COMPUTER NETWORK RESEARCH

ADVANCED RESEARCH PROJECTS AGENCY
SEMIANNUAL TECHNICAL REPORT

December 31, 1971

Principal Investigator: Leonard Kleinrock

Co-Principal Investigators: Gerald Estrin
Michel Melkanoff
Richard R. Muntz

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

156

R

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) School of Engineering and Applied Science Computer Science Department 90024 405 Hilgard, University of California, Los Angeles	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP

3. REPORT TITLE

Computer Network Research

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

5. AUTHOR(S) (First name, middle initial, last name)

Leonard Kleinrock

6. REPORT DATE December 31, 197	7a. TOTAL NO. OF PAGES 115	7b. NO. OF REFS 79
8a. CONTRACT OR GRANT NO. b. PROJECT NO c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	

10. DISTRIBUTION STATEMENT

Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY

13. ABSTRACT

ARPA Semiannual Technical Report, July 1, 1971 through December 31, 1971.

DD FORM 1 NOV 65 1473 (PAGE 1)
S/N 0101-807-6801

UNCLASSIFIED
Security Classification

Sponsored by

ADVANCED RESEARCH PROJECTS AGENCY

COMPUTER NETWORK RESEARCH

SEMIANNUAL TECHNICAL REPORT

December 31, 1971

ARPA Contract DAHC-15-69-C-0285

ARPA Order No. 1380

Principal Investigator: Leonard Kleinrock

Co-Principal Investigators: Gerald Estrin
Michel Melkanoff
Richard R. Muntz

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

2

TABLE OF CONTENTS

Section		Page
	LIST OF FIGURES	iii
1	INTRODUCTION	1
2	ANALYTIC STUDIES	1
2.1	Modeling and Measurement of Time-Sharing Systems	1
2.2	Modeling and Simulation of Computer Networks	4
3	NETWORK MEASUREMENTS	7
3.1	Experiments Performed	8
3.2	Tools Built	8
3.3	Tools Specified	14
4	NETWORK AND SYSTEMS SOFTWARE	18
4.1	Network Progress	18
4.2	System Development	19
5	CONCLUSIONS	20
5.1	Publications and Presentations	20

APPENDIX A

Demultiplexing Considerations for Statistical Multiplexors, by Wesley W. Chu

APPENDIX B

Computer Communication Network Design--Experience with Theory and Practice, by Howard Frank, Robert E. Kahn, and Leonard Kleinrock

APPENDIX C

The Flow Deviation Method: An Approach to Store-and-Forward Communication Network Design, by L. Fratta, M. Gerla, and L. Kleinrock

APPENDIX D

An Anslysis of the Separation between Packets in a Store-and-Forward Network, by G. D. Cole and L. Kleinrock

APPENDIX E

Performance Measurements on the ARPA Computer Network, by Gerald D. Cole

LIST OF FIGURES

Figure		Page
1	Round Trip Times in Milliseconds vs Number of Interfering Senders	9
2	Total Throughput in IMP Words/12.8 sec vs Number of Interfering Senders	10
3	Adjusted Throughput in IMP Words/12.8 sec vs Number of Interfering Senders	11

COMPUTER NETWORK RESEARCH

Advanced Research Projects Agency
Semiannual Technical Report

December 31, 1971

1. INTRODUCTION

This semiannual technical report covers the period 1 July 1971 through 31 December 1971. Our activities have focused on modeling, analysis, measurements and systems software. In Section 2 below we discuss some of our results from modeling and measurement of time-shared computer systems; in this section also are presented results from the network modeling and simulation activities. In Section 3 we describe some of our network measurement activity, including proposed specifications for additional required tools. In Section 4 we describe some of our network and systems software development progress.

2. ANALYTIC STUDIES

2.1. Modeling and Measurement of Time-Sharing Systems

Our study of resource allocation strategies in time-shared computer systems has involved complementary analytic and measurement efforts. This section summarizes our recent work in these areas.

We are continuing research on multiple resource queueing models. A Master's thesis entitled "Comparison of Various Queueing Network Models" by F. Tobagi was completed in June 1971. This thesis concerned itself with the Gordon and Newell model of closed finite queueing systems which have been found appropriate as models for time-shared and multi-programmed computer systems. The idea was to take the recent work on diffusion approximations to queueing systems and to attempt to use these approximations in the Gordon and Newell Model. The reason for this attempt is that the Gordon and Newell solution becomes extremely difficult with medium size and large systems. The results of the thesis indicate that the diffusion approximation is indeed useful if it is applied to the critical (saturation) node in the network; as a by-product it was also found that a complete decomposition using methods similar to Jackson's model* is also an excellent candidate for approximating the Gordon and Newell systems.

In the Gordon and Newell Model all of the customers in the system are required to be statistically identical. For example, they must all have the same service time distribution at each resource. We are currently investigating networks of queues models in which each customer may exhibit distinct behavior.

Our work on time-sharing scheduling algorithms is continuing. R. Muntz will present a paper entitled "Waiting Time Distribution for Round-Robin Queueing Systems," which solves for the distribution of waiting times conditioned on service required for round-robin scheduling with finite quanta at the XXII Polytechnic Institute of Brooklyn Symposium on Computer-Communication Networks and Teletraffic. Previous results gave only the mean value of waiting time conditioned on service required.

Currently implemented and presently being documented is a time-shared scheduling algorithm simulator prepared by Walter Sheets for his Master's thesis. This simulation language permits one to synthesize a scheduling algorithm and then simulate the behavior of that algorithm with artificially

*"Networks of Waiting Lines," Operations Res., 5:518-521 (1957).

generated traffic. The program is extremely flexible and permits one to easily determine the gross behavior of scheduling algorithms prior to any intense analytical effort.

A measurement project on the SEX time-sharing system was begun by J. Wong as his Master's thesis. Part of the thesis was devoted to the problem of scheduling I/O requests to multiple disks on the same channel. An optimal algorithm was determined for such a configuration. The remainder of the thesis concerned memory allocation. The SEX system attempts to preload the working set of a process when it is to run. Measurements showed that interactive processes exhibited poor behavior since each interaction typically took very little execution time and required a change in the working set. System calls were implemented which allow a process to determine its own working set. The executive command interpreter was modified to set its working set as soon as the command to be executed is decoded. This technique significantly reduced page faulting for this process. We are currently investigating the general notion of allowing a process to advise the operating system of its resource requirements.

During this reporting period an extensive study of program behavior has begun. An interpreter has been constructed for the Sigma-7 which serves as a basic tool for these studies. H. Opderbeck wrote the interpreter and used it for program behavior studies for his Master's thesis. In addition to investigating the performances of known paging algorithms, he also defined a new class of paging algorithms and studied their performance. This new class of algorithms determines the number of pages to be allocated to a process on the basis of its current page fault rate. When a page fault occurs, the new page may replace a page from the process or be allocated as an additional page for the process. This decision is made on the basis of the recent page fault rate. If the page fault rate is low, the memory allocation of the process is periodically reduced. Measurements have shown that the algorithm tends to cause processes with very different addressing behavior to run with approximately the same efficiency. At the ACM/IEEE Second Symposium on Problems in the Optimization of Data Communications Systems (Palo Alto, Calif., Oct. 20-22, 1971), W. W. Chu presented a paper entitled "Demultiplexing Considerations for Statistical Multiplexors." In this work he considered the effect of statistical multiplexors on buffer occupancy and the implications of these effects on the design of time-shared scheduling algorithms. This paper is attached as Appendix A.

These studies are continuing, particularly in investigating the dynamic performance of memory allocation and paging algorithms. These studies are preliminary to the development of models of program behavior and studies of multi-programming system memory allocation strategies.

2.2. Modeling and Simulation of Computer Networks

A paper has been accepted for the Spring Joint Computer Conference 1972 authored by H. Frank, R. E. Kahn and L. Kleinrock entitled "Computer-Communication Network Design--Experience with Theory and Practice." This paper is attached as Appendix B and discusses the design experience for the ARPA Network. It is a rather general paper and contains some important insights, concepts and guidelines for design of networks.

A second paper authored by L. Fratta, M. Gerla and L. Kleinrock entitled "The Flow Deviation Method: An Approach to Store-and-Forward Communication Network Design" has been submitted to the Networks Journal and is attached as Appendix C. This paper discusses a heuristic procedure for obtaining the solution of the routing problem and of the capacity assignment problem in computer networks. This work is continuing and we currently have developed an exact method for solving these problems which seems to be reasonably efficient. It has been implemented in FORTRAN and can solve a 21-node net in about 30 seconds cpu time on the 360/91 with an accuracy of 10^{-4} in the minimum delay computation. We are presently attempting to reduce this computation time by improving a frequently used optimization routine; in particular, we are comparing the performance of a gradient method, a parallel tangent method, and a Newton-Raphson method. Our concentration on optimally selecting capacities from a discrete set is continuing and we are currently working on a suboptimum algorithm in this regard.

A third paper by G. Cole and L. Kleinrock entitled "An Analysis of the Separation Between Packets in a Store-and-Forward Network" has been accepted for the MRI International Symposium XXII on Computer-Communications Networks and Teletraffic, April 4-6, 1972. In this paper an analysis is carried out for the inter-packet spacing which may be expected after a multi-packet message travels through a number of nodes on the way to its destination. This computation is important in attempting to calculate the expected time that a reassembly buffer will be occupied while awaiting reception of a multi-packet message. The paper is attached as Appendix D.

In conducting research for his Ph.D., Gary Fultz has carried out the specification, performance verification and documentation of various store-and-forward computer network <u>routing algorithms</u> during this reporting period. Three major classifications of routing algorithms have been investigated.

a. <u>Deterministic routing strategies</u>. Deterministic routing algorithms compute routes based upon given deterministic decision rules and produce loop-free routes. The procedures investigated assumed multiple priority message traffic, a given network traffic matrix, a known network topology and a convex network cost function (generally average message delay). Both optimal and suboptimal techniques have been examined which attempt to find sets of routes (and hence the network traffic flow) which minimizes the network cost function.

A number of suboptimal algorithms were formulated which gave performance within 1% of that found with the optimal technique, and required less programming effort and computation time than the optimal technique. The suboptimal methods are useful in the abstract design of low-cost networks where the routing algorithm must be repeatedly applied during the optimization of the network structure.

The results of the optimal technique are useful as both a bound on the performance of the suboptimal deterministic routing procedures and stochastic routing procedures and to gain understanding about the decomposition of optimum flows in the network in terms of elementary flows (a special form of basis for the description of any arbitrary network traffic flow). Experimental results have shown that it is relatively easy to design heuristic algorithm structures which are computationally simple and have good performance.

b. Stochastic routing strategies. Stochastic routing algorithms operate as probabilistic decision rules and utilize a certain amount of information concerning the delays within the network. A number of algorithms have been designed and their performance has been found using simulation techniques. This class of algorithms is particularly applicable to operational networks.

Of the algorithms simulated, most of them gave rise to reasonable values of message delay as the network traffic load was varied; however, one specific algorithm showed superior performance when transmission link failures were introduced into the simulation.

Further study is required to assess the actual complexity of these algorithms when implemented in an operational network.

c. Flow control strategies. Flow control routing strategies are perhaps a combination of both deterministic and stochastic techniques, but in addition take advantage of the operational network protocol in their operation. The major objective of these algorithms is to provide small delay times for highly interactive messages, and, at the same time, provide high bandwidth for long messages or for file transfer between remote computers.

Most of the effort to date in this area has been concerned with a review of what other researchers have considered, and special attention has been placed on proposed techniques by BBN. The major effort has been focused upon techniques which alleviate high congestion points within the net and efficient storage allocation procedures within the nodes to insure the acceptance of arriving messages.

It is felt that flow control techniques are an important part of any operational network doctrine and must be included in the design of the basic network routing strategy for efficient network operation.

3. NETWORK MEASUREMENTS

This section covers our activity on network measurement over the current reporting period. Our activity in this field has been increased significantly and some measurements have already been taken; many more will be taken in the immediate future.

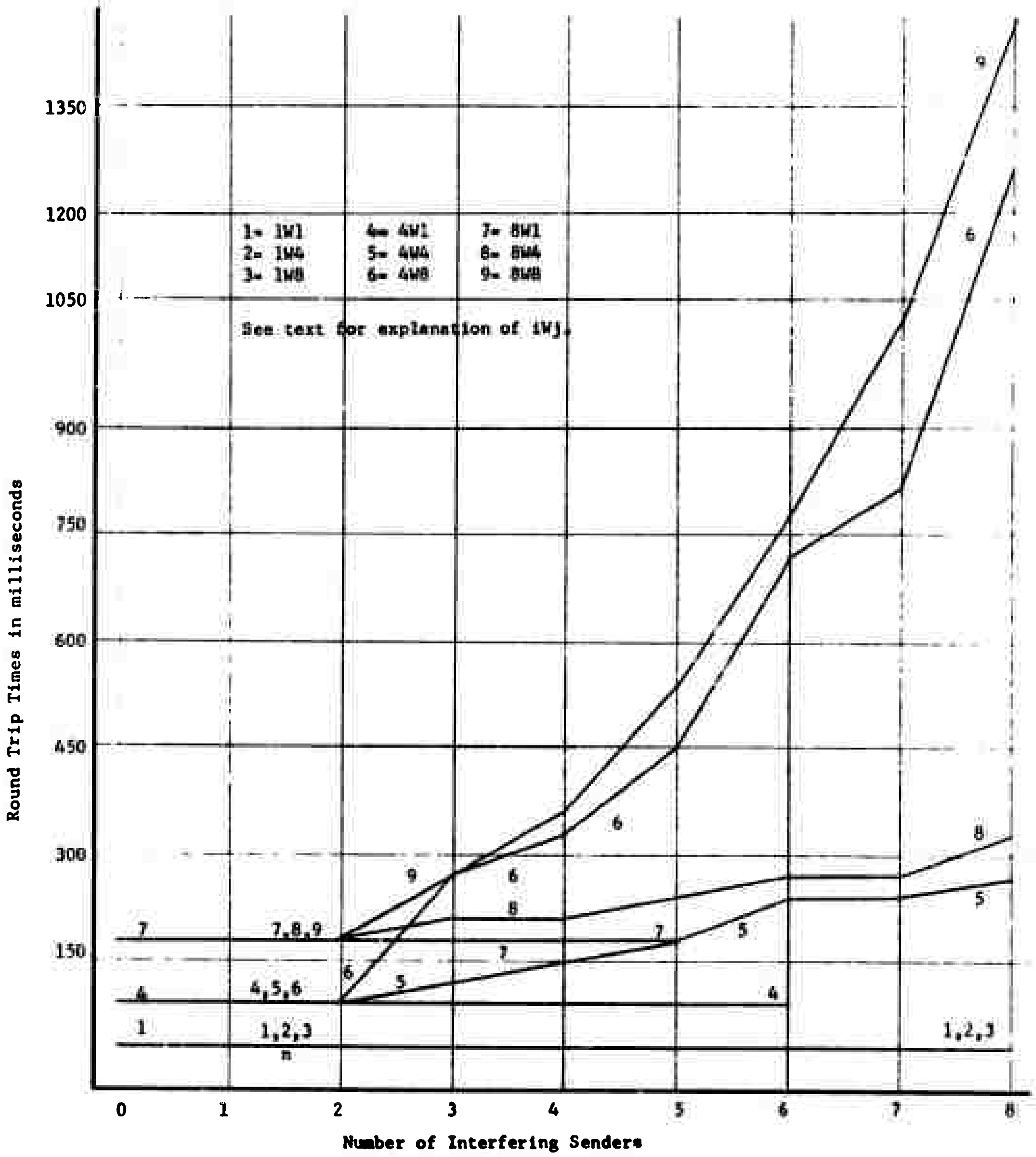
3.1. Experiments Performed

a. A set of measurements of the subnet was performed to determine the behavior of an IMP under a heavy load. The measurements were performed by using the message generators in several IMP's sending similar length messages all to one site's discard fake Host. One of the sending sites was monitored using cumulative statistics messages. Three quantities were calculated: average round trip times (message - RFNM time), average throughput (including retransmissions) and average adjusted throughput (average number of round trips times number of words/message). These three quantities are plotted as a function of the number of interfering senders in Figures 1, 2 and 3. The knee of each curve (where all senders are sending multipacket messages) indicates that there is room for a maximum of three multipacket messages in the reassembly buffer space. This is consistent with what we know about the IMP systems.

b. An experiment using the trace package in the IMP's was used successfully in helping UCLA's Campus Computing Network debug the network hardware/software. They had an internal trace and compared it to the trace we get from the IMP and in doing so were able to isolate their bug.

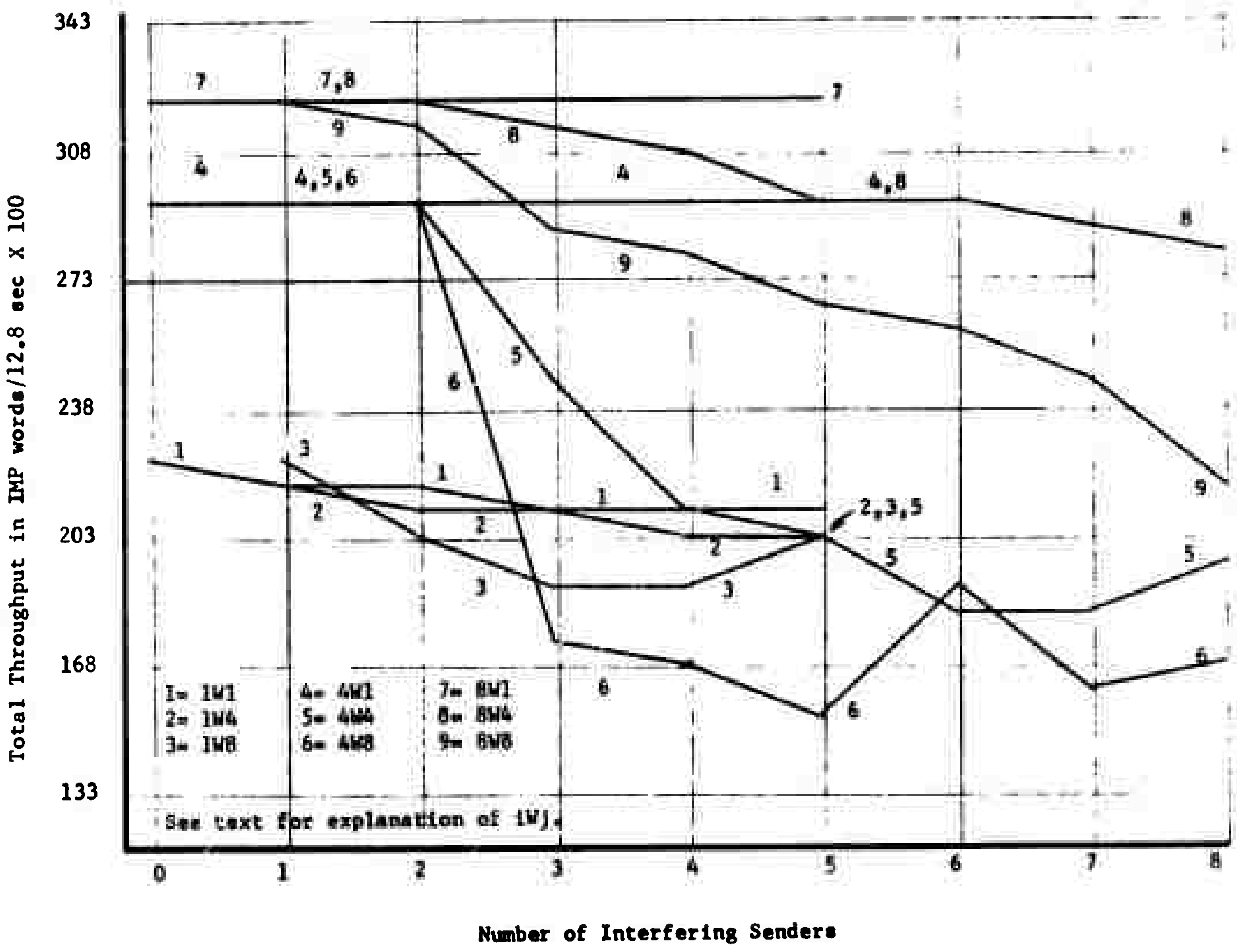
3.2. Tools Built

At the beginning of this reporting period a set of assembly language programs existed for setting IMP parameters and collecting, in a file, messages from the IMP statistics package. Also two FORTRAN compatible subroutines existed for reading these files so that reduction of the data could be performed by a FORTRAN program. (Indeed, some reduction programs were written too.)



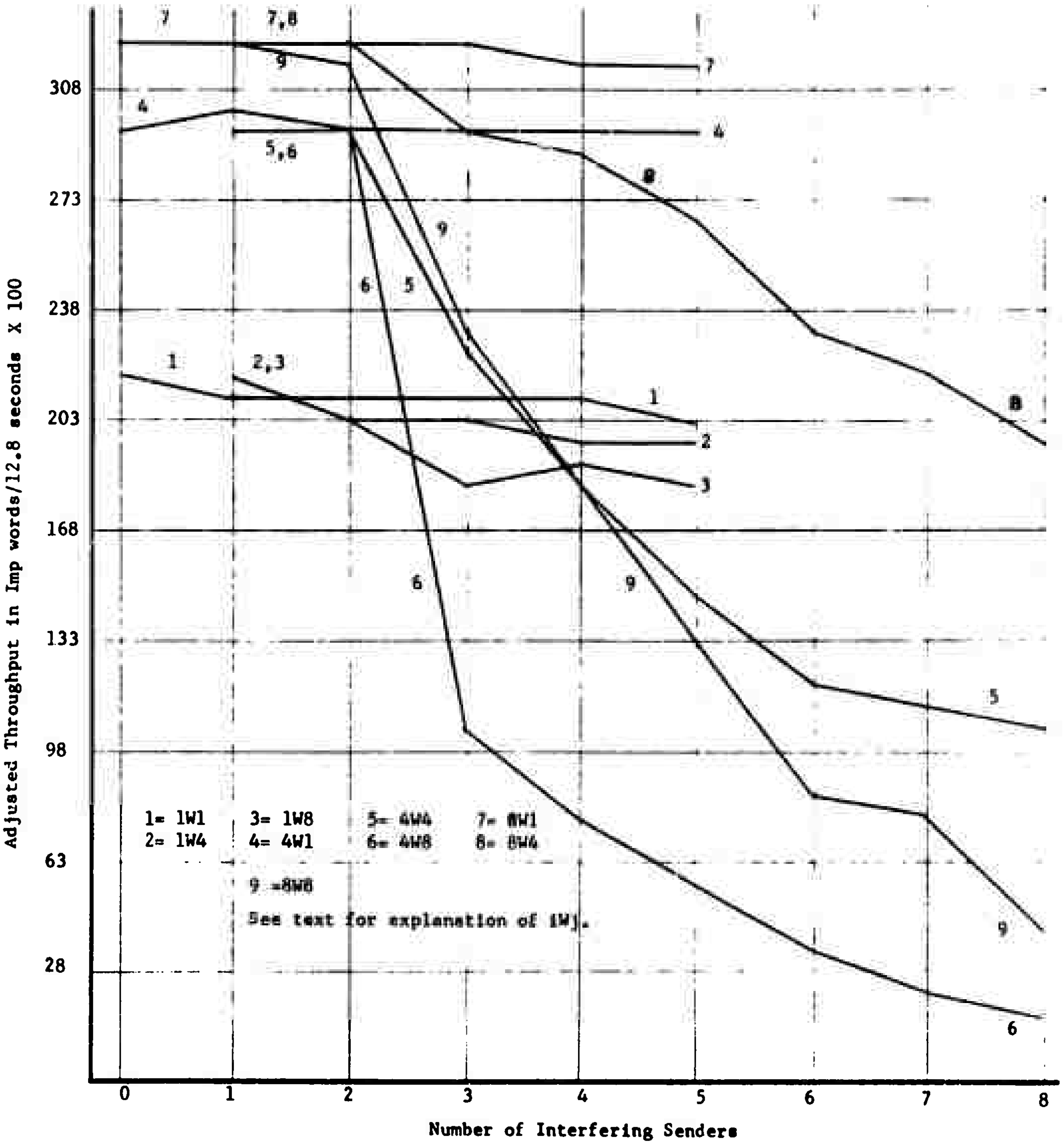
Round Trip Times in milliseconds vs Number of Interfering Senders

Figure 1



Total Throughput in IMP words/12.8 sec vs Number of Interfering Senders

Figure 2



Adjusted Throughput in IMP words/12.8 sec vs Number of Interfering Senders

Figure 3

NOTES ON THE "INTERFERING TRAFFIC" EXPERIMENT

In the three figures, the expression "iWj" is defined to mean:

> The measured IMP (UCLA) is sending i-packet messages and the interfering senders are sending j-packet messages. All messages are destined for RAND's IMP discard.

Total throughput is measured from accumulated statistics and represents an average of 27 samples. Accumulated statistics produce a count of the total number of words sent from UCLA to RAND.

Adjusted throughput is computed on the basis of total data words sent, and thus excludes leader and other control bits.

Round trip times represent the time it takes for a message to be delivered from UCLA to RAND and for the associated RFNM to be returned.

As the number of interfering sites increases, a new IMP traffic generator starts to send messages to RAND's IMP discard program. The order of addition of new interfering sites is fixed, with the nth new interfering sender associated with a fixed site as follows:

Total Number of Interfering Senders		Next Additional Site Number
0	→	1 (UCLA)
1	→	5 (BBN)
2	→	8 (SDC)
3	→	11 (Stanford)
4	→	9 (Harvard)
5	→	4 (Utah)
6	→	16 (NASA-Ames)
7	→	6 (MIT)
8	→	12 (Illinois)

SOME OBSERVATIONS ON THE RESULTS

1. The curve labeled "3" in adjusted throughput increases at 4 interfering senders. We conjecture that since UCLA is sending 1-packet messages and the interfering senders are sending 8-packet messages, the multipacket packet senders interfere so severely with one another while contending for reassembly buffer space, that more single-packet messages can be sent to discard. This may be a false conjecture, since the same effect is not seen when the multipackets are of length 4.

2. Comparing curves 6 and 9 in adjusted throughput, it appears that against the same interference (i.e. 8-packet messages), it is better to send longer messages (i.e. 8 instead of 4) unless you can send single-packet messages (i.e. curve 3).

3. When UCLA is sending multipacket messages and interfering senders are sending multipacket messages, delays jump at 3 interfering senders because there is buffer space (24 packets) for only 3 messages.

4. When interfering sites are sending single packets while UCLA sends multipackets, there isn't much throughput loss due to interference.

5. Similarly, if UCLA sends single-packet messages, multipacket messages do not interfere very much (curves 1, 2, 3).

The above set of programs were augmented by several FORTRAN compatible subroutines for setting the state of IMP parameters, so that now an experiment may be run under the control of a FORTRAN program running under the time sharing system. (In fact, the first experiment discussed in 3.1 was performed in this fashion.)

3.3. Tools Specified

A large part of our effort during this reporting period was to identify needed tools and to begin careful specifications of each. In so doing, we have categorized our existing tools as well as the new required ones.

Among the existing tools, we have the following:

1. Net statistics gatherer

   a. enables/disables IMP background programs via Host originated messages to parameter change
   b. collects resulting statistics messages sent to Host by background programs (cumulative statistics, traces, snapshots)
   c. runs under SEX (a less powerful version, one requiring IMP tty interaction for parameter changes, runs under RAD 75)

2. Net statistics formatter

   a. processes statistics message file(s) collected by (1)
   b. generates tabular reports
   c. runs under SEX (also RAD 75 as a part of the Standalone measurement program)

3. Host traffic generator

   a. generates RFNM driven, fixed or random length messages over one or more links
   b. generates random interval, fixed or random length messages over one or more links
   c. runs under RAD 75 as part of the Standalone Network measurement program

4. IMP traffic generator

   a. generates RFNM driven, fixed length messages on one link
   b. runs as background program in IMP

5. Network status

   a. shows local connection status (open, closed, wait, etc.)
   b. shows what Hosts are up (Hosts from which the NCP received a reset or a reset reply)
   c. runs under SEX

6. IMP facilities

   a. DEBUG - allows examination of core locations
   b. PARAMETER-CHANGE - controls background programs
   c. DISCARD - returns RFNM and throws message away
   d. IMP tty - usable for interaction with DEBUG & PARAMETER CHANGE if HOST is dead

Among the <u>projected tools</u> are:

1. Network survey (almost operational now)

   a. reports Host status - (NCP up/down)
   b. reports connection status - what other Host communicating with Host
   c. reports Host resource status - what software is available

2. Remote Host traffic generator

   a. generates RFNM driven or random interval, fixed or random length traffic on one or more links
   b. runs at remote Host under time-sharing, as a part of a Network handler, or standalone. Controllable from UCLA.

3. Remote Host traffic reflector

   a. receives message
   b. time tags it
   c. sends it back or cycles address sequence

4. Remote Host discard

   a. receives message
   b. time tags it
   c. saves times for later processing at remote Host or transmission back to us

5. Remote Host statistics gatherer

   a. performs functions of our Net statistics gatherer

6. NCP statistics hooks

   a. reports amount of traffic over each connection

7. Ability of IMP to change message lengths to simulate higher and lower speed lines.

So far we have given in SPADE Design Note #94 the careful specification for Host artificial traffic generator. It is as follows:

SPADE Design Note #94

V. Cerf
L. Nelson
A. Ollikainen
20 December 71

Specifications for a Host artificial traffic generator essential features:

1. Transmit/receive Network messages utilizing level one protocol (Host-IMP)

2. Dynamic modification of any or all of the following parameters according to some distribution or density function:

   a. period of message (inter-arrival time)
   b. length of message
   c. destination of message
   d. link # of message
   e. message type

Functions necessary for implementation of foregoing features (a description of how UCLA's traffic generator works):

1. Programmable clock routine, driven by an event list, schedules next message to go on each active link. Minimum required resolution of the clock routine is the minimum RFNM return time on a given link.

2. Interrupt routine observes the state of each link and decides whether to schedule a new message on the future event list depending on:

   a. when RFNM's are received
   b. when clock interrupt occurs

3. Lists of destinations, periods, lengths and message types.

4. Accumulation of actual traffic to each destination.

Experiment traffic parameter specification (an artificial traffic generator should be general enough to allow the following types of traffic specifications):

1. Destination specific

   a. for each destination--specify inter-arrival density function
   b. for each destination--specify length density function

(Requires 2N density functions for N destinations and does not specify message types, contents, etc.)

2. Probabilistic destination

   a. specify density function of traffic fraction destined for each Host
   b. same as in (1) (2 density functions: inter-arrival time and length)

(This requires only 3 density functions. Again message type is left out.)

