy-iteration method is diagrammed in Fig. C.1. The upper box, the value-determination, yields the g and v's corresponding to a given choice of $R(f)$ and $P(f)$. The lower box yields the $P(f)$ and $R(f)$ that increase the gain for a given set of v's. We may enter the iteration cycle in either box with an arbitrary initial policy or an arbitrary set of v's. It is, however, necessary to require that in the policy-improvement routine, if the decision f(i) for state i given by the old policy yields as large a value

for the test quantity as any of the other actions, the decision is left unchanged. The stopping rule is as follows:

> The optimal policy is reached (g is maximized) when the policies in two successive iterations are identical.

The fact that the algorithm given in Fig. C.1 terminates in a finite number of steps and that it actually results in an optimal policy is established in [HOWA 60, 71].

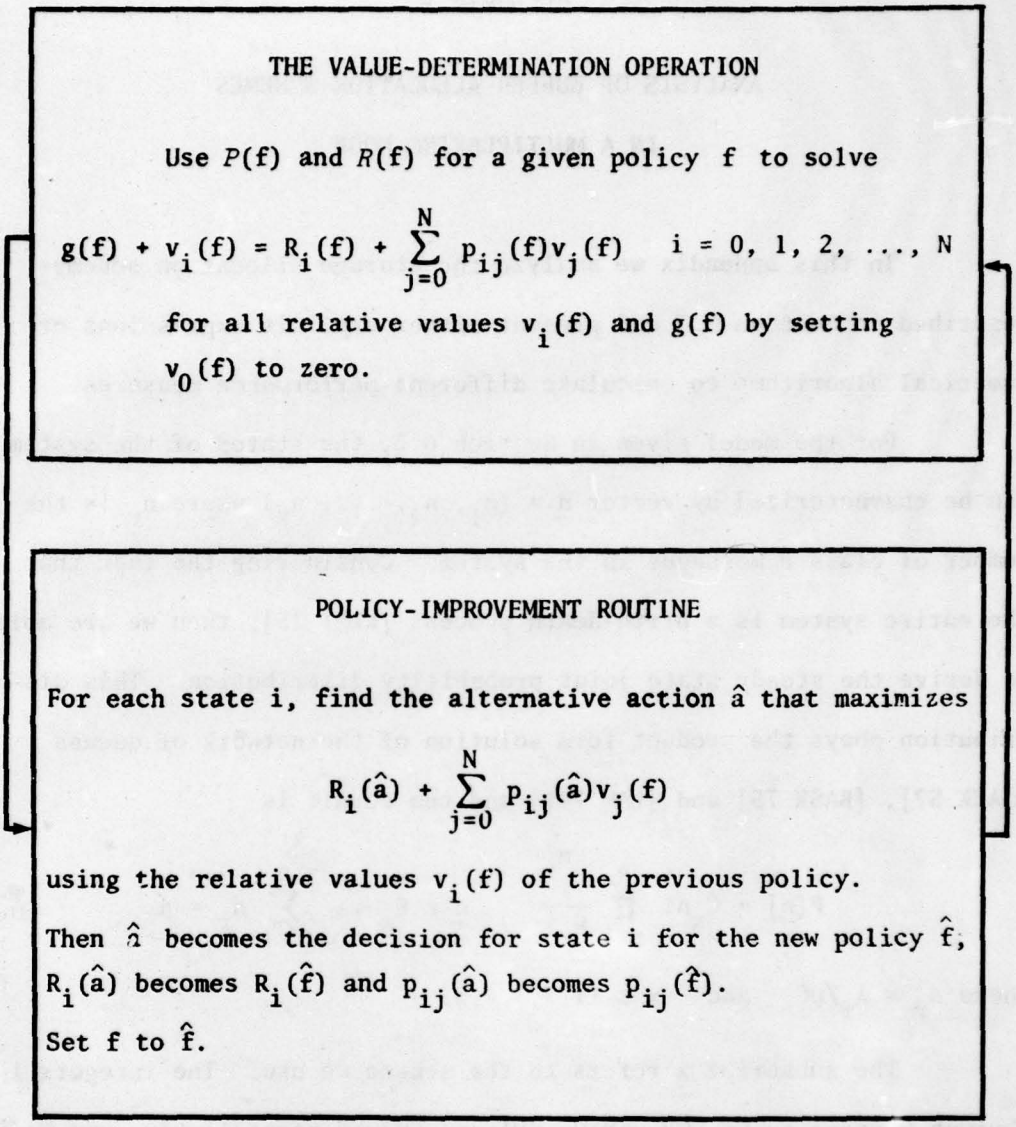


Fig. C.1. The Iteration-Cycle [HOWA 60]

APPENDIX D

ANALYSIS OF BUFFER ALLOCATION SCHEMES IN A MULTIPLEXING NODE

In this appendix we analyze the storage allocation schemes described in Section 6.3 and present either explicit expressions or numerical algorithms to calculate different performance measures.

For the model given in Section 6.3, the states of the system can be characterized by vector $\underline{n} = (n_1, n_2, \ldots, n_R)$ where n_r is the number of class r messages in the system. Considering the fact that the entire system is a birth-death process [KLEI 75], then we are able to derive the steady state joint probability distribution. This distribution obeys the product form solution of the network of queues [JACK 57], [BASK 75] and [LAM 76B] and the result is

$$P[\underline{n}] = C_x n! \prod_{r=1}^{R} \frac{\rho_r^{n_r}}{n_r!} \qquad \underline{n} \in F_x \; ; \quad \sum_{r=1}^{R} n_r = n \tag{D.1}$$

where $\rho_r = \lambda_r/\mu C$ and $x \in \{1,2,3,4,5\}$

The subscript x refers to the scheme we use. The integers 1 through 5 refer to CS, CP, SMXQ, SMA and SMAMXQ respectively, and F_x is the set of possible states for scheme x. C_x is a normalization constant defined so that the probabilities $P[\underline{n}]$ sum to one.

$$C_x^{-1} = \sum_{\underline{n} \in F_x} \left[n! \prod_{r=1}^{R} \frac{\rho_r^{n_r}}{n_r!} \right] \quad ; \quad n = \sum_{r=1}^{R} n_r \tag{D.2}$$

Notice that C_x is also the probability that the system is empty.

In the following we will first calculate C_x and then other quantities of interest. In contrast to the analysis in [KAMO 76B], we will see that except for the CS scheme, values of C_x cannot be found explicitly; as an alternative, however, we give efficient algorithms for their calculations.

Throughout the remainder of this appendix we will use the following notation:

a_r = number of buffers allocated to class r messages (this is used in CP, SMA and SMAMXQ).

b_r = maximum number of buffers of the shared storage which can be occupied by class r messages at any time.

B = total number of buffers.

B_s = total number of shared buffers.

Notice that we have the following relationships:

$B = B_s$ in CS

$B = \sum a_r$ and $B_s = 0$ in CP

$B_s = B - \sum a_r$ in SMA and SMAMXQ.

For each scheme we will consider two special cases, and for each case we will study the limiting behavior of the system. These cases are:

1. $\lambda_r = \eta\lambda_r^o$, $r = 1, 2, \ldots, R$ and $\eta \uparrow \infty$
2. Only one of the input rates, λ_i, grows to infinity.

D.1 Complete Sharing (CS)

The set of feasible states for this scheme is

$$F_1 = \left\{ n \mid \sum_{r=1}^{R} n_r \leq B,\ 0 \leq n_r \quad r = 1, 2, \ldots, R \right\} \tag{D.3}$$

The easiest way to analyze this system is to recall the well-known property of the Poisson process [FELL 50], namely the outcome of merging a number of Poisson streams is still a Poisson stream. By virtue of this fact, the model to be analyzed reduces to an M/M/1 queueing system with finite waiting room B, arrival rate $\lambda = \Sigma \lambda_r$, and service rate μC. The equations describing the behavior of the system are well known [KLEI 75] and we report them here.

$$C_1^{-1} = P(0)^{-1} = \frac{1 - \rho}{1 - \rho^{B+1}} \tag{D.4}$$

$$PB = Pr\left[\sum n_r = B\right] = \frac{1 - \rho}{1 - \rho^{B+1}} \rho^B \tag{D.5}$$

where $\rho = \lambda/\mu C$ and $\lambda = \Sigma \lambda_r$

Notice that the probability of blocking is the same for all classes of messages.

