AD-A047 496 CALIFORNIA UNIV LOS ANGELES DEPT OF COMPUTER SCIENCE F/G 17/2
ADVANCED TELEPROCESSING SYSTEMS.(U)
JUN 77 L KLEINROCK, G ESTRIN, R MUNTZ DAHC15-73-C-0368
UNCLASSIFIED NL

1 OF 3
ADA047 496



AD A047496

(12)
B.S.

COMPUTER NETWORK RESEARCH

ADVANCED TELEPROCESSING SYSTEMS
SEMIANNUAL TECHNICAL REPORT

JUNE 30, 1977

Principal Investigator: Leonard Kleinrock

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

AD No. ____
DDC FILE COPY

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

DDC
RECEIVED
DEC 6 1977
B

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)* (6) Advanced Teleprocessing Systems.		5. TYPE OF REPORT & PERIOD COVERED Semiannual Technical Report January 1 - June 30, 1977
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) Leonard Kleinrock, Gerald Estrin, Richard Muntz, Gerald Popek		8. CONTRACT OR GRANT NUMBER(s) (15) DAHC15-73-C-0368
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 (11) Advanced Research Projects Agency, 1400 Wilson Boulevard, Arlington, Virginia 22209		12. REPORT DATE 30 June 1977 (12)
		13. NUMBER OF PAGES 240 254 p.
14. MONITORING AGENCY NAME & ADDRESS *(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)*
		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)*

405 749

18. SUPPLEMENTARY NOTES

(9) Semiannual technical rept. 1 Jan - 30 Jun 77

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DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-LF-014-6601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, 1977

ARPA Contract DAHC-15-73-C-0368

Principal Investigator: Leonard Kleinrock

Co-Principal Investigators: Gerald Estrin
Richard Muntz
Gerald Popek

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 White Section
DDC Buff Section
UNANNOUNCED
JUSTIFICATION
BY
DISTRIBUTION/AVAILABILITY CODES
Dist. AVAIL. and/or SPECIAL
A

ADVANCED TELEPROCESSING SYSTEMS

Advanced Research Projects Agency
Semiannual Technical Report

June 30, 1977

1. INTRODUCTION

This semiannual technical report covers research carried out by the Advanced Teleprocessing Systems group at UCLA under ARPA Contract DAHC 15-73-C-0368 during the period January 1, 1977 to June 30, 1977. Advancements have been made on all four contracted tasks; namely, ground radio packet switching, satellite studies, resource sharing and security. In the following paragraphs, we describe the progress and point to the list of references which represent the published work resulting from this supported research.

Following this summary is a list of publications produced as a result of the recent research on this contract covering the six months being reported. This list contains only those articles and reports which, in fact, did appear in print. 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. As usual, 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 the effect that packet-switching networks have on "stream traffic". An example of stream traffic is packetized speech; other examples are real-time traffic generated, for example, from real-time sources such as television cameras, temperature-monitoring systems, etc. The characteristic of this traffic is that it has a time constraint as well as a throughput constraint. The time constraint is that the roundtrip delay must not exceed a threshold before the usefulness of the data becomes lost; in addition, the timing between delivered packets must not be too large or the continuity of the signal is lost. The throughput constraint comes about because the traffic continues to be generated and requires a moderate (if not high) bandwidth through the network. However, an advantage of stream traffic is that not all of it must be delivered; in some sense, it is volatile and becomes stale. This kind of traffic is very different from that which the ARPANET was originally designed to carry and, as one might expect, some unusual phenomena occur when sent through a packet-switched network. The particular details in the following report represent the Ph.D. dissertation of William E. Naylor (chairman, Leonard Kleinrock) and the work is entitled "Stream Traffic Communication in Packet-Switched Networks." Whereas the effort was to study stream traffic flow through a network, it turns out that a large effort was devoted to studying the routing procedure (in particular, the periodic updating procedure) in the ARPANET. It was found that this routing procedure

severely affected and degraded stream traffic performance and so a number of suggestions were made and evaluated as regards the priority ordering of updating. A loop-free algorithm for updating routing tables is also presented and proven to be loop-free. Lastly, the tradeoff between gaps in transmission and delay in reception of that transmission is studied. A number of simulations and measurement experiments are reported which confirm and support the analytical treatment given in this work. Further details can be found in the abstract and dissertation reproduced in the main body following the list of publications.

The first four references in the list of publications pertain to the behavior and analysis of landbased computer networks. The first paper describes some additional experimentation for the design of packet networks and summarizes some of the successful design procedures which we have developed. The second paper on the analysis of buffer allocation schemes describes various ways for sharing storage in a multiplexing node within a computer network. The third paper summarizes a portion of a study of hierarchical routing in large networks and shows the tradeoff between the length of routing tables and the path length presented by those routes. The fourth paper has to do with flow control and throughput in the ARPANET. It points out some traps and degradations which may arise in networks, shows how to correct them and then warns against the kinds of effects which produce deadlocks. The fifth paper on packet switching in radio channels represents a contribution to our work in ground radio packet switching. It presents a new scheme known as mini-slotted alternating priority (MSAP) and presents the performance profile for this system. Its particular usefulness is in the heavy traffic case since it achieves a capacity of 1 and still provides rather efficient communications delays. Further, it discusses the regions in which various access schemes are optimal. Overall, the advancements in ground radio packet switching has moved along extremely well. A number of publications have been accepted and will be reported on in the next semiannual technical report.

The main report on stream traffic in packet networks is given after the following list of publications.

List of Publications

Gerla, M. and L. Kleinrock, "On the Topological Design of Distributed Computer Networks," IEEE Transactions on Communications, Vol. COM-25, No. 1, January 1977, pp. 48-60.

Kermani, P. and L. Kleinrock, "Analysis of Buffer Allocation Schemes in a Multiplexing Node," Conference Record, International Conference on Communications, Vol. 2, Chicago, Illinois, June 1977, pp. 30.4-269 to 275.

Kleinrock, L. and F. Kamoun, "Hierarchical Routing for Large Networks: Performance Evaluation and Optimization," Computer Networks, Vol. 1, No. 3, January 1977, pp. 155-174.

Kleinrock, L. and H. Opderbeck, "Throughput in the ARPANET--Protocols and Measurement," IEEE Transactions on Communications, Vol. COM-25, No. 1, January 1977, pp. 95-104

Kleinrock, L. and M. Scholl, "Packet Switching in Radio Channels: New Conflict-Free Multiple Access Schemes for a Small Number of Data Users." Conference Record, International Conference on Communications, Vol. 2, Chicago, Illinois, June 1977, pp. 22.1-105 to 111.

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, 1977.

STREAM TRAFFIC COMMUNICATION IN PACKET-SWITCHED NETWORKS

by

WILLIAM E. NAYLOR

ARPA Contract No. DAHC-15-73-C-0368

SCHOOL OF ENGINEERING AND APPLIED SCIENCE
University of California
Los Angeles

ABSTRACT

This research deals with the transmission of stream traffic through packet switched networks. Stream traffic communication is characterized by 1) a requirement of small response time and moderate throughput, the fact that 2) timing is an integral part of the information and that 3) the information is redundant and somewhat tolerable to loss. Remote voice communication provides an example of stream traffic communication. Traditional dedicated communication systems supporting stream traffic have exhibited fixed capacity and fixed delay. In such systems each user is assigned a communication link whose capacity is large enough to support that user's peak load. While timing is easily preserved by such systems, it is difficult to share communication links. This research explores systems in which the communication channels are shared among many users, thus causing delay and capacity to vary.

The first area of concentration is an examination of some important factors which adversely affect delay in packet switched networks. The ARPANET is used as an example of such a network. We focus on loop control in adaptive routing, priority assignment, and the effect of periodic update routing in large networks. Suggestions are offered for performance improvement in these areas. Analysis and simulation are used to predict the magnitude of improvement. By modifying the ARPANET procedures in the manner suggested,

measurement and simulation indicate that a 40% to 50% reduction in average delay may be achieved.

Even with improved performance a packet switched network exhibits variable delay due in general to the possible queueing of packets at channels in the network. With stream traffic it is important to preserve the relative timing of the information as closely as possible. This is accomplished by smoothing the departure of information with buffering. This is the second main area of concentration. We propose buffering schemes which adapt to changing network delay and which trade output smoothness against buffering delay. The performance of the buffering strategies is compared by analysis and simulation.

TABLE OF CONTENTS

LIST OF TABLES xi

LIST OF FIGURES xiii

CHAPTER 1. INTRODUCTION 1

1.1 Recent history of packet switching 1

1.2 Voice goes digital 4

1.3 Speech compression research 5

1.4 The characteristics of stream traffic 6

1.5 Statement of the problem 8

1.6 Summary of results 9

CHAPTER 2. LOOP CONTROL IN ADAPTIVE ROUTING 14

2.1 Introduction 14

2.2 Examples of looping 15

2.2.1 ARPANET procedures 17

2.2.1.1 Loop-prone routing (LPR) 17

2.2.1.2 Hold-down routing (HDR) 19

2.2.1.3 Modified hold-down routing (MHDR) 20

2.2.2 TIDAS Net procedure 21

2.3 Degradation due to loops 23

2.4 A direct approach to loop control 25

2.4.1 A local loop-free routing algorithm (LLFR) . 25

2.4.2 A loop-free routing algorithm (LFR) 26

2.5 Proofs of loop-freeness 33

2.6 Simulation results 43

2.7 Conclusions 47

TABLE OF CONTENTS (continued)

CHAPTER 3. SYSTEM PRIORITIES 48

3.1 Introduction 48

3.2 Measured results 49

3.3 Analysis 61

3.3.1 System description 61

3.3.2 General theory 65

3.3.3 A model 66

3.3.4 Analytic results 69

3.3.5 Numerical results 71

3.4 Simulation results 82

3.5 Conclusions 84

CHAPTER 4. ON THE EFFECT OF PERIODIC UPDATE ROUTING PROCEDURES 85

4.1 Introduction 85

4.2 Measurement 86

4.3 Analysis 91

4.3.1 ARPANET periodic routing scheme description 91

4.3.2 A simple model 92

4.4 Simulation 101

4.5 Conclusions 112

CHAPTER 5. (SOURCE AND) DESTINATION BUFFERING CONSIDERATIONS FOR STREAM TRAFFIC COMMUNICATION 113

5.1 Introduction 113

5.2 Sending strategy 114

5.3 Adaptive receiving 118

TABLE OF CONTENTS (continued)

5.3.1 Gap control 119

5.3.2 Playback methods 120

5.3.2.1 Expanded time axis (Method E) 121

5.3.2.2 Late data ignored (Method I) 122

5.3.3 Performance evaluation 124

5.3.3.1 Distribution of network delay 127

5.3.3.1.1 Measurement 127

5.3.3.1.2 Simulation 129

5.3.3.1.3 Tractable approximations 137

5.3.3.2 Statistical independence of network delay 152

5.3.3.3 Gap probability 162

5.3.3.3.1 Gap probability for playback method E 162

5.3.3.3.2 Gap probability for playback method I 168

5.3.3.4 Delay predictors 169

5.3.3.4.1 m-sample range 170

5.3.3.4.2 m-sample partial range 171

5.3.3.4.3 A delay tracker 172

5.3.3.5 Communication quality 180

5.3.3.6 Numerical results 185

5.3.3.6.1 The exponential distribution 185

5.3.3.6.2 The shifted exponential distribution 210

5.3.3.6.3 The Erlang family of distributions . 211

5.3.3.7 Simulation results 215

TABLE OF CONTENTS (continued)

5.4 Conclusions 221

CHAPTER 6. CONCLUSIONS AND SUGGESTIONS FOR FUTURE RESEARCH 223

6.1 Conclusions 223

6.2 Suggestions for future investigation 226

BIBLIOGRAPHY 229

LIST OF TABLES

2.1	ARPANET loop persistence time	25
3.1	ARPANET traffic and delay summary	50
3.2(a)	Observed processing delay in msec (516 IMPs)	52
3.2(b)	Observed processing delay in msec (316 IMPs & TIPs)	53
3.3	Variation of processing delay with time of day (516 IMP)	55
3.4	Variation of processing delay with time of day (316 IMP)	56
3.5(a)	Processing delay statistics (node 22 a 516 IMP, 14 MAR 75)	59
3.5(b)	Processing delay statistics (node 27 a 316 IMP, 14 MAR 75)	59
3.5(c)	Processing delay statistics (node 22 a 516 IMP, 24 JUL 75)	60
3.5(d)	Processing delay statistics (node 27 a 316 IMP, 24 JUL 75)	60
3.6	IMP priority levels	62
3.7	Channel function processing time estimates ..	72
3.8	Connectivity of processor types	73

LIST OF FIGURES

2.1	Routing table (node i)	16
2.2	A portion of a network	18
2.3	Comparative performance of adaptive routing procedures (simulation)	45
3.1(a)	Simplified model of ARPANET queue structure .	63
3.1(b)	Simplified model of ARPANET queue structure with priority within the TASK queue	64
3.2(a)	The two level model	68
3.2(b)	The three level model	68
3.3(a)	Mean waiting time on TASK queue (2-connected 516 IMP)	75
3.3(b)	Mean waiting time on TASK queue (3-connected 516 IMP)	76
3.3(c)	Mean waiting time on TASK queue (2-connected 316 IMP)	77
3.3(d)	Mean waiting time on TASK queue (3-connected 316 IMP)	78
3.3(e)	Mean waiting time on TASK queue (2-connected TIP)	79
3.3(f)	Mean waiting time on TASK queue (3-connected TIP)	80
3.4	High and low priority routing (simulation) ..	83
4.1	Round-trip delay measurement (1 hop, s=124) .	89
4.2	Round-trip delay measurement (10 hops, s=124)	89
4.3	Round-trip delay measurement (5 hops, s=248)	90
4.4	Round-trip delay measurement (2 hops, s=165)	90
4.5	Single channel model	92
4.6	Unfinished work (routing only)	95
4.7	Unfinished work at stream arrivals	95

PRECEDING PAGE NOT FILMED BLANK

LIST OF FIGURES (continued)

4.8(a)	Precession of interference pattern (s=249) ..	97
4.8(b)	Precession of interference pattern (s=165) ..	98
4.8(c)	Precession of interference pattern (s=124) ..	99
4.9	Two tandem channel interference pattern example (s=124)	100
4.10	Round-trip delay simulation (1 hop, s=124) ..	104
4.11	Round-trip delay simulation (10 hops, s=248)	104
4.12	Round-trip delay simulation (5 hops, s=248) .	105
4.13	Round-trip delay simulation (2 hops, s=165) .	105
4.14	Round-trip delay simulation with low priority routing processing (1 hop, s=124)	106
4.15	Round-trip delay simulation with low priority routing processing (10 hops, s=248)	106
4.16	Round-trip delay simulation with low priority routing processing (5 hops, s=248)	107
4.17	Round-trip delay simulation with low priority routing processing (2 hops, s=165)	107
4.18	Adaptive routing vs. fixed routing (simulation)	110
5.1	Fill time - throughput requirement tradeoff .	116
5.2(a)	Playback methods (method E)	123
5.2(b)	Playback methods (method I)	123
5.3	End-to-end communication	125
5.4(a)	Delay histogram (measurement, 1 hop)	130
5.4(b)	Delay histogram (measurement, 2 hops)	131
5.4(c)	Delay histogram (measurement, 5 hops)	132
5.4(d)	Delay histogram (measurement, 10 hops)	133
5.4(e)	Delay histogram (simulation, load = .1)	134

LIST OF FIGURES (continued)

5.4(f)	Delay histogram (simulation, load = .5)	135
5.4(g)	Delay histogram (simulation, load = .9)	136
5.5	Shifted exponential fit to delay histogram (simulation, load = .1)	138
5.6(a)	Erlang fit to delay histogram (measurement, 5 hops)	139
5.6(b)	Erlang fit to delay histogram (measurement, 10 hops)	140
5.6(c)	Erlang fit to delay histogram (simulation, load = .5)	141
5.6(d)	Erlang fit to delay histogram (simulation, load = .9)	142
5.7(a)	Shifted Erlang fit to delay histogram (measurement, 1 hop)	145
5.7(b)	Shifted Erlang fit to delay histogram (measurement, 2 hops)	146
5.7(c)	Shifted Erlang fit to delay histogram (measurement, 5 hops)	147
5.7(d)	Shifted Erlang fit to delay histogram (measurement, 10 hops)	148
5.7(e)	Shifted Erlang fit to delay histogram (simulation, load = .1)	149
5.7(f)	Shifted Erlang fit to delay histogram (simulation, load = .5)	150
5.7(g)	Shifted Erlang fit to delay histogram (simulation, load = .9)	151
5.8(a)	Delay correlation (measurement, 1 hop)	154
5.8(b)	Delay correlation (measurement, 2 hops)	155
5.8(c)	Delay correlation (measurement, 5 hops)	156
5.8(d)	Delay correlation (measurement, 10 hops)	157
5.8(e)	Delay correlation (simulation, load = .1) ...	158

LIST OF FIGURES (continued)

5.8(f)	Delay correlation (simulation, load = .5) ...	159
5.8(g)	Delay correlation (simulation, load = .9) ...	160
5.9	Pseudo random sequence correlation	161
5.10	Delay tracker	175
5.11	Delay tracker Markov chain	176
5.12	One parameter quality functions	182
5.13	Region of acceptability	183
5.14	Contours of quality .9	183
5.15(a)	Constant quality contours (n = 1)	184
5.15(b)	Constant quality contours (n = 10)	184
5.16	Mean range vs. sample size for the exponential distribution	192
5.17	Gap probability (Method E) vs. sample size ..	195
5.18	Gap probability (Method E) vs. sample size ..	196
5.19	Gap probability (Method I) vs. sample size ..	199
5.20	Gap probability (Method I) vs. sample size ..	200
5.21	Gap probability (Method E) vs. mean range ...	205
5.22	Gap probability (Method E) vs. mean range ...	206
5.23	Optimal k for given m	207
5.24	Performance at m, k-optimal (Method E)	208
5.25	Convergence to limiting performance	209
5.26	Erlang family of density functions	213
5.27	Performance with Erlang distributions (r = 5, 10, 50)	214

LIST OF FIGURES (continued)

5.28(a) Range prediction performance simulation (Method E, measurement, 1 hop) 217

5.28(b) Delay tracker performance simulation (Method E, measurement, 1 hop) 218

5.29(a) Range prediction performance simulation (Method E, measurement, 10 hops) 219

5.29(b) Delay tracker performance simulation (Method E, measurement, 10 hops) 220

CHAPTER 1

INTRODUCTION

1.1 Recent history of packet switching

Since the development of the ARPANET [Robe 70] in the late 1960s by the Advanced Research Projects Agency of the United States Department of Defense, there has been an ever increasing amount of activity in the area of packet switching research, development, and implementation. The ARPANET, built primarily by Bolt, Beranek and Newman, Inc., Cambridge, Massachusetts (BBN), is the most widely known example of a packet switched network. Other networks are emerging as many agencies, countries and federations are currently funding research and/or building packet switched networks. Among these are WWMCCS [Beno 71], NPL [Barb 70], CYCLADES [Pouz 73], DATAPAC [DPAC 74], COST11 [Barb 72], to name a few. The first commercial packet switching service is now in operation in the U. S. (operated by Telenet Communications Corporation) [Math 75].

Extensions to the ARPANET form of packet switching have taken place and have lead to experimental systems -- ALOHA [Abra 70], PRNET [Kahn 75], SATNET [Klei 73], ETHERNET [Metc 77]. Currently, the interconnection of such networks [McKe 74a] and the standardization of protocols [Pouz 75], [Hove 76] are each of considerable interest in the data

communications community. Also of interest are aspects of secure communications [Farb 75].

The salient features of packet switching

Here we describe the important (so far as we are concerned here) properties of a packet switched network. A packet switched network (of the ARPANET form) incorporates switching computers within the communications media. The network consists of HOSTs (i.e., packet or message sources -- computers or terminals) and a subnet to which the HOSTs are attached. The subnet contains switching nodes (called IMPs in the ARPANET [Hear 70]) which are connected, with some topology, by a set of communications lines or channels. HOSTs are connected to the network at switching nodes. A source HOST sends a message (to a destination HOST) by delivering the message, with the address of the destination HOST, to its connected switching node. The switching node then breaks the message into one or more packets. Each packet is then forwarded (if necessary) by the switching node to one of its neighbors (i.e., another switching node which is directly connected to the forwarding node) the choice of neighbors being governed by the routing procedure. Each node, which encounters the packet, forwards it to a neighbor the choice of which is again based on the routing procedure.

Often there is some form of error detection and recovery in the forwarding process in order to assure the correct receipt of packets. In the ARPANET, for example,

there is a checksum appended to the packet at the sender which is checked by the receiver. Only when the checksum is correct and buffer space is available does the receiver return (sometimes "piggybacked" on reverse traffic) an acknowledgement (ACK) to the sender. If a sent packet has not received its ACK within a certain time (currently 128 milliseconds), then the packet is retransmitted.

The forwarding process is repeated until the packet arrives at a switching node which is connected to the destination HOST, at which point the packet is reassembled with the other packets of the same message. When all its packets have arrived the message is sent to the destination HOST. In the ARPANET this final step creates an end-to-end acknowledgement called a Ready-for-Next-Message (RFNM) which is sent back to the sender.

At each forwarding step, often called a hop, along the way a packet incurs some processing delay which is required to make the routing decision (i.e., to which neighbor to route this packet). Also, it may encounter a queue of packets waiting to be sent to the neighbor and must therefore wait until all (higher or equal priority) packets have been served (i.e., sent to the neighbor) before it may use the channel. In particular, this may result in variable delay. That is, a message sent from A to B at time t0 may experience a different delay than a similar message sent at time t1. The maximum attainable throughput from A to B may vary with time as well, since it will depend on the level of

interference from other communicating HOST pairs.

Packet switching promises greater efficiency of resources by sharing those resources among many users, each of which uses only a small portion of the total capacity. The savings, at least in part, are passed along to the user. With the rapid growth of data communications we shall surely see more and bigger packet switched networks emerge.

Because of its digital nature, today's packet switching technology has been designed and used chiefly for data communications (e.g., terminal-to-computer and computer-to-computer). Other forms of communication (i.e., voice), which have traditionally been accomplished by analog methods, are now beginning to use digital technology.

1.2 Voice goes digital

The advent of inexpensive, highly reliable digital transmission equipment has already led to the use of such circuits in "short-haul" voice communications [Jame 72]. Also some 40 million circuit miles of digital trunks were in place by the end of 1975 [Falk 77]. There seems little doubt that the use of digital transmission will increase as time progresses. We are told by Gallager [Gall 77a] that the military will use an all digital telephone network in the 1980's. Cost will be the most important reason for conversion to digital transmission. According to Falk [Falk 77] "Bell Canada plans to save $40 million a year in capital costs by using new digital equipment". Another

consideration is that a digital signal is much easier to encrypt than is an analog signal. This could lead to a far more secure means of voice communication.

Simple digitization requires a data rate of between 50 and 60 kilobits per second (kbps) to provide quality equal to conventional telephone equipment. This is extremely high compared to current requirements in data communications. Fortunately, speech is highly redundant and can therefore be significantly compressed.

1.3 Speech compression research

There are two methods of compression currently under study by the ARPA Network Speech Compression (NSC) group - (a) continuous variable slope delta modulation (CVSD) [Forg 74] and (b) linear predictive coding (LPC) [Atal 71]. Experimental studies of remote voice communication using these two schemes have already taken place in the ARPANET. The CVSD scheme transmits one bit per sample, whose value (1 or 0) depends on whether the last synthesized point is above or below the current input. This scheme provides acceptable quality in the 8 to 20 kbps region [Forg 74]. The experiments in the ARPANET with this technique used a peak rate of 10 kbps and experienced delays on the order of 2 to 4 seconds between speaker and listener (over a 10 hop network path), as reported by Cohen in [Nayl 74a]. The LPC method is based on a model of the vocal tract and assumes that a speech sample may be approximated by a linear combination of

the previous n samples. The coefficients in the linear combination are slowly varying quantities. This algorithm provides high quality speech in the range of 2.4 to 9.6 kbps [Atal 71]. With refinements such as DELCO [Magi 73], one can achieve average data rates of 1.2 to 4.8 kbps. A recent experiment in the ARPANET had a peak data rate of 4.1 kbps and an average of 1.4 kbps, with acceptable quality, using the LPC compression scheme [McCa 75].

This research has created, in essence, a new form of "data" communication, and provides the opportunity to apply packet switching technology to the area of remote voice communication. Human speech contains a great deal of redundancy and silence. During silence, with most dedicated or switched circuit systems the channel remains unused. Packet switching would allow this dead time to be used by others thus allowing greater channel efficiency which eventually translates to cost savings.

1.4 The characteristics of stream traffic

This dissertation investigates a set of problems related to the use of packet switching technology for the purpose of stream traffic communication. Remote voice communication is an example of stream traffic communication. Let us define stream traffic. Stream traffic is characterized by the following three properties:

1) small response time and moderate throughput are required,

2) timing is an integral part of the information, and

3) the information contains redundancy.

Property 1 allows for the possibility of real-time interactive communication between two or more locations. This property alone makes stream traffic distinguishable from the two classical forms of data communication. Packet switched networks have, in general, been designed to carry traffic which has traditionally been classified into two categories: (a) LD - low delay (interactive), and (b) HT - high throughput (file transfer). As noted by Cohen, Opderbeck and Kleinrock [Opde 74], stream traffic communication falls into yet another category (c) ST - stream traffic, requiring both low delay and moderate throughput.

Not only are the transmission requirements of stream traffic unique, but the information itself is of a somewhat different nature than the usual data communication. Property 2 indicates that each unit (bit, if you will) of information has an associated (possibly implied) time stamp and that the relative timing of the information should be preserved as well as possible by the transmission media. Although sequencing is important, in ordinary data communications there is no notion of timing associated with the information. Unlike traditional data communication, the information in stream traffic is somewhat redundant. This means that the information is less vulnerable to loss within the system than for traditional data communication. The communication media may lose a small fraction of the

information without seriously affecting the quality. This is clearly distinct from, say, remote job entry in which a very precise specification in the form of a program is the object of communication, and therefore no loss at all can be tolerated.

This characterization of stream traffic is intended to fit two- (or more) way interactive voice communication. There are other areas of communication which may possess these properties as well. Distribution and local broadcast of live (or delayed and transmitted from the source in real-time) radio or television programming would appear to have the three properties. While there is no interaction required of such one-way systems, low delay may be required due to lack of buffer space at the receiver. Television in particular requires extremely high throughput by ARPANET standards. Therefore the capacity of a network designed to carry television must have a capacity so that the television transmission consumes only a moderate portion of that capacity.

1.5 Statement of the problem

There are two main areas of emphasis which are explored in this dissertation.

1) Identify and examine those design considerations (of packet switched networks) which have a significant impact on the performance of stream traffic communication in a packet switched environment.

2) Given a (packet switched) network, examine some sending and receiving policies which attempt to balance the preservation of the relative timing against end-to-end delay.

Since the ARPANET is a convenient example of a packet switched network, we examine it as a case study and attempt to extract some general results about designing a packet switched network aimed at the ST class, but still retaining capabilities in the other classes. The emphasis here is to reduce delay in order to satisfy property (1). Loop control, system priorities, and the effects of periodic routing updates are among the issues of concern under area (1).

1.6 Summary of results

In Chapter 2 we investigate the occurrence of loops caused by routing update procedures. By routing we mean the process by which packets are directed from switch to switch through the network. This definition is intended to include the structure and use of local routing information as well as the updating of this local information. Routing algorithms are broken into four classes in [Crow 75]; (a) "non-adaptive" or fixed - where the route between any two nodes remains fixed. (b) "Centralized adaptive" - in which routes are dynamically modified by a central overseer. (c) "Isolated adaptive" - where routes are changed dynamically by each node without sharing information among the nodes. (d)

"Distributed adaptive" - in which routes are dynamically changed by each node with information shared among the nodes in the form of routing update packets. It should be clear that loops would be forbidden in types (a) and (b) by a small amount of care. It would appear to be impossible to prevent loops in type (c). We therefore concentrate our efforts in studying the distributed adaptive type algorithm. The existence of routing loops under a routing procedure is a possible source of performance degradation. Packets which are trapped in loops have increased network delay. If a loop persists sufficiently long, then interaction is hampered if not destroyed (i.e., property (1) is violated). In [Nayl 75] a loop-free routing algorithm is presented. It was shown that this algorithm cannot create loops, but its operational characteristics required further examination. We investigate the operational characteristics of this and other algorithms through the use of simulation. A local loop-free (or ping-pong-free) algorithm is found to perform best among those algorithms tested.

In Chapter 3 we consider priority assignment among tasks within the system. An important issue in the performance of a network is the assignment of the priority among the various functions within a node as well as the priorities assigned to the transmission of packets on the channels. As an example, we have found that under the current ARPANET strategy, packets waited as long, on the average, for the processor as for the transmission channels. This is

due to the fact that routing update packets and non-routing packets (i.e., data and control distinct from routing update packets) are processed in first-come-first-served (FCFS) order within the nodes. In addition some of the nodes in the network spend more than 50% of their total capacity just processing routing updates, compared to "only" 18% routing overhead on the channels. Examination of this problem isolated to a single node suggests that since, in the current ARPANET, routing packets require approximately 40 times the processing time that non-routing packets require, the processing of routing packets should be done at a lower priority. Simulation of the system as a whole, suggests that the local optimum leads to a global optimum in terms of mean delay and network throughput. Therefore, by modifying the priority structure one could reduce delay and thereby assist in complying with property (1).

In Chapter 4 we show some interesting delay behavior for a periodic stream traffic source. The behavior is due to the periodic nature of the routing update procedure. The current ARPANET scheme provides for periodic updates, and with the growing size of the network the rate of routing updates has been increased to allow for better propagation of information through the network when congestion or failure occurs. We show that one must pay a high price, in terms of delay, for this feature during normal operation however. Under normal conditions routing updates are not required very often since most routes need not change much.

Long term data on ARPANET reliability shows that network component failures occur very infrequently compared to the routing update period. These results suggest an asynchronous method of updating routing tables. This would result in lower average delay and thus help provide property (1).

Under the second area of emphasis we concentrate on methods which attempt to preserve the relative timing of the information without destroying property (1). In Chapter 5 we examine buffering strategies, at source and destination, which attempt to minimize gaps in the output stream and at the same time attempt to minimize the delay between speaker and listener. We found that delay varies quite rapidly and therefore delay prediction could not be used to adjust to the network dynamics on a message by message basis; rather we found that the sending strategy should remain fixed for "long" periods.

More extensive results are found to predict the performance of destination buffering schemes. We define some delay prediction techniques and two playout methods. Based on some assumptions on system delay we have developed models of the system behavior in terms of delay and gap probability. The solution of the models requires the knowledge of the system delay distribution. Exact results are obtained for the exponential distribution and shifted exponential distribution. Numerical integration is used to obtain results for an Erlang family of distributions which previous

models [Klei 64] tell us is a somewhat better model of network delay. Finally by simulation we compare the performance of our delay variation estimation technique to a delay tracker method which has been used experimentally in the ARPANET, and to a planned revision of that scheme.

Chapter 6 lists our conclusions and suggests some areas of interest for further research.

Our critical examination of ARPANET procedures has had an impact not only stream traffic communication, but on the efficiency of packet switching in general. The study of the performance of destination buffering schemes has provided a framework in which other such techniques may be examined. We believe strongly that stream traffic communication (remote voice communication in particular) is within the realm of uses for packet switched networks, but much work is needed in order to produce a usable system.

CHAPTER 2

LOOP CONTROL IN ADAPTIVE ROUTING*

2.1 Introduction

In a packet-switched network in which some scheme of adaptive routing of packets is used, there exists the possibility that packets will become trapped in loops. That is, packets may be routed in such a way that they return several times to some set of nodes for at least a finite period of time before reaching their eventual destination, thus wasting network bandwidth and significantly increasing message delay. Routing loops are of concern in stream communication primarily because of this increased delay effect. In this chapter we consider procedures for controlling such loops.

The problem of looping in adaptive routing has been known to exist for some time [Kahn 71], [Fult 72], [Gerl 73], [McQu 74], [Cegr 75], [Pick 76], [Gall 77]. Previous approaches to this problem have been to detect and remove loops [Kahn 71], [Fult 72], or to reduce the incidence of such loops [McQu 74], [Cegr 75]. Gerla [Gerl 73] proved that an optimal routing policy must be loop-free. More recently, with a procedure known as the "last m nodes visited" (LMNV) algorithm [Pick 76], the packets are

---

* This chapter is a revised version of [Nayl 75].

prevented from looping in loops of size m nodes or smaller. Gallager has created an optimal routing scheme which is loop-free [Gall 77]. We first give two examples of existing networks whose adaptive routing algorithms can cause such loops. We next investigate the order of magnitude of the degradation due to loops. We then describe a new ping-pong-free and a new loop-free algorithm. A formalism for dealing with the loop problem is introduced in order to prove the loop-free properties of the algorithms. Then by way of simulation we compare the performance of several of the routing update schemes.

2.2 Examples of looping

Below we describe four routing update procedures at least three of which may cause loops to occur. There is a degree of commonality among these schemes. In each procedure there exists a routing table at each node whose entries indicate the direction (i.e., the channel or neighbor address) in which to send a packet headed for a particular destination (see Figure 2.1). When a packet arrives at a node (say node i), one uses that packet's destination address d as an index into the routing table to determine the channel c(i,d) over which to route that packet. The differences in these routing procedures appear in the updating of these tables.

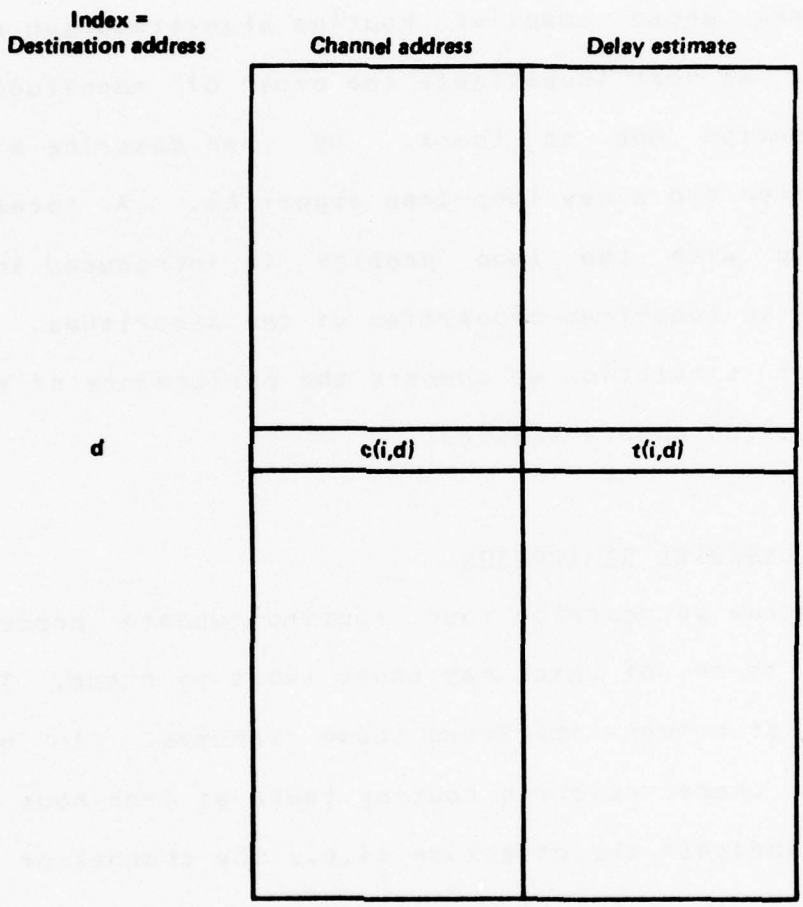


Figure 2.1. Routing table (node *i*)

2.2.1 ARPANET procedures

Each of the first three procedures have been used in the ARPANET [Robe 70], [McQu 72]. The first two can cause loops. The third scheme is currently in operation and is believed to prevent at least ping-pong loops.

2.2.1.1 Loop prone routing (LPR)

In the ARPANET each entry in the routing table consists of (among other things) a delay estimate and the address of the channel for which that delay estimate holds [McQu 74], [McQu 74a], [Klei 76]. The table is updated upon the arrival of a routing message from a node's neighbor. The routing message is a copy of the delay portion of the neighbor's routing table. The delay estimates in that node's routing table are compared with the routing table delay estimate entries in the arriving update message plus the delay from that node to its neighbor (currently 4 units + 1 unit for each packet on that channel's queue). The smaller of the two values replaces the routing table estimate and the "best delay channel" is changed if necessary. More precisely; Node i upon receipt of a routing update packet from neighbor node j performs the following algorithm for each d:

LPR

```
1     t'(i,d) = t(j,d) + q(i,j) + h
2     if t'(i,d) < t(i,d) then c(i,d) = c'(i,j)
3     if c(i,d) = c'(i,j) then t(i,d) = t'(i,d)
```

where t(j,d) = the delay estimate from node j to node d, q(i,j) = the length (in packets) of the queue from node i to neighbor node j, h = the one hop delay bias (currently 4 units in the ARPANET), c(i,d) = the (next) channel on which to route a packet residing at node i destined for node d, and c'(i,j) is the channel which connects neighbors i and j. This is a simplification of the actual algorithm, but the essential features are retained.

Consider the network topology pictured in Figure 2.2. We are concerned with the estimated delay from nodes B and C to some distant node D. We will use the following notation of [Klei 75a]:

d/A d = estimated delay to node D
     A = next neighbor in path to D

B->C node B sends a routing update to C

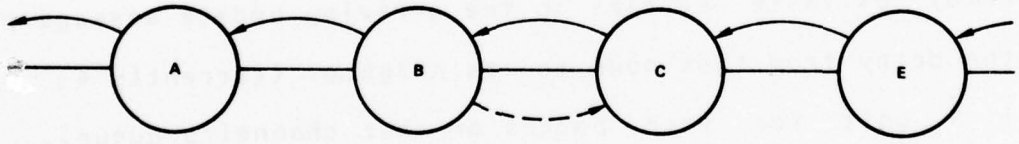


Figure 2.2. A portion of a network.

The following sequence may occur if B's delay estimate to D increases rapidly (at the second step in the sequence).

	Node B	Node C
B->C	t/A	t+4/B
A->B	t+9/A	
C->B	t+8/C	

B sends an update to C which sets up the initial state. Non-empty queues are formed within A and B (B in A's direction and A away from B) when A sends an update to B. This increases B's delay estimate by 9 units. C then instructs B (based on B's old delay estimate) that the delay via C is now smaller. This causes B to point the route toward C, but C still points toward B. Therefore a loop is created.

2.2.1.2 Hold-down routing (HDR)

This kind of loop was eliminated by the addition of the constraint that if the delay estimate changes by more than 8 units on a given line between updates then the node enters "hold-down" state on that line and will not switch (even though the delay estimates may be better in another direction) for approximately two seconds. We will not attempt to explain this further than to say that it allows for news of drastic changes to propagate through a part of the network before routing changes are made. For further details see [McQu 74].

Let us consider what may happen when B's delay estimate to node D gradually increases. We assume that there are twice as many routing messages from C to B as from B to C. This is possible because the rate at which updates occur is based on line speed and line utilization. For a 50 kbps line they occur at a minimum rate of one every 640 msec. and a maximum of one every 128 msec. (See Chapter 4 for a more complete description.) We denote line hold-down by an

exclamation point in the following sequence.

	Node B	Node C
B->C	t/A	t+4/B
A->B	t+5/A	
C->B	(no change)	
A->B	t+9/A	
C->B	t+8/C	
B->C		t+13/B!
C->B	t+17/C!	

This loop (called a loop-trap in [Klei75a]) will remain for a period of approximately two seconds (the hold-down time)! Conditions similar to this have been observed on several occasions, and account, at least in part, for the degradations suffered during experiments with stream communication in the ARPANET [Cohe 74], [Nayl 74], [Forg 75].

2.2.1.3 Modified hold-down routing (MHDR)

Notice that the example of a loop under HDR is a result of the variable rate updating. After loop-traps were observed [Nayl 75], the hold-down scheme was modified in 1975 to make the criteria for entering hold-down independent of update rate. That is to say whenever a delay estimate changes by more than eight units over an arbitrary number of updates then hold-down is entered. This means that a gradual build up of traffic can cause hold-down to occur. In fact hold-down would occur prior to many routing changes.

This change may have produced a loop-free routing algorithm. (We shall not attempt a proof or counterexample here.) Surely some (possibly many) desirable route changes are eliminated or delayed by such a scheme however.

2.2.2 The TIDAS NET procedure

This loop phenomenon is not unique to the ARPANET, as we now show. The entries in the routing table proposed for the TIDAS network in [Cegr 75] contain a delay estimate for each of the node's channels to a destination. The early ARPANET had such tables as well until the size of the network prohibited them. When a packet arrives, the routing procedure chooses the channel with the smallest delay estimate (to the packet's destination) in the table, and routes the packet over that channel. Updates of the table are done in two ways:

a) "inside" (i.e., using only local information: queue lengths and previous delay estimates).

b) "outside" (i.e., using information contained in a routing message from one's neighbor).

The routing message contains a delay estimate vector which gives the delays from the sending neighbor but not through the receiver of the routing message. That is, when A sends a routing message to B, the second best delay estimate is sent for those destinations whose best delay channel is toward B. This method is referred to as "split horizon" updating in [Cegr 75]. We shall use a two-component notation

similar to the above with the network structure shown in Figure 2.2. The following may occur with the stated algorithm.

	Node B	Node C
	t/A, t+12/C	t+2/B, t+10/E
A->B	t+13/A, t+12/C	t+2/B, t+10/E

This loop may be eliminated with the next routing message, or it may set up a chain of loops in the direction of E. We are merely pointing out here that loops form easily under this routing procedure. In fact, in simulations of this procedure "ping ponging" was clearly present. According to Cegrell [Cegr 75] ping-ponging was significantly reduced by the split horizon technique and immediate updating in the case of node or line failures.

Although loops involving more than two nodes are possible with both of the above schemes, we will not present any such examples. Multi-node loops cause a much larger degradation in network performance than do 2-node (ping-pong) loops, but their likelihood of occurrence appears to be significantly smaller than that of the two-node loop. It appears to be impossible to prevent these loops when each node has only local information. The feasibility of each node carrying and adequately updating global information is questionable.

In the next section we examine the magnitude of the degradation in the ARPANET caused by the occurrence of

loops.

2.3 Degradation due to loops

Having shown that loops can occur, we now examine the degradation to network performance caused by loops in terms of the length of time they naturally persist. That is to say, we wish to determine how long it takes to eliminate a loop under the current ARPANET routing update rates. This persistence time is of interest since, if loops disappear quickly, then there would be no need for prevention or rapid detection and removal. We first define the following quantities.

p = Routing update period.

n = Number of nodes in the loop.

l = Total local delay estimate in the loop (in the current implementation l = 4*n, assuming the absence of packets traversing the loop)

d = minimum delay estimate difference over all neighbors outside the loop.

k = Processing time for a routing message.

x = Transmit and propagation delay for a routing message.

y = k + x.

The loop will be cleared whenever a routing update, from outside the loop, arrives and results in a smaller delay estimate directing packets out of the loop. This can happen as quickly as it takes to process an arriving routing

message, if delay estimates outside the loop decrease quickly. Thus the minimum breaking time is essentially zero. If we assume that the delay estimates outside the loop remain fixed, then we must traverse the loop with routing updates a maximum of ceil(d/l) times (where ceil is defined to be the usual ceiling function, i.e., ceil(A) = the smallest integer greater than or equal to A). This causes the internal delay estimate to be greater than the external delay estimate. The maximum time which a loop can persist, assuming the exterior delay remains fixed, is

$$p + y + n(p + y)\text{ceil}(d/l)$$

To compute a minimum we assume that the nodes are synchronized in such a way that a routing message is sent just after the receipt of one from a neighbor. We obtain for our minimum

$$yn(\text{ceil}(d/l)) + (\text{ceil}(d/l)-1)(\text{ceil}(yn/p)p-yn) + k$$

The first term accounts for the correct number of loop traversals required to increase the internal delay estimate beyond the external delay estimate. The last term accounts for the processing of the one required update from the external node. The middle term accounts for the fact that routing updates may happen no more often than one every p msec over each channel. Table 2.1 shows the removal time for loops of size up to ten nodes, using a value of 25 for x and 15 for k (see Chapter 3) and assuming that the loop is caused by a minimum delay estimate difference so that

ceil(d/l) = 1.

Loop Size (nodes)	p=640 msec Maximum (msec)	p=640 msec Minimum (msec)	p=128 msec Maximum (msec)	p=128 msec Minimum (msec)
2	2040	95	504	95
3	2720	135	672	135
4	3400	175	840	175
5	4080	215	1008	215
6	4760	255	1176	255
7	5440	295	1344	295
8	6120	335	1512	335
9	6800	375	1680	375
10	7480	415	1848	415

Table 2.1. ARPANET loop persistence time

In general routing updates among neighbors are neither synchronized nor completely unsynchronized. Thus the expected loop persistence time lies between the two extremes. It is now clear that loops do not disappear quickly, on the average. Therefore it would be useful to examine methods of prevention or at least rapid detection and removal.

2.4 A direct approach to loop control

Below we describe two loop control algorithms for single entry routing tables (as in the ARPANET). The first is ping-pong or local loop-free and the second is general loop-free.

2.4.1 A local loop-free routing algorithm (LLFR)

The basic idea for eliminating local loops is to ignore one's "old" information. Each node ignores routing information which points in that node's direction. Routing message

entries are marked so that the receiving node knows which entries point toward it and which do not. Only those entries which point away are used in the update. More precisely; we add to each entry in the routing update message the best delay channel number c(j,d), and a neighbor table n is appended to the routing update. An entry n(j,k) gives the address of the neighbor connected to j on channel k. (Entries could be marked with a single bit, but this requires the sending of a different routing message to each neighbor.) Each node i performs the following algorithm:

```
LLFR
1     m = k such that n(j,k) = i
2     for each d
3       if c(j,d) ≠ m then do
4         t'(i,d) = t(j,d) + q(j,d) + h
5         if t'(i,d) < t(i,d) then c(i,d) = c'(i,j)
6         if c(i,d) = c'(i,j) then t(i,d) = t'(i,d)
7       end
```

In a later section we prove that this algorithm can create no local loops.

2.4.2 A loop-free routing algorithm (LFR)

The loop-free routing algorithm consists of LLFR at one- and two-connected nodes with the addition of the following for three- (or more) connected nodes (where a node's connectedness is defined to be the number of directly

connected neighbor nodes). When a routing update indicates that a change should occur at a three- (or more) connected node a "loop-check" control packet (LCP) is sent, over the new channel, one for each destination which would be effected by this route change. The idea being that if the LCP returns to the sender, then a loop would be created by such a routing change. If an end-to-end acknowledgement (RCP) for the loop-check packet returns to the sender, then he updates his table and uses the new route. Otherwise (the loop-check packet, an "I'm-checking" packet (ICP), or nothing returns) he ignores the new route until (possibly) another routing update is received. Loop-check packets are routed through the network in the usual way (i.e., along existing allowed paths) except that at any node which is also checking a new route for that same destination; such a node must send an I'm-checking packet to the source of the loop-check packet and discard the loop-check packet. Routing message entries for any destination site involved in loop-check are specially marked so that the information will be ignored by the receiver of the routing message. Stated more precisely the assimilation of a routing message is done under the following algorithm for each d:

LER routing message assimilation

```
1     if x(i,d) = NULL then do
2       if c(j,d) = m then do
3         if c(i,d) = c'(i,j) then do
4           local loop detected!
5           t(i,d) = t(j,d) + l
6         end
7       end
8       else do
9         t'(i,d) = t(j,d) + l
10        if c'(i,j) = x(i,d) then y(i,d) = t'(i,d)
11        else do
12          if t'(i,d) < y(i,d) then do
13            y(i,d) = 0
14            x(i,d) = NULL
15          end
16          if t'(i,d) < t(i,d) and x(i,d) = NULL then do
17            if i is > 2-connected then do
18              x(i,d) = c'(i,j)
19              y(i,d) = t'(i,d)
20              send LCP to d via c'(i,j) with new seq no
21            end
22            else c(i,d) = c'(i,j)
23          end
24        end
25        if c(i,d) = c'(i,j) then t(i,d) = t'(i,d)
26      end
```

```
27    end
```

where x(i,d) is the test channel for destination d at node i, y(i,d) is the test channel delay estimate to node d from node i, m is defined as in LLFR, and l is the local delay (e.g., l = q(i,j) + h). Some additional processing of a packet is required as it arrives from a channel:

LFR additional packet processing

```
101  if ICP for i then do
102    if current sequence number then do
103      x(i,d) = NULL
104      y(i,d) = 0
105    end
106    discard packet
107  end
108  if LCP then do
109    if source = i then do
110      if current sequence number then do
111        x(i,d) = NULL
112        y(i,d) = 0
113      end
114      discard packet
115    end
116    if destination = i then do
117      send RCP to source
118      discard packet
119    end
```

```
120     if x(i,d) ≠ NULL then do
121       discard packet
122       send ICP to source
123     end
124   end
125   if RCP for i then do
126     if current sequence number then do
127       t(i,d) = y(i,d)
128       c(i,d) = x(i,d)
129       x(i,d) = NULL
130       y(i,d) = 0
131     end
132     discard packet
133   end
134   if packet not discarded then do normal packet processing
```

It is useful to consider how these algorithms would affect the ARPANET. These algorithms appear to fit easily within the current ARPANET system with respect to the following:

1. The same routing message may be sent to each of a node's neighbors.
2. Routing updates fit within the current packet size restrictions.
3. The processing complexity is not significantly increased. (In the case of LLFR, processing is certainly less complex than HDR or MHDR.)

We expect that the frequency of sending of the I'm-checking packet is quite small. The frequency of both of these new types of control messages may only be tested by careful measurement, however. Notice that to avoid race conditions, one must not send a routing update while in the process of updating his own table, and routing messages must not be allowed to cross paths (i.e., sent from A to B and from B to A in the same time span). This is easily dealt with by using a pair of sequence numbers in the routing packets. Node A ignores B's update unless it contains A's last sequence number.

We expect that the performance of LLFR is better than HDR, since it is less complex yet guarantees that no local loops occur (as we shall prove). For LFR, the overhead is much higher. There is some extra computation required when a packet arrives. This added computation is insignificant when compared to the slowness of routing changes and the requirement that network bandwidth be used by the loop-check packets.

Routing changes at three- or more connected nodes cannot occur, on the average, more quickly than about 12 msec per hop in the new route (200 bits in a loop-check packet and 200 bits in an RCP gives 8 msec transmission delay. An additional 2 msec is required for processing of the packets. Add to that about 2 msec in propagation delay per hop.) The average network distance in May 1974 was 6.24 hops, with a

maximum of about 11. Hence, one can estimate that on the average a minimum of 75 msec is required to make a routing change after it has been suggested by one's neighbor. If one assumes that loop-check packets are high priority (i.e., next to go out on the channel) and that a full packet has just started out on the line when the loop-check packet arrives, then the delay increases to about 60 msec per hop. This suggests that the maximum switch time is on the order of 700 msec. Our experience shows that, on the average, a small number (<5) of destinations are involved when a switch occurs. At no time would a three-connected node loop-check more than about 1/3 of the nodes (less than 20 nodes at present) in the network after a routing update. So that between 50 and 220 msec of channel time and 5 to 20 buffers along the way might be utilized.

An alternate method of sending the loop-check packets could be used to trade computation for storage and line utilization. One may send a multi-destination packet which, along with the usual packet header, contains some (4 currently in the ARPANET) words of addressing information. These addressing words consist of one bit for each node in the network. The packet is marked by the sender with a one bit for each destination for which a loop-check is required. Then it is sent over the new channel. Each node which encounters the packet is required to:

1. Send an RCP if its own bit is on;
2. Turn its own bit off;

3. Send an ICP to the source marking those destinations being checked by both this node and the source;
4. Turn off the bits for those destinations being checked by both this node and the source;
5. Discard the packet if all addressing bits are now off;
6. (Possibly) duplicate the packet;
7. Mark the copies with those destinations pointed to by each of the channels as indicated in his routing table; and
8. Forward the packet(s) on their way.

This method would cut some of the channel and storage overhead at the expense of some processing overhead.

Below we present a proof that the algorithms do prevent loops. In a later section we present our experience with these algorithms through simulation.

2.5 Proofs of loop-freeness

There are a number of definitions which facilitate the proving of the loop prevention properties of these algorithms. These definitions help in the understanding of these properties as well. The algorithm may be described by a graph structure in which there are directed and non-directed arcs. The non-directed arcs are the lines between neighboring nodes. The directed arcs represent the direction that a packet to a particular destination would be

sent. There exist a set of operators which: (1) replace directed arcs (routing changes); (2) remove or create non-directed arcs (network failures and recoveries).

We define a network to be a graph G consisting of a set X of nodes related by three relations:

n(eighbor), r(oute), and (te)s(t route)

If x1, is a neighbor of x2 in the graph G then x1 is in the set n(x2) and x2 is in n(x1). A node x is k-connected if and only if there are exactly k distinct nodes in n(x). We will refer to a k-connected node as multi-connected if k>2. The next node in the route to a particular destination xd from x1 is r(x1) . We shall fix the destination xd here, and drop reference to it for simplicity. All succeeding definitions (and proofs) refer implicitly to a particular destination. Loop-check packets from node x are routed to the test route s(x) from x . Note that s(x) = NULL for all nodes except those in loop-check mode. A loop, of order k , is defined to be the following condition:

Condition 2.1:

At some instant, there exists a set Y = (y1,y2,...,yk), k>1, a subset of X, such that xd, the destination, is not in Y and

r(yi) = y(i+1) for i = 1,..., k - 1 and

r(yk) = y1 .

Notice that this is static (for possibly a short time)

condition among the routing tables of several nodes. We do not intend to include in this definition transient conditions under which packets may actually loop (e.g., line or node failures).

A routing transformation T on G changes the route or test route of a single node pair. i.e.,

T(G) is defined as follows:

a) T(r(x)) = r(x) for x not = x0,
   T(r(x0)) = r'(x0) for some node x0, and
   T(s(x)) = s(x) for all nodes x; or

b) T(r(x)) = r(x) for all x not = x0,
   T(r(x0)) = s(x0) for some node x0,
   T(s(x)) = s(x) for x not = x0, and
   T(s(x0)) = NULL; or

c) T(r(x)) = r(x) for all x not = x0,
   T(s(x)) = s(x) for x not = x0, and
   T(s(x0)) = NULL for some node x0; or

d) T(r(x)) = r(x) for all nodes x,
   T(s(x)) = s(x) for x not = x0, and
   T(s(x0)) = s'(x0) for some node x0.

In case (a) the route from node x0 is changed to a neighbor which is not the test route (which can happen only at one- or two-connected nodes). In case (b) a new route from node x0 is chosen to be the test route following a successful

test. In case (c) the test route from node x0 is dropped following an unsuccessful test. In case (d) a test route from node x0 is established. (Routing transformations which would occur simultaneously in an operating environment will be ordered by node number, say, and be executed sequentially here.) We claim that LLFR and LFR have T with the following property.

Property 2.1:

a. If r(x) = y, then T(r(y)) not = x;

b. if r(x) = y, then T(s(y)) not = x;

c. if s(x) = y, then T(r(y)) not = x; and

d. if s(x) = y, then T(s(y)) not = x.

Proof of Property 2.1:

LLFR

Property 2.1 (a) is guaranteed by step 3 in the LLFR algorithm. Property 2.1 (b), (c) and (d) are vacuously satisfied by LLFR since no test routes are ever established.

LFR For nodes y which are two- (or less) connected no test routes are ever established, thus proving parts b and d. Part a is guaranteed by step 8 in the LFR algorithm. Part c is similarly guaranteed by step 1. For nodes y which are three- (or more) connected parts b and d are guaranteed by steps 8 and 1 respectively. Multi-connected nodes create actual routes from successful test routes. During the testing of a route by a node, his neighbor (along that route) is restricted from establishing a new route opposite to the

test route by step 1. Hence parts a and c are true for multi-connected nodes as well.

Q.E.D.

Lemma 2.1: Loops of order 2 cannot occur as a result of the application of LLFR or LFR.

Proof: This is a trivial consequence of Property 2.1. Suppose that there exists a loop of order 2 after the application of some T with Property 2.1. Then there exist neighbors x and y such that:

$$T(r(x)) = y \text{ and } T(r(y)) = x \qquad (*)$$

In this case, there are 4 possible states prior to the application of T:

1) r(x) = y and r(y) = x,
2) r(x) = y and r(y) not = x,
3) r(x) not = y and r(y) = x, and
4) r(x) not = y and r(y) not = x.

State 4 cannot exist since T may modify only one r (this is where the synchronizing of neighbors is important in an operating network). State 1 is a previously existing loop and hence was not caused by T. States 2 and 3 are symmetric. Thus we consider only state 2. Since T has Property 2.1, and r(x) = y , then T(r(y)) not = x , in contradiction to the supposition at the beginning of the proof.

Thus (*) cannot occur.

Q.E.D.

We have proved the following theorem.

Theorem 2.1: The LLFR algorithm is free of local loops.

We now concentrate on LFR.

Lemma 2.2: A one-connected node cannot participate in a multi-node loop (i.e., a loop of order k>2).

Proof: A one-connected node has only one neighbor. If a one-connected node participates in a loop it must be of order 2.

Q.E.D.

Lemma 2.3: Loops of order greater than 2 cannot be caused by a routing change at a two-connected node.

Proof: Suppose a loop of order $k > 2$ is created by a transformation at a two-connected node y. The loop then satisfies Condition 2.1 in the following way:

$$T(r(x1)) = r(x1) = x2$$

$$\vdots$$

$$T(r(xi)) = r(xi) = y$$

$$T(r(y)) = r'(y) = x(i + 1)$$

$$\vdots$$

$$T(r(x(k - 2)) = r(x(k - 2)) = x(k - 1)$$

$$T(r(x(k - 1)) = r(x(k - 1)) = x1$$

Also we have that r(y) not = x(i + 1). In fact, since y is two-connected we have r(y) = xi . But r(xi) = y . This is a pre-existing loop of order 2 which by Lemma 2.1 cannot occur.

Q.E.D.

Lemma 2.4: Loops of order k > 2 cannot be broken by a two-connected node.

Proof: Suppose a loop of order k > 2 contains a node of order 2. Any routing change at that node alone would be a violation of Property 2.1.

Q.E.D.

Lemma 2.5: If a routing transformation causes a loop to occur then the node x0 , for which T(r(x0)) not = r(x0), is contained within the loop.

Proof: Suppose after the application of T there exists a loop among a set Y of nodes. If x0 is not in Y , then for all y in Y we have T(r(y)) = r(y) , which implies that the loop existed prior to the application of T. Hence, x0 must be in Y .

Q.E.D.

Lemma 2.5 shows that the loop-check packet has the possibility of being returned to its sender, if a loop would have been caused by that routing change.

Theorem 2.2: Loops cannot occur as a result of LFR.