3. Message types

   a. all same
   b. message type for each destination
   c. message type density function
   d. message type density function for each destination

(We guess that (a) and (b) are sufficient for most experiments.)

4. Experiment duration

   a. specify duration initially
   b. send start/stop messages to generator(s)

Experimental feedback:

1. For each destination, accumulate

   a. total bits sent
   b. total messages
   c. experiment time

2. Record time waiting after IMP says "not yet"

3. Record number of "link table full" messages received

A Ph.D. thesis by Gerald Cole entitled "Computer Network Measurements: Techniques and Experiments" was completed in June 1971. The results of that study have been used as points of departure for our recent measurement activities and has formed the foundation for this work. He presented the paper, "Performance Measurements on the ARPA Computer Network," at the ACM/IEEE Second Symposium on Problems in the Optimization of Data Communications Systems, Palo Alto, Calif., Oct. 20-22, 1971, and is included here as Appendix E.

As we have mentioned above, measurements have begun, new tools have been partly specified, and many new ones have been identified. We are in the process of completing the specifications on this set of tools, and we are finding cooperative Hosts who will permit various of these tools to be implemented at their sites for the purpose of our extensive measurement program which is now well under way.

4. NETWORK AND SYSTEMS SOFTWARE

This section covers work done by the SPADE Group which is under the leadership of Jon Postel. This group is responsible for maintaining and extending the SEX time-sharing system and the development of network software.

4.1. Network Progress

Jon Postel has been appointed a Network Facilitator by ARPA. In this capacity he is responsible for aiding in the development of new applications for the network, new protocols, and generally being of assistance to members of the ARPA Network community. He has, for example, consulted with the programmers at Tinker AFB on protocol and program construction and the Center for Computer-Based Behavioral Studies at UCLA on the design considerations and cost of interfacing with the ARPANET. He regularly attends Network Working Group meetings, has led discussion groups and has been very active, particularly in the defining of network protocol.

Existing network software (NCP, TELNET, LOGGER) have been kept up to date with current network specifications. Some additional user features have been added to these processes which are not network standard. For example, the NCP will automatically time-out a connection. We have also been active in the development of software for use of network facilities. Primary is RJSNET, a program which provides the user side of the UCLA-CCN 360/91 Remote Job Entry Service. Another example is FXFER, a program which provides access to the UCSB 360/75 Simple Minded File Service. This service has proved quite useful in providing quickly accessible file backup. We are currently cooperating with RAND and UCSB in the development of a Data Reconfiguration Service Compiler which will provide convenient means for reformatting files.

4.2. System Development

The SEX time-sharing system is now available for standard user service 70 hours per week. Improvements continue to be made to the system, such as an improved text file manipulation service and extending of graphics facilities. Much effort has gone into providing measurement routines. We are currently implementing a new disk scheduler to cope with the problem of multiple disks on one channel. We are also experimenting with new resource allocation algorithms. This work is closely tied with our research efforts in computer system modeling.

5. CONCLUSIONS

Our activities then have concentrated on modeling, analysis and measurement both of time-shared computer systems and of computer networks. The ARPA Network of course forms the basic model in our network studies. Our network and systems software has progressed very nicely. During the next semiannual period we expect to have specified additional measurement tools and to have exercised them in measuring and monitoring network activity and performance.

5.1. Publications and Presentations

During this period the following papers were published or submitted for publication:

Cantor, D., "On Non-Blocking Switching Networks," to be published in Networks.

Chu, W. W., "Demultiplexing Considerations for Statistical Multiplexors," Proc. of the ACM/IEEE Second Symposium on Problems in the Optimization of Data Communications Systems," Palo Alto, Calif. Oct. 20-22, 1971.

Cole, G. D., "Performance Measurements on the ARPA Computer Network," Proc. of the ACM/IEEE Second Symposium on Problems in the Optimization of Data Communications Systems," Palo Alto, Calif., Oct. 20-22, 1971.

Fratta, L., Gerla, M., and Kleinrock, L., "The Flow Deviation Method: An Approach to Store-and-Forward Communication Network Design," to be published in Networks.

Fultz, G., and Kleinrock, L., "Adaptive Routing Techniques for Store-and-Forward Computer-Communication Networks," Proc. of the IEEE Convention on Communications, Montreal, Canada, June 14-16, 1971, pp. 39-1 to 39-8.

Kleinrock, L., Muntz, and Hsu, J., "Tight Bounds on the Average Response Time for Time-Shared Computer Systems," Proc. of the 1971 International Federation for Information Processing, Ljubljana, Yugoslavia, Aug. 23-28, 1971, TA2, pp. 50-58.

Kleinrock, L., "A Selected Menu of Analytic Results for Time-Shared Computer Systems," to be published in Elektronische Rechenanlagen.

Kleinrock, L., and Muntz, R. R., "Processor-Sharing Models of Mixed Scheduling Disciplines for Time-Shared Systems," to be published in the JACM.

The following presentations were given:

Kleinrock, L., "Some Computer Network Models," Computer Science Department, University of London, June 30, 1971.

Kleinrock, L., "The ARPA Computer Network," IBM Research Laboratories, Zurich, Switzerland, July 12, 1971.

Kleinrock, L., Chairman of the session "System Modeling, Evaluation, Optimization," and banquet speaker at the Second Symposium on Problems in the Optimization of Data Communications Systems, Palo Alto, Calif., Oct. 20-22, 1971.

Muntz, R. R., "Operating Systems--A User's View," presented at the Association for Computing Machinery Professional Development Seminar, August 1971.

Muntz, R. R., and Wong, J., "Process-to-Scheduler Communication to Aid Memory Management," presented at the Workshop on System Performance Evaluation, Argonne National Laboratories, Oct. 6-7, 1971.

Muntz, R. R., and Wong J., "A Solution to the Multi-shaft Problem," XDS Users' Group Meeting, Las Vegas, Nev., Nov. 18-20, 1971.

APPENDIX A

DEMULTIPLEXING CONSIDERATIONS FOR STATISTICAL MULTIPLEXORS

by Wesley W. Chu

Presented at the ACM/IEEE Second Symposium on Problems in the Optimization of Data Communications Systems, Palo Alto, Calif., Oct. 20-22, 1971.

DEMULTIPLEXING CONSIDERATIONS FOR STATISTICAL MULTIPLEXORS*

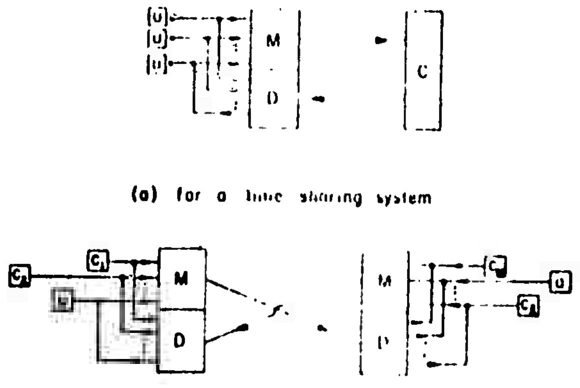
Wesley W. Chu
Computer Science Department
University of California
Los Angeles, California 90024

Abstract

Demultiplexing serves as an important function for statistical multiplexors. Its purpose is to reassemble the received message and distribute it to the appropriate destination. The demultiplexing buffer can be modeled as a finite waiting room queuing model with batch Poisson arrivals and multiple distinct constant servers. Simulation method is used to study the buffer behavior for the uniform, linear, step, and geometric traffic destination functions. The relationships among buffer overflow probability, buffer size, traffic intensity, average message length, and message destination are presented in graphs to provide a guide in the design of demultiplexing buffer. Simulation results reveal that buffer input message which have short average message length and uniform traffic destination yield best buffer behavior. Thus, in planning the CPU scheduling algorithm and in selecting the demultiplexing output rates, the computer communications system that uses the statistical multiplexing technique should also consider the desired output statistics to achieve optimal demultiplexing performance.

I. Introduction

To increase information processing capability and to share computer resources, remotely located computers and/or terminals are connected together by communication links. Many such computer communication systems as time sharing systems and distributed computer systems are already in existence and in operation. The additional cost to such systems is the communications cost. In many cases, the communication cost is a significant portion of the total operating cost. To increase channel utilization and reduce communication cost, asynchronous time division multiplexing has been proposed for data communications.[1] The design considerations of such a multiplexing (statistical multiplexing) system has been reported in recent literature.[2-6] Here, we study the design considerations of the asynchronous time division demultiplexing system which is an integral part of the statistical multiplexor design. In the multiplexing case, the outputs from the system are going to a single destination, for example, a computer, while the outputs from the demultiplexing system are going to many destinations, such as different users and/or computers as shown in Fig. 1.



(a) for a time sharing system

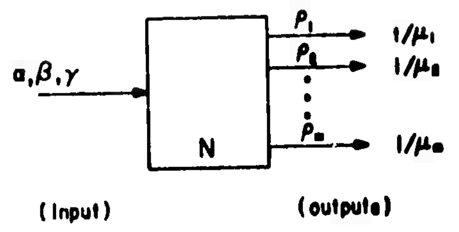
(b) for a distributed computer system

U, user terminal
D, statistical demultiplexor
M, statistical multiplexor
C, computer
Ci, i^th computer

Fig. 1 Statistical Multiplexing Systems

The demultiplexor may consist of a buffer and a switching circuit. The input traffic to the buffer consists of strings of characters (bursts). The switching circuit distributes the output from the buffer to the appropriate destinations according to the designation in the message address. Thus, the demultiplexor performance is strongly influenced by the buffer behavior which depends on the buffer input traffic and its destination characteristics. A queuing model with a finite waiting room, batch Poisson arrivals, and multiple *distinct* constant outputs (Fig. 2) is used to study the buffer behavior.

*This research was supported by the U.S. Office of Naval Research, Research Program Office, Contract No. N00014-69-A-0200-4027, NR 048-129 and the Advanced Research Projects Agency of the Department of Defense Contract No. DAHC 15-69-C-0285.



α	message inter-arrival time
β	average message length
γ	destination function
μ_i	transmission rate for the ith destination
ρ_i	traffic intensity for the ith destination
N	buffer size

Fig. 2 A Model for Demultiplexing Buffer

The complexity of the demultiplexing buffer makes exact analysis of such a model mathematically intractable. Therefore, computer simulation has been used to study the relationships among message destination function, average traffic level, message burst length, overflow probability (the fraction of the total number of messages rejected by the buffer), and buffer size. These relationships are portrayed in graphs and are important in designing demultiplexing systems and store-and-forward computer networks.

II. Analysis of Demultiplexing Buffer

The input messages to the buffer can be represented by three parameters: message arrivals, message lengths, and message destinations. These parameters are intimately related to the buffer behavior. In order to describe the input messages arriving at the buffer, three random number generators are used: α which corresponds to message inter-arrival times, β which corresponds to the number of characters in the message, and γ which corresponds to the destination of the message. In the simulation model, the message inter-arrival times α are assumed to be exponentially distributed,* and the message lengths β are assumed to be geometrically distributed. The destination distribution is transformed from the destination function as discussed in the following:

*In the distributed computer network as shown in Fig. 1b, the traffic input to the demultiplexor is the output from the multiplexor. In some cases, the traffic output from the multiplexor can be approximated as Poisson distributed. Should the multiplexor output become very different from Poisson,[7] then the actual message inter-arrival-time statistics should be used in the simulation.

The destination function describes traffic intensities for the set of m destinations; that is,

$$f_d(i) = \rho_i \qquad i = 1, 2, \ldots, m \qquad (1)$$

where ρ_i is the traffic intensity for the ith destination.

The message destination function for a given set of users depends on their applications and should be derived from measured statistics.

In order to perform random sampling on traffic destination, we need to transform $f_d(i)$ into a message destination distribution, $f_\gamma(i)$. This transformation can be performed by normalizing Eq. (1) to a probability destination function; that is,

$$f_\gamma(i) = f_d(i) \Big/ \sum_{j=1}^{m} \rho_j \qquad i = 1, 2, \ldots, m \qquad (2)$$

Thus,

$$\sum_{i=1}^{m} f_\gamma(i) = 1$$

where m is the total number of message destinations.

A set of random numbers, $\{\xi_\alpha, \xi_\beta, \xi_\gamma\}$, corresponding to random variables α, β, and γ are generated to represent a message arriving at the demultiplexing system. When a message arrives at the buffer, two operations take place: First, the status of the designated facility is interrogated. If the facility is busy and the buffer is not full, the burst enters the buffer and is concatenated with the queue of characters waiting for that facility. If the facility is idle, and if the buffer is not full, the first character of the burst is sent to the facility while the remaining characters enter into the buffer and output at each subsequent clock time. Second, the contents of the register that keep track of the total length of the buffer is updated. Because of distinct destinations, the total volume of output from the buffer varies with α, β, and γ. The output from the demultiplexing buffer depends on both the number of characters in the buffer and their destinations. The simulation program keeps a record of the number of characters in the buffer at the beginning of each message service interval. When the length of an arriving message exceeds the unoccupied storage space of the buffer, then a buffer overflow event has occurred. The frequency of occurrence of such an overflow event gives the estimate of buffer overflow probability, P_{of}. A flow chart of the simulation program is shown in Fig. 3.

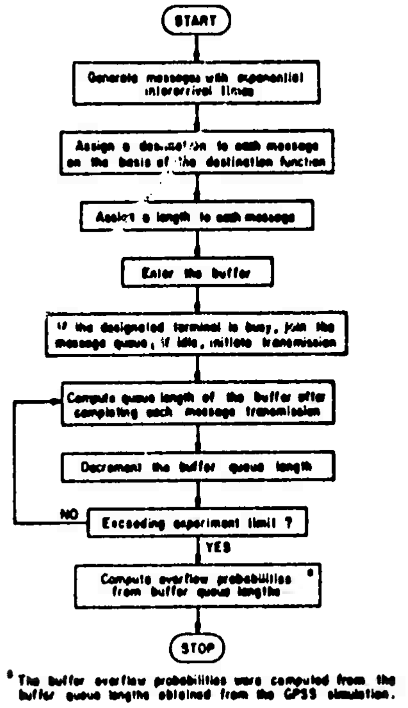

START
Generate messages with exponential interarrival times
Assign a destination to each message on the basis of the destination function
Assign a length to each message
Enter the buffer
If the designated terminal is busy, join the message queue; if idle, initiate transmission
Compute queue length of the buffer after completing each message transmission
Decrement the buffer queue length
NO
Exceeding experiment limit ?
YES
Compute overflow probabilities from buffer queue lengths *
STOP
* The buffer overflow probabilities were computed from the buffer queue lengths obtained from the GPSS simulation.

Fig. 3 A Logic Flow Chart for the Buffer Simulation Model

One of the most important parameters that describes the traffic congestion of the demultiplexing system is traffic intensity. Since the output messages from the buffer are sent to various destinations, the traffic intensity of each destination would be different and would equal

$$\rho_i = \frac{\lambda_i \bar{\ell}_i}{\mu_i} \tag{3}$$

where

λ_i = message arrival rate for the ith destinations

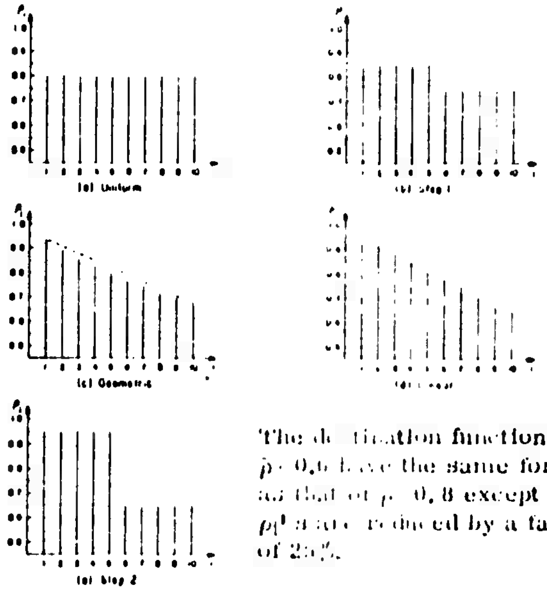
$\bar{\ell}_i$ = average message length for the ith destination

μ_i = transmission rate for the ith destination.

The average traffic level, $\bar{\rho}$, for the demultiplexing system is the average of the traffic intensities of the m destinations; that is,

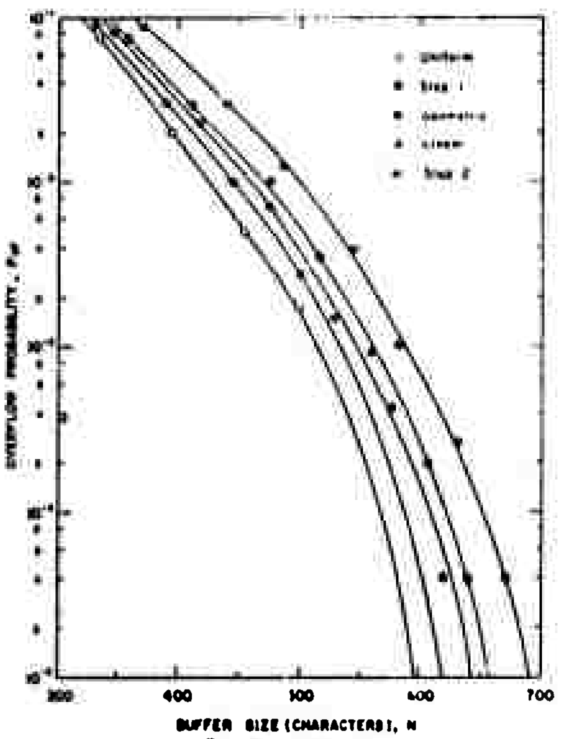
$$\bar{\rho} = \frac{1}{m} \sum_{i=1}^{m} \rho_i \tag{4}$$

To study the effect of destination distributions on buffer behavior, five types of destination functions are used in the simulation model as shown in Fig. 4. The relationships among P_{of}, buffer sizes, and destination distributions for selected $\bar{\ell}$'s and ρ's are obtained from simulation and portrayed in Figs. 5 and 6.



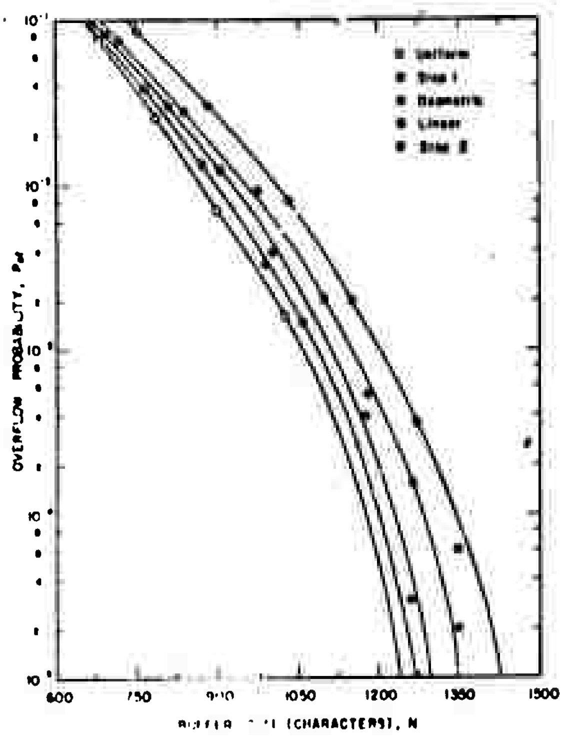
The destination functions for $\bar{\rho}$ = 0.6 have the same forms as that of $\bar{\rho}$ = 0.8 except the ρ_i's are reduced by a factor of 25%.

Fig. 4 Various Types of Destination Functions

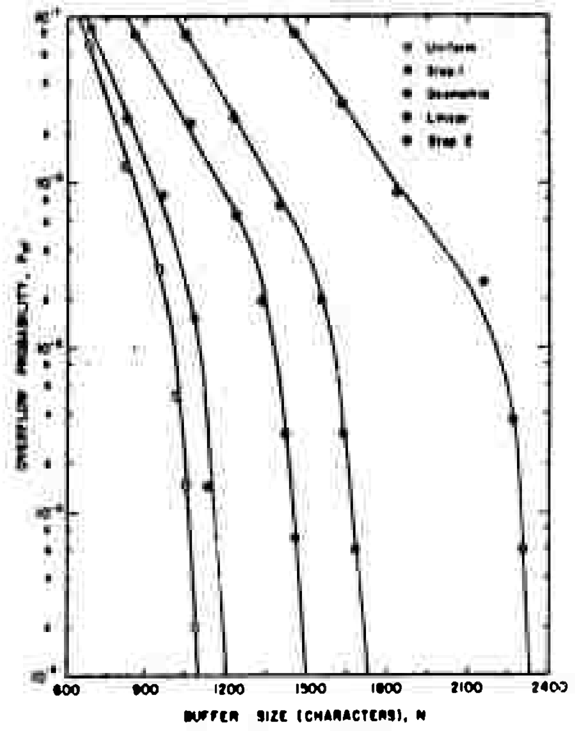


5a) $\bar{\ell}$ = 10 Characters

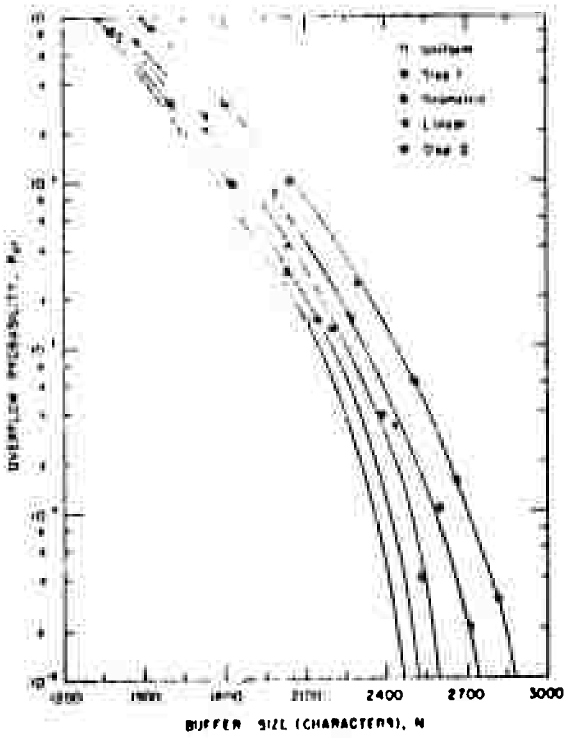
29



5b) $\bar{\ell}$ = 20 Characters

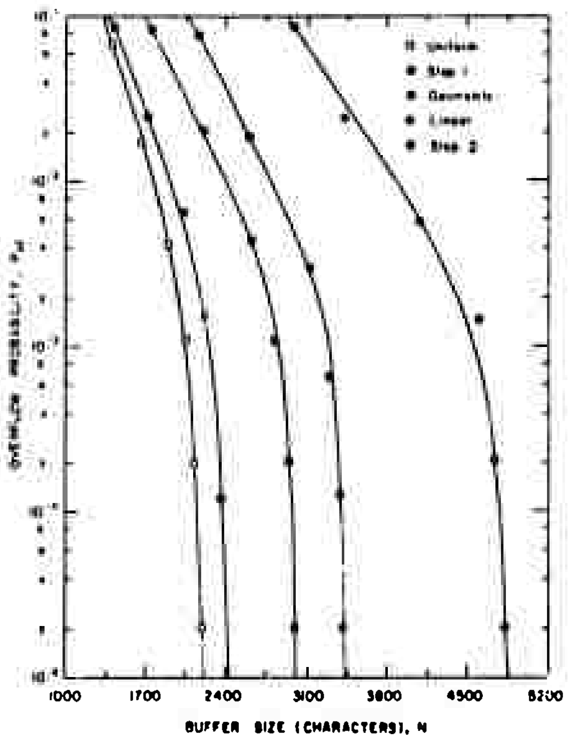


6a) $\bar{\ell}$ = 10 Characters

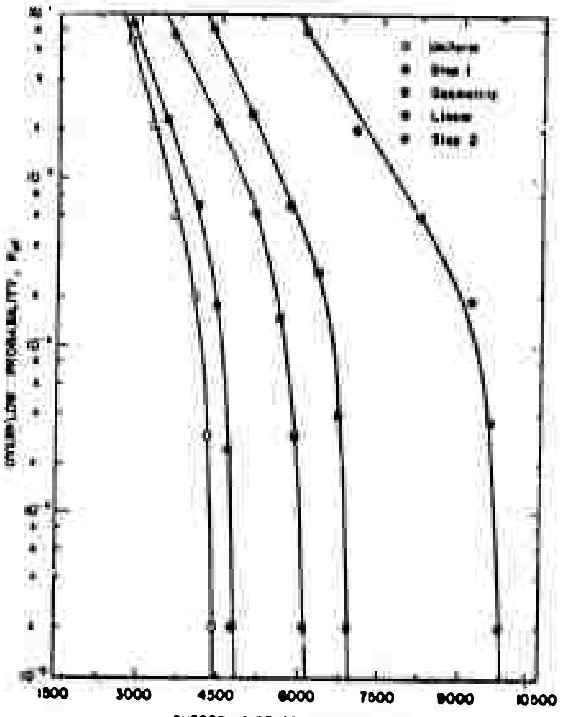


5c) $\bar{\ell}$ = 40 Characters

Fig. 5 Buffer Overflow Probabilities vs Buffer Size for Average Traffic Level, $\bar{\rho}$ = 0.6



6b) $\bar{\ell}$ = 20 Characters



6e) $\bar{\ell}$ = 40 Characters

Fig. 6 Buffer Overflow Probabilities vs Buffer Size for Average Traffic Level, $\bar{\rho}$ = 0.8

The expected queuing delay due to buffering during demultiplexing is another important parameter in the design consideration of statistical multiplexors. Since most systems allow a very low overflow probability, the expected waiting time due to buffering can be approximated by that of an infinite waiting room queuing model with Poisson arrivals and geometric service time. The expected waiting time w_i for message sending to the ith destination is

$$W_i = \frac{\lambda_i E(X_i^2)}{2(1-\rho_i)} = \frac{\rho_i (2\bar{\ell}_i - 1)}{2(1-\rho_i)}$$

character-service-times (5)

where $E(X_i^2)$ is the second moment of the message length X_i for the ith destination.

Since the average arrival rate and the average message length for each destination are different, the expected queuing delay due to demultiplexing for each destination is also different and should be computed from its associated traffic intensity and average message length.

III. Discussion of Results

The relationships between buffer size and buffer overflow probability for selected average traffic levels, average burst lengths, and traffic destination functions are portrayed in Figs. 5 and 6. For a given average message length, and desired level of overflow probability, the required buffer size increases as the average traffic level increases. Further, the required buffer size varies drastically with the message destination functions. Comparing the five types of traffic destination functions used in our simulation, for a given average traffic level, the uniform destination function required the smallest buffer size, and the step 2 destination function required the largest buffer size. This is because both buffer size and queuing delay increase exponentially with ρ_i. Thus, for a given average traffic level, different types of destination functions yield different buffer behavior. For example, if one of the destinations is heavily loaded, the average traffic level of the system could be very low, but an extremely large buffer is needed. Further, the expected queuing delay for message sending to that heavily loaded destination would be very long. We notice that the effect of destination functions on buffer behavior increases as average traffic level increases.

Simulation results of buffer behavior shed light on the relationships between the demultiplexor traffic input and the demultiplexing system performance. More specifically, the results show that to minimize the size of the demultiplexing buffer and queuing delay due to buffering, we should schedule the inputs to the demultiplexor buffer approximately equally. This can be achieved by controlling the volume of input traffic to the buffer for various destinations, selecting the output transmission rate of the buffer, or both. In a time-sharing system, input traffic to the demultiplexor buffer is governed by the computer scheduling algorithm, such as the size of CPU quantum time. If we know the relationship* between the CPU quantum time and the volume of the CPU outputs, then a variable quantum time scheduling algorithm would be effective to schedule an equal amount of output traffic to various destinations. The buffer output rate can also be changed to adjust the traffic intensity. To achieve a uniform traffic destination distribution, we should assign high transmission rate lines to those destinations that have high volumes of traffic.

In our simulation model, ten buffer output destinations are used. Since the input traffic to the buffer is random, the ten output destinations

*This relationship depends on the computer system and program behavior in question, and could be obtained via measurement or simulation.

can also be viewed as ten groups of outputs provided the outputs in each group have approximately the same value of traffic intensity.* When a demultiplexing buffer has more than ten outputs, we could group them into ten groups according to their traffic intensities, and then select the appropriate graphs to estimate its buffer behavior.

The simulation was performed by using the GPSS program on the IBM 360/91 at UCLA. The computing time required for the simulation depends largely on the sample size and the number of destinations used. For a given α, β, and a destination distribution, the computing time required for an experiment (which represents a curve on the graph) of 10^5 samples and ten destinations is about five minutes. As is common in many stochastic experiments, their results are sensitive with sample sizes. A larger sample size yields a more accurate estimation. For a compromise between accuracy and computing time required, a sample size of 10^5 was selected for each experiment.

IV. Example

Consider the design of the demultiplexing system for a time sharing system that employs the statistical multiplexing technique as shown in Fig. 1a. The input traffic to the demultiplexor is generated from the computer and can be approximated as Poisson arrivals. The message length can be approximated as geometrically distributed with a mean of 20 characters. Further, the message destination can be partitioned into ten groups and the destination function has a step like function as shown in Fig. 4c. The average traffic level is about 0.8. From Fig. 6b, to achieve an overflow probability of 10^{-5}, the required buffer size is 4,850 characters. From Eq. (5), the expected queuing delay due to buffering for the higher traffic destination group is 271 character-service-times and the lower traffic destination group is 35 character-service-times. Therefore, the average-service-time is (271 + 35)/2 = 153 characters-service-times. Now suppose we change the CPU scheduling to a variable quantum time scheduling algorithm. By assigning larger quantum times to these lower traffic intensity destinations, the destination function changes from a step-like function to an approximately uniform function. From Fig. 6b, the required buffer size to achieve $P_{of} = 10^{-5}$ is 2,200 characters, and from Eq. (5), the queuing delay is 78 character-service-times. We note that both the required buffer size and the queuing delay due to buffering have been improved after changing the CPU scheduling algorithm. In a practical system design, we should further consider the inquiry response time constraints, the overhead cost of various scheduling algorithms, and the CPU throughput.

*Simulation results verified this conjecture for two or three outputs in a group.

V. Conclusions

The statistical demultiplexor consists of a buffer and a switching circuit. Due to the complex output structure of the demultiplexing buffer, its behavior is studied via computer simulation. Simulation results reveal that the traffic destination function has a drastic effect on the demultiplexing buffer behavior. The best buffer behavior is achieved by the uniform destination function; that is, the amount of traffic sending to various destinations are equal. Since the CPU scheduling algorithm in a time-sharing system strongly affects the traffic destination function, computer operating systems have large influences on demultiplexor performance. For example, a variable CPU quantum time scheduling algorithm may be effective in producing a uniform destination function and may yield better demultiplexing performance. Thus the simulation model and the results in this paper serve as a useful guide in designing the demultiplexing system and in planning an optimal computer communication system.