The marginal distributions can easily be found as follows:

Let

$$Pr[n_r = k, n] \triangleq Pr[\text{number of class r messages in the system is k and total number of messages in the system is n}]$$

then

$$Pr[n_r = k, n] = \sum_{\substack{\sum_{i \neq r} n_i = n-k \\ n_i \geq 0}} P[n_1, n_2, \ldots, n_{r-1}, k, n_{r+1}, \ldots, n_R]$$

Using the value of $P[\underline{n}]$ from Eq. (D.1) we have

$$\Pr[n_r = k,\ n] = P(0) \sum_{\substack{\sum_{i\neq r} n_i = n-k \\ n_i \geq 0}} n! \left(\prod_{\substack{i=1 \\ i\neq r}}^{R} \frac{\rho_i^{n_i}}{n_i!} \right) \frac{\rho_r^k}{k!}$$

which can be written as

$$\Pr[n_r = k,\ n] = P(0)\ \frac{n!}{(n-k)!}\ \frac{\rho_r^k}{k!} \sum_{\substack{\sum_{i\neq r} n_i = n-k \\ n_i \geq 0}} (n-k)! \prod_{\substack{i=1 \\ i\neq r}}^{R} \frac{\rho_i^{n_i}}{n_i!}$$

or equivalently

$$\Pr[n_r = k,\ n] = P(0) \binom{n}{k} \rho_r^k \left(\sum_{\substack{i=1 \\ i\neq k}}^{R} \rho_i \right)^{n-k}$$

Notice that the (R - 1) classes are lumped into a single class with input rate $\sum_{i=1, 1\neq k}^{R} \lambda_r$. Using the facts that

$$\rho = \sum_{i=1}^{R} \rho_i$$

and

$$\Pr[n_r = k] = \sum_{n=k}^{B} \Pr[n_r = k,\ n]$$

we get

$$\Pr[n_r = k] = P(0) \sum_{n=k}^{B} \left[\binom{n}{k} \rho_r^k (\rho - \rho_r)^{n-k} \right] \qquad \text{(D.6)}$$

The total average number of messages in the system is

$$\bar{n} = \frac{\rho}{1-\rho}\ \frac{1 - (B+1)\rho^B + B\rho^{B+1}}{1 - \rho^{B+1}} \qquad \text{(D.7)}$$

and by using Little's result [LITT 61], the average time spent in the system for the accepted messages is found to be

$$T = \frac{1/\mu C}{1-\rho} \; \frac{1 - (1+B)\rho^B + B\rho^{B+1}}{1-\rho^B} \tag{D.8}$$

For the marginal statistics we have the following:

$$\bar{n}_r = E[\text{number of class r messages in the system}] \tag{D.9}$$

$$\bar{n}_r = \frac{\lambda_r}{\lambda}\,\bar{n}$$

and

$$T_r = T \tag{D.10}$$

Special Case I: $\eta\lambda_r = \lambda_r^o$, $r = 1, 2, \ldots, R$, and $\eta \uparrow \infty$.

We have the following results:

$$\lim_{\eta\uparrow\infty} \Pr[\underline{n}] = \begin{cases} 0 & \text{if } n = \sum n_r \neq B \\ \dfrac{B!}{(\rho^o)^B} \displaystyle\prod_{r=1}^{R} \frac{(\rho_r^o)^{n_r}}{n_r!} & \text{if } n = \sum n_r = B \end{cases} \tag{D.11}$$

where $\rho^o = \sum \lambda_r^o$. We also have

$$\lim_{\eta\uparrow\infty} PB = \lim_{\eta\uparrow\infty} \Pr[\sum n_r = B] = 1$$

also

$$\lim_{\eta\uparrow\infty} \Pr[n_r = k] = \binom{B}{k}\left(\frac{\rho_r^o}{\rho^o}\right)^k \left(1 - \frac{\rho_r^o}{\rho^o}\right)^{B-k} \tag{D.12}$$

Further, we have

$$\lim_{\eta\uparrow\infty} \Pr[n_r = 0] = \left(1 - \frac{\rho_r^o}{\rho^o}\right)^B \neq 0 \tag{D.13}$$

As for the limiting throughput, we have

$$\lim_{\eta \uparrow \infty} \lambda' = \lim_{\eta \uparrow \infty} \lambda(1 - PB) = \mu C \tag{D.14}$$

$$\lim_{\eta \uparrow \infty} \lambda_r' = \lim_{\eta \uparrow \infty} \lambda_r'(1 - PB) = \frac{\lambda_r^o}{\lambda^o} \mu C \qquad r = 1, 2, \ldots, R \tag{D.15}$$

$$\lim_{\eta \uparrow \infty} T_r = B\mu C \tag{D.16}$$

Eq. (D.13) reveals an interesting fact; namely, even if the input rates are scaled up to infinity, there is a non-zero probability that one (or more) of the classes of messages are absent from the system. This is a distinct drawback of the CS scheme.

Special Case II: $\lambda_i \uparrow \infty$

From Eqs. (D.4) - (D.6) we can get the following results:

$$\lim_{\lambda_i \uparrow \infty} \Pr[\textstyle\sum n_r = n] = \begin{cases} 0 & n \neq B \\ 1 & n = B \end{cases} \tag{D.17}$$

$$\lim_{\lambda_i \uparrow \infty} [n_r = k] = \begin{cases} 1 & k = 0 \\ 0 & k \neq 0 \end{cases}, \quad r \neq i \tag{D.18}$$

$$\lim_{\lambda_i \uparrow \infty} \Pr[n_i = k] = \begin{cases} 0 & k \neq B \\ 1 & k = B \end{cases} \tag{D.19}$$

$$\lim_{\lambda_i \uparrow \infty} \lambda_r' = \lim_{\lambda_i \uparrow \infty} \lambda_r(1 - PB) = \begin{cases} 0 & r \neq i \\ \mu C & r = i \end{cases} \tag{D.20}$$

These results show how the system can be dominated by a single class of high-rate messages.

D.2 Complete Partitioning (CP)

The set of possible states will be

$$F_2 = \{\underline{n} \mid 0 \leq n_r \leq a_r \ , \quad r = 1, 2, \ldots, R\} \tag{D.21}$$

and we have $B = \sum a_r$.

In this case the computation of the normalization term C_2 is not as simple as C_1 and in fact no explicit expression for this constant has been derived. Notice that the summation in

$$C_2^{-1} = \sum_{\underline{n} \in F_2} \left[n! \prod_{r=1}^{R} \frac{\rho_r^{n_r}}{n_r!} \right] \ , \quad n = \sum n_r$$

is taken over $\sum_{n=0}^{B} \binom{R+n-1}{n} = \binom{R+B}{B}$ possible system states and a direct summation is out of the question. Unfortunately, because of the n! term we cannot even use the generating function approach as in [MOOR 72], [WILL 74] or [KAMO 76B], and we must resort to numerical techniques. These techniques have been studied by others and interested readers can refer to [BUZE 73], [KOBA 75], [WONG 75] and [REIS 76A and 76B]. However, the problem at hand is different from theirs and we must modify their techniques.

We define the following set of states:

$$S(n,m) \triangleq \{\underline{n} = (n_1, n_2, \ldots, n_m) \mid \sum_{r=1}^{m} a_r = n, \ 0 \leq n_r \leq a_r, \ r = 1, 2, \ldots, m\} \tag{D.22}$$

S(n,m), as defined above, is the set of all system states in which only the first m classes may be present in the system and the total number of

messages in the system is n. Notice that $F_2 = \bigcup_{n=0}^{B} S(n,R)$ and we have

$$C_2^{-1} = \sum_{n=0}^{B} \left[\sum_{\underline{n} \in S(n,R)} n! \prod_{r=1}^{R} \frac{\rho_r^{n_r}}{n_r!} \right]$$

We also define an auxiliary function u(n,m) as follows

$$u(n,m) \triangleq \begin{cases} \sum_{\underline{n} \in S(n,m)} \left[n! \prod_{r=1}^{m} \frac{\rho_r^{n_r}}{n_r!} \right] & 0 \leq n \leq a_m^* \\ 0 & n > a_m^* \end{cases} \qquad (D.23)$$

where $a_m^* \triangleq \sum_{r=1}^{m} a_r$, and clearly we have $B = a_R^*$. C_2 is related to u(n,m) according to the following equation:

$$C_2^{-1} = P(0)^{-1} = \sum_{n=0}^{B} u(n,R) \qquad (D.24)$$

We also have

$$\Pr\left[\sum_{r=1}^{R} n_r = n \right] = P(0)u(n,R) \qquad (D.25)$$

In what follows, we will present an algorithm to calculate iteratively u(0,R), u(1,R), ..., u(B,R). Following the approach of [BUZE 73], we have

$$u(n,m) = \sum_{k=0}^{n} \left[\sum_{\substack{\underline{n} \in S(n,m) \\ n_m = k}} u(n,m) \right]$$

However, because $\underline{n} \in S(n,m)$ and $n_m = k$, then $k \leq a_m^*$ and we have

$$u(n,m) = \sum_{k=0}^{\lfloor n, a_m \rfloor} \left[\sum_{\substack{\underline{n} \in S(n,m) \\ n_m = k}} u(n,m) \right]$$

where $\lfloor a,b \rfloor = \inf\ (a,b)$. Factoring out the constant terms, we observe that for $m > 1$

$$u(n,m) = \sum_{k=0}^{\lfloor n, a_m \rfloor} \left\{ n! \frac{\rho_m^k}{k!} \left[\sum_{\underline{n} \in S(n-k,m-1)} \prod_{r=1}^{m-1} \frac{\rho_r^{n_r}}{n_r!} \right] \right\}$$

or

$$u(n,m) = \sum_{k=0}^{\lfloor n, a_m \rfloor} \left[\binom{n}{k} \rho_m^k u(n-k, m-1) \right]$$

However, by definition (Eq. (D.23)), we should have $(n-k) \leq a^*_{m-1}$ or $k \geq (n - a^*_{m-1})$ and we have for $m > 1$

$$u(n,m) = \begin{cases} \sum_{k=\lceil 0, n-a^*_{m-1} \rceil}^{\lfloor n, a_m \rfloor} \left[\binom{n}{k} \rho_m^k u(n-k,m-1) \right] & 0 \leq n \leq a^*_m \\ 0 & n > a^*_m \end{cases} \qquad \text{(D.26)}$$

and for $m = 1$

$$u(n,1) = \rho_1^n \qquad 0 \leq n \leq a_1 \qquad \text{(D.27)}$$

and

$$u(0,m) = 1 \qquad 1 \leq m \leq R \qquad \text{(D.28)}$$

Eqs. (D.26 - D.28) provide us with an iterative scheme to calculate $u(n,R)$. One must first initialize the first column and row of matrix u

according to Eqs. (D.27) and (D.28) and each element of columns 2 to R is calculated according to Eq. (D.26).