Proof: We have previously proved that 2-order loops do not occur (Lemma 2.1) and that k-order loops for k > 2, are independent of (i.e., cannot be caused or removed by) two-connected (or one-connected) nodes (Lemmas 2.2, 2.3 and 2.4). We shall now show that loops of order k > 2 cannot be caused by transformations at multi-connected nodes. Suppose that we restrict the number of concurrent multi-connected node transformations to one. That is to say, only one node

may test any particular destination. Suppose that a loop would be created by a transformation at node x, a multi-connected node. Since x is multi-connected it is required to test the new route with a loop-check packet. There are four possible outcomes from the sending of this loop-check packet:

1. The loop-check packet returns,
2. An I'm-checking packet returns (If some other node is checking),
3. Nothing returns (If the message is lost or becomes trapped in a loop independent of x, should one occur), or
4. An RCP returns.

In cases 1 through 3 no switch will be made in the route from x, and hence no loop created. In case 4 we have the fact that the loop-check packet from x reached its destination. Since no other routing changes occur, any of x's packets will also reach the destination by following the exact route that the LCP took. Therefore no loop exists. Hence a routing transformation at only one multi-connected node cannot cause a loop.

We now proceed by induction on the number of concurrent multi-connected node routing switches. Assume that n-1 or less concurrent switches cannot cause a loop. Suppose n concurrent switches have caused a loop. Consider the activity of node x, that node which is last to start loop check. Again there are four possible responses to x's LCP.

They are as listed above. If 1, 2 or 3 occurs then x will not switch routes which contradicts the supposition that n concurrent switches caused the loop. Therefore we need only consider response 4. There are two possible ways in which this may occur.

(a) x's LCP arrives at none of the other n-1 nodes involved, or

(b) x's LCP arrives at (at least) one of the other n-1 nodes after that node has completed switching.

In case (a) the switch at x must be independent of the loop. No switch occurs at any of the nodes which x's LCP traverses. Therefore x cannot be part of the loop, since all packets from x will reach the destination as did the LCP. Therefore the switch at x could not have been required to form the loop. That is the loop would have been created whether or not x switches. This contradicts the supposition again. In case (b) we have that the loop could have been caused by less than n concurrent switches, which contradicts the induction hypothesis. Hence there exists no n such that n concurrent transformations at multi-connected nodes causes a loop.

Q.E.D.

An adaptive routing procedure should not be vulnerable to network component failures (i.e., line or node outages). We have not shown that the algorithm is invulnerable to network failures. It is clear, however, that a loop cannot be

caused by a failure (i.e., the removal of an arc). A recovery cannot cause a loop either because when a line comes up it has no direction in a path. Therefore, no loops are created by a failure or a repair. However, on failure (or repair) some routes must change. This may cause flooding of the network with LCP's which is clearly one of the undesirable features of LFR.

2.6 Simulation results

In order to examine the performance of these routing procedures we here present the results of simulation. The simulation was performed under the following conditions:

Topology: ARPANET (June 1975) modified to exclude satellite links.

Lines: ARPANET capacities (mostly 50 kb/s, some 230.4 kb/s).

Nodes: Infinite storage, two processor speeds (316 - 1.6 usec/cycle and 516 - .96 usec/cycle).

Traffic pattern: Uniform traffic matrix (i.e., same intensity between all pairs of nodes). Arrivals to the network form a Poisson process. Exactly the same traffic was generated for each of the several routing schemes examined.

Message length distribution: Exponential truncated at 8064 bits, with a mean of 122 bits. (The mean is one half of that reported in [Klei 74] to allow for RFNM's which have zero text bits. By reducing

the average message size we attempt to more closely model the actual packet size distribution which consists primarily of data packets and RFNM's in the ARPANET.)

The results of simulation are shown in Figure 2.3. Network wide mean (one way) delay is plotted as a function of relative offered traffic intensity (load) in this figure. We observe that all algorithms produce the same network wide mean delay up to an offered traffic load of 5 (500 packets per second). At 10 we see that LPR blows up (i.e., average delay > 500 msec). (Actually the simulation failed to reach equilibrium at this load.) This is due to the creation of many loops. Also at load 10 HDR and LLFR perform better than LFR. (At this load no multinode loops are yet prevented). The most interesting result is that while both HDR and LFR blow up at a load of 14, LLFR continues with finite delay to at least 16 (the highest load simulated). LLFR exceeds the performance both in terms of delay and throughput at all levels of offered load. This is not too surprising since multinode loops occur very infrequently. Therefore the overhead which LFR requires to guarantee loop-freeness is ill spent in the topology and traffic pattern of the simulation. Since local loops may (do) occur with HDR, it performs worse than LLFR which prevents such loops.

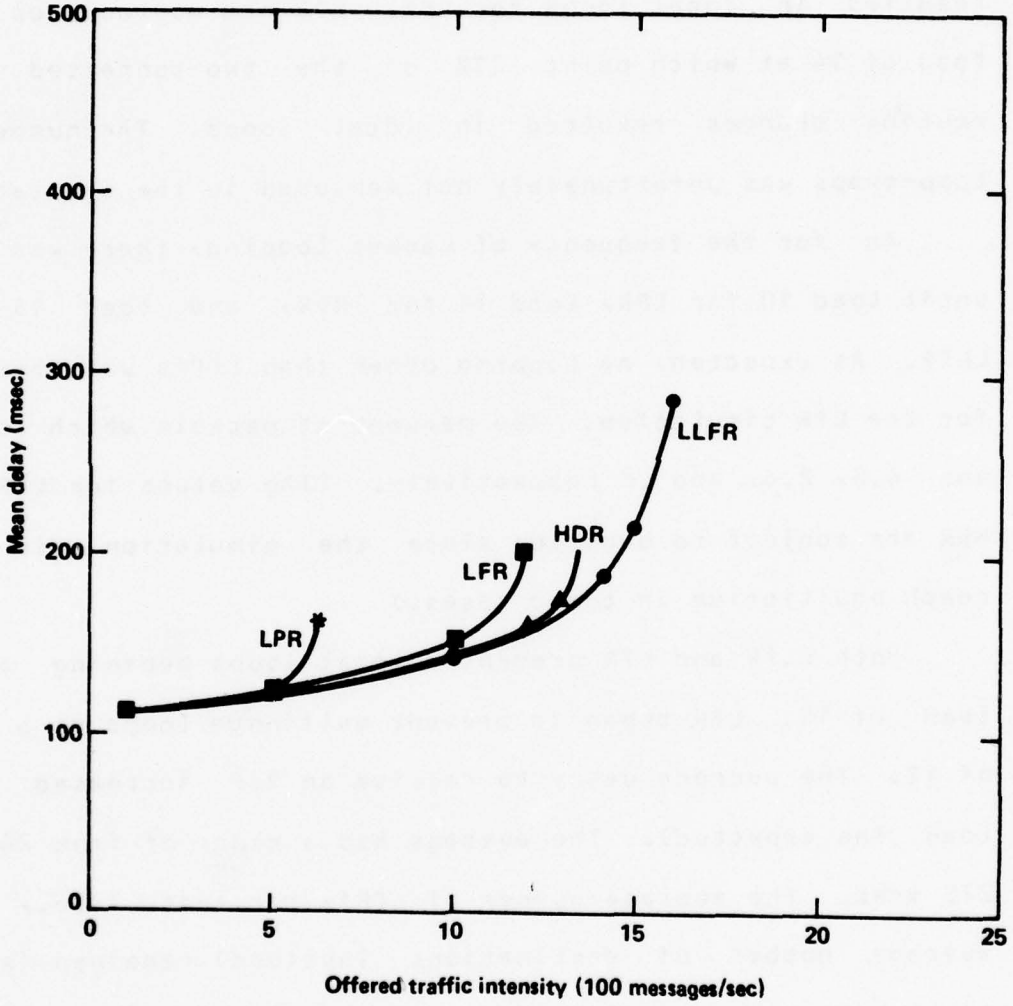


Figure 2.3. Comparative performance of adaptive routing procedures (simulation).

There were no loops observed below a load of 10 in this simulation. In different topologies (and non-uniform traffic rates) it is possible to create loops at lower loads. At load 10, 30% of the routing changes at two-connected nodes resulted in local loops for LPR. HDR had no loops up to a load of 14 at which point 21% of the two-connected node routing changes resulted in local loops. The number of loop-traps was unfortunately not measured in the simulation.

As for the frequency of packet looping, there was none until load 10 for LPR, load 14 for HDR, and load 15 for LLFR. As expected, no looping other than LCP's was observed for the LFR simulation. The percent of packets which looped was 4.8, 2.6, and .8 respectively. (The values for LPR and HDR are subject to question since the simulation did not reach equilibrium in these cases.)

Both LLFR and LFR prevented local loops begining at a load of 10. LFR began to prevent multinode loops at a load of 12. The average delay to receive an RCP increased with load (as expected). The average had a range of from 246 to 275 msec. The average number of LCP's per test (i.e., the average number of destinations involved) remained fairly small but increased with load from a minimum of 1.1 to a maximum of 2.8. This suggests that multiaddressed LCP's would not help efficiency significantly. It also suggests that LFR has the nasty habit of increasing its overhead when the network is heavily loaded.

In the simulations we have assumed that each of the

routing update schemes requires the same amount of processing as does HDR. This is clearly not the case indicating that at least for LLFR we have shown a lower bound on performance with respect to HDR. It should be noted that LLFR and LFR eliminate the need for performing any computation (beyond the initial test) for a fraction 1 - 1/k of the entries in the routing table on the average, where k is the connectivity of the neighbor. The issue of processing overhead is discussed more fully in the next chapter.

2.7 Conclusions

Our intuition, which is now supported by simulation, tells us that multi-node loops occur very infrequently in a "reasonable" topology and traffic pattern. Therefore, since LFR is so complicated, its practical worth is questionable. However, it is interesting to know that such a scheme exists.

Local loops, on the other hand, occur much more frequently and in fact were observed in the ARPANET as well as in simulation. Therefore LLFR is of practical significance. Indeed the simulation results indicate that LLFR is the best among the procedures studied so far. In Chapters 3 and 4 other routing procedures are investigated. We defer our recommendations regarding routing until after Chapter 4.

CHAPTER 3

SYSTEM PRIORITIES

3.1 Introduction

The inclusion of levels of priority within a system allows that system to give different levels of service to different classes of customers. This is a useful property of a system which is to support a variety of activities simultaneously. The ARPANET, for example, was originally designed to support both interactive (terminal-to-computer and computer-to-terminal) traffic as well as moving large amounts of data from place to place in the network (computer-to-computer communication). BBN has thoughtfully provided a priority structure within the network nodes. In this chapter we partially examine that priority structure and its effect on performance. The result of this examination points out the general rule that careful consideration must be given to priority at all points of service within the system in order to guarantee the desired level of service to each customer class.

Initial analysis and design of packet switched networks assumed that the nodal processing time was small and fixed [Klei 64], [Fran 70], [Hear 70], [Klei 70], [Fran 72] and others. Often, in fact, this contribution to delay has been neglected in comparison to the time spent waiting for and

using the channels in the network. In short, the transmission lines were the bottleneck in the system. The ARPANET in fact was designed with this underlying assumption. (One must in general attempt to fully utilize the most expensive component in a system.) Here we present measurements which demonstrate that nodal processing time is no longer small or fixed in the current ARPANET. Nodal processing delay is in fact a significant portion of the overall network delay. This need not be the case. The problem is one of assigning the proper priority to tasks within the processor. This we show through the use of analysis and simulation.

3.2 Measured Results

A large increase in network wide mean round-trip delay was noticed between the "weeklong" data collections of August 1973 reported in [Klei 74] and May 1974. While the traffic characteristics did not change significantly, delay increased from 93 to 249 msec. This delay (while fairly small) exceeds the ARPANET specification of 200 msec and has remained at about this level even as late as March 1977 when another collection revealed a mean delay of 228 msec. Table 3.1 summarizes the information derived from the three weeklong collections. There are three factors (listed in the table) which would naturally have caused an increase in delay between August 1973 and May 1974. Notice that in May 1974 (a) messages contained slightly more packets on the average, (b) messages traveled longer distances (in hops) on

the average and (c) the mean channel utilization (with overhead) was larger than in August 1973. The reason for the sharp increase in channel utilization is due to the fact that the frequency of routing updates was increased approximately by a factor of five. All of these three factors are accounted for in the model of [Klei 74]. Yet the model is unable to predict all of the increase. While actual delay increased by 168% and 145%, the model's predictions increased by only 95% and 44% from August 1973 to May 1974 and from August 1973 to March 1977 respectively. This is due (at least in part) to the fact that the model of [Klei 74] assumed a small and fixed nodal processing delay which is no longer true.

	August 1973	May 1974	March 1977
Input rate (pkts/sec)	51	44	108
Mean bits/msg	243	234	266
Mean pkts/msg	1.11	1.12	1.18
Mean traffic weighted shortest hop path	3.24	4.46	2.93
Mean channel utilization with overhead	.071	.204	.237
Mean channel utilization without overhead	.0077	.011	.012
Mean measured round-trip delay (msec)	93	249	228
[Klei 74] model delay prediction (msec)	73	142	101

Table 3.1 ARPANET traffic and delay summary

In examining the results of measurement experiments involving the use of packet trace [Klei 76a], it becomes clear that the processing delay has a measured value which is much larger than the available estimates. Some examples follow. Table 3.2 shows the results of an experiment in which packet trace was collected with a frequency of 256 (i.e., a trace block was generated for every 256th packet traversing a node) from several nodes in the ARPANET. The data was collected during two approximately 20 hour intervals in March 1975. Statistics for individual output lines are included (though one would not expect processing delay to vary with output line). The first six column headings are clear, but the last three require some explanation. Smpl-Size (sample size) refers to the actual number of packets used to arrive at the previously listed statistics (i.e., Mean, SDev, Min, Max). The mean, standard deviation, minimum, and maximum are expressed in milliseconds (msec). Only store-and-forward packets which did not originate at the From-Node may be included in the Smpl-Size, as packets from a HOST do not have their "time-in" recorded properly and thus are excluded from the statistics. They are, however, included in the To-Ch-Tot (to channel total) which gives the total number of observed packets to be routed on that output channel. The To-Task-Tot (to Task total) entry is the total number of observed packets (store-and-forward and reassembly) which were processed by the "Task" [McQu 72] routine in the IMP.

From Node	To Node	Mean	SDev	Min	Max	Smpl Size	To-Ch Tot	To-Task Tot
6 MIT	31 CCAT	2.97	4.20	.4	36.1	3349	3724	8958
6 MIT	47 WPAT	3.12	4.46	.4	36.4	3260	4001	8958
6 MIT	44 MIT2	3.36	4.68	.4	28.3	230	402	8958
5 BBN	31 CCAT	3.10	4.51	.4	33.9	2852	3119	8065
5 BBN	50 RCC	3.06	4.21	.4	25.0	1345	1746	8065
5 BBN	49 RCCT	3.15	4.39	.4	30.4	1581	2149	8065
14 CMU	38 PURD	3.61	4.73	.4	32.6	1959	2329	5930
14 CMU	27 BELV	3.13	4.60	.4	33.3	2071	2165	5930
14 CMU	18 RADT	3.31	4.74	.4	23.8	645	674	5930
29 ABRD	46 RUTT	2.67	3.78	.4	30.6	1592	1610	4720
29 ABRD	19 NBST	2.67	3.67	.4	23.3	1691	1693	4720
29 ABRD	27 BELV	3.02	4.00	.4	23.9	1346	1396	4720
11 STAN	16 AMES	2.14	3.15	.4	26.0	2027	2484	5020
11 STAN	22 ISI	2.07	2.96	.4	23.4	1935	2180	5020
22 ISI	11 STAN	3.43	5.39	.4	38.1	572	1437	9096
22 ISI	48 AFWT	3.59	4.60	.4	27.3	544	1311	9096
22 ISI	52 ISIT	3.46	4.83	.4	26.6	577	3683	9096
1 UCLA	8 SDC	2.81	4.13	.3	29.5	1210	1486	4346
1 UCLA	3 UCSB	2.90	3.78	.4	20.7	937	1001	4346
1 UCLA	35 UCSD	2.73	3.72	.3	22.6	1121	1598	4346
16 AMES	15 AMST	4.19	5.81	.3	40.6	983	1282	4983
16 AMES	45 MOFF	3.71	4.84	.4	26.8	1284	1403	4983
16 AMES	36 HAWT	3.76	5.16	.4	31.6	410	463	4983
16 AMES	11 STAN	4.18	5.40	.4	31.5	1427	1543	4983
2 SRI	51 SRI3	3.02	4.08	.4	25.5	578	1069	4852
2 SRI	32 XROX	2.96	4.15	.4	27.6	1113	1301	4852
2 SRI	21 LLL	2.89	4.20	.4	24.8	1041	1728	4852
10 LL	44 MIT2	1.80	2.77	.4	14.3	73	119	265
10 LL	18 RADT	2.46	4.00	.4	26.4	81	94	265

Table 3.2(a). Observed processing delay in msec
(516 IMPs)

From Node	To Node	Mean	SDev	Min	Max	Smpl Size	To-Ch Tot	To-Task Tot
34 LBL	21 LLL	11.5	11.3	.7	62.2	3485	3492	10147
34 LBL	4 UTAT	10.6	10.8	.7	67.7	4867	4956	10147
34 LBL	45 MOFF	12.2	11.5	.7	64.0	1477	1610	10147
4 UTAT	34 LBL	17.1	30.7	.7	323.0	4769	5017	10534
4 UTAT	12 ILL	17.1	31.2	.7	333.2	4693	5102	10534
43 TYMT	32 XROX	18.2	16.8	.7	90.9	274	2724	5200
43 TYMT	33 FNWT	18.2	17.1	.7	71.2	222	630	5200
23 USCT	8 SDC	12.2	13.1	.7	76.2	800	1647	4559
23 USCT	25 DOCT	12.6	13.0	.7	81.6	990	1559	4559
25 DOCT	23 USCT	10.8	11.7	.7	62.9	1041	1054	2096
25 DOCT	24 GWCT	11.4	11.7	.7	62.8	1020	1038	2096
28 ARPT	20 ETAT	9.70	11.1	.6	54.7	351	644	1114
28 ARPT	17 MTRT	10.6	10.6	.7	50.6	287	308	1114
27 BELV	14 CMU	9.88	10.0	.7	57.8	1247	1300	2791
27 BELV	26 SDAC	9.94	10.8	.7	52.8	391	391	2791
27 BELV	29 ABRD	9.18	9.94	.7	51.9	1061	1073	2791
44 MIT2	6 MIT	5.47	6.64	.7	32.5	113	219	391
44 MIT2	10 LL	4.67	5.87	.7	26.5	125	127	391

Table 3.2(b). Observed processing delay in msec
(316 IMPs & TIPs)

Only a portion of the total processing delay is measured by the trace mechanism, viz: (a) part of the Modem-to-IMP routine processing time, (b) the time spent waiting on queue for the Task routine, and (c) most of the Task routine processing time. The sum of (a), (b), an (c) is given by the difference between T(2) and T(1) in the trace block. By the estimates in [McQu 72] this value would have a minimum of less than 150+250=400 cycles (approximately .38 msec for a 516 IMP and .64 msec for a 316 IMP). The observed minimum values confirm these estimates.

Table 3.2 has been separated into two parts with 516 IMPs listed first followed by the 316 IMPs. There is a clear distinction between these processor types (not surprising since one runs at about 1.67 times the speed of the other). (Not as clear a difference was observed between those nodes with VDH [BBN 69] and/or TIP [Orns 72] software and those without.) In both cases we see a wild variation in processing delay, with mean values from 4.5 to 10.5 times the minimum for 516s and from 6.7 to 26 times the minimum for 316s. This phenomenon is nearly independent of time of day as shown in Tables 3.3 and 3.4, which show the observations of two selected nodes over seven time intervals. (The rather odd time intervals correspond to arbitrary file boundaries and provide similar sample sizes.) Notice again the large variation in processing delay and the very high mean values. Notice also that the mean values remain high even under the more lightly loaded evening hours, indicating (but not proving) that this effect is not due to the competition of packets for the services of the Task routine.

For the model's predictions in Table 3.1 we have assumed an average processing time of less than 1 msec per hop. Taking a rough average of the values in Table 3.2 we find that average processing delay for 516 IMPs is about 3 msec and about 11 msec for 316 IMPs. Roughly one third of the IMPs in May 1974 were 516 IMPs. Assuming a uniform spread of the traffic, the average nodal processing delay is more than 8 msec. If we add 7 msec per hop to the predicted

delays, we would expect to obtain a better prediction. Doing so we arrive at a value of 204 msec for May 1974 and 142 for March 1977. For May 1974 this new prediction is slightly better than was the original prediction for August 1973. The March 1977 prediction is not as good. One reason may be the flow control mechanism which was changed significantly between May 1974 and March 1977. Delay is measured in such a way to include the delay due to flow control, while there is no provision for this in the model.

From Node	To Node	Mean	SDev	Min	Max	Smpl Size	To-Ch Tot	To-Task Tot	Time Begin	Time End
22 ISI	11 STAN	4.67	7.12	.4	38.1	118	254	1404	0952	1111
22 ISI	48 AFWT	4.06	4.58	.4	17.5	82	286	1404	0952	1111
22 ISI	52 ISIT	3.00	3.76	.4	15.9	125	484	1404	0952	1111
22 ISI	11 STAN	4.24	5.87	.4	27.3	66	219	1447	1111	1246
22 ISI	48 AFWT	4.13	4.78	.4	17.9	72	334	1447	1111	1246
22 ISI	52 ISIT	3.66	5.83	.4	26.6	52	432	1447	1111	1246
22 ISI	11 STAN	2.50	3.53	.4	13.5	55	205	1245	1246	1421
22 ISI	48 AFWT	3.31	4.51	.4	21.3	82	221	1245	1246	1421
22 ISI	52 ISIT	3.99	4.93	.4	26.1	78	479	1245	1246	1421
22 ISI	11 STAN	3.58	5.40	.4	26.2	50	223	1273	1421	1628
22 ISI	48 AFWT	4.71	5.94	.4	25.3	34	86	1273	1421	1628
22 ISI	52 ISIT	3.55	5.36	.4	26.3	76	581	1273	1421	1628
22 ISI	11 STAN	3.27	5.43	.4	34.0	86	219	1387	1628	1853
22 ISI	48 AFWT	3.50	4.85	.4	27.3	97	141	1387	1628	1853
22 ISI	52 ISIT	2.28	3.28	.4	17.0	54	585	1387	1628	1853
22 ISI	11 STAN	2.37	4.20	.4	30.3	106	174	1517	1853	2325
22 ISI	48 AFWT	2.99	3.92	.4	17.8	104	149	1517	1853	2325
22 ISI	52 ISIT	4.42	6.06	.4	26.4	95	743	1517	1853	2325
22 ISI	11 STAN	3.08	4.15	.4	20.5	91	143	823	2325	0529
22 ISI	48 AFWT	3.29	4.38	.4	25.7	73	94	823	2325	0529
22 ISI	52 ISIT	3.15	4.18	.4	20.0	97	379	823	2325	0529

Table 3.3. Variation of processing delay with time of day (516 IMP)

From Node	To Node	Mean	SDev	Min	Max	Smpl Size	To-Ch Tot	To-Task Tot	Time Begin	Time End
27 BELV	14 CMU	7.07	7.27	.7	29.8	140	153	341	0952	1111
27 BELV	26 SDAC	7.71	7.37	.7	25.8	81	81	341	0952	1111
27 BELV	29 ABRD	8.66	8.72	.7	40.7	98	101	341	0952	1111
27 BELV	14 CMU	10.9	11.0	.7	49.7	165	176	431	1111	1246
27 BELV	26 SDAC	11.5	11.5	.7	38.8	114	114	431	1111	1246
27 BELV	29 ABRD	10.8	10.8	.7	42.7	130	131	431	1111	1246
27 BELV	14 CMU	10.2	11.5	.7	57.8	214	231	485	1246	1421
27 BELV	26 SDAC	12.7	13.1	.7	52.8	92	92	485	1246	1421
27 BELV	29 ABRD	12.6	13.2	.7	51.9	153	155	485	1246	1421
27 BELV	14 CMU	9.24	9.61	.7	45.1	128	132	256	1421	1628
27 BELV	26 SDAC	6.94	7.82	.7	22.1	20	20	256	1421	1628
27 BELV	29 ABRD	7.28	7.90	.7	29.9	102	104	256	1421	1628
27 BELV	14 CMU	10.0	9.69	.7	51.7	188	194	402	1628	1853
27 BELV	26 SDAC	5.09	5.53	.7	16.0	17	17	402	1628	1853
27 BELV	29 ABRD	7.84	7.93	.7	34.7	187	187	402	1628	1853
27 BELV	14 CMU	9.61	9.11	.7	40.6	166	168	356	1853	2325
27 BELV	26 SDAC	12.6	13.2	.7	49.4	27	27	356	1853	2325
27 BELV	29 ABRD	8.38	8.84	.7	43.0	159	161	356	1853	2325
27 BELV	14 CMU	10.9	9.97	.7	41.0	246	246	520	2325	0529
27 BELV	26 SDAC	5.60	6.30	.7	21.0	40	40	520	2325	0529
27 BELV	29 ABRD	8.73	9.83	.7	45.7	232	234	520	2325	0529

Table 3.4. Variation of processing delay with time of day (316 IMP)

In the previously described experiment it was not possible to empirically prove that the excessive delay is not due to the interference of packets, since sampling was done. To establish this, we resort instead to another experiment in which we selected two nodes and collected packet trace with a frequency of one. On close examination of the data we find that for most packets having a long processing delay no other packet overlaps it in time. We note that while

some overflows occurred (i.e., lost data due to heavy traffic) in the packet trace data the statistical properties are nearly identical to those of the sampled data. We conclude that most of the time the excessive delay is not due to the interference of other packets.

At first glance, these long processing delays seem rather insignificant in the entire scheme of things. Let's examine them a bit more closely. Table 3.5 gives the averages for processing delay, waiting time, and acknowledgement time, for three packet types - (1) subnet control (largely end-to-end acknowledgements, i.e., RFNMs), (2) user priority, and (3) non-priority packets. Our interest is with the store-and-forward delays, but we list the reassembly information for completeness. The sample sizes listed consist of two numbers. The first number indicates the sample size for the Channel Waiting and Acknowledgement statistics, and the second refers to the Processing statistics. Notice that the time spent waiting for and being served by the Task routine is on the same order as the time spent waiting for the channel for the 516 IMP and always greater (about double) in the case of the 316 IMP. We note that for these measurements, non-priority packets traversed a 316 node faster, on the average, than both control packets and priority packets (the order of service by Task is FCFS) even though the channel waiting time is greater (as one would expect) for non-priority packets! Examination of more extensive data reveals that 10 of 36 cases exhibit this behavior. In

fact in only 16 of 58 cases, including all node types, was processing delay smaller for both priority and control packets. The experiments which we have conducted in transmitting simulated speech data from the UCLA PDP 11/45 [Nayl 74] as well as similar experiments conducted by the University of Southern California Information Sciences Institute, Marina Del Rey, California (ISI) [Cohe 74] support this observation. In those experiments no discernible difference in delay could be found between priority and non-priority messages! In a report from Network Analysis Corporation, Glen Cove, New York (NAC) [NAC 75] the authors state that a 40% decrease in mean round-trip delay can be achieved by reducing the processing delay to its original estimate of 1 msec. Our own projections (which appear in Section 3.3.5) show that a 44% decrease could be achieved. Therefore processing delay currently accounts for at least 2/5 of the total delay, which is quite significant.

	Store-and-Forward		Reassembly	
	Mean	SDev	Mean	SDev
Control Packets				
Sample Size	(3790,	1116)	(0,	0)
Processing	3.81	5.22	0.0	0.0
Channel Waiting	2.31	10.40	0.0	0.0
Acknowledgement	19.11	15.56	0.0	0.0
User Priority Packets				
Sample Size	(731,	317)	(2098,	2066)
Processing	2.57	3.83	3.36	4.69
Channel Waiting	2.69	14.41	5.87	109.68
Acknowledgement	22.65	15.03	4.36	18.75
User Non-priority Packets				
Sample Size	(1910,	260)	(567,	468)
Processing	3.18	4.95	8.27	27.69
Channel Waiting	3.88	26.64	12.20	34.87
Acknowledgement	22.93	16.87	6.12	11.04

Table 3.5(a). Processing delay statistics
(node 22 a 516 IMP, 14 MAR 75)

	Store-and-Forward		Reassembly	
	Mean	SDev	Mean	SDev
Control Packets				
Sample Size	(1546,	1507)	(0,	0)
Processing	10.43	10.30	0.0	0.0
Channel Waiting	4.89	8.80	0.0	0.0
Acknowledgement	20.73	11.16	0.0	0.0
User Priority Packets				
Sample Size	(575,	575)	(11,	11)
Processing	9.63	10.36	14.82	10.78
Channel Waiting	5.00	8.60	0.36	0.50
Acknowledgement	23.12	10.94	3.64	5.43
User Non-priority Packets				
Sample Size	(643,	617)	(16,	16)
Processing	7.63	9.08	16.38	22.62
Channel Waiting	5.86	12.68	1.56	3.35
Acknowledgement	26.22	12.61	5.13	5.06

Table 3.5(b). Processing delay statistics
(node 27 a 316 IMP, 14 MAR 75)

	Store-and-Forward Mean	Store-and-Forward SDev	Reassembly Mean	Reassembly SDev
Control Packets				
Sample Size	(7777,	2738)	(0,	0)
Processing	3.75	4.78	0.0	0.0
Channel Waiting	2.72	15.23	0.0	0.0
Acknowledgement	21.43	13.04	0.0	0.0
User Priority Packets				
Sample Size	(1328,	423)	(4344,	4326)
Processing	2.92	4.12	4.14	4.98
Channel Waiting	3.07	22.03	0.34	0.61
Acknowledgement	24.16	14.37	2.50	0.97
User Non-priority Packets				
Sample Size	(4884,	2356)	(613,	611)
Processing	3.42	4.64	4.81	5.44
Channel Waiting	5.01	17.04	0.44	0.97
Acknowledgement	27.20	13.66	4.58	3.37

Table 3.5(c). Processing delay statistics
(node 22 a 516 IMP, 24 JUL 75)

	Store-and-Forward Mean	Store-and-Forward SDev	Reassembly Mean	Reassembly SDev
Control Packets				
Sample Size	(4317,	4258)	(0,	0)
Processing	8.12	8.39	0.0	0.0
Channel Waiting	3.33	6.63	0.0	0.0
Acknowledgement	19.74	11.33	0.0	0.0
User Priority Packets				
Sample Size	(1543,	1543)	(4,	4)
Processing	8.47	8.16	3.75	4.86
Channel Waiting	3.56	7.79	0.75	0.50
Acknowledgement	22.24	10.52	2.50	1.73
User Non-priority Packets				
Sample Size	(1737,	1700)	(3,	3)
Processing	6.58	7.85	8.33	7.02
Channel Waiting	4.77	8.[illegible]	1.00	0.0
Acknowledgement	23.42	10.22	1.00	0.0

Table 3.5(d). Processing delay statistics
(node 27 a 316 IMP, 24 JUL 75)

The acknowledgement time is larger than one might expect. This can have only a second order effect on delay. If packets are retained longer than necessary when buffers are in short supply then some packets may have to be

retransmitted. This is not occurring, for if it were the effect would be recorded as waiting time since retransmissions change the "sent time" in the trace block.

3.3 Analysis

To pinpoint the cause of the excessive processing delay we shall use a model of the nodal priority structure. A description of that structure follows.

3.3.1 System description

The IMPs are required to perform a set of functions in order that the network operate smoothly. Among these functions are: receiving, routing and sending packets; processing routing updates and periodically sending routing updates to neighbor nodes. A priority level is assigned to each of the various functions within the IMP. A partial list of the levels appears in Table 3.6. A function is activated by an interrupt mechanism. Only the function associated with the highest active priority level may occupy the processor at a given instant.

Abbreviation	Level	Meaning, Comments
M2I (highest)	0	Modem to IMP, Channel input
I2M	2	IMP to Modem, Channel output
I2H	3	IMP to HOST, HOST output
H2I	4	HOST to IMP, HOST input
T.0	5	Timeout, Periodic functions
TSK	6	Task, Primarily packet routing
BCK (lowest)	7	Background, Statistics, etc.

Table 3.6. IMP priority levels

The portion of the priority structure in which we are most interested here is that governing the processing of routing update packets and other (non-routing) packets. Under the current scheme, pictured in Figure 3.1(a), all packets arrive and are treated by the input routine running at the M2I level which places them, without examination as to type or priority, on the Task queue. They are then processed in first-come-first-served order. When the Task routine encounters a routing packet it, in effect, switches priority to level T.0 so as not to be interrupted by any function of equal or lower priority. (This has the effect of disallowing the sending of a routing update packet to a neighbor while the table is in the process of being updated.) Non-routing packets are, on the other hand, interruptable by T.0 functions. More detailed descriptions appear in [Cole 71] and in [McQu 72].

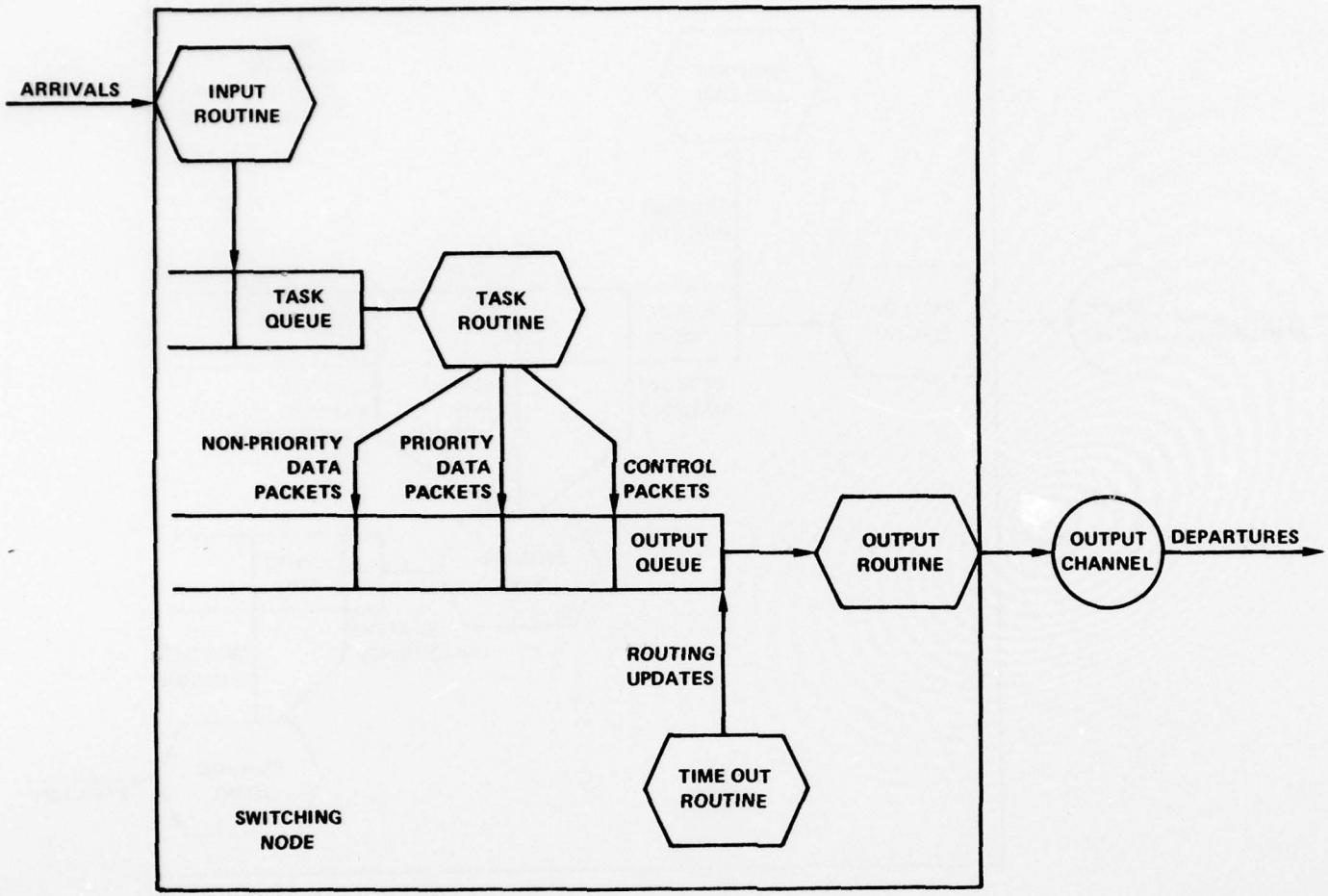


Figure 3.1(a). Simplified model of ARPANET queue structure.

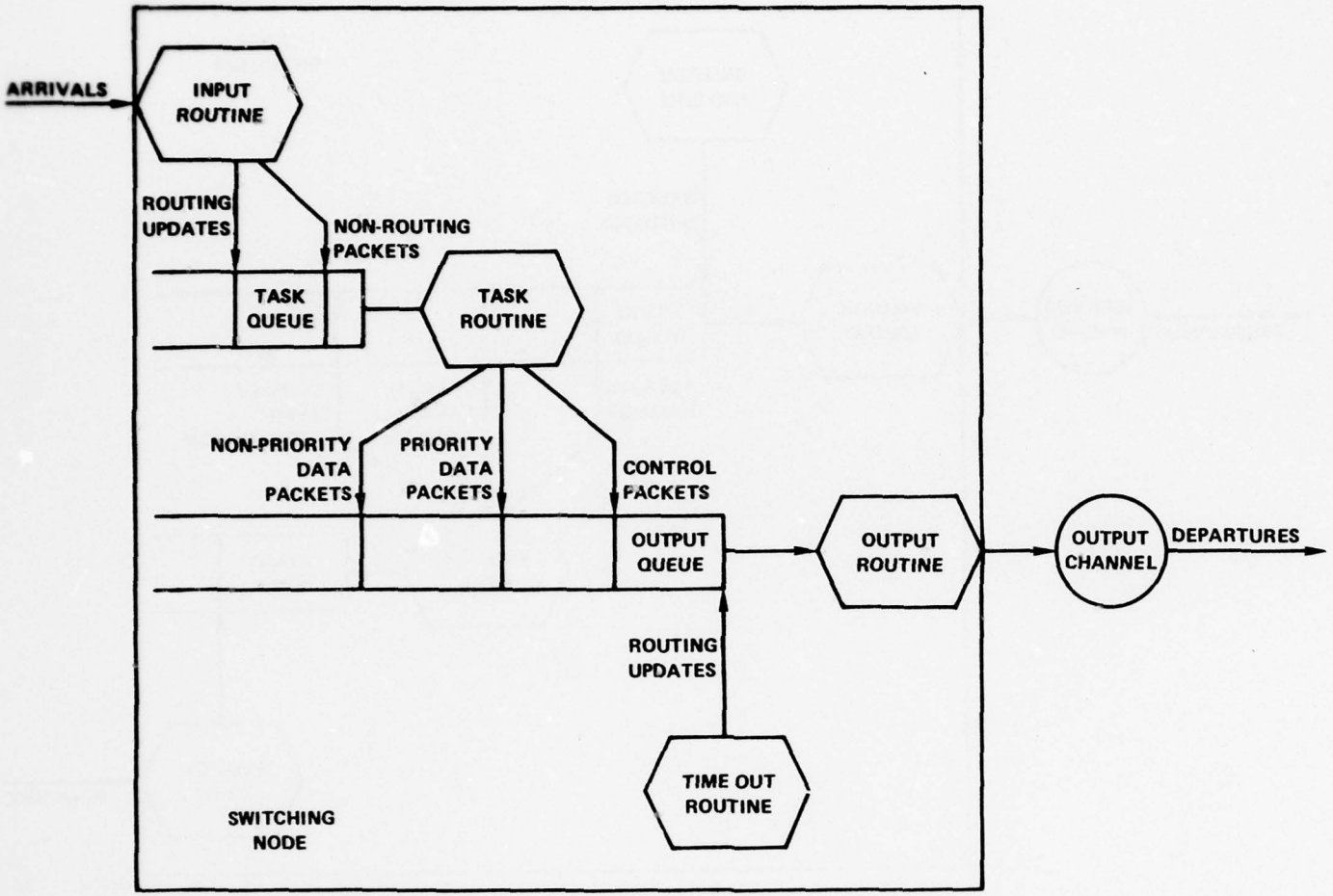


Figure 3.1(b). Simplified model of ARPANET queue structure with priority within the TASK queue.

Since, as we see later, routing update packets require much processing compared with non-routing packets, it seems appropriate to reorder the priorities and treat routing update packets in the BCK level and other packets at the TSK level. Figure 3.1(b) shows the logical structure of such a system. Notice that the Task queue has two levels of priority instead of one as before. Also notice that unlike the channel, the Task routine serves the queue by preempting, if necessary, the (low priority) routing updates. This scheme is not new. Originally routing packets were treated in the BCK level. The chief reason for the current set of priorities coupled with higher frequency updating is to propagate routing information faster through the larger network [Sant 75].

3.3.2 General theory

This system can be modeled by a single server head of the line (HOL) preemptive priority queueing system [Klei 76]. The solution given in [Klei 76] for the mean time in system for priority class p is valid under the M/G/1 assumption (which we later adopt). There are, however, some partial results for the G/G/1 case which we may apply here. Schrage [Schr 68] proves that the shortest remaining processing time first (SRPT) scheduling algorithm is optimal in that it minimizes the average number of jobs in the system. That is, the number of jobs in a system under the SRPT algorithm is less than or equal to the number in the system

under any other scheduling algorithm for the same sequence of jobs. Kleinrock [Klei 76] points out that if the cost of delaying a customer is constant over all priority classes, then the SRPT rule achieves the minimum cost. Hence, the theoretical results indicate that an arrival should join the queue behind all those customers in the system with remaining service time which is less than (or equal to) the service time required of that arrival, and in front of all others in the system. That is all non-routing packets should preempt, if necessary, the processing of routing packets, since as we see below, routing packets require much more processing than do non-routing packets.

Having stated the general theoretical result we now wish to show the order of magnitude of the improvement one may achieve by causing routing updates to be processed at a lower priority than other packets.

3.3.3 A model

In the following analysis we wish to examine the tradeoff between the existing and the proposed system over the full range of packet traffic intensity. Some simplifying assumptions will allow the direct application of the theory. For simplicity, we shall lump all high priority functions into the M2I priority class. We consider two such high priority functions - (a) modem input and (b) modem output; and ignore all others. We shall overestimate the time used in these functions which will tend to give a lower bound on

performance (i.e., a pessimistic value). We assume that the arrival of customers to each class is independent of the arrivals to the other classes and may be modeled by a Poisson process. The Poisson assumption on the arrival of packets is equivalent to assuming that we have a large number of input channels collectively sending at the same rate as the actual number of channels. We further assume that each queue has no restriction as to its length (i.e., infinite nodal storage) and that the overhead for changing tasks is negligible. The current system then, may be modeled by a two level preemptive resume HOL priority queueing system and the proposed system by a three level system. These systems are pictured in Figure 3.2. In each system the input/output function has preemptive priority over the other class(es). The three level system (in part (b) of the figure) divides the Task queue arrivals into routing updates and non-routing packets, the latter having preemptive priority over the former.

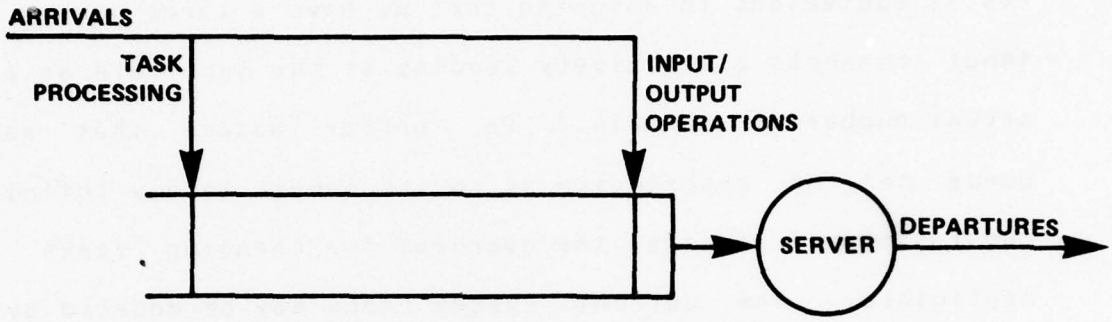


Figure 3.2. (a) The two level model.

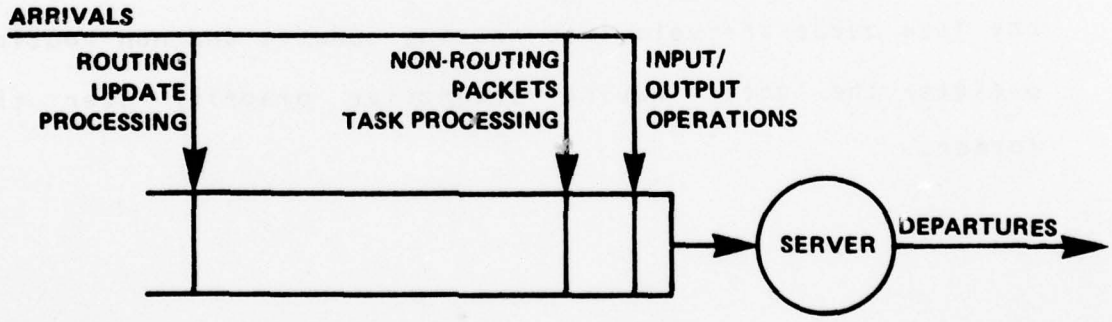


Figure 3.2. (b) The three level model.

3.3.4 Analytic results

The general solution from [Klei 76] for the average time in system for a customer of the pth class is

$$T(p,P) = \frac{\underline{x}(p)[1-s(p)] + \sum_{j=p}^{P} r(j)\underline{x2}(j)/2}{[1-s(p)][1-s(p+1)]} \qquad (3.1)$$

where P is the number of classes (1 is the lowest priority class and P is the highest), $\underline{x}(p)$ is the mean service time for a customer in class p, $\underline{x2}(p)$ is the second moment of service time for a customer in class p, r(j) is the input rate of customers of class j, and s(p) is the cumulative utilization due to classes of priority p and higher (where the utilization for class j is $\underline{x}(j)r(j)$). Let x(i) = the service time for the M2I class, x(u) = the service time for routing update processing, x(n) = the service time for non-routing packets, r(u) = the arrival rate of routing update packets, and r(n) = the arrival rate of non-routing packets. For the purpose of further simplifying the model we shall assume that the service time in each class is fixed. In particular, this means that the second moment is equal to twice the mean. We assume that each packet processed by the Task routine arrives and eventually leaves via the M2I level (since each packet processed by the Task routine must arrive via the input routine and exit via the output routine in our model). We also assume that the number of routing update packets received by a node is equal to the number sent. We

therefore assume (an independent arrival process with) an arrival rate of 2[r(u)+r(n)] to the M?I class. From equation 3.1 we have

$$T(1,2) = \frac{x(u)r(u)+x(n)r(n)}{[r(u)+r(n)][1-2x(i)[r(u)+r(n)]]} + \frac{[x^2(u)r(u)+x^2(n)r(n)]/2 + [r(u)+r(n)]x^2(i)}{[1-x(u)r(u)-x(n)r(n)-2x(i)[r(u)+r(n)]][1-2x(i)[r(u)+r(n)]]} \quad (3.2)$$

Equation 3.2 gives the average time in system for routing and non-routing packets in the two level system. The solution to the three level system is

$$T(1,3) = \frac{x(u)}{1-x(n)r(n)-2x(i)[r(u)+r(n)]} + \frac{\dfrac{r(u)x^2(u)/2 + r(n)x^2(n)/2 + [r(u)+r(n)]x^2(i)}{1-x(n)r(n)-2x(i)[r(u)+r(n)]}}{1-x(u)r(u)-x(n)r(n)-2x(i)[r(u)+r(n)]} \quad (3.3)$$

$$T(2,3) = \frac{x(n)}{1-2x(i)[r(u)+r(n)]} + \frac{r(n)x^2(n)/2 + [r(u)+r(n)]x^2(i)}{[1-x(n)r(n)-2x(i)[r(u)+r(n)]][1-2x(i)[r(u)+r(n)]]} \quad (3.4)$$

Since both systems are preemptive, there is no difference between T(2,2) and T(3,3). Therefore we need only concern ourselves with T(1,2), T(2,3) and T(1,3). T(1,2) - T(2,3) gives the reduction in delay for non-routing packets when changing from the two level system to the three level system. T(1,3) - T(1,2) gives the increase for routing update processing for the same change. Comparison of the two systems depends on the relative values of x(u), x(n), r(u) and r(n) which we shall now estimate in order to present some numerical results.

3.3.5 Numerical results

We estimate that no more than 300 machine cycles are required to process an incoming packet or an outgoing packet. We further assume that a maximum of 300 cycles are required to do the TSK level processing of a non-routing packet. These values are purposely larger than the estimate in [McQu 72]. This is in order to exaggerate their effect in the model and therefore provide a pessimistic projection of performance (as mentioned earlier). Routing packets require much more time to digest. In a recent experiment carried out BBN [BBN 75] with a 316 IMP, it was found that 18% of the HOST throughput was lost when going from zero to one 50kbps line. A similar loss was noted with each additional line. Assuming that the processor was fully utilized during the experiment, this indicates that routing and other

processes associated with IMP-to-IMP lines require 14400 cycles/routing update period. The time could be significantly reduced by implementing the "local loop" prevention algorithm outlined in [Nayl 75] and chapter 2. That is, many (one third on the average for 3-connected neighbors) of the routing update entries are skipped since they point toward the updating node. We estimate those processes involved in handling one line (with no data traffic) as follows:

Function	Cycles/update period
Receive routing message	
Input (140 on 316)	280
M2I	300
TSK	300
Receive and process IHY	100
Send routing packet	300
T.O	300
I2M	300
Output (140 on 316)	280
Process routing packet	12000
Other	540
Total	14400

Table 3.7 Channel function processing time estimates

We shall use for x(u) a value of 12000, which is a conservative estimate to minimize the effect of routing. This conflicts with the pessimistic estimates of input, output and non-routing packet processing and would tend to give an optimistic value of performance (i.e., lower than actual delay). However, it does result in a lower bound (or pessimistic estimate) for the improvement in performance

between the two systems. The service time of the various tasks is a function of the processor speed. In August 1976 there were four processor types in the ARPANET -- the 516 IMP, 316 IMP, 316 TIP, and Pluribus [Hear 76] IMP according to [NIC 76]. We shall consider only the first three which have speeds of .96, 1.6 and 2.2 usec/cycle respectively [McQu 73]. The rate of arrival of routing packets to a node is governed both by the number of connected channels, and the utilization of those channels as mentioned in chapter 2. The three nodal types were distributed with connectivity [NIC 76] as shown in Table 3.8. Also shown in Table 3.8 is the predicted "zero-load" average waiting time expressed in milliseconds for each case (i.e., the waiting time when r(n) = 0). Each pair in the table consists of the number of nodes in the network with this connectivity followed by the zero load delay prediction. At that time there was one five-connected Pluribus IMP in the network.

Connectivity	516	316	TIP
1	0, .63	2, 1.87	3, 3.79
2	5, 1.40	9, 4.57	12, 10.38
3	11, 2.36	5, 8.83	9, 24.93
4	1, 3.61	1, 16.66	0, 86.17

Table 3.8 Connectivity of processor types

The zero load average waiting time for non-routing packets in the three level system is essentially zero. The range is from .000645 to .0130 msec. Comparing these values with

those in Table 3.8 we see the dramatic effect of this change in priorities. The zero load average waiting time for non-routing packets in the two level system is from about 1000 to 6000 times what it is in the three level system!

The behavior of these systems over the complete range of channel utilization (where packets of 300 bits average length occupy the channels) is shown in Figure 3.3 parts (a) through (f). Also shown in the figure are measured data points (as squares) where such data exists. We show only the results for the two and three connected nodes (those which are most prevalent in the ARPANET). Each part of Figure 3.3 consists of three curves. The curve which lies between the two others shows the performance of the two level system and the other two show the three level system's performance. Average waiting time for routing and non-routing packets is shown in the highest and lowest curves respectively. The gain in delay for non-routing packets is the difference between the middle and the lowest curve. The cost in delay for routing is the difference between the highest and the middle curve. For parts (a) and (b) the bottom curve is almost indistinguishable from the axis (i.e., nearly zero waiting time through the entire range). Notice in parts (b) and (d) that the model is a good lower bound to the measured results. Since we overestimated the high level processing time, an upper bound was expected. The fact that a lower bound was achieved is most likely due to the underestimation of $x(u)$.

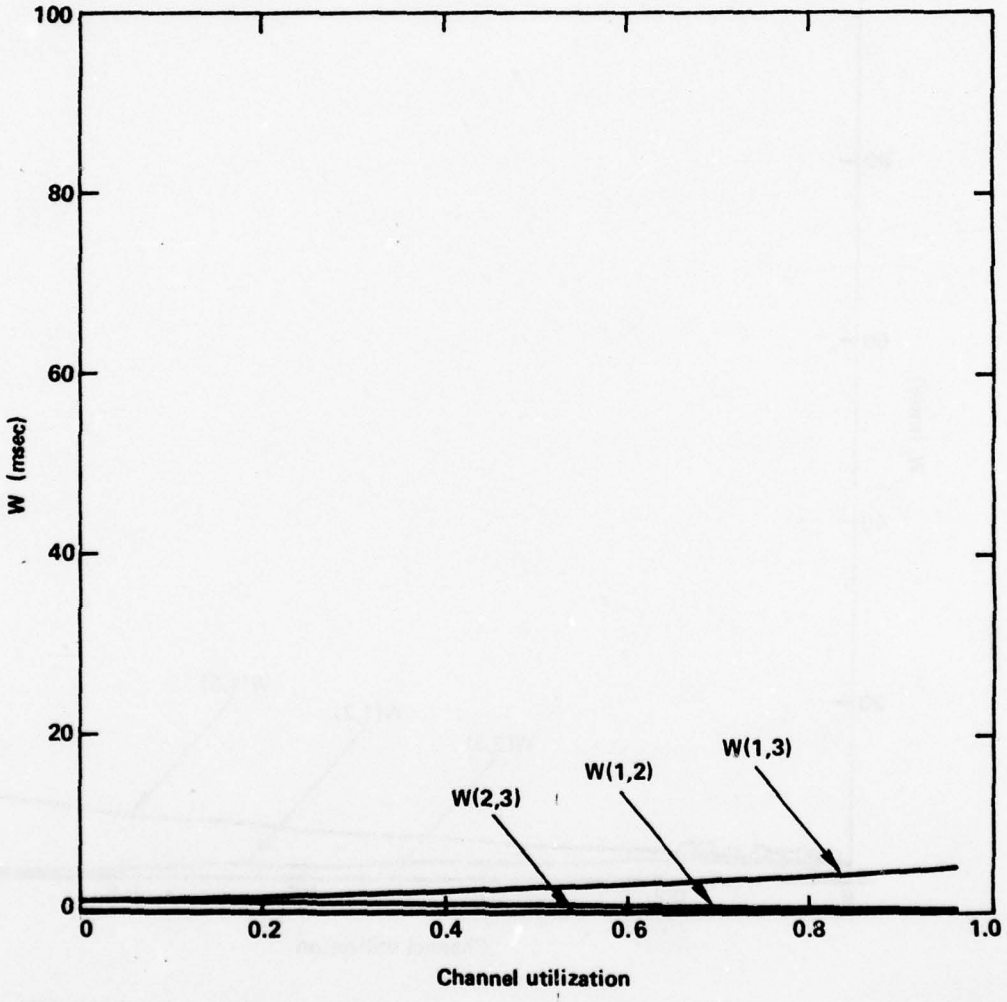


Figure 3.3. (a) Mean waiting time on TASK queue (2-connected 516 IMP).

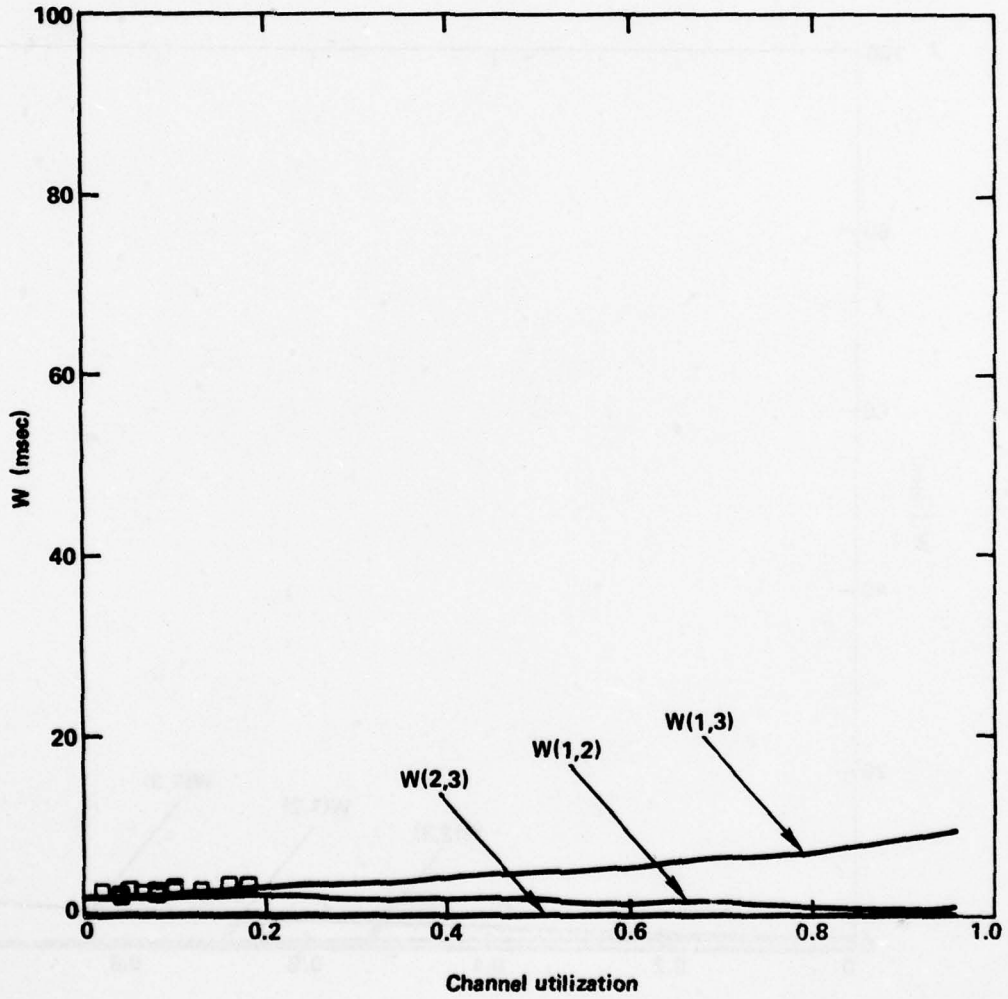


Figure 3.3. (b) Mean waiting time on TASK queue (3-connected 516 IMP).