Acknowledgment

The author wishes to thank Leo Liang of UCLA for his assistance in the GPSS simulation.

References

1. W. W. Chu, "A Study of the Technique of Asynchronous Time Division Multiplexing for Time Sharing Computer Communications," Proceedings of Second Hawaii International Confrence on System Science, pp. 607-710, 1969.
2. W. W. Chu, "Design Considerations of Statistical Multiplexors," Proceedings of ACM Symposium on Problems in the Optimization of Data Communication Systems, Pine Mountain, Georgia, pp. 36-60, 1969.
3. W. W. Chu, "Selection of Optimal Transmission Rate for Statistical Multiplexors," Proceedings of 1970 International Conference on Communications, San Francisco, California, pp. 28-22 to pp. 28-25, 1970.
4. T. G. Gordon, et al., "Design of Performance of a Statistical Multiplexor," Proceedings of 1970 International Conference on Communications, San Francisco, pp. 28-7 to 28-21, 1970.

5. H. Rudin, Jr., "Performance of a Simple Multiplexor-Concentrator for Data Communication," Proceedings of 1970 International Conference on Communications, San Francisco, California, pp. 35-32 to 35-38, 1970.

6. J.H. Chang, "Comparison of Synchronous and Asynchronous Time Division Multiplexing Techniques," Proceedings of 1970 International Conference on Communications, San Francisco, California, pp. 16-10 to 16-17, 1970.

7. C.D. Pack, "The Effect of Multiplexing on a Computer Communication System," Submitted to the Communication of ACM for publication.

APPENDIX B

COMPUTER COMMUNICATION NETWORK DESIGN--EXPERIENCE WITH THEORY AND PRACTICE

by Howard Frank, Robert E. Kahn, and Leonard Kleinrock

COMPUTER COMMUNICATION NETWORK DESIGN-EXPERIENCE WITH THEORY AND PRACTICE

by

Howard Frank, Network Analysis Corporation
Glen Cove, New York

Robert E. Kahn, Bolt Beranek and Newman Inc.
Cambridge, Mass.

Leonard Kleinrock, University of California
Computer Science Department
Los Angeles, Calif.

ABSTRACT

The design of the ARPA Computer Network brought together many individuals with diverse backgrounds and philosophies. In this paper, we review the methods used in the design of the Network from the vantage of over two years experience in its development. The design variables, system constraints, and performance criteria for the network are discussed along with an evaluation of the tools used to design an efficient and reliable system. The design procedures and the conclusions reached about the network's properties appear to be generally applicable to message switched networks. Consequently, the results of this paper should be useful in the design and study of other store-and-forward computer communication networks.

*This work was supported by the Advanced Research Projects Agency of the Department of Defense under Contract No. DAHC 15-70-C-0120 at the Network Analysis Corporation, Contract No. DAHC 15-69-C-0179 and DAHC-71-C-0088 at Bolt Beranek and Newman Inc., and Contract No. DAHC 15-69-C-0285 at the University of California at Los Angeles.

I. INTRODUCTION

The ARPA Network (ARPANET) project brought together many individuals with diverse backgrounds, philosophies, and technical approaches from the fields of computer science, communication theory, operations research and others. The project was aimed at providing an efficient and reliable computer communications system (using message switching techniques) in which computer resources such as programs, data, storage, special purpose hardware etc. could be shared among computers and among many users [38]. The variety of design methods, ranging from theoretical modeling to hardware development, were primarily employed independently, although cooperative efforts among designers occurred on occasion. As of November 1971, the network has been an operational facility for many months, with about 20 participating sites, a network information center accessible via the net, and well over a hundred researchers, system programmers, computer center directors and other technical and administrative personnel involved in its operation.

In this paper, we review and evaluate the methods used in the ARPANET design from the vantage of over two years experience in the development of the network. In writing this paper, the authors have each made equal contributions during a series of intensive discussions and debates. Rather than present merely a summary of the procedures that were used in the network design, we have attempted to evaluate one anothers methods to determine their advantages and drawbacks. Our approaches and philosophies have often differed radically and, as a result, this has not been an easy or undisturbing process. On the other hand, we have found our collaboration to be extrememly rewarding and, remarkably, we have uncovered many similar properties about the network's behavior that seem to be generally applicable to message switched networks.

The essence of a network is its design philosophy, its performance characteristics, and its cost of implementation and operation. Unfortunately, there is no generally accepted definition of an "optimal" network or even of a "good" network. For example, a network designed to transmit large amounts of data only during late evening hours might call for structural and performance characteristics far different from one servicing large numbers of users who are rapidly exchanging short messages during business hours. We expect this topic, and others such as the merits of message switching vs. circuit switching or distributed vs. centralized control to be a subject of discussion for many years [1,14,32,34].

A cost analysis performed in 1967-1968 for the ARPA Network indicated that the use of message switching would lead to more economical communications and better overall availability and utilization of resources than other methods [34,36]. In addition to its impact on the availability of computer resources, this decision has generated widespread interest in store-and-forward communications. In many instances, the use of store-and-forward communication techniques can result in greater flexibility, higher reliability, significant technical advantage, and substantial economic savings over the use of convential common carrier offerings. An obvious trend towards increased computer and communication interaction has begun. In addition to the ARPANET, research in several laboratories is underway, small experimental networks are being built, and a few examples of other government and commercial networks are already apparent [6,7,31,40,41,47,48].

In the ARPANET, each time-sharing or batch processing computer, called a Host, is connected to a small computer called an Interface Message Processor (IMP). The IMPs, which are interconnected by leased 50 kilobit/second circuits, handle all network communication for their Hosts. To send a message to another Host, a Host precedes the text of its message with an address and simply delivers it to its IMP. The IMPs then determine the route, provide error control , and notify the sender of its receipt. The collection of Hosts, Imps, and circuits forms the message switched resource sharing network. A good description of the ARPANET, and some early results on protocol development and modeling are given in [3,15]. Some experimental utilization of the ARPANET is described in [42]. A more recent evaluation of such networks and a forward look is given in [35,39].

The development of the Network involved four principal activities:

(1) The design of the IMPs to act as nodal store-and forward switches,
(2) the topological design to specify the capacity and location of each communication circuit within the network,
(3) The design of higher level protocols for the use of the network by time-sharing, batch processing and other data processing systems, and
(4) System modeling and measurement of network performance.

Each of the first three activities were essentially performed independently of each other, whereas the modeling effort partly affected the IMP design effort, and closely interacted with the topological design project.

The IMPs were designed by Bolt Beranek and Newman Inc. (BBN) and were built to operate independent of the exact network connectivity; the topological structure was specified by Network Analysis Corporation (NAC) using models of network performance developed by NAC and by the University of California at Los Angeles (UCLA). The major efforts in the area of system modeling were performed at UCLA using theoretical and simulation techniques. Network performance measurements have been conducted during the development of the network by BBN and by the Network Measurement Center at UCLA. To facilitate effective use of the net, higher level (user) protocols are under development by a group of representatives of universities and research centers. This group, known as the Network Working Group, has already specified a Host to Host protocol and a Telnet protocol, and is in the process of completing other function oriented protocols [4,29]. We make no attempt to elaborate on the Host protocol design problems in this paper.

II. THE NETWORK DESIGN PROBLEM

A variety of performance requirements and system contraints were considered in the design of the net. Unfortunately, many of the key design objectives had to be specified long before the actual user requirements could be known. Once the decision to employ message switching was made, and fifty kilobit/second circuits were chosen, the critical design variables were the network operating procedure and the network topology; the desired values of throughput, delay, reliability and cost were system performance and constraint variables. Other constraints affected the structure of the network, but not its overall properties, such as those arising from decisions about the length of time a message could remain within the network, the location of IMPs relative to location of Hosts, and the number of Hosts to be handled by a single IMP.

In this section, we identify the central issues related to IMP design, topological design, and network modeling. In the remainder of the paper, we describe the major design techniques which have evolved.

2.1 IMP PROPERTIES

The key issue in the design of the IMPs was the definition of a relationship between the IMP subnet and the Hosts to partition responsibilities so that reliable and efficient operation would be achieved. The decision was made to build an autonomous subnet, independent (as much as possible) of the operation of any Host. For reliability, the IMPs were designed to be robust against all line failures and the vast majority of IMP and Host failures. This decision required routing strategies that dynamically adapt to changes in the states of IMPs and circuits, an elaborate flow control strategy to protect the subnet against Host malfunction and congestion due to IMP buffer limitations. In addition, a statistics and status reporting mechanism was needed to monitor the behavior of the network.

The number of circuits that an IMP must handle is a design constraint directly affecting both the structures of the IMP and the topological design. The speed of the IMP and the required storage for program and buffers depend directly upon the total required processing capacity, which must be high enough to switch traffic from one line to another when all are fully occupied. Of great importance is the property that all IMPs operate with identical programs. This technique greatly simplifies the problem of understanding network operation and makes network modifications easy to perform.

The detailed physical structure of the IMP is not discussed in this paper [2,15]. However, the operating procedure, which guides packets through the net, is very much of interest here. The flow control, routing, and error control techniques are integral parts of the operating procedure and can be studied apart from the hardware by which they are implemented. Most hardware modifications require changes to many IMPs already installed in the field, while a change in the operating procedure can often be made more conveniently by a change to the single operating program common to all IMPs, which can then be propagated from a single location via the net.

2.2 TOPOLOGICAL PROPERTIES

The topological design resulted in the specification of the location and capacity of all circuits in the network. Projected Host - Host traffic estimates were known at the start to be either unreliable or wrong. Therefore, the network was designed under the assumption of equal traffic between all pairs of nodes. (Additional superimposed traffic was sometimes included for those nodes with expectation of higher traffic requirements). The topological structure was determined with the aid of specially developed heuristic programs to achieve a low cost, reliable network with a high throughput and a general insensitivity to the exact traffic distribution. Currently, only 50 kilobit/second circuits are being used in the ARPANET. This speed line was chosen to allow rapid transmission of short messages for interactive processing, as well as to achieve high throughput for transmission of long messages. For reliability, the network was constrained to have at least two independent paths between each pair of IMPs.

The topological design problem requires consideration of the following two questions:

(1) Starting with a given state of the network topology, what circuit modifications are required to add or delete a set of IMPs?

(2) Starting with a given state of network topology, when and where should circuits be added or deleted to account for long term changes in network traffic?

If the locations of all network nodes are known in advance, it is clearly most efficient to design the topological structure as a single global effort. However, in the ARPANET, as in most actual networks, the initial designation of node locations is modified on numerous occasions. On each such occasion, the topology can be completely reoptimized to determine a new set of circuit locations.

In practice, there is a long lead time between the ordering and the delivery of a circuit and major topological modifications cannot be made without substantial difficulty. It is therefore prudent to add or delete nodes with as little disturbance as possible to the basic network structure consistent with overall economical operation. Figure 1 shows the evolution of the ARPANET from the basic four IMP design in 1969 to the presently planned 26 IMP version. Inspection of the 24 and 26 IMP network designs reveals a few substantial changes in topology that take advantage of the new nodes being added. Surprisingly enough, a complete reoptimization of the 26 IMP topology yields a network only slightly less expensive (about 1% per year) than the present network design [28].

2.3 NETWORK MODELS

The development of an accurate mathematical model for the evaluation of time delay in computer networks is among the more difficult of the topics discussed in this paper. On the one hand, the model must properly reflect the relevant features of the network structure and operation, including practical constraints. On the other hand, the model must result in a mathematical formulation which is tractible and from which meaningful results can be extracted. However, the two requirements are often incompatible and we search for an acceptable compromise between these two extremes.

The major modeling effort thus far has been the study of the behavior of networks of queues [21]. This emphasis is logical since in message switched systems, messages experience queueing delays as they pass from node to node and thus a significant performance measure is the speed at which messages can be delivered. The queueing models were developed at a time when there were no operational networks available for experimentation and model validation, and simulation was the only tool capable of testing their validity. The models, which at all times were recognized to be idealized statements about the real network, were nonetheless crucial to the ARPANET topological design effort since they afforded the only known way to quantitatively predict the properties of different routing schemes and topological structures. The models have been subsequently demonstrated to be very accurate predictors of network throughput and indispensible in providing analytical insight into the network's behavior.

The real problem confronting the designer is to create a system which provides suitable network performance at an acceptable system cost. In particular, for given performance constraints on average packet delay, throughput, reliability and total network cost, he must design the IMP hardware and its operating procedure , select the topology, and assign capacities to each channel. This design must typically be conducted without the benefit of accurate (or often even reasonable) estimates of the user traffic requirements. Futhermore, the design is constrained, for example, by the discrete nature of the available channel capacities, and the number of channels which an IMP can service. A design constraint for the ARPANET is that, on the average, a packet must travel from origin to destination in 0.2 seconds. This complete problem formulation is sufficiently complex that it is difficult to begin the mathematical analysis, much less to obtain an "optimum" solution. The key to the successful development of tractible models has been to factor the problem into a set of simpler queueing problems. There are also heuristic design procedure that one can use in this case. These procedures seem to work quite well and are described later in the paper.

However, if one specializes the problem and removes some of the real constraints, theory and analysis become useful to provide understanding, intuition and design guidelines for the original constrained problem. This approach uncovers global properties of network behavior, which provide keys to good heuristic design procedures and ideal performance bounds.

III. DESIGN TECHNIQUES

In this section we describe the approaches taken to the design problems introduced in Section II. We first summarize the important properties of the ARPANET design:

(1) A communications cost of less than 30 cents per thousand packets(approximately a megabit).
(2) Average packet delays under 0.2 seconds through the net.
(3) Capacity for expansion to 64 IMPs without major hardware or software redesign.
(4) Average total throughput capability of 10 - 15 kilobits/second for all Hosts at an IMP.
(5) Peak throughput capability of 85 kilobits/second per pair of IMPs in an otherwise unloaded network.
(6) Transparent communications with maximum message size of approximately 8000 bits and error rates of one bit in 10^{-12} or less.
(7) Approximately 98% availability of any IMP and close to 100% availability of all operating IMPs from any operable IMP.

The relationships between the various design efforts is illustrated by these properties. The topological design provides for both a desired average throughput and for two or more paths to be fully used for communication between any pair of Hosts.The operating procedure should allow any pair of Hosts to achieve those objectives. The availability of IMPs to communicate reflects both the fact that IMPs are down about 2% of the time and that the topology is selected so that circuit failures contribute little additional to the total system downtime.

3.1 IMP DESIGN

The IMP design consists of two closely coupled but nonetheless separable pieces -- the physical hardware specification (based on timing and reliability considerations and the operating procedure) and the design and implementation of the operating procedure using the specified IMP hardware. The IMP originally developed for the ARPANET contains a 16-bit one microsecond computer that can handle a total bandwidth of approximately one megabit/second (e.g. twenty 50 kilobit/second circuits) if all the messages are maximum size. In the worst case, where all the messages are short, the IMP capacity is about 0.3 megabits/second [15]. Hardware is likely to change as a function of the require IMP capacity but an operating procedure that operates well at one IMP capacity is likely to be transferable to machines that provide different capacity. However, as a network grows in size and utilization, a more comprehensive operating procedure that takes account of known structural properties, such as a hierarchical topology, may be appropriate.

Four primary areas of IMP design, namely message handling and buffering, error control, flow control, and routing are discussed in this section. The IMP provides buffering to handle messages for its Host and packets for other IMPs. Error control is required to provide reliable communication of Host messages in the presence of noisy communication circuits.

The design of the operating procedure should allow high throughput in the net under heavy traffic loads. Two potential obstacles to achieving this objective are: (1) The net can become congested and cause the throughput to decrease with increasing load, and (2) The routing procedure may be unable to always adapt sufficiently fast to the rapid movement of packets to insure efficient routing. A Flow Control and Routing procedure is needed that can efficiently meet this requirement.

3.1.1 Message Handling and Buffering

In the ARPANET, the maximum message size was constrained to be approximately 8000 bits. A pair of Hosts will typically communicate over the net via a sequence of transmitted messages. To obtain delays of a few tenths of a second for such messages and to lower the required IMP buffer storage, the IMP program partitions each message into one or more packets each containing at most approximately 1000 bits. Each packet of a message is transmitted independently to the destination where the message is reassembled by the IMP before shipment to that destination Host. Alternately, the Hosts could assume the responsibility for reassembling messages. For an asynchronous IMP-Host channel, this slightly simplifies the IMP's task. However, if every IMP-Host channel is synchronous, and the Host provides the reassembly, the IMP task is considerably simplified at the expense of "IMP-like" software in each Host and error control on the IMP-Host circuit. Since each Host would require a different program to perform these functions, this would be a very costly process compared to the simplicity of writing a common program for all IMPs. On balance, it is least costly to provide reassembly by the IMPs or to restrict the message lengths to a single packet at the possible expense of lowered throughput.

The method of handling buffers should be simple to allow for fast processing and a small amount of program. The number of buffers should be sufficient to store enough packets for the circuits to be used to capacity; the size of the buffers may be intuitively selected with the aid of simple analytical techniques. ~~and intuition.~~ For example, fixed buffer sizes are useful in the IMP for simplicity of design and speed of operation, but inefficient utilization can arise because of variable length packets. If each buffer contains A words of overhead and provides space for M words of text, and if message sizes are uniformly distributed between 1 and L, it can be shown [45] that the choice of M that minimizes the expected storage is approximately $\sqrt{AL}$. In practice, M is chosen to be somewhat smaller on the assumption that most traffic will be short and that the amount of overhead can be as much as, say, 25% of buffer storage.

3.1.2 Error Control

The IMPs must assume the responsibility for providing error control. There are four possibilities to consider:

(1) Messages are delivered to their destination out of order.
(2) Duplicate messages are delivered to the destination.
(3) Messages are delivered with errors.
(4) Messages are not delivered.

The task of proper sequencing of messages for delivery to the destination Host actually falls in the province of both error control and flow control. If at most one message at a time is allowed in the net between a pair of Hosts, proper sequencing occurs naturally. A duplicate packet will arrive at the destination IMP after an acknowledgment has been missed, thus causing a successfully received packet to be retransmitted. The IMPs can handle the first two conditions by assigning a sequence number to each packet as it enters the network and processing the sequence number at the destination IMP. A Host that performs reassembly can also assign and process sequence numbers and check for duplicate packets. For many applications, the order of delivery to the destination is immaterial. For priority messages, however, it is typically the case that fast delivery requires a perturbation to the sequence.

Errors are primarily caused by noise on the communication circuits and are handled most simply by error detection and retransmission between each pair of IMPs along the transmission path. This technique requires extra storage in the IMP if either circuit speeds or circuit lengths substantially increase. Failures in detecting errors can be made to occur on the order of years to centuries apart with little extra overhead (20 -30 parity bits per packet with the 50 kilobit/second circuits in the ARPANET). Standard cyclic error detection codes may be usefully applied here.

A reliable system design insures that each transmitted message is accurately delivered to its intended destination. The occasional time when an IMP fails and destroys a useful in-transit message is likely to occur far less often than a similar failure in the Hosts and has proven to be unimportant in practice, as are errors due to IMP memory failures. A simple end to end retransmission strategy will protect against these situations, if the practical need should arise. However, the IMPs must be designed so that they can be removed from the network without destroying their internally stored packets.

3.1.3 Flow Control

A network in which packets may freely enter and leave can become congested or logically deadlocked and cause the movement of traffic to halt [5,17]. Flow control techniques are required to prevent these conditions from occurring. The provision of extra buffer storage will mitigate against congestion and deadlocks, but cannot in general prevent them.

The sustained failure of a destination Host to accept packets from its IMP at the rate of arrival will cause the net to fill up and become congested. Two kinds of logical deadlocks, known as reassembly lockup and store-and-forward lockup may occur. In reassembly lockup, packets of partially reassembled messages are blocked from reaching the destination IMP, (thus preventing the message from being completed and the reassembly space freed), by other packets in the net that are waiting for reassembly space at that destination to become free. In a store and forward lockup, the destination has room to accept arriving packets, but the packets interfere with each other by tying up buffers in transit in such a way that none of the packets are able to reach the destination [17]. These phenomena have only been made to occur during very carefully arranged testing of the ARPANET and by simulation [49].

In the original ARPANET design, the use of software links and RFNMS protected against congestion by a single link or a small set of links. However, the combined traffic on a large number of links could still produce congestion. Although this strategy did not protect against lockup, for the levels of traffic encountered by the net to date, the method has provided ample protection.

A particularly simple flow control algorithm that augments the original IMP design to prevent congestion and lockup is also described in [17]. This scheme includes a mechanism whereby packets may be discarded from the net at the destination IMP when congestion is about to occur, with a copy of each discarded packet to be retransmitted a short time later by the originating Host's IMP. Rather than experience excessive delays within the net as traffic levels are increased, the traffic is queued outside the net so that the transit time delays internal to the net continue to remain small. This strategy prevents the insertion of more traffic into the net than it can handle.

It is important to note the dual requirement for small delays for interactive traffic and high bandwidth for the fast transfer of files. To allow high bandwidth between a pair of Hosts, the net must be able to accept a steady flow of packets from one Host and at the same time be able to rapidly quench the flow at the entrance to the source IMP in the event of imminent congestion at the destination. This usually requires that a separate provision be made in the algorithm to protect short interactive messages from experiencing unnecessarily high delays.

3.1.4 Routing

Network routing strategies for distributed networks require routing decisions to be made with only information available to an IMP and the IMP must execute those decisions to effect the routing [14,15]. A simple example of such a strategy is to have each IMP handling a packet independently route it along its current estimate of the shortest path to the destination.

For many applications, it suffices to deal with an idealized routing strategy which may not simulate the IMP routing functions in detail or which uses information not available to the IMP. The general properties of both strategies are usually similar , differing mainly in certain implementation details such as the availability of buffers or the constraint of counters and the need for the routing to quickly adapt to changes in IMP and circuit status.

The IMPs perform the routing computations using information received from other IMPs and local information such as the alive/dead state of its circuits. In the normal case of time varying loads, local information alone, such as the length of internal queues, is insufficient to provide an efficient routing strategy without assistance from the neighboring IMPs. It is possible to obtain sufficient information from the neighbors to provide efficient routing, with a small amount of computation needed per IMP and without each IMP requiring a topological map of the network. In certain applications where traffic patterns exhibit regularity, the use of a central controller might even be preferable. However, for most applications which involve dynamically varying traffic flow, it appears that a central controller cannot be used more effectively than the IMPs to update routing tables if such a controller is constrained to derive its information via the net. It is also a less reliable approach to routing than to distribute the routing decisions among the IMPs.

The routing information cannot be propagated about the net in sufficient time to accurately characterize the instantaneous traffic flow. An efficient algorithm, therefore, should not focus on the movement of individual packets, but rather use topological or statistical information in the selection of routes. For example, by using an averaging procedure, the flow of traffic can be made to build up smoothly. This allows the routing algorithm ample time to adjust its tables in each IMP in advance of the buildup of traffic.

The scheme originally used in the ARPA network had each IMP select one output line per destination onto which to route packets. The line was chosen to be the one with minimum estimated time delay to the destination. The selection was updated every half second using minimum time estimates from the neighboring IMPs and internal estimates of the delay to each of the neighbors. Even though the routing algorithm only selects one line at a time per destination, two output lines will be used if a queue of packets waiting transmission on one line builds up before the routing update occurs and another line is chosen. Modifications to the scheme which allow several lines per destination to be used in an update interval (during which the routing is not changed) are possible using two or more time delay estimates to select the paths.

In practice, this approach has worked quite effectively with the moderate levels of traffic experienced in the net. For heavy traffic flow, this strategy will be inefficient, since the routing information is based on the length of queues, which we have seen can change much faster than the information about the change can be distributed. Fortunately, this information is still usable, although it can be substantially out of date and will not, in general, be helpful in making efficient routing decisions in the heavy traffic case.

A more intricate scheme, recently developed by BBN, allows multiple paths to be efficiently used even during heavy traffic [18]. Preliminary simulation studies indicate that it can be tailored to provide efficient routing in a network with a variety of heavy traffic conditions. This method separates the problem of defining routes onto which packets may be routed from the problem of selecting a route when a particular packet must be routed. By this technique, it is possible to send packets down a path with the fewest IMPs and excess capacity , or when that path is filled, the one with the next fewest IMPs and excess capacity, etc.

A similar approach to routing was independently derived by NAC using an idealized method that did not require the IMPs to participate in the routing decisions. Another approach using a flow deviation technique has recently been studied [13]. The intricacies of the exact approach lead to a metering procedure that allows the overall network flow to be changed slowly for stability and to perturb existing flow patterns to obtain an increased flow. These approaches all possess, in common, essential ingredients of a desirable routing strategy.

3.2 TOPOLOGICAL CONSIDERATIONS

An efficient topological design provides a high throughput for a given cost. Although many measures of throughput are possible, a convenient one is the average amount of traffic that a single IMP can send into the network when all other IMPs are transmitting according to a specified traffic pattern. Often, it is assumed that all other IMPs are behaving identically and each IMP is sending equal amounts of traffic to each other IMP. The constraints on the topological design are the available common carrier circuits, the target cost or throughput, the desired reliability, and the cost of computation required to perform the topological design.

Since, there was no clear specification of the amount of traffic that the network would have to accomodate initially, it was first constructed with enough excess capacity to accomodate any reasonable traffic requirements. Then as new IMPs were added to the system, the capacity was and is still being systematically reduced until the traffic level occupies a substantial fraction of the network's total capacity. At this point, the net's capacity will be increased to maintain the desired percentage of loading. At the initial stages of network design, the "two-connected" reliability constraint essentially determined a minimum value of maximum throughput. This constraint forces the average throughput to be in the range 10-15 kilobits per second per IMP since two communication paths between every pair of IMPs are needed. Alternatively, if this level of throughput is required, then the reliability specification of "two-connectivity" can be obtained without additional cost.

3.2.1 Reliability Computations

A simple and natural characterization of network reliability is the ability of the network to sustain communication between all operable pairs of IMPs. For design purposes, the requirement of two independent paths between nodes insures that at least two IMPs and/or circuits must fail before any pair of operable IMPs cannot communicate. This criterion is independent of the properties of the IMPs and circuits, does not take into account the "degree" of disruption that may occur and hence, does not reflect the actual availability of resources in the network. A more meaningful measure is the average fraction of IMP pairs that cannot communicate because of IMP and circuit failures. This calculation requires knowledge of the IMP and circuit failure rates, and could not be performed until enough operating data was gathered to make valid predictions.

To calculate network reliability, we must consider elementary network structures known as cutsets. A cutset is a set of circuits and/or IMPs whose removal from the network breaks all communication paths between at least two operable IMPs. To calculate reliability, it is often the case that all cutsets must be either enumerated or estimated. As an example , in a 23 IMP, 28 circuit ARPA Network design similar to the one shown in Figure 1(d), there are over twenty million ways of deleting only circuits so that the remaining network has at least one operable pair of IMPs with no intact communication paths. Table 1 indicates the numbers of cutsets in the 23 IMP network as a function of the number of circuits they contain.

A combination of analysis and simulation can be used to compute the average fraction of non-communicating IMP pairs. Detailed descriptions of the analysis methods are given in [44] while their application to the analysis of the ARPANET is discussedin [43]. The results of an analysis of the 23 IMP version of the network are shown in Figure 2. The curve marked A shows the results under the assumption that IMPs do not fail, while the curve marked B shows the case where circuits do not fail. The curve marked C assumes that both IMPs and circuits fail with equal probability. In actual operation, the average failure probability of both IMPs and circuits is about 0.02. For this value, it can be seen that the effect of circuit failures is far less significant than the effect of IMP failures. If an IMP fails in a network with n IMPs, at least n-1 other IMPs cannot communicate with it. Thus, good network design cannot improve upon the effect directly due to IMP failures, which in the ARPANET is the major factor affecting reliabilty. Further, more intricate reliability analyses which consider the loss of throughput capacity because of circuit failures have also been performed and these losses shown to be negligible [28]. Finally, unequal failure rates due to differences in line lengths have been shown to have only minor effects on the analysis and can usually be neglected [27].

3.2.2 Topological Optimization

During the computer optimization process, the reliability of the topology is assumed to be acceptable if the network is at least two-connected. The object of the optimization is to decrease the ratio of cost to throughput subject to an overall cost limitation. This technique employs a sophisticated network optimization program that utilizes circuit exchange heuristics, routing and flow analysis algorithms, to generate low cost designs. In addition, two time delay models were initially used to(1)calculate the throughput corresponding to to an average time delay of 0.2 seconds, (2) estimate the packet rejection rate due to all buffers filling at an IMP. As experience with these models grew, the packet rejection rate was found to be negligible and the computation discontinued. The delay computation (equation (7) in section 3.3.2) was subsequently first replaced by a heuristic calculation to speed the computation and later eliminated after it was found that time delays could be guaranteed to be acceptably low by preventing cutsets from being saturated. This "threshold" behavior is discussed further in section 3.3.

The basic method of optimization was described in [12] while extensions to the design of large networks are discussed in [9]. The method operates by initially generating, either manually or by computer, a "starting network" that satisfies the overall network constraints but is not, in general, a low cost network. The computer then iteratively modifies the starting network in simple steps until a lower cost network is found that satisfies the constraints or the process is terminated. The process is repeated until no further improvements can be found. Using a different starting network can result in a different solution. However, by incorporating sensible heuristics and by using a variety of carefully chosen starting networks and some degree of man-machine interaction, "excellent" final networks usually result. Experience has shown that there are a wide variety of such networks with different topological structures but similar cost and performance.

The key to this design effort is the heuristic procedure by which the iterative network modifications are made. The method used in the ARPANET design involves the removal and addition of one or two circuits at a time. Many methods have been employed, at various times, to identify the appropriate circuits for potential addition or deletion. For example, to delete uneconomical circuits a straightforward procedure simply deletes single circuits in numerical order, reroutes traffic and reevaluates cost until a decrease in cost per megabit is found. At this point, the deletion is made permanent and the process begins again. A somewhat more sophisticated procedure deletes circuits in order of increasing utilization, while a more complex method attempts to evaluate the effect of the removal of any circuit before any deletion is attempted. The circuit with the greatest likelihood of an improvement is then considered for removal and so on.

There are a huge number of reasonable heuristics for circuit exchanges. After a great deal of experimentation with many of these, it appears that the choice of a particular heuristic is not critical. Instead, the speed and efficiency with which potential exchanges can be investigated appears to be the limiting factor affecting the quality of the final design. Finally, as the size of the network increases, the greater the cost becomes to perform any circuit exchange optimization. Decomposition of the network design into regions becomes necessary and additional heuristics are needed to determine effective decompositions. It presently appears that these methods can be used to design relatively efficient networks with a few hundred IMPs while substantially new procedures will be necessary for networks of greater size.