Having calculated the matrix u, we can find expressions for other statistics of interest. C_2 and $Pr[\sum n_r = n]$ have been already defined in Eqs. (D.24) and (D.25). For the total average number of messages in the system we have

$$\bar{n} = P(0) \sum_{n=1}^{B} u(n,R)n \tag{D.29}$$

We can find the marginal distribution of the message class of the last column of the matrix u (class R). From Eq. (D.1) we have

$$Pr[n_R = k] = \sum_{\substack{\underline{n}\in F_2 \\ n_R=k}} P(0)n! \prod_{r=1}^{R} \frac{\rho_r^{n_r}}{n_r!}$$

When the number of class R messages in the system is k, the total number of messages in the system is at least k and at most $(a^*_{R-1} + k)$. Using the definition of S(n,R) we have

$$Pr[n_R = k] = \sum_{n=k}^{k+a^*_{R-1}} \sum_{\substack{\underline{n}\in S(n,R) \\ n_R=k}} P(0)n! \prod_{r=1}^{R} \frac{\rho_r^{n_r}}{n_r!}$$

After factoring out the constant terms and some rearrangments we get

$$Pr[n_R = k] = P(0) \sum_{n=k}^{k+a^*_{R-1}} \sum_{\substack{\underline{n}\in S(n,R) \\ n_R=k}} \left[\frac{n!\rho_R^k}{k!(n-k)!} (n-k)! \prod_{r=1}^{R-1} \frac{\rho_r^{nr}}{n_r!} \right]$$

which can be reduced to

$$\Pr[n_R = k] = P(0) \sum_{n=k}^{k+a^*_{R-1}} \left\{ \binom{n}{k} \rho_R^k \left[\sum_{\underline{n} \in S(n-k, R-1)} (n-k)! \prod_{r=1}^{R-1} \frac{\rho_r^{n_r}}{n_r!} \right] \right\}$$

Finally, using $u(n-k, R-1)$ as defined in Eq. (D.23) we get

$$\Pr[n_R = k] = P(0) \sum_{n=k}^{k+a^*_{R-1}} \left[\binom{n}{k} u(n-k, R-1) \right] \rho_R^k \qquad (D.30)$$

In particular the probability of being absent from the system is

$$\Pr[n_R = 0] = P(0) \sum_{n=0}^{a^*_{R-1}} [u(n, R-1)] \qquad (D.31)$$

and the probability of blocking of the R^{th} class will be

$$PB_R = \Pr[n_R = a_R] = P(0) \sum_{n=a_R}^{B} \left[\binom{n}{a_R} u(n-a_R, R-1) \right] \rho_R^{a_R} \qquad (D.32)$$

From Eq. (D.30) we can find other measures of interest such as throughput of the R^{th} class, λ'_R; average number of class R messages in the system, n_R, etc. In order to find statistics of other classes of messages, we have to rearrange the ordering of classes and position the class of interest as the R^{th}.

Special Case I: $\lambda_r = \eta \lambda_r^o$, $r = 1, 2, \ldots, R$ and $\eta \uparrow \infty$

We have the following limiting statistics:

$$\lim_{\eta \uparrow \infty} \Pr[\textstyle\sum n_r = n] = \begin{cases} 0 & \text{if } n \neq B \\ 1 & \text{if } n = B \end{cases} \qquad (D.33)$$

and

$$\lim_{\eta \uparrow \infty} \Pr[n_r = k] = \begin{cases} 0 & \text{if } k \neq a_r \\ 1 & \text{if } k = a_r \end{cases} \qquad (D.34)$$

Comparing Eq. (D.34) with (D.13) we notice that the CS scheme does not display this property.

For the limiting throughput we have

$$\lim_{\eta \uparrow \infty} \lambda' = \mu C \tag{D.35}$$

$$\lim_{\eta \uparrow \infty} \lambda' = \lim_{\eta \uparrow \infty} \lambda_r (1 - PB_r) = \frac{a_r}{B} \mu C \tag{D.36}$$

which shows that in the limit, each class will use a portion of the output channel equal to the fraction of the storage that it is using.

Special Case II: $\lambda_i \uparrow \infty$

In contrast to the CS scheme in which all of the other classes would be denied any service, the limiting throughput of other classes is not zero. The expressions for the limiting probabilities of blocking and other measures are in general very complicated; we derive these expressions for the case R = 2. Consider a storage of size B with two input classes with (normalized) rates ρ_1 and ρ_2, and let a_1 and a_2 be the allocated number of buffers to these two classes ($a_1 + a_2 = B$). The probability of blocking of each class can be found by using Eq. (D.30).

$$PB_i = \frac{\sum_{n=a_i}^{B} \binom{n}{a_i} \rho_i^{a_i} \rho_j^{n-a_i}}{\sum_{n=0}^{B} \sum_{k=\lceil 0, n-a_1 \rceil}^{\lfloor n, a_2 \rfloor} \binom{n}{k} \rho_2^k \rho_1^{n-k}} \qquad \begin{matrix} i,j = 1,2 \\ i \neq j \end{matrix}$$

Assume that λ_1 is kept fixed and $\lambda_2 \uparrow \infty$. In the limit $n_2 = a_2$ and we have

$$\lim_{\lambda_2 \uparrow \infty} PB_2 = 1$$

The limit of PB_1 and $\lambda_2 \uparrow \infty$ will be

$$\lim_{\lambda_2 \uparrow \infty} PB_1 = \lim_{\lambda_2 \uparrow \infty} \frac{\sum_{n=a_1}^{B} \binom{n}{a_1} \rho_1^{a_1} \rho_2^{n-a_1}}{\sum_{n=0}^{B} \sum_{k=\lceil 0, n-a_1 \rceil}^{\lfloor n, a_2 \rfloor} \binom{n}{k} \rho_2^{k} \rho_1^{n-k}}$$

$$= \lim_{\lambda_2 \uparrow \infty} \frac{\binom{n}{a_2} \rho_1^{a_1} \rho_2^{a_2}}{\sum_{n=a_2}^{B} \binom{n}{a_2} \rho_2^{a_2} \rho_1^{n-a_2}}$$

Therefore we have

$$\lim_{\lambda_2 \uparrow \infty} PB_r = \begin{cases} \dfrac{\binom{B}{a_1} \rho_1^{B}}{\sum_{n=a_2}^{B} \binom{n}{a_2} \rho_1^{n}} & r = 1 \\ 1 & r = 2 \end{cases} \qquad \text{(D.37)}$$

The limiting throughput of class 1 is

$$\lambda_1' = \lambda_1 (1 - PB_1)$$

Because the input rate of class 2 is infinite, we have

$$\lambda_2' = \mu C - \lambda_1'$$

Therefore we have

$$\lim_{\lambda_2 \to \infty} \lambda_r' = \begin{cases} \dfrac{\sum_{n=a_2}^{B-1} \binom{n}{a_1} \rho_1^{n+1}}{\sum_{n=a_2}^{B} \binom{n}{a_1} \rho_1^{n}} \mu C & r = 1 \\ \dfrac{\rho_1^{a_2} + \sum_{n=a_{2+1}}^{B} \binom{n-1}{a_2-1} \rho_1^{n}}{\sum_{n=a_2}^{B} \binom{n}{a_2} \rho_1^{n}} \mu C & r = 2 \end{cases} \qquad \text{(D.38)}$$

The above expressions show that if one of the input rates grows to infinity it does not prevent the other class from using the system. This was not the case for the CS scheme.

D.3 Sharing with Maximum Queue Length (SMXQ)

As we explained in Section 6.3, in SMXQ a pool of buffers is shared by different classes of messages in such a way that there is a limit on the number of messages of each class which can be present in the system simultaneously. The set of feasible states is

$$F_3 = \{n \mid \sum_{r=1}^{R} n_r \leq B \quad ; \quad 0 \leq n_r \leq b_r, \quad r = 1, 2, \ldots, R\} \qquad \text{(D.39)}$$

where b_r is the maximum queue length for class r messages. Clearly, if $\sum b_r \leq B$ this scheme reduces to CP, and if $b_r \geq B$, $r = 1, 2, \ldots, R$, it will be identical to CS.

To calculate the normalization constant C_3 and other statistics as we did for the CP scheme, we must resort to numerical techniques. The algorithm for this calculation, except for minor differences, is

similar to the one for the CP scheme which we shall briefly explain.

The sets S(n,m) and the auxiliary function u(n,m) will be defined as follows:

$$S(n,m) \triangleq \{\underline{n} = (n_1, n_2, \ldots, n_m) \mid \sum_{r=1}^{m} n_r = n,\ 0 \le n_r \le b_r,\ r = 1, 2, \ldots, m\} \tag{D.40}$$

$$u(n,m) \triangleq \sum_{\underline{n} \in S(n,m)} \left[n! \prod_{r=1}^{m} \frac{\rho_r^{n_r}}{n_r!} \right], \quad n \le b_m^* \tag{D.41}$$

where $b_m^* = \sum_{r=1}^{m} b_r$.