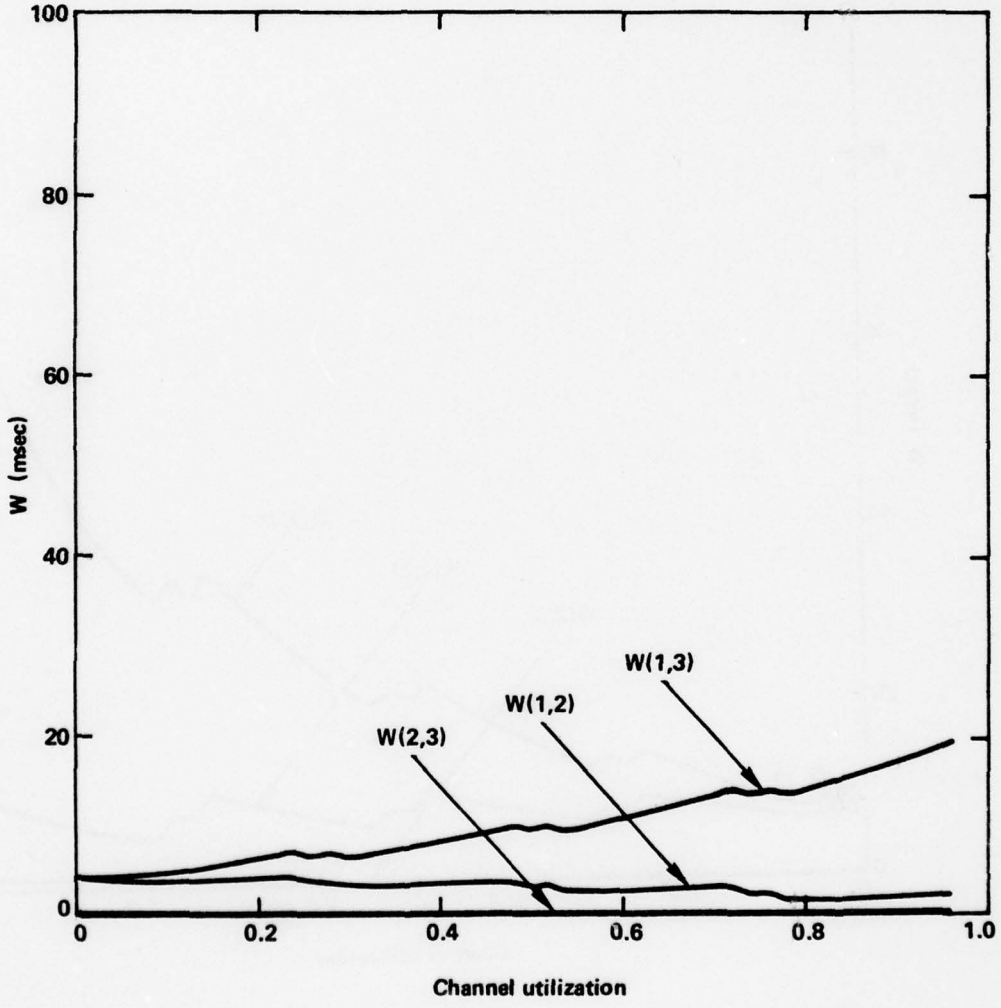
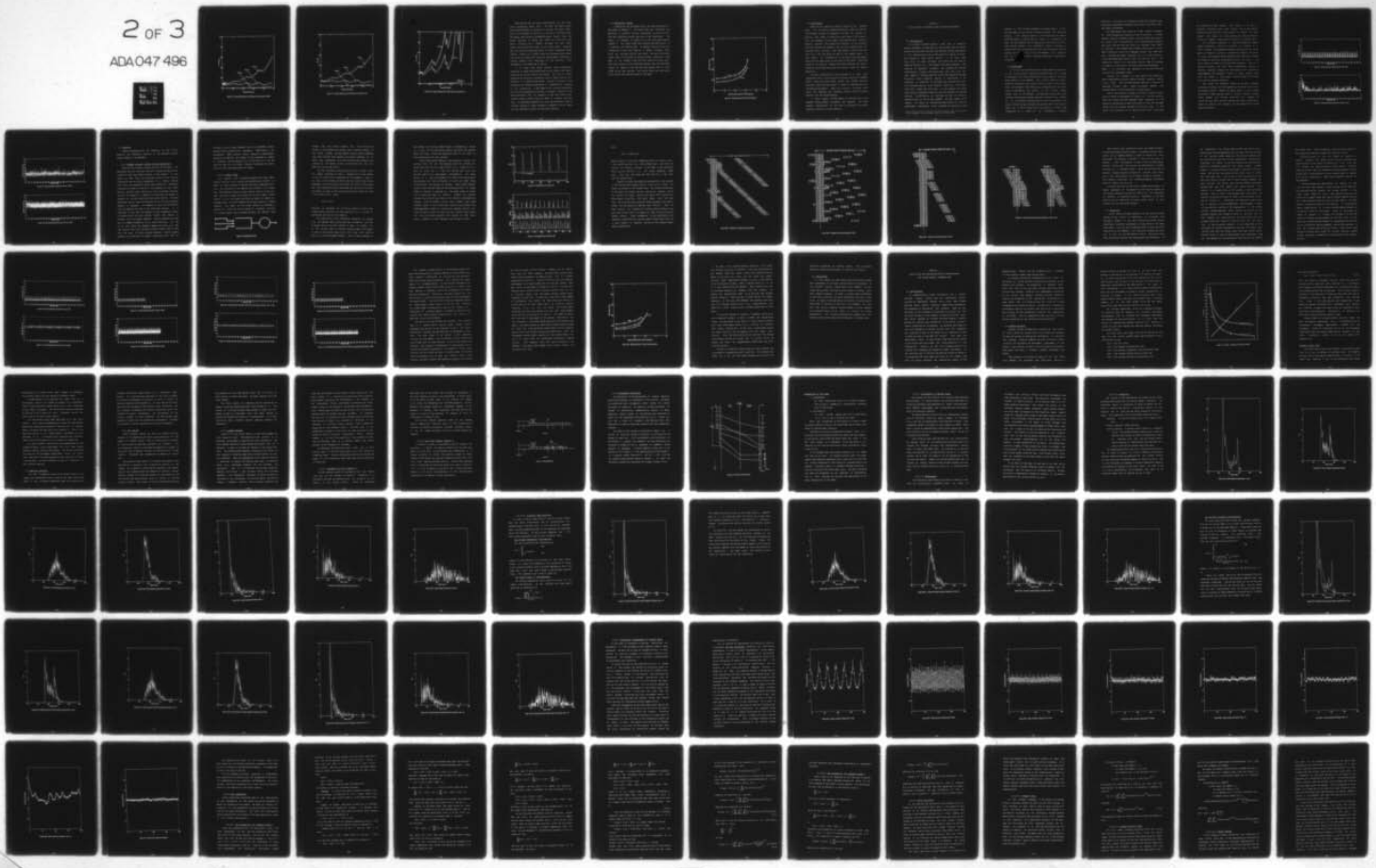


Figure 3.3. (c) Mean waiting time on TASK queue (2-connected 316 IMP).

AD-A047 496 CALIFORNIA UNIV LOS ANGELES DEPT OF COMPUTER SCIENCE F/G 17/2
ADVANCED TELEPROCESSING SYSTEMS.(U)
JUN 77 L KLEINROCK, G ESTRIN, R MUNTZ DAHC15-73-C-0368
UNCLASSIFIED NL

2 OF 3
ADA047 496



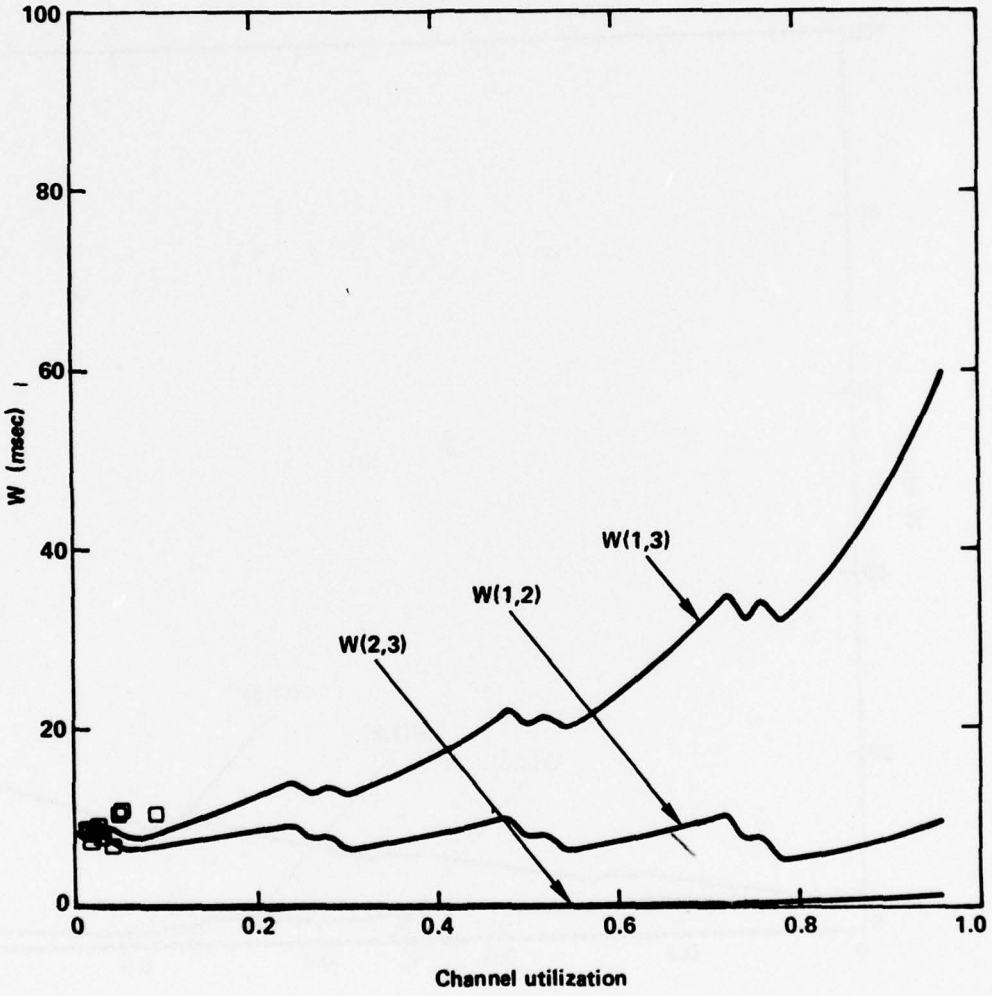


Figure 3.3. (d) Mean waiting time on TASK queue (3-connected 316 IMP).

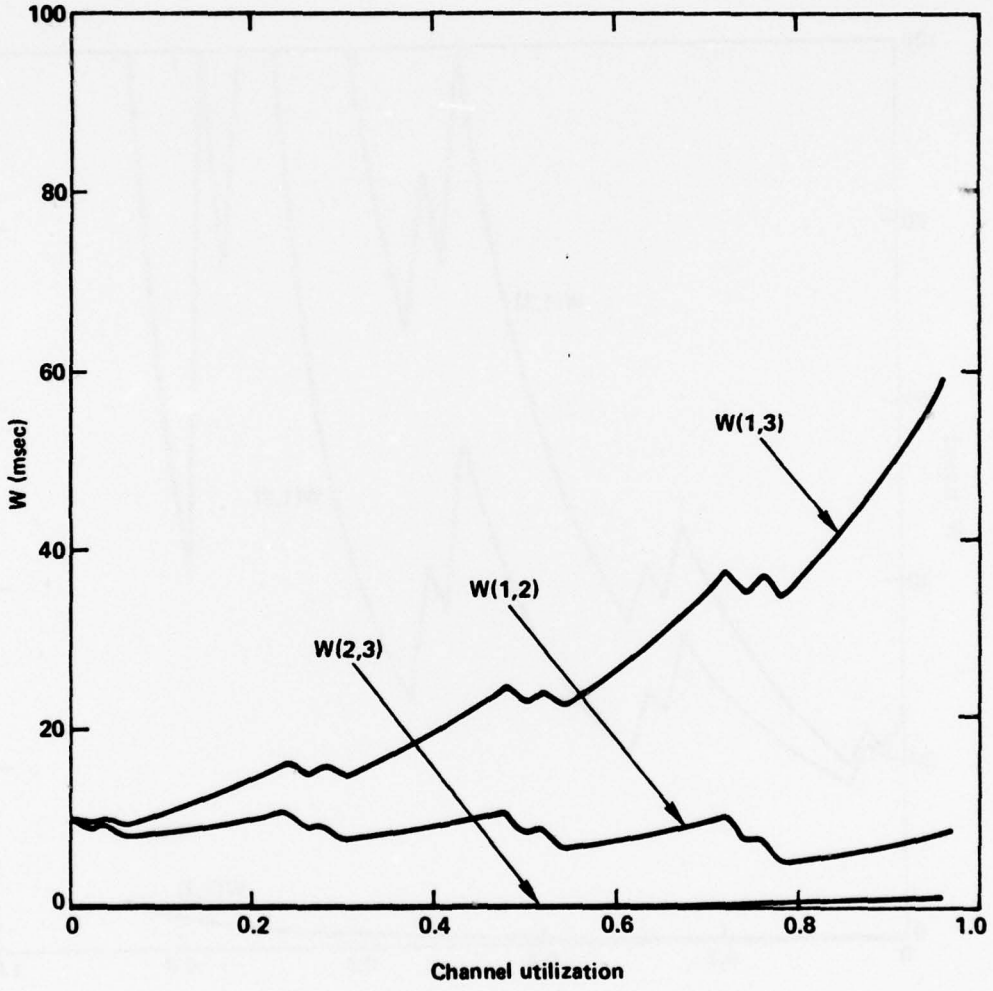


Figure 3.3. (e) Mean waiting time on TASK queue (2-connected TIP).

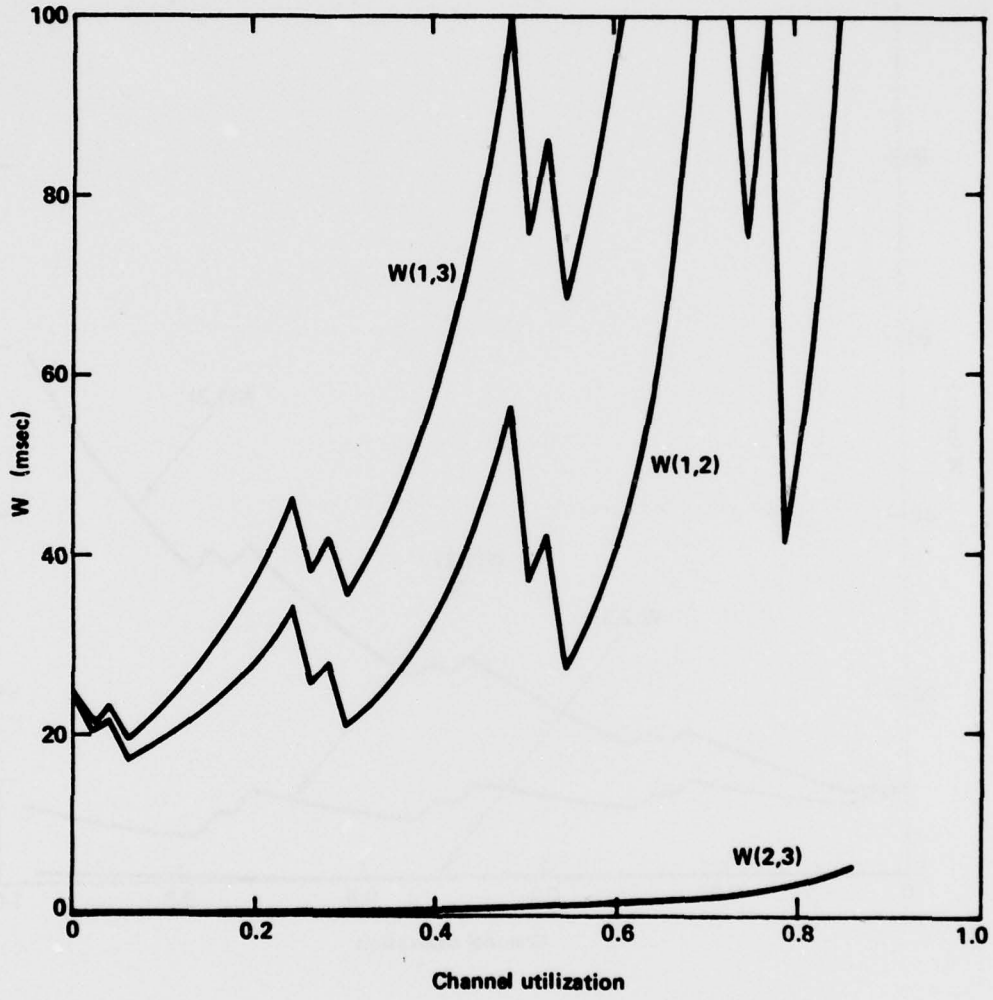


Figure 3.3. (f) Mean waiting time on TASK queue (3-connected TIP).

Substituting the two level system model for the constant processing delay term in the [Klei 74] model causes the predicted delay to increase to 165 msec for March 1977. This is an increase of 126% (i.e., from 73 to 165 msec) over the August 1973 value of predicted delay. Recall that the actual increase in delay was 145% (i.e., form 93 to 228 msec). A better (i.e., larger) estimate for x(u) would drive this prediction closer to the actual delay. Assuming that the ratio of predicted delay to actual (measured) delay remains fixed (i.e., 73/93), then the projected mean round-trip delay for March 1977 would be approximately 129 msec if routing updates were processed at low priority. This represents a 44% reduction in delay!

It is clear from these results that a substantial reduction in nodal processing delay would result from reordering the priorities within the system. But this is a local optimization which may eventually lead to worse overall performance in a network-wide sense. Because the speed of the processing of routing information is substantially reduced by the reordering, it may happen that routing information will not be propagated as quickly as needed. Looking at the three-connected TIP for example, we see that routing processing incurs infinite delay at about .6 channel utilization. To determine whether this local optimization leads to a global optimum or leads instead to disaster, let us examine the simulation results in the next section.

3.4 Simulation results

A simulation was performed under the same conditions as described in chapter 2. The local loop free algorithm was modified to perform routing processing at low priority. This new algorithm (LLFLR) was used in the simulation. The result is compared in Figure 3.4 to the LLFR algorithm of chapter 2. The figure shows the network-wide mean delay as a function of offered load. It appears that the local optimization in this case leads to a global optimum, since LLFLR out performs LLFR at each load. One would therefore expect that if low priority routing update processing were done in the ARPANET, then the mean round-trip delay would fall much closer to the predicted (rather than the measured) values in Table 3.1 for May 1974 and August 1977, as indicated in the last section. This would reduce the mean delay to be within the specification of 200 msec.

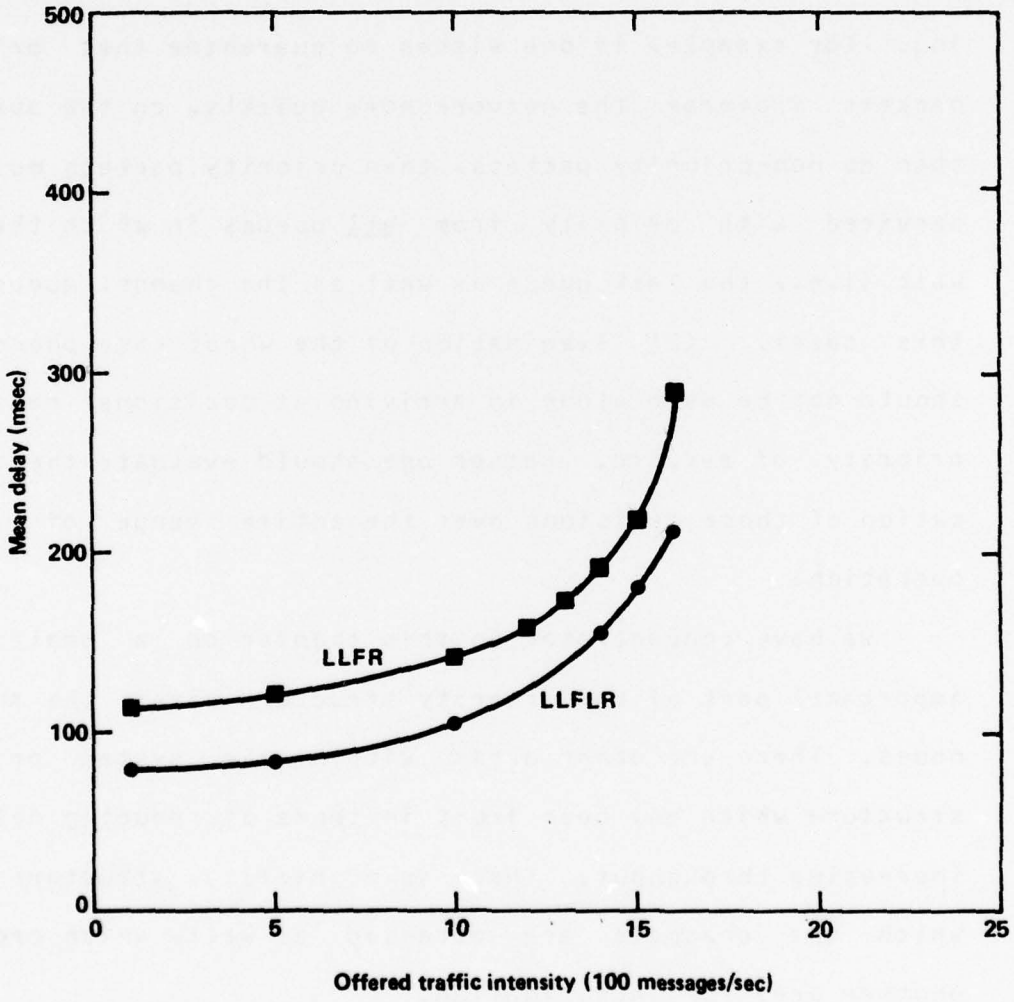


Figure 3.4. High and low priority routing (simulation).

3.5 Conclusions

There are two important general lessons to be learned here. (1) In order to provide different levels of service to different classes of customers, one must be careful to provide that order of service at each step of the processing. For example, if one wishes to guarantee that priority packets traverse the network more quickly, on the average, than do non-priority packets, then priority packets must be serviced with priority from all queues in which they may wait (i.e., the Task queue as well as the channel queue, in this case). (2) Examination of the worst case phenomenon should not be used alone in arriving at decisions regarding priority of service. Rather one should evaluate the implication of those decisions over the entire range of system operation.

We have concentrated in this chapter on a small (but important) part of the priority structure within the ARPANET nodes. There are other areas within the system priority structure which may bear fruit in terms of reducing delay or increasing throughput. There is a priority structure with which the channels are accessed as well, which provides another area for investigation.

In this chapter a high degree of overhead due to the ARPANET routing update procedure was exposed. The next chapter concentrates on the issue of overhead in periodic adaptive routing in large networks.

CHAPTER 4

ON THE EFFECT OF PERIODIC UPDATE ROUTING PROCEDURES

4.1 Introduction

In a packet switched network, some form of adaptive routing procedure is desirable so that packets may be routed around line and node failures and possibly around congestion in the network and to allow the network to adjust to drastics changes in input traffic matrix. It is clear that there must be some overhead associated with any form of adaptive routing (i.e., the channel time and processor time required to generate, transmit, and process the routing information). Clearly, one would hope that the cost for such adaptive routing does not exceed the benefits derived therefrom. Since adaptive routing is considered to be necessary in practice, its overhead has received only partial consideration by most authors [Cegr 75], [Fult 72], [McCo 75], [McQu 74], [Pick 76]. In this chapter, we study some unusual phenomena caused by the interference of routing updates. Specifically, we consider the cost (in terms of message delay) of the current ARPANET routing update procedure. We begin by presenting some results of a set of measurement experiments which prompted an analysis of the

---

* This chapter is a revised version of [Nayl 76].

effects of the routing procedure on message delay. A simplified model of the system is then discussed and analyzed exactly. This exact solution is a bit unwieldy for highly detailed models in which case we resort to simulation to show the performance and to demonstrate the effect of modified routing schemes. The simulation results indicate a rather high cost associated with the use of periodic routing updates in networks the size of the ARPANET. This suggests the use of "passive" routing scheme with catastrophe-triggered [illegible].

4.2 Measurement

We set out to determine by what means and how accurately one could predict round-trip network delay for a stream of messages with fixed interarrival times based on previous delay samples in the ARPANET. This traffic pattern is exhibited by fixed data rate sources such as speech [Forg 75a]. These measurement experiments were conducted with no intention of considering the effects of the periodic update scheme used in the ARPANET. We observed a much lower than expected correlation between successive delays (see Chapter 5). In experiments sending data as fast as possible, successive delays display higher correlation [Klei 75a] than do those for a fixed interarrival time source. A closer look (suggested by D. Cohen of the Information Sciences

Institute, University of Southern California) revealed some interesting phenomena regarding the effect of periodic routing update procedures.

The experiments took place on Friday evening December 12, 1975, between the hours of 9 and 11 p.m. PST. (A light network load and therefore nearly constant delay was expected during this time period.) Full single-packet messages were sent from the UCLA PDP 11/45 to a "discard fake HOST" (a portion of the ARPANET IMP software which mimics a "real HOST" acting as a sink for a message stream, see [BBN 69]) over (minimum hop path) distances of 1, 2, 5 and 10 hops at fixed interdeparture times of 124, 165, and 248 msec. The round-trip delay (i.e., the delay from the time the message is ready to be sent until the end-to-end acknowledgement -- RFNM -- is returned) was measured by the PDP 11/45 and recorded for subsequent study.

Figures 4.1 through 4.4 show some of the round-trip delay measurements plotted against message sequence number (i.e., time). Here we show network delay as a function of (message arrival) time. These particular samples are representative of the collection of experiments.

Notice the unusual increases in delay at regular intervals (of about 30 messages) in Figure 4.1. In each interval there is a group of three dominant peaks separated by two regularly spaced points where the delay is near the minimum. Notice also the regular decrease in the first peak in each group with time until it is replaced with a full sized peak

at intervals of five groups. This curve is in fact a periodic function (with some "noise" due to the "other" background data traffic) with a period of about 150 messages! (Our model in a later section and the correlation results in Chapter 5 show that the period is actually 160.)

This periodic behavior is less noticeable at longer network distances. There is a pattern of climbing to a local maximum and suddenly dropping and starting the climb again in Figures 4.2 and 4.3. Generally speaking increases in delay are gradual while decreases are immediate (though the opposite condition occasionally occurs as well). The curve shown in Figure 4.4 varies quite severely. There is a pattern however, and very close examination reveals a periodic function (again with some noise) with a period of approximately 130 messages. Plots of other samples show that the shape of the curves is more related to the data rate than to network distance.

One would not normally expect to see such a regular variation in delay with time assuming no control on the other network traffic. Rather one might expect to see a random function of time, possibly with a slowly varying average which changes with network load [Cohe 74]. (Indeed, the latter was our hope; we were looking for delay predictors.) Below we present an analysis of the regularity of these delay functions.

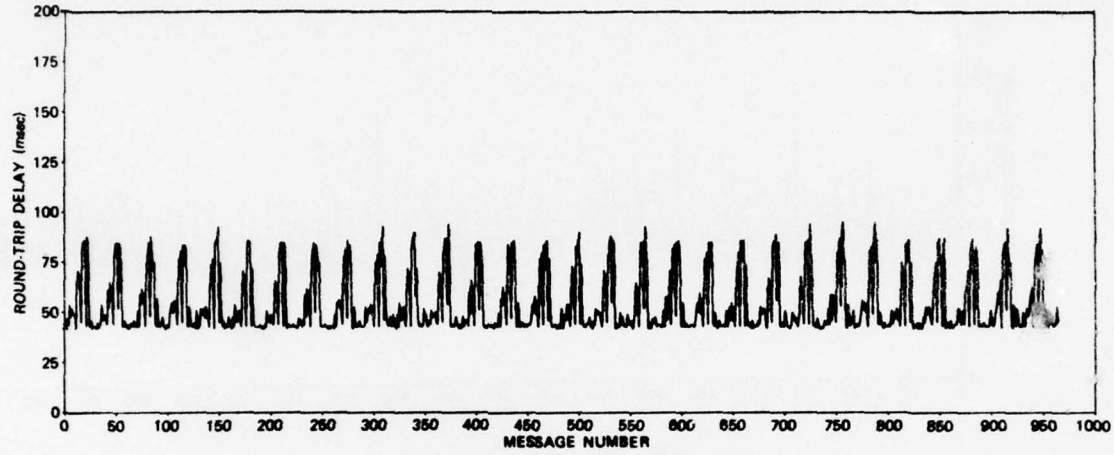


Figure 4.1. Round-trip delay measurement (1 hop, s=124).

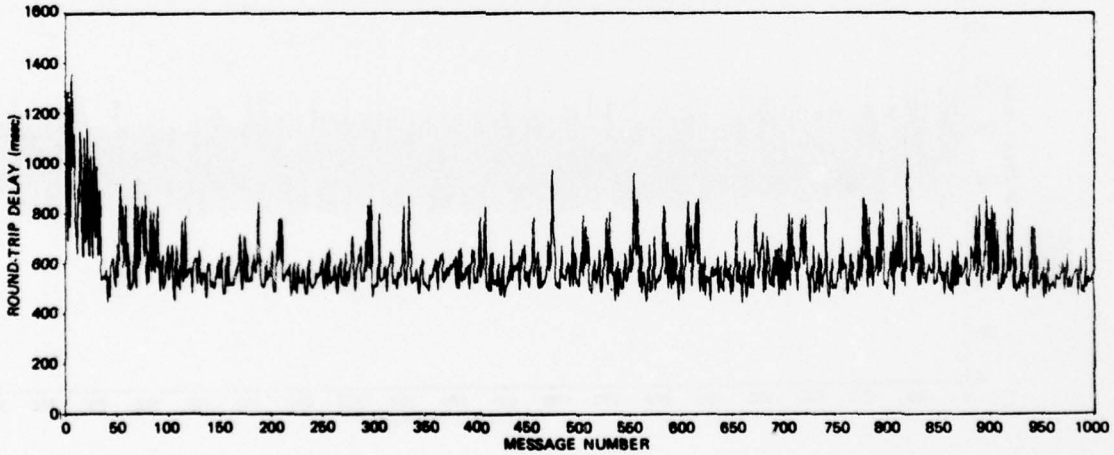


Figure 4.2. Round-trip delay measurement (10 hops, s=124).

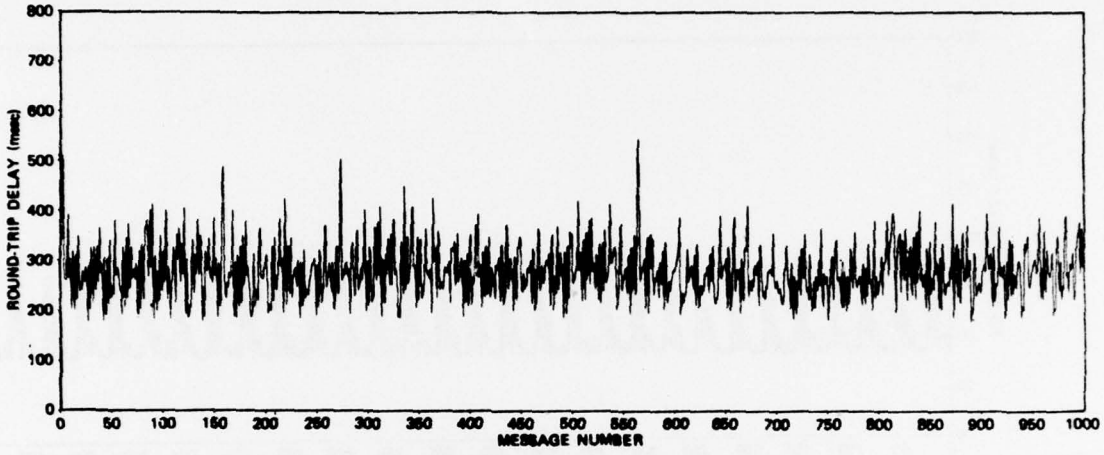


Figure 4.3. Round-trip delay measurement (5 hops, s=248).

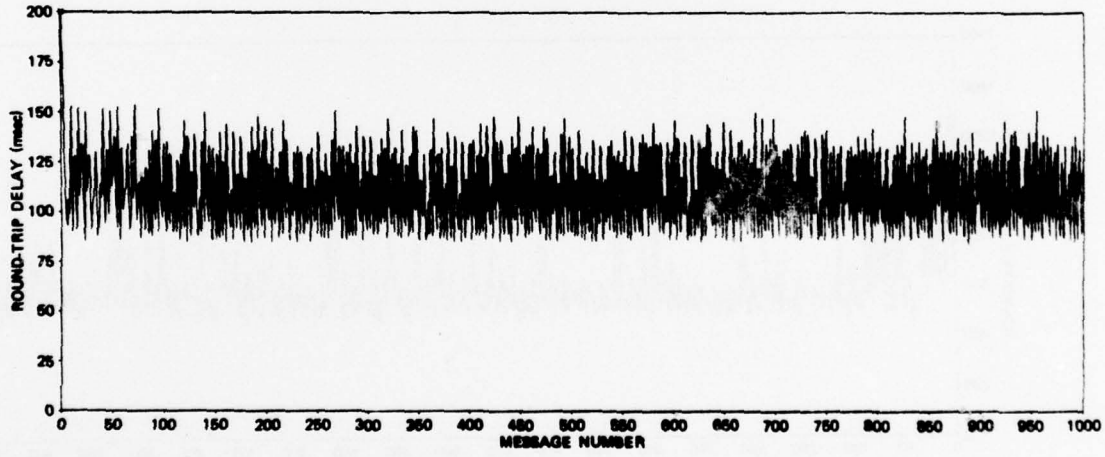


Figure 4.4. Round-trip delay measurement (2 hops, s=165).

4.3 Analysis

Before proceeding with the analysis, we must first describe the important features of the periodic routing update scheme in the ARPANET.

4.3.1 ARPANET periodic routing scheme description

There exists a basic routing interval of 640 msec. The beginning time for the basic period is chosen essentially at random for each half-duplex channel in the network. It was noted, by BBN, that as the network grew in size, routing information was not being propagated in a timely fashion with only one update per basic period [BBN 74]. Therefore provision was made to send up to five updates in one basic period. During a basic period the line utilization (including that for updates) is measured to determine the number of updates to be sent during the next basic period. For each additional 20% of line utilization, one of the five possible updates is dropped. For example, at 65% line utilization only two updates are sent in the next basic period. It is important to note that the updates are not necessarily evenly spaced within the basic period. Rather this period is divided into five equal segments. Routing updates are sent only at segment boundaries (i.e., every 128 msec). For the 20 to 40% range, for example, updates are sent at 0, 128, 384, and 512 msec into the basic period (rather than 0, 160, 320, 480 msec for evenly spaced updates). Routing update packets were 1160 bits in length, requiring 23.2 msec to

transmit on the 50 kbps channels used in the ARPANET (recent changes have increased this somewhat). Additionally, approximately 12000 machine cycles (based on a measurement reported in [BBN 75], see Chapter 3) are required to digest an incoming routing update (i.e., 11.52 msec for a 516 IMP and 19.2 msec for a 316 IMP). Chapter 3 examines the importance of this overhead within the nodes.

4.3.2 A simple model

Our analysis uses a queueing system with some "background" traffic (i.e., routing and ambient data traffic) to which we add a stream of deterministically generated traffic. The inclusion of the ambient data traffic is to model the interference caused by other packet sources in the system. We wish to study the system time of this added stream traffic (i.e., the round-trip delay as shown in Figures 4.1 through 4.4). We first examine the waiting time on a single channel. The model for the single channel is pictured in Figure 4.5. There are three classes of customers arriving to a single queue; routing update packets (R), ambient data

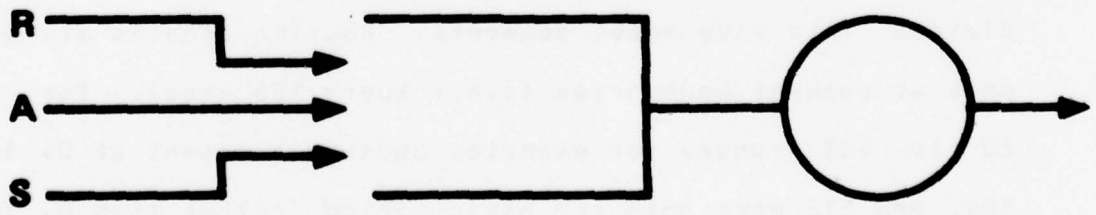


Figure 4.5. Single channel model.

packets (A), and stream packets (S). All arrivals are served in first-come-first-served (FCFS) fashion (though in the actual system, routing update packets take precedence over both priority data packets and control packets, all of which take precedence over non-priority data packets; see Chapter 3). The effect of this is negligible for our purpose here as we show later.

Let w(t) represent the waiting time for a stream traffic packet arriving at time t. Suppose U(t) is the amount of work of type R and A remaining in the system at time t. As long as the stream arrivals have no priority over other customers, an arrival must wait until all work in the system has been completed before receiving any service of its own. The amount of work in the system found by an arrival at time t is at least U(t). Hence

$$w(t) \geq U(t)$$

Equality is achieved for an arrival when it is the first stream traffic arrival in a busy period (i.e., a period of continuous activity by the server).

For simplicity of the following analysis, we assume that the ambient data traffic has zero intensity. U(t) for a system void of any data traffic is shown in Figure 4.6. At the arrival time of a routing update packet, the amount of work in the system jumps up by 23.2 msec (the service time of a routing update packet). With no other packets in

the system, the routing update packet is immediately served at a rate of one second per second, and exits the channel after 23.2 msec. After the departure there is no work in the system until the next arrival.

A more interesting measure of performance is w(sn), the waiting time for the nth message, where the constant interarrival time of the stream traffic is s. Figure 4.7 is a plot of U(sn) for s = 248, 165, and 124 msec (i.e., the periods used in the measurement experiments). For these values of s (and the message size used in the measurement experiments) the line utilization is in the 20 to 40% range so that every fifth update is dropped. These single channel curves nicely display some of the characteristics of delay shown in Figures 4.1 through 4.4. The major shape of Figure 4.1 is nearly the same as U(124n). U(165n) and U(248n) display the relative variation of the previous corresponding figures. That is U(165n) varies more rapidly than does U(248n) which in turn varies more rapidly than U(124n); and correspondingly Figure 4.4 varies more rapidly than do Figures 4.2 and 4.3 which in turn vary more rapidly than Figure 4.1. One can clearly identify the period of each U(sn) curve. Indeed, let P(s) represent the period of U(sn).

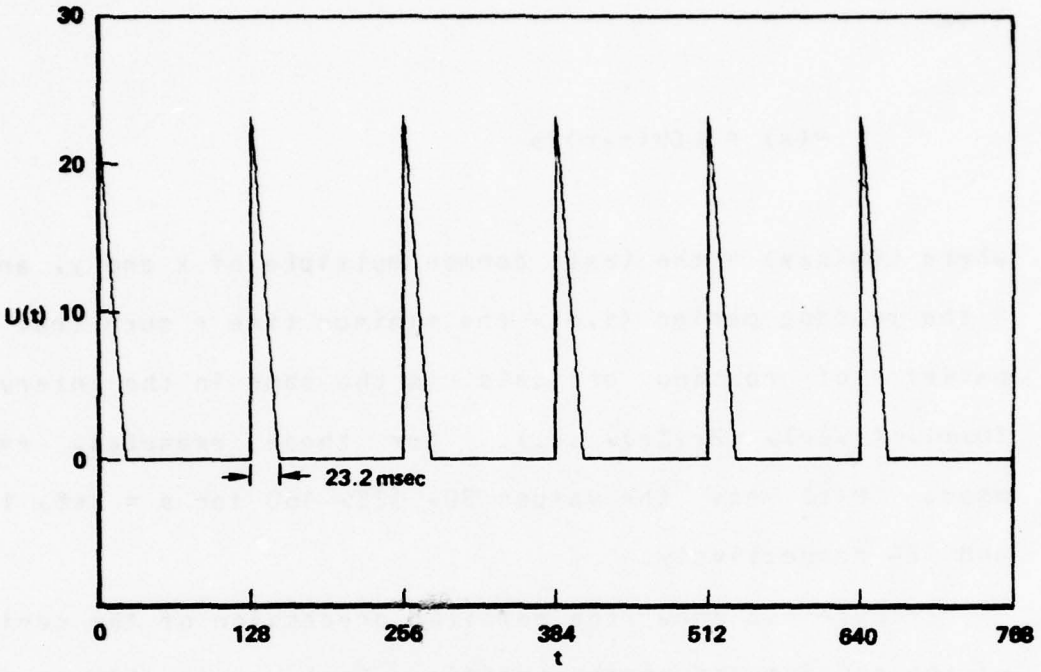


Figure 4.6. Unfinished work (routing only).

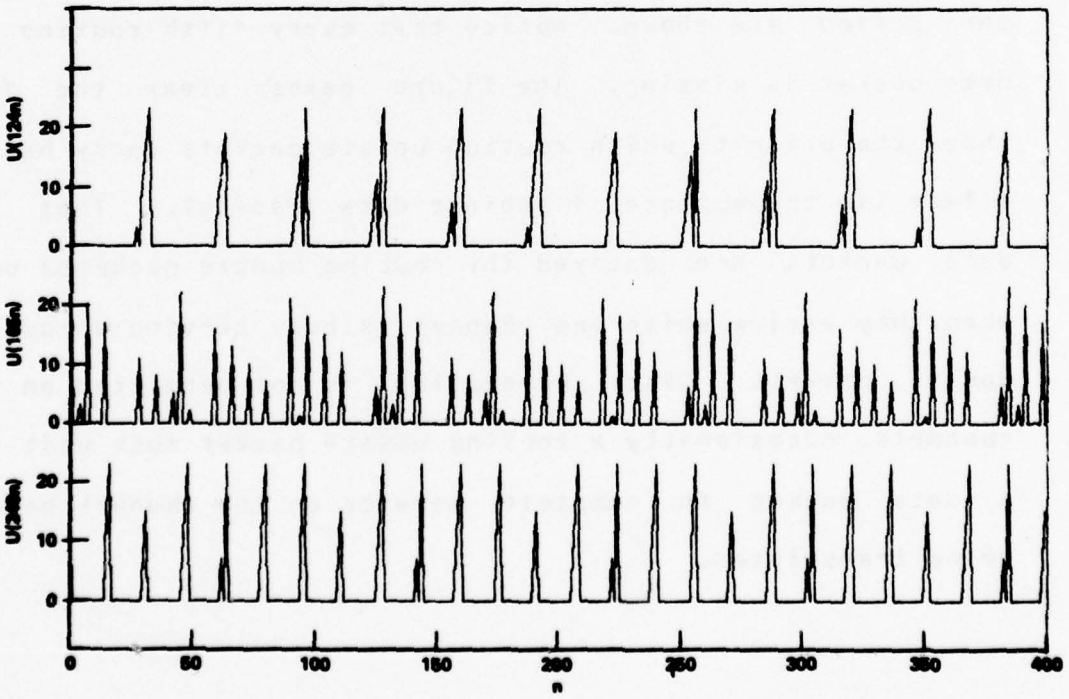


Figure 4.7. Unfinished work at stream arrivals.

Then

$$P(s) = LCM(s,r)/s$$

where LCM(x,y) = the least common multiple of x and y, and r = the routing period (i.e., the minimum time r such that the pattern of routing arrivals is the same in the intervals (0,r), (r,2r), (2r,3r), ...). For these examples, r=640 msec. P(s) has the values 80, 128, 160 for s = 248, 165, and 124 respectively.

Figure 4.8 shows the detailed precession of the periods of the routing and stream traffic. Each row in Figure 4.8 shows one period of the stream data. The relative positions of the data (S) packets and the routing (R) packets within the period are shown. Notice that every fifth routing update packet is missing. The figure makes clear the fact that the priority which routing update packets carry has no effect (in the absence of ambient data traffic). That is, data packets are delayed (by routing update packets) only when they arrive while the channel is busy serving a routing update packet. Since preemption is not permitted on the channels, occasionally a routing update packet must wait for a data packet to complete service on the channel before being transmitted.

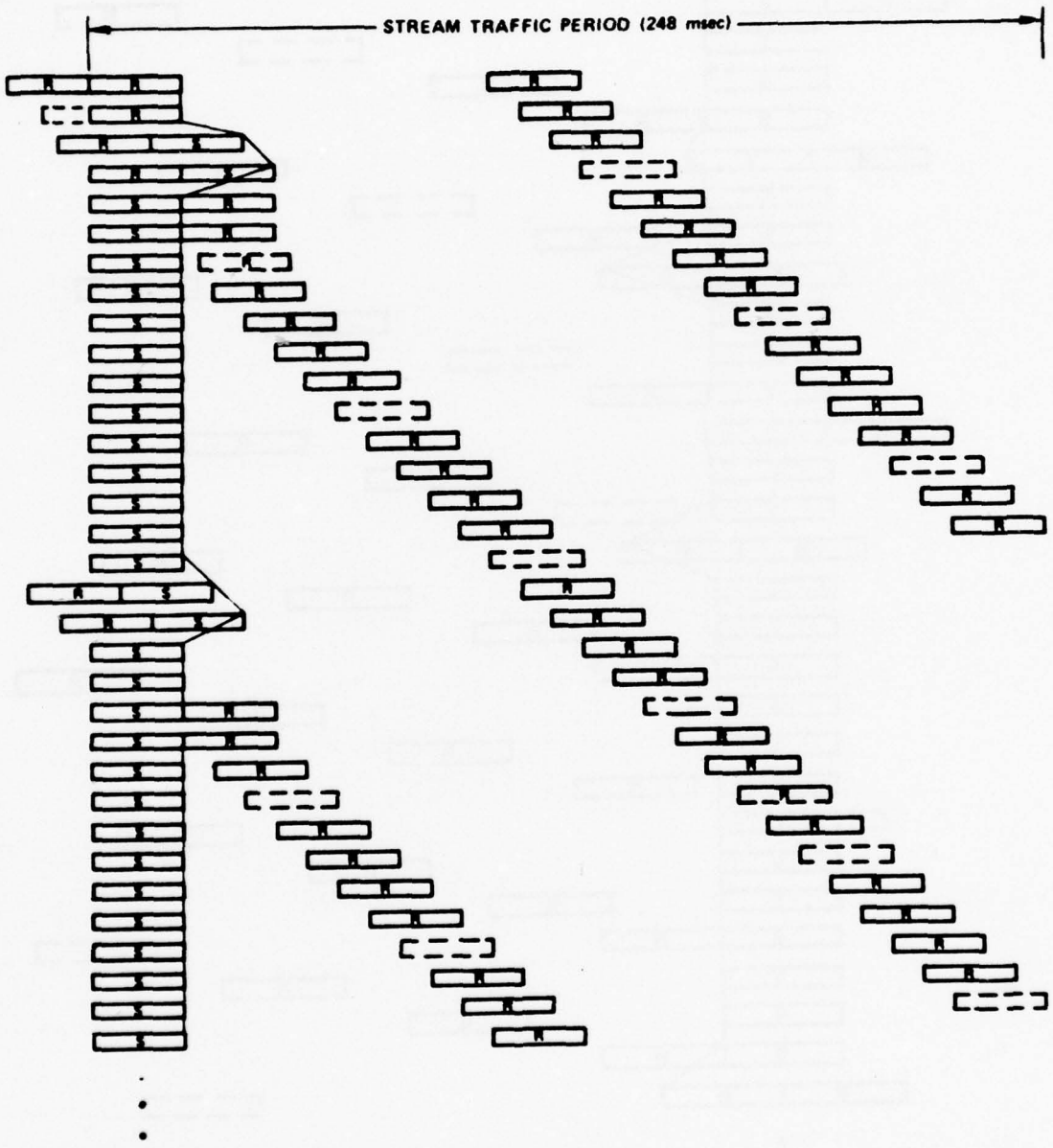


Figure 4.8(a). Precession of interference pattern (s=248).

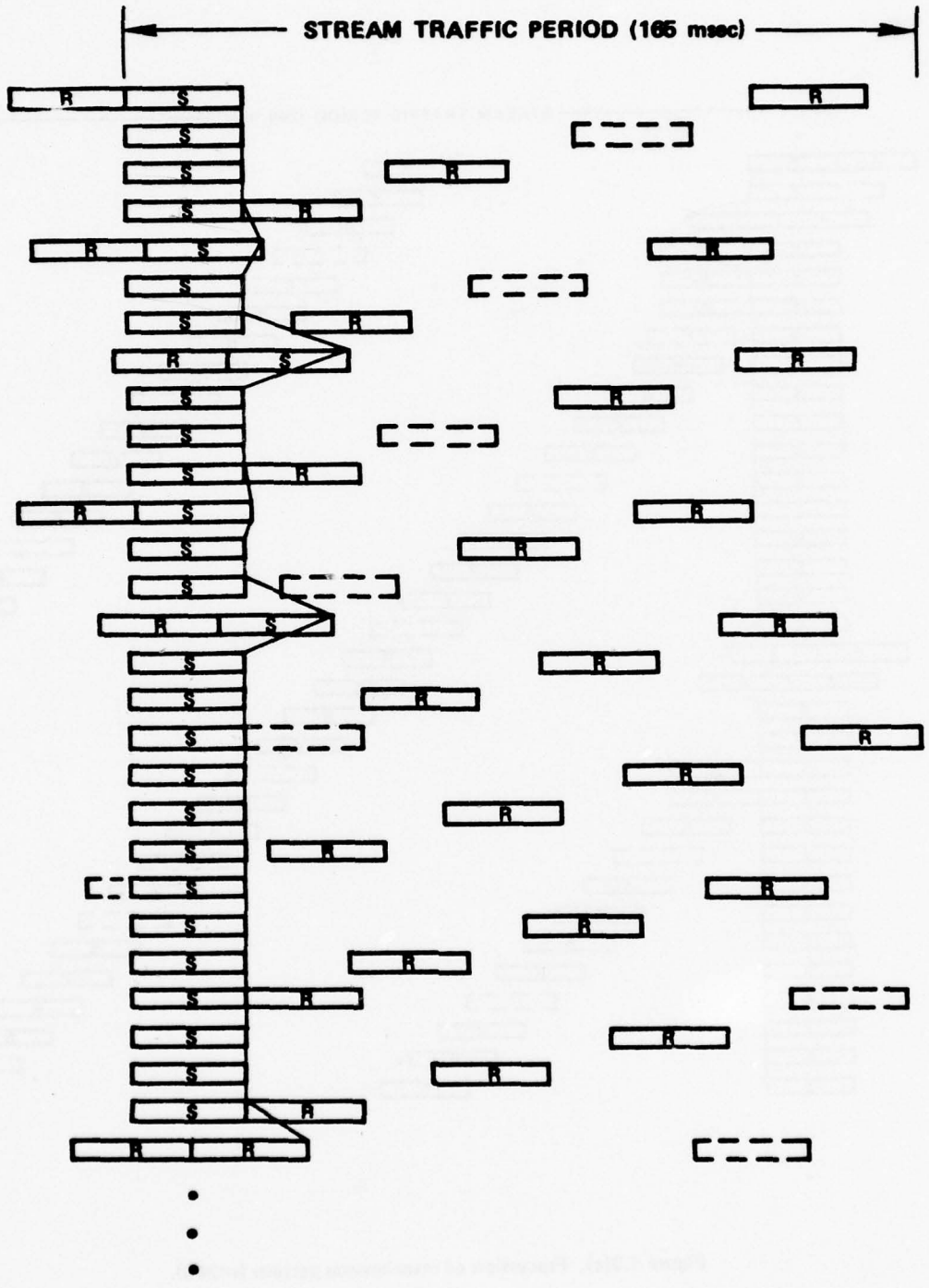


Figure 4.8(b). Precession of interference pattern (s=165).

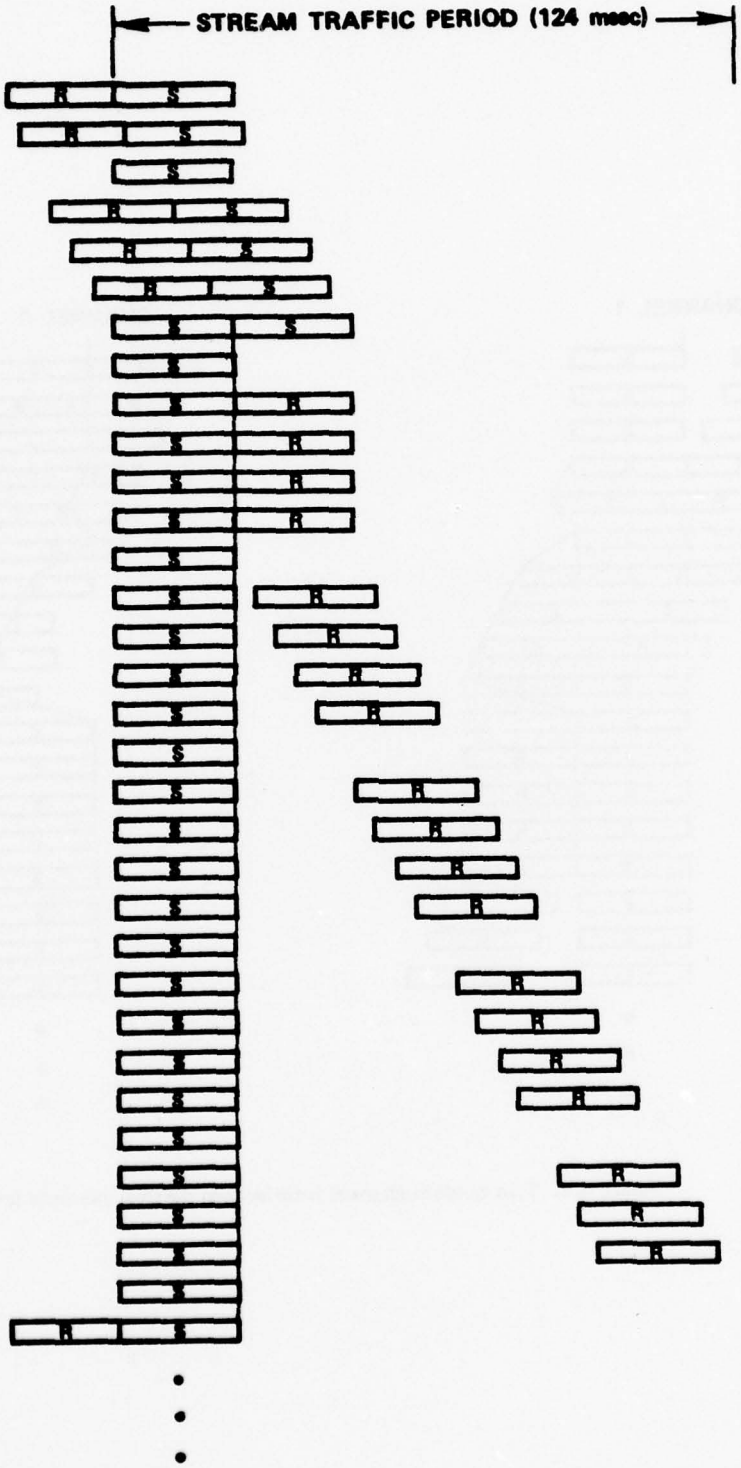


Figure 4.8(c). Precession of interference pattern (s=124).

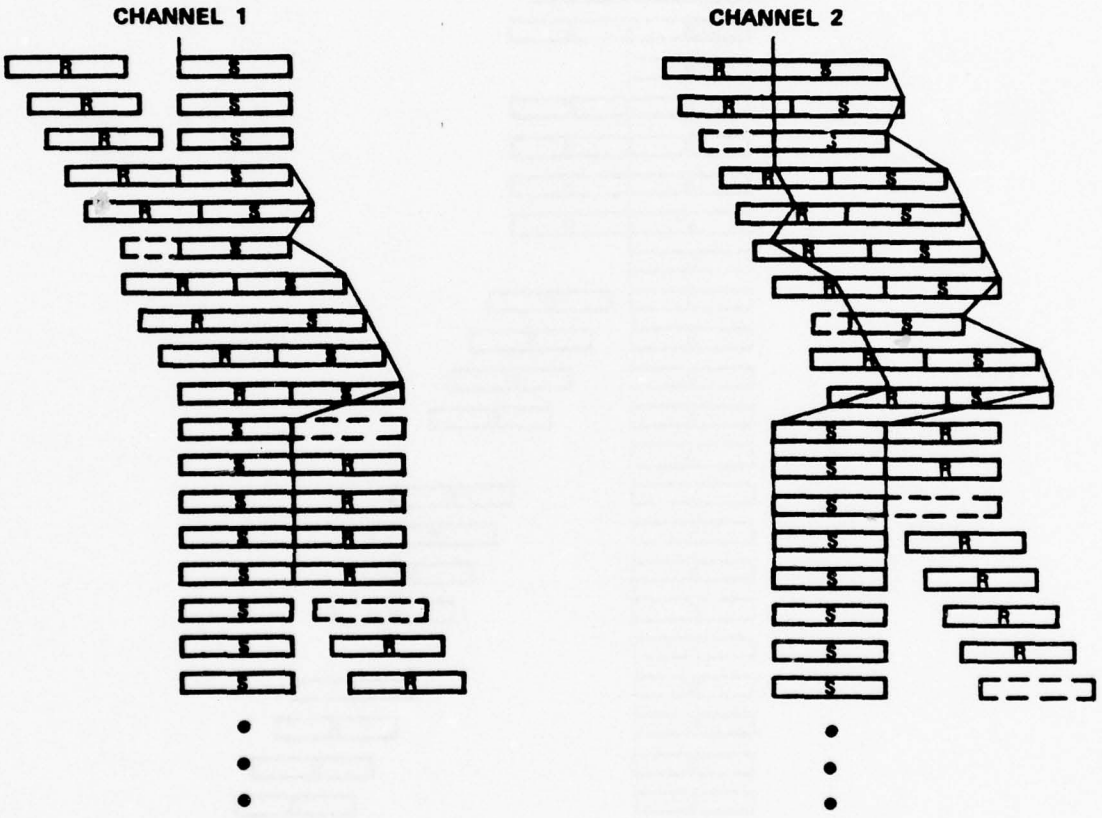


Figure 4.9. Two tandem channel interference pattern example (s=124).

When packets pass through more than one tandem channel, a more complicated pattern arises. An example pattern for a system of two tandem channels is shown in Figure 4.9. Once the packets are delayed at channel 1, their arrival rate at channel 2 then corresponds to that of the routing update packets. During this time we have two deterministic streams arriving at a constant offset. Hence the waiting time at channel 2 remains constant (except for routing drop outs) until the data packets are no longer delayed at channel 1. This example illustrates the climbing-dropping phenomenon shown in Figures 4.2 and 4.3.

An exact solution of the n-node tandem server model is highly dependent on the phasing of the routing updates on the various channels, as seen in Figure 4.9. This dependency produces an unwieldy solution. Therefore, in the next section, we use simulation to study a model which is more closely tied to the actual system.