The topological design requires a routing algorithm to evaluate the throughput capability of any given network. Its properties must reflect those of an implementable routing algorithm, for example, within the ARPANET. Although the routing problem can be formulated as a "multicommodity flow problem" [10] and solved by linear programming for networks with 20 - 30 IMPs [8], faster techniques are needed when the routing algorithm is incorporated in a design procedure. The design procedure for the ARPA Network topology iteratively analyses thousands of networks. To satisfy the requirements for speed, an algorithm which selects the least utilized path with the minimum number of IMPs was initially used [12]. This algorithm was later replaced by one which sends as much traffic as possible along such paths until one or more circuits approach a few percent of full utilization [28]. These highly utilized circuits are then no longer allowed to carry additional flow. Instead, new paths with excess capacity and possibly more intermediate nodes are found. The procedure continues until some cutset contains only nearly fully utilized circuits. At this point no additional flow can be sent. For design purposes, this algorithm is a highly satisfactory replacement for the more complicated multi-commodity approach. Using the algorithm, it has been shown that the throughput capabilities of the ARPA Network are substantially insensitive to the distribution of traffic and depend mainly only on the total traffic flow within the network.

3.3 ANALYTIC MODELS OF NETWORK PERFORMANCE

The effort to determine analytic models of system performance has proceeded in two phases: (1) the prediction of average time delay encountered by a packet as it passes through the network, and (2) the use of these queueing models to calculate optimum channel capacity assignments for minimum possible delay. The model used as a standard for the average packet delay was first described in [21] where it served to predict delays in stochastic communication networks. In [22], it was modified to describe the behavior of ARPA-like computer networks while in [23], it was refined further to apply directly to the ARPANET.

3.3.1 The Single Server Model

Queueing theory [20] provides an effective set of analytical tools for studying packet delay. Much of this theory considers systems in which messages place demands for transmission (service) upon a single communication channel (the single server). These systems are characterized by $A(\tau)$, the distribution of interarrival times between demands and $B(t)$, the distribution of service times. When the average demand for service is less than the capacity of the channel, the system is said to be stable.

When $A(\tau)$ is exponential (i.e. Poisson arrivals), and packets are transmitted on a first-come-first-served basis, the average time T in the stable system is

$$T = \frac{\lambda \overline{t^2}}{2(1-\rho)} + \bar{t} \qquad (1)$$

where λ is the average arrival rate of messages, $\bar{t}$ and $\overline{t^2}$ are the first and second moments of $B(t)$ respectively, and $\rho = \lambda\bar{t} < 1$. If the service time is also exponential,

$$T = \frac{\bar{t}}{1-\rho} \qquad (2)$$

When $A(\tau)$ and $B(t)$ are arbitrary distributions, the situation becomes complex and only weak results are available. For example, no expression is available for T; however the following upper bound yields an excellent approximation [19] as $\rho \rightarrow 1$:

$$T \leq \frac{\lambda(\sigma_a^2 + \sigma_b^2)}{2(1 - \rho)} + \bar{t} \qquad (3)$$

where σ_a^2 and σ_b^2 are the variance of the interarrival time and service time distributions, respectively.

3.3.2 Networks Of Queues

Multiple channels in a network environment give rise to queueing problems that are far more difficult to solve than single server systems. For example, the variability in the choice of source and destination for a message is a network phenomenon which contributes to delay. A principal analytical difficulty results from the fact that flows throughout the network are correlated. The basic approach to solving these stochastic network problems is to decompose them into analyzable single-server problems which reflect the original network structure and traffic flow.

Early studies of queueing networks indicated that such a decomposition was possible [50,51]; however, those results do not carry over to message switched computer networks due to the correlation of traffic flows. In [21] it was shown that, for a wide variety of communication nets the length of a given packet could be considered as an independent random variable as it passes from node to node. Although this "independence" assumption is not physically realistic, it results in a mathematically tractable model which does not seem to affect the accuracy of the predicted time delays. As the si[illegible] and connectivity of the network increases, the assumption becomes increasingly more realistic. With this assumption, a successful decomposition which permits a channel-by-channel analysis is possible, as follows.

The packet delay is defined as the average time which a packet spends in the network from its entry until it reaches its destination. The average packet delay is denoted as T. Let Z_{jk} be the average delay for those packets whose origin is IMP j and whose destination is IMP k. We assume a Poisson arrival process for such packets with an average of γ_{jk} packets per second and an exponential distribution of packet lengths with an average of $1/\mu$ bits per packet. With these definitions, if γ is the sum of the quantities γ_{jk}, then [21]

$$T = \sum_{j,k} \frac{\gamma_{jk}}{\gamma} Z_{jk} \tag{4}$$

Let us now reformulate equation (4) in terms of single channel delays. We first define the following quantities for the ith channel: C_i as its capacity (bits/second); λ_i as the average packet traffic it carries (packets/second); T_i as the average time a packet spends waiting for and using the ith channel. By relating the $\{\lambda_i\}$ to the $\{\gamma_{jk}\}$ via the paths selected by the routing algorithm, it is easy to see that [21]

$$T = \sum_{i} \frac{\lambda_i}{\gamma} T_i \tag{5}$$

With the assumption of Poisson traffic and exponential service times, the quantities T_i are given by equation (2). For an average packet length of $1/\mu$, $\bar{t} = 1/\mu C_i$ and thus

$$T_i = \frac{1}{\mu C_i - \lambda_i} \tag{6}$$

Thus we have successfully decomposed the analysis problem into a set of simple single-channel problems.

A refinement of the decomposition permits a non-exponential packet length distribution and uses equation (1) rather than equation (2) to calculate T_i; as an approximation, the Markovian character of the traffic is assumed to be preserved. Furthermore, for computer networks we include the effect of propagation time and overhead traffic to obtain the following equation for average packet delay [22,23]

$$T = K + \sum_i \frac{\lambda_i}{\gamma}\left[\frac{1}{\mu' C_i} + \frac{\lambda_i/\mu C_i}{\mu C_i - \lambda_i} + P_i + K\right] \qquad (7)$$

Here, $1/\mu'$ represents the average length of a Host packet, and $1/\mu$ represents the average length of all packets (including acknowledgements, headers, requests for next messages, parity checks, etc.) within the network. The expression $1/\mu' C_i + [(\lambda_i/\mu C_i)/(\mu C_i - \lambda_i)] + P_i$ represents the average packet delay on the ith channel. The term $(\lambda_i/\mu C_i)/(\mu C_i - \lambda_i)$ is the average time a packet spends waiting at the IMP for the ith channel to become available. Since the packet must compete with acknowledgments and other overhead traffic, the overall average packet length $1/\mu$ appears in the expression. The term $1/\mu' C_i$ is the time required to transmit a packet of average length $1/\mu'$. Finally, K is the nodal processing time, assumed constant and for the ARPA IMP, approximately equal to 0.35 ms; P_i is the propagation time on the ith channel (about 20 ms for a 3000 mile channel).

Assuming a relatively homogeneous set of C_i and P_i, no individual term in the expresson for delay will dominate the summation until the flow in one channel (say channel i_0) approaches capacity. At that point, the term T_{i_0}, and hence T will grow rapidly. The expression for delay is then dominated by one or more terms and exhibits a threshold behavior. Prior to this threshold, T remains relatively constant.

The accuracy of the time delay model, as well as this threshold phenomenon was demonstrated on a 19 node network [14] and on the ten node ARPA net derived from Figure 1(c) by deleting the rightmost five IMPs. Using the routing procedure described in the last section, [28] and equal traffic between all node pairs, the channel flows λ_i were found for the ten node net and the delay curves shown in Figure 3 were obtained. Curve A was obtained with fixed 1000 bit packets*, while curve B was generated for exponentially distributed variable length packets with average size of 500 bits. In both cases A and B, all overhead factors were ignored. Note that the delay remains small until a total throughput slightly greater than 400

----

*In case A, the application of equation (1) allows for constant packet lengths (i.e. zero variance).

kilobits/second is reached. The delay then increases rapidly. Curves C and D respectively represent the same situations when the overhead of 136 bits per packet and per RFNM and 152 bits per acknowledgment are included. Notice that the total throughput per IMP is reduced to 250 kilobits/second in case C and to approximately 200 kilobits/second in case D.

In the same figure, we have illustrated with x's the results of a simulation performed with a realistic routing and metering strategy. The simulation omitted all network overhead and asumed fixed lengths of 1000 bits for all packets.

It is difficult to develop a practical routing and flow control procedure that will allow each IMP to input identical amounts of traffic. To compare the delay curve A with the points obtained by simulation, the curve should actually be recomputed for the slightly skewed distribution that resulted. It is notable that the delay estimates from the simulation (which used a dynamic routing strategy) and the computation (which used a static routing strategy and the time delay formula) are in close agreement. In particular, they both accurately determined the vertical rise of the delay curve in the range just above 400 kilobits/second, the formula by predicting infinite delay and the simulation by rejecting the further input of traffic.

In practice and from the analytic and simulation studies of the ARPANET , the average queueing delay is observed to remain small (almost that of an unloaded net) and well within the design constraint of 0.2 seconds until the traffic within the network approaches the capacity of a cutset. The delay then increases rapidly. <u>Thus, as long as traffic is low enough and the routing adaptive enough to avoid the premature saturation of cutsets by guiding traffic along paths with excess capacity, queueing delays are not significant.</u>

3.3.3 Continuous Capacity Optimization

One of the most difficult design problems is the optimal selection of capacities from a <u>finite</u> set of options. Although there are many heuristic approaches to this problem, analytic results are relatively scarce. (For the specialized case of centralized networks, an algorithm yielding optimal results is available [11]). While it is possible to find an economical assignment of discrete capacities for, say, a 200 IMP network, very little is known about the relation between such capacity assignments, message delay, and cost.

To obtain theoretical properties of optimal capacity assignments, we first ignore the constraint that capacities are obtainable only in discrete sizes. In [21] the problem was posed where the network topology and average traffic flow were assumed to be known and fixed and an optimal match of capacities to traffic flow was found. Also, the traffic was assumed to be Markovian (Poisson arrivals and exponential packet lengths) and the independence assumption and decomposition method were applied. For each channel, the capacity C_i was found which minimized the average message delay T, at a fixed total system cost D. (Since λ_i/μ is the average bit rate on the ith channel, the solution to <u>any</u> optimal assignment problem must provide more than this minimal capacity to each channel. This is clear since both equations (6) and (7) indicate that T_i will become arbitrarily large with less than (or equal to) this amount of capacity. It is not critical exactly how the <u>excess</u> capacity is assigned, as long as $C_i > \lambda_i/\mu$.) The optimization further assumed that a total of D dollars was available to provide the channel capacities and that the cost of the ith channel was linear at a rate of d_i dollars per unit of channel capacity; that is $D = \sum_i d_i C_i$. The simpler form for T_i in equation (6) is used in this formulation and T is as given in equation (5).

The solution to this problem assigns a capacity to the ith channel in an amount equal to λ_i/μ plus some excess capacity proportional to the square root of that traffic. With T evaluated for this assignment,

$$T = \frac{\bar{n}}{\mu D_e}\left(\sum_i \sqrt{d_i(\lambda_i/\lambda)}\right)^2 \qquad (8)$$

Here $\lambda = \sum_i \lambda_i$ represents the total rate at which packets flow within the net and D_e is the difference between D and the amount which must be spent to provide each channel with capacity λ_i/μ , namely

$$D_e = D - \sum_i \frac{\lambda_i d_i}{\mu} \tag{9}$$

Moreover, $\bar{n} = \frac{\lambda}{\gamma}$ is easily shown to represent the average path length for a packet.

If d_i = 1 for all channels, $D = \sum C_i = C$ where C represents the total capacity within the network.* In this case,

$$T = \frac{\bar{n}}{\mu C(1 - \bar{n}\rho)} \left(\sum_i \sqrt{\lambda_i/\lambda}\right)^2 \tag{10}$$

Here $\rho = \gamma/\mu C$ is the ratio of the rate γ/μ at which bits enter the network to the rate C at which the net can handle bits. the quantity ρ represents a dimensionless form of network "load." As the load ρ approaches $1/\bar{n}$, the delay T grows very quickly, and this point $\rho = 1/\bar{n}$ represents the maximum load which the network can support. If capacities are assigned optimally, all channels saturate simultaneously at this point. In this formulation $\bar{n}$ is a design parameter which depends upon the topology and the routing procedure, while ρ is a parameter which depends upon the input rate and the total capacity of the network. Equation (10) provides insight into topological structure and routing procedures [21].

In a recent paper [26], it was observed that, in minimizing T, a wide variation was possible among the packet delays T_i. As a result, the problem of finding the sets of channel capacities which minimize $T^{(k)}$ was considered, where

$$T^{(k)} = \left[\sum_i \frac{\lambda_i}{\gamma} (T_i)^k\right]^{\frac{1}{k}} \tag{11}$$

The solution for the optimal channel capacity assignment with a given value k, denote by $C_i^{(k)}$ is

----

*The assumption d_i = 1 is of practical importance in the case of satellite channels [33].

$$C_i^{(k)} = \frac{\lambda_i}{\mu} + \left(\frac{D_e}{d_i}\right) \frac{(\lambda_i d_i^k)^{1/k+1}}{\sum_j (\lambda_j d_j^k)^{1/k+1}} \qquad (12)$$

With this capacity assignment,

$$T^{(k)} = \left(\frac{\bar{n}}{\mu D_e}\right) \left(\sum (\lambda_i d_i^k/\lambda)^{1/1+k}\right)^{1+k} \qquad (13)$$

Note that the assignment $C_i^{(1)}$ is the previously mentioned assignment which minimizes T, and $T^{(1)}$ is the previously stated value of delay. As k increases the variation in the T_i decreases and as $k \longrightarrow \infty$

$$C_i^{(k)} \longrightarrow \frac{\lambda_i}{\mu} + \frac{D_e}{\sum_j d_j} \qquad (14)$$

In the limit, the channel capacity is assigned to give each channel its minimum required amount λ_i/μ plus a constant additional amount. All the T_i are equal and $T = \frac{\bar{n}}{\mu D_e} \sum_j d_j$. Moreover, setting k = 0 yields

$$C_i^{(0)} = \frac{\lambda_i}{\mu} + \frac{\lambda_i D_e}{\bar{n} \gamma d_i} \qquad (15)$$

For this assignment, the value of T is identical to the value it achieves when $k \rightarrow \infty$, although different channel capacity assignments typically occur at these extremes [46]. If all d_i =1 a channel capacity is assigned in proportion to the traffic carried by that channel (commonly known as the proportional capacity assignment). Although the value of T is minimized for the capacity assignments which result when k = 1, T increases slowly as k varies from unity and, moreover, the variance of packet delay is minimized when k = 2.

In studying the ARPANET [23] a closer representation of the actual tariffs for high speed telephone data channels used in that network was provided by setting $D = \sum_i d_i C_i^{\alpha}$ where $0 \leq \alpha \leq 1$.* This approach requires the solution of a non-linear equation by numerical techniques. On solving the equation, it can be shown that the packet delay T varies insignificantly with α for $.3 \leq \alpha \leq 1$. This indicates that the closed form solution discussed earlier with $\alpha = 1$ is a reasonable approximation to the more difficult non-linear problem.

In practice, the selection of channel capacities must be made from a small finite set. Although some theoretical work has been done in this case by approximating the discrete cost-capacity functions by continuous ones, much remains to be done [13,25]. Because of the discrete capacities and the time varying nature of network traffic, it is not generally possible to match channel capacities to the anticipated flows within the channels. If this were possible, all channels would saturate at the same externally applied load. Instead, capacities are assigned on the basis of reasonable estimates of average or peak traffic flows. It is the responsibility of the routing procedure to allow the traffic to adapt to the available capacity [14]. Often two IMP sites will engage in heavy communication and thus saturate one or more critical network cutsets. In such cases, the routing will not be able to send additional flow across these cuts. The network will therefore experience "premature" saturation in one or a small set of channels leading to the threshold behavior described earlier.

----

*Of course the tariffs reflect the discrete nature of available channels. The use of the exponent α provides a continuous fit to the discrete cost function. For the ARPANET, $\alpha \cong .8$.

IV. DISCUSSION

A major conclusion from our experience in network design is that message switched networks of the ARPA type are no longer difficult to specify. They may be implemented straightforwardly from the specifications, they can be less expensive than other currently available technical approaches; they perform remarkably well as a communication system for interconnecting time-sharing and batch processing computers and can be adapted to directly handle teletypes, displays and many other kinds of terminal devices and data processing equipment [16,30].

The principal tools available for the design of networks are analysis, simulation, heuristic procedures, and experimentation. Analysis, simulation and heuristics have been the mainstays of the work on modeling and topological optimization while simulation, heuristic procedures and experimental techniques have been the major tools for the actual network implementation. Experience has shown that all of these methods are useful while none are all powerful. The most valuable approach has been the simultaneous use of several of these tools.

Eac approach has room for considerable improvement. The analysis efforts have not yet yielded results in many important areas such as routing. However, for prediction of delay, this approach leads to a simple threshold model which is both accurate and understandable. Heuristic procedures all suffer from the problem that is presently unclear how to select appropriate heuristics. It has been the innovative use of computers and analysis that has made the approach work well. For designing networks with no more than a few hundred IMPs, present heuristics appear adequate but a good deal of additional work is required for networks of greater size. Simulation is a well developed tool that is both expensive to apply and limited in the overall understanding that it can yield. For these reasons, simulation appears to be most useful only in validating models, and in assisting in detailed design decisions such as the number of buffers that an IMP should contain. As the size of networks continue to grow, it appears that simulation will become virtually useless as a total design tool. The ultimate standard by which all models and conclusions can be tested is experimentation. Experimentation with the actual network is conceptually relatively straightforward and very useful. Althouch, experiments are often logistically difficult to perform, they can provide an easy means for testing models, heuristics and design parameters.

The outstanding design problems currently facing the network designer are to specify and determine the properties of the routing, flow control and topological structure for large networks. This specification must make full use of a wide variety of circuit options. Preliminary studies indicate that initially, the most fruitful approaches will be based on the partitioning of the network into regions, or equivalently, constructing a large network by connecting a number of regional networks. To send a message, a Host would specify both the destination region and the destination IMP in that region. No detailed implementation of a large network has yet been specified but early studies of their properties indicate that factors such as cost, throughput, delay and reliability are similar to those of the present ARPANET, if the ARPA technology is used [9].

Techniques applicable to the design of large networks are presently under intensive study. These techniques appear to split into the same four categories as small network design but approaches may differ significantly. For example, large nets are likely to demand the placement of high bandwidth circuits at certain key locations in the topology to concentrate flow. These circuits will require the development of a high speed IMP to connect them into the net. It is likely that this high speed IMP will have the structure of a high speed multiplexor, and may require several cooperating processors to obtain the needed computer power for the job. Flow control strategies for large networks seem to extrapolate nicely from small network strategies if each region in the large network is viewed as a node in a smaller network. However, this area will require additional study as will the problem of specifying effective adaptive routing mechanisms. Recent efforts indicate that efficient practical schemes for small networks will soon be available. These schemes seem to be applicable for adaptive routing and flow control in networks constructed from regional subnetworks. The development of practical algorithms to handle routing and flow control is still an art rather than a science. Simulation is useful for studying the properties of a given heuristic, but intuition still plays a dominant role in the system design.

Several open questions in network design presently are: (1) what structure should a high bandwidth IMP have; (2) How can full use be made of a variety of high bandwidth circuits; (3) How should large networks be partitioned for both effective design and operation; and (4) what operational procedures should large networks follow. Much work has already been done in these areas but much more remains to be done. We expect substantial progress to be achieved in the next few years, and accordingly, the increased understanding of the properties of message switched networks of all sizes.

ACKNOWLEDGEMENT

The ARPA Network is in large part the conception of Dr. L. G. Roberts of the Advanced Research Projects Agency to whom we owe a debt of gratitude for his support and encouragement. We also acknowledge the helpful contributions of S. Crocker and B. Dolan of ARPA. At BBN, NAC , and UCLA many individuals, too numerous to list, participated in the network effort and we greatfully acknowledge their contributions.

REFERENCES

1. P. Baran, S. Boehm, P.Smith, "On Distributed Communications," Series of 11 reports by Rand Corporation, Santa Monica, California, 1964.

2. BBN Report No. 1822, "Specifications for the Interconnection of a Host and an IMP," 1971 revision.

3. S. Carr, S. Crocker, V. Cerf, "Host-Host Communication Protocol in the ARPA Network. SJCC, 1970, pp.589-597.

4. S. Crocker et al, "Function Oriented Protocols For The ARPA Network," SJCC, 1972, in this issue.

5. D. W. Davies, "The Control of Congestion in Packet Switching Networks," Proc. of the Second ACM IEEE Symposium in the Optimization of Data Communications Systems, Palo Alto, California, Oct. 1971.

6. D. Farber, "Loop Networks," Private Communication, 1970.

7. W, D, Farmer, E. E. Newhall, "An Experimental Distribution Switching System to Handle Bursty Computer Traffic,"Proc. of the ACM Symposium on Problems in the Optimization of Data Communication Systems, 1969, pp. 1-34

8. H. Frank, W. Chou, "Routing in Computer Networks," NETWORKS, John Wiley, 1971, Vol. 1, No. 2, pp. 99-112.

9. H. Frank , W. Chou, "Cost and Throughput in Computer-Communication Networks," to appear in the Infotech Report on the State of the Art of Computer Networks, 1972.

10. H. Frank and I. T. Frish, Communication, Transmission and Transportation Networks, Addison Wesley, 1972.

11. H. Frank, I. T. Frisch, W. Chou, R. Van Slyke, "Optimal Design of Centralized Computer Networks," Networks, John Wiley, Vol.1, No. 1, pp. 43-57, 1971.

12. H. Frank, I. T. Frisch, W. Chou, "Topological Considerations in the Design of the ARPA Computer Network," SJCC, May 1970, pp. 581-587.

13. L. Fratta, M. Gerla, L. Kleinrock, "The Flow Deviation Method; An Approach to Store-and-Forward Network Design," to be published.

14. G. Fultz, and L. Kleinrock, "Adaptive Routing Techniques for Store-and Forward Computer -Communication Networks,"Proc. of the International Conference on Communications, 1971, pp. 39-1 to 39-8.

15. F. Heart, R. Kahn,S. Ornstein, W. Crowther and D. Walden, "The Interface Message Processor for the ARPA Computer Network," SJCC, 1970, pp. 551-567.

16. R. E. Kahn, "Terminal Access to the ARPA Computer Network," Proc. of Third Courant Institute Symposium, Nov. 1970. To be published by Prentice Hall, Englewood Cliffs, N.J.

17. R. E. Kahn, and W. R. Crowther, "Flow Control in a Resource Sharing Computer Network," Proc. of the Second ACM IEEE Symposium on Problems in the Optimization of Data Communication Systems, Palo Alto, California, October 1971.

18. R. E. Kahn, W. R. Crowther, "A Study of the ARPA Network Design and Performance," BBN Report No. 2161, August 1971.

19. J. F. C. Kingman, "Some Inequalities for the Queue GI/G/1," Biometrica, 1962, pp. 315-324.

20. L. Kleinrock, Queueing Systems; Theory and Application, to be published by John Wiley, 1972.

21. L.Kleinrock, Communication Nets: Stochastic Message Flow and Delay, McGraw-Hill, 1964.

22. L. Kleinrock, "Models for Computer Networks," Proc. of the International Conference on Communications, 1969, pp.21 -9 to 21-16.

23. L. Kleinrock, "Analysis and Simulation Methods in Computer Network Design," SJCC, 1970, pp.569-579.

24. T. Marill and L. G. Roberts, "Toward a Cooperative Network of Time-Shared Computers," FJCC, 1966

25. B. Meister, H. Muller, and H. Rudin, Jr., "Optimization of a New Model for Message-Switching Networks," Proc. of the International Conference on Communications, 1971, pp. 39-16 to 39-21.

26. B. Meister, H. Muller, and H. Rudin, "New Optimization Criteria for Message-Switching Networks," IEEE Transactions on Communication Technology, Com-19, June 1971, pp.256-260.

27. N. A, C. Third Semiannual Technical Report for the Project, "Analysis and Optimization of Store-and-Forward Computer Networks," Defense Documentation Center, Alexandria, Va. June 1971.

28. N. A. C. Fourth Semiannual Technical Report for the Project "Analysis and Optimization of Store-and-Forward Computer Networks," Defense Documentation Center, Alexandria, Va. Dec. 1971

29. Network Working Group, "The Host/Host Protocol for the ARPA Network," N. I. C. No. 7147, 1971. Available from the Network Information Center, Stanford Research Institute, Menlo Park, Calofornia.

30. Ornstein et al, "The Terminal IMP for the ARPA Network," SJCC, 1972, in this issue.

31. J. Pierce, "A Network for Block Switching of Data," to appear in Bell System Technical Journal.

32. E. Port, F. Clos, "Comparisons of Switched Data Networks on the Basis of Waiting Times," IBM Research Report RZ 405, IBM Research Laboratories, Zurich , Switzerland, Jan, 1971. (Copies available from IBM Watson Research Center, P. O, Box 218, Yorktown Heights, New York 10598).

33. H. G. Raymond, "A Queueing Theory Approach to Communication Satellite Network Design," Proc. of the International Conference on Communication, pp. 42-26 to 42-31, 1971.

34. L. G. Roberts, "Multiple Computer Networks and Inter-Computer Communications," ACM Symposium on Operating Systems, Gatlinburg, Tenn., 1967.

35. L G. Roberts, "A Forward Look", SIGNAL, Vol. XXV, No. 12, pp. 77-81, August, 1971.

36. L. G. Roberts, "Resource Sharing Networks," IEEE International Conference, March, 1969.

37. L. G. Roberts, "Access Control and File Directories in Computer Networks," IEEE International Convention, March, 1968.

38. L. G. Roberts, B. Wessler, Computer Network Development to Achieve Resource Sharing," SJCC, 1970, pp.543-549.

39. L. G. Roberts and B. Wessler,"The ARPA Computer Network," in Computer Communication Networks, edited by

Abramson and Kuo, Prentice Hall, 1972.

40. R. A, Scantlebury, P. T. Wilkinson, "The Design of a Switching System to Allow Remote Access to Computer Services by other Computers, "Second ACM/IEEE Symposium on Problems in the Optimization of Data Communications Systems, Palo Alto, California, October, 1971.

41. R.A. Scantlebury, "A model for the Local Area of a Data Communication Network - Objectives and Hardware Organization," ACM Symposium on Problems in the Optimization of Data Communication Systems , Pine Mountain, Ga., 1969, pp.179-201.

42. R. H. Thomas, D. A. Henderson, "McRoss - A Multi - Computer Programming System," SJCC, 1972, in this issue.

43. R. Van Slyke, H. Frank, "Reliability of Computer-communication Networks," Proc. of the Fifth Conference on Applications of Simulation, New York, December, 1971.

44. R. Van Slyke, H. Frank, "Network Reliability Analysis - I," Networks, Vol. 1, No. 3, 1972.

45. E. Wolman, "A Fixed Optimum Cell Size for Records o Various Length," JACM, 1965, pp. 53-70.

46. L. Kleinrock, "Scheduling, Queueing and Delays in Time-Sharing Systems and Computer Networks, to appear in Computer Commmunication Networks, edited by Abramson and Kuo, Prentice Hall, 1972.

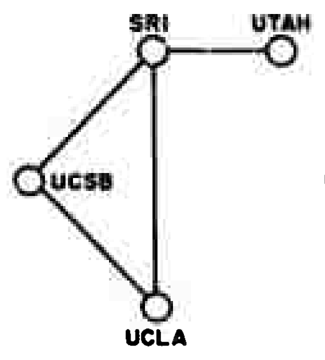
47. Fredericksen et al, "OS 360 Network Interface Users Guide," IBM Research, Yorktown Heights, New York, 1971.

48. M. Beere, "Tymnet - A Serendipidous Evolution," Second ACM IEEE Symposium on Problems in the Optimization of Data Communication Systems, Palo Alto, California, 1971.

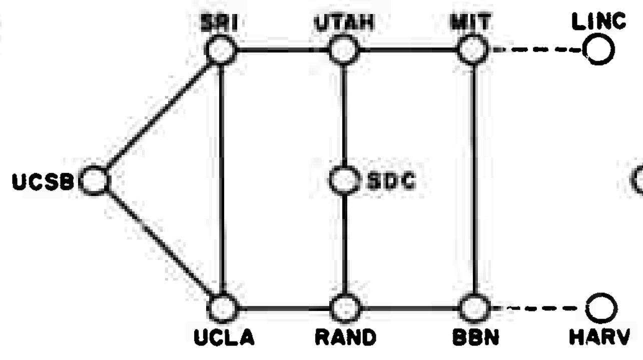
49. W. Teitelman, R. E. Kahn, "A Network Simulation and Display Program," Third Princeton Conference on Information Sciences and Systems, 1969, p.29

50. P. Burke, "The Output of a Queueing System," Operations Research, 1956, pp. 699-704.

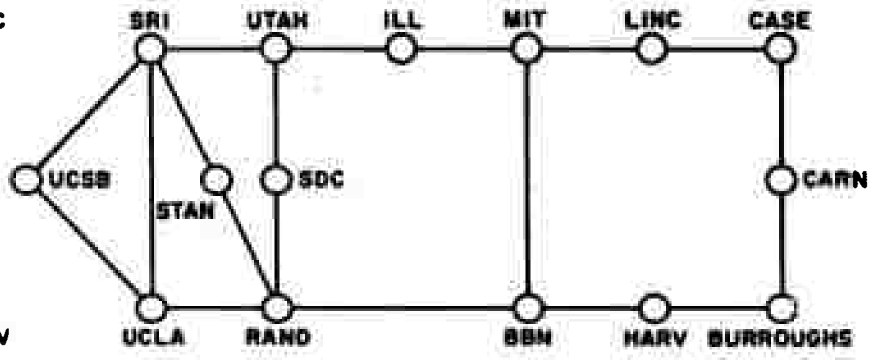
51. J. R. C. Jackson, "Networks of Waiting Lines," Operations Research, Vol. 5, 1957, pp. 518-521.