By using an approach very similar to CP, we have for $m > 1$

$$u(n,m) = \begin{cases} \displaystyle\sum_{k=\lceil 0, n-\lfloor B, a_{m-1}^* \rfloor \rceil}^{\lfloor n, b_m \rfloor} \left[\binom{n}{k} \rho_m^k u(n-k, m-1) \right] & n \le \lfloor B, b_m^* \rfloor \\ 0 & n > \lfloor B, b_m^* \rfloor \end{cases} \tag{D.42}$$

and for $m = 1$

$$u(n,1) = \rho_1^n \qquad 0 \le n \le \lfloor B, b_1 \rfloor \tag{D.43}$$

$$u(0,m) = 1 \qquad m = 1, 2, \ldots, R \tag{D.44}$$

where as before, $\lceil a, b \rceil = \inf(a,b)$ and $\lfloor a, b \rfloor = \sup(a,b)$

Having calculated the matrix u, we can find the statistics of interest. In particular

$$C_3^{-1} = P(0)^{-1} = \sum_{n=0}^{B} u(n,R) \tag{D.45}$$

$$\Pr\left[\sum_{r=1}^{R} n_r = n\right] = P(0)u(n,R) \tag{D.46}$$

$$\bar{n} = E[\text{number of messages in the system}] = P(0) \sum_{n=1}^{B} u(n,R)n \tag{D.47}$$

For the marginal statistics of class R we have

$$\Pr[n_R = k] = \frac{\sum_{n=k}^{\lfloor B, b^*_{R-1}+k \rfloor} \left[\binom{n}{k} u(n-k, R-1)\right] \rho_R^k}{\sum_{n=0}^{B} u(n,R)} \qquad 0 \le k \le \lfloor B, b_R \rfloor \tag{D.48}$$

In particular

$$\Pr[n_R = 0] = \frac{\sum_{n=0}^{\lfloor B, b^*_{R-1} \rfloor} u(n, R-1)}{\sum_{n=0}^{B} u(n,R)} \tag{D.49}$$

The blocking probability of class R messages is the probability that either the entire storage is full, or n_R is equal to b_R. This leads to

$$PB_R = \frac{u(B,R) + \sum_{n=b_R}^{\lfloor B-1, b^*_R \rfloor} \left[\binom{n}{b_R} u(n-b_R, R-1)\right] \rho_R^{b_R}}{\sum_{n=0}^{B} u(n,R)}$$

In order to find marginal statistics of other classes, we must rearrange the ordering of the classes. Now we consider two special cases:

Special Case I: $\lambda_r = \eta\lambda_r^o$, $r = 1, 2, \ldots, R$ and $\eta \uparrow \infty$

In this case we have

$$\lim_{\eta\uparrow\infty} PB_r = 1 \qquad r = 1, 2, \ldots, R$$

$$\lim_{\eta\uparrow\infty} Pr[n_R = k] = \begin{cases} 0 & \text{for} \quad 0 \leq k < B - b_{R-1}^* \\ \neq 0 & \text{for} \quad k \geq B - b_{R-1}^* \end{cases} \qquad \text{(D.51)}$$

In particular

$$\lim_{\eta\uparrow\infty} Pr[n_R = 0] = 0 \quad \text{if} \quad B > b_R$$

This shows that in order to guarantee that at infinite input rate ($\eta \uparrow \infty$) there is always one or more packets of a certain class of messages, say r, present in the system, we should have

$$B > \sum_{\substack{i=1 \\ i\neq r}}^{B} b_i$$

This is one of the motivations behind using the SMXQ scheme. For the throughput of the class R messages we have for $b_R^* > B$

$$\lim_{\uparrow\infty} \lambda_R' = \frac{h(B-1, r) - \binom{B-1}{b_R} h(B-1-b_R, R-1)\, \rho_R^{o\,b_R}}{h(B,R)} \lambda_R^o \qquad \text{(D.52)}$$

where $\rho_i^o = \lambda_i^o/\mu C$ and $h(n,m) = u(n,m)\Big|_{\rho_i=\rho_i^o}$; and for $B > b_R^*$

$$\lim_{\eta\uparrow\infty} \lambda_r' = \frac{b_r}{b_R^*} \mu C \qquad \text{(D.53)}$$

which is the same as for the CP scheme.

Special Case II: $\lambda_i \uparrow \infty$

If $b_i \geq B$, then intuitively

$$\lim_{\lambda_i \uparrow \infty} PB_r = 1 \qquad r = 1, 2, \ldots, R$$

and

$$\lim_{\lambda_i \uparrow \infty} \lambda_r' = \begin{cases} 0 & r \neq i \\ \mu C & r = i \end{cases} \tag{D.54}$$

This behavior is similar to the CS scheme. If, however, $b_i < B$, then

$$\lim_{\lambda_i \uparrow \infty} PB_r = \begin{cases} 1 & r = i \\ < 1 & r \neq i \end{cases} \tag{D.55}$$

In fact, b_i buffers will be occupied by class i messages with probability one and the $B - b_i$ remaining buffers will be shared by the other classes (according to the SMXQ scheme). The expressions for the limiting probabilities are complicated and we only present the expressions for the case, R = 2.

$$\lim_{\lambda_2 \uparrow \infty} PB_r = \begin{cases} \dfrac{\binom{B}{b_1} \rho_1^B}{\sum_{n=b_2}^{B} \binom{n}{b_2} \rho_1^n} & r = 1 \\ 1 & r = 2 \end{cases} \tag{D.56}$$

and

$$\lim_{\lambda_2 \uparrow \infty} \lambda_r' = \begin{cases} \dfrac{\sum_{n=b_2}^{B-1} \binom{n}{b_1} \rho_1^{n+1}}{\sum_{n=b_2}^{B} \binom{n}{b_2} \rho_1^{n}} \mu C & r = 1 \\ \dfrac{\rho_1^{b_2} + \sum_{n=b_2+1}^{B} \binom{n-1}{b_2-1} \rho_1^{n}}{\sum_{n=b_2}^{B} \binom{n}{b_2} \rho_1^{n}} \mu C & r = 2 \end{cases} \tag{D.57}$$

Comparing Eqs. (D.56) and (D.57) with Eqs. (D.37) and (D.38), we notice that in this case the behavior is like the CP scheme when a pool of B buffers is partitioned into $(B - b_2)$ and b_2 buffers.

D.4 Sharing with Minimum Allocation (SMA)

As Eq. (D.51) suggests, in order to guarantee that at infinite input rates one or more packets of a certain class of messages will always be in the system, at least one buffer storage must be permanently allocated to that class of messages. This is the idea behind the SMA scheme in which, out of a pool of B buffers, a_r $(r = 1, 2, \ldots, R)$ buffers are permanently allocated to class r messages, and the remaining $B_s = B - \sum a_r$ buffers, similar to the CS scheme, are shared by all of the classes. As a result, the set of possible states will be

$$F_4 = \{\underline{n} = (n_1, n_2, \ldots, n_r) \mid \sum_{r=1}^{R} \lceil 0, n_r - a_r \rceil \leq B_s;\ 0 \leq n_r \leq B_s + a_r,\ r = 1, 2, \ldots, R\} \tag{D.58}$$

where $B_s = B - \sum a_r$ and $\lceil a, b \rceil = \sup(a, b)$.

Calculation of the normalization factor C_4 is more complex than for the previous schemes as we will see shortly. We start with ordering the classes in an arbitrary manner and define the following sets:

$$T(b,n,m) \triangleq \Big\{ \underline{n} = (n_1, n_2, \ldots, n_m) \mid \sum_{r=1}^{m} n_r = n ;$$

$$\sum_{r=1}^{m} \lceil 0, n_r - a_r \rceil = b ; \quad 0 \le n_r \le b + a_r , \quad r = 1, 2, \ldots, m \Big\}$$

where $0 \le b \le B_s$, $0 \le n \le B$, $1 \le m \le R$ (D.59)

T(b,n,m) is the set of states in which there are no messages of classes m+1, m+2, ..., R in the system, the total number of messages in the system is n, and the number of messages in the shared storage is b. Instead of a two-dimensional matrix u, we must use a three-dimensional matrix

$$t(b,n,m) \triangleq \sum_{\underline{n} \in T(b,n,m)} \left[n! \prod_{r=1}^{m} \frac{\rho_r^{n_r}}{n_r!} \right] \qquad n = \sum_{r=1}^{m} n_r \qquad (D.60)$$

To derive a general expression for t(b,m,m) as we did for SMXQ, we start with conditioning on the number of class m messages in the system

$$t(b,n,m) = \sum_{k=0}^{n} \left\{ \sum_{\substack{\underline{n} \in T(b,n,m) \\ n_m = k}} \left[n! \sum_{r=1}^{m} \frac{\rho_r^{n_r}}{n_r!} \right] \right\}$$

Factoring out the constant terms, we get

$$t(b,n,m) = \sum_{k=0}^{n} \left\{ \sum_{\substack{\underline{n} \in T(b,n,m) \\ n_m = k}} \left[n! \frac{\rho_m^k}{k!} \prod_{r=1}^{m-1} \frac{\rho_r^{n_r}}{n_r!} \right] \right\}$$

The inner summation should be taken over all states in which the number of messages in the system is n and the number of messages in the shared buffers is b, and when there are k messages of class m in the system. The above equation can be rearranged as follows:

$$t(b,n,m) = \sum_{k=0}^{\lfloor n, a_m \rfloor} \left\{ \frac{n!\rho_m^k}{k!} \sum_{\underline{n} \epsilon T(b,n-k,m-1)} \left[\prod_{r=1}^{m-1} \frac{\rho_r^{n_r}}{n_r!} \right] \right\}$$

$$+ \sum_{k=1}^{\lfloor n-a_m, b \rfloor} \left\{ \frac{n!\rho_m^k}{k!} \sum_{\underline{n} \epsilon T(b-k,n-(k+a_m),m-1)} \left[\prod_{r=1}^{m-1} \frac{\rho_r^{n_r}}{n_r!} \right] \right\}$$

In the first summation there are $k \leq a_m$ class m messages in the system, hence there are no messages of class m in the shared buffer. In the second summation there are $k + a_m$ class m messages in the system, hence there are k (> 0) class m messages in the shared buffer. After some algebra, and using the definition of t(b,n,m), we can derive the following expression for the element of matrix t.