4.4 Simulation

Having shown the exact behavior for the single channel system, we now resort to simulation to illustrate some phenomena present in the more complicated tandem system. Essentially identical experiments, as described in the above measurement section, were performed using a rather detailed simulation of the ARPANET. The simulation program was written in PL/I for the IBM 360/91 at UCLA. There were three main differences between the measurement and simulation. In

the simulation, the ambient data traffic was set to zero, node-to-node acknowledgements were not used, and the phasing of the routing update packets on the various channels was not synchronized with that of the measurement experiments. Therefore, we expect to see the same period displayed but possibly a different shape for the delay curves. Indeed Figures 4.1 and 4.10 show a curve with a period of 320 messages. Notice that the period differs from that of U(sn) for this data rate. This is due to the fact that while the data rate causes a stable rate of four routing updates per basic period, the return path carrying RFNM's alternates between four and five updates per basic period. This results in a stable rate of nine updates in each two basic periods. That is, 1280 is the minimum time r for which the pattern of routing arrivals is fixed in the intervals (0,r), (r,2r), (2r,3r),... . Therefore, the round-trip delay curve has a period of LCM(124,1280)/124 = 320 messages. Figure 4.1 may exhibit a period of 160 messages due to the node-to-node acknowledgements which elevate the traffic on the backward channel just enough to force a constant rate of four updates per basic period. One also notices that the minimum values for Figures 4.1 and 4.10 are not the same. This is due to several factors. In the simulation we have estimated the channel propagation time and the nodal processing time; both are likely lower than their actual value. Another factor is that we have assumed zero acceptance time for the message at the destination node and for the RFNM at

the source node. These assumptions drive the delay down for the simulation and hence lowers the vertical offset.

Figures 4.2 and 4.3 are too random to identify a period. However, one should notice the slow climbing and rapid falling in the curves in Figures 4.2, 4.3, 4.11 and 4.12 (though the measured data is quite noisy). The most rapid variation, both for simulation and measurement, is the packet rate of approximately six per second shown in Figures 4.13 and 4.4 respectively. Both curves possess a period of 128 messages.

Nestled between the large delays in Figures 4.1 and 4.10 are some small mounds. These are due to the interference between routing updates and stream traffic in the "TASK" queue [McQu 72] (i.e., data packets waiting to be placed on an output queue and routing update packets waiting and being digested into the local routing table). The TASK queue is currently served in FCFS fashion. A conceptually simple modification is to serve routing update packets at low priority from the TASK queue. (This is discussed at length in Chapter 3.) Figures 4.14 through 4.17 show the effect of sending routing updates at the same rate as before, but processing data packets by preempting the processing of interfering routing updates. Notice the decrease in both the average and variance of delay. These curves show the best possible delay under the current periodic update scheme, assuming preemption is not allowed on the channels as well.

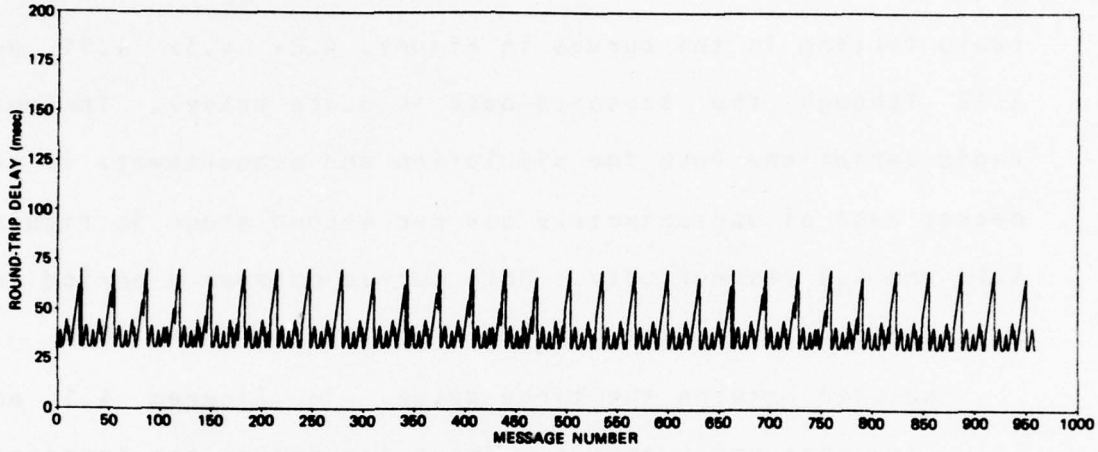


Figure 4.10. Round-trip delay simulation (1 hop, s=124).

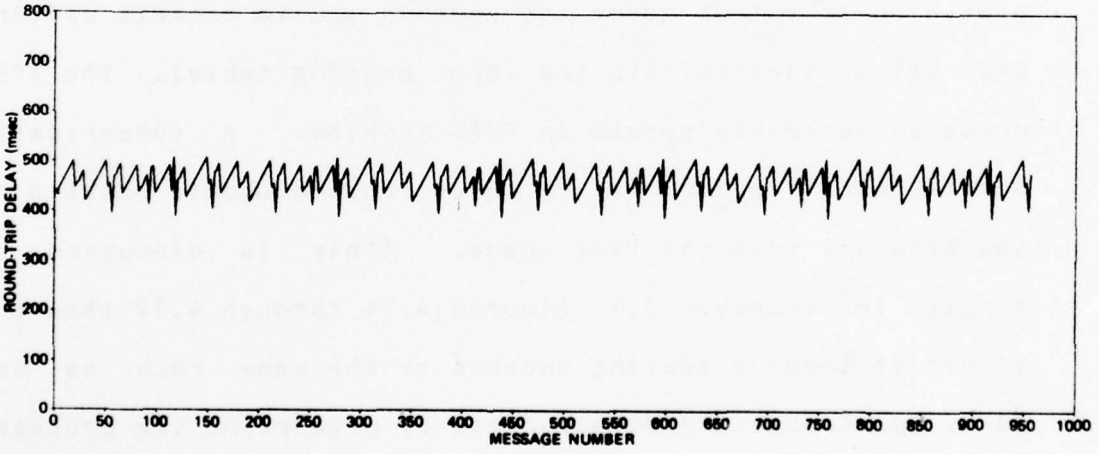


Figure 4.11. Round-trip delay simulation (10 hops, s=248).

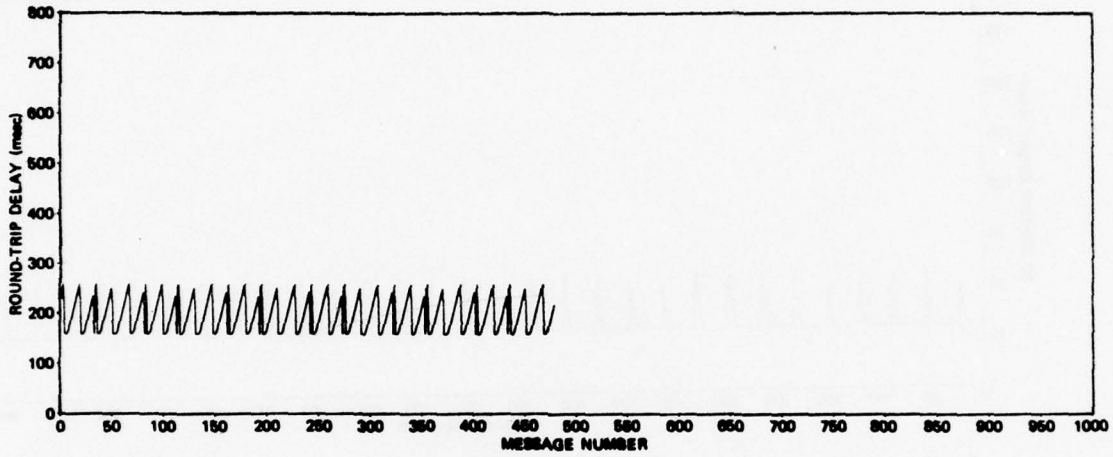


Figure 4.12. Round-trip delay simulation (5 hops, s=248).

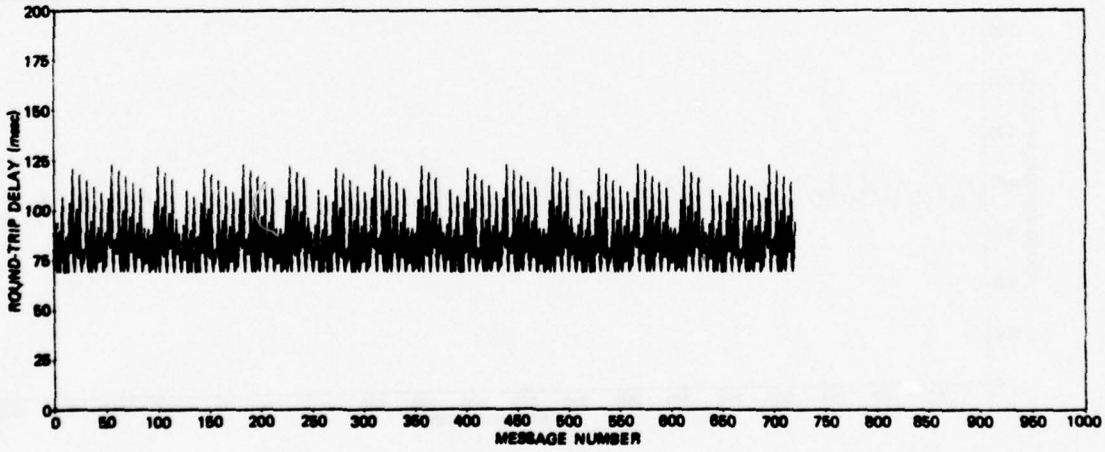


Figure 4.13. Round-trip delay simulation (2 hops, s=165).

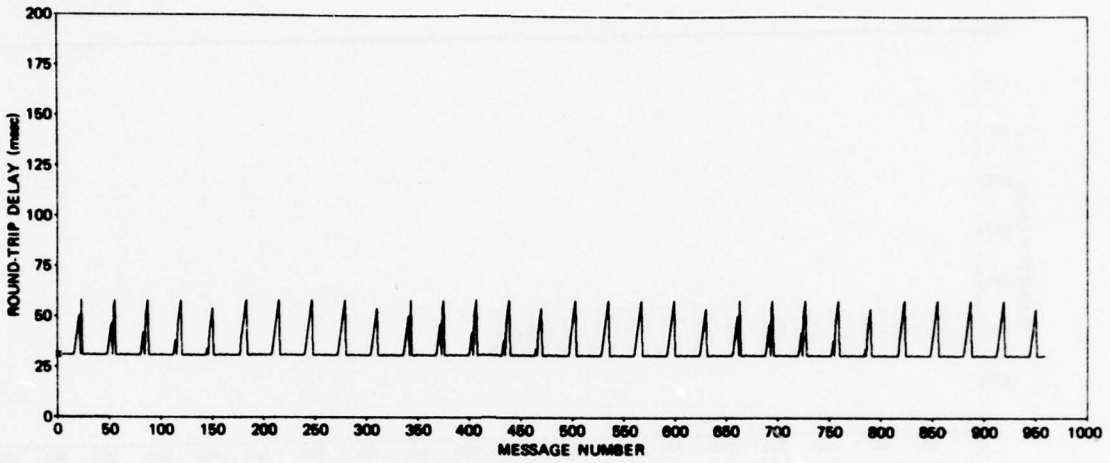


Figure 4.14. Round-trip delay simulation with low priority routing processing (1 hop, s=124).

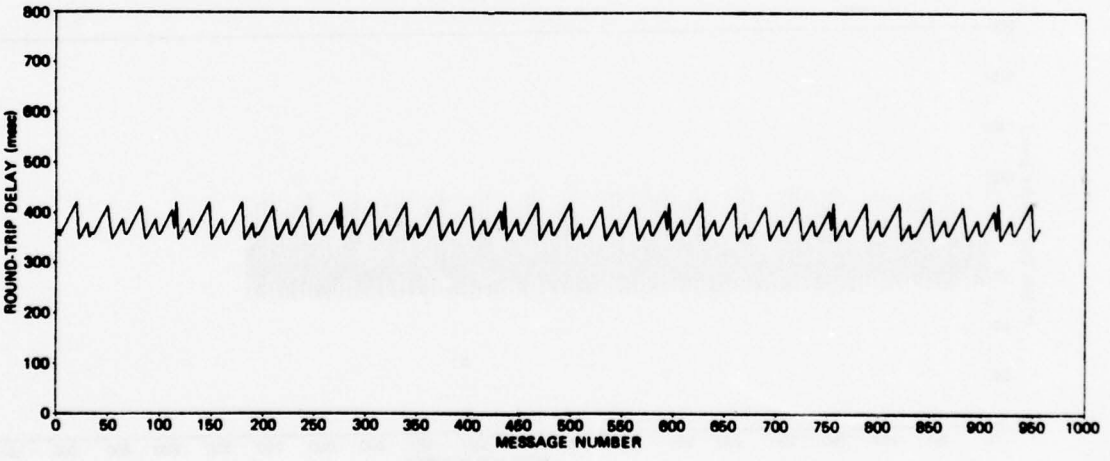


Figure 4.15. Round-trip delay simulation with low priority routing processing (10 hops, s=248).

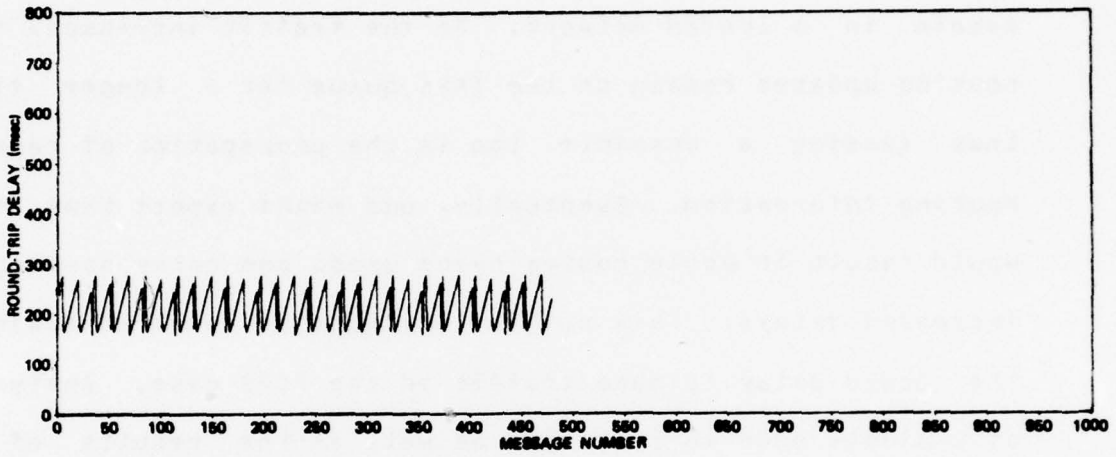


Figure 4.16. Round-trip delay simulation with low priority routing processing (5 hops, s=248).

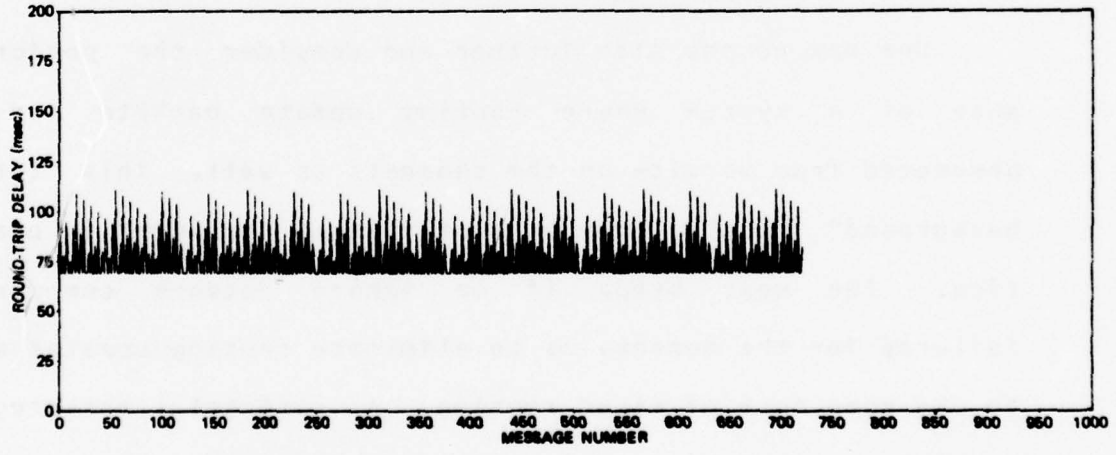


Figure 4.17. Round-trip delay simulation with low priority routing processing (2 hops, s=165).

For sending stream traffic in an otherwise empty network, the processing of routing updates at low priority provides superior performance to the current FCFS approach. There may be some question whether this superiority would remain in a loaded network. As the traffic increases, the routing updates remain on the TASK queue for a longer time thus causing a possible lag in the propagation of recent routing information. Eventually, one might expect that lag would result in worse routes being used, and correspondingly increased delays. This possibility must be compared against the added delay to data traffic in the FCFS case. Analysis of a single node in isolation as well as the results of a simulation of a network appear in Chapter 3, with the conclusion that routing update processing at low priority is good both locally and globally.

One may go one step further and consider the performance of a system where routing update packets may be preempted from service on the channels as well. This "total background" routing may be difficult to implement in practice. The next step, if we ignore network component failures for the moment, is to eliminate routing updates and to use some form of fixed routing. As with total background routing, this causes no interference of data packets due to routing information processing. One wonders if fixed routing will be less effective since it cannot adapt to changing traffic patterns (i.e., at very low traffic levels fixed routing performs better than adaptive schemes, but perhaps

may fail at higher traffic levels). However, we do recall from [Klei 64] that properly designed fixed routing procedures may be superior to adaptive ones. This is further supported by [Pric 72]. Figure 4.18 also shows the relative performance of a fixed scheme (FR) and the two others (HDR and LLFLR as described in Chapters 2 and 3). To give perspective here, we note that a relative traffic load of 1.0 is slightly higher than the weekly average traffic level reported in [Klei 74]. We note also that with a high degree of confidence, the values for mean delay are correct to within $\pm$7% for the first three levels of relative traffic intensity (i.e., 1.0, 5.0, and 10.0). The values above intensity 10.0 are less precise, but the only questionable points are for LLFLR and FR at load 14. The range of values for LLFLR and FR overlapped in the several simulations for this intensity. The fixed routes used were the shortest hop paths. One could choose an even better fixed routing scheme by using the flow deviation method [Gerl 73] for example. Notice that fixed routing always performed better than the foreground routing (HDR), and only at very high traffic levels is it worse than the background processing routing (LLFLR). This suggests that the cost of routing in the ARPANET is extremely high indeed, since traffic levels are currently very low.

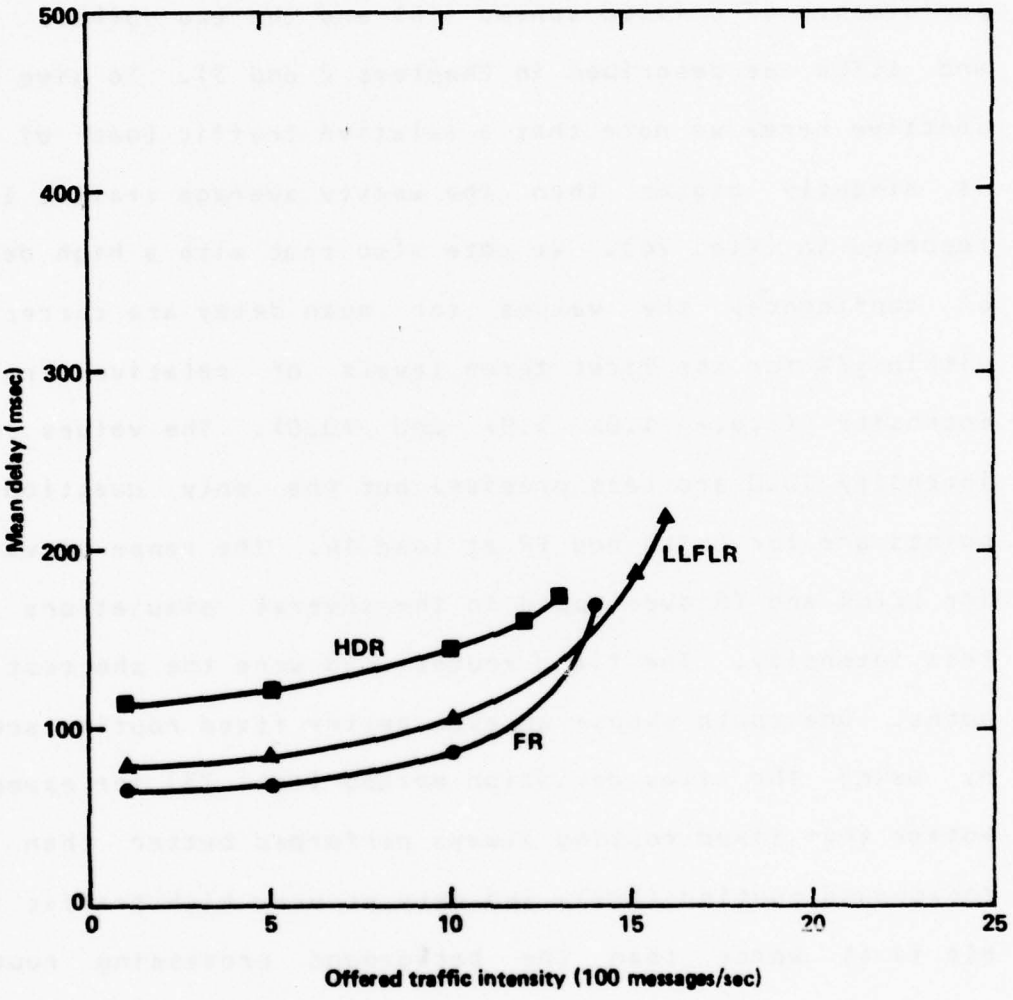


Figure 4.18. Adaptive routing vs. fixed routing (simulation).

We have so far ignored network failures. It is clear that failures do occur in practice. Long term monitoring of the ARPANET [McKe 74] shows a mean time between failures (MTBF) of 431 hours for lines and 221 hours for nodes. Failures cause topological changes to occur in the network in the following two ways. When a channel fails, it is as if it were removed from the network. When a node fails, all its attached channels are removed from the network. We define the network-wide MTBF to be the mean time between channel removals. Then with the 57 nodes and 65 full duplex channels, in the June 1975 ARPANET, and assuming that each node is of average connectivity (i.e., approximately 2.24 [BBN 75]), these figures yield a network-wide MTBF of 3.76 hours.

A far better method of routing, it appears, would be to use a "passive" scheme. In such a scheme one establishes routes and continues to use them in a fixed routing fashion until some catastrophe occurs (i.e., a failure or possibly even severe congestion). At the time a catastrophe occurs one could "turn on" routing updates until the tables "stabilize". From the above data one can see that the average time between turning on routing due to failure would be almost two hours (or approximately 10000 times the basic routing period)!

Recently a technique called "explicit path routing" was introduced by Jueneman and Kerr [Juen 76]. This scheme provides for a set of fixed paths between any node pair and

requires no updating of routing tables. This procedure deserves further investigation in light of our results.

4.5 Conclusions

In this chapter we have shown some interesting message delay phenomena for periodic stream traffic attributable to the periodic routing scheme used in the ARPANET. We conclude that periodic routing is quite costly in medium sized (and bigger) networks. In order to assure good performance in the face of failure (or heavy congestion) one pays a high price in terms of message delay for periodic routing update procedures in networks of the size of the ARPANET. We suggest the use of a passive routing scheme, in which updates are scheduled (only) as the result of failure (or heavy congestion). The results presented here suggest that this method could provide superior performance at reduced cost.

CHAPTER 5

(SOURCE AND) DESTINATION BUFFERING CONSIDERATIONS FOR STREAM TRAFFIC COMMUNICATION

5.1 Introduction

In communicating stream information via a packet switched network rather than the traditional circuit switched (or dedicated) network there arise some unique problems which require solution. Among these problems is the packaging of information into packets. Some ad hoc solutions to this problem are discussed in section 5.2. But the chief concern of this chapter is dealing with the variable delay imposed by a packet switched network. Several methods of limiting this variability in the output by destination buffering are considered. By delaying the output of the first message of a stream, one may limit the frequency and duration of gaps in the output (i.e., intervals of time in which no data is available to output). Such gaps are undesirable since, to some extent, they destroy the rhythm of the output and thus hinder the intelligibility of the information. Clearly, as the first message delay is increased, the frequency and duration of gaps decreases. In the limiting case of infinite destination buffering delay it is guaranteed that zero gaps will occur in the output but this of course destroys the interactive nature of the

communication. Therein lies the tradeoff which is examined in this chapter, namely gaps versus delay.

By assuming statistical independence of the delay experienced by stream packets traversing the network, we may analyze this tradeoff. The assumption is somewhat justified, but fails to be true under certain conditions, which we partially examine. Based on this independence assumption we can derive some general analytic results. Numerical results are presented for the exponential distribution in some detail and to a lesser extent for a class of r-stage Erlangian distribution of delay. Following the theoretical performance results, we present some results of a simulation which allows for the relaxation of some of the assumptions of the model. This is accomplished with the use of a trace driven simulation of the various buffering methods.

5.2 Sending strategy

Whenever stream information is produced by the source in units which are smaller than a full packet (or message) we have the option of sending the information in partially full packets. Doing so reduces the time required to create a packet, but increases the throughput requirement of the source since packets contain a non-zero amount of overhead. Once again, one finds a tradeoff between throughput and delay!

This tradeoff is pictured in Figure 5.1 for the fixed rate ARPANET LPC algorithm (see [Cohe 76]). There is a

linear increase in packet fill time as we move from the minimum of one parcel to the maximum of 14 parcels per packet. (A parcel consisting of 67 bits contains one set of LPC coefficients and as such is the smallest unit which may be separately interpreted at the destination.) The slope is 19.2 msec/parcel. It is clear that an inverse relationship holds for the throughput in terms of either packets/sec or bits/sec as shown. If the network protocol uses a full packet time to send a partially full packet (e.g., a slotted channel as in the ARPA SATNET [Klei 73] and PRNET [Kahn 75]), then the bits/sec curve is of no use.

Two extreme approaches exist for a sending strategy: (1) minimize delay (at whatever the resulting throughput requirement); and (2) minimize the throughput requirement (at whatever the resulting delay). It is clear that the choice of sending full packets satisfies (2). However, it is not so clear that sending the smallest packets minimizes the overall delay.

Let f(p), x(p), c(p), w(p), and D(p) be defined as follows for the stream packets given that packets of size p parcels are in use.

f(p) = the fill time,

x(p) = the network transmission time,

c(p) = the network propagation and processing time,

w(p) = the average network waiting time, and

D(p) = the average destination buffering time.

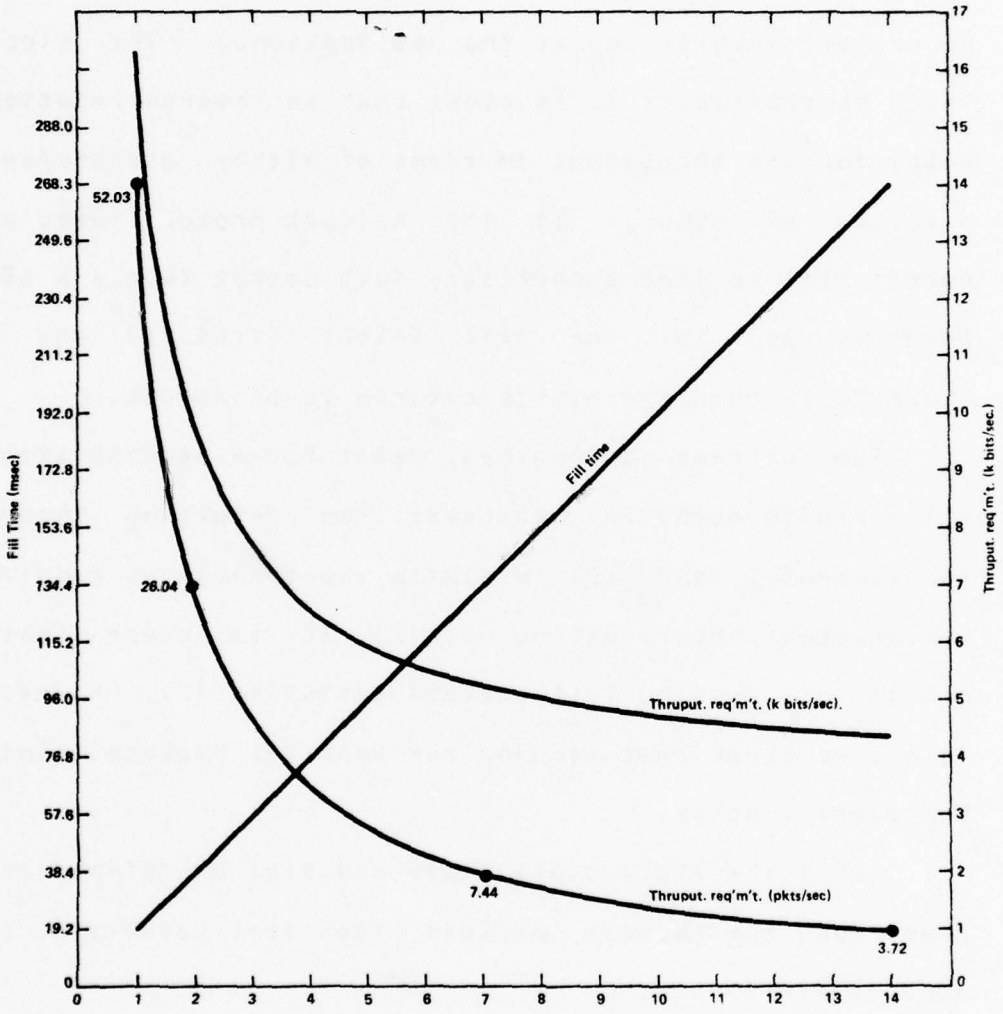


Figure 5.1. Fill time – through put requirement tradeoff.

We wish to minimize

$$f(p) + x(p) + c(p) + w(p) + D(p) \qquad (5.1)$$

over the choice of p between 1 and 14. While f(p) and x(p) increase with p, w(p) and D(p) decrease as p increases. Since channel utilization decreases as p increases we expect the average waiting time to decrease. This in turn causes a decrease in the variation of delay and therefore D(p) may be decreased to give the same gap probability (see section 5.3). Also, for the same (packet) gap probability, as the number of packets increases (as p decreases) more parcels are effected by gaps, and thus D must be increased to provide equal intelligibility. Therefore is is difficult to find that p which minimizes overall delay.

Between these two extreme sending policies there are many alternative approaches (e.g., (a) minimize delay at "reasonable" throughput requirement). Beyond p=5 the throughput curves are relatively flat (i.e., the throughput requirement does not change significantly between p=5 and p=14). Therefore a choice of p=5, 6, or 7 seems to fit approach (a).

Variable packet size

One would like to go a step further in minimizing equation 5.1, and use packets of variable size. For example, one may wish to use smaller packets when w is likely to be large thus reducing f and x, and conversely send larger

packets when w is likely to be small enough to guarantee that overall delay will not exceed a tolerable level.

An examination of this approach has shown that on a microscopic scale (i.e., packet by packet) it is infeasible since it was found that future delay is not well predicted by past delay in general. The correlation results presented in section 5.3.3.2 attest to this. Therefore during the short term it is wise to use fixed p.

On a macroscopic scale some adjustment of p may prove useful. This may be accomplished by using the destination monitoring techniques discussed in section 5.3. As network delay increases one may wish to use smaller p. This may be done only when w is not significantly increased by this decrease in p. If network delay increases quite substantially then an increase in p is in order (to reduce w).

We have offered here only ad hoc suggestions for the sending process. With a good model for w(p) some useful analytic results would be attainable. But since, as pointed out earlier, the throughput requirement curves are fairly flat past p=5, the payoff gained by such analysis is likely to be minimal. We shall therefore direct our attention to the receiving process.

5.3 Adaptive receiving

The aim of adaptive receiving of stream traffic is to output the information with as close to the same timing with which it was originally generated under the constraint of

allowing interactive communication (i.e., reasonably small delay). It is particularly important in the case of speech to reduce the frequency and duration of gaps in order to insure acceptable intelligibility. In this section we first describe three methods of gap control using buffering, and two playback strategies for handling those gaps which slip through the control procedures. The performance of the various schemes is then analyzed. We then present some numerical results for the theoretical performance, followed by simulation results comparing the various procedures.

5.3.1 Gap control

Since variable delays do occur in a packet switched network, if no smoothing were done, gaps would occur in the output of a packet stream. That is, there would be periods of time in which no data was available for playout. This has disastrous effects on the understandability of speech in particular and in general violates the definition of stream traffic. Therefore gap prevention or reduction is necessary.

One method of gap control is to delay the output of the beginning of a sentence (i.e., a period of activity by the sender) by an amount D, which may be selected at the beginning of each sentence based on sampling of previous delays. Another scheme is to slow the playout in the absence of the next packet and speed playout when an excess of waiting packets exists. (The effect of this on intelligibility may

not warrant its use.) One may mix these two to arrive at still others or invent new ones. We shall consider only the first scheme.

The first packet of a sentence will be delayed by an amount called the destination wait time denoted as D. The choice of D should be made large enough to reduce the frequency of gaps to a tolerable level but small enough to retain the interactive nature of a conversation. In a later section we shall consider several adaptive methods for choosing D.

5.3.2 Playback methods

Unless D is very large, there is a non-zero probability that a gap will occur. The purpose of this section is to describe two methods of dealing with this eventuality. The first method (method E) would expand the playout time of a sentence in order to include all packets in the output process. The second method (method I) preserves the timing at the expense of ignoring some late arriving packets (or partial packets). These two approaches lie at opposite ends of a continuum of choices for dealing with gaps. At constant delay (e.g., dedicated channel) the two extremes are equivalent. Also if D is infinite the extremes coincide. However, with finite D and variable delay the extremes separate. The separation increases as delay variability increases or as D decreases. At infinite delay variability method E requires infinite time to output a sentence (of

more than one packet), while method I would output only the first packet of a sentence and ignore any other packets. Thus method E preserves the information at the expense of the interpacket timing, and method I preserves the interpacket timing at the expense of discarding some information. With finite delay variability and finite D one can envision methods which lie between the two extremes. For example, one may wish to discard only those packets which arrive both late and out of order (i.e., a packet is discarded if its successor is currently being output). Such a scheme has properties of both methods E and I. The time axis is expanded when a packet is just late, yet some data may be discarded in order to preserve "reasonable" timing. Another example is to limit the expansion of time and/or the fraction of discarded data to a certain amount and switch methods if the threshold is exceeded.

An important consideration, which shall not be discussed here, is the filling of gaps (i.e., with silence or something else). Several alternatives (which are beyond the scope of this discussion) have been used and are discussed in [Forg 76].

5.3.2.1 Expanded time axis (Method E)

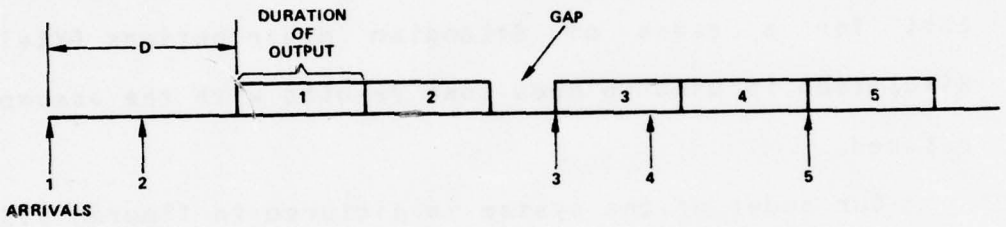
In this method content is regarded as the most important part of the stream of packets. Therefore, it is desired to deliver all packets which are produced by the source of the stream traffic. Timing is considered

important only to the extent that a packet is delivered in the same sequence in which it was generated. A "late" packet is one which arrives after it is desired for output (i.e., after its predecessor has finished playout). In this method a late packet causes all succeeding packets to be delayed in playout, thus expanding the time axis or the length (in time) of the sentence. An example of this is illustrated in Figure 5.2(a).

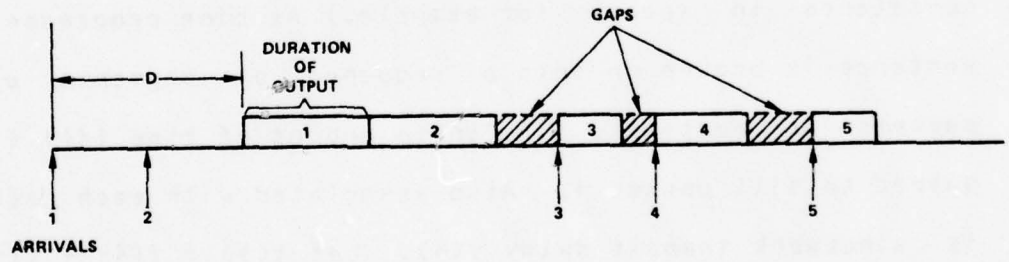
This is the general approach taken by the network speech compression research group at the University of Southern California Information Sciences Institute (ISI). In the ISI scheme extremely late out of order packets are ignored.

5.3.3.2 Late data ignored (Method I)

In method I timing is considered to be of primary importance. A packet (or partial packet) will be used only when it is not late. By discarding late information it is always possible to retain the sentence length (in time). Figure 5.2(b) shows the previous example sentence from Figure 5.2(a) passing through method I. Notice that sentence time is preserved but that more gaps occur than in method F. This is true in general as we shall see in later sections. Method I is in use by experimenters at the Massachusetts Institute of Technology Lincoln Laboratory.



(a) Method E



(b) Method I

Figure 5.2. Playout Methods.

5.3.3 Performance evaluation

An examination of the performance of several adaptive receiving techniques is presented in this section. We begin by considering some assumptions which render the system tractable for analysis. The analysis is then performed. An attempt to characterize communication quality is made. Numerical results based on the analysis are presented in detail for the exponential distribution, and in lesser detail for a class of Erlangian distributions [Klei 75]. Simulation is used to show some results with the assumptions relaxed.

Our model of the system is pictured in Figure 5.3. A period of activity, called a sentence, is initiated by the sender at some time. (This corresponds to the detection of no-silence in speech for example.) As time progresses the sentence is broken up into a sequence of segments called packets or messages. A finite amount of time f(i) is required to fill packet i. Also associated with each packet i is a network transit delay y(i). Let t(i) = f(i) + y(i) be the source-to-destination delay of packet i. We make the following assumptions regarding the random variable t(i):

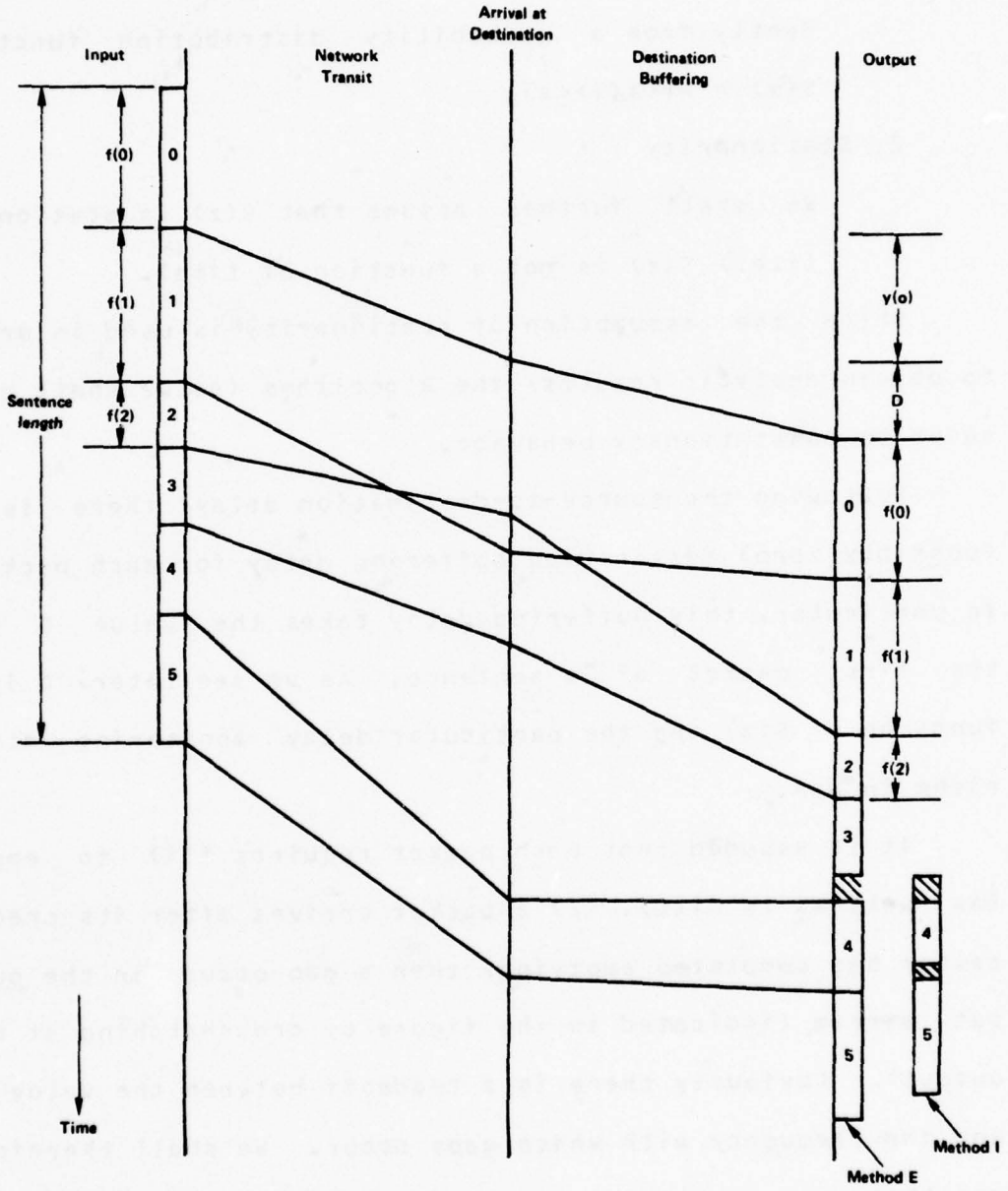


Figure 5.3. End-to-end communication.

Assumptions of the model

1. Independence

   We shall assume that each t(i) is chosen independently from a probability distribution function $S(z) = Pr[t(i) \leq z]$.

2. Stationarity

   We shall further assume that S(z) is stationary (i.e., S(z) is not a function of time).

While the assumption of stationarity is used in order to obtain analytic results, the algorithms (as we shall see) adapt to nonstationary behavior.

Following the source-to-destination delay there is a (possibly zero) destination buffering delay for each packet. In particular, this buffering delay takes the value D for the first packet of a sentence. As we see later, D is a function of S(z) and the particular delay monitoring algorithm in use.

It is assumed that each packet requires f(i) to empty (as well as to fill). If a packet arrives after its predecessor has completed emptying, then a gap occurs in the output stream (indicated in the figure by crosshatching at the output). Obviously there is a tradeoff between the value D and the frequency with which gaps occur. We shall therefore use these two parameters as our measures of performance. Let us first consider the validity and implication of the major assumptions of the model.

5.3.3.1 Distribution of network delay

The purpose of this section is to present some observed distributions of network delay, and provide guidelines for producing tractable approximations of these distributions. Both ARPANET measurement and simulation were used obtain delay distribution examples.

The approximation derived from the independence assumption of Kleinrock [Klei 64] would suggest an Erlangian (i.e., the sum of exponentials) distribution of delay. This assumption renders the model of each channel an M/M/1 queue which yields an exponentially distributed system time. The time through a series of queues therefore would follow an Erlangian distribution.

Exact formulas have been derived for the distribution of message delay in an isolated non-interfered path in a network, for Poisson arrivals and deterministic service time, by Rubin (see [Rubi 75]). Our problem is to find the delay distribution of a stream traffic source in a general network which does not seem to fit the assumptions of the previous two models. Namely the stream packets must not be allowed to change length as they proceed through the network, and our problem would be trivial in a non-interfered path.

5.3.3.1.1 Measurement

The measurement experiments described in Chapter 4 provide the distributions presented here. In order to

eliminate the start-up effect, the first 100 samples have been discarded in each case. The resulting histograms are plotted in Figure 5.4(a-d). In (a) and (b) there is a trimodal behavior exhibited. This shows the effect of the periodic routing update procedure quite clearly (again). The first mode corresponds to the round-trip delay experienced by a message which always finds empty queues along its path. The third mode, falling about 45 msec after the first mode, corresponds to the delay of those messages and corresponding end-to-end acknowledgements (RFNM) which must wait for a total of 45 msec on queues while traversing the network. This amount of time corresponds to the transmission time of two routing update packets, the processing of four such packets (approximately two in the slower 316 nodes), or some combination of the above. The second mode, occurring roughly 10 msec beyond the first, is probably due to the packet (or its RFNM) waiting behind the processing of one routing update along the way. Significance could probably be assigned to the other peaks as well but this would belabor the point.

As we move to five hops (c) and then to ten hops (d) we notice that the trimodal behavior ceases to appear and the histograms take the shape of an Erlangian density. Perhaps Kleinrock's independence assumption produces an acceptable approximation for stream traffic as well!

5.3.3.1.2 Simulation

The results of the measurement are biased by the large overhead associated with the routing update procedure. We therefore resort to simulation in order to remove this effect, to control the level of interfering traffic in the network, and to study one way delay instead of round-trip delay. The simulation had the following characteristics:

Topology: A ring of 21 nodes and 21 full duplex channels

Channel capacity: 50000 bits/sec

Traffic pattern: A uniform traffic matrix of exponentially distributed message lengths with a mean of 500 bits and Poisson arrivals as background traffic. Inserted with this was one stream traffic source sending to a destination ten hops away at a rate of one 500 bit packet every 250 msec.

The background traffic was set to three particular levels in order to produce .1, .5 and .9 channel utilization. One-way network delay was measured for the stream traffic and the resulting histograms appear in Figure 5.4(e-g). The .1 load histogram (e) suggests a shifted exponential density with perhaps an impulse at the shift value. For load .5 and .9 (f and g respectively) the histograms have more of an Erlangian shape (with an impulse in the .5 case).

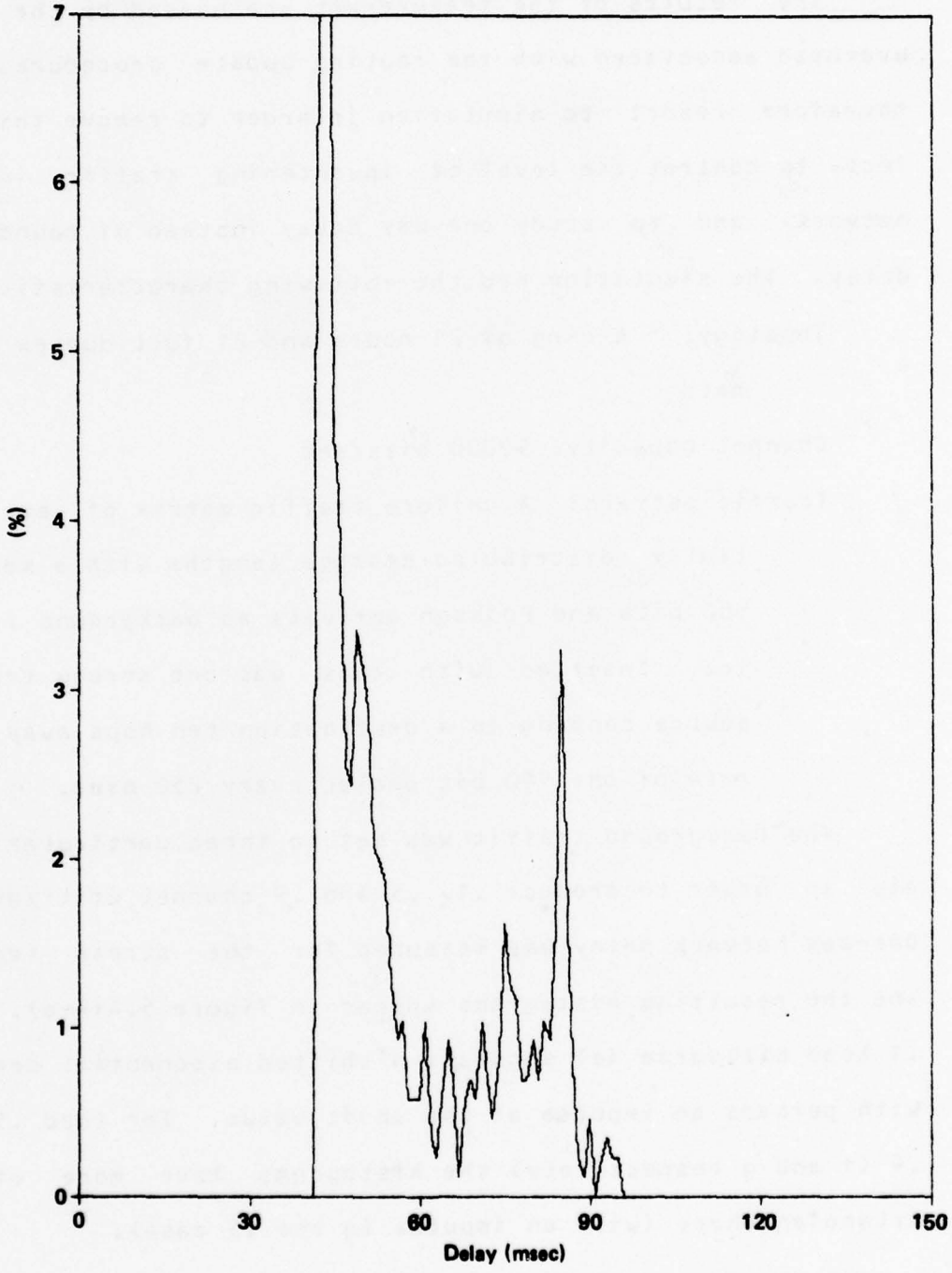


Figure 5.4(a). Delay histogram (measurement, 1 hop).

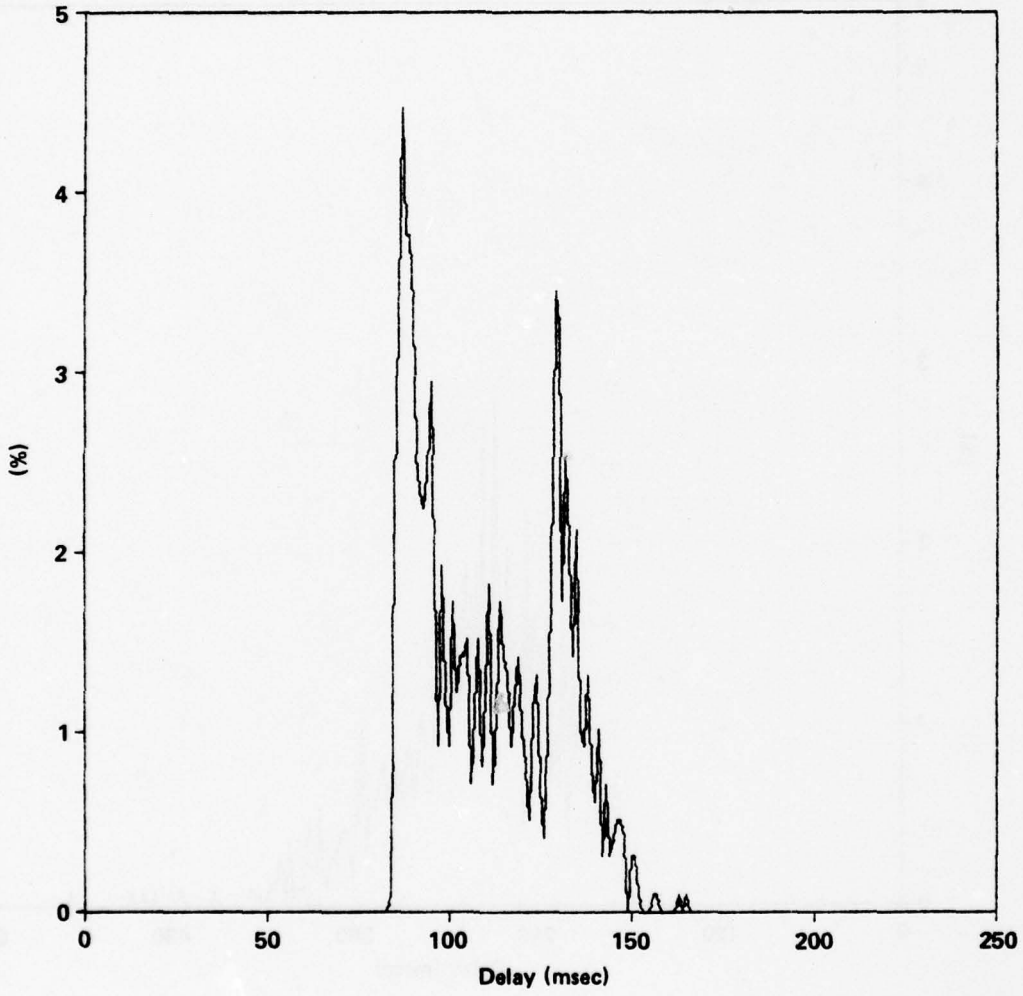


Figure 5.4(b). Delay histogram (measurement, 2 hops).

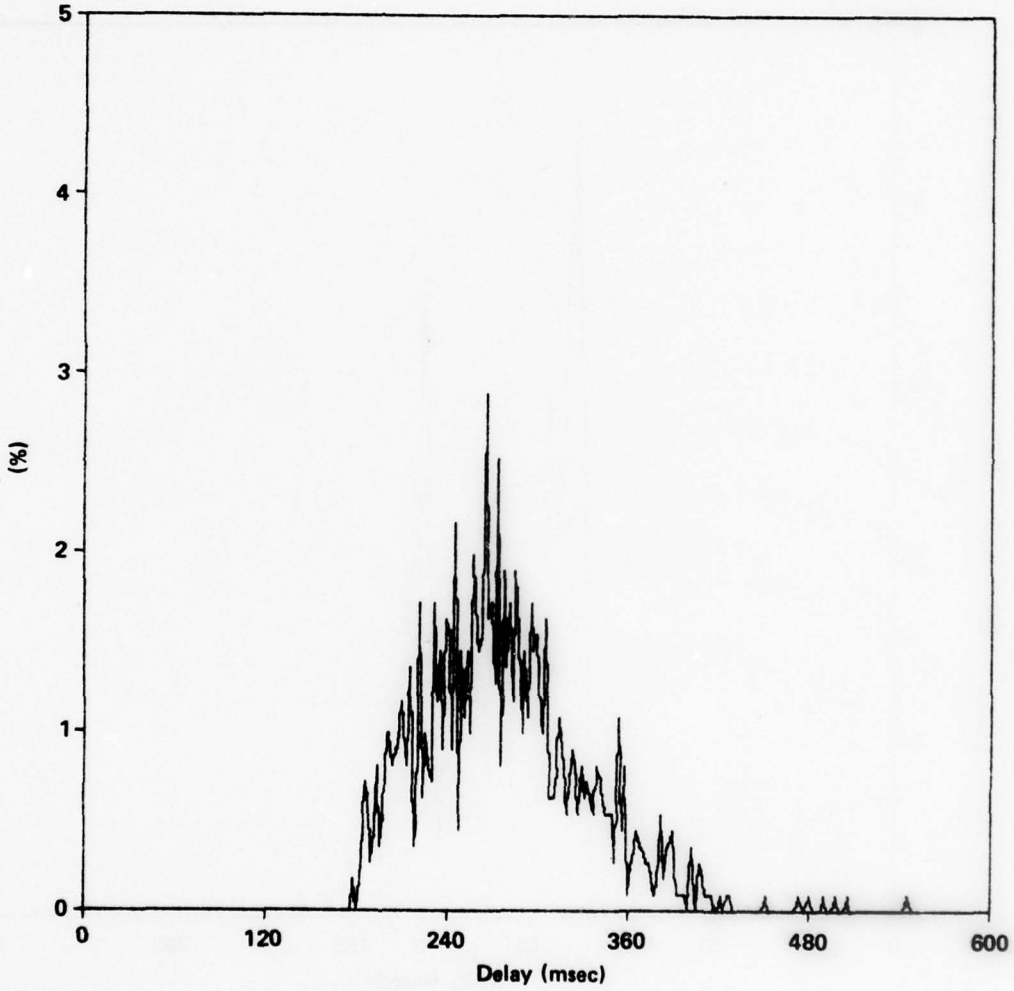


Figure 5.4(c). Delay histogram (measurement, 5 hops).

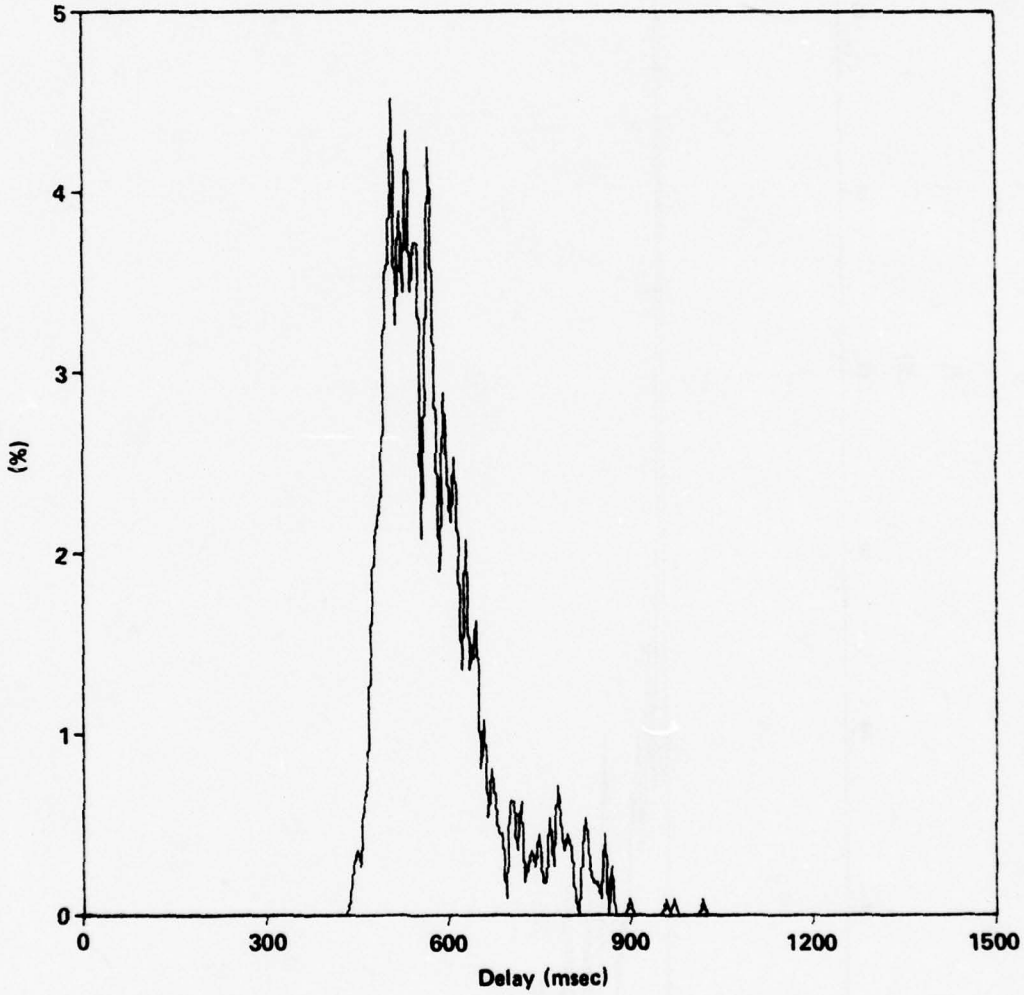


Figure 5.4(d). Delay histogram (measurement, 10 hops).