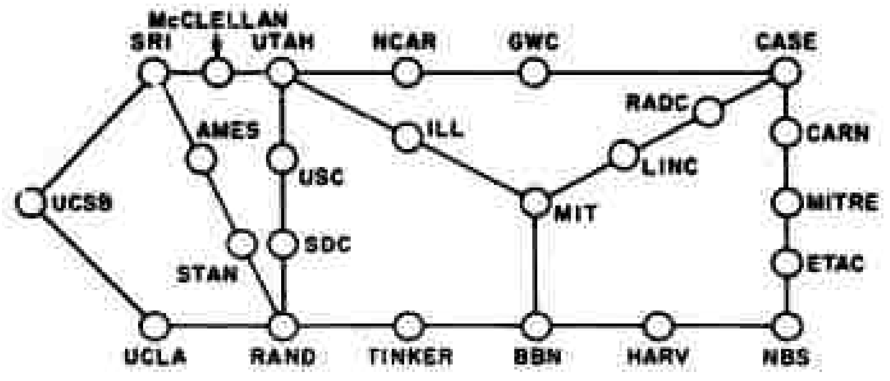
4 -IMP NETWORK - 12/1/69
(a)



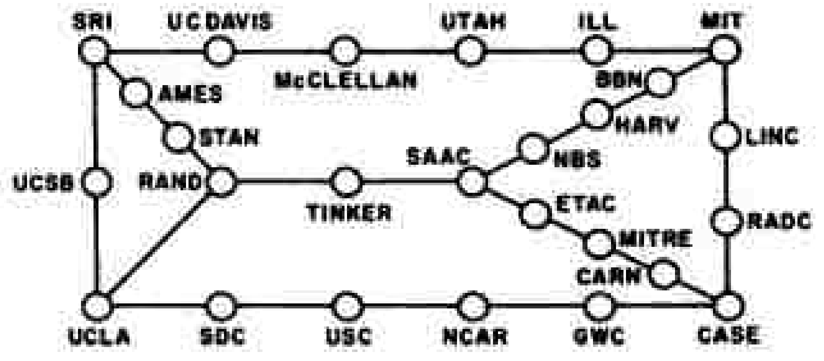
10-NODE NETWORK - 7/1/70
(b)



15-IMP NETWORK - 3/1/71
(c)

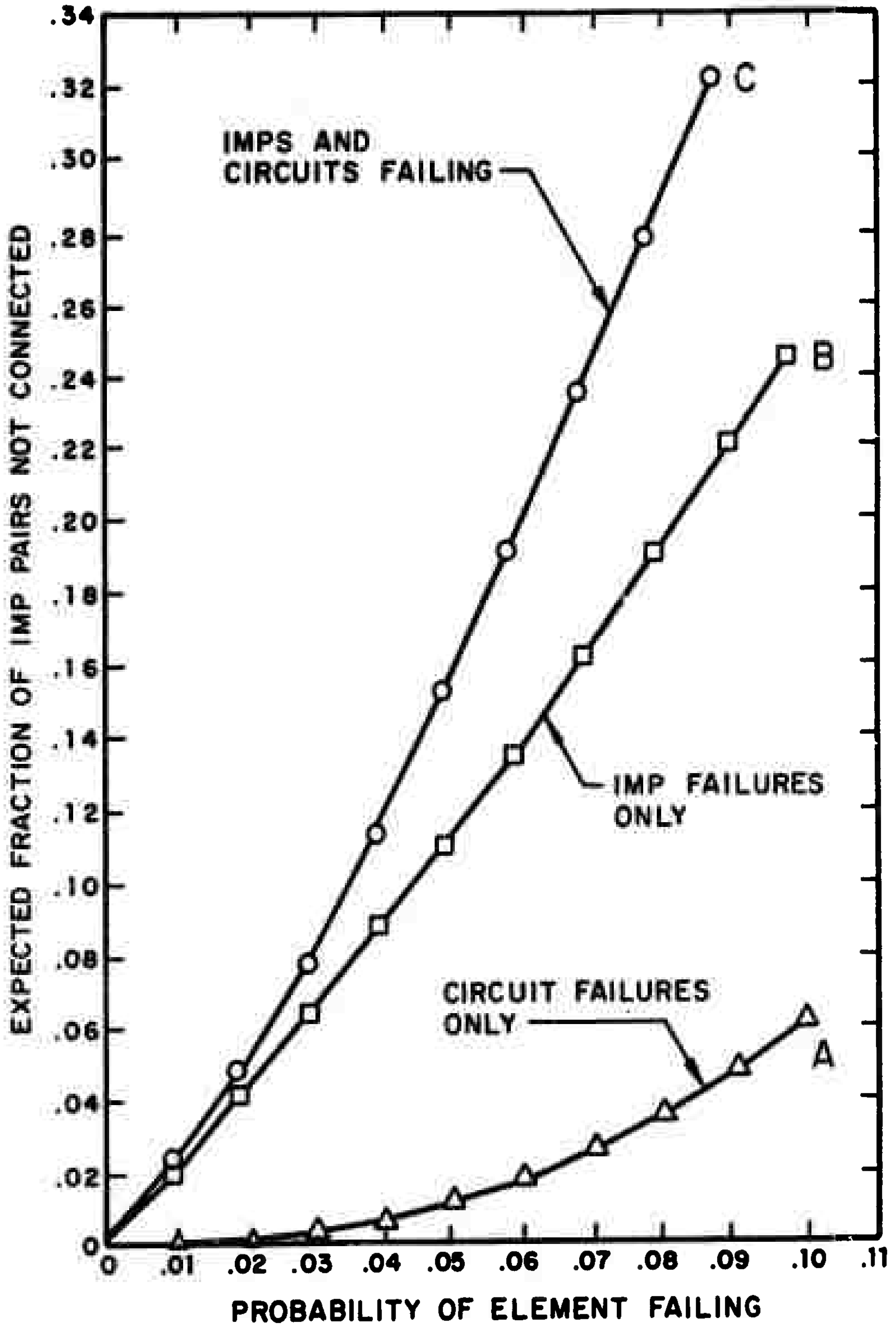


24-IMP NETWORK - 4/1/72
(d)



26-IMP NETWORK - PLANNED
(e)

FIGURE 1



EXPECTED FRACTION OF IMP PAIRS NOT CONNECTED
PROBABILITY OF ELEMENT FAILING
IMPS AND CIRCUITS FAILING
IMP FAILURES ONLY
CIRCUIT FAILURES ONLY
A
B
C
0
.02
.04
.06
.08
.10
.12
.14
.16
.18
.20
.22
.24
.26
.28
.30
.32
.34
0
.01
.02
.03
.04
.05
.06
.07
.08
.09
.10
.11

FIGURE 2

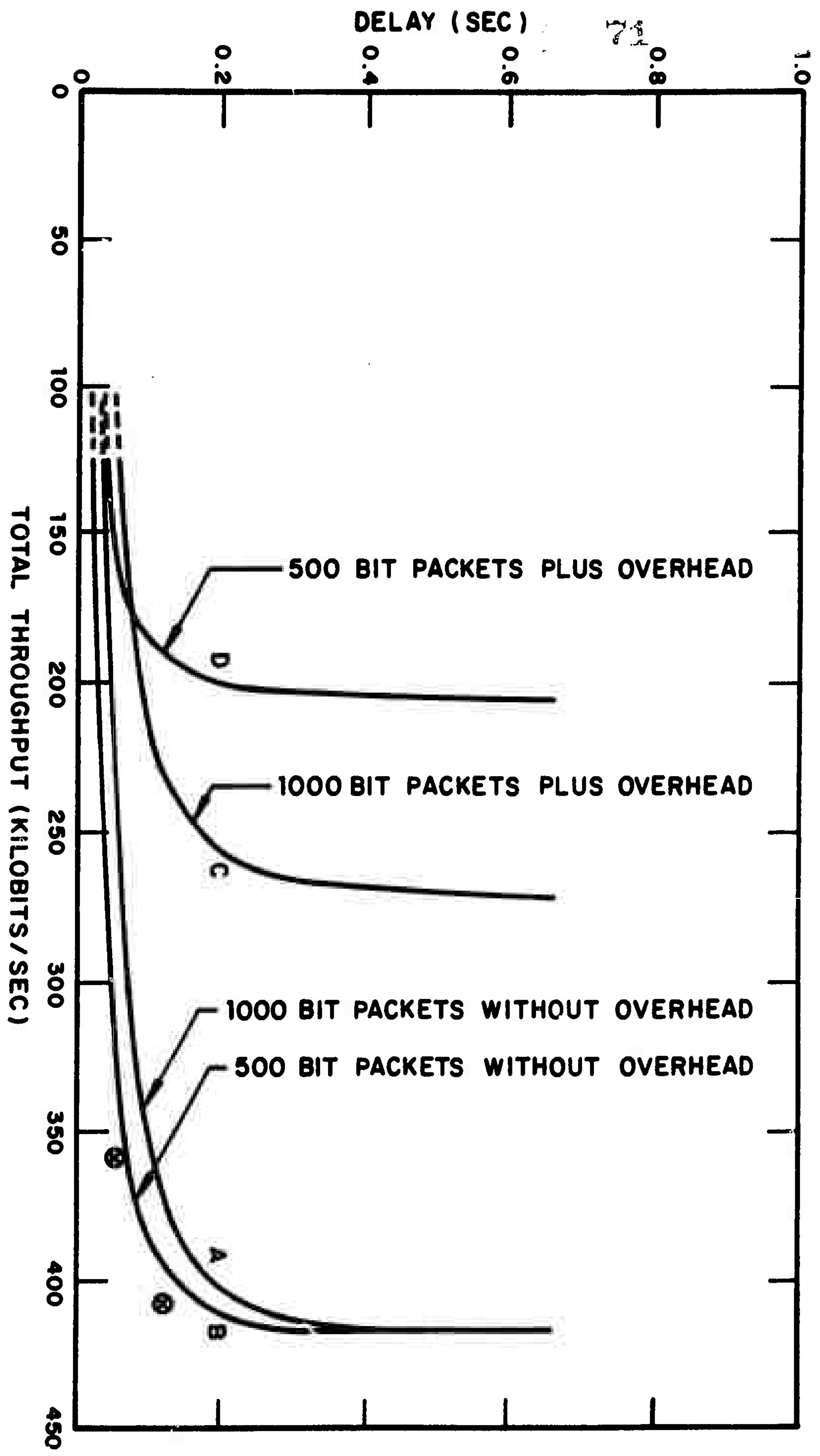

DELAY (SEC)
0
0.2
0.4
0.6
0.8
1.0
TOTAL THROUGHPUT (KILOBITS/SEC)
0
50
100
150
200
250
300
350
400
450
500 BIT PACKETS PLUS OVERHEAD
1000 BIT PACKETS PLUS OVERHEAD
1000 BIT PACKETS WITHOUT OVERHEAD
500 BIT PACKETS WITHOUT OVERHEAD
D
C
A
B

FIGURE 3

TABLE 1

23 NODE 28 LINK ARPA

Number of Circuits Failed	Number of Combinations to be Examined	Number of Cutsets
28	1	Same as Number of Combinations
27	28	
26	378	
25	3276	
24	20475	
23	98280	
22	376740	
21	1184040	
20	3108105	
19	6906900	
18	13123110	
17	21474180	
16	30421755	
15	37442160	
14	40116600	
13	37442160	
12	30421755	
11	21474180	
10	13123110	
9	6906900	
8	3108108	
7	1184040	
6	376740	349618
5	98280	≈ 70547
4	20475	≈ 9852
3	3276	827
2	378	30
1	28	0

APPENDIX C

THE FLOW DEVIATION METHOD: AN APPROACH TO STORE-AND-FORWARD COMMUNICATION NETWORK DESIGN

by L. Fratta, M. Gerla and L. Kleinrock

7:1

THE FLOW DEVIATION METHOD: AN APPROACH TO STORE-AND-FORWARD COMMUNICATION NETWORK DESIGN*

L. Fratta, M. Gerla and L. Kleinrock

ABSTRACT

A "flow deviation" method for the sub-optimization of multicommodity flow in networks is described. The method is very general but proves to be particularly fast and successful in store-and-forward (S/F) network problems. The flow deviation (FD) method for networks is quite similar to the gradient method for continuous functions; here the concept of gradient is replaced by the concept of "marginal" shortest path. As in the gradient method, the application of successive flow deviations leads to a local optimum. When the solution contains several local optima, randomization of the starting flow configuration attempts to locate them. Two interesting applications relative to the ARPA Computer Network are presented.

*This work was supported by the Advanced Research Projects Agency of the Department of Defense (DAHC15-69-C-0285).

THE FLOW DEVIATION METHOD: AN APPROACH TO STORE-AND-FORWARD COMMUNICATION NETWORK DESIGN*

L. Fratta, M. Gerla and L. Kleinrock

1. Introduction

In this paper we consider a sub-optimal procedure (the "flow-deviation" method) for assigning flow within store-and-forward communication networks so as to reduce cost and/or delay for a given topology and for given external flow requirements. We begin by defining the basic model below and follow that with some examples. We then discuss various approaches to the problem and then introduce and describe the "flow deviation" method. This method is evaluated under some further restrictions and is then applied to various problem formulations for the ARPA network [6], [7].

We consider a collection of nodes S_i, $(i = 1, \dots m)$ called sources, and a collection of nodes D_j, $(j = 1, \dots k)$, called destinations. We are required to route a quantity r_{ij} of type (i,j) commodity from source S_i to destination D_j, through a given network (Fig. 1).

The multicommodity flow problem consists of finding the optimal routing of all such commodities, which minimizes (or maximizes) a well-defined performance function (e.g., cost or delay), such that a set of constraints (e.g., channel capacity constraints) are satisfied.

*This work was supported by the Advanced Research Projects Agency of the Department of Defense (DAHC15-69-C-0285).

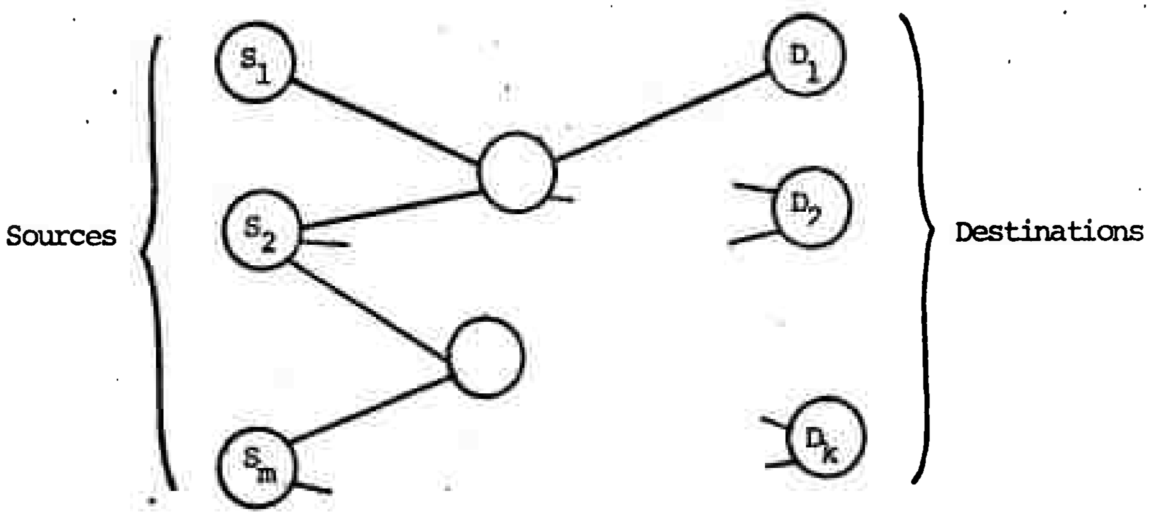


Figure 1

More generally, consider a network of n nodes N_i, $(i = 1, \ldots n)$, and suppose that a required quantity r_{ij} of type (i,j) commodity must be sent from N_i to N_j for all i and j; this more general case can be reduced to the first one by setting certain of the r_{ij} to zero (Fig. 2)

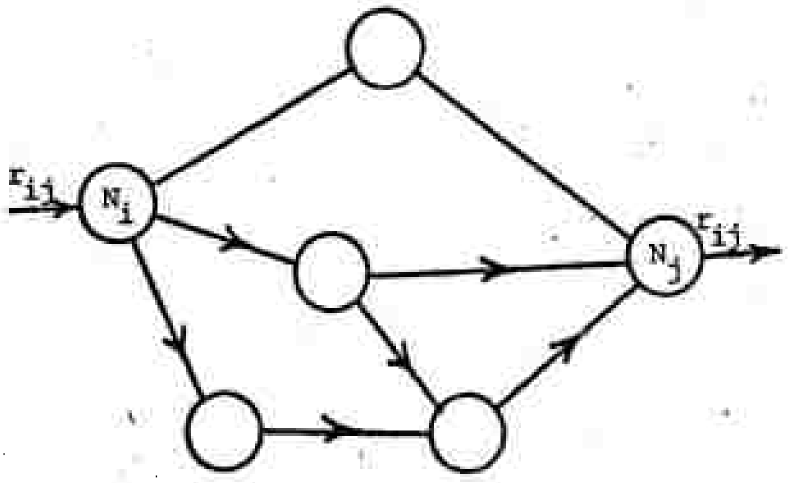


Figure 2

The multicommodity problem can formally be expressed in the following way:

Given: - a network of n nodes and b branches

- an $n \times n$ matrix $R = [r_{ij}]$, called the requirement matrix, whose entries are non-negative.

Minimize over F (or maximize)[1] $P(F)$,

where F is the flow configuration and P is a well-defined performance function.

Furthermore, F must satisfy the following constraints:

Constraints:

1. F must be feasible. Two conditions are required for feasibility:

Condition a: if the arcs of the network are "directed," we must have

$$f_{ijk\ell} \geq 0 \qquad \forall i,j,k,\ell$$

where $f_{ijk\ell}$ is the portion of commodity (i,j) flowing on the directed arc (k,ℓ). In other words, commodity (i,j) can be routed only on directed paths from N_i to N_j.[2]

[1]Without loss of generality, only the min. problem is considered in the following.

[2]The multicommodity flow configuration F is given in terms of $f_{ijk\ell}$. We note that F may be also described by a vector $F = (f_1, f_2, \dots f_b)$ where b is the total number of arcs and

$$f_m = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ijk\ell}$$

where we have labelled the arcs in such a way that arc (k,ℓ) is labelled

Condition b:

$$\sum_{k=1}^{n} f_{ijk\ell} - \sum_{m=1}^{n} f_{ij\ell m} = \begin{cases} r_{ij} & \text{if } \ell = j \\ -r_{ij} & \text{if } \ell = i \\ 0 & \text{otherwise} \end{cases}$$

In other words, this condition implies the conservation of the flows (commodity by commodity) at each node and guarantees the required flow r_{ij} between N_i and N_j.

2. F must satisfy some additional constraints, different from problem to problem (e.g., capacity constraints on each channel ~~and~~ or cost constraints).

m and f_m is the total flow (of all commodities) on branch m. As a third representation for F we may identify the routes taken by all portions of the flow requirement r_{ij} in travelling from N_i to N_j for all i and j. That is, suppose from N_i to N_j there are s such routes. Then route $\pi_{ij,k}$ has weight α_k if it carries a fraction α_k of the requirement r_{ij}, and

$$\sum_{1}^{s} \alpha_k = 1$$

In the following we use whichever of these representations is most convenient.

2. Examples of multicommodity problems

Let us consider as an example a store-and-forward (S/F) communication network [1]. In such a network, messages travelling from N_i to N_j are "stored" in queue at any intermediate node N_k, while awaiting transmission, and are sent "forward" to N_ℓ, the next node in the route from N_i to N_j, when channel (k,ℓ) permits. Thus, at each node there are different queues, one for each output channel. The message flow requirements between nodes arise at random times and the messages are of random lengths, therefore the flows in the channels and the queue lengths in the nodes are random variables. Under appropriate assumptions,[3] an analysis of the system can be carried out [1]; in particular, it is possible to relate the average delay T suffered by a message travelling from source to destination (the average is over time and over all pairs of nodes) to the average flows in the channels.

The result of the analysis is:

$$T = \sum_{i=1}^{b} \frac{\lambda_i}{\gamma} T_i \tag{1}$$

where T = total average delay per message [sec/messg]

b = # of arcs in the network

λ_i = message rate on channel i [messg/sec]

$\gamma = \sum_{i=1}^{n} \sum_{j=1}^{n} r_{ij}$ = total message arrival rate from external sources [messg/sec]

T_i = average delay suffered by a message on channel i [sec/messg]

[3]Assumptions: Poisson arrivals at nodes, exponential distribution of message length, independence of arrival processes at different nodes, independence assumption of service times at successive nodes [1].

T_i is the sum of two components:

$$T_i = T_i' + T_i''$$

where $T_i' = \dfrac{1}{\mu C_i - \lambda_i}$ = transmission and queueing delay

$T_i'' = p_i$ = propagation delay

and C_i = capacity of channel i [bits/sec]

$1/\mu$ = average message length [bits/messg]

We can rewrite (1) as follows:

(1') $$T = \frac{1}{\gamma}\sum_1^b \left| \frac{\lambda_i/\mu}{C_i - \lambda_i/\mu} + (\lambda_i/\mu)\,\mu\, p_i \right|$$

Letting $\lambda_i/\mu = f_i$, (1') becomes:

(1") $$T = \frac{1}{\gamma}\sum_1^b \left| \frac{f_i}{C_i - f_i} + f_i p_i' \right|$$

where : f_i = bit rate on channel i [bits/sec]

$p_i' = \mu p_i$

The average delay T is the most common performance measure for S/F networks, and the multicommodity problem consists of finding that routing, or flow pattern F, which minimizes T.

We may now pose two problems:

<u>Problem A:</u> "Routing assignment"

<u>given:</u> topology, channel capacities and a requirement mtx R

$$\underset{\text{over } F}{\text{minimize}}: \quad T(F) = \frac{1}{\gamma} \sum_{i=1}^{b} \left(\frac{1}{C_i - f_i} + p_i' \right) f_i$$

<u>constraints:</u>

(i) F feasible

(ii) $f_i \leq C_i$, $i = 1, \ldots b$

The problem is in the standard multicommodity form and the <u>additional</u> constraints are capacity constraints.

A second interesting problem in S/F networks is formulated below.

Assume that we have a given network topology in which the channel capacities have to be assigned. A cost is associated with the values of the capacities, and the total cost of the network is given. In addition, the flow routes must be determined.

The problem statement is:

Problem B': "Routing and capacities assignment, general cost-cap. function"

<u>given:</u> topology, requirement mtx R, number of dollars available D

$$\underset{\text{over } \underline{C}, F}{\text{minimize}}: \quad T(\underline{C},F) = \frac{1}{\gamma} \sum_{i=1}^{b} \left(\frac{1}{C_i - f_i} + p_i' \right) f_i \tag{2}$$

<u>constraints:</u>

(i) F feasible

(ii) $f_i \leq C_i$, $i = 1, \ldots b$

(iii) $\sum_{i=1}^{b} d_i(C_i) \leq D$

where $\underline{C} = (C_1, C_2 \ldots C_b)$

$d_i(C_i)$ = cost-cap. function for arc i

The minimization can be carried out first on $\underline{C}$, keeping F fixed, and then on F.

If the cost-cap. functions are linear (i.e., $d_i(C_i) = d_iC_i$), then the minimization over $\underline{C}$ can easily be performed by the method of Lagrange multipliers and we get the optimum capacities as functions of the flows [1]:

(3) $$C_i = f_i + \frac{D_e}{d_i} \frac{\sqrt{f_id_i}}{\sum_{j=1}^{b} \sqrt{f_jd_j}}$$

where $$D_e = D - \sum_{i=1}^{b} f_id_i$$

Since $$D \geq \sum_{i=1}^{b} d_iC_i \qquad \text{for (iii)}$$

and $$\sum_{i=1}^{b} d_iC_i \geq \sum_{i=1}^{b} d_if_i \qquad \text{for (ii)}$$

then $$D \geq \sum_{i=1}^{b} d_if_i$$

and $$D_e = D - \sum_{i=1}^{b} d_if_i \geq 0 \qquad \text{(iv)}$$

It is easy to see from (3) that (iv) implies also (ii) and (iii), hence both (ii) and (iii) can be replaced by (iv).

By introducing the expression of the optimum capacities given by (3) into (2), we obtain:

<u>Problem B:</u> "Routing and capacities assignment, linear cost-cap. function"

<u>given:</u> topology, requirement mtx R, # of dollars D

<u>minimize over F</u>:
$$T(F) = \frac{\left(\sum_{i=1}^{b} \sqrt{f_i d_i}\right)^2}{\gamma D_e} + \frac{1}{\gamma} \sum f_i p_i'$$

<u>constraints:</u>

(i) F feasible

(ii) $D_e \geq 0$

Again the problem is reduced to an optimal flow problem of the standard multicommodity form. The <u>additional</u> constraint is now a cost constraint. These two examples will be referred to in the sequel as Problem A and Problem B, and will be further investigated at the end of the paper.

3. Approaches to the Multicommodity Problem for Some Special Cases

Linear programming approach

a. With a linear objective function and linear constraints, the problem can be solved with linear programming techniques by taking advantage of the decomposition principle [9].

Any feasible flow F can be expressed as a convex combination of w non-bifurcated[4] feasible flows Φ_i, (i = 1, ... w), i.e.:

$$F = \sum_{i=1}^{w} \alpha_i \Phi_i$$

where $\sum_{i=1}^{w} \alpha_i = 1$. It can be shown that $w \leq b + 1$, where b is the number of arcs [9]. Suppose that the Φ_i's are known, then the variables of the linear program are the $\{\alpha_i\}$, and the feasibility constraint on F is replaced by:

$$\sum_{i=1}^{w} \alpha_i = 1$$

The number of possible candidates Φ_i for the optimal solution is enormous[5]; however, during the application of the revised simplex method, only a small

[4] A flow configuration is non-bifurcated if commodity (i,j) is flowing on one path only, $\forall$ i,j.

[5] It is equal to the number of all the combinations of paths from i to j, $\forall$ i,j.

number of candidates is considered, using shortest path criteria in the selection of the pivot column.[5'] The optimal solution can be expressed as a set of Φ_i and α_i, $i = 1, \ldots, w$.

b. With a non-linear performance function, but still with linear constraints, the decomposition method can be applied recursively by linearizing the performance function around the new optimal solution at each iteration. The procedure, however, often becomes very time-consuming.

We make two useful observations at this point:

Observation #1:

The multicommodity problem, as it appears in standard form, is a constrained optimization problem and constrained optimization methods (e.g., linear programming) are therefore required. However, if we represent the flow F as a convex combination of flows Φ_i with weights α_i, the feasibility constraint on F is replaced by the convexity constraint $\sum \alpha_i = 1$. Furthermore, if we introduce appropriate penalty functions which account for the additional constraints (e.g., capacity or cost constraints as shown below), we arrive at a problem where the only remaining constraint is:

$$\sum_{i=1}^{w} \alpha_i = 1$$

[5']Private communication from Professor D. Cantor, Department of Mathematics, UCLA.

It will be shown later, however, that in many practical cases the optimum flow can be approximated with a single non-bifurcated flow; thus, all α_i's are zero but one, which is unity, and the constraint can be disregarded. The problem becomes an unconstrained problem, and methods of unconstrained optimization, similar to the gradient method, much faster than the constrained ones, are available.[6]

On the other hand, if the optimal solution cannot be approximated by only one non-bifurcated flow, and more than one route must be considered between each pair of nodes, than the problem can be successfully solved with Lagrange multipliers.[7]

Observation #2:

Consider Problem A and Problem B, two typical S/F problems. One notices that the performance function T goes to ∞ when the flow F approaches the boundary defined by the additional constraints (i.e., when any channel becomes saturated in Problem A, or when the excess D_e reduces to zero in Problem B).

This result is quite general for S/F networks, because, when F approaches the boundaries of the constraint set, some saturation usually occurs, the queues at the nodes grow large and the delay T increases rapidly. Thus, in the S/F case, the penalty functions mentioned in Observation #1 are, as a rule, already incorporated in the performance measure.

[6]Of course, we are still faced with the problem of finding a permissible starting flow configuration; the problem is analogous to the Phase I of the linear programming [9] and will be discussed later.

[7]We are now developing at UCLA an algorithm that will take care of this more general case.

If the above-mentioned properties hold, then the multicommodity problem becomes an unconstrained optimization problem and, as such, can be solved by the flow deviation (FD) method, an unconstrained optimization method.

4. The Philosophy of FD Method

Suppose we have successfully transformed the original multicommodity problem into an unconstrained minimization problem. We may then apply the FD method, an unconstrained optimization method which, in many respects, resembles the gradient method. In order to point out the similarities, let us apply the gradient method to the following minimization problem:

(continued)

$$\min \; f(\underline{x}) \;,\; \underline{x} \in X$$

where X is an n-dimensional vector space. If $f(\underline{x})$ is continuous and differentiable in X, and the n first partial derivatives are continuous, then the gradient:

$$\nabla f(\underline{x}) \Big|_{\underline{x} = \underline{x}_o}$$

where $$\nabla \equiv (\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \ldots, \frac{\partial}{\partial x_n})$$

gives the direction of maximum increase of $f(\underline{x})$ at $\underline{x}_o$.

As we have a minimization problem, the gradient method consists of taking a step of proper size in the direction:

$$-\nabla f(\underline{x}) \Big|_{\underline{x} = \underline{x}_o}$$

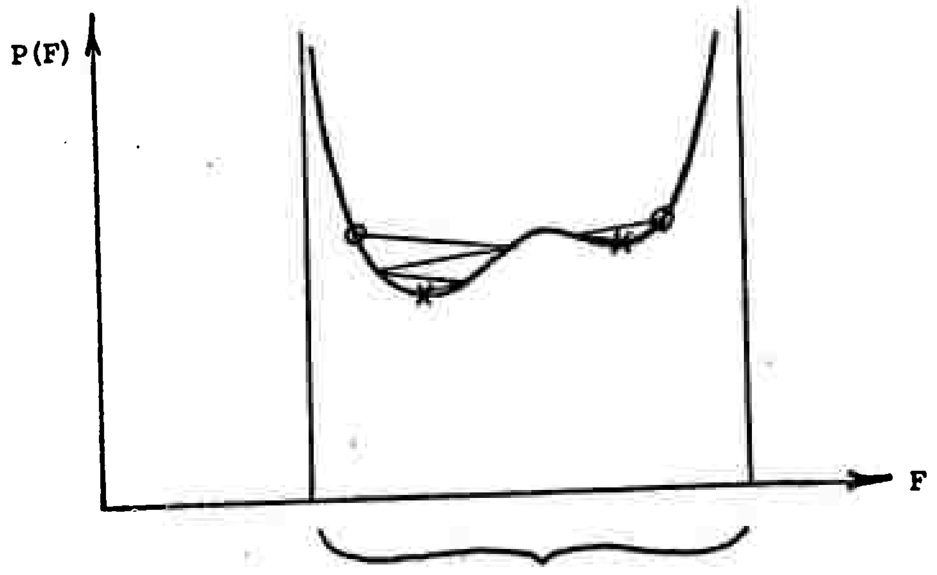
In the network case, the flow F corresponds to the variable $\underline{x}$. Suppose for simplicity that F is a non-bifurcated flow. Then F may be described as a collection of routes, one for each pair of nodes. Let π_{ij} represent the route between N_i and N_j. Then the performance function can be expressed as:

$$P(F) = P(\{\pi_{ij}\})$$

The "marginal Shortest Path (SP) matrix," which will be defined later, provides a collection of new routes, π'_{ij}, $\{\pi'_{ij}\}$, the paths of maximum decrease of $P(F)$ for infinitesimal deviation of flows. In other words, $P(F)$ decreases most rapidly by deviating an infinitesimal portion of the commodity (i,j) from π_{ij} to π'_{ij}, for all i and j.