For $1 \leq m \leq R$

$$t(b,n,m) = \begin{cases} \sum_{k=\lceil 0, n-b-a^*_{m-1} \rceil}^{\lfloor n, a_m \rfloor} \left[\binom{n}{k} \rho_m^k \, u(b, n-k, m-1) \right] \\ + \rho_m^{a_m} \sum_{k=1}^{\lfloor n-a_m, b \rfloor} \left[\binom{n}{k+a_m} \rho_m^k \, u(b-k,\ n-(k+a_m),\ m-1) \right] \\ \qquad\qquad \text{if } n \leq b + a^*_m \\ 0 \quad \text{if } n > b + a^*_m \end{cases} \tag{D.61}$$

and

$$t(b,0,m) = \begin{cases} 1 & b = 0 \\ 0 & b > 0 \end{cases} \tag{D.62}$$

$$t(0,n,1) = \begin{cases} \rho_1^n & 0 \leq n \leq a_1 \\ 0 & n > a_1 \end{cases} \tag{D.63}$$

$$t(b,n,1) = \begin{cases} 0 & n \neq b + a_1 \\ \rho^n & n = b + a_1 \end{cases} \quad b > 0 \tag{D.64}$$

Using the matrix t, we can find values of interest.

$$C_4^{-1} = P(0)^{-1} \sum_{b=0}^{B_s} \sum_{n=0}^{b+a_R^*} t(b,n,R) \tag{D.65}$$

$$Pr[\sum n_r = n] = P(0) \sum_{b=0}^{B_s} t(b,n,R) \tag{D.66}$$

For the marginal distribution of class R we have

$$Pr[n_R = k] = P(0) \sum_{b=\lceil 0,k-a_R \rceil}^{B_s} \sum_{n=k}^{a_{R-1}^*+b-\lceil 0,k-a_R \rceil} \left[\binom{n}{k} \rho_R^k \, t(b-\lceil 0,k-a_R \rceil, n-k, R-1) \right] \tag{D.67}$$

In particular

$$Pr[n_R = 0] = P(0) \sum_{b=0}^{B_s} \sum_{n=0}^{b+a_{R-1}^*} t(b,n,R-1) \tag{D.68}$$

The probability of blocking of class R is the probability that the shared storage is full and that $n_R \geq a_R$. This gives us

$$PB_R = P(0) \left\{ \sum_{k=a_R}^{B_s+a_R} \rho_R^k \sum_{n=k}^{B-k} \left[\binom{n}{k} u(B_s - (k-a_R),\ n-k,\ R-1) \right] \right\} \tag{D.69}$$

Expressions for the limiting probabilities when $\lambda_r = \eta\lambda_r^o$, $r = 1, \ldots, R$ and $\eta \uparrow \infty$ are very complex and are not presented here. We can show that

$$\lim_{\eta\uparrow\infty} PB_r = 1, \qquad r = 1, 2, \ldots, R$$

and

$$\lim_{\eta\uparrow\infty} \Pr[n_r = k] = \begin{cases} 0 & 0 \leq k \leq a_r \\ \neq 0 & a_r \leq k \leq B_s + a_r \end{cases}$$

For the special case that one of the input rates grows to infinity we can show that

$$\lim_{\lambda_i\uparrow\infty} \Pr[n_r = k] = \begin{cases} > 0 & 0 \leq k \leq a_r \qquad r \neq i \\ = 0 & k > a_r \qquad r \neq i \end{cases}$$

and

$$\lim_{\lambda_i\uparrow\infty} \Pr[n_i = k] = \begin{cases} 0 & \text{if} \quad k \neq B_s + a_i \\ 1 & \text{if} \quad k = B_s + a_i \end{cases}$$

D.5 Sharing with Minimum Allocation and Maximum Queue (SMAMXQ)

In this scheme, as in the SMA scheme, a minimum number of buffers (a_r, $r = 1, 2, \ldots, R$) are allocated to each class and there is also a limit (b_r, $r = 1, 2, \ldots, R$) on the number of messages of each class which can be present in the shared storage. The set of feasible states is

$$F_5 = \{\underline{n} \mid \sum_{r=1}^{R} \lceil 0, n_r - a_r \rceil \leq B_s ; \; 0 \leq n_r \leq a_r + b_r, \; r = 1, 2, \ldots, R\} \tag{D.70}$$

The analysis of this scheme is quite identical to the SMA scheme. We start by defining the following sets for any ordering of the classes of messages.

$$T(b,n,m) \triangleq \left\{ \underline{n} = (n_1, n_2, n_3, \ldots, n_a) \mid \sum_{r=1}^{m} n_r = n, \quad \sum_{r=1}^{m} \lceil 0, n_r - a_r \rceil = b; \right.$$
$$\left. 0 \leq n_r \leq a_r + \lfloor b_r, b \rfloor , \quad r = 1, 2, \ldots, m \right\} \tag{D.71}$$

As in SMA, T(b,n,m) is the set of states in which there are no messages of classes m+1, m+2, ..., R in the system, and there are n messages in the system, of which b are in the shared buffer. We also define a three-dimensional matrix t as follows:

$$t(b,n,m) \triangleq \sum_{\underline{n} \in T(b,n,m)} \left[n! \prod_{r=1}^{m} \frac{\rho_r^{n_r}}{n_r!} \right] \quad n = \sum_{r=1}^{m} n_r \tag{D.72}$$

With an approach similar to SMA we can find the expression for the values of the matrix t

$$t(b,n,m) = \begin{cases} \displaystyle\sum_{k=\lceil 0, n-a^*_{m-1} - \lfloor b, b^*_{m-1} \rfloor \rceil}^{\lfloor n, a_m \rfloor} \left[\binom{n}{k} \rho_m^k \, t(b, n-k, m-1) \right] \\ \quad + \rho_m^{a_m} \displaystyle\sum_{k=1}^{\lfloor n-a_m, b, b_m \rfloor} \left[\binom{n}{k + a_m} \rho_m^k \, t(b-k, n-(k+a_m), m-1) \right] \\ \qquad\qquad \text{if } n \leq a^*_m + \lfloor b, b^*_m \rfloor \\ 0 \qquad \text{if } n > a^*_m + \lfloor b, b^*_m \rfloor \end{cases} \tag{D.73}$$

and

$$t(b,0,m) = \begin{cases} 1 & b = 0 \\ 0 & b > 0 \end{cases}$$

$$t(0,n,1) = \begin{cases} \rho_1^n & 0 \leq n \leq a_1 \\ 0 & n > a_1 \end{cases}$$

$$t(b,n,1) = \begin{cases} 0 & n \neq b + a_1 \\ \rho^n & n = b + a_1 \end{cases} \qquad b > 0$$

Having found matrix t, we can calculate the performance measures of interest. The expressions for these measures are exactly similar to the ones for the SMA scheme and we do not present them here.

A summary of the results presented in this appendix was originally published in [KERM 77].

BIBLIOGRAPHY

ABRA 70 Abramson, N. "The ALOHA System - Another Alternative for Computer Communications," AFIPS Conference Proceedings, Fall Joint Computer Conference, Las Vegas, Nevada, November 1970, Vol. 37, pp. 281-285.

ABRA 73 Abramson, N. "Packet Switching With Satellites," AFIPS Conference Proceedings, National Computer Conference, New York, June 1973, Vol. 42, pp. 695-702.

AGNE 74 Agnew, C.E. "Dynamic Modeling and Control of Congestion-Prone Systems," Department of Engineering-Economy Systems, Stanford University, Stanford, California, Technical Report 6, January 1974.

BARA 64 Baran, P. "On Distributed Communications," Rand Corporation, Santa Monica, California, Rand Series Report, August 1964.

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

BBN 73 Bolt Beranek and Newman Inc. "Specification for the Interconnection of a Host and an IMP," Bolt Beranek and Newman Inc., Cambridge, Mass., BBN Report 1882 (revised), April 1973.

BELS 75 Belsnes, D. "Flow Control in the Packet Switching Networks," Communications Networks, D. Barber, Ed., Online Conference Limited, London, 1975, pp. 349-361.

BLAC 62 Blackwell, D. "Discrete Dynamic Programming," Annals of Mathematical Statistics, Vol. 33, pp. 719-726, 1962.

BUZE 73 Buzen, J.P. "Computational Algorithms for Closed Queueing Networks with Exponential Servers," Communications of the ACM, Vol. 16, No. 9, pp. 527-531, September 1973.