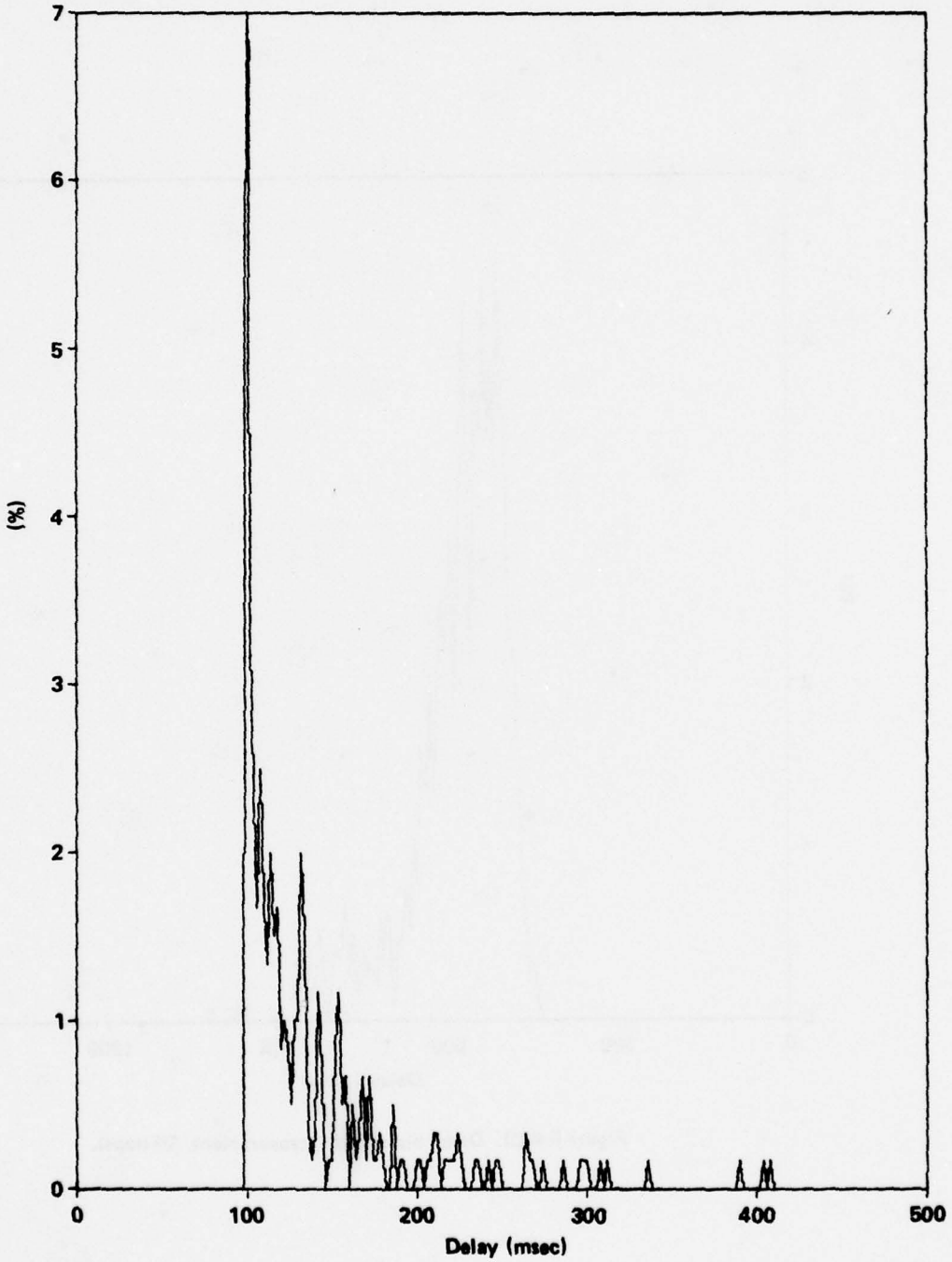


Figure 5.4(e). **Delay histogram (simulation, load= .1).**

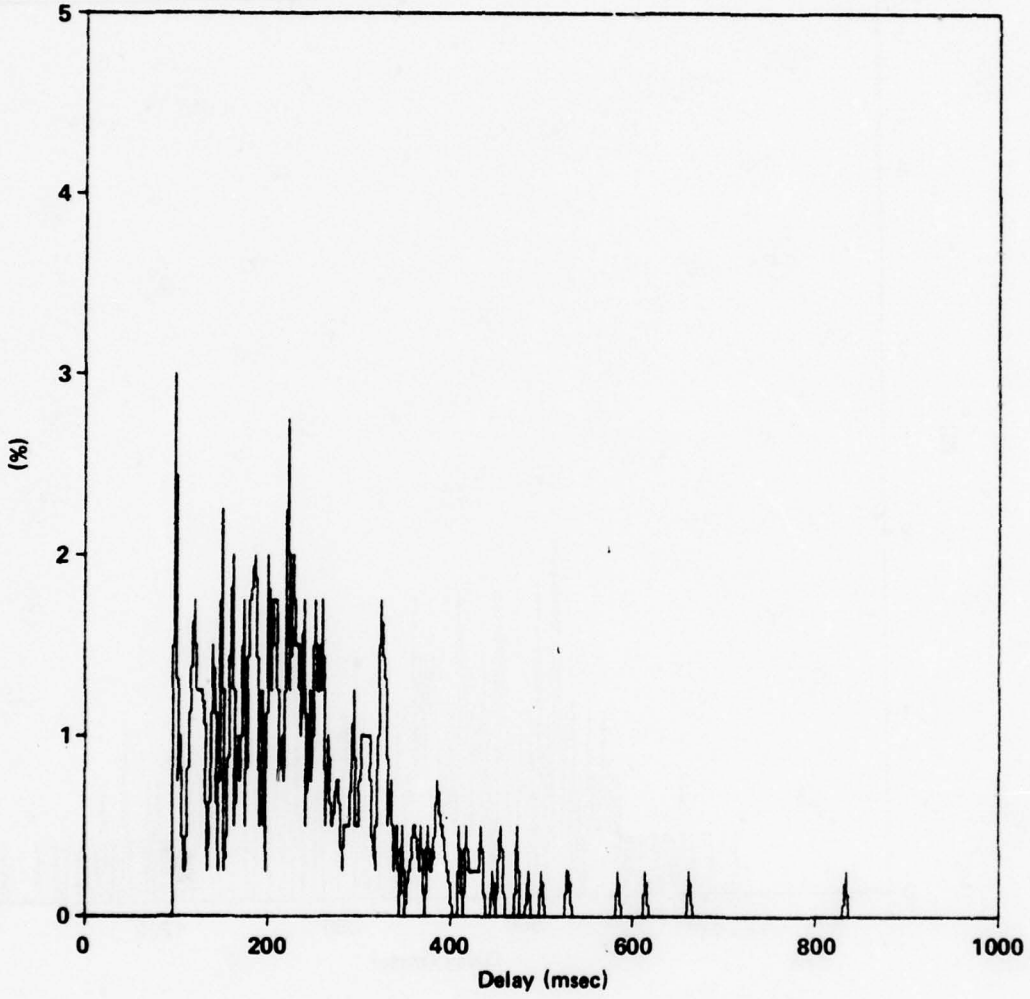


Figure 5.4(f). Delay histogram (simulation, load= .5).

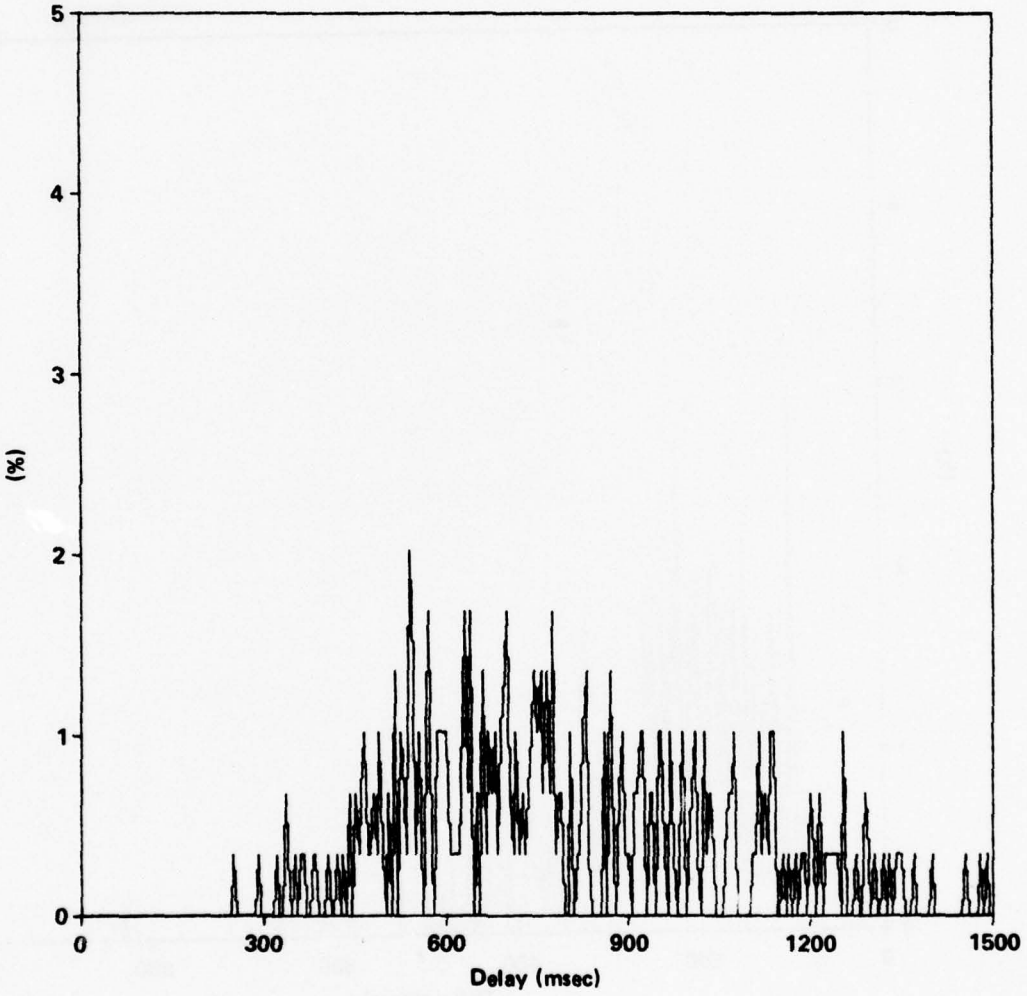


Figure 5.4(g). Delay histogram (simulation, load= .9).

5.3.3.1.3 Tractable approximations

In order to obtain some analytic results we must assume that the delay distribution may be characterized in a mathematically tractable form. In this section we consider some tractable approximations to the measured and simulated delay distributions. We have already suggested two - (1) the shifted exponential and (2) the Erlangian class.

The shifted exponential distribution

This distribution may be formulated as

$$S(t) = \begin{cases} 0 & t \leq b \\ 1 - \frac{1}{w} e^{-(t-b)/w} & t \geq b \end{cases}$$

where b is the amount of shift and b+w is the mean value. Figure 5.5 shows the histogram of the simulation in Figure 5.4(e) plotted together with a shifted exponential with the same mean value and with b equal to the minimum observed value. This appears to be a fairly close fit.

The Erlang family of distributions

The shifted exponential was a close fit to one of the sample distributions. We now consider the Erlang family which may be formulated as

$$dS(t) = \frac{\frac{r}{x}\left(\frac{rt}{x}\right)^{r-1} e^{-rt/x}}{(r-1)!}\, dt$$

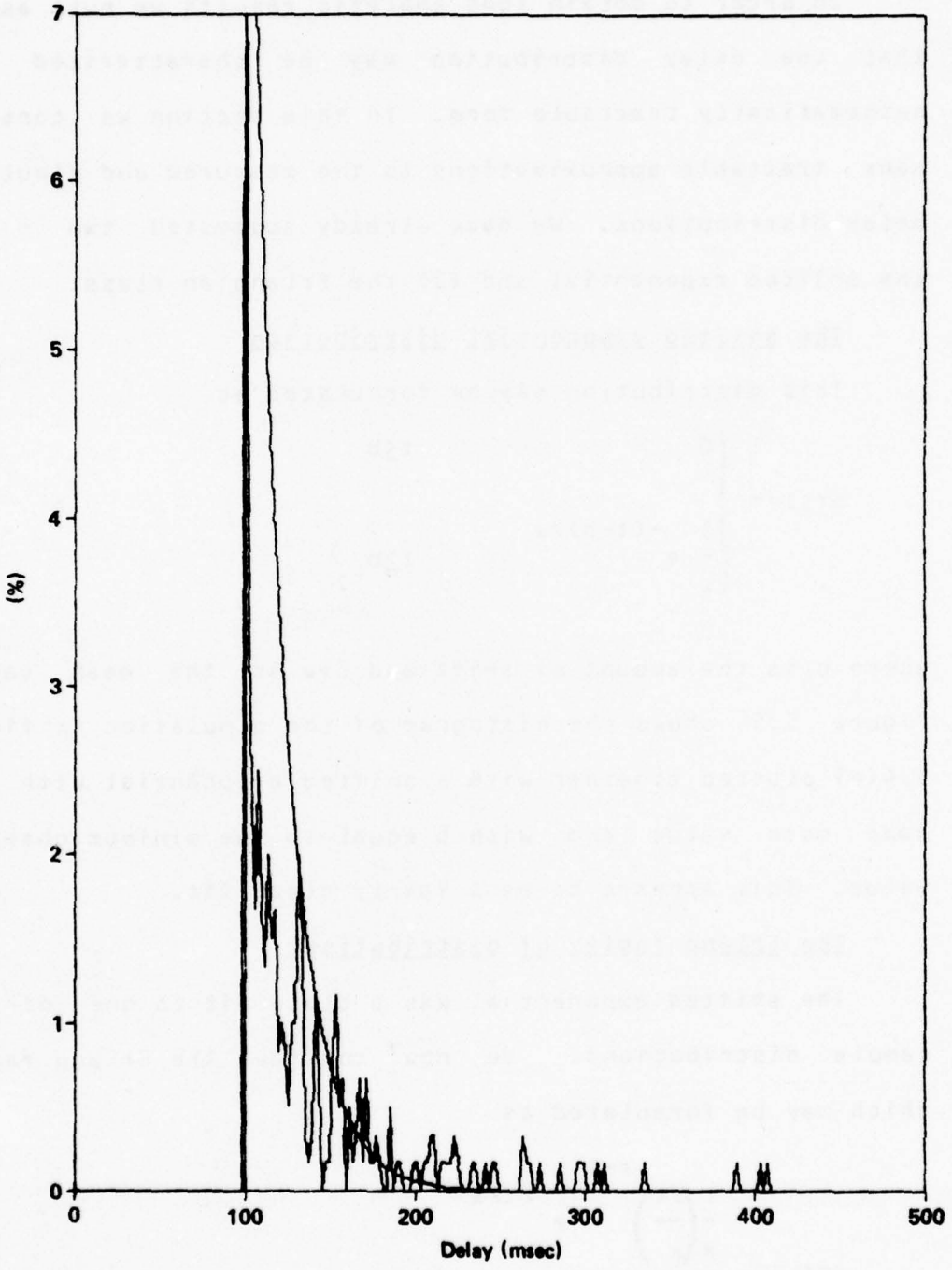


Figure 5.5. Shifted exponential fit to delay histogram (simulation, load = .1).

The formula has been written so that mean value is independent of r. As Kleinrock [Klei 75] points out on page 124, the standard deviation of this distribution is x/sqrt(r). (Figure 5.26 shows this density function for several values of r.)

By selecting r one may change the coefficient of variation (defined to be the standard deviation divided by the mean) between zero and one. All the observed distributions have coefficients of variation in this range. Figure 5.6 shows those observed histograms (which appear to be Erlangian) plotted together with the member of the Erlang family of the appropriate r and mean value. This appears to be a close fit, particularly for the simulation.

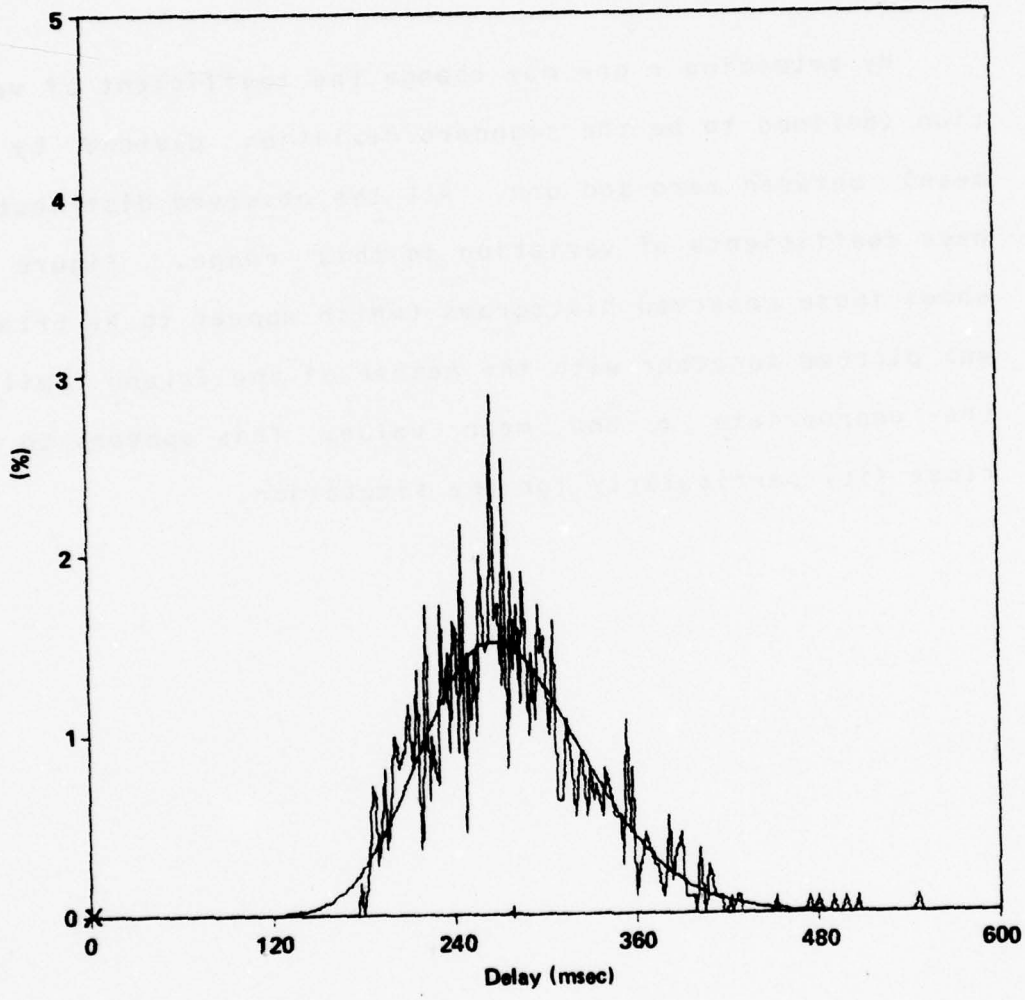


Figure 5.6(a). Erlang fit to delay histogram (measurement, 5 hops).

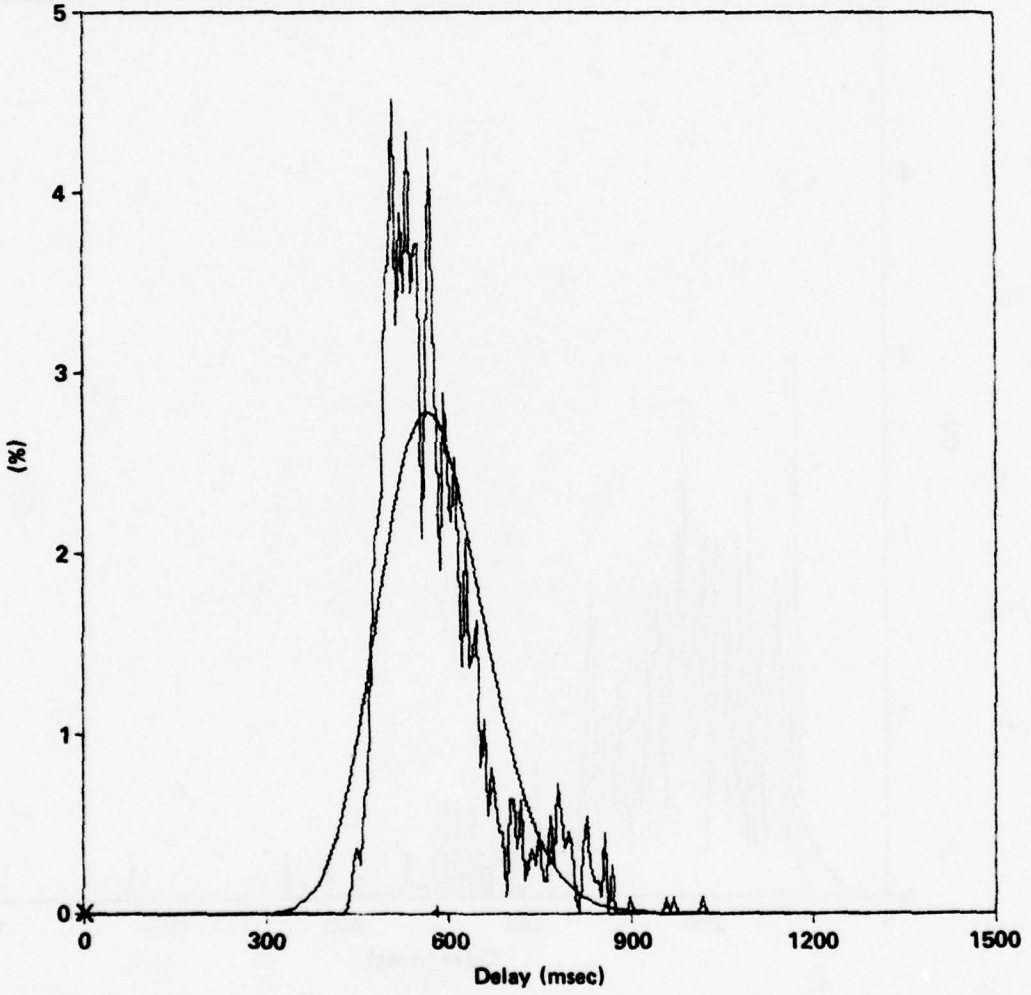


Figure 5.6(b). Erlang fit to delay histogram (measurement, 10 hops).

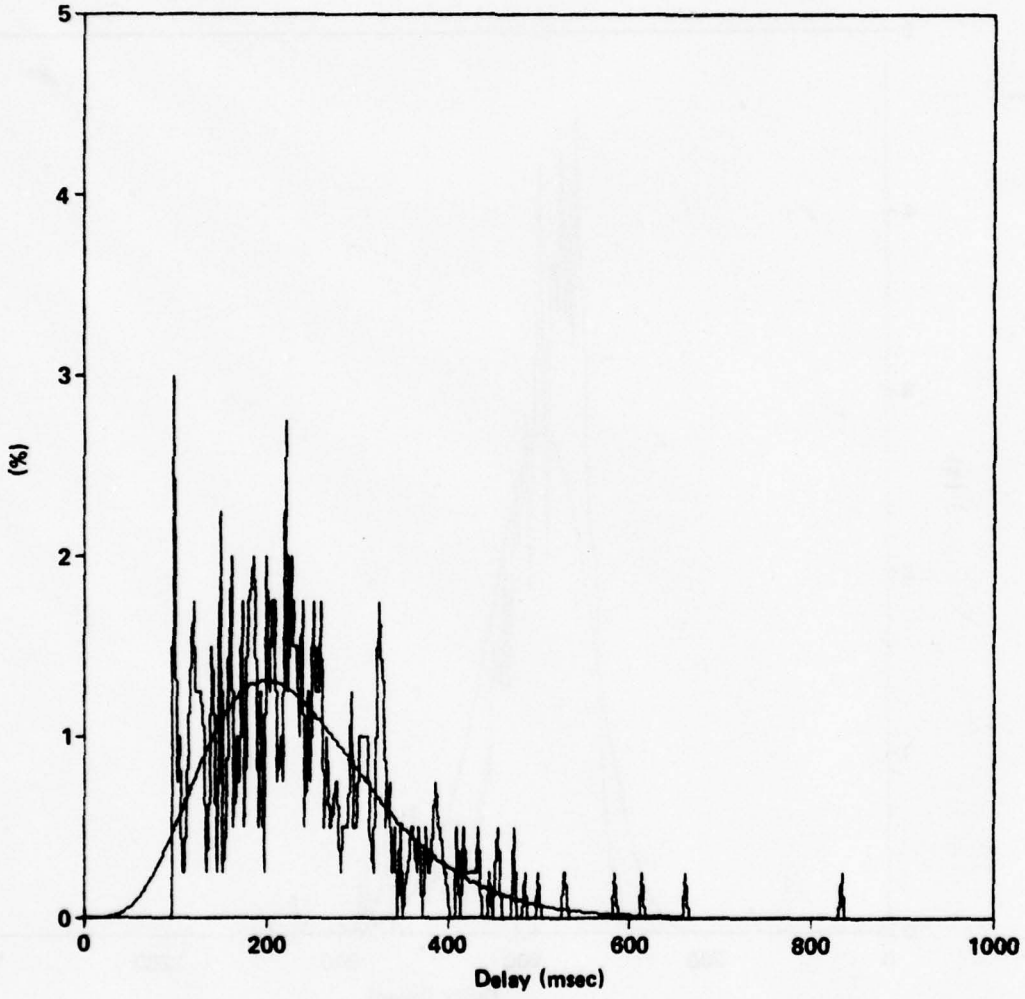


Figure 5.6(c). Erlang fit to delay histogram (simulation, load = .5).

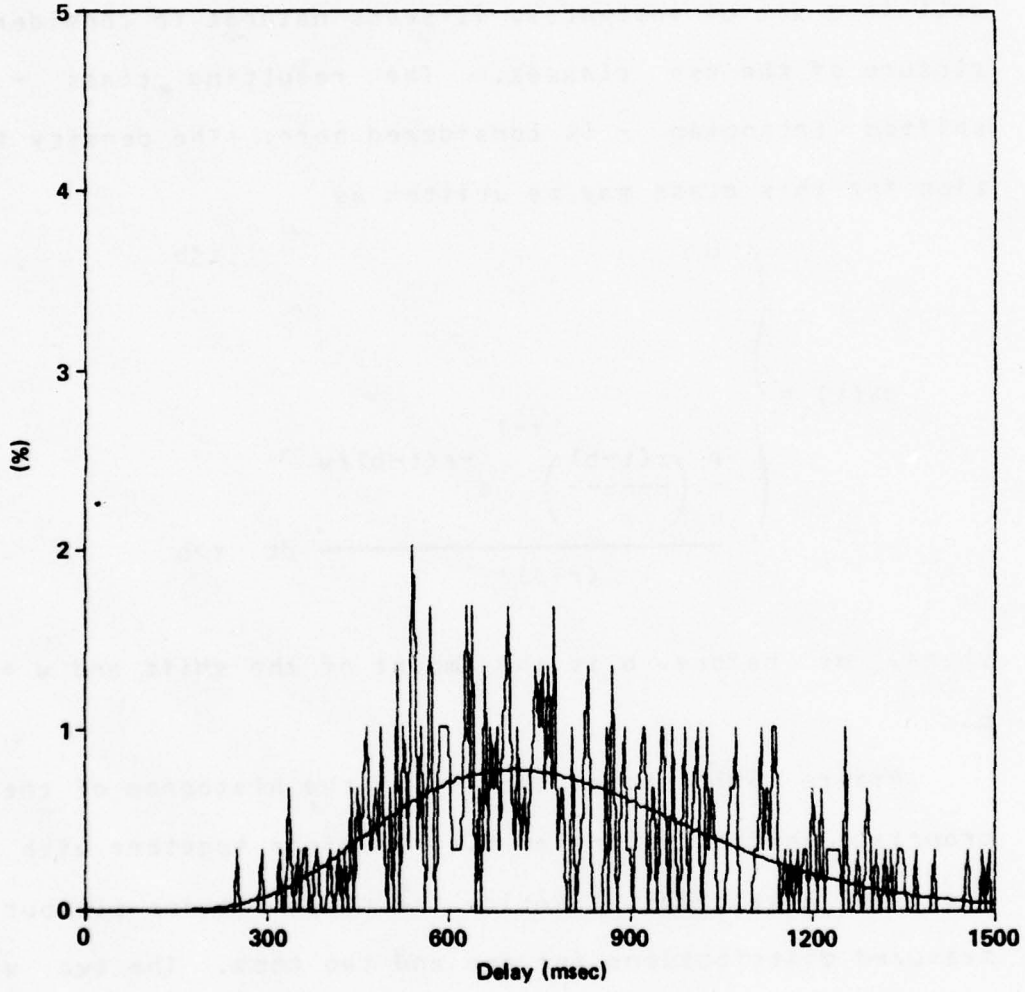


Figure 5.6(d). Erlang fit to delay histogram (simulation, load = .9).

The shifted Erlangian distributions

So far we have seen that neither the shifted exponential nor the Erlang family is by itself sufficiently rich as to model all of the observed behavior. Since each class did well in a set of instances, it seems natural to consider the closure of the two classes. The resulting class - the shifted Erlangian - is considered here. The density function for this class may be written as

$$dS(t) = \begin{cases} 0 & t \leq b \\ \dfrac{r}{w}\left(\dfrac{r(t-b)}{w}\right)^{r-1} e^{-r(t-b)/w} & \\ \hline (r-1)! \quad dt & t > b \end{cases}$$

where, as before, b is the amount of the shift and w = x - b.

Figure 5.7 shows the fit of the histogram of the appropriate shifted Erlangian distributions together with the observed histograms. Notice the close fit for all but the measured distributions for one and two hops. The two which are not well approximated with the shifted Erlang class could it appears be approximated by a waited sum of shifted exponentials, but we shall not attempt this here.

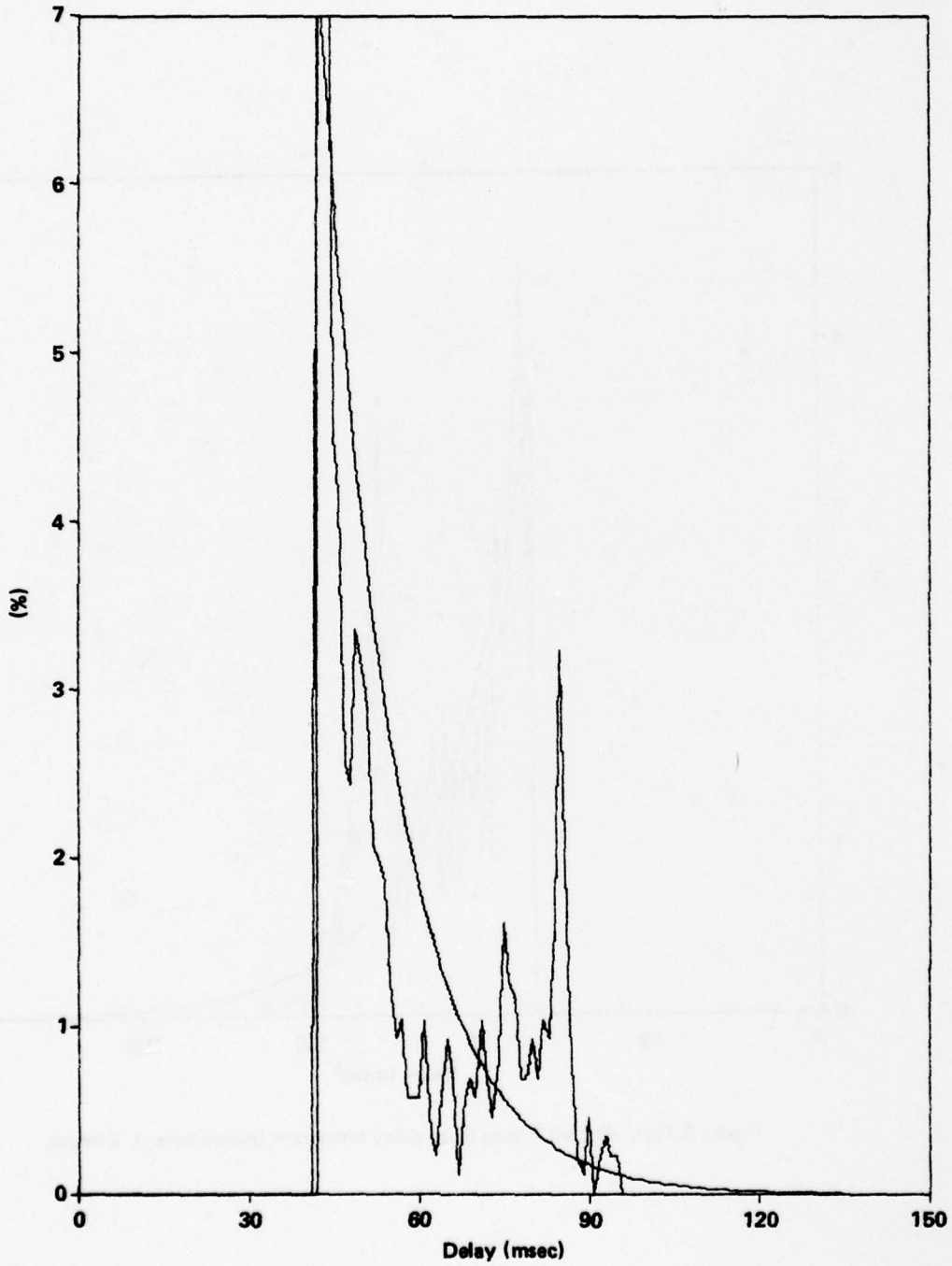


Figure 5.7(a). Shifted Erlang fit to delay histogram (measurement, 1 hop).

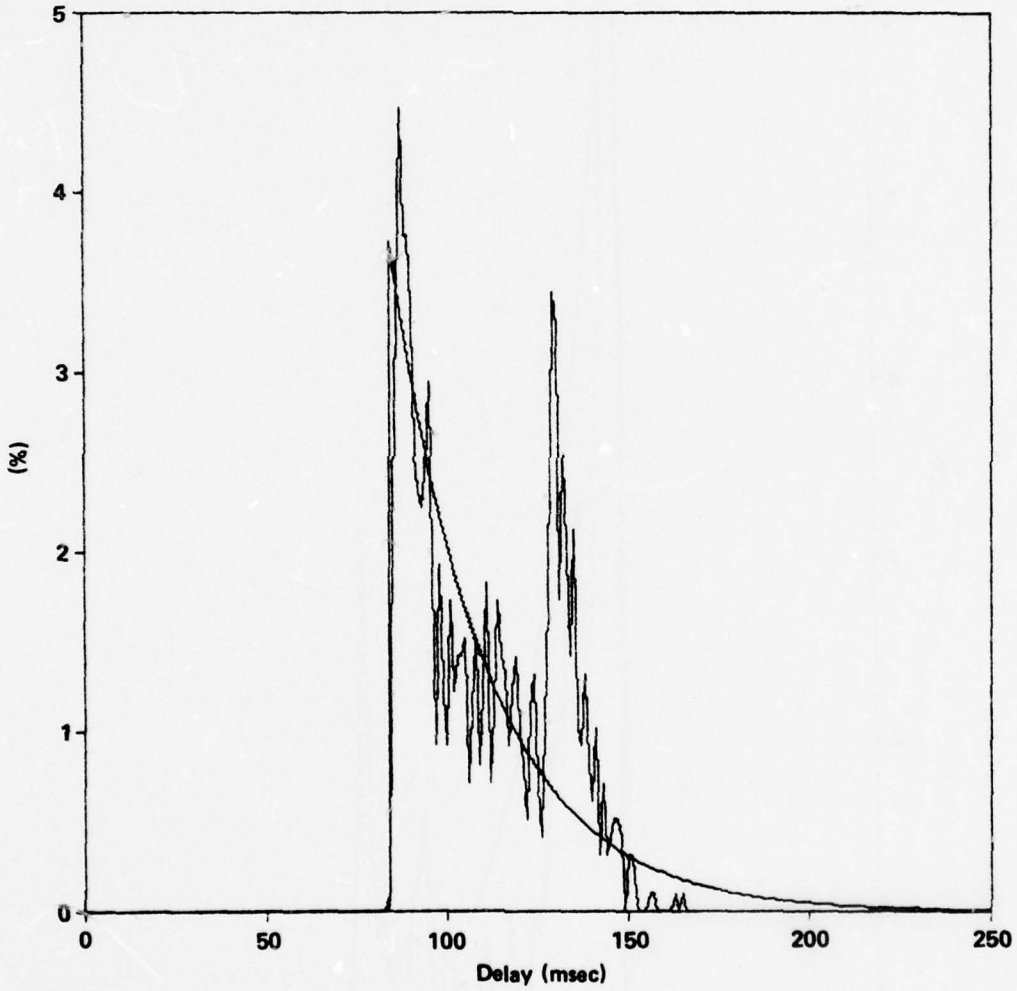


Figure 5.7(b). Shifted Erlang fit to delay histogram (measurement, 2 hops).

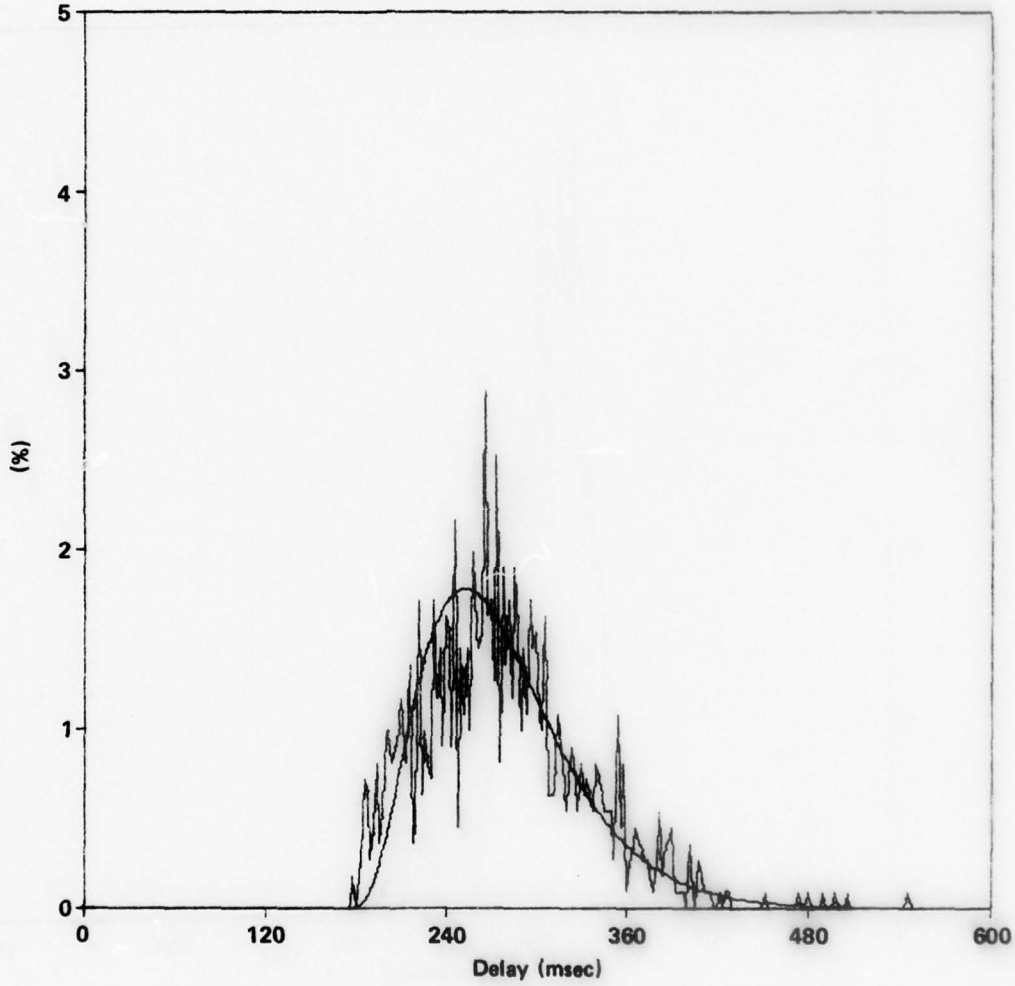


Figure 5.7(c). Shifted Erlang fit to delay hisogram (measurement, 5 hops).

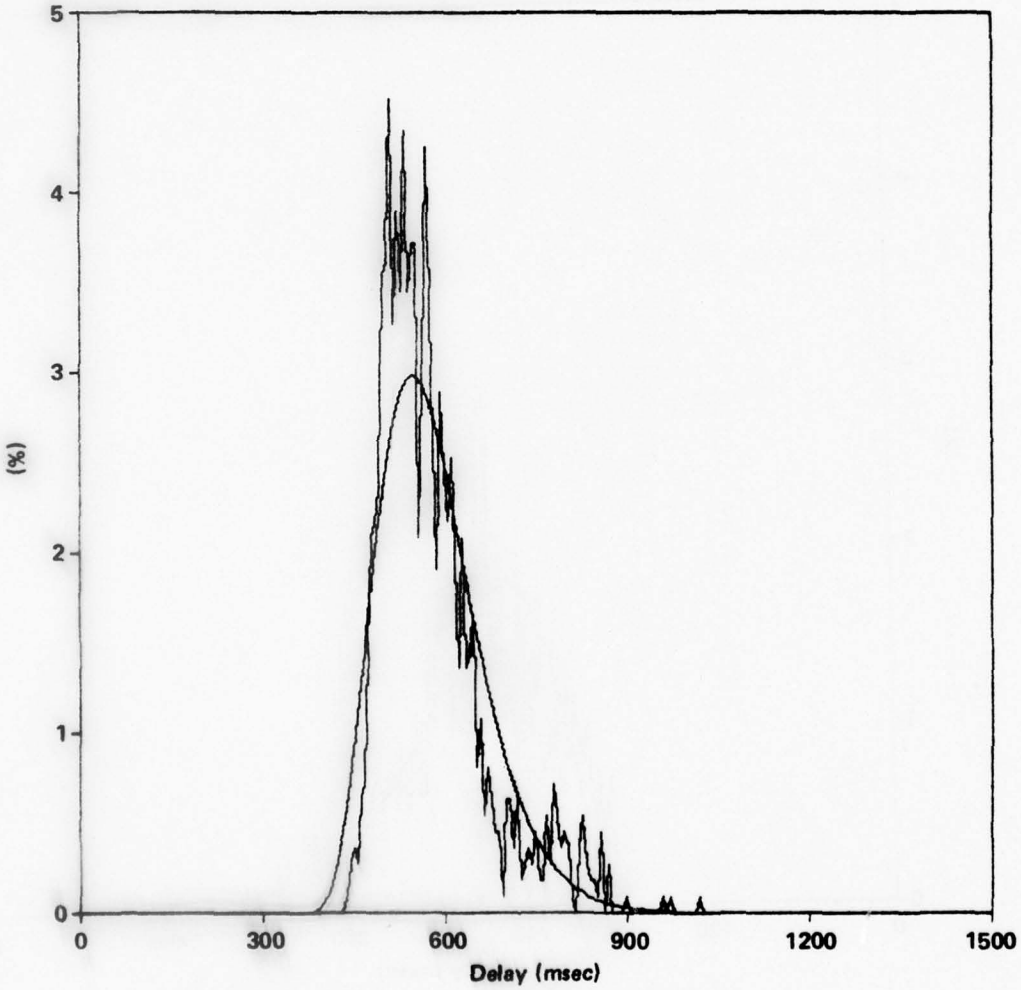


Figure 5.7(d). Shifted Erlang fit to delay histogram (measurement, 10 hops).

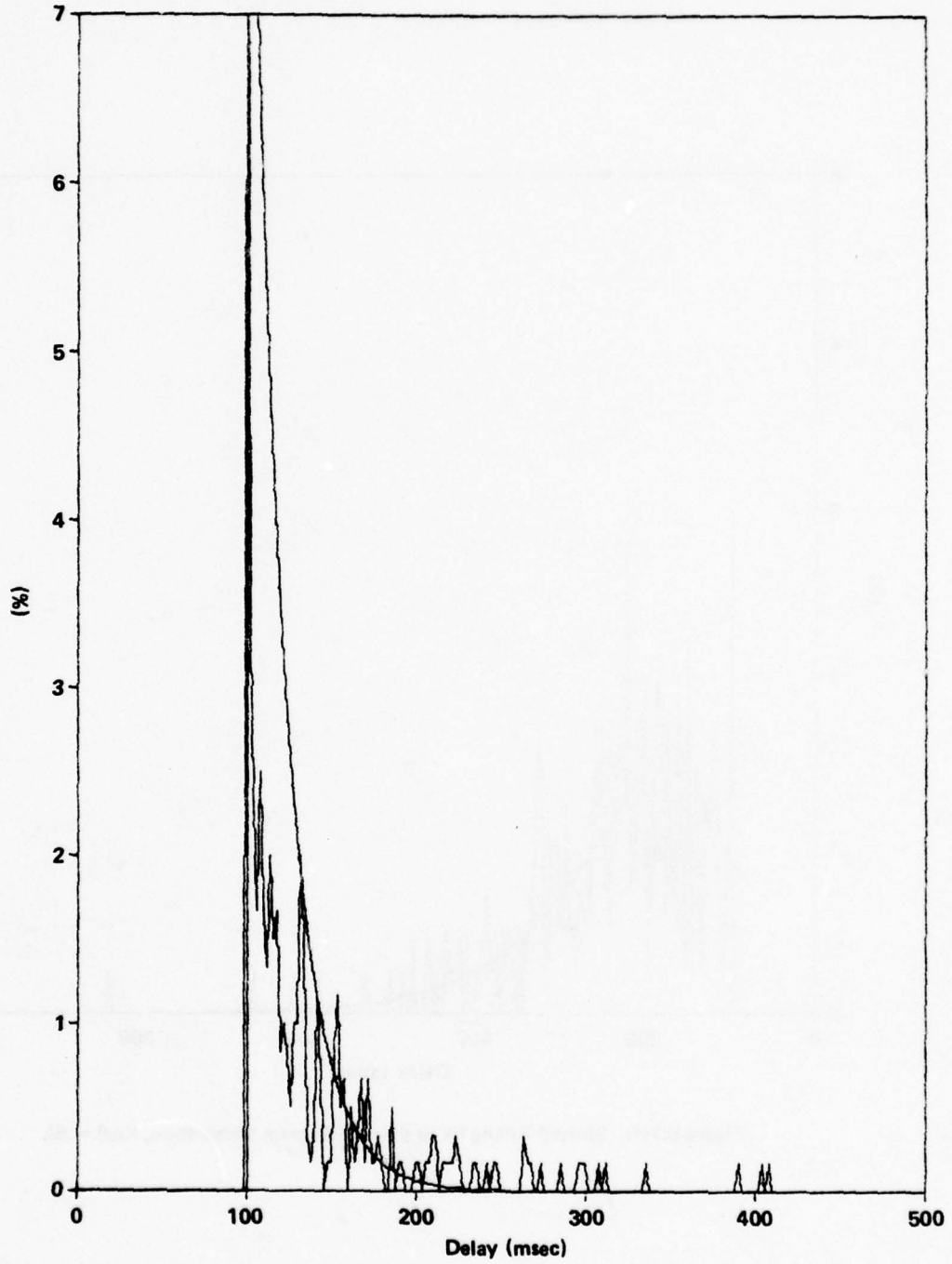


Figure 5.7(e). Shifted Erlang fit to delay histogram (simulation, load = .1).

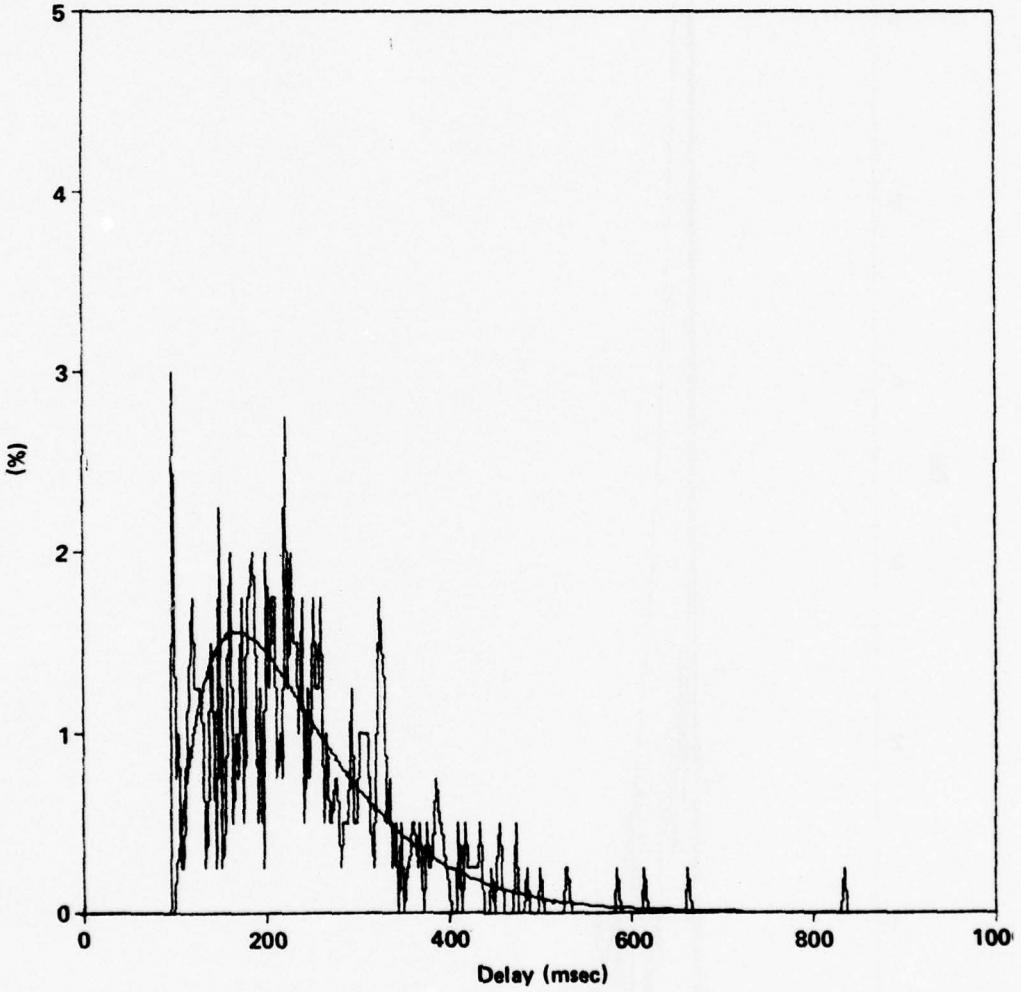


Figure 5.7(f). Shifted Erlang fit to delay histogram (simulation, load = .5).

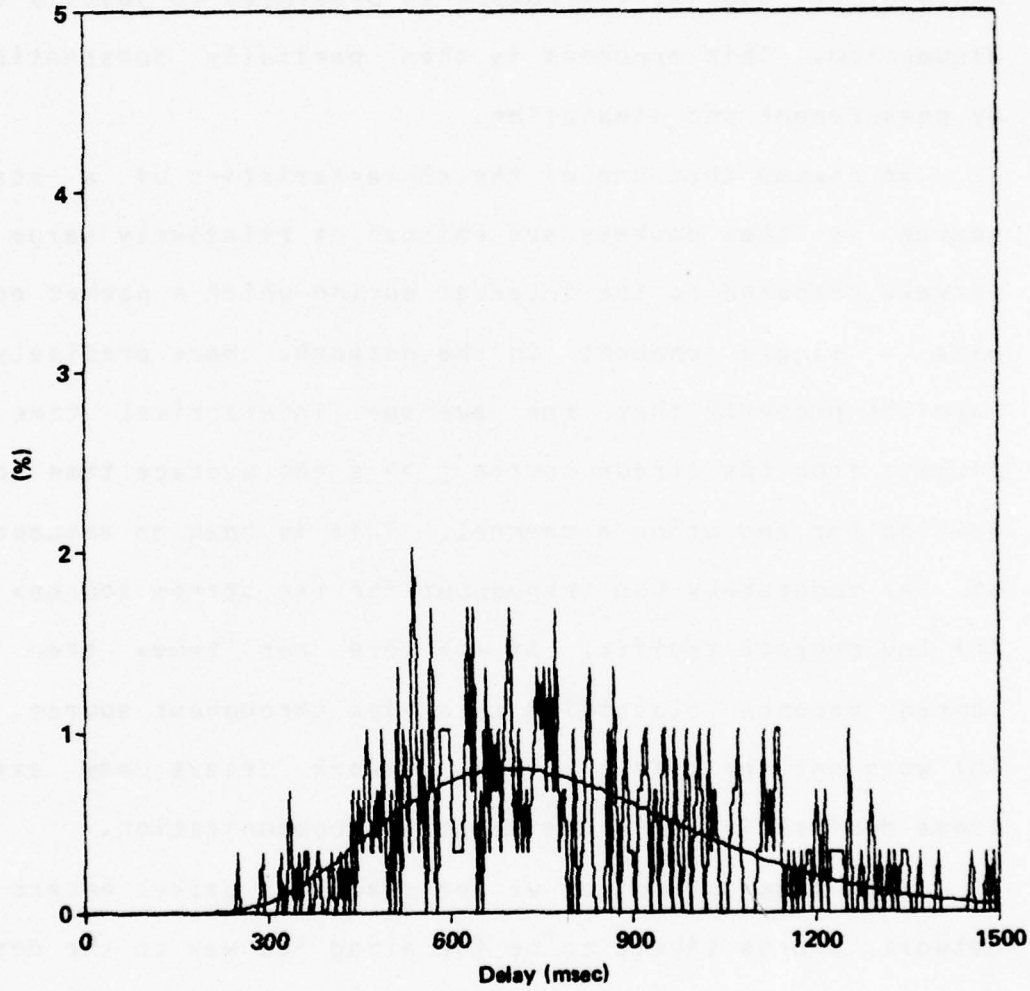


Figure 5.7(g). Shifted Erlang fit to delay histogram (simulation, load = .9).

5.3.3.2 Statistical independence of network delay

In the study of stochastic processes, statistical independence is often assumed to make analysis simpler (even possible). We make use of such an assumption here. In this section an heuristic argument is presented to justify this assumption. This argument is then partially substantiated by measurement and simulation.

We assume that one of the characteristics of a stream source is that packets are emitted at relatively large intervals compared to the interval during which a packet occupies a single channel in the network. More precisely we have the property that the average interarrival time of packets from the stream source t >> s the average time spent waiting for and using a channel. This is both an assumption of (a) moderately low throughput for the stream source, and (b) low overall traffic. If (a) were not true, then the source becomes classified as a high throughput source. If (b) were not the case then the network delays may exceed those desired for interactive stream communication.

With this assumption we see that each packet enters the network and is likely to be far along its way to the destination before its successor enters the network. Therefore each packet arrives to find the network in a state which is "independent" of the influence of his predecessors which are no longer in sight. Each packet then receives an independent "look", if you will, at the network. We conclude that the delay experienced by successive packets should be

approximately independent.

Let us examine the measurement and simulation results. A necessary, but not sufficient, condition for statistical independence is that of linear independence. Linear dependence may be tested simply by computing the correlation coefficient [Fell 57] (p. 221) of a sequence of delays with itself shifted by an amount j. By changing the shift j we obtain a sequence of correlation coefficients. This is related to the "autocorrelation" sequence defined in [Oppe 75] (p. 384). Our sequence however is normalized by first subtracting the mean from each value (which gives the "autocovariance" sequence) and dividing the result by the variance of the original sequence. The result of this computation for j = 1, 2, ..., 200 is shown in Figure 5.8(a-d) for the measured sequences of delay and in Figure 5.8(e-g) for the delay sequences produced by the simulation described in the previous section. The figures show that a near zero linear dependence exists for the measured delay of 5 hops or more and for a load of .1 in the simulation. For a load of .5 successive delays are less than 30 percent correlated but beyond j=1 there is little correlation. As expected delay at a load of .9 is highly correlated for the first few values of j. Parts (a) and (b) of Figure 5.8 show a regular pattern of correlation. This is further evidence of the periodic nature of the interference of the routing update procedure.