Thus, the marginal SP mtx in network flows has the same interpretation as the gradient for a function of the continuous variable $\underline{x}$.

In the most general case of a problem with several local minima, the performance function P(F) can be schematically represented as in Fig. 3.



constraint set

⊙ permissible starting solutions

X local optima

Figure 3

Our method consists of finding a permissible starting configuration for the flow F and then applying successive FD's until a local optimum is reached.

In the case of several local optima, a randomization of the starting configuration is required [2]; after repeating the optimization with many different initial flow configurations, the minimum of the local minima so found may be chosen as a sub-optimal solution for the minimum of $P(F)$ over F.

5. Description of the Flow Deviation Method

Suppose we want to minimize a performance function $P(F)$, which already incorporates the "penalty functions" as discussed above in observation 2, and we have a feasible, non-bifurcated flow configuration F_0 (see Fig. 4).

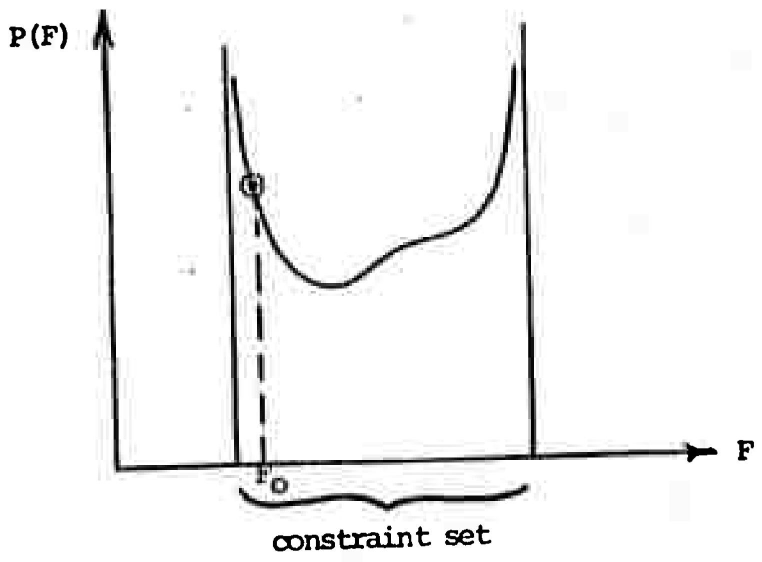


Figure 4

With this assumption, in F_0 the commodity (i,j) flows from N_i to N_j on one path only. We define a flow deviation ϕ around F_0 as a flow configuration in which there are two paths associated with each commodity (i,j) (i.e. with each pair of nodes N_i and N_j): one is the old path π_{ijo}, present in F_0, the other is π_{ij1}, the new path (see Fig. 5). The flow deviation Φ is obtained by inducing a circulation ϕ_{ij} of commodity (i,j) from N_i to N_j along path π_{ij1} and from N_j to N_i along π_{ijo}, for all i and j. If $\phi_{ij} \leq r_{ij}$, $\forall$ i,j, then the flow union $F_0 \oplus \Phi$ is also a feasible flow configuration (see Fig. 5).

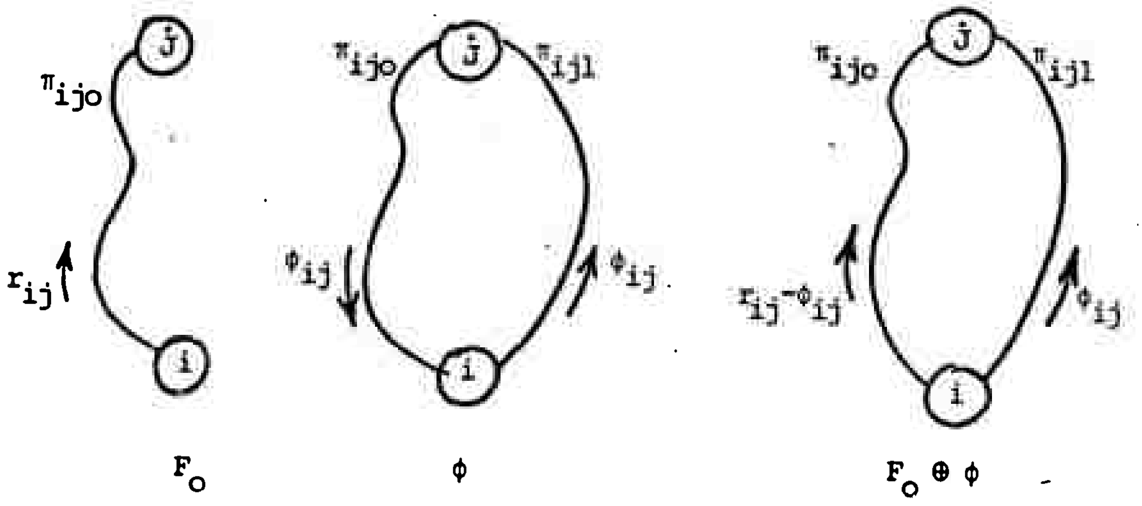


Figure 5

An <u>infinitesimal flow deviation</u> $\delta\Phi$ consists of infinitesimal $\delta\phi_{ij}$, $\forall$ i,j.

Consider now the variation of the performance function corresponding to an infinitesimal flow deviation $\delta\Phi$:

$$\delta P(F_0) = P(F_0 \oplus \delta\Phi) - P(F_0)$$

We wish to find $\delta\Phi$ for which $\delta P(F_0)$ is minimum, i.e., the deviation of maximum decrease.[8] But:

$$P(F_0) = P(f_1, f_2, f_3, \ldots f_b)$$

and

$$\delta P(F_0) = \sum_{k=1}^{b} \left(\frac{\partial P}{\partial f_k}\right)_{F=F_0} \delta f_k \tag{4}$$

where b is the number of arcs and δf_k is the infinitesimal flow variation in arc k, induced by $\delta\Phi$.

Referring to Fig. 5, let us define the indicator functions ξ_{ij}^k and ν_{ij}^k

$$\xi_{ij}^k = \begin{cases} 1 & \text{iff } \pi_{ij0} \text{ includes arc } k \\ 0 & \text{otherwise} \end{cases}$$

$$\nu_{ij}^k = \begin{cases} 1 & \text{iff } \pi_{ij1} \text{ includes arc } k \\ 0 & \text{otherwise} \end{cases}$$

Thus $\{\xi_{ij}^k\}$ indicates the <u>old routes</u>, relative to F_0 and $\{\nu_{ij}^k\}$ indicates the <u>new routes</u>, created by the deviation around F_0. It is easy to see that, in the non-bifurcated flow F_0, f_k is given by:

$$f_k = \sum_{i=1}^{n} \sum_{j=1}^{n} r_{ij}\, \xi_{ij}^k$$

[8] We recall that, without loss of generality, only the minimization problem is considered.

The variation δf_k of the flow in the arc k after the deviation is therefore:

(5) $$\delta f_k = \sum_{i=1}^{n} \sum_{j=1}^{n} \delta\phi_{ij} \left(\nu_{ij}^k - \xi_{ij}^k\right)$$

From (4) and (5) we get:

$$\delta P = \sum_{k=1}^{b} \left(\frac{\partial P}{\partial f_k}\right)_{F_0} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\nu_{ij}^k - \xi_{ij}^k\right) \delta\phi_{ij}$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{n} \delta\phi_{ij} \sum_{k=1}^{b} \left(\frac{\partial P}{\partial f_k}\right)_{F_0} \left(\nu_{ij}^k - \xi_{ij}^k\right)$$

For <u>fixed step size</u> $\delta\phi_{ij}$, we may minimize δP by properly choosing the new routes $\{\nu\}$. Hence, our problem becomes:

(5a) $$\min_{\text{over } \{\nu\}} \delta P = \sum_{i=1}^{n} \sum_{j=1}^{n} \delta\phi_{ij} \min_{\text{over } \{\nu\}} \left| \sum_{k=1}^{b} \left(\frac{\partial P}{\partial f_k}\right)_{F_0} \left(\nu_{ij}^k - \xi_{ij}^k\right) \right|$$

For each arc k, we now introduce a <u>key definition of length</u> ℓ_k as follows:[9]

(5b) $$\ell_k = \left(\frac{\partial P}{\partial f_k}\right)_{F=F_0}$$

Thus, for each (i,j) pair we have to minimize:

[9]The case of $\partial P/\partial f_k < 0$ (and therefore of $\ell_k < 0$) could create negative cycles. The occurrence of negative cycles, however, is of no importance in the practical case of S/F networks, since the delay is an <u>increasing</u> function of the flow in an arc and therefore $\ell_k = (\partial T/\partial f_k) > 0$.

(5c) $$\min_{\{\nu\}} \left| \sum_{k=1}^{b} \left(\frac{\partial P}{\partial f_k}\right)_{F_0} \left(\nu_{ij}^{k} - \xi_{ij}^{k}\right)\right| = \min_{\{\nu\}} \left| \sum_{k=1}^{b} \ell_k \left(\nu_{ij}^{k} - \xi_{ij}^{k}\right)\right|$$

$$= \left(\min_{\{\nu\}} \sum_{k=1}^{b} \ell_k \nu_{ij}^{k}\right) - \sum_{k=1}^{b} \ell_k \xi_{ij}^{k}$$

If we interpret ℓ_k as the length of the arc k, then

$$\min_{\{\nu\}} \sum_{k} \ell_k \nu_{ij}^{k}$$

is the shortest distance from node i to node j (i.e., the length of the shortest path connecting i to j) under the metric ℓ. The shortest path (SP) from i to j therefore defines the selected path for deviation of flow (called deviation path).

After the deviation paths are found for all pairs of nodes,[10] the next step is to determine the $\{\phi_{ij}\}$, i.e., the size of the deviation. A general algorithm based on the FD method is outlined as follows:

1. Find a feasible starting F_0.[11]
2. Let initially $n = 0$.
3. Compute the SP mtx for F_n.
4. Determine optimum deviation size and compute F_{n+1} (see discussion below).
5. If $[P(F_n) - P(F_{n+1})] > \varepsilon$ where ε is an acceptable threshold on the performance improvement, let $n \to n + 1$ and go to 3. Otherwise stop.

[10]A very simple algorithm, due to Floyed [9], gives the shortest route, i.e., the deviation path, for all pairs of nodes, in the form of a shortest path (SP) mtx. The amount of computation required is on the order of n^3, where n is the number of nodes.

[11]See the application section for examples.

6. Limitations of the Method

a. Even starting with a non-bifurcated flow F_0, we notice that, at the end of the first iteration, the flow from i to j takes two routes π_{ij0} and π_{ij1}; on π_{ij0} the amount of commodity (i,j) is $(r_{ij} - \phi_{ij})$, on π_{ij1} it is ϕ_{ij} (see Fig. 5).

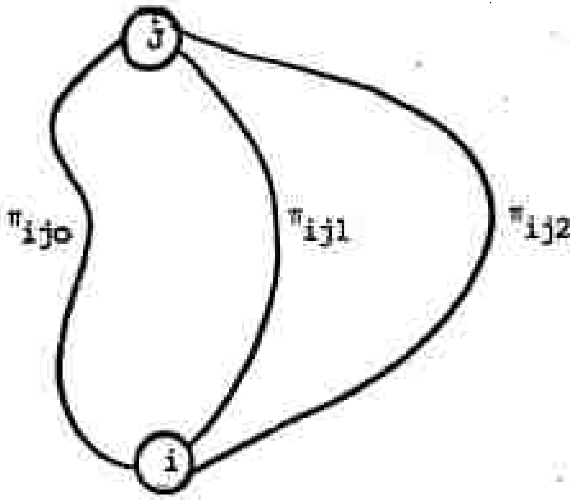


Figure 6

At the next iteration, as the flow configuration F has changed, also the metric ℓ_k changes (see Eq. (5b)) and a new shortest path π_{ij2} from i to j is found (see Fig. 6). Suppose π_{ij2} is different from π_{ij0} and π_{ij1}. The problem is now how to define the FD, i.e., how to redistribute the requirement r_{ij} amoung the <u>three</u> paths. Since each iteration adds a new path (typically) to each pair of nodes, we realize that a criterion for this redistribution is needed. A reasonable criterion is the following: Suppose there are n routes on which the requirement flows from i to j. Choose the longest of these n routes under the new metric ℓ_k and define the deviation as a deviation of flow from the longest to the shortest path. Other criteria can be devised, but none of them (including this one) guarantees optimality.

b. The existence of many routes between each pair of nodes represents a problem also from the point of view of the description of the flow. We must keep a record of all the routes with the associated weights (flows that they carry), and, if the network is large, the problem of storing these data in the computer during solution can be severe.

A suboptimal criterion to keep the number of parallel routes $\leq M$ is the following: If there are $m > M$ routes from i to j, eliminate those $m - M$ with smallest flows.

c. After having resolved points (a) and (b) with some suboptimal criterion, we are still faced with choosing the deviations ϕ_{ij}. In the gradient method the determination of the step size is a one-dimensional optimization problem. Here we have $n(n - 1)$ variables ϕ_{ij} (where n is the number of nodes in the network). Furthermore, the presence of the penalty functions in $P(F)$ guarantees that the constraints are satisfied only for a small deviation $\delta\Phi$. Large ϕ_{ij} might produce a flow F_1

$$F_1 = F \oplus \Phi$$

which is outside the constraint set. Thus the determination of the optimum ϕ_{ij} represents a fairly complex problem.[12]

7. The FD Method for Large Networks with Balanced Traffic Requirement

It can be shown that the limitations mentioned in the preceding section have practically no effect on large networks with a balanced traffic requirement. The definitions of "large" network and of "balanced traffic" are more qualitative than quantitative, and they vary from problem to problem.

We begin by discussing the notion of "balanced" networks. Let us consider a network with n nodes. If its associated graph is complete (i.e., all pairs of nodes are connected by directed arcs), then the number of arcs is $n(n - 1)$. Suppose we restrict our analysis to graphs which have Kn arcs with fixed $K \leq n - 1$; K represents the average arc to node density of the graph.[13]

Let $r = \dfrac{\sum_{ij} r_{ij}}{n(n-1)} = \dfrac{\gamma}{n(n-1)}$ be the average flow requirement/pair of nodes

Let $m = \max_{ij} [r_{ij}/r]$

m gives a measure of how balanced the traffic is. Notice that $m \geq 1$ always and that $m = 1$ corresponds to a uniform traffic requirement matrix

[12] The rigorous approach to this problem is to consider the global flow as a convex combination of non-bifurcated flows and to apply the method of the Lagrange multipliers. This approach is now under investigation here at the Computer Science Department of UCLA.

[13] In many practical cases, the number of arcs grows linearly with the number of nodes.

($r_{ij} = r$ for all $i \neq j$); m is large for very diversified requirements.[13']

Let us now relate these ideas to the notion of "large" networks. Let $\bar{f}$ be the average channel flow in the network; $\bar{f}$ is given by:

$$\bar{f} = \frac{(\text{total flow in the net})}{(\#\text{ of arcs})} = \frac{nr(n-1)\bar{n}}{Kn} = \frac{r(n-1)\bar{n}}{K}$$

where $\bar{n}$ is the average number of arcs in a path (typically, it is an increasing function of n), and is commonly referred to as the average path length.

Suppose the maximum requirement is routed on one path only. Then the ratio η between the maximum requirement and the total average channel flow is

$$\eta = \frac{mr}{\bar{f}} = \frac{mrK}{r(n-1)\bar{n}} = \frac{mK}{(n-1)\bar{n}} \tag{6}$$

If we have a "large" net with "balanced" traffic, then $\eta \ll 1$. In this situation one notices that:

1. The ratio between each commodity flow and the total flow in a channel is $\ll 1$, therefore the deviation of the commodity from the old path to the new SP can be considered as <u>infinitesimal</u>. For this reason, limitation (c) mentioned in the preceding section can be neglected and both improvement in $P(F)$ and satisfaction of the constraints are virtually guaranteed.

[13'] Many other appropriate definitions of m are possible, for example, $m' = \left[\sum\left(1 - \frac{r_{ij}}{r}\right)^2\right]^{1/2}$, in which case $m' = 0$ corresponds to the uniform traffic requirement.

2. Again, as the deviation can be considered as infinitesimal, it does not pay much to split the requirement r_{ij} between more than one route from i to j. Thus the optimum flow is approximately non-bifurcated and, in the application of the algorithm, we can deviate the entire commodity from the old path to the new shortest path at each iteration. In this way the limitations (a) and (b) of the FD method can be neglected. As most of the S/F networks can be classified as "large" and "balanced," we will consider only this category in the applications section.

8. Case of P(F) Convex. The Routing Assignment

Suppose $P = P(f_1, f_2 \ldots f_b)$ is a convex function with respect to each of the f_i's (see Fig. 7): This is the case of Problem A (the routing assignment), mentioned in Section 2. If P(F) is convex, then the equivalent length ℓ_i of the arc i, given by $\partial P/\partial f_i$, is an increasing function of f_i. The fact that ℓ_i increases with f_i corresponds to a tendency of the FD method towards _spreading_ the flows in the network. If, after an iteration of the algorithm, many new routes include the arc i, then f_i becomes large, and so does the equivalent length ℓ_i. This effect tends to block the further increase of f_i and to make the other arcs appear more favorable in the next iteration. From an intuitive point of view one can see that, due to this spreading tendency the FD method drives the flow to a unique final configuration, no matter what starting configuration was chosen.[14] This intuitive argument is substantiated by the mathematical

[14] This is rigorously true only if the flow deviations are _infinitesimal_. In the practical cases the size of the steps is _finite_, and the minimum flow can be determined only with an accuracy proportional to the step size. For the same reason, different starting configurations might produce different final flow configurations, whose "distance" from the optimal solution is again proportional to the step size.

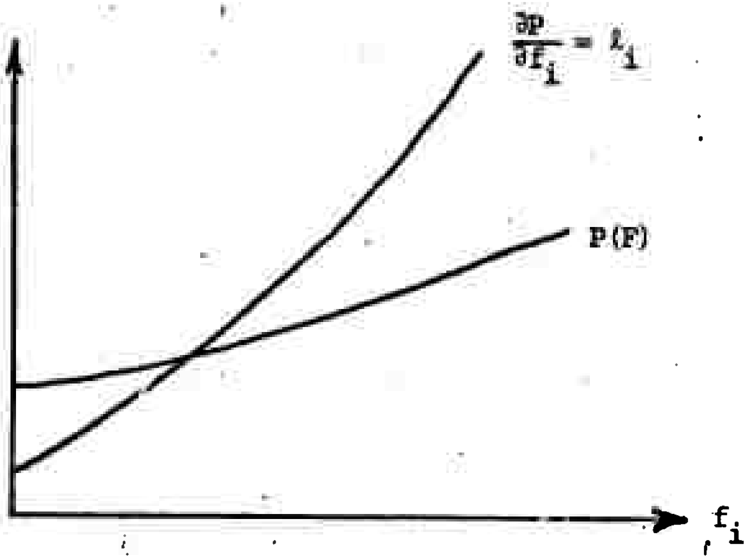


Figure 7

argument that, in a problem with convex performance function and convex constraint set (both <u>feasible</u> and <u>additional</u> constraints define a convex constraint set), a local minimum is also a global minimum. In order to understand how the FD method operates in the case of P(F) convex, let us consider its application to Problem A. Let π_{ij0} be the old route and π_{ij1} be the new shortest route between nodes N_i and N_j (see Fig. 8a), and let α be the amount of commodity (i,j) deviated from π_{ij0} to π_{ij1}. From (1") it is easy to see that $P(\alpha)$, a sum of convex functions of α, is also convex in α (see Fig. 8b). The fact that π_{ij1} is "shorter" than π_{ij0} corresponds to a negative slope at $\alpha = 0$ (see Fig. 8b): hence the FD method deviates an amount of commodity (i,j) in the proper direction from the path π_{ij0} to the path π_{ij1}.[14']

[14'] The amount of commodity deviated depends on the particular algorithm used: for example, if the "large and balanced net" assumption holds, then a "non-bifurcated" algorithm can be applied and the total amount is deviated.

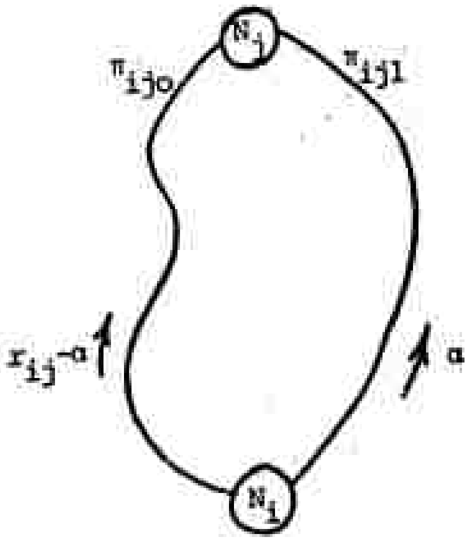


Figure 8a

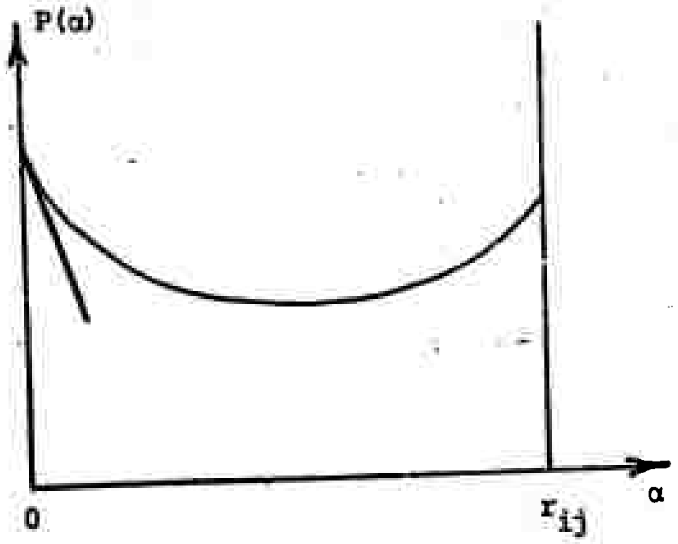


Figure 8b

9. <u>Case of $P(F)$ Concave: The Routing and Capacities Assignment</u>

Suppose $\underline{P} = P(f_1, f_2, \ldots f_b)$ is concave with respect to each of the f_i's (see Fig. 9).

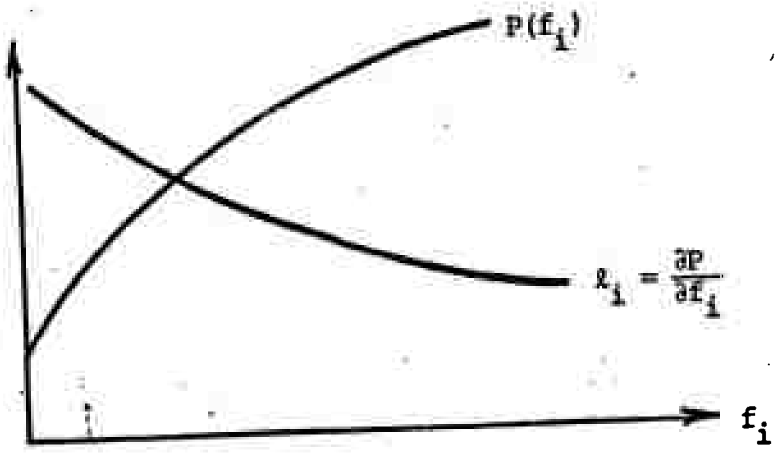


Figure 9

Then, the equivalent length $\ell_i = \frac{\partial P}{\partial f_i}$ is decreasing with f_i, and this corresponds to a tendency of the FD method to concentrate the flows in the network. If, after an iteration of the algorithm, many new routes include the arc i, then f_i increases, ℓ_i decreases and therefore, on the next iteration, arc i will probably be included in more SP's.

As it will be shown in the Applications Section, Problem B (routing and capacities assignment, linear cost-cap case) has P(F) concave. In that case we know that we can reduce the average delay T by concentrating the capacities [1]; so the FD method actually works in this direction, because it concentrates the flows, and therefore the capacities (see Eq. (3)). It can further be shown for Problem B that P(F) is also a concave function of α, where α is the deviation applied to a pair of nodes i,j (see Fig. 10):

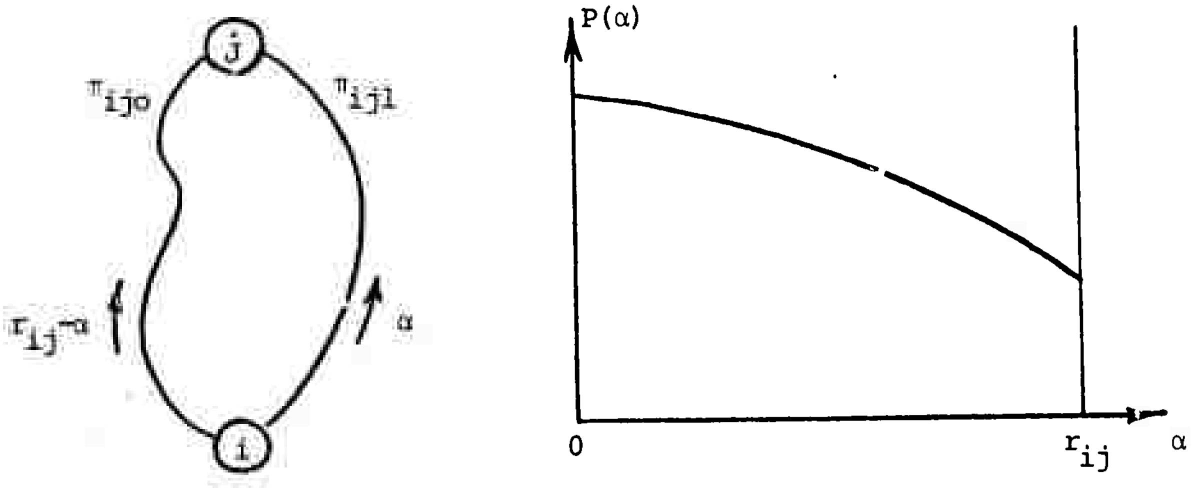


Figure 10

In the case of Fig. 10, if the flow was initially in π_{ij0}, and if the SP mtx tells us that π_{ij1} is shorter than π_{ij0} under the metric ℓ_k, then the FD method deviates all the flow from π_{ij0} to π_{ij1}.

An interesting property of the case of $P(F)$ concave with respect to the flow deviation α is that, even without the "large" and "balanced" net assumption, the optimal flow F is not bifurcated. Let us prove this strong property:

Suppose the optimal solution has two routes from i to j, π_{ij1} with weight $w_1 = r_{ij} - \alpha_0$ and π_{ij2} with weight $w_2 = \alpha_0$, $(0 \leq \alpha_0 \leq r_{ij})$

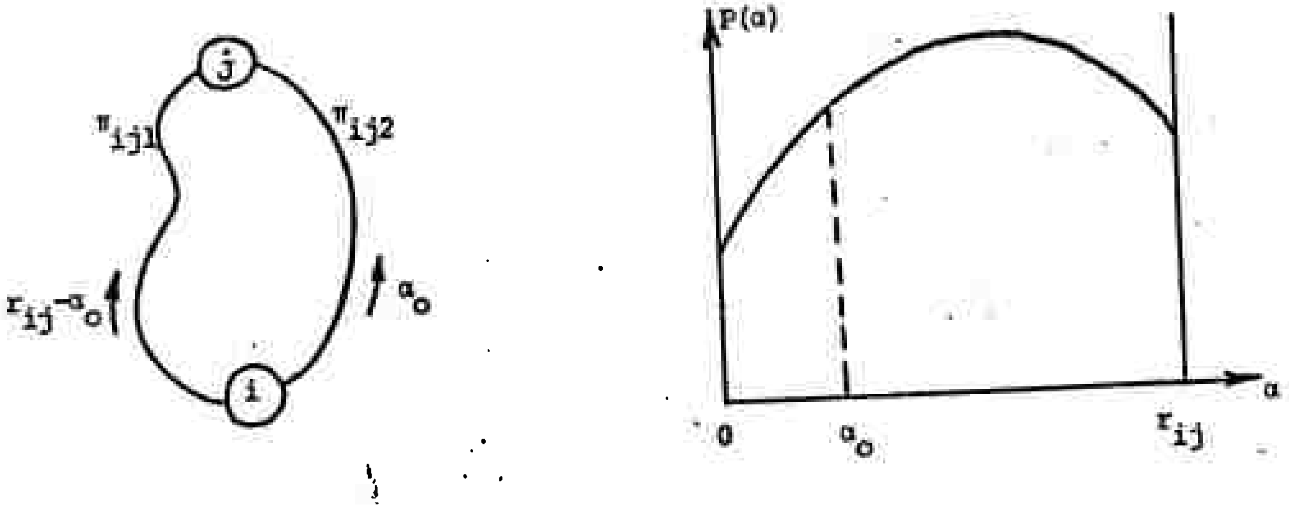


Figure 11

As we can see from Fig. 11, due to the concavity of $P(\alpha)$, the minimum is on one of the extremes of the interval $(0,r_{ij})$, and the optimal solution <u>cannot</u> be bifurcated, unless $P(\alpha)$ is constant with α. In the latter case, any value of α is optimal, extreme included. The same argument can be extended to the situation of more than two paths from i to j, by taking them two at a time. We have proved, therefore, that the optimal solution is never bifurcated in the case of $P(F)$ concave.

As far as the "optimality" of the method is concerned, the starting flow configuration and the order in which we update the flows strongly determine the local optimum reached, since the more flow we route on an arc, the more we would like to route on it at the next iteration. So, the application of the FD method results in finding <u>local</u> minima, and is not as powerful here as in the convex case. On the other hand, the non-bifurcated nature of the optimum solution greatly simplifies the algorithm in the cases where the "large" and "balanced" net assumption is not valid.

10. <u>Applications</u>

As an application of the FD method, two multicommodity problems relative to the ARPA Computer Network are presented.