CARR 70 Carr, S., S. Crocker, and V. Cerf. "HOST-HOST Communication Protocol in the ARPA Network," AFIPS Conference Proceedings, Spring Joint Computer Conference, Atlantic City, New Jersey, May 1970, Vol. 36, pp. 589-597.

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

CHAT 76 Chatterjee A., N.D. Georganas and P.K. Verma. "Analysis of a Packet-Switched Network with End-to-End Congestion Control and Random Routing," International Conference on Computer Communications Proceedings, Toronto, Ontario, Canada, August 1976, pp. 488-494.

CHEN 75A Chen, F.T., H.D. Chadwick and R.M. Penn. "The DATRAN Network: System Description," Proceedings of IEEE National Telecommunications Conference, New Orleans, Louisiana, December 1975, pp. 15-1 to 15-4.

CHEN 75B Chen, F., Y. Kuzushima, K. Ishii, H. Ishibashi and S. Tajaka. "Digital Multiplexing Hierarchy for an Integrated Data Transmission and Switching System," Proceedings of the National Telecommunications Conference, New Orleans, Louisiana, December 1975, pp. 15-5 to 15-11.

CHOU 76 Chou, W. and M. Gerla. "A Unified Flow and Congestion Control Model for Packet Networks," International Conference on Computer Communications Proceedings, Toronto, Ontario, Canada, August 1976, pp. 475-482.

CHRE 73 Chretien, G.J., W.M. Konig and J.H. Rech. "The SITA Network, Summary Descriptions," Computer Communication Network Conference, University of Sussex, Brighton, U.K., December 1973.

CHU 72 Chu, W.W. "Demultiplexing Considerations for Statistical Multiplexors," IEEE Transaction on Communications, Vol. COM-20, No. 3, pp. 603-609, June 1972.

CHUN 67 Chung, K.L. Markov Chains with Stationary Transition Probabilities," Springer-Verlag, New York, 1967.

CLOS 72A Closs, F. "Message Delay and Trunk Utilization in Line-Switched and Message-Switched Data Networks," Proceedings of First USA-Japan Computer Conference, Tokyo, 1972, pp. 524-530.

CLOS 72B Closs, F. "Time Delay and Trunk Capacity Requirements in Line-Switched and Message-Switched Networks," International Switching Symposium Record, Boston, Massachusetts, 1972, pp. 428-433.

CLOS 73 Closs, F. "Packet Arrival and Buffer Statistics in a Packet Switching Node," Data Networks, Analysis and Design, 3rd Data Communication Symposium, St. Petersburg, Florida, November 1973, pp. 12-17.

CLOW 73 Clowes, G.J. and C.S. Jayasuriya. "Traffic Considerations in Switched Data Networks," Proceedings of the Third IEEE Symposium on Data Analysis and Design, St. Petersburg, Florida, November 13-15, 1973, pp. 18-22.

COFF 71 Coffman, E.G. Jr., M.J. Elphick, and A. Shoshani. "System Deadlocks," Computing Surveys, Vol. 3, No. 2, pp. 67-78, June 1971.

COLE 71 Cole, G.D. "Computer Network Measurements: Techniques and Experiments," Computer Science Department, School of Engineering and Applied Science, University of California, Los Angeles, UCLA-ENG-7165, October 1971.

COX 62 Cox, D.R. Renewal Theory, Butler, Tonner Ltd., Rome and London, 1962.

CROW 75 Crowther, W.R., et al. "Issues in Packet Switching Network Design," AFIPS Conference Proceedings, National Computer Conference, Anaheim, California, 1975, Vol. 44, pp. 161-175.

DANT 75A Danthine, A.A.S. and J. Bremer. "Communication Protocols in a Network Context," ACM SIGCOMM-SIGOPS Interface Workshop on Interprocess Communications, Santa Monica, California, March 1975, pp. 87-92.

DANT 75B Danthine, A.A.S. and L. Eschenauer, "Influence on the Node Behavior of the Node-to-Node Protocol," Inter-Network Working Group (INWG) Protocol Note #22, March 1975.

DAVI 68 Davies, D.W. "The Principles of a Data Communication Network for Computers and Remote Peripherals," IFIP Congress 68, Edinburgh, Scotoand, August 1968, pp. 704-714.

DAVI 71 Davies, D.W. "The Control of Congestion in Packet Switching Networks," Proceedings of the 2nd ACM-IEEE Symposium on Problems of the Optimization of Data Communication, Palo Alto, California, October 1971, pp. 46-49.

DAVI 73 Davies, D.W. and D.L.A. Barber. Communication Networks for Computers, John Wiley & Sons, Inc., London, 1973.

DRUK 75 Drukey, D.L. "Finite Buffers for Purists," TRW Incorporated, Redondo Beach, California, TRW Systems Group Report No. 75.6400-10-97, 1975.

EVAN 67 Evans, J. "Experience Gained from the American Airlines SABRE System Control Program," Proceedings of the ACM National Conference, August 1967, pp. 77-83.

EVER 57 Everett, R.R., C.A. Zraket, and H.D. Benington. "SAGE, A Data-Processing System for Air Defense," Eastern Joint Computer Conference Proceedings, Washington, D.C., December 1957, Vol. 14, pp. 148-155.

FAYO 77 Fayolle, G., E. Gelenbe, G. Pujolle. "An Analytic Evaluation of the Performance of the 'Send-And-Wait' Protocol," IEEE Transactions on Communications, to be published.

FELL 50 Feller, W. An Introduction to Probability Theory and Its Applications, Vol. I, John Wiley & Sons, Inc., New York, 1950.

FISC 76 Fisher, M.J. and T.C. Harris. " A Model for Evaluating the Performance of in Integrated Circuit and Packet Switched Multiplex Structure," IEEE Transactions on Communications, Vol. COM-24, No. 2, pp. 195-202, February 1976.

FORG 75 Forgie, J.W. "Speech Transmission in Packet-Switched Store-and-Forward Networks," AFIPS Conference Proceedings, National Computer Conference, Anaheim, California, May 1975, pp. 137-142.

FRAN 70 Frank, H., I. Frisch and W. Chou. "Topological Considerations in the Design of the ARPA Computer Network," AFIPS Conference Proceedings, Spring Joint Computer Conference, Atlantic City, New Jersey, May 1970, Vol. 36, pp. 581-587.

FRAN 72 Frank, H. and W. Chou. "Topological Optimization of Computer Networks," Proceedings of IEEE, Vol. 60, No. 11, pp. 1385-1387, November 1972.

FUCH 70 Fuchs, E. and P. Jackson, "Estimates of Distributions of Random Variables for Certain Computer Communication Traffic Models," Communications of the ACM, Vol. 13, No. 12, pp. 752-757, December 1970.

FULT 72 Fultz, G.L. "Adaptive Routing Techniques for Message Switching Computer-Communication Networks," Department of Computer Science, School of Engineering and Applied Science, University of California, Los Angeles, UCLA-ENG-7252, July 1972.

GERL 73A Gerla, M. "The Design of Store-and-Forward (S/F) Networks for Computer Communications," Department of Computer Science, School of Engineering and Applied Science, University of California, Los Angeles, UCLA-ENG-7319, January 1973. (Also published as Ph.D. Dissertation).

GERL 73b Gerla, M., W. Chou, and H. Frank. "Computational Considerations and Routing Problems for Large Computer Communication Networks," Proceedings of the National Telecommunication Conference, Atlanta, Georgia, November 1973, pp. 2:28-1 to 28-11.

GERL 78 Gerla, M. and G. DeStasio. "Integration of Packet and Circuit Transport Protocols in the TRAN Data Network," submitted to Network Protocol Symposium to be held in Liege, Belgium, February 1978.

GIES 76 Giessler A., J. Haenle, A. Koenig and E. Pade. "Packet Networks with Deadlock-Free Buffer Allocation, An Investigation by Simulation," Computer Networks, Submitted for publication, 1976.

HARC 75 Harcharik, J.R. "TYMNET, Present and Future," IEEE Eascon Meeting, Washington, D.C., September 1975.

HEAR 70 Heart, F.E., R.E. Kahn, S.M. Ornstein, W.R. Crowther and D.C. Walden. "The Interface Message Processor for the ARPA Computer Network," AFIPS Conference Proceedings, Spring Joint Computer Conference, Atlantic City, New Jersey, May 1970, Vol. 36, pp. 551-567.

HILD 66 Hildebrand, F.B. Introduction to Numerical Analysis, McGraw-Hill, New York, 1966.

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

HOWA 71 Howard, R. Dynamic Probabilistic Systems, Vol. 1: Markov Models and Vol. 2: Semi-Markov and Decision Processes, Wiley, New York, 1971

ITOH 73 Itoh, K., T. Kato, O. Hashida and Y. Yoshida. "An Analysis of Traffic Handling Capacity of Packet Switched and Circuit Switched Networks," Proceedings of the Third Data Communication Symposium, St. Petersburg, Florida, November 1973, pp. 29-37.

JACK 57 J.R. Jackson. "Networks of Waiting Lines," Operations Research, Vol. 5, pp. 518-521, 1957.

JACK 69 Jackson, P.E. and C.D. Stubbs. "A Study of Multiaccess Computer Communications," AFIPS Conference Proceedings, Spring Joint Computer Conference, Boston, May 1969, Vol. 34, pp. 491-504.

JEWE 63 Jewell, W.S. "Markov-Renewal Programming, I and II," Operations Research, Vol. 11, No. 6, pp. 938-971, November-December 1963.