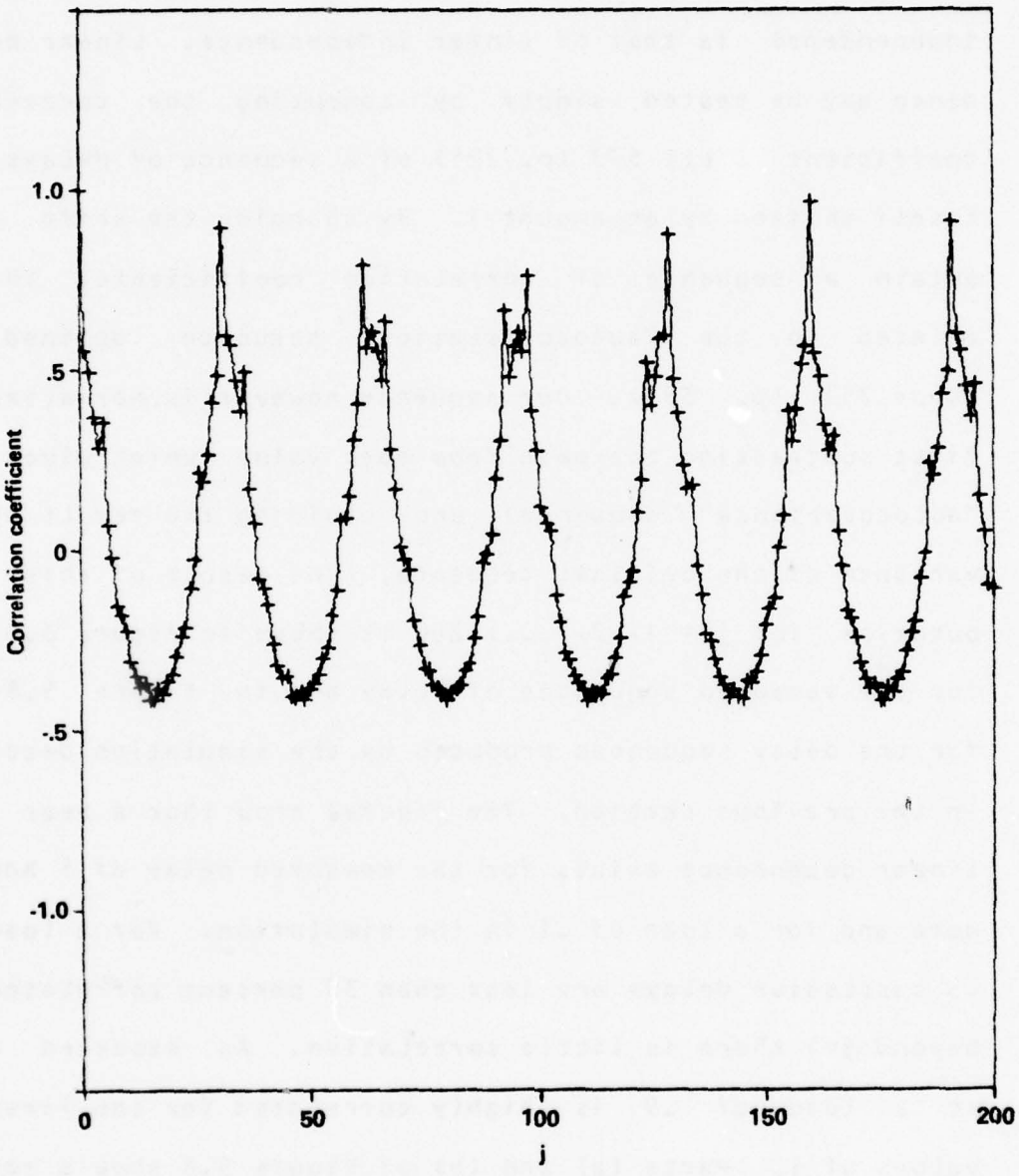


Figure 5.8(a). Delay correlation (measurement, 1 hop).

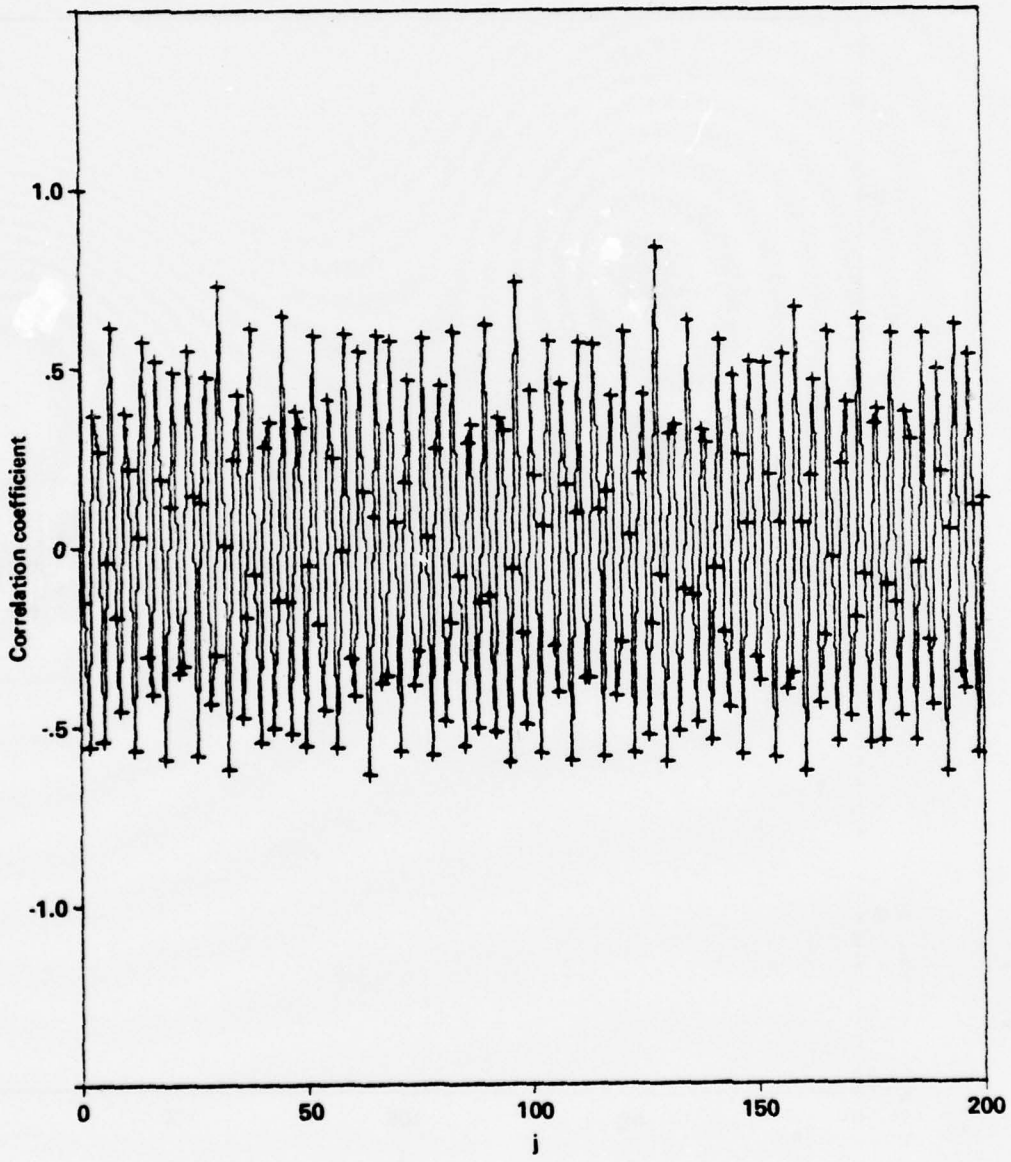


Figure 5.8(b). Delay correlation (measurement, 2 hops).

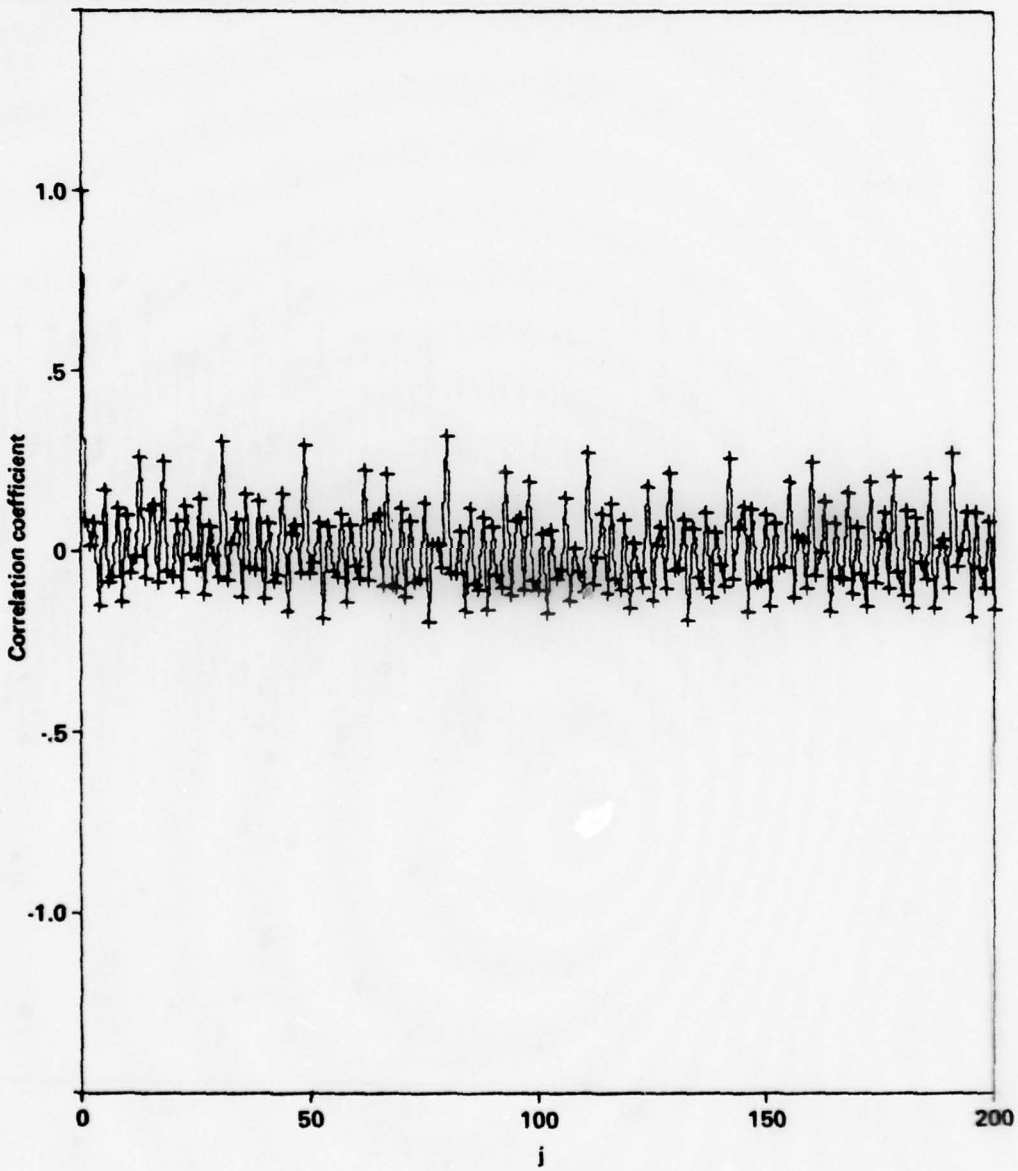


Figure 5.8(c). Delay correlation (measurement, 5 hops).

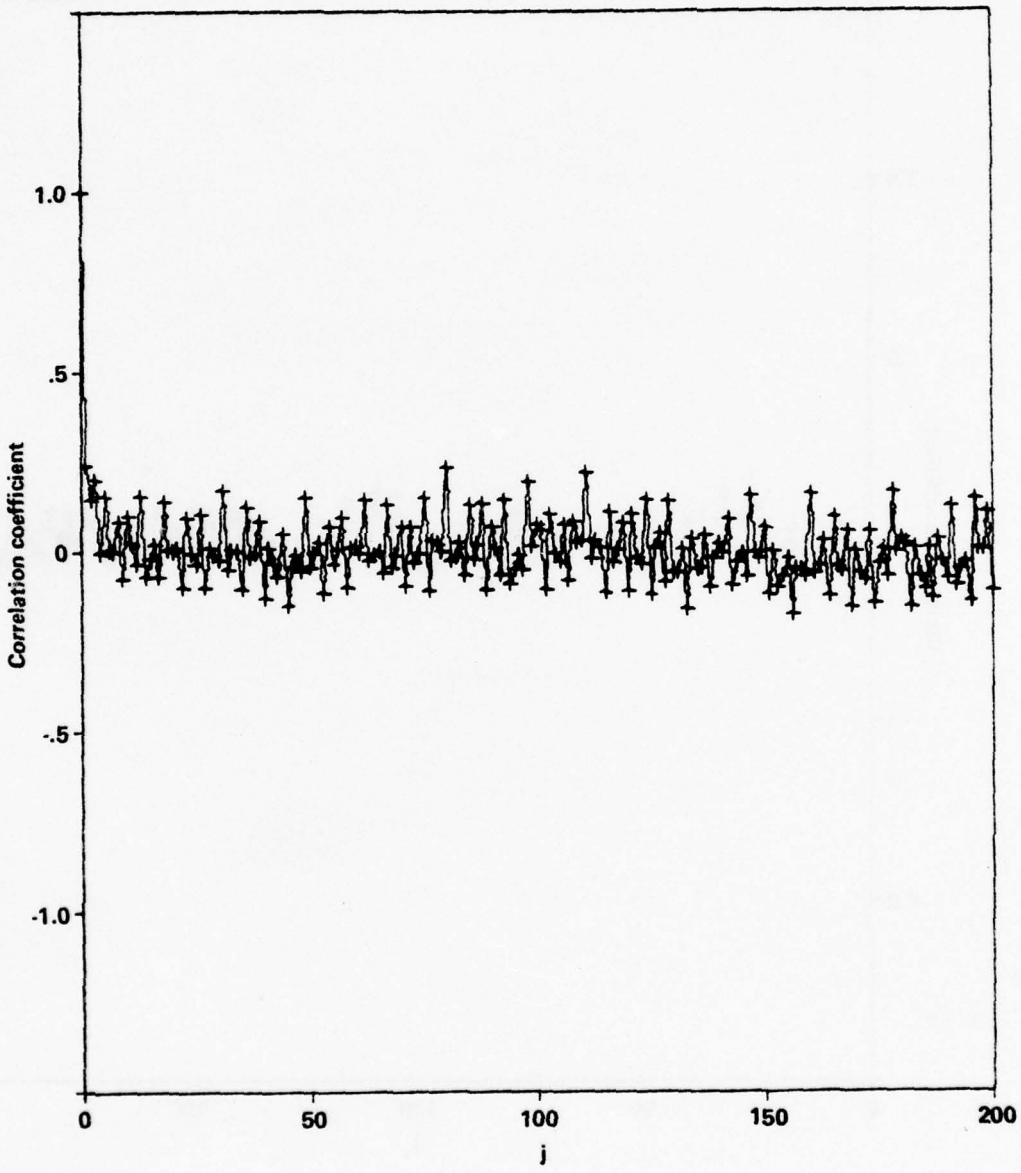


Figure 5.8(d). Delay correlation (measurement, 10 hops).

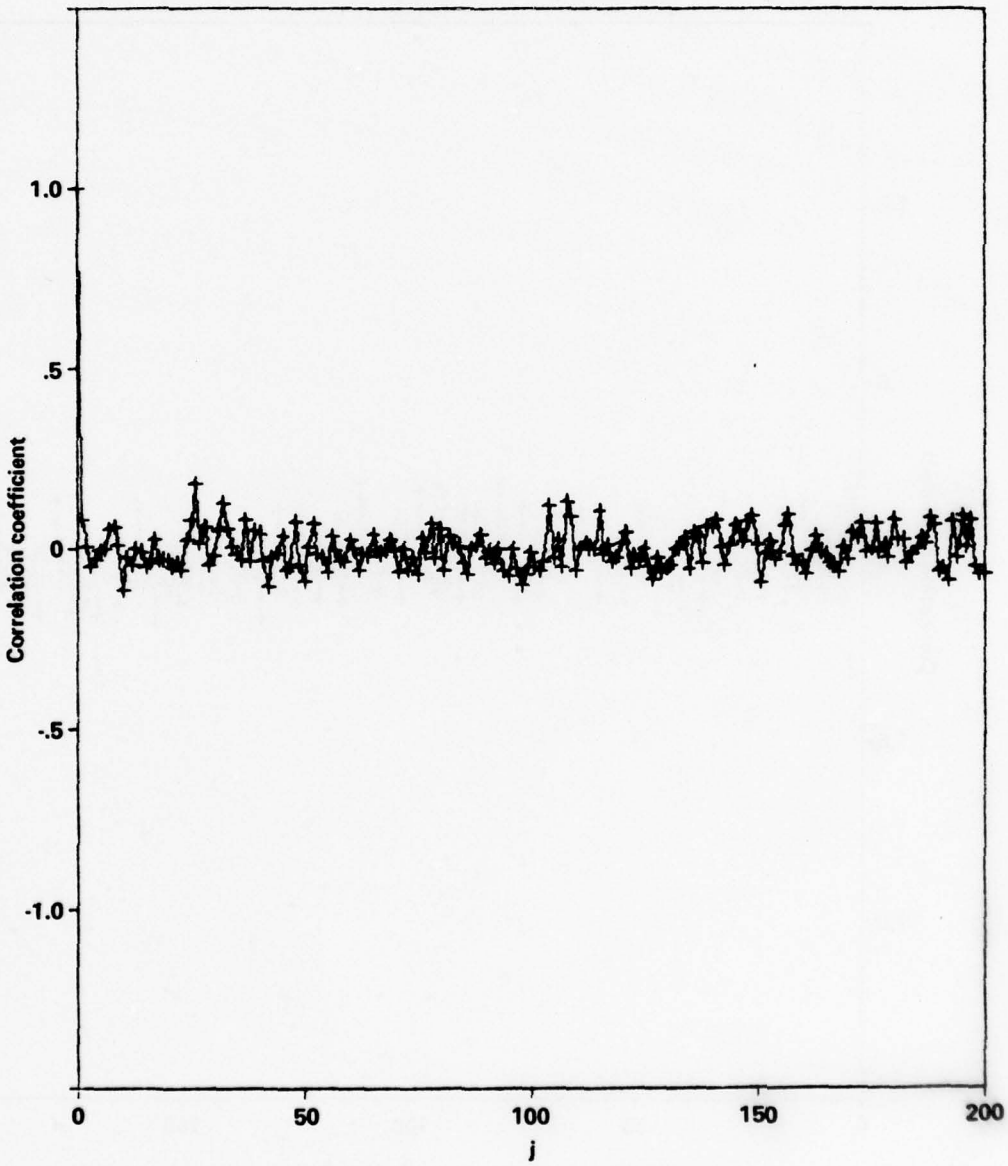


Figure 5.8(e). Delay correlation (simulation, load = .1).

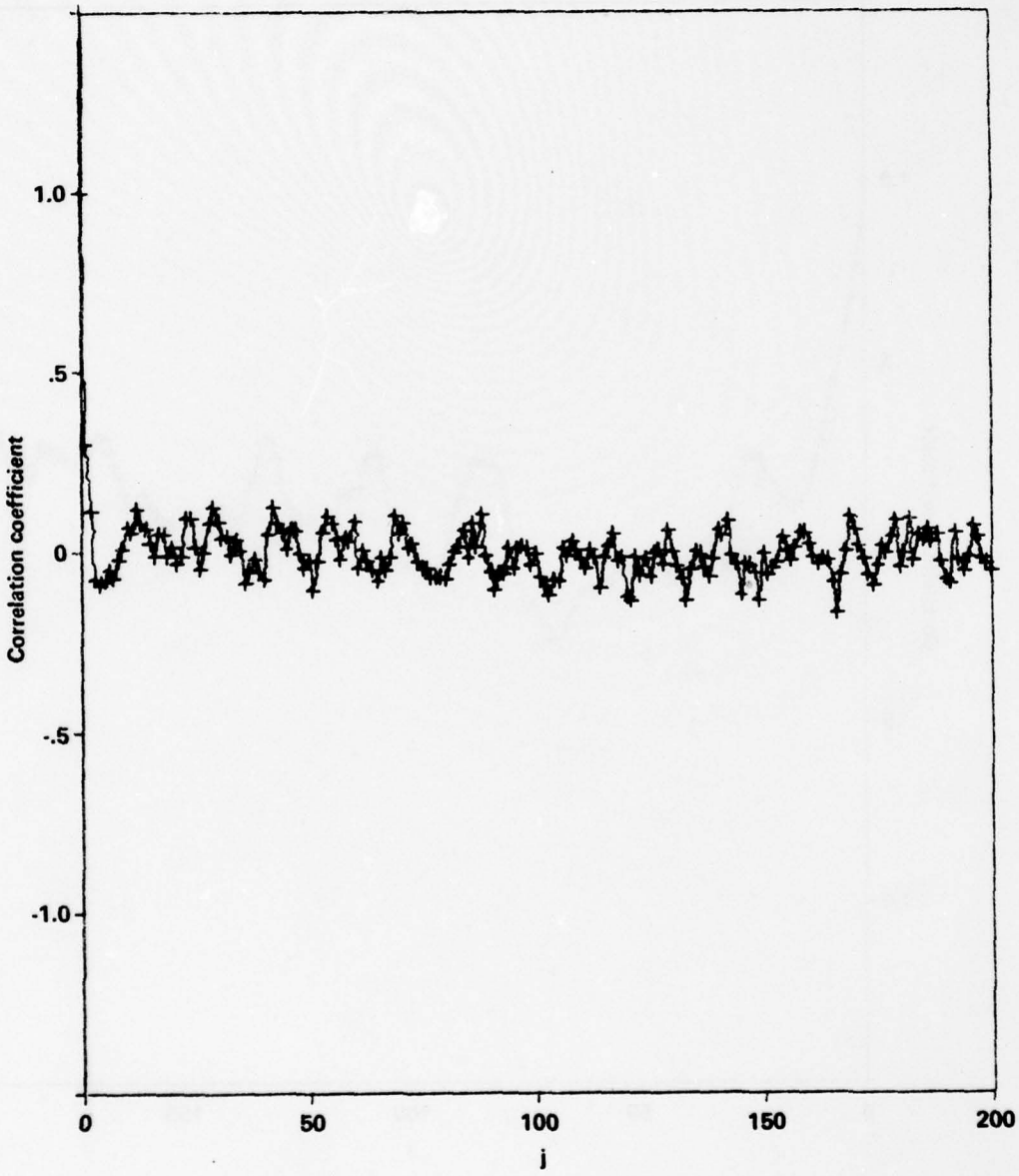


Figure 5.8(f). Delay correlation (simulation, load = .5).

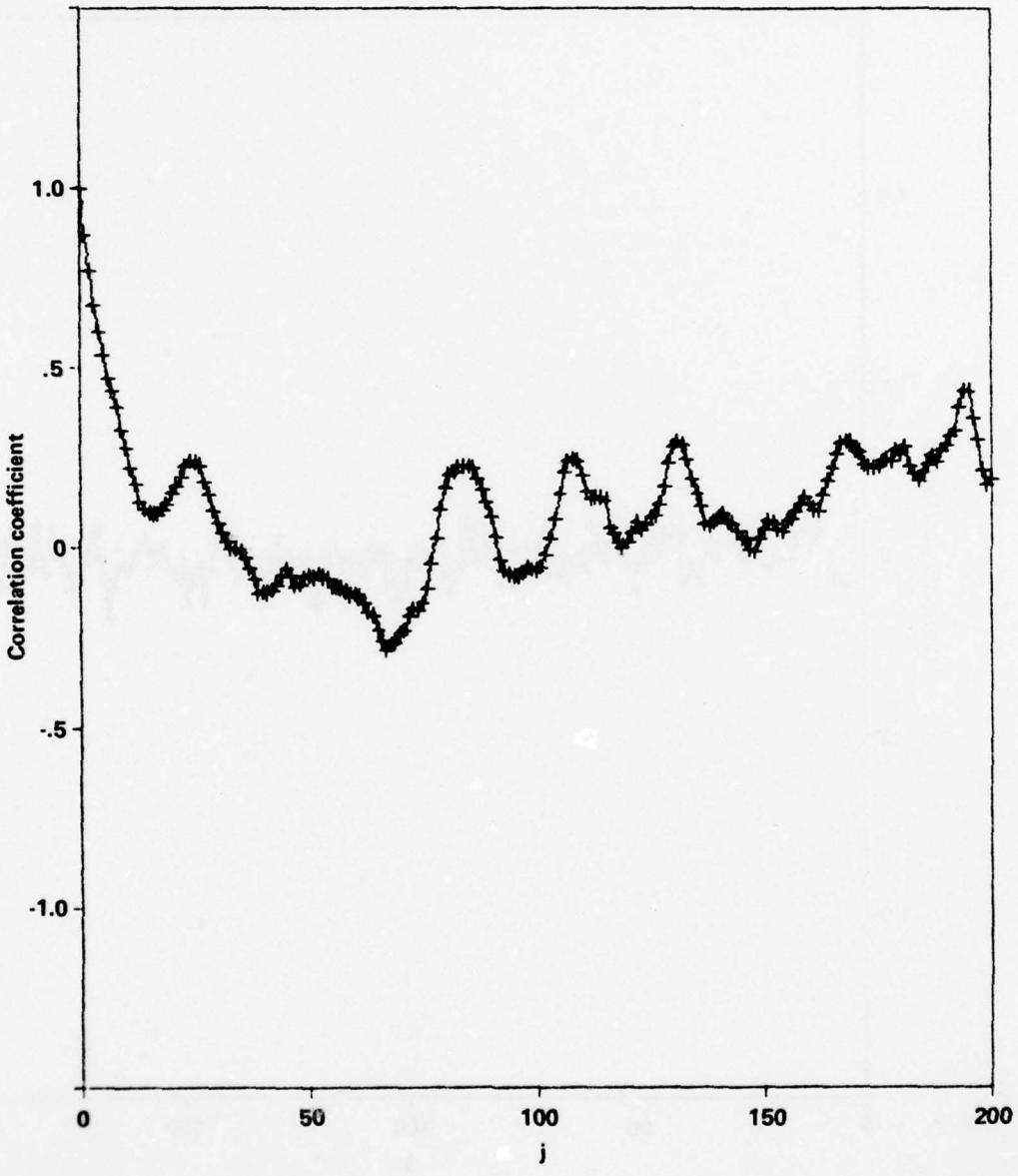


Figure 5.8(g). Delay correlation (simulation, load = .9).

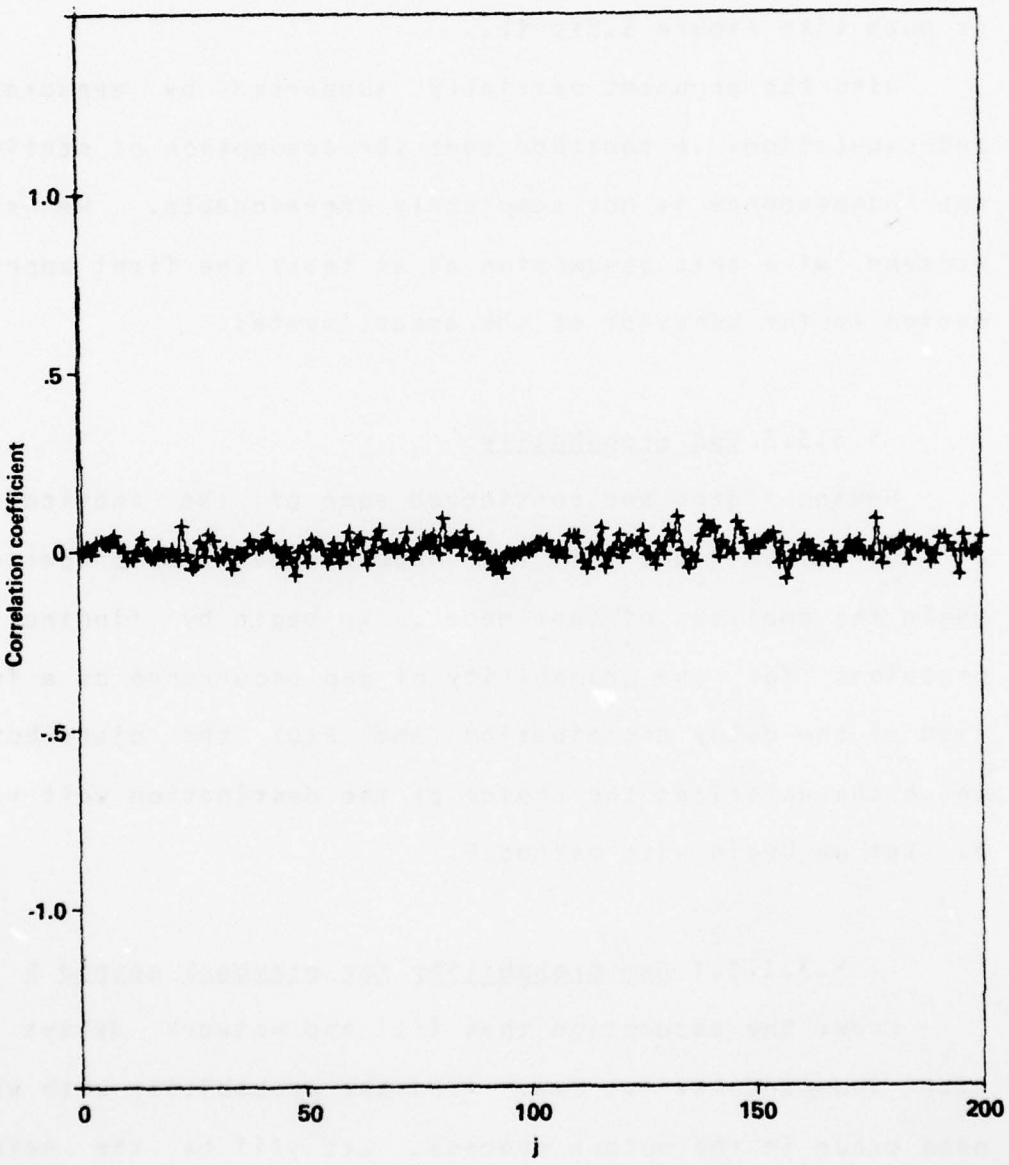


Figure 5.9. Pseudo random sequence correlation.

For comparison to Figure 5.8 we include Figure 5.9 which shows the correlation coefficient computed in the same way for a sequence of pseudo-random numbers. Its appearance is much like Figure 5.8(c-f).

With the argument partially supported by measurement and simulation, we conclude that the assumption of statistical independence is not completely unreasonable. We shall proceed with this assumption as at least the first approximation to the behavior of the actual system.

5.3.3.3 Gap probability

Having stated and considered some of the implications of the assumptions of the model, we are now prepared to begin the analysis of that model. We begin by finding expressions for the probability of gap occurrence as a function of the delay distribution and R(D) the distribution which characterizes the choice of the destination wait value D. Let us begin with method E.

5.3.3.3.1 Gap probability for playback method E

Under the assumption that fill and network delays are each independent, we may find the probability with which gaps occur in the output process. Let y(i) be the network delay and f(i) be the fill time of message i. Let t(i) = y(i) + f(i) - y(i-1) - f(i-1) be the difference in delay experienced by message i and i-1. Then the {t(i)} are mutually independent and identically distributed random

variables since we have assumed that the {y(i)} and {f(i)} are. Let the nth partial sum of {t(i)} be P(n) = P(n-1) + t(n), and let P(0) = 0. Notice that P(n) = y(n) + f(n) - [y(0) +f(0)]. Now let us define the strict ascending ladder indices {N(k)} for {P(n)} as in [Prab 65] (p. 140) as follows:

$$N(0)=0$$

$$N(1) = \min(n : P(n)>0)$$

$$N(k) = \min(n : n>N(k-1) \text{ and } P(n)>P(N(k-1)))$$

To proceed, we need the following theorem.

Theorem. A gap in the output process in method E occurs at message i if and only if i is a ladder index N(k) for some k>0 and P(i)>D, where D is the destination wait time.

Proof: An output gap occurs if and only if a message arrives after it is needed for playout. If message zero begins filling at time zero, then the time at which message i arrives at the destination is

$$f(0) + f(1) + \ldots + f(i) + y(i)$$

(i.e., the time to fill all previous messages plus the time to fill message i plus the network delay of message i).

Suppose that $P(i) \leq D$ for all i. Then for each i we have

$$P(i) = y(i) + f(i) - (y(0) +f(0)) \leq D \text{ for all } i. \quad (5.2)$$

The time that message one is required for playout is

$$f(0) + y(0) + D + f(0)$$

(i.e., the time of arrival of message zero plus the destination wait time plus the time to playout message zero). From equation 5.2 we have

$$y(1) + f(1) + f(0) \leq y(0) + f(0) + D + f(0).$$

Therefore message one is not late and does not cause a gap. Equation 5.2 may be rewritten as

$$f(i) + y(i) \leq y(0) + f(0) + D.$$

By adding $f(0) + f(1) + \ldots + f(i-1)$ to both sides we have

$$\sum_{j=0}^{i-1} f(j) + f(i) + y(i) \leq \sum_{j=0}^{i-1} f(j) + y(0) + f(0) + D.$$

This states that message i arrives at or before its required time. Hence no gaps will occur when $P(i) \leq D$ for all i.

Suppose, on the other hand, that there exists an i such that $P(i) > D$. Choose k1 to be the minimum such i. Then k1 is a ladder index by definition, since it is the first occurrence of a value of P(i) greater than D. We have

$$f(0) + y(0) + D < f(k1) + y(k1)$$

and therefore

$$f(0) + y(0) + D + \sum_{j=0}^{k1-1} f(j) < \sum_{j=0}^{k1-1} f(j) + f(k1) + y(k1)$$

and hence the first gap occurs at ladder index k1 whose value P(k1) is first to exceed D.

After the occurrence of this gap at k1, message k1 is output immediately upon arrival and therefore message k1+1 will be needed at time

$$\sum_{j=0}^{k1} f(j) + y(k1) + f(k1)$$

The next gap, if any, will occur at message k2 where k2 is the minimum i for which

$$\sum_{j=0}^{i} f(j) + y(i) > \sum_{j=0}^{k1} f(j) + y(k1) + \sum_{j=k1}^{i-1} f(j)$$

(i.e., message i arrives after it is needed for playout). The following three statements are each equivalent to the last.

$$f(i) + y(i) > f(k1) + y(k1)$$

$$f(i) + y(i) - f(0) - y(0) > f(k1) + y(k1) - y(0) - f(0)$$

$$P(i) > P(k1)$$

Therefore, since k2 was chosen as the minimum such i, k2 is the ladder index immediately following k1.

We may now prove the general case by induction. Assume that the first n-1 gaps occurred at the first n-1 ladder indices whose values are greater than D. Let l be the last ladder index at which a gap occurred. After the occurrence of this gap at l, message l is output immediately upon arrival (as was message k1) and therefore message l+1 will be needed at time

$$\sum_{j=0}^{l} f(j) + y(l) + f(l)$$

The next gap, if any, will occur at message l1 where l1 is the minimum i for which

$$\sum_{j=0}^{i} f(j) + y(i) > \sum_{j=0}^{l} f(j) + y(l) + \sum_{j=l}^{i-1} f(j)$$

(i.e., message i arrives after it is needed for playout). Once again, the following three statements are each equivalent to the last.

$$f(i) + y(i) > f(l) + y(l)$$

$$f(i) + y(i) - f(0) - y(0) > f(l) + y(l) - y(0) - f(0)$$

$$P(i) > P(l)$$

Hence l1 is the ladder index immediately following l. Therefore each ladder index whose corresponding value is greater than D has an associated gap; and a gap occurs only at a ladder index whose corresponding value is greater than D.

Q.E.D.

Thus the probability that the kth message in a sentence produces a gap is equal to the probability that k is a ladder index and P(k) > D, i.e.,

$$Pr[gap \mid k,D] = Pr[k \text{ is a ladder index and } P(k)>D]$$

According to [Prab 65] (p. 141) we have

$$Pr[gap \mid k,D] = Pr[P(k)>0,\ P(k-1)>0,\ldots,\ P(1)>0,\ \text{and } P(k)>D]$$

Since P(k)>D implies P(k)>0 and P(1) is guaranteed to be greater than zero, we have

$$Pr[gap \mid k,D] = Pr[P(k)>D,\ P(k-1)>0,\ldots,\ P(2)>0]$$

Assume that the (fill plus network) delay is distributed with probability distribution function S(y), that the delay

of the first message in the sentence is t, and that D is the destination wait time. Then

$$\Pr[\text{gap} \mid k,t,D] = [1-S(t+D)][1-S(t)]^{k-2}$$

We may remove the condition on k to obtain the probability that a gap occurs at a randomly selected message in a sentence of length n, given t and D, i.e.,

$$\Pr[\text{gap} \mid t,D,n] = \frac{1}{n}\sum_{k=2}^{n}[1-S(t+D)][1-S(t)]^{k-2}$$

Removing the condition on t we have

$$\Pr[\text{gap} \mid D,n] = \frac{1}{n}\sum_{k=0}^{n-2}\int_{t=0}^{\infty}[1-S(t+D)][1-S(t)]^{k}\,dS(t)$$

Removing the condition on D we have

$$\Pr[\text{gap}/n] = \frac{1}{n}\sum_{k=0}^{n-2}\int_{D=0}^{\infty}\int_{t=0}^{\infty}[1-S(t+D)][1-S(t)]^{k}\,dS(t)\,dR(D) \qquad (5.3)$$

where R(D) is the distribution function of the destination wait time D. Since

$$\sum_{i=0}^{n-1} x^{i} = \frac{1-x^{n}}{1-x}$$

we have

$$\Pr[\text{gap}/n] = \frac{1}{n}\int_{D=0}^{\infty}\int_{t=0}^{\infty}[1-S(t+D)]\,\frac{1-[1-S(t)]^{n-1}}{S(t)}\,dS(t)\,dR(D) \qquad (5.4)$$

We shall evaluate this expression numerically in succeeding sections.

5.3.3.3.2 Gap probability for playback method I

Since there is no expansion of the time axis in method I, we expect that when a message exceeds the delay of the first message by more than D a gap results. The destination arrival time of message k is (as before) given by

$$\sum_{j=0}^{k} f(j) + y(k)$$

The time at which message k is required is

$$f(0) + y(0) + D + \sum_{j=0}^{k-1} f(j)$$

Hence we have a gap whenever

$$\sum_{j=0}^{k} f(j) + y(k) > f(0) + y(0) + D + \sum_{j=0}^{k-1} f(j)$$

or

$$f(k) + y(k) > f(0) + y(0) + D.$$

Therefore the probability of a gap at message k, given that f(0) + y(0) = t and D is the destination wait time, is 1-S(t+D). For sentences of length n messages we have

$$\Pr[\text{gap} \mid t, D, n] = \frac{1}{n} \sum_{k=1}^{n-1} [1-S(t+D)] = \frac{n-1}{n} [1-S(t+D)]$$

Removing the condition on t we have

$$Pr[gap \mid D,n] = \frac{n-1}{n} \int_{t=0}^{\infty} [1-S(t+D)]\, dS(t)$$

Removing the condition on D we have

$$Pr[gap \mid n] = \frac{n-1}{n} \int_{D=0}^{\infty} \int_{t=0}^{\infty} [1-S(t+D)]\, dS(t)dP(D) \quad (5.5)$$

Comparing the right hand side of this to that of equation 5.3 we notice, as expected, that this expression is larger. Therefore, in general, the gap probability is lower in method E than in method I. Numerical results appear below.

5.3.3.4 Delay predictors

In the previous two sections we have assumed a distribution for the destination wait time D. The purpose of this section is to explore some possible delay predictors each yielding a distribution function for D. Three schemes are discussed and all are based on monitoring which may be performed at the destination node. The first scheme predicts delay variation by computing the range of delay for previous samples. We have found that this scheme "learns" quickly but performs poorly when occasional long delays occur. In order to ignore these "spikes" in delay, and thus achieve smaller D, we introduce the second scheme which views only that portion of the range below a threshold. The third scheme attempts to track the changing delay by modifying an internal counter to predict message arrival times.

The basic idea then, in each scheme, is to measure the

delay, and based on the information gained, to make some intelligent choice for D at each sentence boundary. It is clear that the larger is D the fewer are the gaps which will occur in the output stream. However, large D tends to destroy the interactive nature of the communication. Ideally a scheme would "optimally" balance these two properties. It is difficult to define optimality here. However, the following general statement holds: Large destination wait time and/or frequent gap occurrence each yield a poor quality of communication.

5.3.3.4.1 m-sample range

In both playback methods, if the delay of a message within a sentence exceeds the delay of the first message of that sentence by more than D, then at least one gap will occur. The total range of the previous m (m>1) samples is a pessimistic estimate for the difference between maximum delay within a sentence and the delay of the first message in that sentence. It is pessimistic because, on the average, the first delay will fall somewhere between the extremes. The problem is to find the distribution of the range of m samples. Our derivation below follows that of [Gumb 67] (pp. 97-98). We begin with the joint probability that t is the minimum (denoted t=min) and t+z is the maximum (denoted t+z=max) among m samples each drawn independently from distribution S(y).

$$\Pr[t=\min,\ t+z=\max \mid m \text{ samples}]$$

$$= \Pr[\text{at least one sample} = t,$$
$$\text{at least one sample} = t+z, \text{ and}$$
$$m-2 \text{ samples fall in the interval } [t,t+z]\ ]$$

$$= m\, dS(t)\ \ (m-1)\, dS(t+z)\ \ [S(t+z)-S(t)]^{m-2}$$

By removing the conditions on t and t+z we find that the distribution of range $R(D \mid m) = \Pr[\text{range} \leq D \mid m \text{ samples}]$ is given by

$$R(D \mid m) = \int_{t=0}^{\infty} \int_{z=0}^{D} m(m-1)[S(t+z)-S(t)]^{m-2}\, dS(t+z)\, dS(t)$$

Yielding

$$R(D \mid m) = m \int_{t=0}^{\infty} [S(t+D)-S(t)]^{m-1}\, dS(t) \qquad (5.6)$$

This equation cannot be reduced further without knowledge of S(t).

5.3.2.4.2 m-sample partial range

The full range estimate functions fairly well (as we show later) but it is not without fault. Since the entire range is used, isolated cases of high delay cause an unduly large value to be chosen for the next destination wait time. For this reason we wish to examine the partial range of m samples where the k highest values are ignored ($m>1$ and $k<m-1$). We wish to find, as before, the distribution R(D |

m,k) of the partial range given the distribution S(y) from which the m samples are chosen.

We begin, as before, with the joint probability that t and t+z are among the m samples drawn from S(y), where t is the minimum, and t+z is the maximum among the m-k smallest samples.

Pr[drawing such a sample]

= Pr[at least one sample = t,
at least one sample = t+z,
m-k-2 samples are in [t,t+z], and there are
k samples which exceed t+z]

$$= m\,dS(t)\,(m-1)\,dS(t+z)\,[S(t+z)-S(t)]^{m-k-2}\binom{m-2}{k}[1-S(t+z)]^{k}$$

Therefore

$$R(D \mid m,k) = m(m-1)\binom{m-2}{k} \int_{t=0}^{\infty}\int_{z=0}^{D} [S(t+z)-S(t)]^{m-k-2}\,[1-S(t+z)]^{k}\,dS(t+z)\,dS(t) \qquad (5.7)$$

5.3.3.4.3 A delay tracker

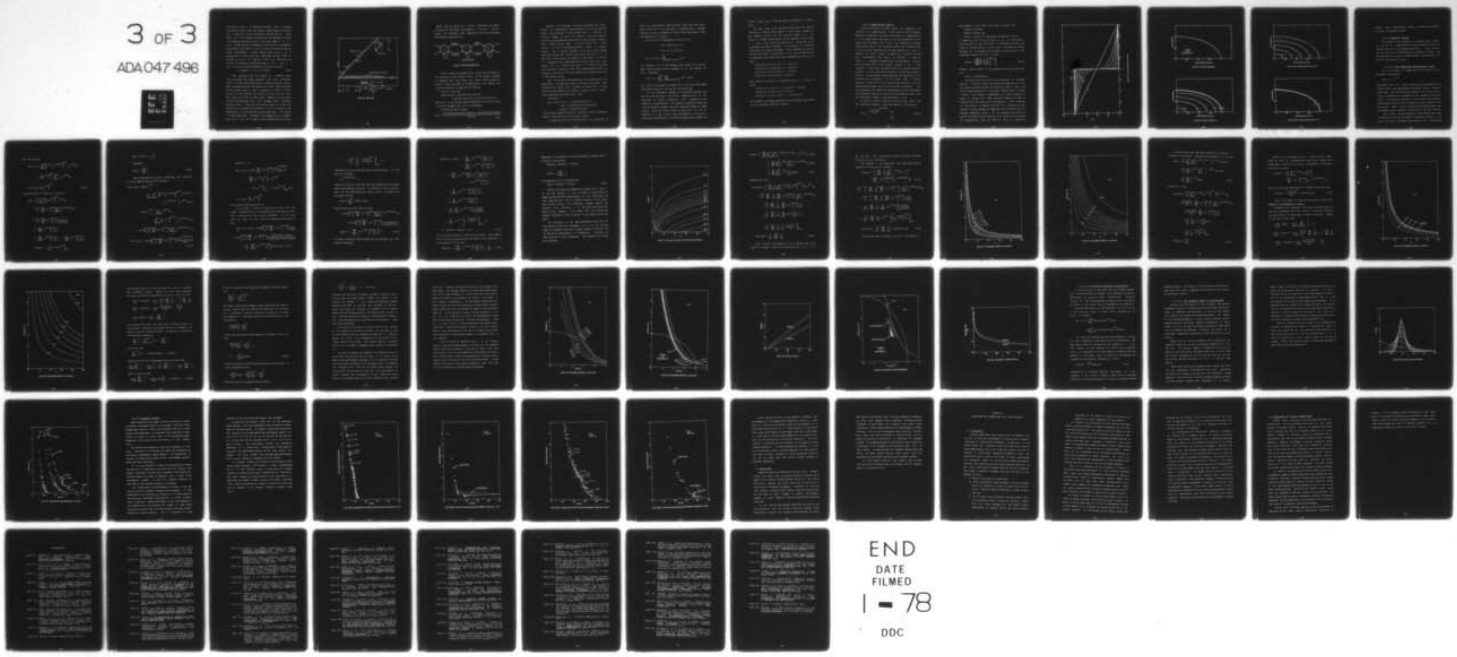
A third method of delay prediction was developed by James Forgie [Forg 76a] of the Massachusetts Institute of Technology Lincoln Laboratory, Lexington, Massachusetts. It compares the time stamp of an arriving message with the "arrival clock" and adjusts that clock by plus (or minus)

one count if the message arrived earlier (or later) than predicted by the clock value. At the beginning of a sentence a quantity called the "reconstitution delay" c is subtracted from the arrival clock to set the playout clock. It is important to note that the value of c is chosen at the beginning of a conversation by the user. The waiting packet begins playout when the playout clock reaches the value of the time stamp within the packet. Time is measured in units equal to a frame size (19.2 msec for Linear Predictive Coding, LPC, algorithm used experimentally for speech communication in the ARPANET, see [Cohe 76]) and the clock is increased by one at each frame time.

Figure 5.10 illustrates the operation of the algorithm. The first message of the sample, over which the arrival clock a(x) is adjusted, arrives at the destination and initiates the clock value (assumed to be at time zero). The time stamp in each message is shown as the ordinate of a dot at the point (x,y), where x is the arrival time of the message at the destination and y is the time stamp within that message. Notice that an "early" arrival has its time stamp above the arrival clock and a "late" arrival appears below. The arrival clock therefore attempts to "track" the points (x,y) by adding one at points where $y > a(x)$ and subtracting one where $y < a(x)$. Were the clock unadjusted, the value would be floor(x) (the largest integer less than or equal to x). Let the clock difference l(x) be defined to be a(x) - floor(x). Then a(x) predicts the time stamp of a message

AD-A047 496 CALIFORNIA UNIV LOS ANGELES DEPT OF COMPUTER SCIENCE F/G 17/2
ADVANCED TELEPROCESSING SYSTEMS.(U)
JUN 77 L KLEINROCK, G ESTRIN, R MUNTZ DAHC15-73-C-0368
UNCLASSIFIED NL

3 OF 3
ADA047496



END
DATE FILMED
1-78
DDC

arriving at time x. It therefore predicts that a message arriving at time x with time stamp y should (have) arrive(d) at x - (a(x) -y); or more precisely floor(x-a(x)-y). Since a(x) and y are integers we have that the predicted arrival of a message arriving at time x with time stamp y is y - l(x). At the arrival of message m (the start of a sentence) the playout clock p(xm) is set to a(xm)-c, and then steps at frame boundaries. Playout begins when p(x) $\geq$ ym or equivalently r units after the predicted arrival time of message m (i.e., at time ym - l(xm) + r). Hence for the value D as previously defined we have D = ym - l(xm) + c -xm. We shall disallow negative D and thus we write

$$D = \max\{0, c + ym - xm - l(xm)\}. \qquad (5.8)$$

This algorithm may be modeled by a Markov chain [Klei 75] where the states represent the clock difference value l(x) and transitions occur at message arrival points. When a message arrives with delay less (more) than predicted by the current state k, then a transition to state k+1 (k-1) occurs. As the function l(x) starts at value zero the Markov chain will start in state zero. Suppose t is the delay of the first message (i.e., that message which initiates arrival clock). Given that the current state is k, the probability of a transition from state k to k+1 is equal to the probability of experiencing a delay less than t + kf, where f is the frame size. Likewise, the probability of a transition from k to k-1 is equal to the probability of a delay

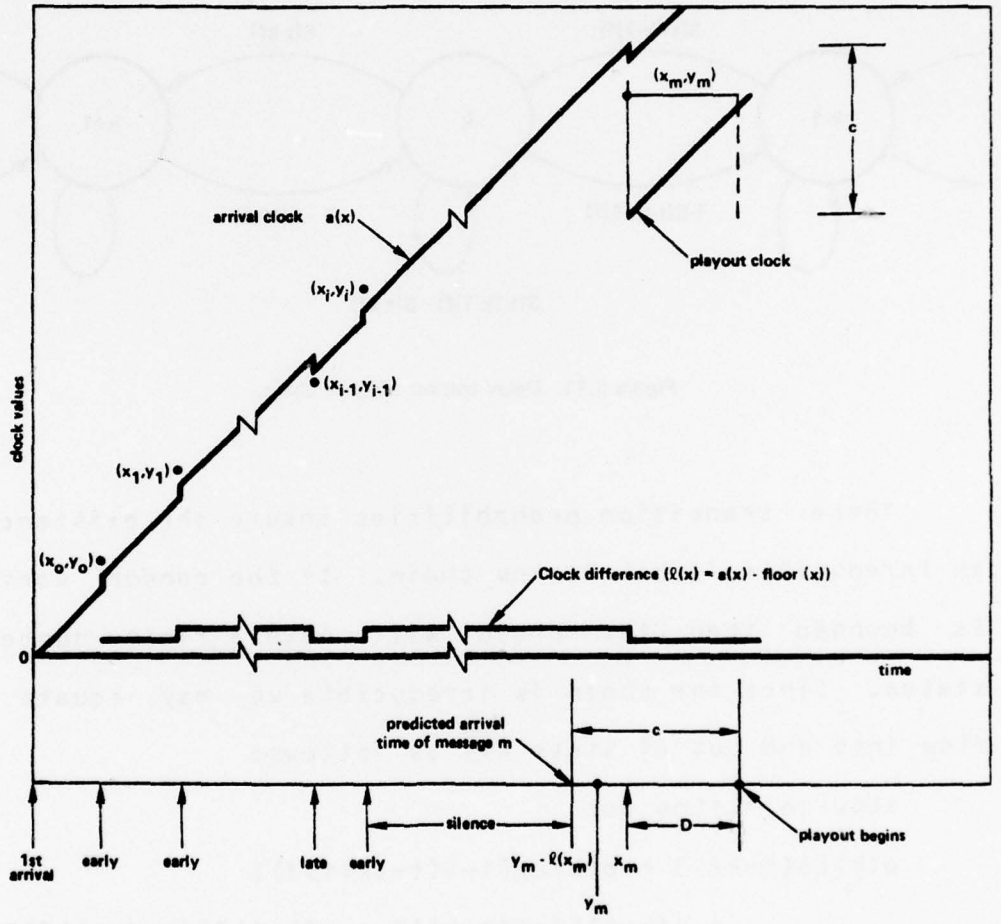


Figure 5.10. Delay tracker.

reater than or equal to t - (k-1)f. Otherwise, we remain in state k if the delay lies between t - kf and t - (k-1)f. Figure 5.11 displays this graphically for the continuous distribution function S(t).

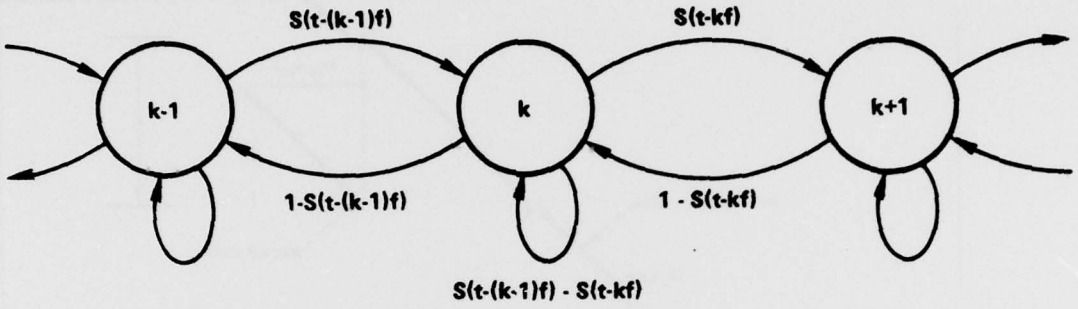


Figure 5.11. Delay tracker Markov chain.

These transition probabilities insure the existence of an irreducible (sub-) Markov chain. If the random variable is bounded then the chain will have a finite number of states. Since the chain is irreducible we may equate the flow into and out of state k+1 as follows:

flow in = flow out

$$p(k)[S(t-kf)] + p(k+2)[1-S(t-(k+1)f)] = p(k+1)[1-S(t-kf)] + p(k+1)[S(t-(k+1)f)]$$

where p(k) is the steady-state probability of being in state k (we have assumed that S(t) is continuous).

Solving for p(k) we have

$$p(k) = \frac{p(k+1)[1-S(t-kf)+S(t-(k+1)f)] - p(k+2)[1-S(t-(k+1)f)]}{S(t-kf)} \qquad (5.9)$$

Equation (5.9) defines a recursive procedure for calculating the steady-state state probabilities for the Markov chain. Unfortunately, in the current form it requires an infinite number of steps in general. One can, however, achieve an approximation using the following general properties. There exists some l for which $S(t-(l+1)f) = 0$ and $S(t-lf) > 0$ (i.e., the random variable, delay, is bounded below). For a random variable which has a finite lower bound and finite mean value we have that; for any $e > 0$ we may find a finite positive integer l' such that $1 - S(t+l'f) < e$. Therefore the steady-state of being in any state numbered $-l'-1$ or less is less than e. If $L=l+l'+1$, we need only perform L steps (plus the usual normalization step) to achieve a value for the state probabilities each within e/L of the correct value.

Thus far we have considered only the steady-state probabilities. One may also wish to consider the m-step state probabilities given that we begin in state zero. Let $p(k \mid m,0)$ be the probability of being in state k after m transitions starting from state zero. Then

$$p(0 \mid 0,0) = 1$$

$$p(k \mid l,0) = p(k-1 \mid l-1,0)[S(t-(k-1)f)]$$
$$+ p(k \mid l-1,0)[S(t-(k-1)f)-S(t-kf)]$$
$$+ p(k+1 \mid l-1,0)[1-S(t-kf)]$$

This defines another recursive procedure which terminates after m iterations for at most 2m+1 states.

By computing the state probabilities as indicated we

find the distribution function R(D). Note that the probability that l(x) = k is just p(k). Assuming that a message arrives to start a sentence at time x with time stamp y then from equation 5.8 we have

$$R(D \mid x,y,c) = Pr[\max\{D, c+y-x-l(x)\} \leq D\ ]$$

$$= Pr[\ c+y-x-l(x) \leq D\ ]$$

$$= Pr[\ l(x) \geq c+y-x-D\ ]$$

$$R(D \mid x,y,c) = \sum_{i=\mathrm{floor}(c+y-x-D)}^{l+1} p(i)$$

The quantity x-y is the message delay (both fill and network) which we have earlier assumed to be distributed as S(t). Therefore

$$R(D \mid c) = \int_{t=0}^{\infty} \sum_{i=\mathrm{floor}(c-t-D)}^{l+1} p(i)\ \ dS(t)$$

We shall not evaluate this numerically, but later show some simulation results in the use of this algorithm.

We have assumed a constant c in the analysis thus far and this was the case in the actual implementation of the algorithm at Lincoln Laboratory (i.e., constant for the length of a conversation). An automatic method of adjusting c to the network dynamics has been developed (again by James Forgie). After an initial value is given to c, it is increased if a gap occurs and decreased if a string of s (currently contemplated to be 100) messages pass through the

system without gaps. This may again be modeled by a Markov chain.

The new chain will be three dimensional with states indexed by k the old state labels, i the current length of the gapless string, and the current value of c. Let G(k,i,c) = Pr[gap occurs given l(x)=k, gapless string length = i, and c = the reconstitution delay]. Transitions occur at message arrivals as before. It is simpler to visualize transitions as taking two steps. First consider the part of a transition which is governed by the gap probabilities. The possible transitions and their probabilities are as follows

$$Pr[(k,i,0) \text{ to } (k,i+1,0)] = 1 - G(k,i,0)$$

$$Pr[(k,s,c) \text{ to } (k,0,c-1)] = 1 - G(k,s,c)$$

$$Pr[(k,i,c) \text{ to } (k,i+1,c)] = 1 - G(k,i,c)$$

$$Pr[(k,i,c) \text{ to } (k,0,c+1)] = G(k,i,c)$$

Next consider the part of a transition due to a change in delay.

$$Pr[(k,i,c) \text{ to } (k,i,c)] = S(t-(k-1)f) - S(t-kf)$$

$$Pr[(k,i,c) \text{ to } (k+1,i,c)] = S(t-kf)$$

$$Pr[(k,i,c) \text{ to } (k-1,i,c)] = 1 - S(t-(k-1)f)$$

We attempt no further solution to this problem, but rather, we later present simulation results.

5.3.3.5 Communication quality

Before proceeding, we need a basis for comparing the "quality" of communication in our stream traffic context. Among the parameters affecting the quality of communication in the system under discussion are G, the probability of an output gap occurrence and f+t+D, the speaker-to-listener (end-to-end) delay. The end-to-end delay and the gap probability are both influenced by the destination wait time D. Ideally one would choose to operate at the point D=0 and G=0. Unfortunately this cannot be achieved in general. There is a tradeoff between these two parameters. Experience has shown that we can tolerate nonzero G up to some threshold value [Forg 76] beyond which the quality degrades rapidly. For the experiment reported in [Forg 76] roughly 6 percent of the packets caused gaps and the quality was considered to be acceptable. It is likely that a similar behavior is true for D (i.e., beyond some value of f+t+D the communication ceases to be of an interactive nature). Studies regarding the effect of transmission delay on telephone conversations reported in [Krau 67] and [Klem 67] show that the threshold is above .6 sec and below 1.8 sec.

We wish to choose a quality function Q(D,G) which has this threshold behavior. One such family of functions is

$$q(y) = \begin{cases} \dfrac{1 + (1-y)^{1/(2n-1)}}{2} & y \leq 2 \\ 0 & y \geq 2 \end{cases} \qquad (5.10)$$

Some members of this family are shown in Figure 5.12.

Suppose we define

$Q(D,0) = q(D/d)$ and

$Q(0,G) = q(G/g)$ for some given thresholds d and g.

It remains to fill in the DG plane so as to continuously coincide with the axis functions. As mentioned earlier, the point of highest quality is the origin. Then quality should decrease as we increase the distance from the origin. One reasonable choice for $Q(D,G)$ is then

$$Q(D,G) = q\left(\left[\left(\frac{D}{d}\right)^2 + \left(\frac{G}{g}\right)^2\right]^{1/2}\right) \qquad (5.11)$$

Another choice would be the product of the axis qualities i.e.,

$Q(D,G) = Q(D,0)Q(0,G)$.