The ARPA Computer Network is a S/F communication network connecting several computer facilities in the United States. A detailed description of the Network is given in Refs. [3-8]. One of the earlier proposed topologies of the Network is given in Fig. 16. The efficiency of the system is measured in terms of average delay T per message.

The first application is Problem A in Section 2: We have to find the routing of the flows in this network which minimizes T, given the capacities. The second application is Problem B: We are given the total cost of the network and we want to determine the optimum routing and the optimum assignment of the channel capacities, under linear cost-cap. functions.

Application 1: The routing assignment

From Section 2, the average delay per message T is given by:

$$T = \frac{1}{\gamma} \sum_{1}^{b} \left(\frac{1}{C_i - f_i} + p_i' \right) f_i$$

The multicommodity problem in this case is stated as follows:

given: - topology 21 nodes (see Fig. 16)

- assumed unif. requirement mtx: $r_{ij} = r$ [bits/sec]
- capacities (C_i = 50 kbits/sec, for all i)

minimize over F:

$$T(F) = \frac{1}{\gamma} \sum_{1}^{b} f_i \left(\frac{1}{C_i - f_i} + p_i' \right)$$

constraints:

(i) F feasible

(ii) $f_i \leq C_i$

We recognize that T is in the form of Problem A.

We first notice that: $\lim_{f_i \to C_i} T(F) = \infty$. So the constraint (ii) is contained in $T(F)$ as a penalty function. Next, we want to show that the "large and balanced network" condition holds. From Eq. (6), η, the ratio for the test of "large and balanced net" condition is given by:

$$\eta = \frac{m(Kn)}{n(n-1)\bar{n}}$$

In the present case:

$n = 21$

$\bar{n} \geq 1$ (lower bound on the average path length)

nK = number of channels = 52

$m = 1$ (uniform requirement)

Hence: $$\eta \leq \frac{52}{(21)(20)} \cong 0.12 << 1$$

and the test is satisfied.

The FD method can therefore be applied in the simple, non-bifurcated form corresponding to the "large and balanced net" situation.

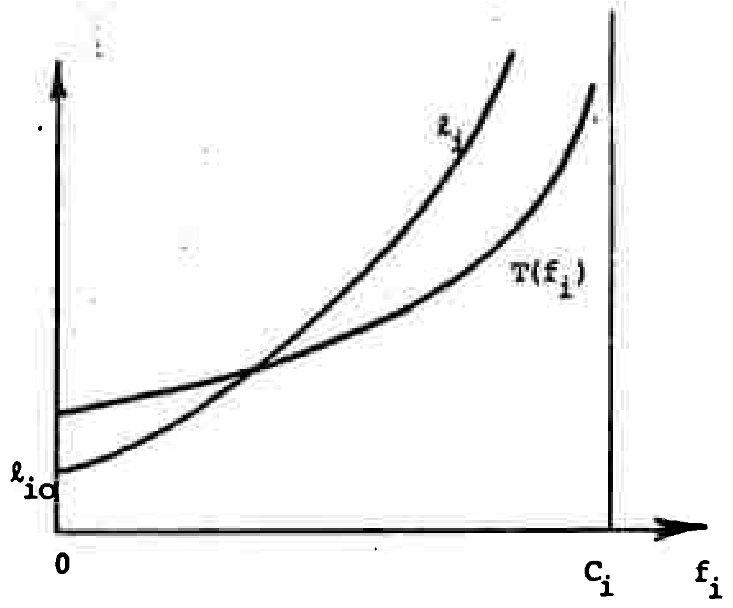


Figure 12

The equivalent length ℓ_i (see Fig. 12) is given by:

e. If $\beta_s > 1$, then we have found a feasible flow. Go to Phase 2.

f. If $\beta_s \leq 1$, we cannot send the requirement r on the routes we just found without violating some capacity constraints. In this case a requirement of $0.95\,\beta_s r$, very close to saturation is chosen. The FD method is then applied to this flow, and a new, better routing is found that raises the saturation level to $\beta_s' r$.

g. If $\beta_s' = \beta_s$, no improvement occurred; $\beta_s r$ is the max level of traffic that the network can accept. Hence the problem is infeasible.

h. Suppose $\beta_s' > \beta_s$:

  h1: If $\beta_s' > 1$ go to Phase 2

  h2: If $\beta_s' \leq 1$, update and go to step f

Phase 2:

a. For the requirement r and the flow currently under consideration compute the equivalent lengths ℓ_k given by Eq. (7).

b. Compute SP matrix.

c. Give flow deviations according to SP matrix.[16]

[16] It is clear that, in order to maintain the assumption of infinitesimal deviation, it would be best to recompute the SP matrix after the deviation of each single commodity. However, the overhead due to the recomputation of the SP matrix at each step might be severe, so a compromise between accuracy and speed has to be found. In our example, the two extreme cases of updating the SP matrix after processing each commodity, and after processing all the commodities, have been investigated: the difference in the final results was less than 5%, which shows that, at least in this particular example, the updating of the SP matrix is not a very critical factor.

d. If the improvement in the performance function T is larger than a predetermined threshold ε, go to 2(a). Otherwise stop.

Figure 17 illustrates an application of the algorithm with a uniform requirement $r = 1.2$ kbits/sec. By routing the traffic according to the SP matrix computed at $\beta r = 0$, saturation occurs at $\beta_s r = 1.02$ kbits/sec, i.e. at $\beta_s = .85$ (see curve a). As $\beta_s < 1$, we are still in Phase 1 and the FD method is applied to a flow corresponding to a reduced requirement $r' = .95\, \beta_s r = .96$ kbits/sec; new routes are found (see curve b), and saturation occurs now at $\beta'_s r = 1.250$ kbits/sec, i.e., at $\beta'_s = 1.04$. As $\beta'_s > 1$, a feasible solution at $r = 1.2$ kbits/sec is available, and Phase 2 can be started. The FD method is now applied using the full traffic requirement and the optimum routing is found (see curve c). Notice that, as expected, the routing (a) gives the best performance at low traffic levels, while (c) gives the best performance at the required traffic level.

This algorithm has been programmed in FORTRAN and run on the 360/91 at UCLA. The three curves plotted in Fig. 17 were computed in 8 sec., CPU time.

<u>Application 2: Routing and capacities assignment, linear cost-cap. case</u>

It was shown in Section 2 that this problem is equivalent to the following multicommodity problem:

<u>given</u>: - topology

- assumed uniform requirement mtx

$$\underline{\text{minimize}}\text{ over } F: \quad T(F) = \frac{\left(\sum_{1}^{b} \sqrt{f_i d_i}\right)^2}{\gamma D_e} + \frac{1}{\gamma} \sum_{1}^{b} f_i p'_i$$

constraints:

(i) F feasible

(ii) $$D_e = D - \sum_{i-1}^{b} f_i d_i \geq 0$$

where d_i = cost/unit of capacity in channel i

D = total number of dollars

We first notice that:

$$\lim_{D_e \to 0} T(F) = \infty$$

So the constraint $D_e \geq 0$ is contained in $T(F)$ as a penalty function.

Next, consider the plot of T and ℓ_i versus f_i (Fig. 13), where:

$$\ell_i(f) = \frac{\partial T}{\partial f_i} = \frac{1}{\gamma}\left(\frac{\sum_{j=1}^{b} \sqrt{f_j d_j}}{D_e}\right)\sqrt{\frac{d_i}{f_i}} + \frac{1}{\gamma}\left(\frac{\sum_{j=1}^{b} \sqrt{f_j d_j}}{D_e}\right)^2 d_i + \frac{p_i'}{\gamma}$$

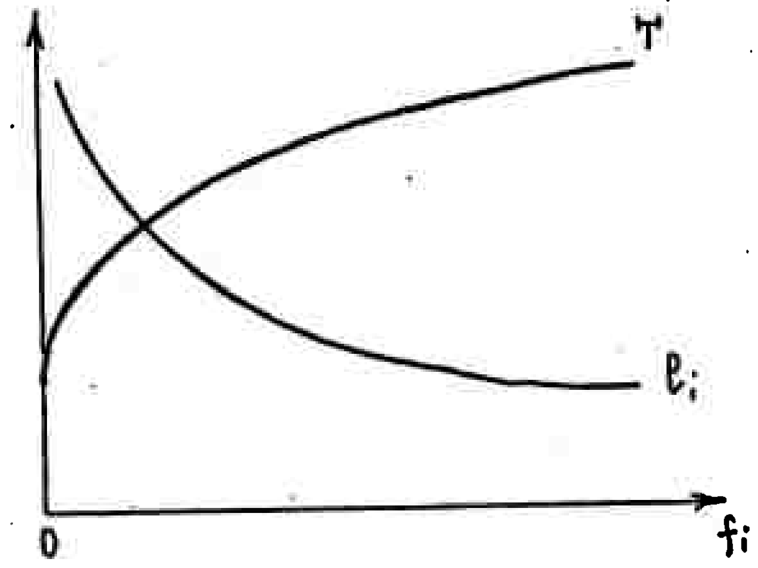


Figure 13

Note that:

$$\lim_{f_i \to 0} \ell_i(f) = \infty$$

due to the term $\frac{1}{\sqrt{f_i}}$

It can further be proved that T is concave with respect to a flow deviation α [1]. From the concavity of T we recognize that we are in the situation of Section 3 and the properties there mentioned hold. Hence the FD method can be applied in the simple, non-bifurcated form. As it was pointed out in Section 9, the FD method leads to local optima, and several starting flow configurations must be tried. In order to get several starting flow configurations, the following randomization procedure is repeated for each iteration of the algorithm.

a. Assign initial equivalent lengths ℓ_k to the channels <u>at random</u>, so that $m \le \ell_k \le M$, where m and M are respectively lower and upper limits conveniently chosen.

b. Compute SP matrix.

c. Route the requirements r on the SP's, and get the starting flow configuration F_0.

d. If $(D - \sum f_i d_i) > 0$, F_0 is feasible, the FD method can be applied.

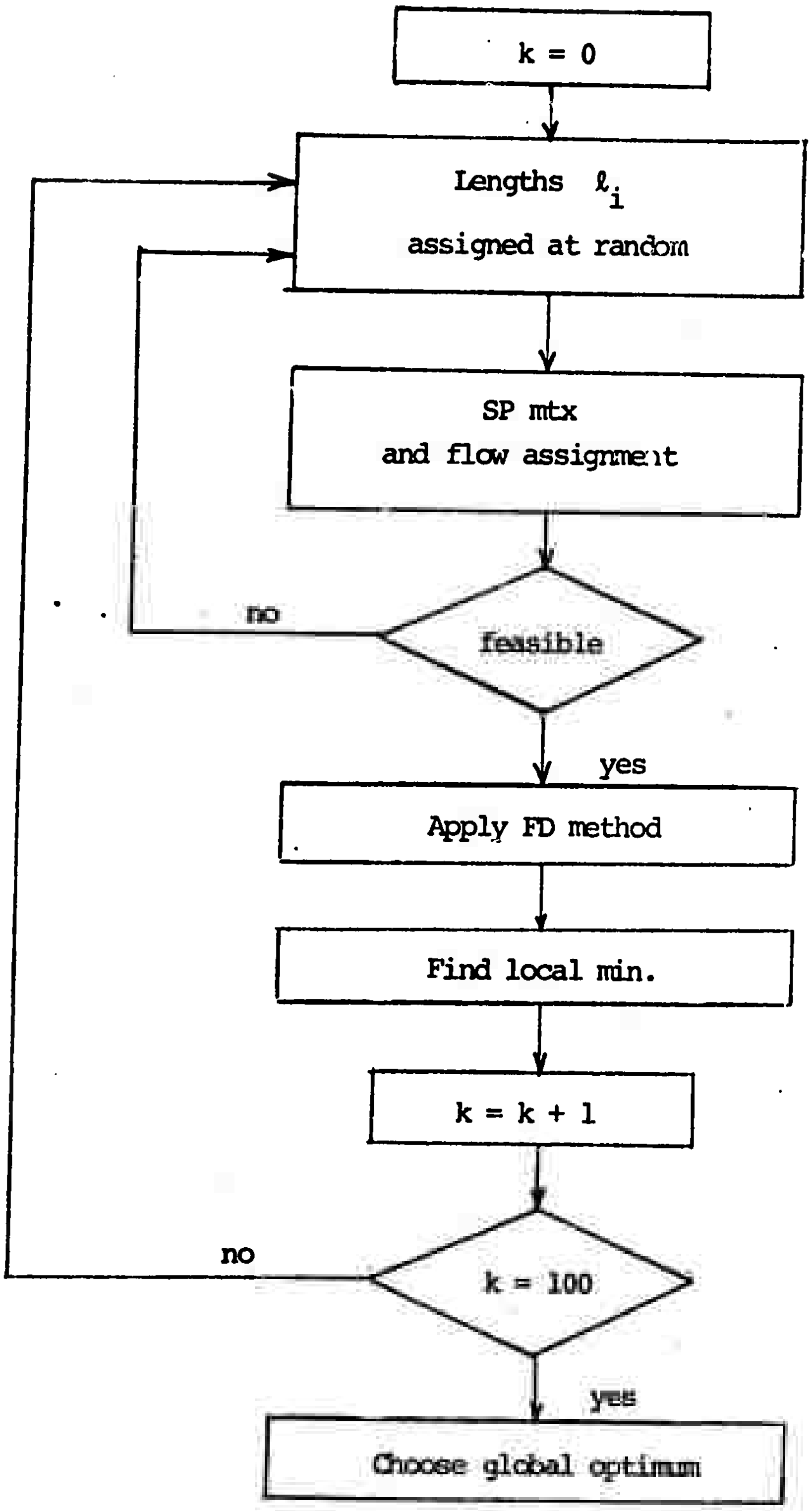
e. If $(D - \sum f_i d_i) < 0$, F_0 is unfeasible and is rejected. Go to (a).

The initial random choice of the lengths guarantees a certain randomness in the starting feasible flow, thus providing a method for finding several local minima. After a convenient number of iterations, the global minimum is chosen as the minimum of the local minima. This provides a "suboptimal" solution.

A block diagram of the method is given in Fig. 14.

Note 1

In the ARPA network case, the cost-capacity functions are step functions; in order to be able to use our method, this staircase function has been approximated by several straight lines, each used in the proper range of capacities (see Fig. 15). In this way, given a flow configuration F, a proper set of linear cost-cap. functions is chosen: during the application of the FD method, this set is updated iteration by iteration, as F changes. At the end of the "continuous" optimization, performed using linear cost-capacity functions, the continuous capacities are discretized to the upper values of capacity available (in the case of Fig. 15, $C_c \to C_d$). Passing from the continuous to the discrete solution, the delay T obviously decreases, but the cost constraint might be violated. In this case, the "local" optimal solution is rejected, or, better, is considered as a local optimum of a different problem, where the number of dollars is conveniently larger. This technique is clearly suboptimal, and it is difficult to tell how close this solution is to the real optimum. The rigorous approach, however, requires integer programming techniques and is very time consuming. Other suboptimal techniques have been proposed [7], [10], but it is still an open problem.



k = 0
Lengths ℓ_i assigned at random
SP mtx and flow assignment
feasible
no
yes
Apply FD method
Find local min.
k = k + 1
k = 100
no
yes
Choose global optimum

Figure 14

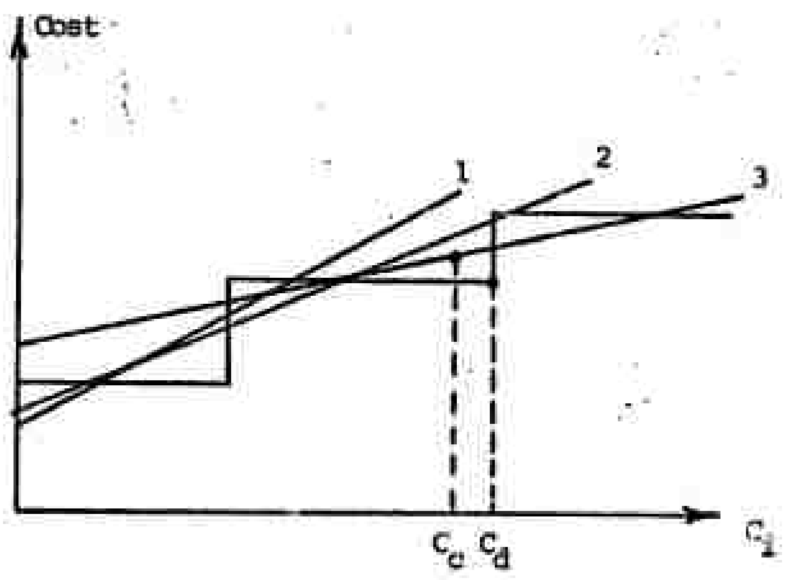


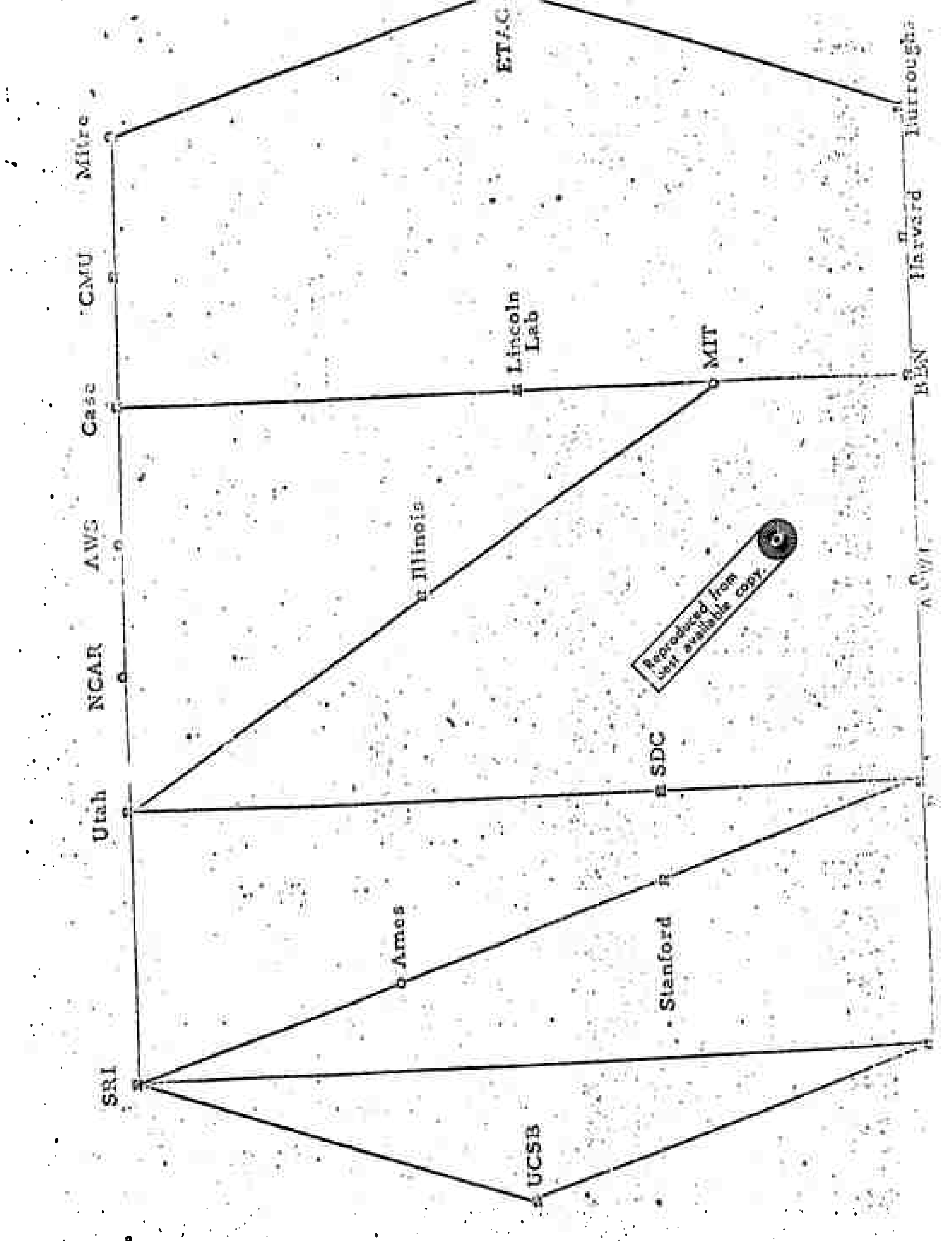
Figure 15

Note 2

We can evaluate the optimum configurations corresponding to several assignments of total cost D, keeping the requirement r as a parameter. In Fig. 18, several curves T = T(D) are plotted for different values of r. Only the marked points on these curves correspond to suboptimum solutions, as the ensemble of capacities available is discrete. The set of capacities and corresponding costs considered in the optimization can be found in Table 1.

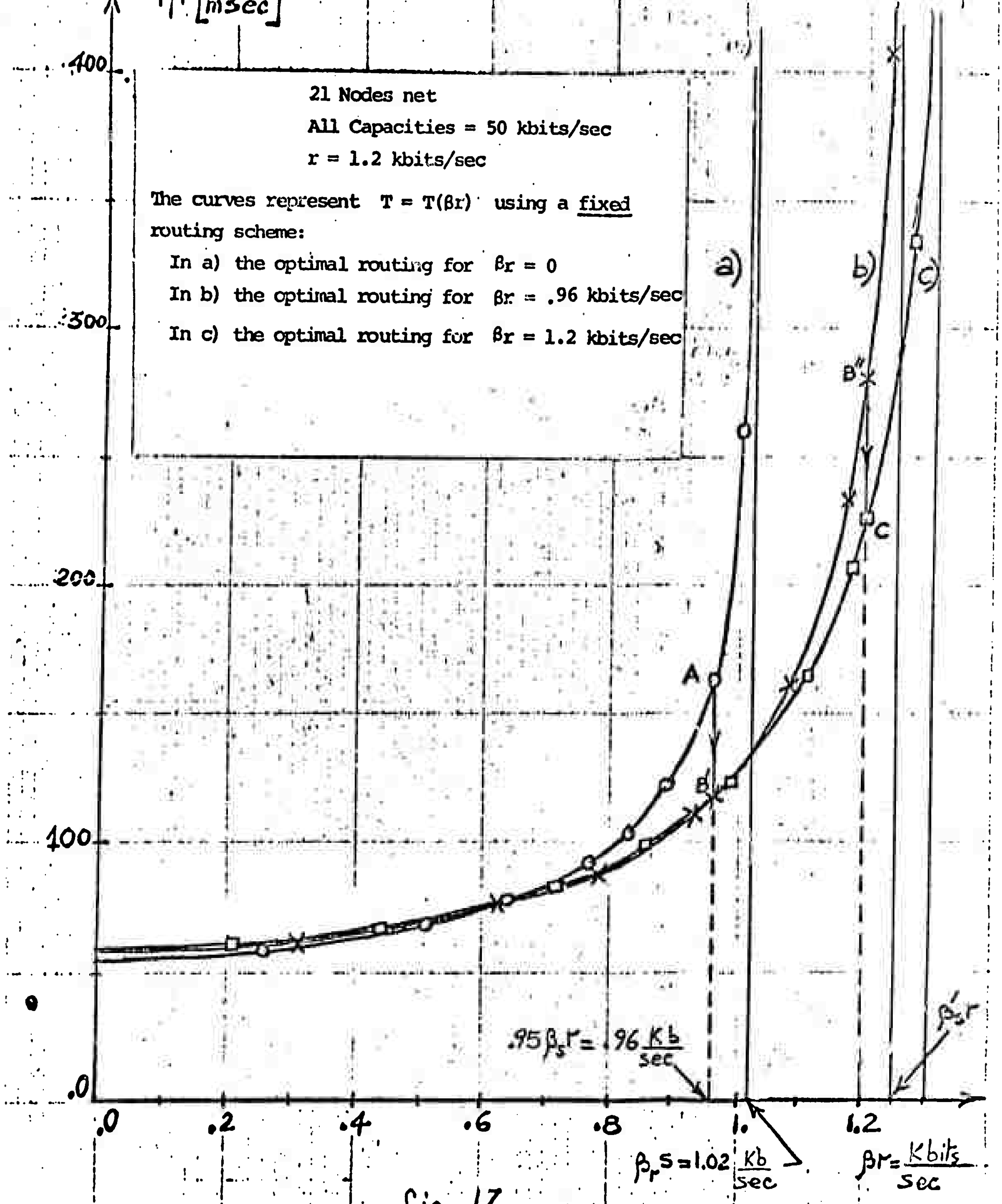
12. Conclusion

From the theoretical point of view, the FD method is valid for any unconstrained multicommodity problem. In the case of constrained problems, one must consider the trade-off between introducing complicated penalty functions and being able to use an algorithm generally easier than the constrained optimization algorithms, i.e. linear or non-linear programming.



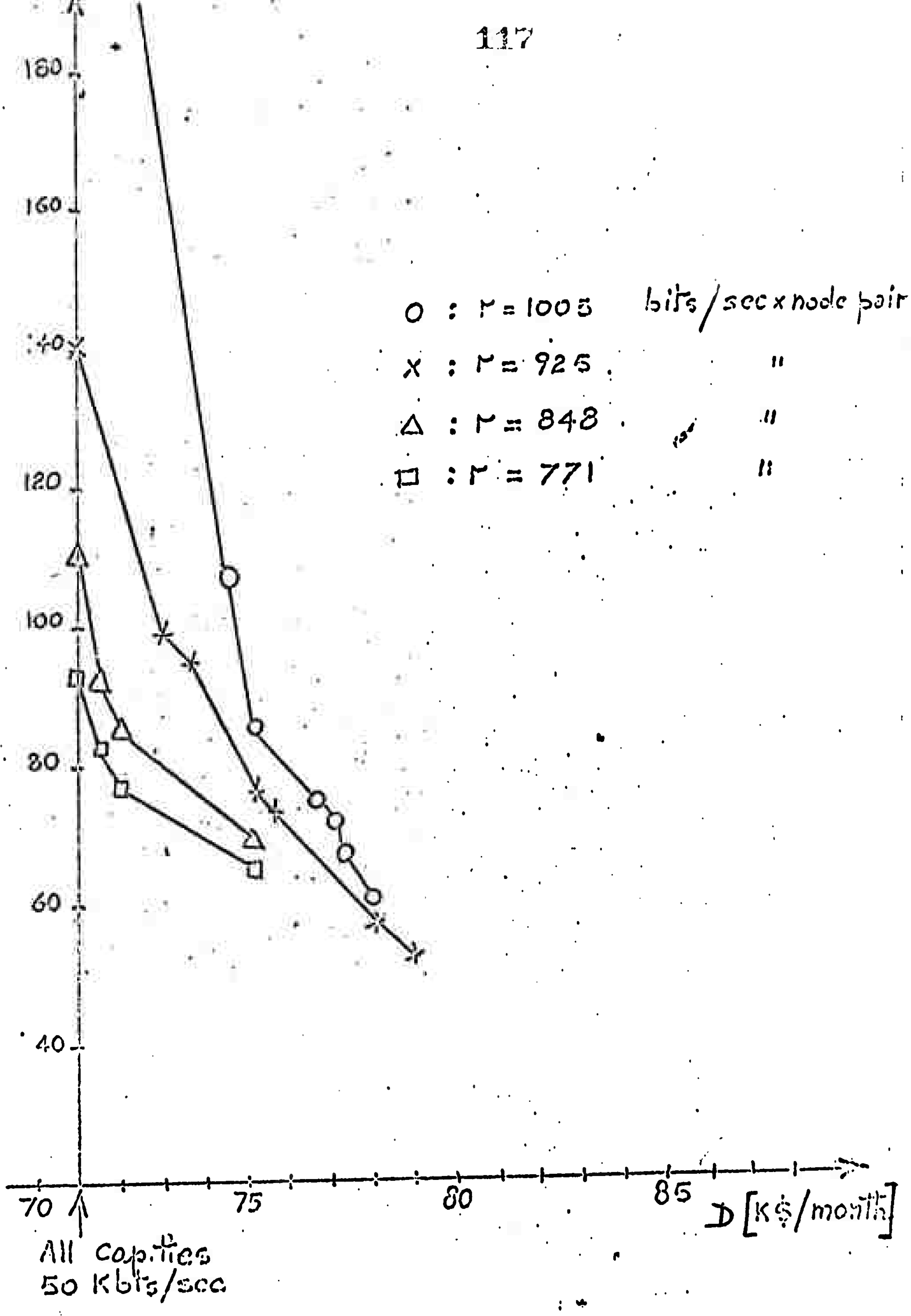
Mitre
CMU
Case
AWS
NCAR
Utah
SRI
UCSB
ETAC
Lincoln Lab
MIT
Illinois
Ames
Stanford
SDC
Burroughs
Harvard
BBN
Reproduced from best available copy.

fig. 16



T [msec]
400
300
200
100
0
.0
.2
.4
.6
.8
1.
1.2
21 Nodes net
All Capacities = 50 kbits/sec
r = 1.2 kbits/sec
The curves represent T = T(βr) using a fixed routing scheme:
In a) the optimal routing for βr = 0
In b) the optimal routing for βr = .96 kbits/sec
In c) the optimal routing for βr = 1.2 kbits/sec
a)
b)
c)
A
B'
B''
C
.95 β_s r = .96 Kb/sec
β_r s = 1.02 Kb/sec
β'_s r
βr = Kbits/sec

fig. 17



180
160
140
120
100
80
60
40
70
75
80
85
D [K$/month]
O : r = 1005 bits/sec x node pair
X : r = 925 "
△ : r = 848 "
□ : r = 771 "
All copies 50 Kbits/sec

fig. 18

TABLE 1

CHANNEL CAPACITIES AND CORRESPONDING

COSTS USED IN THE OPTIMIZATION

Capacity [kbits/sec]	Termination Cost [$/month]	Line Cost [$/month/mile]
7.2	810	.35
19.2	850	2.10
50	850	4.20
108	2400	4.20
230.4	1300	21.00

Note: The total cost per month of a channel is given by:

total cost = termination cost + (line cost) X (length in miles)

If the problem automatically incorporates the penalty functions in the objective function (as S/F network problems usually do), and if the "large and balanced net" assumption is valid, then the FD method seems to be more convenient than other methods.

REFERENCES

1. Kleinrock, L., Communication Nets; Stochastic Message Flow and Delay, McGraw-Hill, 1964.

2. Steiglitz, K., P. Weiner, and D. J. Kleitman, "The Design of Min. Cost Survivable Networks", Trans. on Circuit Theory, November 1969, pp. 455-460.

3. Roberts, L., "Multiple Computer Networks and Intercomputer Communications," ACM Symposium on Operating Systems Principles, Gatlinburg, Tennessee, October 1967,

4. Roberts, L. and B. Wessler, "Computer Network Development to Achieve Resource Sharing," AFIPS Conference Proceedings, SJCC, May 1970, pp. 543-549.