KAHN 71 Kahn, R.E. and W.R. Crowther. "Flow Control in a Resource-Sharing Computer Network," Proceedings of the 2nd ACM-IEEE Symposium on Problems of the Optimization of Data Communication, Palo Alto, California, October 1971, pp. 108-116.

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 76A Kamoun, F. and L. Kleinrock. "Analysis of Shared Storage in a Computer Network Environment," Proceedings of the Ninth Hawaii International Conference on System Science, Honolulu, January 1976, pp. 89-92.

KAMO 76B 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).

KERM 77 Kermani, P. and L. Kleinrock. "Analysis of Buffer Allocation Schemes in a Multiplexing Node," Proceedings of the International Conference on Communications, Chicago, Illinois, June 13-15, 1977, Vol. 2, pp. 30.4-266 to 30.4-275.

KIMB 75 Kimbleton, S.R. and G.M. Schneider. "Computer Communication Networks: Approaches, Objectives, and Performance Considerations," Computing Surveys, Vol. 7, No. 3, pp. 129-173, September 1975.

KLEI 64 Kleinrock, L. Communication Nets: Stochastic Message Flow and Delay, McGraw-Hill, New York, 1964.

KLEI 70 Kleinrock, L. "Analytic and Simulation Methods in Computer Network Design," AFIPS Conference Proceedings, Spring Joint Computer Conference, Atlantic City, New Jersey, May 1970, Vol. 36, pp. 569-579.

KLEI 74A Kleinrock, L. "Resource Allocation in Computer Systems and Computer-Communication Network," IFIP Congress Proceedings, Information Processing 74, Stockholm, Sweden, August 1974, pp. 11-18.

KLEI 74B Kleinrock, L. and W.E. Naylor. "On Measured Behavior of the ARPA Network," AFIPS Conference Proceedings, National Computer Conference, Chicago, May 1974, Vol. 43, pp. 767-780.

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

KLEI 76A Kleinrock, L. and P. Kermani. "Virtual Cut Through," Modeling and Analysis Note #3.0.0, Department of Computer Science, University of California, Los Angeles, January 1976.

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

KOBA 75 Kobayashi, H. and M. Reiser. "Queueing Networks with Multiple Closed Chains: Theory and Computational Algorithms," IBM Journal of Research and Development, Vol. 19, No. 3, pp. 283-294, May 1975.

KUMM 76A Kummerle, K. and H. Rudin. "Packet and Circuit Switching: Cost/Performance Boundaries," IBM Zurich Research Center, Research Report RZ 805 (#27122), November 1976.

KUMM 76B Kummerle, K. and H. Rudin. "Packet and Circuit Switching: A Comparison of Cost and Performance," National Telecommunications Conference Proceedings, Dallas, Texas, November 1976, Vol. III, pp. 42-5.1 to 42.5.7.

LAM 74 Lam, S.S. "Packet Switching in a Multi-Access Broadcast Channel with Applications to Satellite Communication in a Computer Network," Department of Computer Science, School of Engineering and Applied Science, University of California, Los Angeles, UCLA-ENG-7429, March 1974.

LAM 75A Lam, S.S. "Store-and-forward Buffer Requirements in a Packet Switching Network," IBM Thomas J. Watson Research Center, Yorktown Heights, New York, Research Report No. 5432, May 1975. Also IEEE Transactions on Communications, Vol. Com-24, No. 4, pp. 394-403, April 1976.

LAM 75B Lam, S.S. "Queueing Networks with Loss and Triggered Arrivals," IBM Thomas J. Watson Research Center, Yorktown Heights, New York, Research Report RC 5540(#24065), July 1975.

LAM 75C Lam, S.S. and L. Kleinrock. "Dynamic Control Schemes for a Packet Switched Multi-Access Broadcast Channel," AFIPS Conference Proceedings, National Computer Conference, Anaheim, California, May 1975, Vol. 44, pp. 143-154.

LAM 76 Lam, S.S. "Delay Analysis of a Packet-Switched System," National Telecommunications Conference Proceedings, Dallas, Texas, November 1976, Vol. 1, pp. 16-3.1 to 16-3.6.

LELA 76 Le Lann, G. and H. Le Goff, "Advances in Performance Evaluation of Communication Protocols," International Conference on Computer Communication Proceedings, Toronto, Ontario, Canada, August 1976, pp. 361-366.

LITT 61 Little, J. "A Proof of the Queueing Formula L = λW, "Operations Research, Vol. 9, No. 2, pp. 383-387, March 1961.

LUEN 73 Luenberger, D.G. Introduction to Linear and Nonlinear Programming, Addison Wesley Inc., Reading, Massachusetts, 1973.

LUTH 72 Luther, W.J. "The Conceptual Basis of CYBERNET," Networks, Randal Rostin, Editor, Prentice Hall, Englewood Cliffs, New Jersey, 1972.

MAUC 74 Mauceri, L.J. "Control of an Expanding Network - 'An Operational Nightmare'," Networks, Vol. 4, pp. 287-297, 1974.

MCCO 75 McCoy, C. Jr. "Improvement in Routing for Packet-Switched Networks," Naval Research Laboratory, Washington, D.C. NRL Report 7848, 1975.

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, Fall Joint Computer Conference, Anaheim, California, December 1972, Vol. 41, pp. 741-754.

MCQU 74 McQuillan, J.M. "Adaptive Routing Algorithms for Distributed Computer Networks," Bolt Beranek and Newman Inc., Cambridge, Massachusetts, Report No. 2831, May 1974.

MILL 74 Miller, B.L. "Dispatching from Depot Repair in a Recoverable Item Inventory System: On the Optimality of a Heuristic Rule," Management Science, Vol. 21, No. 3, pp. 316-325, November 1974.

MIYA 75 Miyahara, H., T. Hasegawa and Y. Teshigawara. "A Comparative Evaluation of Switching Methods in Computer Communication Networks," Proceedings of the IEEE International Conference on Communications, San Francisco, California, June 1, 1975, pp. 6-6 to 6-10.

MOOR 72 Moore, F.R. "Computational Model of a Closed Queueing Network With Exponential Servers," IBM Journal of Research and Development, Vol. 16, No. 6, pp. 567-572, November 1972.

NEWE 68 Newell, G.F. "Queues with Time-Dependent Arrival Rates I- The Transition Through Saturation," Journal of Applied Probability, Vol. 5, No. 2, pp. 436-451, August 1968.

NEWE 71 Newell, G.F. Applications of Queueing Theory, Chapman and Hall Ltd., London, 1971.

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.

ORTE 70 Ortega, J.M. and W.C. Rheinboldt. Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, New York, 1970.

PAOL 75 Paoletti, L.M. "AUTODIN," Computer Communication Networks, R.L. Grimsdale and F.F. Kuo, editors, Noordhoff International Publishing, Leyden, The Netherlands, 1975, pp. 345-372.

PARZ 62 Parzen, E. Stochastic Processes, Holden-Day, San Francisco, 1962.

PENN 74 Pennotti, M.D. "The Control of Congestion in Message-Switched Networks," Ph.D. Dissertation, (Electrical Engineering), Polytechnic Institute of New York, June 1974.

PENN 75 Pennotti, M.C. and M. Schwartz. "Congestion Control in Store and Forward Tandem Links," IEEE Transactions on Communications, Vol. COM-23, No. 12, pp. 1434-1443, December 1975.

PETE 73 Peters, R.A. and K.M. Simpson. "Data Communication in Europe. 1972-1985," Datamation, Vol. 19, No. 12, pp. 76-80, December 1973.

PORT 71 Port, E. and F. Closs. "Comparison of Switched Data Networks on the Basis of Waiting Times," IBM Zurich Research Center, Technical Report RZ405 (#14721), January 1971.

PORT 76 Port E., K. Kummerle, H. Rudin, C. Jenny and P. Zafiropulo. "A Network Architecture for the Integration of Circuit and Packet Switching" International Conference on Computer Communication Proceedings, Toronto, Canada, August 1976, pp. 505-514.

POUZ 75 Pouzin, L. "Presentation and Major Design Aspects of the CYCLADES Computer Network," Computer Communication Networks, R.L. Grimsdale and F.F. Kuo, editors, Noordhoff International Publishing, Leyden, the Netherlands, 1975, pp. 415-434.

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.

RAND 75 Randall, T., J. Edwards and P. Wallford. "DATRAN's Time Division Data Switching System," Proceedings of IEEE National Telecommunications Conference, New Orleans, Louisiana, December 1975, pp. 15-17 to 15-21.

REIS 76a Reiser, M. and H. Kobayashi. "On the Convolution Algorithm for Separable Queueing Networks," IBM Thomas J. Watson Research Center, Yorktown Heights, New York, Research Report RC 5914(#25202), January 1976.

REIS 76b Reiser, M. "Numerical Method in Separable Queueing Networks," IBM Thomas J. Watson Research Center, Yorktown Heights, New York, Research Report RC 5842(#25286), February 1976.

RICH 75 Rich, M.A, and M. Schwartz. "Buffer Sharing in Computer-Communication Network Nodes," Proceedings of the IEEE International Conference on Communications, San Francisco, California, June 1975, Vol. III, pp. 33-17 to 33-20.

ROBE 67 Roberts, L.G. "Multiple Computer Networks and Inter-Computer Communications," Proceedings of the ACM Symposium on Operation Systems, Gatlinburg, Tennessee, October 1967.