Any quality function will define contours of constant quality. We shall define the region of acceptability to be the region enclosed between the positive D and G axes and the DG plane projection of the contour whose value is $Q(d,0) = Q(0,g)$. This region is independent of n and an example is shown in Figure 5.13. Other contours, however increase in distance from the origin with increasing n. This property is illustrated in Figures 5.14 and 5.15. Therefore by adjusting n we may change the importance of the origin as the optimal operating point. If, for example, one wishes to assign the same value of quality to all points in the region of acceptability then an infinite value of n should be

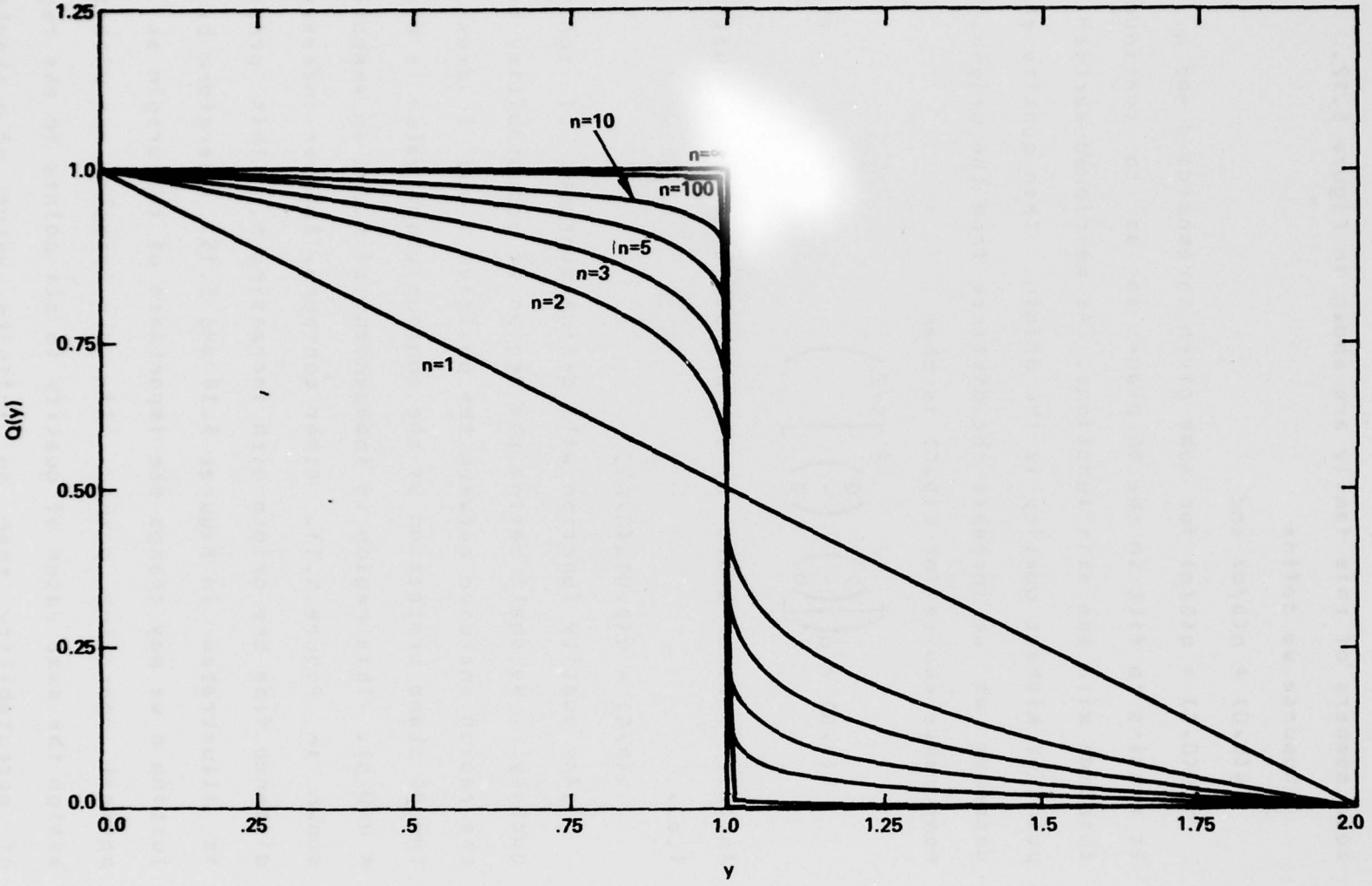


Figure 5.12. One parameter quality functions.

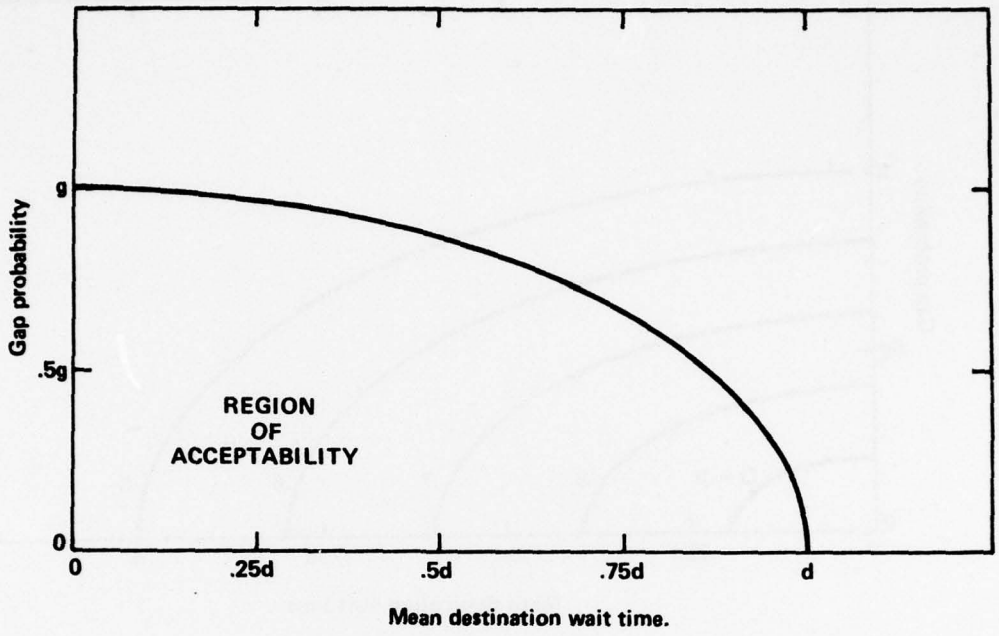


Figure 5.13. Region of acceptability.

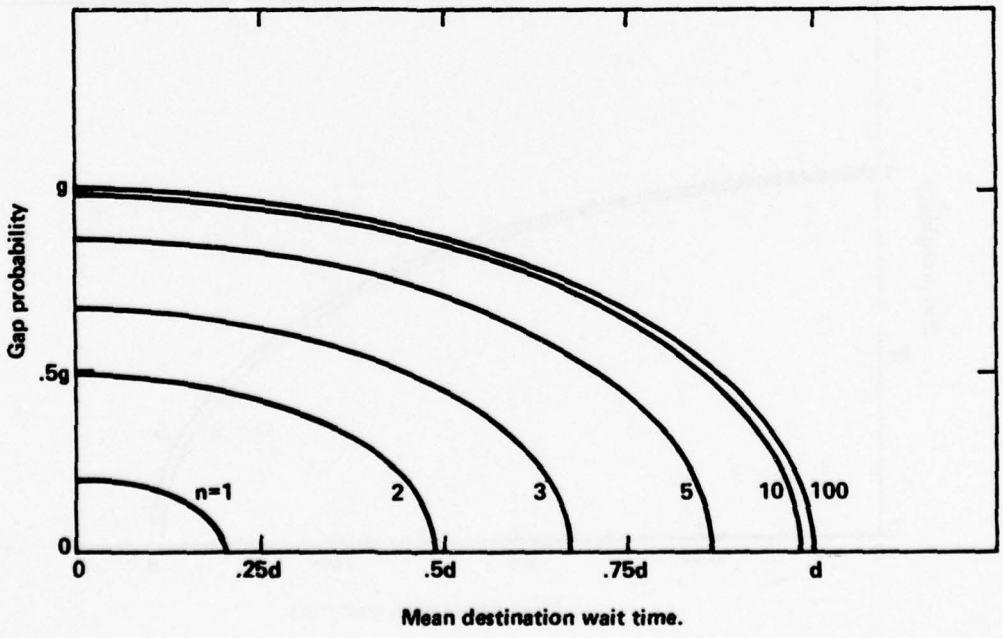


Figure 5.14. Contours of quality .9.

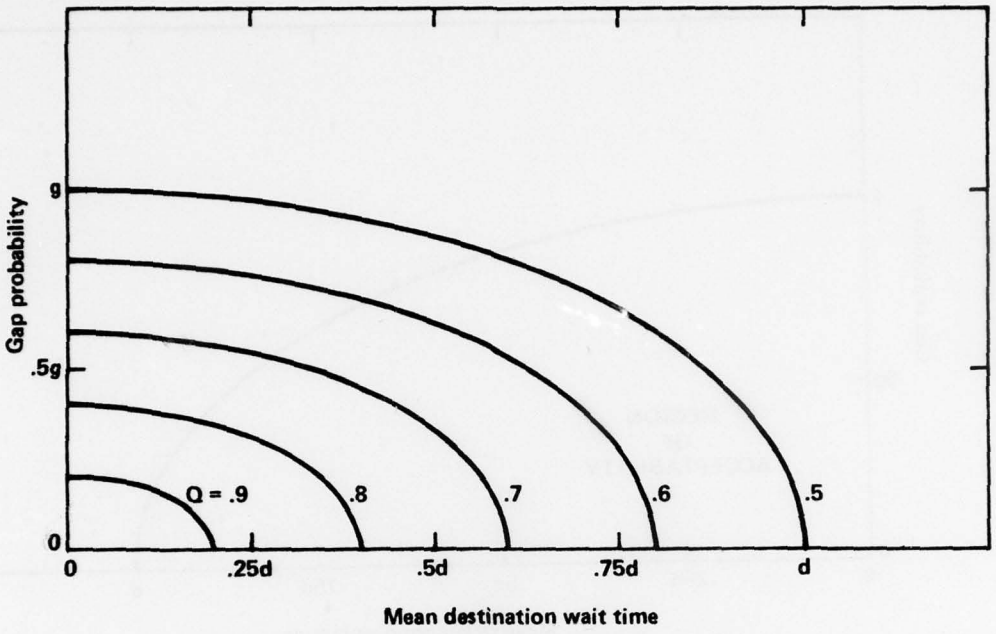


Figure 5.15(a). Constant quality contours (n=1).

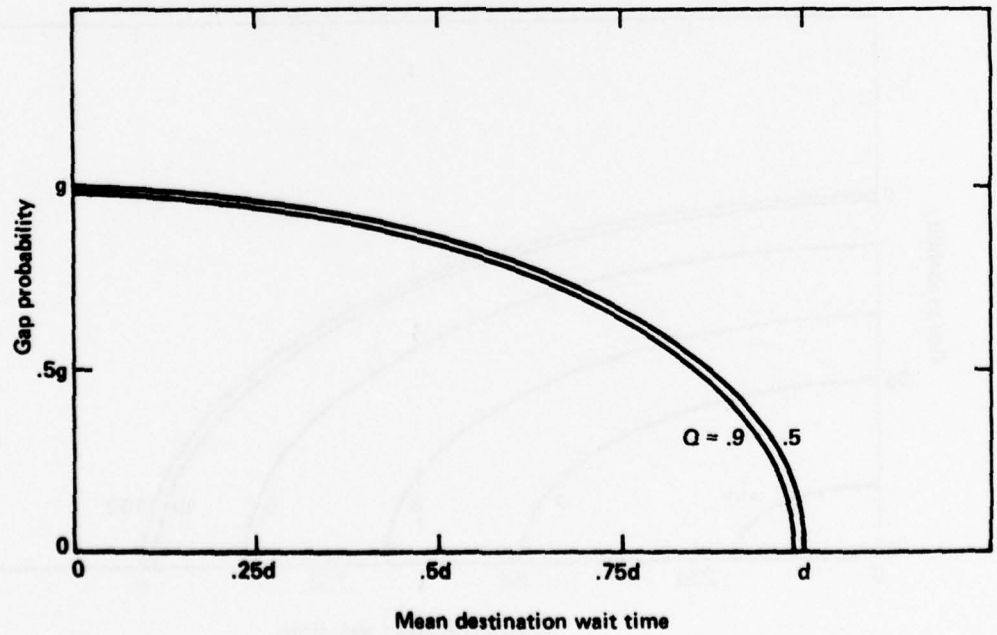


Figure 5.15(b). Constant quality contours (n = 10).

chosen. With communication quality defined by equations 5.10 and 5.11 we may proceed.

5.3.3.6 Numerical results

The purpose of this section is to present some numerical results by substituting a delay distribution into the formulas derived in the preceding sections. Exact results will be given for an exponential distribution of delay. For the more general class of r-stage Erlang distributions numerical integration is used.

5.3.3.6.1 The exponential distribution of delay

For the purpose of demonstration, we choose the exponential distribution

$$S(t) = 1 - e^{-t/x} .$$

Admittedly this is a poor choice for the distribution of network delay! The measurements presented earlier substantiate that. This distribution, however, has a large coefficient of variation when compared to the measured distributions. It therefore provides a lower bound on performance, which should hold for less tractable but more realistic distributions. Some r-stage Erlang distributions (more what one expects from a network) are treated in the next section.

Since $R(D)$, the distribution of destination wait time, is required in the gap probability formulas, we shall first find $R(D \mid m,k)$. We begin with equation 5.6. Substituting

for S(t) we have

$$R(D \mid m) = m\int_{t=0}^{\infty}\left[e^{-t/x} - e^{-(t+D)/x}\right]^{m-1} \frac{1}{x} e^{-t/x}\, dt$$

$$= m\left[1-e^{-D/x}\right]^{m-1} \int_{t=0}^{\infty} \frac{1}{x} e^{-mt/x}\, dt$$

$$R(D \mid m) = \left[1-e^{-D/x}\right]^{m-1} \qquad (5.12)$$

The mean value of D given m is given by

$$\underline{D}(m) = \frac{m-1}{x}\int_{D=0}^{\infty} D\left[1-e^{-D/x}\right]^{m-2} e^{-D/x}\, dD$$

$$= \frac{m-1}{x} \sum_{i=0}^{m-2} (-1)^i \binom{m-2}{i} \int_{D=0}^{\infty} De^{-(i+1)D/x}\, dD$$

$$= \frac{m-1}{x} \sum_{i=0}^{m-2} (-1)^i \binom{m-2}{i} \frac{x^2}{(i+1)(i+1)}$$

$$= x \sum_{i=1}^{m-1} (-1)^{i-1} \binom{m-1}{i} \frac{1}{i}$$

$$= x \sum_{i=1}^{m-2} (-1)^{i-1} \binom{m-2}{i} \frac{1}{i} + x \sum_{i=1}^{m-1} (-1)^{i-1} \binom{m-2}{i-1} \frac{1}{i}$$

$$= \underline{D}(m-1) + \frac{x}{m-1}\left.1-(1-z)^{m-1}\right|_{z=1}$$

$$\underline{D}(m) = \underline{D}(m-1) + \frac{x}{m-1}$$

Therefore,

$$\underline{D}(m) = x \sum_{i=1}^{m-1} \frac{1}{i} \tag{5.13}$$

Next we evaluate $R(D \mid m,k)$. Beginning with equation 5.7 and substituting for $S(t)$ we have

$$R(D \mid m,k) = m(m-1) \binom{m-2}{k} \int_{t=0} \int_{z=0}^{D} \left[e^{-t/x} - e^{-(t+z)/x}\right]^{m-k-2} e^{-k(t+z)/x} \left(\frac{1}{x} e^{-(t+z)/x}\right) \left(\frac{1}{x} e^{-t/x}\right) dz\, dt$$

$$= m(m-1) \binom{m-2}{k} \frac{1}{x} \int_{t=0} e^{-mt/x} dt \int_{z=0}^{D} \left[1-e^{-z/x}\right]^{m-k-2} \frac{1}{x} e^{-(k+1)z/x} dz$$

$$= (m-1)\binom{m-2}{k} \sum_{i=0}^{m-k-2} (-1)^i \binom{m-k-2}{i} \int_{z=0}^{D} e^{-(i+k+1)z/x} dz$$

$$R(D \mid m,k) = x(m-1)\binom{m-2}{k} \sum_{i=0}^{m-k-2} (-1)^i \binom{m-k-2}{i} \frac{1-e^{-(i+k+1)D/x}}{i+k+1} \tag{5.14}$$

Suppose k = 0.

$$R(D \mid m,0) = x(m-1) \sum_{i=0}^{m-2} (-1)^i \binom{m-2}{i} \frac{1-e^{-(i+1)D/x}}{i+1}$$

$$= \sum_{i=1}^{m-1} (-1)^{i-1} \binom{m-1}{i} \; 1-e^{-iD/x}$$

$$= \; 1-(1-z)^{m-1} \Big|_{z=1} - \; 1-(1-z)^{m-1} \Big|_{z=e^{-D/x}}$$

$$R(D \mid m,0) = \left[1-e^{-D/x}\right]^{m-1}$$

which agrees with equation 5.12.

It is useful to check to see that R(D | m,k) has the proper characteristics of a probability distribution function. Clearly R(0 | m,k) = 0 (as desired). Let us show that the limit of R(D | m,k) as D approaches infinity is one.

$$\lim_{D} R(D \mid m,k) = (m-1)\binom{m-2}{k} \sum_{i=0}^{m-k-2} (-1)^i \binom{m-k-2}{i} \frac{1}{i+k+1}$$

$$= (m-1)\binom{m-2}{k} \sum_{i=0}^{m-k-2} (-1)^i \binom{m-k-2}{i} \frac{(i+1)(i+2)\ldots(i+k)}{(i+1)(i+2)\ldots(i+k)(i+k+1)}$$

$$= \frac{1}{k!} \sum_{i=k+1}^{m-1} (-1)^{i-k-1} \binom{m-1}{i} (i-1)(i-2)\ldots(i-k)$$

$$= \frac{(-1)^k}{k!} \frac{d^k}{dz^k} \frac{1-(1-z)^{m-1}}{z} \Bigg|_{z=1} = 1$$

Therefore $R(D \mid m,k)$ has the proper limiting value. For any finite D, we have

$$1-e^{-(i+k+1)D/x} < 1$$

Hence $R(D \mid m,k) \leq 1$ and thus has the properties of a probability distribution function. It remains to find an expression for the mean value of D given m and k. We begin with the usual formula.

$$\underline{D}(m,k) = \int_{D=0}^{\infty} D \, dR(D \mid m,k)$$

Substituting for $R(D \mid m,k)$ we have

$$= (m-1)\binom{m-2}{k} \sum_{i=0}^{m-k-2} (-1)^i \binom{m-k-2}{i} \int_{D=0}^{\infty} \frac{D}{x} e^{-(i+k+1)D/x} \, dD$$

$$= (m-1)\binom{m-2}{k} \sum_{i=0}^{m-k-2} (-1)^i \binom{m-k-2}{i} \frac{x}{(i+k+1)(i+k+1)}$$

$$\underline{D}(m,k) = x \sum_{i=k+1}^{m-1} (-1)^{i-k-1} \binom{m-1}{i}\binom{i-1}{k} \frac{1}{i} \qquad (5.15)$$

In order to simplify this further let us consider the following difference

$$\underline{D}(m+1,k) - \underline{D}(m,k) = x \sum_{i=k+1}^{m} (-1)^{i-k-1} \binom{m}{i}\binom{i-1}{k} \frac{1}{i}$$

$$- x \sum_{i=k+1}^{m-1} (-1)^{i-k-1} \binom{m-1}{i}\binom{i-1}{k} \frac{1}{i}$$

$$= x \sum_{i=k+1}^{m-1} (-1)^{i-k-1} \left[\binom{m}{i} - \binom{m-1}{i}\right]\binom{i-1}{k} \frac{1}{i}$$

$$+ (-1)^{i-m-1} \binom{m-1}{k} \frac{1}{m}$$

$$= x \sum_{i=k+1}^{m} (-1)^{i-k-1} \binom{m-1}{i-1}\binom{i-1}{k} \frac{1}{i}$$

$$= \frac{x}{m} \sum_{i=k+1}^{m} (-1)^{i-k-1} \binom{m}{i}\binom{i-1}{k}$$

$$= \frac{x}{mk!} \sum_{i=k+1}^{m} (-1)^{i-k-1} \binom{m}{i} \frac{(i-1)!}{(i-1-k)!}$$

$$= \frac{x}{mk!} (-1)^k \frac{d^k}{dz^k} \frac{1-(1-z)^m}{z} \Bigg|_{z=1} = \frac{x}{m}$$

or $\underline{D}(m+1,k) = \underline{D}(m,k) + x/m$ (5.16)

This partially defines a recursive procedure. What remains is to find the starting point for each k (i.e., $\underline{D}(k+2,k)$). From equation 5.15 we have

$$\underline{D}(k+2,k) = x \sum_{i=k+1}^{k+1} (-1)^{i-k-1} \binom{k+1}{i}\binom{i-1}{k} \frac{1}{i} = \frac{x}{k+1} \quad (5.17)$$

Beginning with equation 5.17 and performing repeated application of 5.16 we have

$$\underline{D}(k+3,k) = \underline{D}(k+2,k) + x/(k+2)$$

$$\vdots$$

$$\underline{D}(m,k) = x \sum_{i=k+1}^{m-1} \frac{1}{i}$$

From equation 5.13 it follows that

$$\underline{D}(m,k) = \underline{D}(m,0) - \underline{D}(k+1,0) \qquad (5.18)$$

Figure 5.16 shows the behavior of $\underline{D}(m,k)$ for m from 2 to 100 and several k. The k=0 curve is logarithmic in shape, having its most rapid ascent at the beginning and then tapering off. This is a useful property in the selection of both m and k. The difference between $\underline{D}(m,0)$ and $\underline{D}(m-1,0)$ (i.e., $\underline{D}=1/(m-1)$) is the curve from which the $\underline{D}(m,k)$ curves emanate. This shows that the difference between successive values of $\underline{D}(m,k)$ is very small beyond say m=20.

Let GE(n,m,k) be the gap probability given that (1) each sentence contains n messages, (2) D is the partial range of m samples with the k largest samples discarded, and (3) the delay distribution is exponential. Initially we shall drop the m and k parameters. Then equation 5.3 yields the following

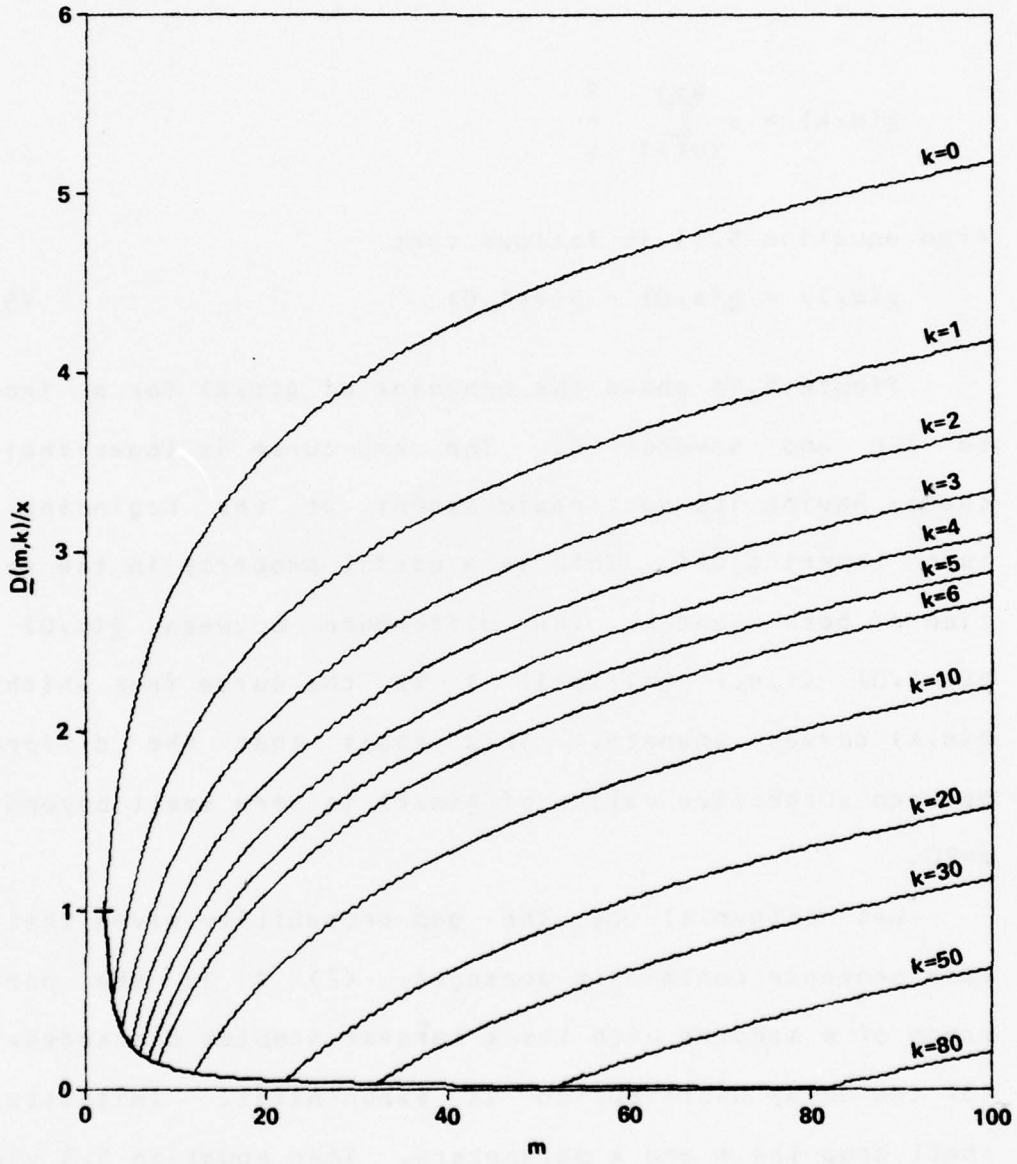


Figure 5.16. Mean range vs. sample size for the exponential distribution.

$$GE(n) = \frac{1}{n} \sum_{i=0}^{n-2} \int_{D=0}^{\infty} \int_{t=0}^{\infty} e^{-t/x} e^{-D/x} e^{-ti/x} \frac{1}{x} e^{-t/x} \, dt \, dR(D)$$

$$= \frac{1}{n} \sum_{i=0}^{n-2} \int_{D=0}^{\infty} e^{-D/x} \int_{t=0}^{\infty} \frac{1}{x} e^{-t(i+2)/x} \, dt \, dR(D)$$

$$= \frac{1}{n} \sum_{i=0}^{n-2} \int_{D=0}^{\infty} e^{-D/x} \frac{1}{i+2} \, dR(D) \qquad (5.19)$$

Suppose k=0. Then

$$GE(n,m,0) = \frac{1}{n} \sum_{i=0}^{n-2} \frac{1}{i+2} \int_{D=0}^{\infty} e^{-D/x} \left[1-e^{-D/x}\right]^{m-2} \frac{m-1}{x} e^{-D/x} \, dD$$

$$= \frac{m-1}{n} \sum_{i=0}^{n-2} \frac{1}{i+2} \sum_{j=0}^{m-2} (-1)^j \binom{m-2}{j} \frac{1}{x} \int_{D=0}^{\infty} e^{-D(j+2)/x} \, dD$$

$$= \frac{m-1}{n} \sum_{i=0}^{n-2} \frac{1}{i+2} \sum_{j=0}^{m-2} (-1)^j \binom{m-2}{j} \frac{1}{j+2}$$

$$= \frac{1}{mn} \sum_{i=0}^{n-2} \frac{1}{i+2} \sum_{j=2}^{m} (-1)^j \binom{m}{j} (j-1)$$

$$= \frac{1}{mn} \sum_{i=0}^{n-2} \frac{1}{i+2} \frac{d}{dz} \frac{(1-z)^m - 1}{z} \Bigg|_{z=1}$$

$$GE(n,m,0) = \frac{1}{mn} \sum_{i=0}^{n-2} \frac{1}{i+2} \qquad (5.20)$$

This function decreases as m or n increase (for n>2). Figure 5.17 shows a family of m versus GE for n = 2, 4, 8,

16, 24, 32. Each curve falls rapidly initially and then flattens out as m increases.

For general k we substitute the right hand side of equation 5.14 into 5.19 to obtain

$$GE(n,m,k) = \frac{1}{n}\sum_{i=0}^{n-2}\frac{1}{i+2}\int_{D=0}^{\infty} e^{-D/x}\binom{m-2}{k}(m-1)\sum_{j=0}^{m-k-2}(-1)^j\binom{m-k-2}{j}\frac{1}{x}e^{-(j+k+1)D/x}\,dD \quad (5.21)$$

$$= \frac{m-1}{n}\binom{m-2}{k}\sum_{i=0}^{n-2}\frac{1}{i+2}\sum_{j=0}^{m-k-2}(-1)^j\binom{m-k-2}{j}\frac{1}{x}\int_{D=0}^{\infty} e^{-(j+k+2)D/x}\,dD$$

$$= \frac{m-1}{n}\binom{m-2}{k}\sum_{i=0}^{n-2}\frac{1}{i+2}\sum_{j=0}^{m-k-2}(-1)^j\binom{m-k-2}{j}\frac{1}{j+k+2}$$

$$= \frac{1}{mnk!}\sum_{i=0}^{n-2}\frac{1}{i+2}\sum_{j=0}^{m}(-1)^{j-k-2}\binom{m}{j}\frac{(j-1)!}{(j-k-2)!}$$

$$= \frac{1}{mnk!}\sum_{i=0}^{n-2}\frac{1}{i+2}(-1)^{k+1}\frac{d^{k+1}}{dz^{k+1}}\left.\frac{1-(1-z)^{m-1}}{z}\right|_{z=1}$$

$$GE(n,m,k) = \frac{k+1}{mn}\sum_{i=0}^{n-2}\frac{1}{i+2} = (k+1)GE(n,m,0) \quad (5.22)$$

Figure 5.18 shows GE versus m for n=16 and several k.

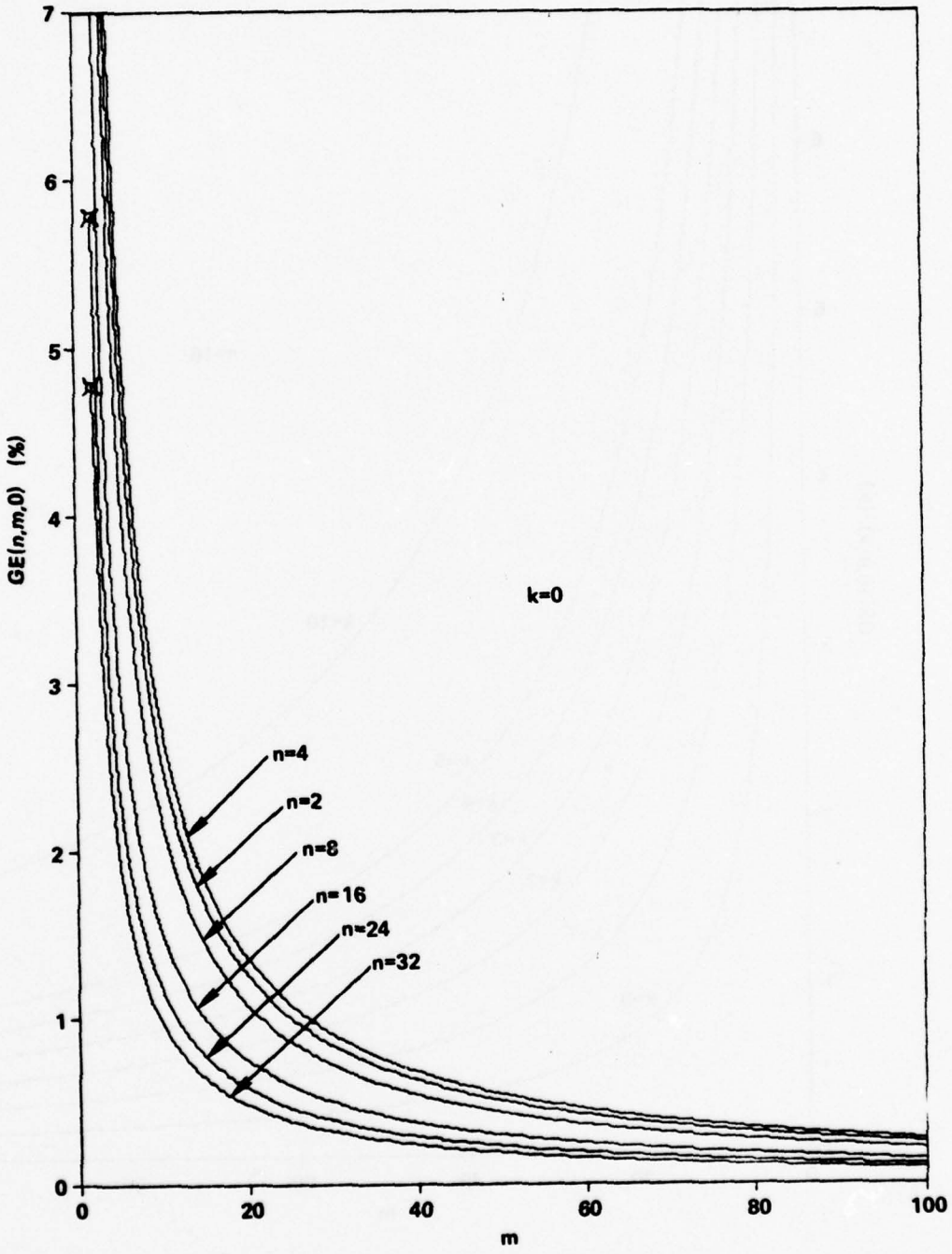


Figure 5.17. Gap probability (Method E) vs. sample size.

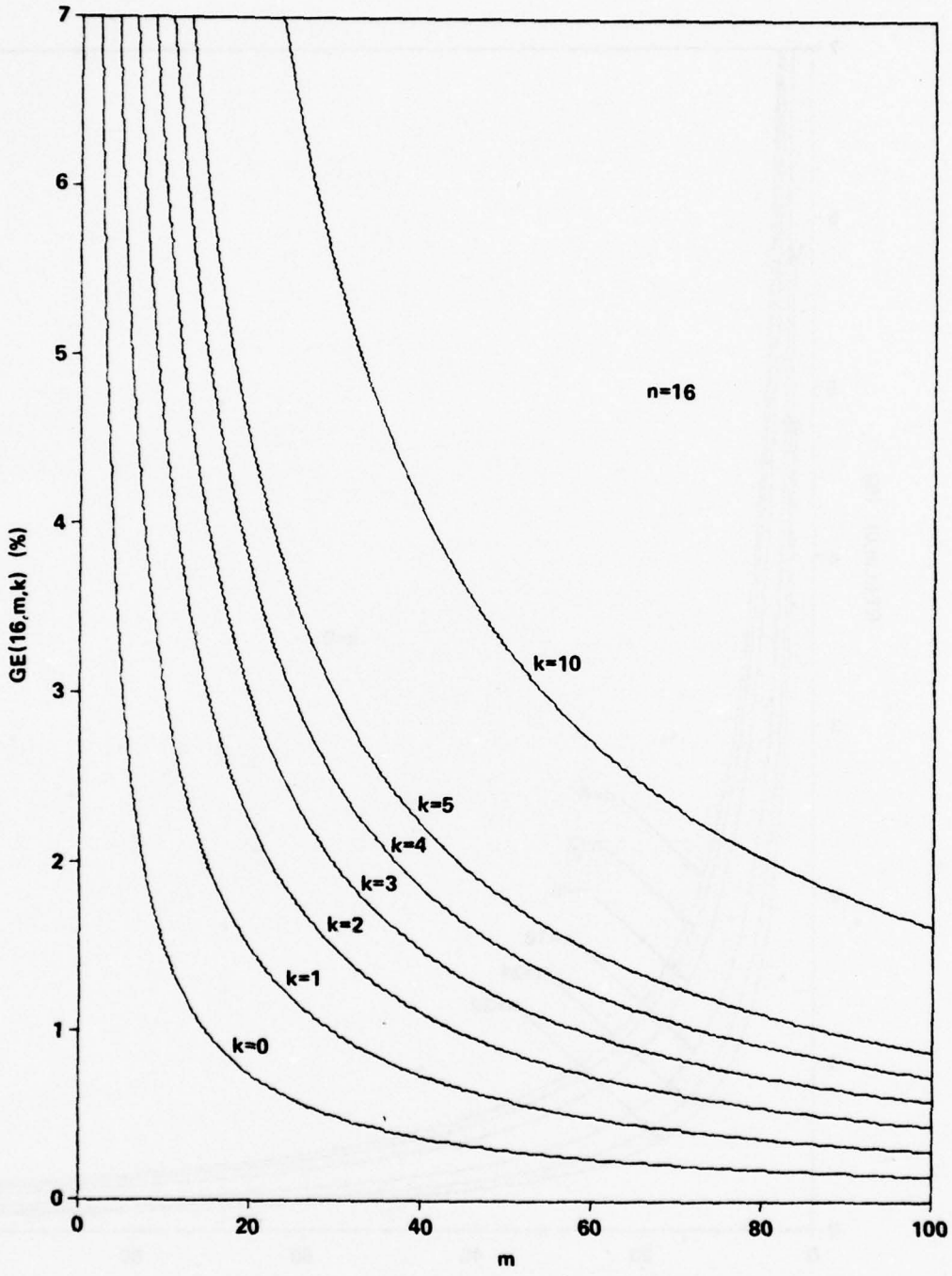


Figure 5.18. Gap probability (Method E) vs. sample size.

We define GI(n,m,k), the gap probability for method I, similarly to GE(n,m,k). Beginning with equation 5.5 we have

$$GI(n) = \frac{n-1}{n} \int_{D=0}^{\infty} \int_{t=0}^{\infty} e^{-t/x} e^{-D/x} \frac{1}{x} e^{-t/x} \, dt \, dR(D)$$

$$= \frac{n-1}{n} \int_{D=0}^{\infty} e^{-D/x} \int_{t=0}^{\infty} \frac{1}{x} e^{-2t/x} \, dt \, dR(D)$$

$$= \frac{n-1}{2n} \int_{D=0}^{\infty} e^{-D/x} \, dR(D) \qquad (5.23)$$

Suppose k=0. Then

$$GI(n,m,0) = \frac{n-1}{2n} \int_{D=0}^{\infty} e^{-D/x} \left[1-e^{-D/x}\right]^{m-2} \frac{m-1}{x} e^{-D/x} \, dD$$

$$= \frac{(n-1)(m-1)}{2n} \sum_{i=0}^{m-2} (-1)^i \binom{m-2}{i} \frac{1}{x} \int_{D=0}^{\infty} e^{-D(i+2)/x} \, dD$$

$$= \frac{(n-1)(m-1)}{2n} \sum_{i=0}^{m-2} (-1)^i \binom{m-2}{i} \frac{1}{i+2}$$

$$= \frac{n-1}{2mn} \sum_{i=2}^{m} (-1)^i \binom{m}{i} (i-1)$$

$$= \frac{n-1}{2mn} \frac{d}{dz} \frac{(1-z)^m - 1}{z} \Bigg|_{z=1}$$

$$GI(n,m,0) = \frac{n-1}{2mn} \qquad (5.24)$$

Unlike GE, GI increases with n. It has a finite upper bound of 1/2m as sentences grow very large. Figure 5.19 illustrates a family of GI curves. For general k, equations 5.23 and 5.14 yield

$$GI(n,m,k) = \frac{n-1}{2n} \int_{D=0}^{\infty} e^{-D/x} (m-1)\binom{m-2}{k} \sum_{i=0}^{m-k-2} (-1)^i \binom{m-k-2}{i} \frac{1}{x} e^{-(i+k+1)D/x} dD$$

From the derivation following 5.21 leading to 5.22 we have

$$GI(n,m,k) = \frac{(k+1)(n-1)}{2mn} = (k+1)GI(n,m,0) \qquad (5.25)$$

Figure 5.20 shows GI versus m for various k with n=16.

Remarks on the behavior at large m

GE(n,m,0) and GI(n,m,0) each have limiting values of zero as m approaches infinity. $\underline{D}(m,0)$ is unbounded however. One must allow k to grow large in order to bound $\underline{D}$. Suppose we let k=m-j (j>1). Then

$$\lim_{m\to\infty} \underline{D}(m,k) = \lim_{m\to\infty} \sum_{i=m-j+1}^{m-1} \frac{1}{i} = 0$$

$$\lim_{m\to\infty} GE(n,m,k) = \lim_{m\to\infty} \frac{m-j+2}{mn} \sum_{i=0}^{n-2} \frac{1}{i+2} = \frac{1}{n} \sum_{i=0}^{n-2} \frac{1}{i+2}$$

$$\lim_{m\to\infty} GI(n,m,k) = \lim_{m\to\infty} \frac{(m-j+2)(n-1)}{2mn} = \frac{n-1}{2n}$$

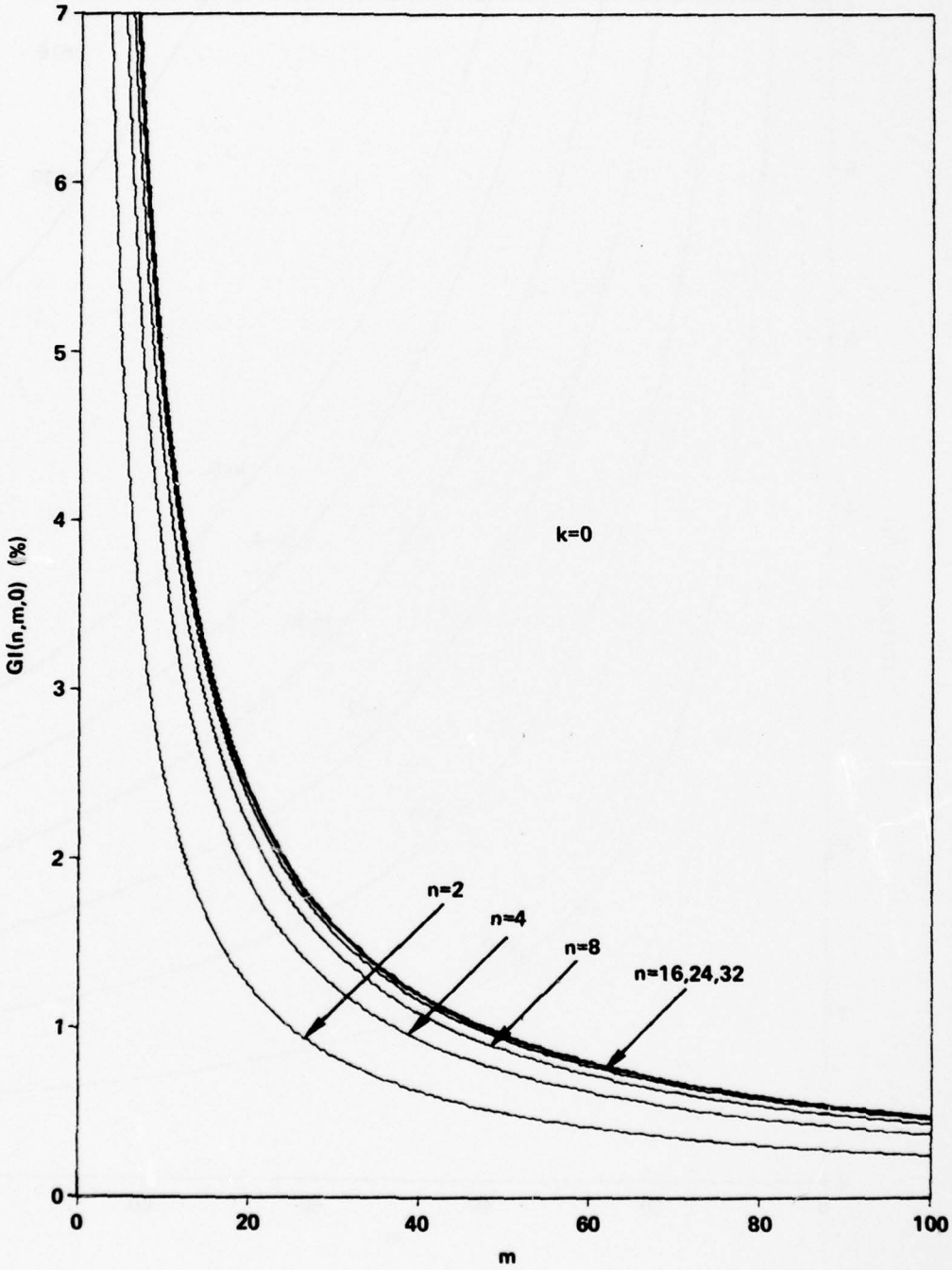


Figure 5.19. Gap probability (Method I) vs. sample size.

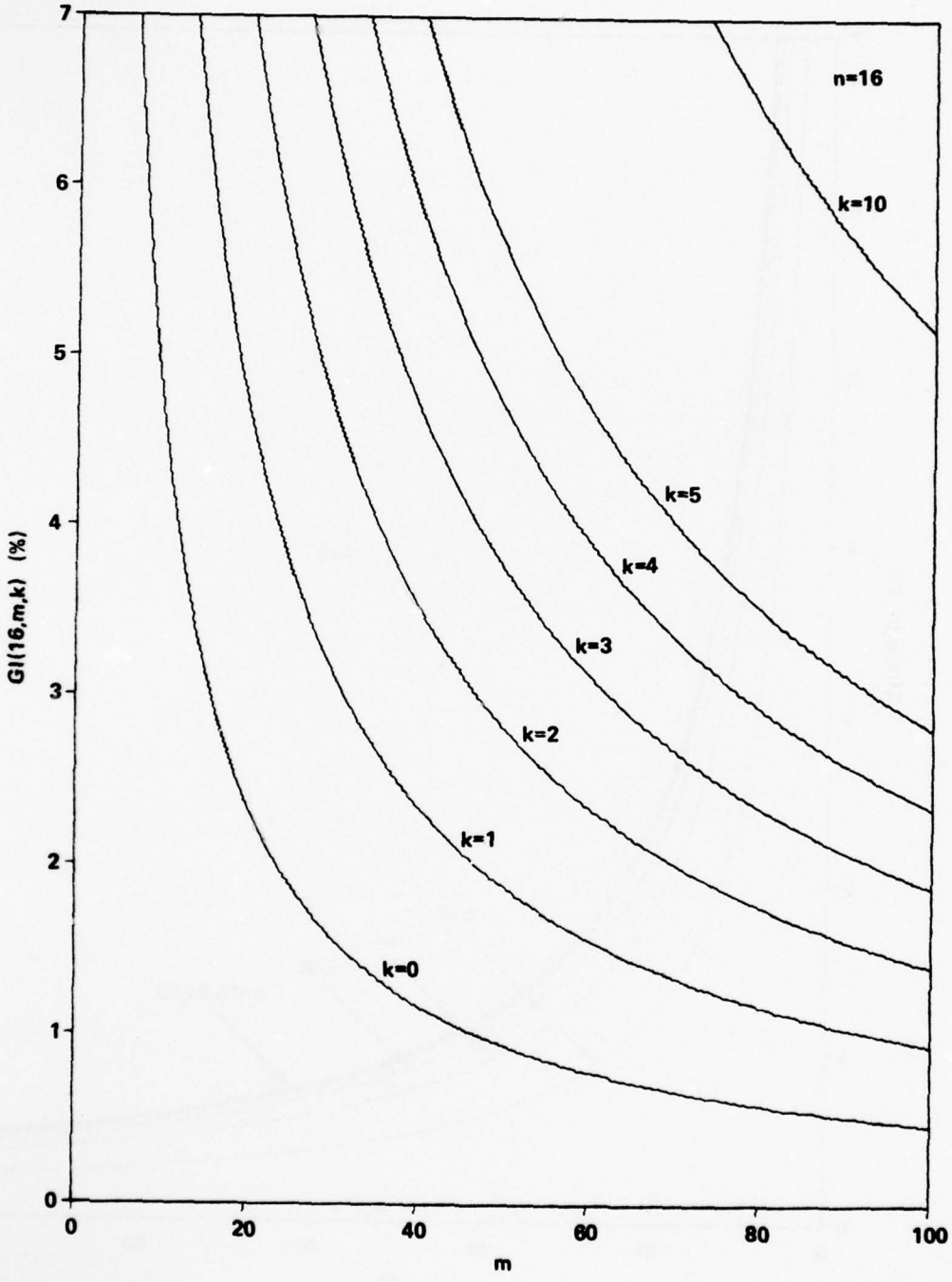


Figure 5.20. Gap probability (Method I) vs. sample size.

This approach puts D at zero but forces GE and GI outside the acceptable region. Suppose we let m and k approach infinity such that k/m approaches the value a (0<a<1). Then

$$\lim_{m\to} GE(n,m,k) = \lim_{m\to} \frac{k+1}{mn} \sum_{i=0}^{n-2} \frac{1}{i+2} = \frac{a}{n} \sum_{i=0}^{n-2} \frac{1}{i+2}$$

$$\lim_{m\to} GI(n,m,k) = \lim_{m\to} \frac{(k+1)(n-1)}{2mn} = \frac{a(n-1)}{2n}$$

$$\lim_{m\to} D(m,k) = \lim_{m\to} \sum_{i=k+1}^{m-1} \frac{1}{i}$$

An expression for this last limit must be found in order to analytically determine the optimum value of a (based on the quality function defined earlier). We begin by recognizing the following inequality

$$\sum_{i=k}^{m} \frac{1}{i} < \int_{t=k-1}^{m-1} \frac{1}{t}\,dt < \sum_{i=k-1}^{m-1} \frac{1}{i}$$

Now notice that

$$\int_{t=k}^{m} \frac{1}{t}\,dt = \log(m)-\log(k) = \log(m/k)$$

Taking the limit as m approaches infinity we see that

$$\lim_{m\to\infty} \sum_{i=k}^{m} \frac{1}{i} = \lim_{m\to\infty} \frac{1}{m} - \frac{1}{k-1} + \sum_{i=k-1}^{m-1} \frac{1}{i} = \lim_{m\to} \sum_{i=k-1}^{m-1} \frac{1}{i}$$

Hence it must be that

$$\lim_{m\to\infty} \sum_{i=k+1}^{m-1} \frac{1}{i} = \lim_{m\to\infty} \log \frac{m-1}{k+1} = \log(1/a) = -\log(a)$$

We wish to minimize the normalized distance from the origin; i.e.,

$$\left[\left(\frac{D}{d}\right)^2 + \left(\frac{G}{g}\right)^2\right]^{1/2}$$

The square root may be dropped without affecting the minimization. Notice that the limit of GE and GI may be written as a (different) constant (actually a function of n) times the parameter a. Therefore, the form of the function to be minimized is

$$\left(\frac{-\log(a)}{d}\right)^2 + \left(\frac{aK}{g}\right)^2$$

Setting the derivative (with respect to a) equal to zero we have

$$\frac{2\log(a)}{a}\left(\frac{1}{d}\right)^2 + 2a\left(\frac{K}{g}\right)^2 = 0$$

$$a^2 = -\left(\frac{g}{dK}\right)^2 \log(a) \qquad (5.26)$$

Checking the value of the second derivative at the value of a thus obtained we have

$$-2\left(\frac{1}{a}\right)^2 \log(a) + 2\left(\frac{1}{a}\right)^2\left(\frac{1}{d}\right)^2 + 2\left(\frac{K}{g}\right)^2$$

Using the value in equation 5.26 we obtain

$$2\left(\frac{K}{g}\right)^2 2 - \frac{1}{\log(a)} > 0 \quad \text{for } 0<a<1$$

Therefore the solution is indeed a minimum. Numerical solution of 5.26 (for n=16) yields a=.0801 for method E and a=.0546 for method I. This gives corresponding optimum points of GE=.0119 at $\underline{D}$=2.52x and GI=.0256 at $\underline{D}$=2.91x. Since one would be required to remember the k+1 largest values (and the minimum value), the process which allows m and k to grow arbitrarily large is computationally infeasible. The knowledge of our proximity to the optimum is however quite important.

In Figure 5.21 we plot GE versus $\underline{D}$ for k=0 and various n. The choice of m is quite critical. For the case of n=32 it can be shown that the acceptable range for m is approximately 16 to 24. If m is smaller GE is too large; if m is larger then $\underline{D}$ is too large. (The larger region of acceptability for GI causes a somewhat wider choice of acceptable m.)

In order to examine the behavior of different values of k, we fix n (at 16) and plot GE versus $\underline{D}$ in Figure 5.22. Notice that performance increases (i.e., normalized distance from the origin decreases) as k increases (with corresponding increase in m). That is, for those curves plotted, for any m,k pair there exists m' such that the performance at m',k+1 exceeds the performance at m,k. Also, for each m there is an optimum value of k (which maximizes the quality

function). Figure 5.23 shows the value of the optimal k for each m<200. The average slope of the functions approximates the value of the parameter a. Figure 5.24 gives the performance for m<200 at the optimum k for each m. Also shown is the limiting performance. The performance approaches the limit quickly at first and therefore one may achieve performance quite close to the optimum for small m (and therefore small k). To see this more clearly we have plotted in Figure 5.25 the fraction by which the performance at each m,k-optimal point exceeds the limiting performance. That is, if L(Q) is the limiting performance, then [Q(m,k-opt)-L(Q)]/L(Q) is the ratio plotted versus m in the figure. Notice that convergence to the limit is faster than a negative exponential (which would appear as a straight line in this figure).

A rule of thumb for choosing m and k is as follows. Choose m>40 and k approximately 8 percent of m for method E and 5.5 percent for method I. We see in Figure 5.25 that using such a rule for the exponential distribution of delay, n=16 and the given regions of acceptability achieves performance within 5 percent of the limiting performance. In subsequent sections we relax these assumptions and find that the rule continues to provide good performance.

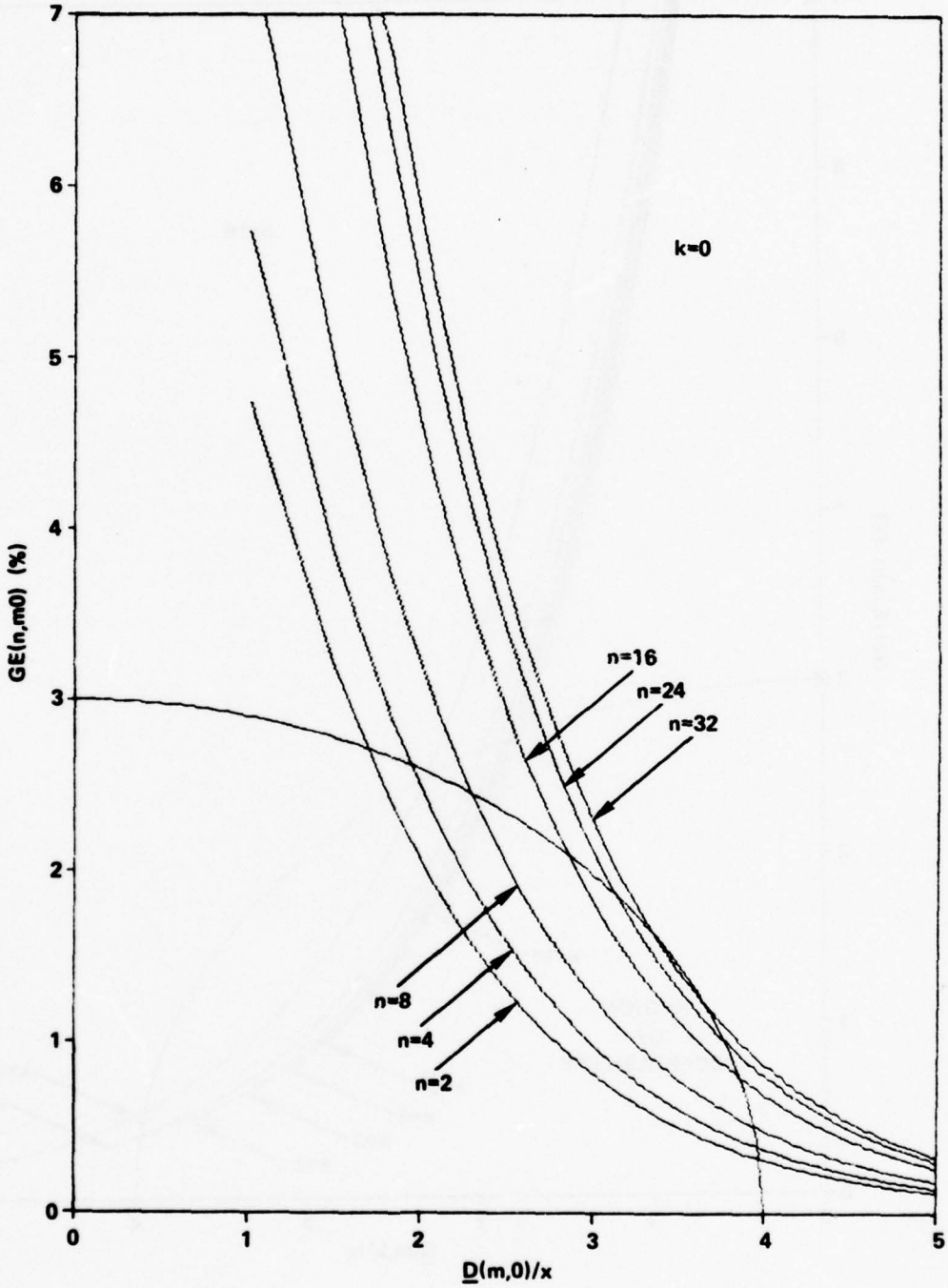


Figure 5.21. Gap probability (Method E) vs. mean range.

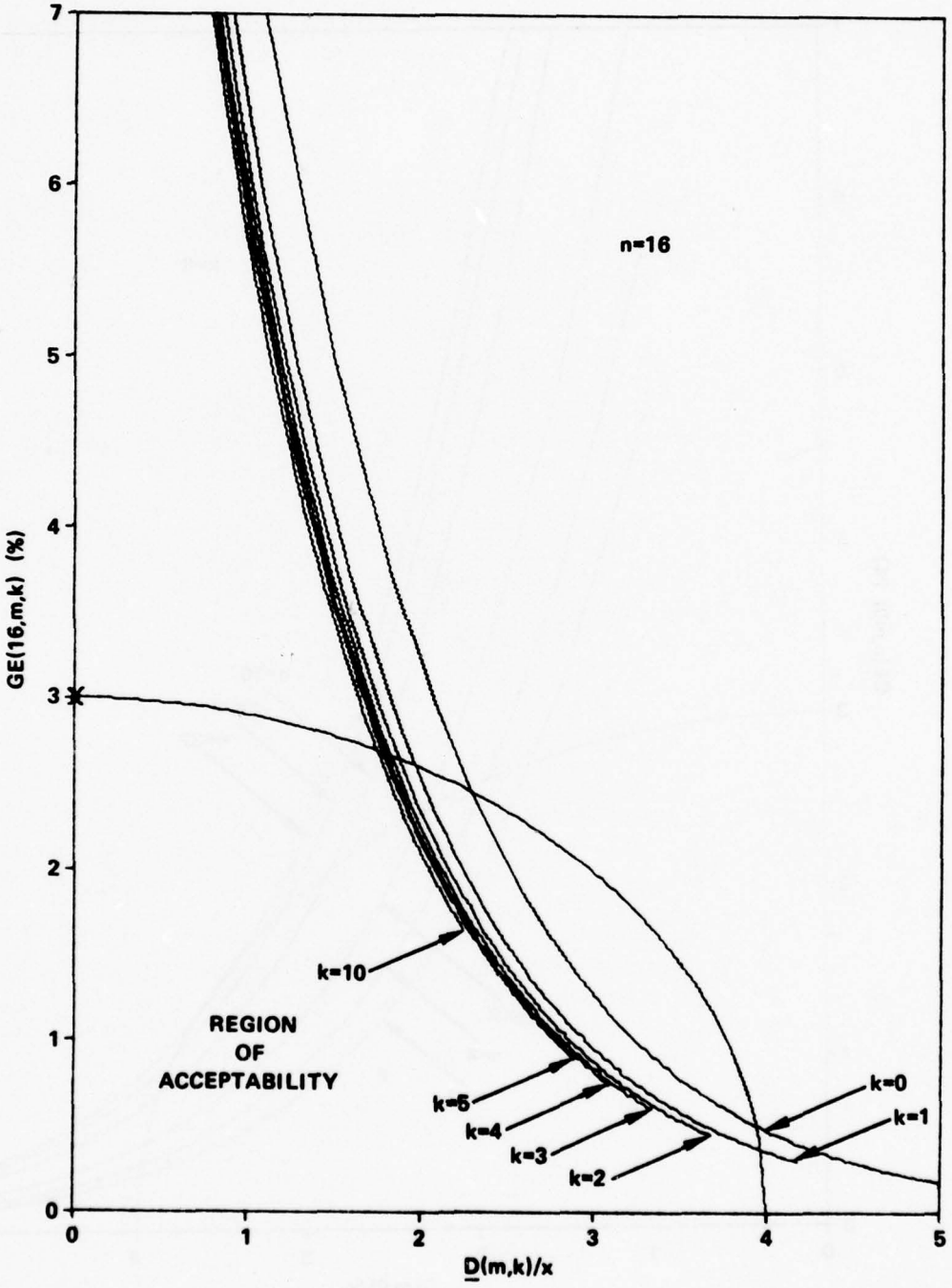


Figure 5.22. Gap probability (Method E) vs. mean range.

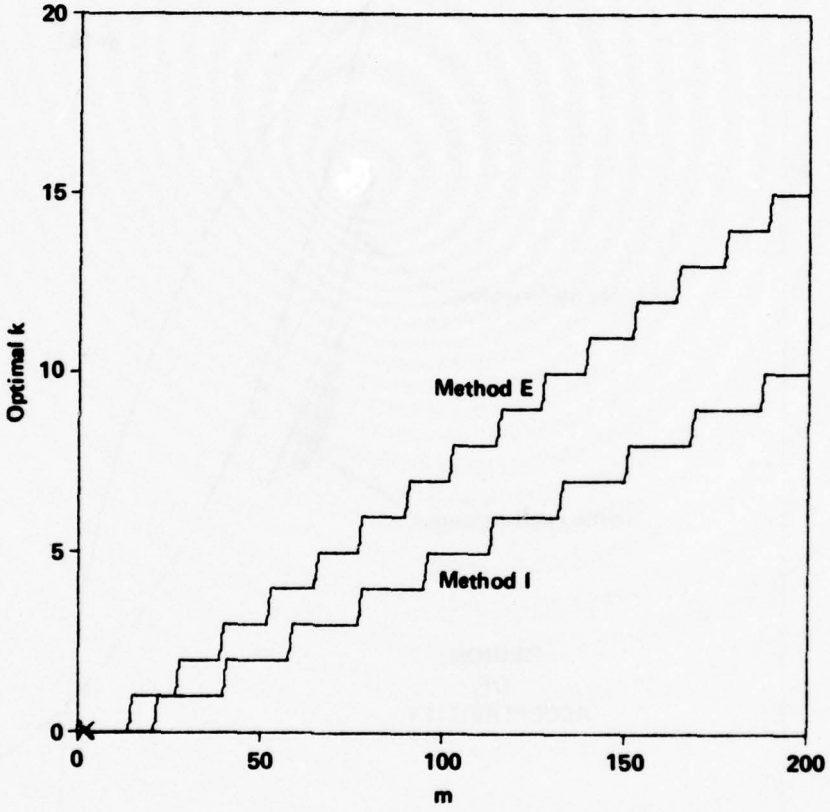


Figure 5.23. Optimal k for given m.

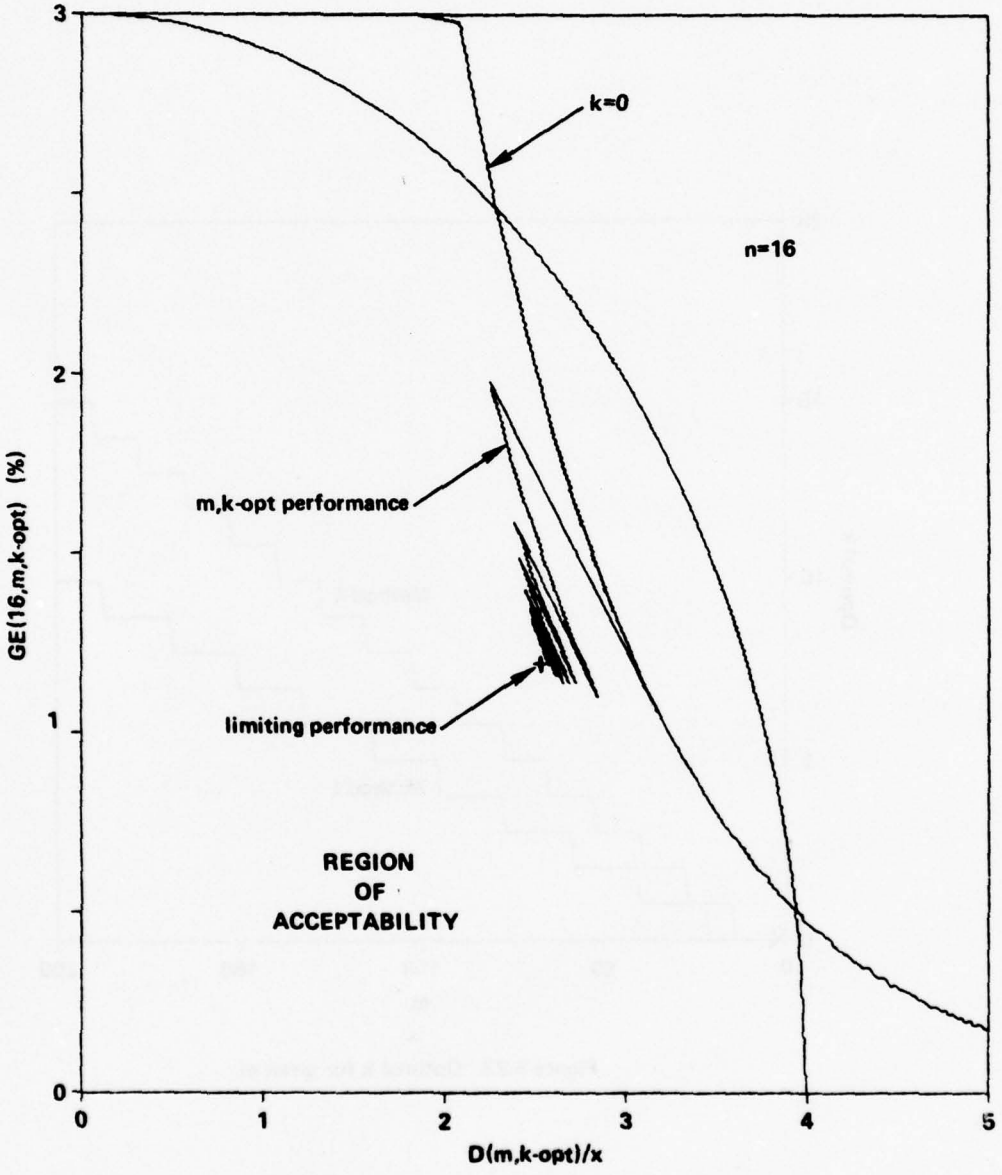


Figure 5.24. Performance at m,k-optimal (Method E).

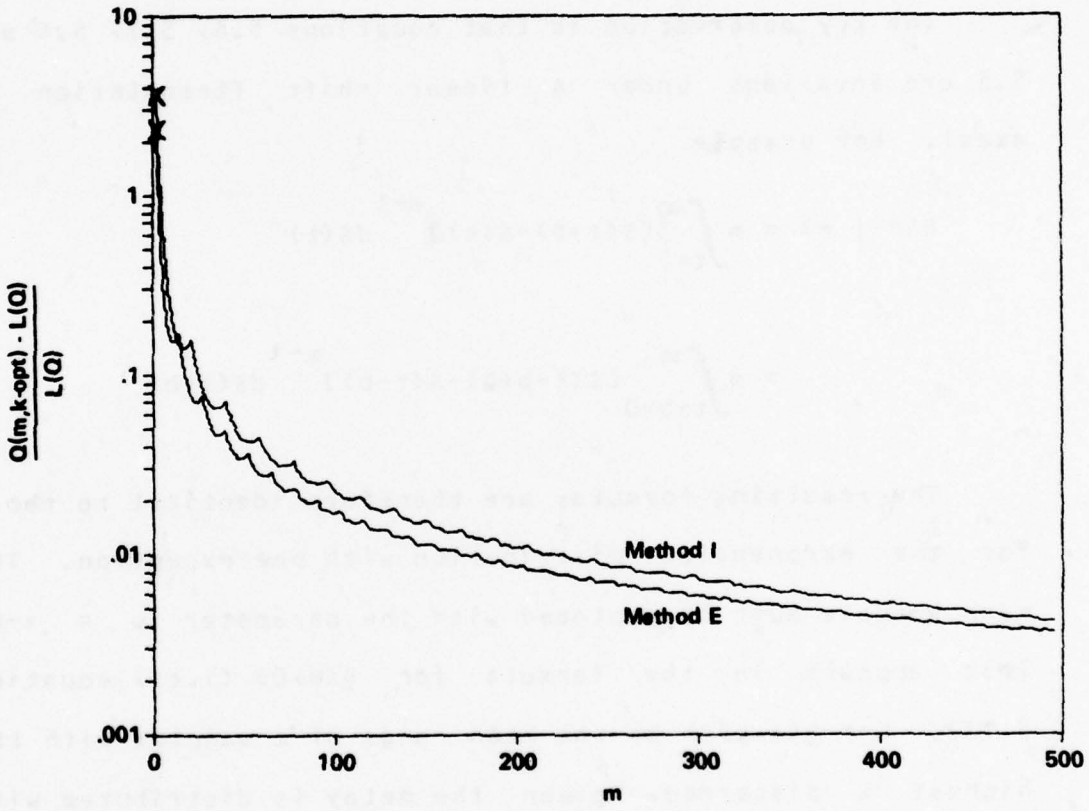


Figure 5.25. Convergence to limiting performance.

5.3.3.6.2 The shifted exponential distribution

In section 5.3.3.1.3 we noted that the shifted exponential (and the shifted Erlangian) could sometimes be used to approximate an observed delay distribution. Analytic results for this distribution are shown in this section to be similar to the results for the exponential distribution.

The key observation is that equations 5.6, 5.7, 5.4 and 5.5 are invariant under a linear shift (translation of axes). For example

$$R(D \mid m) = m\int_{t=0}^{\infty}[S(t+D)-S(t)]^{m-1}\,dS(t)$$

$$= m\int_{t-b=0}^{\infty}[S(t-b+D)-S(t-b)]^{m-1}\,dS(t-b)$$

The resulting formulas are therefore identical to those for the exponential distribution with one exception. The mean value x must be replaced with the parameter $w = x-b$. This appears in the formula for $\underline{D}(m,0)$ (i.e., equation 5.13). Let $\underline{D}(m,k,b)$ be the mean range of m samples with the highest k discarded, given the delay is distributed with exponential distribution shifted by b. Then

$$\underline{D}(m,k,b) = \frac{x-b}{x}\,\underline{D}(m,k,0) \qquad (5.27)$$

Therefore as b increases $\underline{D}(m,k,b)$ decreases. As b approaches x the limiting range is zero, which is expected since the limiting distribution is that of a constant valued

random variable. The results of the previous section therefore apply here after an appropriate scaling of the D values by the factor (x-b)/x.

5.3.3.6.3 The Erlangian family of distributions

Section 5.3.3.1.3 shows that the Erlangian (and shifted Erlangian) family of distributions provide useful approximations to observed distributions. In this section results for a family of Erlangian distributions appear. The family to be considered appears in Figure 5.26 as a family of probability density functions. Symbolic integration of equations 5.6, 5.7, 5.4 and 5.5 becomes exceedingly complicated for an Erlangian distribution. Therefore we resort to a numerical integration technique in order to solve for D, GE and GI.

Using the so called parabolic rule of numerical integration [Hild 74] (p. 96), we evaluate equations 5.7, 5.4, and 5.5. The results appear in Figure 5.27 as a set of points in the D-GE plane. We have fixed n at 16 and show the values obtained for r=5, 10, and 50, with m=2, 10, 20, 30, 40, 50, and 100, and k=0, 1, 2, 3, 4, 5, and 10 (and k<m-1).

There many similarities between these results and those for the exponential distribution (i.e., r=1). Connecting the points of constant k we see that the hyperbolic shape persists. We also notice that the distance between constant k curves shrinks rapidly with increasing k, as before.

Notice that GE (and GI, not shown) decreases much more rapidly, as $\underline{D}$ increases, than for the r=1 case. (In fact a different scale was required to show some degree of detail.) This is not unexpected; as mentioned earlier the r=1 case was expected to provide a lower bound on performance. It is expected that as variability of delay decreases less buffering will be necessary to provide the same degree of gap probability. This is indeed the case and at the limit as r approaches infinity delay becomes constant and buffering is no longer required to reduce the gap probability.

Relative distance from the origin (i.e., communication quality) is based on the region of acceptability which we have not specified for this family of distributions. We have circled the points m = 40, 50, and 100 with k = 3, 4, and 10 respectively (i.e., those closest to the rule of thumb). Notice that points provide a rather good balance of mean delay with gap probability.

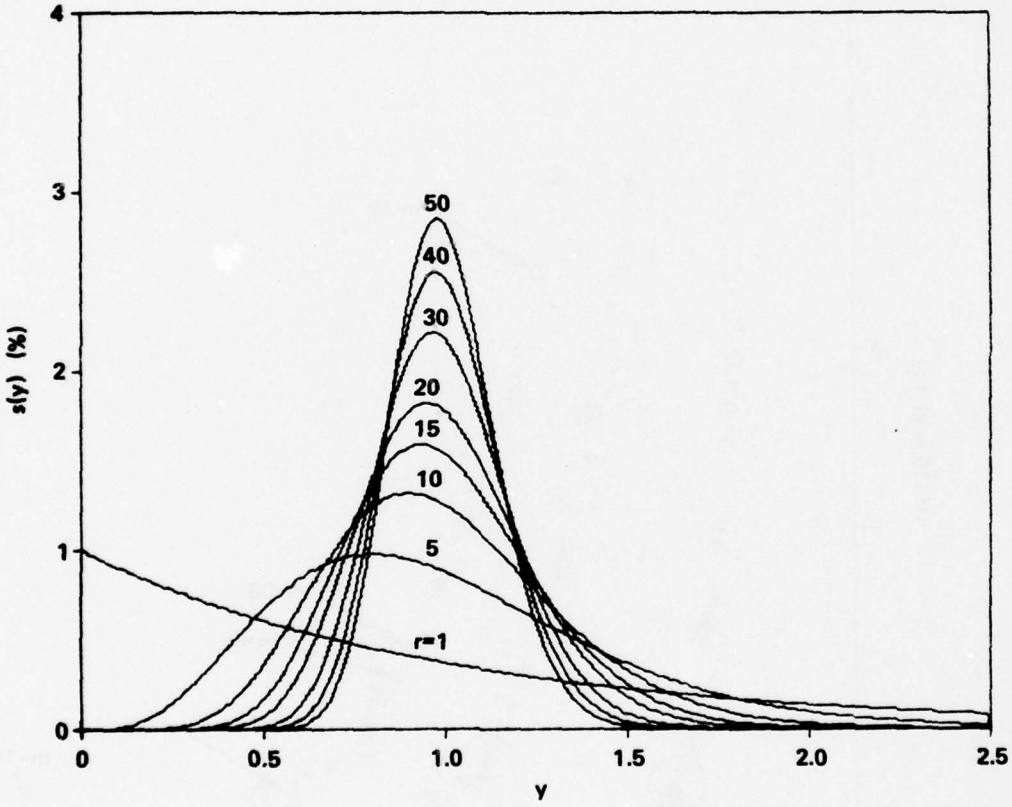


Figure 5.26. Erlang family of density functions.

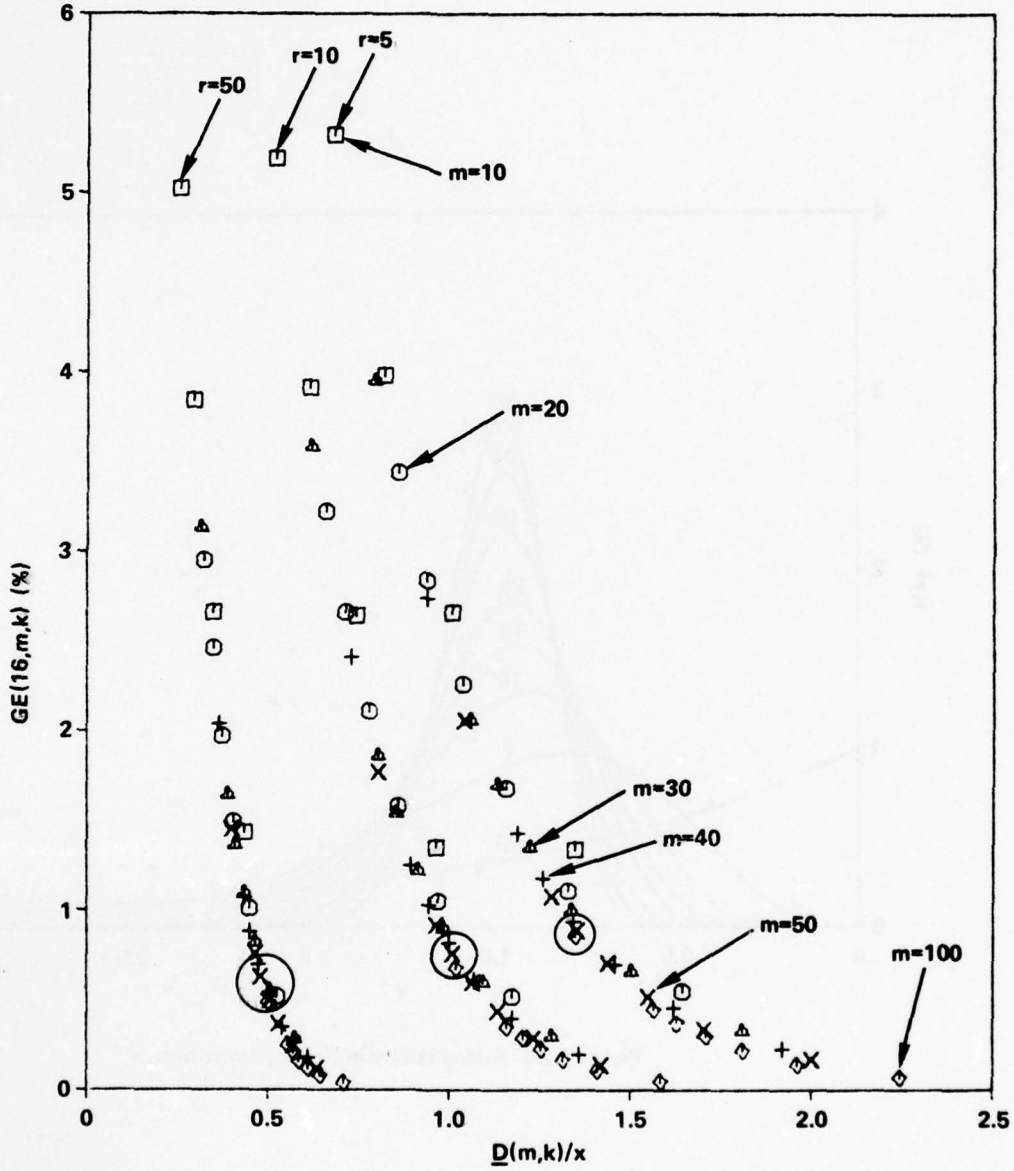


Figure 5.27. Performance with Erlang distributions. (r=5,10,50)

5.3.3.7 Simulation results

Some assumptions were made in earlier sections in order to provide a tractable model of the system. Even with these assumptions, analytic results were obtained only for a small class of distributions. We therefore have resorted to simulation in order to relax some of the assumptions made. Presented in this section are the results of that simulation.

The simulation program is driven by a sequence of delays. Therefore it is possible to remove the assumption of statistical independence among samples. This assumption is relaxed by using actual delay strings from ARPANET measurements and from simulation.

It is also possible to remove the assumption of fixed sentence size. Suggested by the early work of Norwine and Murphy [Norw 38], we have used a geometric distribution of sentence length (in terms of packets), with a mean which corresponds roughly to the 4.14 seconds reported in [Norw 38] as the average length of a talkspurt.

The simulations were performed as follows. First the delay string is "randomly" divided into sentences. At the beginning of each sentence a selection of D is made based on the particular algorithm (range monitor or delay tracker) being simulated. Knowing the delay of the first message of the sentence and D we determine the number of gaps which would have occurred were this the sequence of delays experienced by an actual sentence. This is repeated for each

sentence in the string and the results are recorded.

We show only the results using two of the measured delay strings. Q is now expressed in msec instead of a multiple of the mean delay. In Figures 5.28(a) and 5.29(a) are the results of the range monitoring techniques using m = 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, and 100, with k = 0, 1, 2, 3, 4, 5, and 10 ($k<m-1$). The performance retains the hyperbolic character to a large extent. Notice that a wide range of m,k pairs give performance below GE=.03 (and GI=.06, not shown). The $m>40$, $k=.08m$ rule provides good performance here as well, though not apparent from the figure.

In Figures 5.28(b) and 5.29(b) are the results of the delay tracker methods. The constant c scheme (represented by x's in the figures, each x is the result of a single value or c) performs quite well considering its simplicity. One must, however, be quite careful in the selection of c. Note that the range of "good" c varies with delay distribution, which is effected by a number of factors of which the user is unaware (i.e., network distance, network load, etc.).

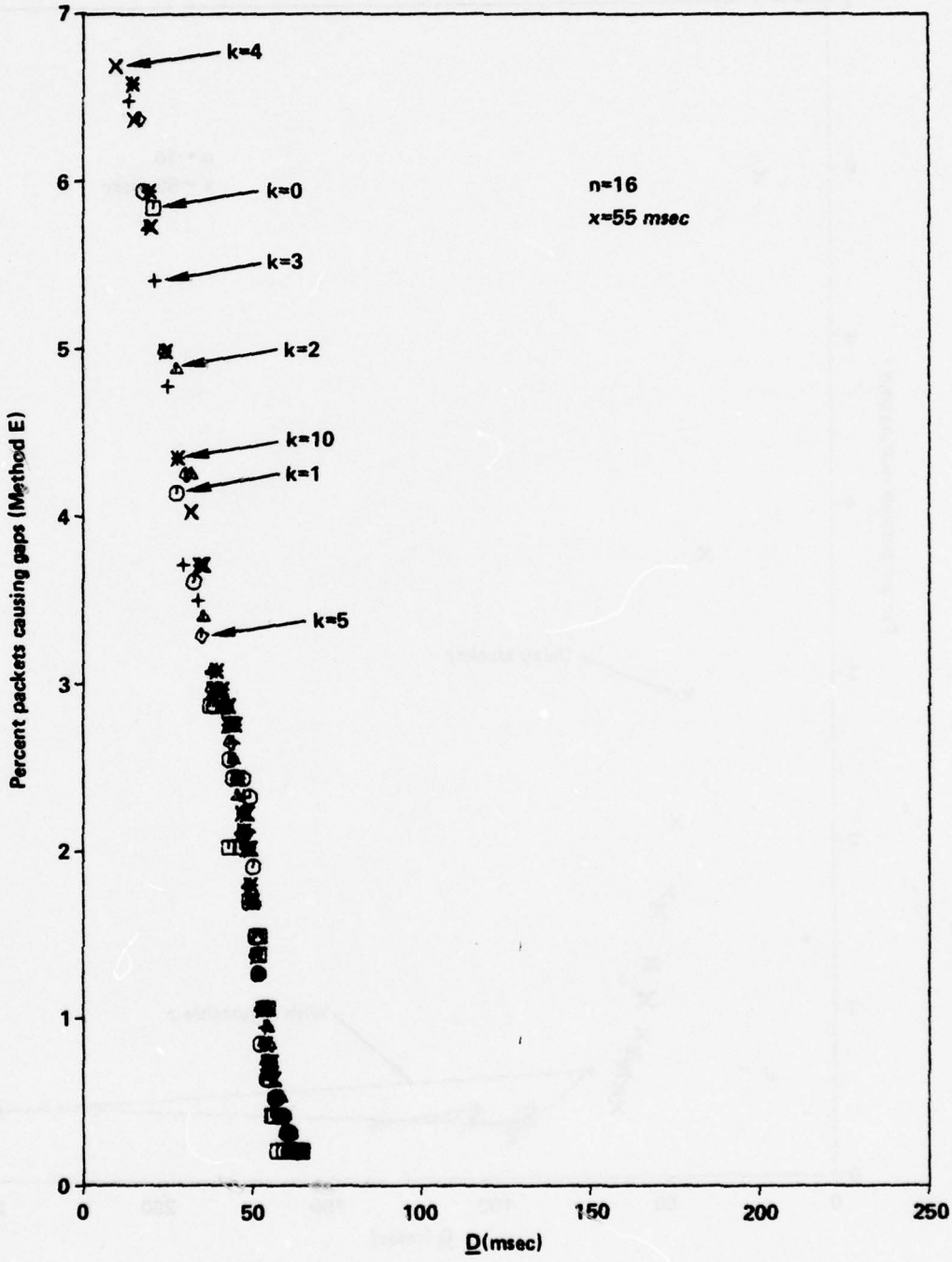


Figure 5.28(a). Range prediction performance simulation (Method E, measurement, 1 hop).

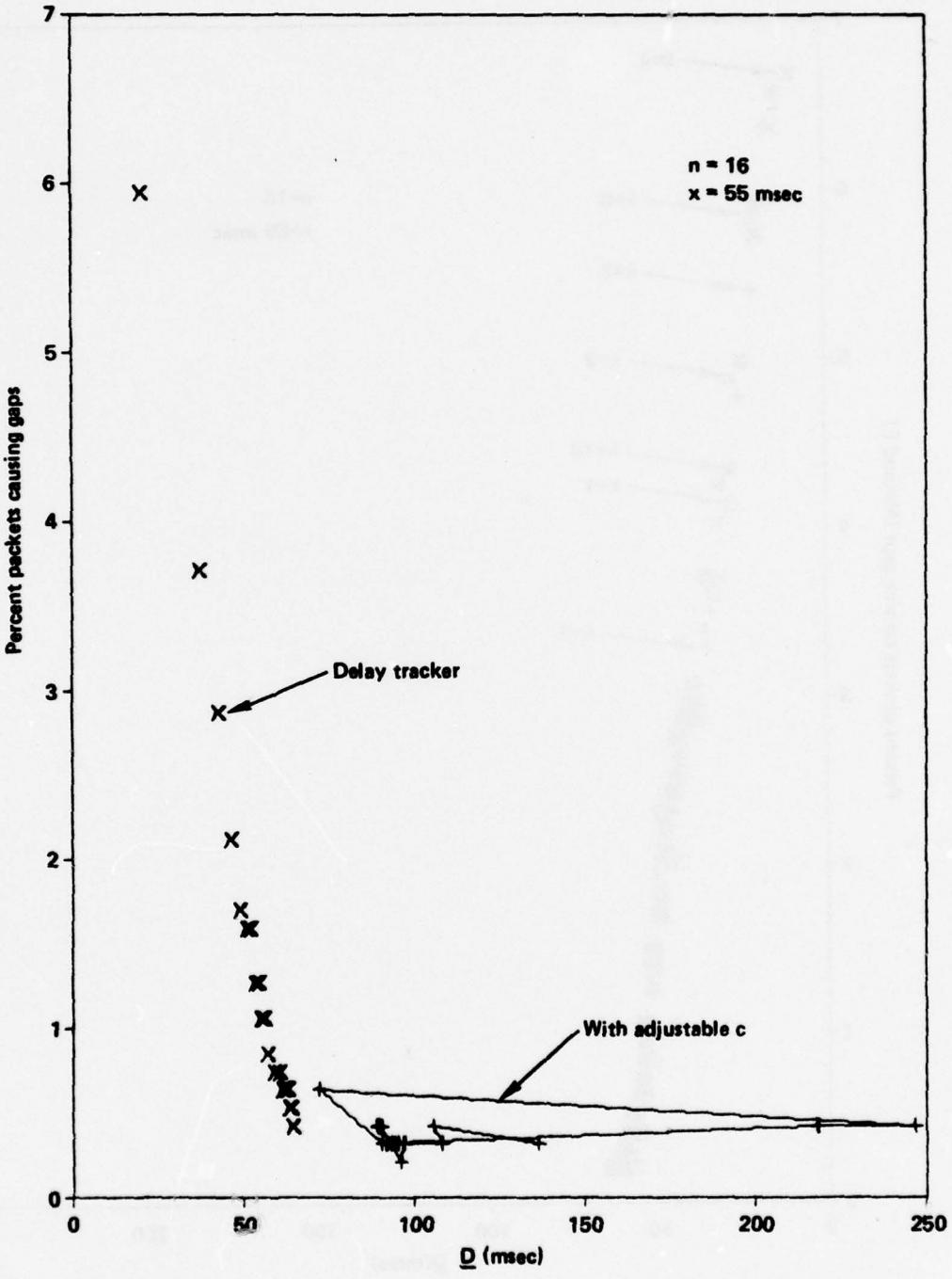


Figure 5.28(b). Delay tracker performance simulation (Method E, measurement, 1 hop).

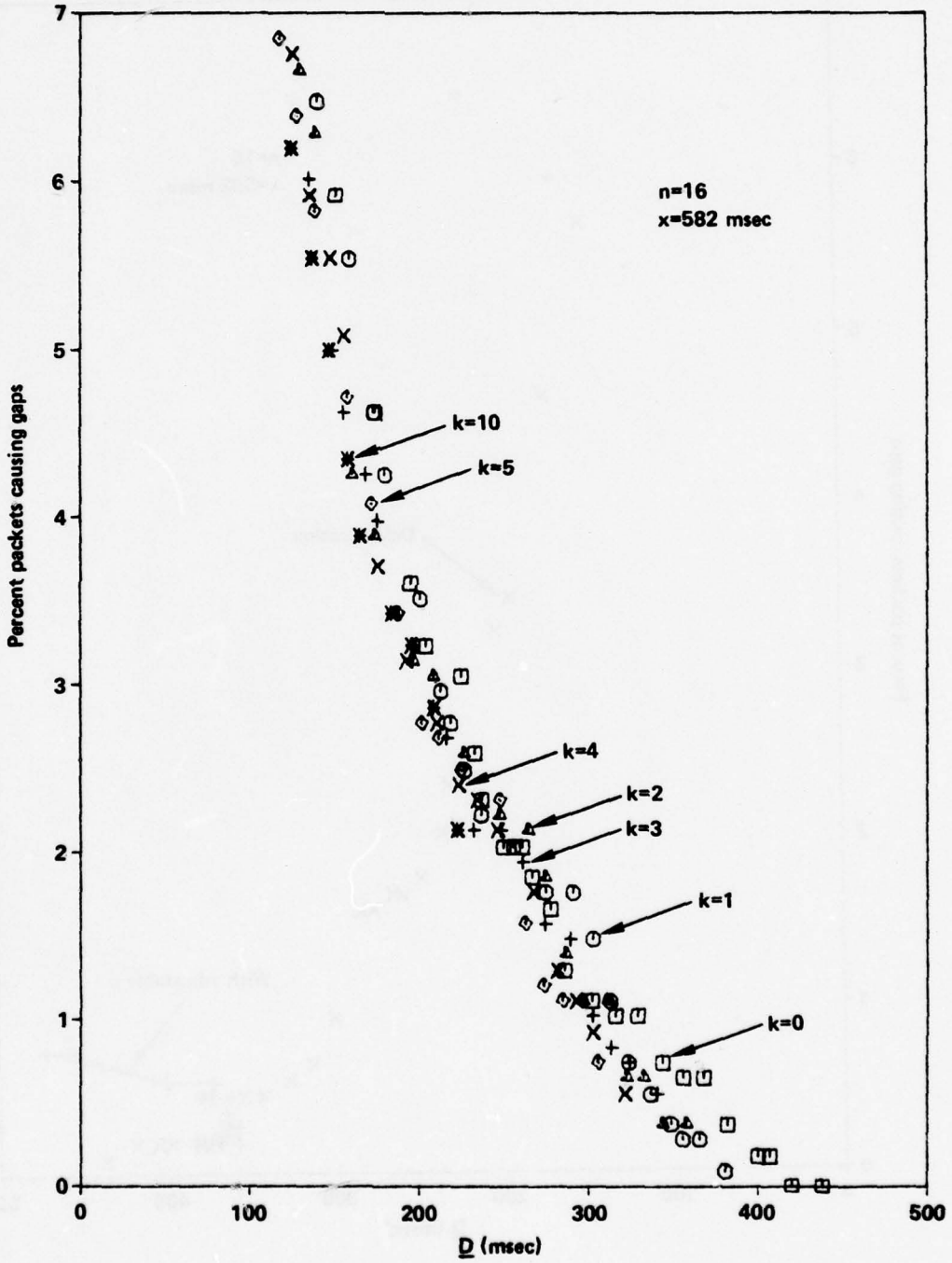


Figure 5.29(a). Range prediction performance simulation (Method E, measurement, 10 hops).

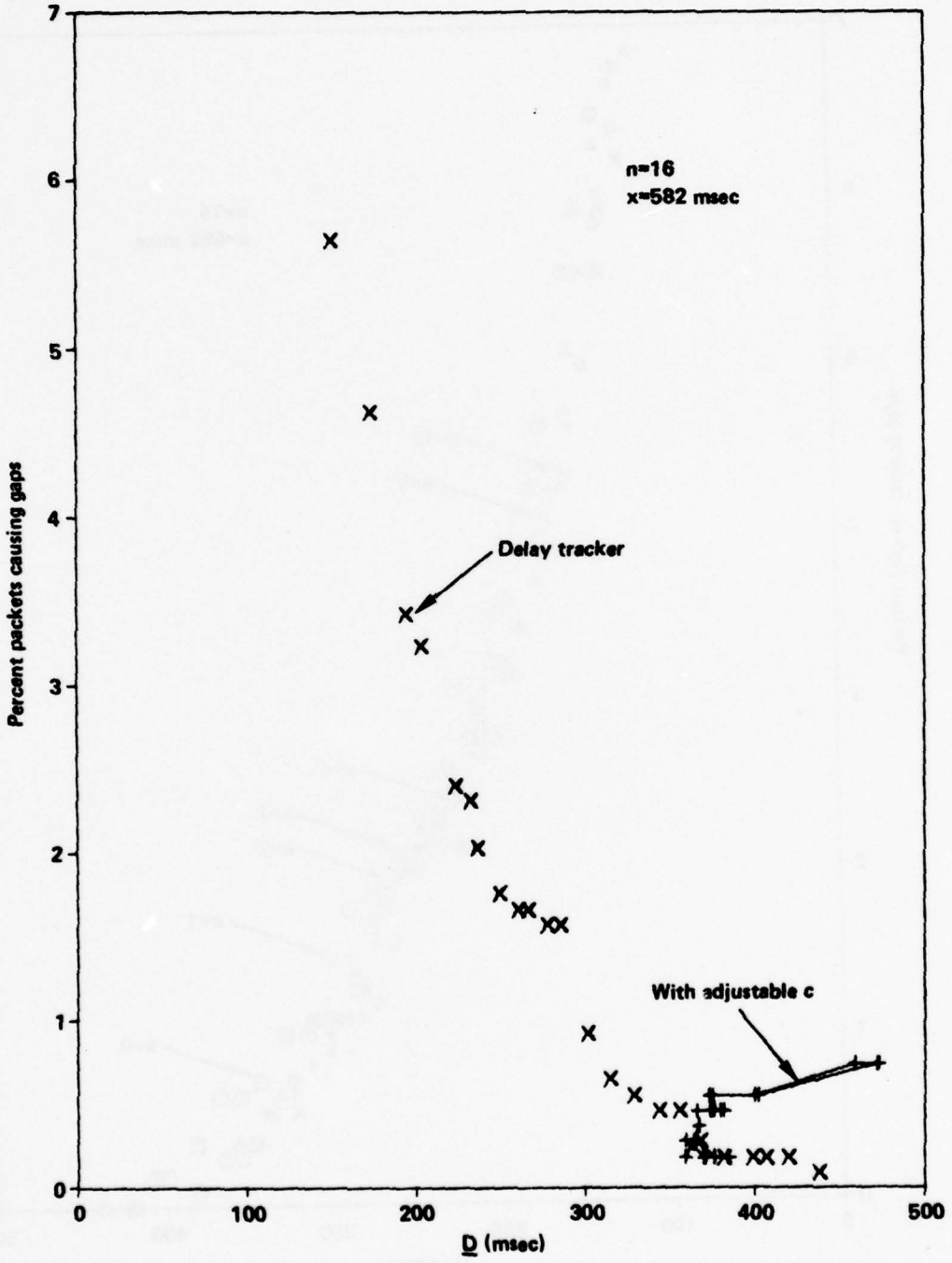


Figure 5.29(b). Delay tracker performance simulation (Method E, measurement, 10 hops).

A very interesting fact is now apparent regarding the performance of the scheme which dynamically selects c. Each point (a + sign in the figures) shows the performance for a particular initial value of c. Larger values of c may cause both larger mean delay and more gaps! While the performance is quite good in terms of gap probability, D depends critically on the selection of the initial value of c. This instability is effected by the selection of the gapless string length parameter s. It would probably be possible to find an s which would result in stable behavior for these particular delay strings. Finding an s which guarantees stability for all (or even some most likely) strings, however, is probably impossible.

5.4 Conclusions

Regarding source buffering for stream traffic communication, we found that no packet-by-packet strategy could adapt to the rapidly changing network delay in a way which effectively reduced end-to-end delay (or its variance) or significantly increased throughput. Therefore, we conclude that sending strategy should remain fixed for long periods and only adapt to major changes in network performance, perhaps by using feedback from the destination buffering algorithm.

In the area of destination buffering, the results are more promising. One can devise buffering schemes which effectively balance the frequency (and duration) of output

gaps against end-to-end delay. We have produced a framework by which such schemes may be compared. By applying that framework, we have found, for example, that among these techniques, delay range prediction is advantageous to delay tracking. This is the case because delay tracking requires user input and may be unstable in certain cases. Our approach of looking at each end of a continuum of playback schemes has produced bounds of performance within which the performance of all schemes in the continuum must lie. We have gained in understanding the tradeoff between gaps and delay. The above should provide some useful tools for designing new and better buffering and playback schemes.

In a more general sense we have gained some insight into the area of network delay distribution and the independence of successive delay.

CHAPTER 6

CONCLUSIONS AND SUGGESTIONS FOR FUTURE RESEARCH

6.1 Conclusions

We have identified three areas within the ARPANET procedures in which the performance of stream traffic communication may be significantly enhanced. In each case it was demonstrated (by simulation) that the traditional forms of data communication (i.e., the LD and HT classes defined in Chapter 1) would have improved performance as well. The underlying cause of each of the areas of poor performance was eventually found to be the routing update procedure. However, the lessons learned are worth separate mention. We now state three rules of thumb, derived from this research, which apply in general to packet switching (at least of the ARPANET variety).

1) Routing loops must be controlled.

2) Great care must be taken whenever a locally optimum policy is violated, in order to insure that the violation improves global performance in some significant way.

3) In (at least) large networks, periodic update routing procedures require excessive overhead. Therefore, for large networks we must update routing information by schemes which are asynchronously

triggered by the failure or repair of network components or by heavy congestion in the network.

In Chapter 2 we found that the most effective periodic update routing procedure (of those considered) was the local loop-free scheme LLFR. The general loop-free scheme LFR was less effective because of the infrequent occurrence of multinode loops coupled with the increased overhead required to prevent multinode loops. However, it should be clear that under an asynchronous update procedure we must require each update to produce a loop-free set of routing tables. Therefore LFR is a candidate for a distributed asynchronous update routing procedure in large networks. We could call this new technique ALFR (pronounced as a New Englander pronounces the first letter of the of the Greek alphabet).

Rule 2 is generated by our discovery, in Chapter 3, that processing routing packets at low priority produced both local and global performance improvement in terms of average delay. Under an asynchronous update scheme this discovery may not have been made. Perhaps under such a scheme, since updates are triggered by something very "bad" (or good), high priority routing may produce better performance globally even though it would still be worse locally.

We found in Chapter 4 that periodic update routing is costly in large networks. With periodic update routing, in order to insure that news of a failure is propagated through a large network it is necessary to reduce the period of the routing updates. As the network grows bigger so does the

overhead due to routing. With such high overhead the purpose of adaptive routing (i.e., to reduce average delay and increase throughput in the face of changing topology or traffic patterns) is defeated.

Our investigation of sending and receiving strategies has yielded the following results. We found no effective method of using delay prediction on a message-by-message basis to assist in the sending strategy due to the large variability of delay for successive messages. Therefore, we conclude that the sending strategy should remain fixed for some "long" periods of time and modified only as a result of large variations in network performance. The study of destination buffering proved to be more fruitful. The delay tracking methods are attractive because of implementation ease, but the requirement that the user specify a starting value is undesirable. For "good" starting values the delay trackers perform as well as the slightly more complicated range predictors. Unfortunately, however, a good selection of starting value is dependent on factors of which the user is completely unaware. On the other hand, using a fixed (or automatically adjusted) m-k pair allows the automatic selection of "destination wait time" based more on the network dynamics. Therefore range prediction is preferable to delay tracking.

6.2 Suggestions for future investigation

Several areas related to this work have been left untouched here. We have considered only part of the larger problem of network design. That is, to optimize the design (e.g., minimize the cost) of a network which is to carry several classes of traffic, each with a different set of constraints, over the set of design variables which includes topology, capacity assignment, routing procedure, system priorities, buffering, and flow and congestion control. Some of the more fruitful individual areas would seem to be routing and flow and congestion control. We stated earlier that asynchronous update routing procedures deserve investigation. Some work has been done by Kleinrock and Fultz [Fult 72] in this area for small networks. The topic should be further studied especially with regard to large networks.

In Chapter 5 we modeled the stream traffic delay as an independent random variable possessing some probability distribution function. A better characterization using random processes would allow the creation of more accurate models of the destination buffering problem as well as aid in the above design problem. The buffering problem itself could also be extended to include a notion of gap duration. The model should also be used or extended to provide information regarding the minimum buffer size required (at the destination device) to guarantee certain performance.

Finally the buffering analysis could be extended immediately to the other packet technologies mentioned in

Chapter 1 by providing a delay distribution in each case. Also, the destination buffering and associated delay monitoring could be used to provide feedback to the source in order that changes be made in sending strategy and the compression scheme, at least in the case of speech.

BIBLIOGRAPHY

[Abra 70] Abramson, N. "The ALOHA System -- Another Alternate for Computer Communication," AFIPS Conference Proceedings, Vol. 37, FJCC, Houston, Texas, November 1970, pp. 281-285.

[Atal 71] Atal, B. S. and S. L. Hanauer. "Speech Analysis and Synthesis by Linear Prediction of the Speech Wave," J. Acoustic Society of America, Vol. 50, No. 2 (Part 2), pp 637-655.

[Barb 70] Barber, D. L. A. and D. W. Davies. "The NPL Network," National Physical Laboratory, Teddington, England, Division of Computer Science, October 1970.

[Barb 72] Barber, D. L. A. "The European Computer Network Project," The First International Conference on Computer Communications, Washington, D. C., October 1972, pp. 192-200.

[BBN 69] Bolt, Beranek and Newman, Inc. "Specifications for the Interconnection of a HOST and an IMP," Cambridge, Massachusetts, Report No. 1822, May 1969, (revised January 1976).

[BBN 74] Bolt, Beranek and Newman, Inc. "Interface Message Processors for the ARPA Computer Network," Cambridge, Massachusetts, Quarterly Technical Report No. 4, Report No. 2717, January 1974.

[BBN 75] Bolt, Beranek and Newman, Inc. "Interface Message Processors for the ARPA Computer Network," Cambridge, Massachusetts, Quarterly Technical Report No. 2, Report No. 3106, July 1975.

[Beno 71] Benoit, J. W., I. W. Cotton, and D. C. Wood. "Proposed Implementation Plan for a WWMCCS Intercomputer Network," Mitre Corp., Washington, D. C., Report No. WP-9807, December 1971.

[Cegr 75] Cegrell, T. "A Routing Procedure for the TIDAS Message-Switching Network," IEEE Transactions on Communications, Vol. COM-23, No. 6, June 1975, pp. 575-585.

[Cohe 74] Cohen, D. Private communication, 1974-76.

[Cohe 76] Cohen, D. "Specifications for the Network Voice Protocol," University of Southern California, Information Sciences Institute, Marina Del Rey, California, Network Speech Compression Note 68, March 1976.

[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 (Also published as a Ph. D. dissertation).

[Crow 75] Crowther, W. R., F. E. Heart, A. A. McKenzie, J. M. McQuillan, and D. C. Walden. "Issues in Packet Switching Network Design," AFIPS Conference Proceedings, Vol. 44, NCC, Anaheim, California, May 1975, pp 161-175.

[Davi 71] Davies, D. W. "The Control of Congestion in Packet Switching Networks," Proceedings of the Second ACM IEEE Symposium on Problems of the Optimization of Data Communications Systems, Palo Alto, California, October 1971, pp 46-49.

[DPAC 74] "DATAPAC Standard Network Access Protocol," Trans-Canada Telephone Systems, Computer Communications Group, Ottawa, Canada, November 1974.

[Falk 77] Falk, H. "'Chipping in' to Digital Telephones," IEEE Spectrum Vol. 14, No. 2, February 1977, pp 42-46.

[Farb 75] Farber, D. J. and K. C. Larson. "Network Security via Dynamic Process Renaming," Proceedings of the Fourth Data Communications Symposium, Quebec City, Canada, October 1975 pp. 8.13-8.18.

[Fell 57] Feller, W. An Introduction to Probability Theory and Its Applications, Second edition, Vol. I, Wiley, New York, 1957.

[Forg 74] Forgie, J. W. "Speech Understanding Systems," Massachusetts Institute of Technology, Lincoln Laboratory, Lexington, Massachusetts, ESD-TR-74-218, May 1974.

[Forg 75] Forgie, J., C. McElwain, and C. Weinstein. "Network Measurement Facility on TX-2," Massachusetts Institute of Technology, Lincoln Laboratory, Network Speech Compression Note 55, February 1975.

[Forg 75a] Forgie, J. W. "Speech Transmission in Packet-Switched Store-and-Forward Networks," AFIPS Conference Proceedings, Vol. 44, NCC, Anaheim, California, May 1975, pp. 137-142.

[Forg 75c] Forgie, J. W. and C. L. McElwain. "ARPANET Delay Measurements," Massachusetts Institute of Technology, Lincoln Laboratory, Network Speech Compression Note 70, July 1975.

[Forg 76] Forgie, J. W. and C. W. McElwain. "Some Comments on NSC Note 78 'Effects of Lost Packets on Speech Intelligibility'," Massachusetts Institute of Technology, Lincoln Laboratory, Network Speech Compression Note 92, March 1976.

[Forg 76a] Forgie, J. W. Personal communication, October 1976.

[Fran 70] Frank, H., I. T. Frisch, and W. Chou. "Topological Considerations in the Design of the ARPA Computer Network," AFIPS Conference Proceedings, Vol. 36, SJCC, Atlantic City, May 1970, pp. 581-587.

[Fran 72] Frank, H., R. E. Kahn, and L. Kleinrock. "Computer Communication Network Design -- Experience with Theory and Practice," AFIPS Conference Proceedings, Vol. 40, SJCC, Atlantic City, May 1972, pp. 255-270.

[Fult 72] Fultz, G. L. "Adaptive Routing Techniques for Message Switching Computer-Communication Networks," Computer Science Department, School of Engineering and Applied Science, University of California, Los Angeles, UCLA-ENG-7252, July 1972 (Also published as a Ph. D. dissertation).

[Gall 77] Gallager, R. G. "A Minimal Delay Routing Algorithm Using Distributed Computation," IEEE Transactions on Communications, Vol. COM-25, No. 1, January 1977, pp. 73-85.

[Gall 77a] Gallager, E. F. "The Military Goes Digital," IEEE Spectrum Vol. 14, No. 2, February 1977, pp 42-45.

[Gerl 73] Gerla, M. "The Design of Store-and-Forward (S/F) Networks for Computer Communication," Computer Science Department, School of Engineering and Applied Science, University of California, Los Angeles, UCLA-ENG-7319, January 1973 (Also published as a Ph. D. dissertation).

[Gumb 67] Gumbel, E. J. Statistics of Extremes, Fourth printing, Columbia University Press, New York, 1967.

[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, Vol. 36, SJCC, Atlantic City, May 1970, pp. 551-567.

[Hear 76] Heart, F. E. and D. C. Walden. "Communications Applications for the Pluribus Computer," Conference Record, National Telecommunications Conference, Dallas, Texas, November 1976, pp. 7.1.1-7.1.5.

[Hild 74] Hildebrand, F. B. Introduction to Numerical Analysis, Second Edition, McGraw-Hill, New York, 1974.

[Hove 76] Hovey, R. B. "Packet Switched Networks Agree on a Standard Interface," Data Communications, May/June 1976, pp. 25-39.

[Jame 72] James, R. T. and P. E. Muench. "AT&T Facilities and Services," Proceedings of the IEEE, Vol. 60, No. 11, November 1972, pp 1342-1349.

[Juen 76] Jueneman, R. R. and G. S. Kerr. "Explicit Path Routing in Communications Networks," Proceedings of the Third International Conference on Computer Communication, Toronto, Canada, August 1976, pp. 340-342.

[Kahn 71] Kahn, R. E. and W. R. Crowther. "A Study of the ARPA Network Design and Performance," Bolt, Beranek and Newman, Inc., Cambridge, Massachusetts, Report No. 2161, August 1971.

[Kahn 71a] Kahn, R. E. and W. R. Crowther. "Flow Control in a Resource Sharing Computer Network," Proceedings of the Second ACM IEEE Symposium on Problems of the Optimization of Data Communications Systems, 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, Vol. 44, NCC, Anaheim, California, May 1975, pp. 177-186.

[Klei 64] Kleinrock, L. Communication Nets: Stochastic Message Flow and Delay, McGraw-Hill, New York, 1964, reprinted by Dover, New York, 1972.

[Klei 70] Kleinrock, L. "Analytic and Simulation Methods in Computer Network Design," AFIPS Conference Proceedings, Vol. 36, SJCC, Atlantic City, May 1970, pp. 569-579.

[Klei 73] Kleinrock, L. and S. S. Lam. "Packet-Switching in a Slotted Satellite Channel," AFIPS Conference Proceedings, Vol. 42, NCC, New York, June 1973, pp. 703-710.

[Klei 74] Kleinrock, L. and W. E. Naylor. "On Measured Behavior of the ARPA Network," AFIPS Conference Proceedings, Vol. 43, NCC, Chicago, May 1974, pp. 767-780.

[Klei 75] Kleinrock, L. Queueing Systems Volume I: Theory, Wiley, New York, 1975.

[Klei 75a] Kleinrock, L. and H. Opderbeck. "Throughput in the ARPANET -- Protocols and Measurement," Proceedings of the Fourth Data Communications Symposium, Quebec City, Canada, October 1975, pp. 6.1-6.11.

[Klei 76] Kleinrock, L. Queueing Systems, Volume II: Computer Applications, Wiley, New York, 1976.

[Klei 76a] Kleinrock, L., W. E. Naylor, and H. Opderbeck. "A Study of Line Overhead in the ARPANET," Communications of the Association for Computing Machinery, January 1976, pp. 3-13.

[Klem 67] Klemmer, E. T. "Subjective Evaluation of Transmission Delay in Telephone Conversations," Bell System Technical Journal, Vol. 46, 1967, pp. 1141-1147.

[Krau 67] Krauss, R. M. and P. D. Bricker. "Effects of Transmission Delay and Access Delay on the Efficiency of Verbal Communication," Journal of the Acoustical Society of America, Vol. 41, No. 2, 1967, pp. 286-292.

[Magi 73] Magill, D. T. "Adaptive Speech Compression for Packet Communication Systems," Stanford Research Institute, Menlo Park, California, Network Speech Compression Note 9, NIC 20643, December 1973.

[Math 75] Mathison, S. L. "Telenet Inaugurates Service," Computer Communications Review, Vol. 5, No. 4, October 1975, pp. 24-28.

[McCa 75] McCammon, M. "Ordering of NVP Messages," Culler/Harrison, Inc., Goleta, California, Network Speech Compression Note 61, May 1975.

[McCo 75] McCoy, C., Jr. "Improvements in Routing for Packet-Switched Networks," Naval Research Laboratory, Washington, D. C., Report No. 7848, 1975 (Also published as a Ph. D. dissertation, School of Engineering and Applied Science, George Washington University, Washington, D. C.)

[McKe 74] McKenzie, A. A. Letter to S. D. Crocker, 16 January 1974.

[McKe 74a] McKenzie, A. M. "Some Computer Network Interconnection Issues," AFIPS Conference Proceedings, Vol. 43, NCC, Chicago, May 1974, pp. 857-859.

[McQu 72] McQuillan, J. M., W. R. Crowther, B. P. Cosell, D. C. Walden, and F. E. Heart. "Improvements in the Design and Performance of the ARPA Network," AFIPS Conference Proceedings, Vol. 41, FJCC, Anaheim, California, December 1972, pp. 741-754.

[McQu 73] McQuillan, J. M. "Throughput in the ARPA Network -- Analysis and Measurement," Bolt, Beranek and Newman, Inc., Cambridge, Massachusetts, Report No. 2491, January 1973.

[McQu 74] McQuillan, J. M. "Adaptive Routing Algorithms for Distributed Computer Networks," Bolt, Beranek and Newman, Inc., Cambridge, Massachusetts, Report No. 2831, 1974 (Also published as a Ph. D. dissertation, Harvard University, Cambridge, Massachusetts).

[McQu 74a] McQuillan, J. M. Private communication, October 1974.

[Metc 77] Metcalf, R. M. and D. R. Boggs. "ETHERNET: Distributed Packet Switching for Local Computer Networks," Communications of the Association for Computing Machinery, to appear 1977.

[NAC 75] Network Analysis Corporation, "ARPANET Reconfiguration Alternatives for TIP Users Performance Improvement," Glen Cove, New York, May 1975.

[Nayl 73] Naylor, W. E. "Real-Time Transmission in Packet Switched Networks," University of California, Los Angeles, Network Measurement Group Note 15, NIC 19014, September 1973.

[Nayl 74] Naylor, W. E. "A Status Report on the Real-Time Speech Transmission Work at UCLA," University of California, Los Angeles, Network Speech Compression Note 52, December 1974.

[Nayl 74a] Naylor, W. E. "Minutes of the Network Speech Compression Group Meeting (Stanford Research Institute, 13-14 September 1974)," University of California, Los Angeles, Network Speech Compression Note 44, October 1974.

[Nayl 75] Naylor, W. E. "A Loop-Free Adaptive Routing Algorithm for Packet Switched Networks," Proceedings of the Fourth Data Communications Symposium, Quebec City, Canada, October 1975 pp. 7.9-7.14.

[Nayl 76] Naylor, W. E. and L. Kleinrock. "On the Effects of Periodic Routing Updates in Packet Switched Networks," Conference Record, National Telecommunications Conference, Dallas, Texas, November 1976, pp. 16.2.1-16.2.7.

[NIC 76] "ARPANET Directory," Network Information Center, Stanford Research Institute, Menlo Park, California, July 1976.

[Norw 38] Norwine, A. C. and O. J. Murphy. "Characteristic Time Intervals in Telephonic Conversation," Bell System Technical Journal, Vol. 17, 1938, pp.281-291.

[Opde 74] Opderbeck, H. and L. Kleinrock. "The Influence of Control Procedures on the Performance of Packet Switched Networks," Conference Record, National Telecommunications Conference, San Diego, California, December 1974, pp. 810-817.

[Oppe 75] Oppenheim, A. V. and R. W. Schafer. Digital Signal Processing, Prentice-Hall, Englewood Cliffs, New Jersey, 1975.

[Orns 72] Ornstein, S. M., F. E. Heart, W. R. Crowther, H. K. Rising, S. B. Russell, and A. Michel. "The Terminal IMP for the ARPA Computer Network," AFIPS Conference Proceedings, Vol. 40, SJCC, Atlantic City, May 1972, pp. 243-254.

[Pick 76] Pickholtz, R. L. and C. McCoy, Jr. "Effects of a Priority Discipline in Routing of Packet-Switched Networks," IEEE Transactions on Communications, Vol. COM-24, No. 5, May 1976, pp. 506-516.

[Pouz 73] Pouzin, L. "Data Networks: Analysis and Design," Proceedings of the Third Data Communications Symposium, St. Petersburg, Florida, November 1973, pp. 80-87.

[Pouz 75] Pouzin, L. "Standards in Data Communications and Computer Networks," Proceedings of the Fourth Data Communications Symposium, Quebec City, Canada, October 1975, pp. 2.8-2.12.

[Prab 65] Prabhu, N. U. Queues and Inventories - A Study of Their Basic Stochastic Processes, Wiley, New York, 1965.

[Pric 72] Price, W. L. "Survey of NPL Simulation Studies 1968-72," National Physical Laboratory, Teddington, England, CLM 60, November 1972.

[Robe 70] Roberts, L. G. and B. G. Wessler. "Computer Network Development to Achieve Resource Sharing," AFIPS Conference Proceedings, Vol. 36, SJCC, Atlantic City, May 1970, pp. 543-549.

[Rubi 75] Rubin, I. "Message Path Delays in Packet-Switching Communication Networks," IEEE Transactions on Communications, Vol. COM-23, No. 2, February 1975, pp. 186-192.

[Sant 75] Santos, P. Personal communication, 1975.

[Schr 68] Schrage, L. "A Proof of the Optimality of the Shortest Remaining Processing Time Discipline," Operations Research, Vol. 16, 1968, pp. 687-690.