ROBE 70 Roberts, L.G., and B.D. Wessler. "Computer Network Development to Achieve Resource Sharing," AFIPS Conference Proceedings, Atlantic City, New Jersey, 1970, Vol. 36, pp. 543-549.

ROSN 76 Rosner, R.D. and B. Springer. "Circuit and Packet Switching; A Cost and Performance Trade Off Study," Computer Networks, Vol. 1, No. 1, pp. 7-26, June 1976.

ROSS 70 Ross, S.M. Applied Probability Models with Optimization Applications, Holden-Day, San Francisco, 1970.

RUDI 76 Rudin, H. "Flow Control; Introduction," International Conference on Computer Communication Proceedings, Toronto, Ontario, Canada, August 1976, pp. 463-466.

SCHW 77 Schwartz, M. "Computer-Communication Network Design and Analysis," Prentice Hall, Englewood Cliffs, New Jersey, 1977.

SOBE 74 Sobel, M.J. "Optimal Operation of Queues," Mathematical Methods in Queueing, Western Michigan University, Kalamazoo, Michigan, May 10-12, Spring-Verlag, Berlin, pp. 231-261, 1974.

SUNS 75 Sunshine, C.A. Interprocess Communication Protocols For Computer Networks," Stanford Electronics Laboratories, Stanford University, Technical Report #105, December 1975.

SYSK 60 Syski, R. Congestion Theory in Telephone Systems, Oliver and Boyd, London, 1960.

TYME 71 Tymes, LaRoy. "TYMNET - A Terminal-Oriented Communication Network," AFIPS Conference Proceedings, Spring Joint Computer Conference, Atlantic City, New Jersey, May 1971, Vol. 38, pp. 211-216.

VEIN 66 Veinott, A.F. Jr. "The Status of Mathematical Inventory Theory," Management Science, Vol. 12, No. 11, pp. 745-777, July 1966.

WALD 72 Walden, D.C. "A System for Interprocess Communication in a Remote Sharing Computer Network," Communications of ACM, Vol. 15, No. 4, pp. 221-230, April 1972.

WILK 56 Wilkinson, R.I. "Theories for Toll Traffic Engineering in the U.S.A.," The Bell Systems Technical Journal, Vol. 35, No. 2, pp. 421-514, March 1956.

WILL 74 Williams, A.C. and R.A. Bhandiwad. "A Generation Function Approach to Queueing Network Analysis of Multiprogrammed Computers," Mobil Oil Corp., Princeton, New Jersey, April 1974.

WONG 75 Wong, J.W-N. "Queueing Network Models for Computer Systems," Department of Computer Science, School of Engineering and Applied Science, University of California, Los Angeles, UCLA-ENG-7579, October 1975. (Also published as Ph.D. dissertation).

WONG 77 Wong, J.W-N. "Distribution of End-to-End Delay in Message-Switched Networks," Computer Networks, to be published, 1977.

ZAFI 76 Zafiropulo, P., E. Port and K. Kummerle. "Extension of a Circuit-Switched User/Network Interface to Packet-Switching," International Conference on Computer Communications Proceedings, Toronto, Canada, August 1976, pp. 515-522.

ZIMM 75 Zimmerman, H. "The Cyclades End-to-End Protocol," ACM-IEEE 4th Data Communication Symposium, Quebec City, Canada, October 1975, pp. 7-21 to 7-26.

COMPUTER SYSTEMS MODELING AND ANALYSIS
REPORT SERIES

Turn, R., "Assignment of Inventory of a Variable Structure Computer," January 1963, UCLA-ENG-6305.

Martin, D.F., "The Automatic Assignment and Sequencing of Computations on Parallel Processor Systems," January 1966, UCLA-ENG-6604 (AEC/ONR).

Coffman, E.G., "Stochastic Models of Multiple and Time-Shared Computer Operations," June 1966, UCLA-ENG-6638, (AEC/ARPA/ONR) NTIS AD636-976.

Bovet, D.P., "Memory Allocation in Computer Systems," June 1968, UCLA-ENG-6817 (AEC/ARPA/ONR).

Baer, J.L., "Graph Models of Computations in Computer Systems," October 1968, UCLA-ENG-6846 (AEC:UCLA-10P14-51/ ARPA/ONR) NTIS AD678-753.

Russell, E.C., "Automatic Program Analysis," March 1969, UCLA-ENG-6912 (AEC:UCLA-10P14-72/ARPA/ONR) NTIS AD 686-401.

Koster, R., "Low Level Self-Measurement in Computers," December 1969, UCLA-ENG-6957 (AEC:UCLA-10P14-84).

Cerf, V.G., "Measurement of Recursive Programs," May 1970, UCLA-ENG-7043 (AEC:UCLA-10P14-90/ARPA).

Volansky, S.A., "Graph Model Analysis and Implementation of Computational Sequences," June 1970, UCLA-ENG-7048 (AEC:UCLA-10P14-93).

Cole, G.D., "Computer Network Measurements: Techniques and Experiments," October 1971, UCLA-ENG-7165 (ARPA) NTIS AD 739-344.

Hsu, J., "Analysis of a Continuum of Processor-Sharing Models for Time-Shared Computer Systems," October 1971, UCLA-ENG-7166 (ARPA) NTIS AD739-345.

Ziegler, J.F., "Nodal Blocking in Large Networks," October 1971, UCLA-ENG-7167 (ARPA) NTIS AD 741-647.

Cerf, V.G., E. Fernandez, K. Gostelow, and S. Volansky, "Formal Control-Flow Properties of a Model of Computation," December 1971, UCLA-ENG-7178 (AEC:UCLA-10P14-105).

Gostelow, K.P., "Flow Control, Resource Allocation, and the Proper Termination of Programs," December 1971, UCLA-ENG-7223 (AEC:UCLA-10P14-106).

Cerf, V.G., "Multiprocessors, Semaphores, and a Graph Model of Computation," April 1972, UCLA-ENG-7223 (AEC:UCLA-10P14-110).

Fultz, G.L., "Adaptive Routing Techniques for Message Switching Computer Communication Networks," July 1972, UCLA-ENG-7252 (ARPA) NTIS AD 749-678.

Fernandez, E., "Activity Transformations on Graph Models of Parallel Computations," October 1972, UCLA-ENG-7287 (AEC:UCLA-10P14-116).

Gerla, M., "The Design of Store-and-Forward (S/F) Networks for Computer Communications," January 1973, UCLA-ENG-7319 (ARPA) NTIS AD 758-704.

Tatan, R., "Optimal Control of Tandem Queues," May 1973, UCLA-ENG-7333 (AFOSR).

Postel, J.B., "A Graph Model Analysis of Computer Communications Protocols," January 1974, UCLA-ENG-7410 (ARPA) NTIS AD 777-506.

Opderbeck, H., "Measurement and Modeling of Program Behavior and its Applications," April 1974, UCLA-ENG-7418 (ONR).

Lam, S.S. "Packet Switching in a Multi-Access Broadcast Channel with Application to Satellite Communication in a Computer Network," April 1974, UCLA-ENG-7429 (ARPA) NTIS AD 781-276.

Yavne, M., "Synthesis of Properly Terminating Graphs," May 1974, UCLA-ENG-7434 (AEC:UCLA-34P214-2).

Webster, F., "An Implementation of the Burroughs D-Machine," June 1974, UCLA-ENG-7449 (AEC:74P214-6).

Sylvain, P., "Evaluating the Array Machine," August 1974, UCLA-ENG-7462 (NSF).

Kleinrock, L., "Computer Network Research Final Technical Report," August 1974, UCLA-ENG-7467 (ARPA).

Tobagi, F.A., "Random Access Techniques for Data Transmission over Packet Switched Radio Networks," December 1974, UCLA-ENG-7499 (ARPA).

White, D.E., "Fault Detection Through Parallel Processing in Boolean Algebra," March 1975, UCLA-ENG-7504 (ONR).

Wong, J.W-N., Queueing Network Models for Computer Systems," October 1975, UCLA-ENG-7579 (ARPA).

Loomis, M.E., "Data Base Design: Object Distribution and Resource-Constrained Task Scheduling," March 1976, UCLA-ENG-7617 (ERDA:UCLA-34P214-26).

Kamoun, F., "Design Considerations for Large Computer Communication Networks," April 1976, UCLA-ENG-7642 (ARPA).

Korff, P.B., "A Multiaccess Memory," July 1976, UCLA-ENG-7607 (NSF-OCA-MC57203633-76074).

Ng, Y-N., "Reliability Modeling and Analysis for Fault-Tolerant Computers," September 1976 UCLA-ENG-7698 (NSF-MSC-7203633-76981).

Lee, D.D., "Simulation Study of a Distributed Processor Computer Architecture," November 1976, UCLA-ENG-76106 (ONR).

Scholl, M.O., "Multiplexing Techniques for Data Transmission over Packet Switched Radio Systems," December 1976, UCLA-ENG-76123 (ARPA).

Erlich, Z., "On Centralized Bus Transportation Systems with Poisson Arrivals," December 1976, UCLA-ENG-76124 (ARPA).

Naylor, W.E., "Stream Traffic Communication in Packet Switched Networks," August 1977, UCLA-ENG-7760 (ARPA).

Kermani, P., "Switching and Flow Control Techniques in Computer Communication Networks," February 1978, UCLA-ENG-7802 (ARPA).