5. Heart, R., 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," SJCC, May 1970, pp. 551-567.

6. Kleinrock, L., "Analytic and Simulation Methods in Computer Network Design," AFIPS Conference Proceedings, SJCC, May 1970, pp. 569-579.

7. Frank, H., I. T. Frisch, and W. Chou, "Topological Considerations in the Design of the ARPA Computer Network," AFIPS Conference Proceedings, SJCC, May 1970, pp. 581-587.

8. Carr, S., S. Crocker, and V. Cerf, "HOST-to-HOST Communication Protocol in the ARPA Network," AFIPS Conference Proceedings, SJCC, May 1970, pp. 589-597.

9. Hu, T. C., Integer Programming and Network Flows, Addison-Wesley, 1969.

10. Meister, B., H. R. Mueller, and H. R. Rudin, Jr., "Optimization of a New Model for Message-Switching Networks," ICC '71, pp. 39-16 to 39-21.

APPENDIX D

AN ANALYSIS OF THE SEPARATION BETWEEN PACKETS IN A STORE-AND-FORWARD NETWORK

by G. D. Cole and L. Kleinrock

AN ANALYSIS OF THE SEPARATION BETWEEN PACKETS IN A STORE-AND-FORWARD NETWORK*

by

G.D. COLE
and
L. KLEINROCK

Computer Science Department
University of California
Los Angeles, California

ABSTRACT

The ARPA computer network utilizes message-switching to provide communications between the various HOST computer sites. This intercommunication is in terms of messages, which are segmented into smaller size packets for store-and-forward transmission between the message-switching computers which are referred to as Interface Message Processors (IMP's). These segmented (multi-packet) messages must be reassembled at their destination IMP prior to delivery. Each IMP receives packets on two or more input lines, causing the occasional buildup of queues on a given output line. The result of such queues is that the packets of a multi-packet message can become separated, i.e., "foreign" packets can become interspersed with those of a given message. This separation causes a longer reassembly time at the destination IMP, and is therefore of interest from the viewpoints of message delay and buffer utilization.

The analysis of the packet separation assumes Poisson arrivals of interfering traffic, and considers both first-come, first-served and priority queue disciplines at the output transmission facility. Equations are derived for the expected inter-packet separation in each case, and both expected values are shown to have the same finite limiting value as the packets pass through an arbitrarily large number of nodes. Empirical results are also given based on measurement data taken on the ARPA net, and the relationship between the modeling and measurement efforts is discussed.

* This research was supported by the Advanced Research Projects Agency of the Department of Defense under Contract Number DAHC-15-69-C-0285.

1. INTRODUCTION

The ARPA network of computers involves an extensive store-and-forward message switching network to provide data communications between the various HOST sites. This net utilizes Interface Message Processors (IMP's) at each store-and-forward node and 50K bit/second transmission lines to provide both a responsive data path for interactive work and a reasonable throughput for file transfers.[1-5]

The HOST sites consist of a variety of computer systems, with these hardware and software resources being shareable via the network. Several of the sites have developed particular specialties such as graphics, artifical intelligence, man-machine interaction, etc. UCLA is one such node, having specialized in the areas of network performance analysis, simulation, and measurement, and serves as the Network Measurement Center (NMC). An extensive set of measurement tools has been developed, both within the IMP's and at the NMC HOST computer.[6] These facilities provide an experimenter with "probes" to monitor the network traffic, or if desired, to create artificial traffic to simulate a given message traffic condition. These measurement tools have been found to be very effective in gaining insight into the network behavior, and in verifying (or correcting) the accuracy of analytic models of network performance. This paper considers one such model, namely the analysis of the separation between message segments when other traffic is in contention for the communication facilities, and discusses the usage of the measurement tools in this development.

A message can be at most 8064 bits in length which, at a 50K bit/second rate, requires about 160 milliseconds for serial transmission of a full message over one of the transmission lines. For a variety of reasons*, messages are segmented into shorter units (called packets) of up to 1008 bits each. This segmentation is performed by the source IMP, with reassembly of the packets into a complete message being handled by the destination IMP; Thus the HOST-to-HOST transmissions are

* Message segmentation into shorter blocks is desirable for several reasons including efficiency of buffer allocation and error control, as well as reduction of retransmission delays and delays to short interactive messages.

in terms of messages, and IMP-to-IMP transmissions involve packets. When an IMP receives and accepts a packet from a neighboring IMP, it sends an acknowledgement (ACK) in reply to this neighbor indicating that the packet was acceptable; this allows the previous IMP to free the storage buffer being utilized for that packet.

Each IMP has two or more input channels upon which packets can arrive. (An arrival is said to occur when the last bit of the packet has been received.) Therefore, it is possible for incoming packets to arrive more rapidly than they can be forwarded, and queues can build up with packets from two or more messages becoming interspersed as shown in Figure 1. The effect of this interspersal is to cause the packets of any given message to become separated, and therefore introduces an additional delay in the reassembly of the message.

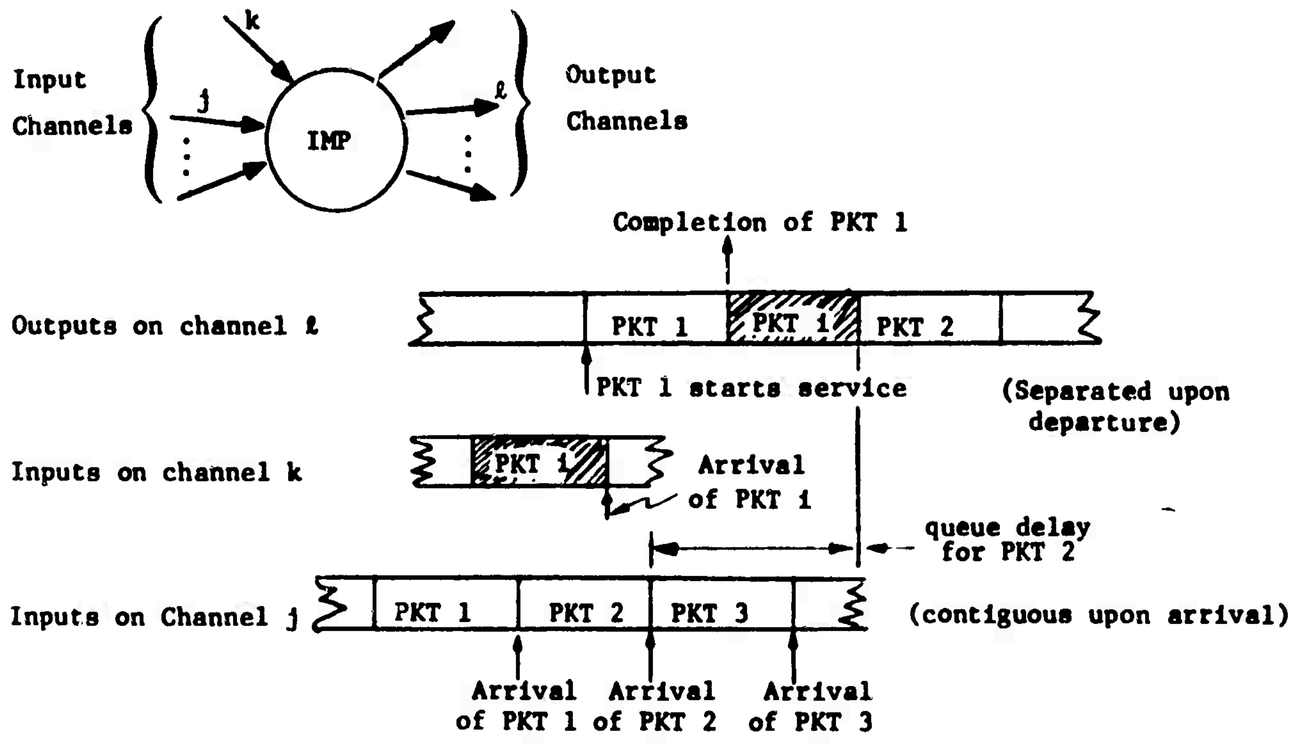


Figure 1, The Interspersing of Message Packets at a Store-and-Forward Node

The object of this study is to investigate this packet separation effect, and to determine the relationships between the pertinent network parameters and this separation delay. The initial analysis considers a first come, first served (FCFS) flow of packets, with the results being extended to include the introduction of priority traffic, and the finite arrival rate at the source node. Network measurement and artificial traffic generation facilities were utilized to test the results of the model against actual network data.

2. THE PACKET SEPARATION ANALYSIS

There is an intuitive feeling that the separation between the packets of a message should grow as the message passes through more and more store-and-forward nodes. This intuition is based on the fact that the probability of interference traffic arrivals increases as the separation increases, and the additional arrivals further increase the separation. However, we will show in Theorems 1 and 2 that the expected separation between packets is bounded for both the FCFS and priority cases, and interestingly, that this bound is the same for both cases, namely:

$$\lim_{n\to\infty} E[\text{interpacket gap after n nodes}] = \frac{\rho_i}{1-\gamma} X_s \tag{1}$$

where ρ_i is the traffic intensity of the interfering traffic, γ is as defined below, and X_s is the time required to service (i.e., transmit) a maximal length packet.* The expected separation at the output can be derived as shown in the following theorem.

Theorem 1

Utilizing a FCFS queue discipline, the expected interpacket gap for a segmented message is:

$$E[\tau_n] = [1 - (\gamma)^{n-1}] \frac{\rho_i}{1-\gamma} \cdot X_s \tag{2}$$

* Implicit in the definition of X_s is the assumption that all but the last packet of a multi-packet message are of this maximal length.

The object of this study is to investigate this packet separation effect, and to determine the relationships between the pertinent network parameters and this separation delay. The initial analysis considers a first come, first served (FCFS) flow of packets, with the results being extended to include the introduction of priority traffic, and the finite arrival rate at the source node. Network measurement and artificial traffic generation facilities were utilized to test the results of the model against actual network data.

2. THE PACKET SEPARATION ANALYSIS

There is an intuitive feeling that the separation between the packets of a message should grow as the message passes through more and more store-and-forward nodes. This intuition is based on the fact that the probability of interference traffic arrivals increases as the separation increases, and the additional arrivals further increase the separation. However, we will show in Theorems 1 and 2 that the expected separation between packets is bounded for both the FCFS and priority cases, and interestingly, that this bound is the same for both cases, namely:

$$\lim_{n\to\infty} E[\text{interpacket gap after n nodes}] = \frac{\rho_i}{1-\gamma} X_s \tag{1}$$

where ρ_i is the traffic intensity of the interfering traffic, γ is as defined below, and X_s is the time required to service (i.e., transmit) a maximal length packet.* The expected separation at the output can be derived as shown in the following theorem.

<u>Theorem 1</u>

Utilizing a FCFS queue discipline, the expected interpacket gap for a segmented message is:

$$E[\tau_n] = [1 - (\gamma)^{n-1}] \frac{\rho_i}{1 - \gamma} \cdot X_s \tag{2}$$

* Implicit in the definition of X_s is the assumption that all but the last packet of a multi-packet message are of this maximal length.

where τ_n is the interpacket spacing due to interfering traffic at the output of the nth node along a store-and-forward path; X_s, is the service time of a maximal length packet; $\rho_i = \lambda_i \bar{x}_i$ is the traffic intensity in terms of λ_i, the average arrival rate of interfering packets, and their average service requirement, $\bar{x}_i$, and $\gamma = \rho_i + \alpha$ where α is the probability that an interference packet (on input) takes the same output channel as the observed traffic. All ρ_i's are assumed to be equal along the entire network path. Interfering packet arrivals at each node are assumed to be Poisson.

Proof:
If we consider a "tagged" multipacket message at the originating node, consisting of packets $P_1, P_2, \ldots, P_j, \ldots$, we find a situation as shown in Figure 2. The packets are presented to the queue as a group, and for the FCFS system, are transmitted continguously. The arrivals at the second node are therefore spaced at regular intervals of $T_1 = X_s$ seconds, where X_s is the time required to service one maximal length packet, i.e., approximately 20 msec. for a 1008 bit packet at 50Kb/sec. However, at this second node there may be interfering traffic arriving on other input channels (with an average arrival rate of λ_i) and as shown in Figure 2(b), these packets may become interspersed with the packets of the tagged message. The gap time (τ_2) between the now non-contiguous packets has an expected value given by:

$$E[\tau_2] = \bar{n}_{i2} \bar{x}_i \tag{3}$$

where the subscripts i refer to the interfering traffic and the subscript 2 indicates that this delay is at the output of the second node along the path. $\bar{n}_{i2}$ is the average number of interfering packets which arrive during the inter-arrival time between the tagged packets; for Poisson interference arrivals this is:

$$\bar{n}_{i2} = \lambda_i X_s \tag{4}$$

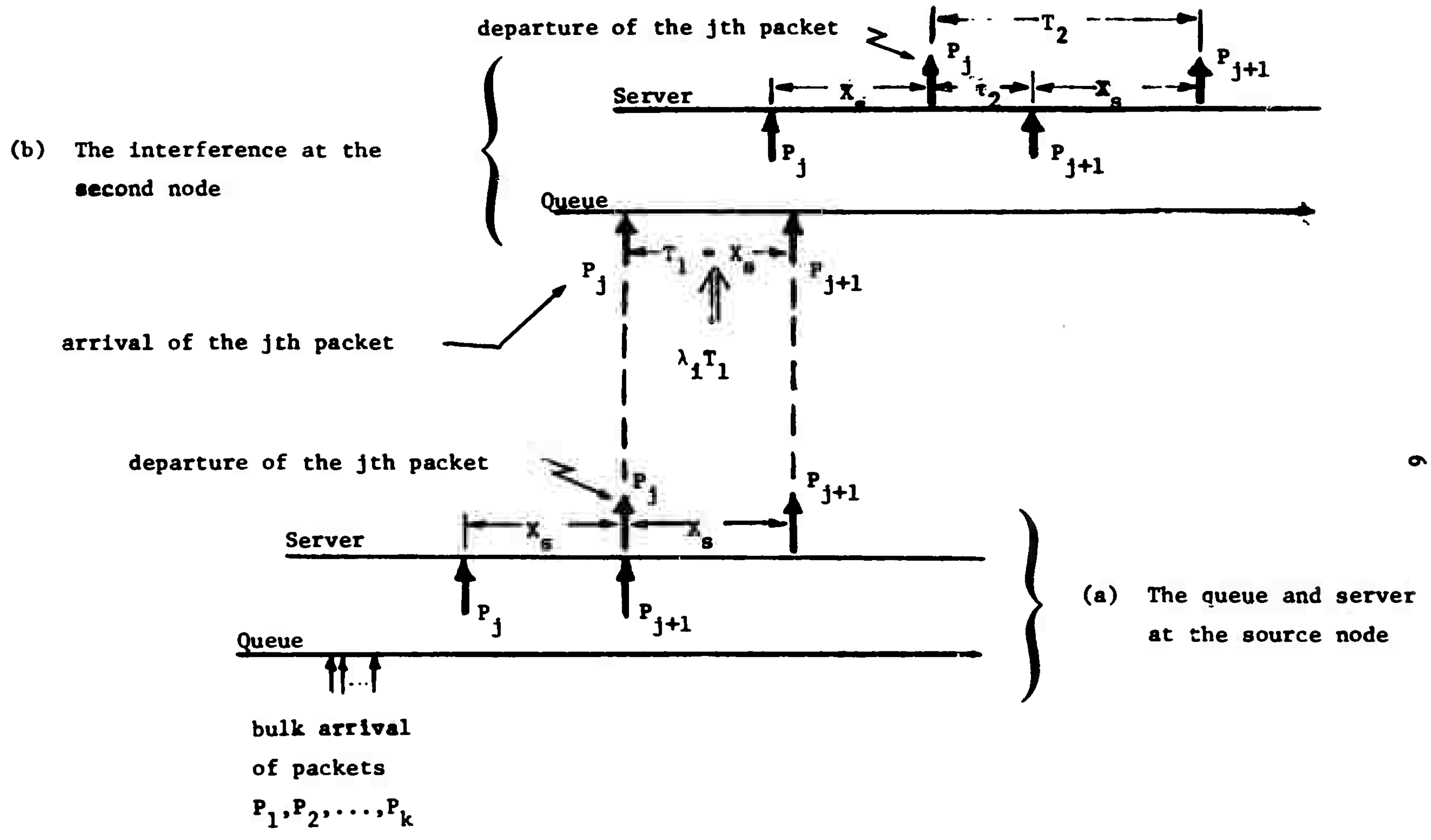

departure of the jth packet
T_2
P_j
P_{j+1}
Server
X_s
τ_2
X_s
P_j
P_{j+1}
(b) The interference at the second node
Queue
P_j
$T_1 - X_s$
P_{j+1}
arrival of the jth packet
$\lambda_1 T_1$
departure of the jth packet
P_j
P_{j+1}
Server
X_s
X_s
P_j
P_{j+1}
(a) The queue and server at the source node
Queue
bulk arrival of packets $P_1, P_2, \ldots, P_k$

Figure 2, The Effect of Interfering Traffic to Create a Separation Between Arrivals of Packets of a Message

Equation (3) then becomes,

$$E[\tau_2] = (\lambda_1 X_s)\, \bar{x}_1,$$

Defining:

$$\rho_1 = \lambda_1 \bar{x}_1. \tag{5}$$

we have,

$$E[\tau_2] = \rho_1 X_s \tag{6}$$

This separation, τ_2, then contributes to the output interval at the second node such that:

$$E[T_2] = X_s + E[\tau_2], \tag{7}$$

which becomes the average interarrival time at the third store-and-forward node. There is a non-zero probability that the interfering packets which separate the tagged packets will be routed to the same output channel at this node,

$$\alpha = P[\text{interfering packet stays as interference on output}] \tag{8}$$

The resulting gap between the tagged packets at the output of the third node then becomes:

$$E[\tau_3] = \lambda_1(X_s + E[\tau_2])\bar{x}_1 + \alpha\, E[\tau_2]$$

or substituting for $E[\tau_2]$, and simplifying:

$$E[\tau_3] = \rho_1 X_s (1 + \gamma) \tag{9}$$

where:

$$\gamma = \rho_1 + \alpha$$

A similar line of reasoning at the fourth node produces,

$$E[\tau_4] = \rho_i(X_s + E[\tau_3]) + \alpha\, E[\tau_3]$$

or:

$$E[\tau_4] = \rho_i\, X_s\, [1 + \gamma + \gamma^2]$$

and leads to the general iterative expression:

$$E[\tau_n] = \rho_i(X_s + E[\tau_{n-1}]) + \alpha\, E[\tau_{n-1}] \tag{10}$$

or:

$$E[\tau_n] = \rho_i\, X_s \sum_{j=0}^{n-2} (\gamma)^j \tag{11}$$

Equation (11) can be expressed in closed form as:

$$E[\tau_n] = [1 - (\gamma)^{n-1}]\, \frac{\rho_i}{1 - \gamma}\, X_s \tag{12}$$

with τ_n being the gap between a pair of tagged packets at the output of the nth store-and-forward node. This completes the proof of Theorem 1.

The expected interpacket gap is shown in Figure 3 for several values of ρ_i, and indicates several interesting aspects about the gap size. For relatively small values of ρ_i, the expected gap length reaches an asymptotic value after going through a few nodes, with larger values of ρ_i requiring long paths (in excess of ten) to reach a similar asymptote. The fact that in each case an asymptote exists is of interest, since it shows that the gap does not grow in an unbounded fashion. Of particular interest is the mathematical form of this asymptotic value,*

$$\lim_{n\to\infty} E[\tau_n] \equiv \bar{\tau}_\infty = \frac{\rho_i}{1 - \gamma}\, X_s \tag{13}$$

* This form is believed to be analogous to conservation results for various time-sharing queue disciplines (Ref. 7).

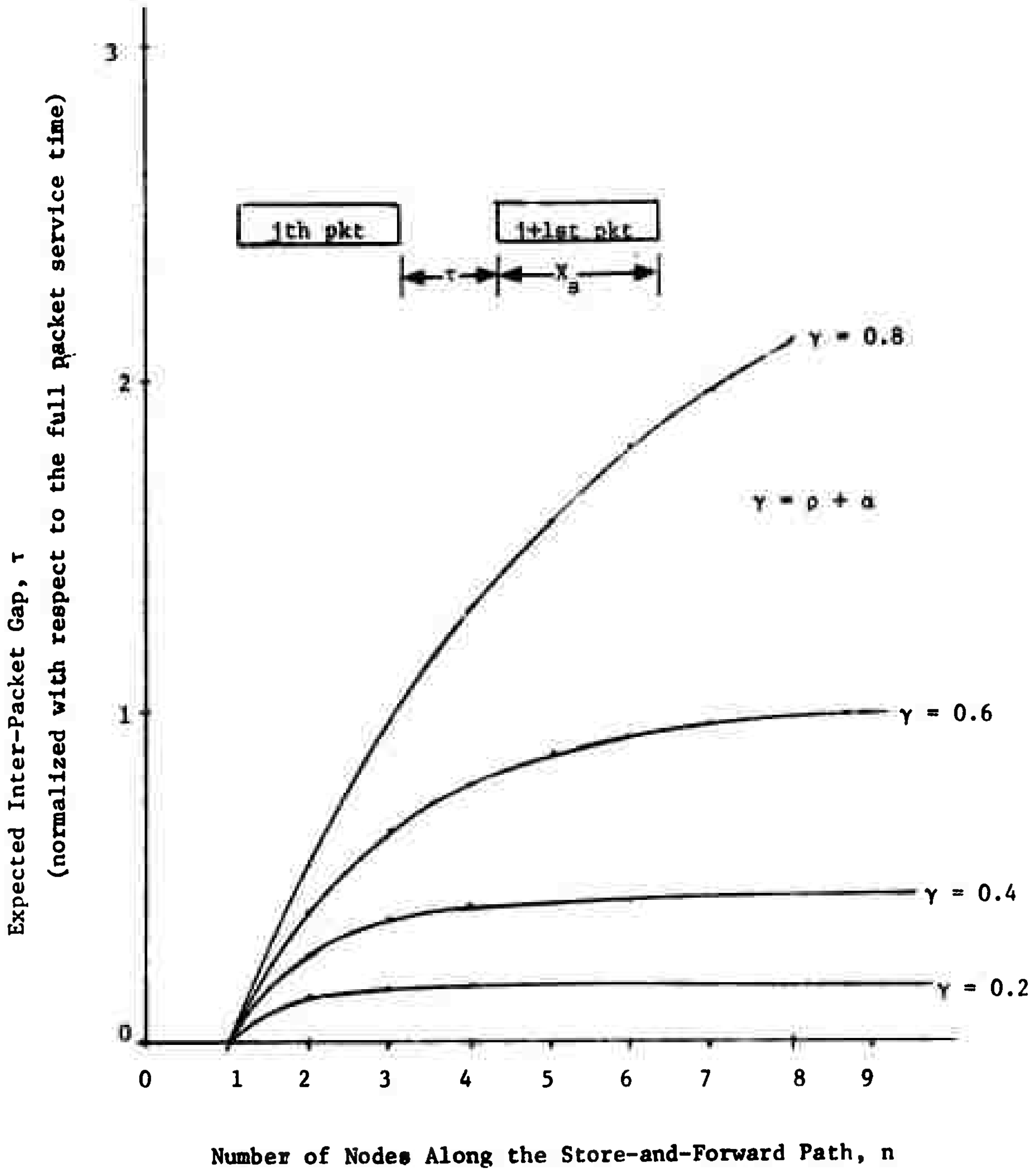


FIGURE 3, THE EXPECTED INTER-PACKET SEPARATION GOING THROUGH n NODES

It is also interesting to note that for the expected interpacket gap to become within a factor, F, of the asymptotic value, $\bar{\tau}_\infty$, we need:

$$n \geq 1 + \frac{\log(1 - F)}{\log \gamma} \tag{14}$$

This can be shown by noting that the expression for $E[\tau_n]$ can be written as:

$$E[\tau_n] = [1 - (\gamma)^{n-1}]\ \bar{\tau}_\infty = F\ \bar{\tau}_\infty$$

such that:

$$(\gamma)^{n-1} = 1 - F$$

and by taking logarithms of both sides,

$$n = 1 + \frac{\log(1 - F)}{\log \gamma}$$

Therefore, any value of n equal to or greater than this value will cause the expected gap to be within the factor, F, of the asymptotic delay value.

A variation of interpacket gap model was obtained by considering the case of non-uniform traffic intensities along the store-and-forward path.* This change meant that the expected interpacket delay would not necessarily increase at each subsequent node, resulting in a variation of the iterative relationship of Equation (10) to include this case,

$$E[\tau_n] = \max\{\rho_{in}(X_s + E[\tau_{n-1}]\ , E[\tau_{n-1}]\} \tag{15}$$

where ρ_{in} is the traffic intensity of the interfering packets at the nth node. (This previous analysis was for the $\alpha=0$ case.)

* This extension was made by Gary Fultz, working under the supervision of one of the authors (LK), who was concerned with differing traffic intensities across the net in routing simulation studies. This interpacket gap time was of concern to him because of its effect on the reassembly time of messages at the eventual destination. (See Ref. 8.)

3. THE INTRODUCTION OF PRIORITIES

The interpacket gap analysis in a priority system differs in two aspects from that of the FCFS system. In the FCFS system, any arrivals occurring between the arrivals of packets P_j and P_{j+1} were also serviced between P_j and P_{j+1}. In the priority system, the interfering arrivals will be any non-priority packets between the arrivals of P_j and P_{j+1}, and any priority arrivals after P_j begins service but before P_{j+1} begins. The resulting interpacket gap given in Theorem 2 is shown to increase more rapidly than that of the non-priority case, but to have the same limiting value.

<u>Theorem 2</u>

The expected interpacket gap for a segmented message with higher priority traffic competing for the server is:

$$E[\tau_n] = \frac{X_s}{1-\gamma}\left[\rho - K\left(\frac{\rho_{np+\alpha}}{1-\rho_p}\right)^n\right] \quad n = 2,3,\ldots \tag{16}$$

where the conditions and variables are as defined in Theorem 1 with ρ_p and ρ_{np} being the traffic intensities of priority and non-priority traffic respectively, and $\rho = \rho_p + \rho_{np}$. The value K is,

$$K = \frac{\rho_{np} + \alpha\rho_p}{\rho_{np} + \alpha}$$

<u>Proof:</u>

Assume that a multipacket message is initially entered into the network as shown in Figure 4(a). The jth and j+1st packets of the message are not necessarily served contiguously (as in the FCFS case) since priority arrivals during the service of P_j will be serviced before P_{j+1} as will other priority messages which arrive during the service of the priority messages. The expected time between the departures of P_j and P_{j+1} is:

$$E[T_1] = X_s + E[\tau_1] \tag{17}$$

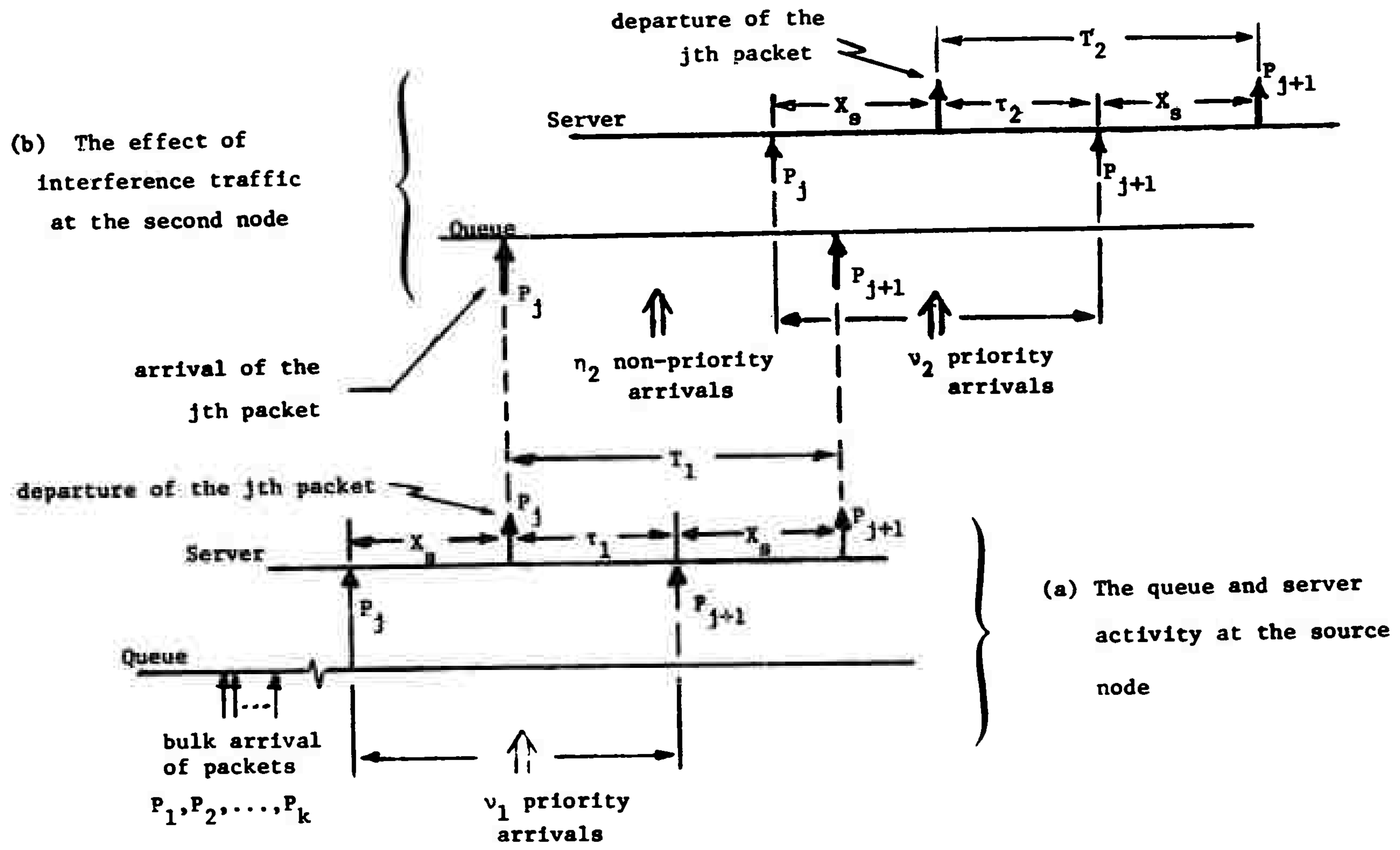

(b) The effect of interference traffic at the second node
departure of the jth packet
T_2
P_{j+1}
Server
X_s
τ_2
X_s
P_j
P_{j+1}
Queue
P_j
P_{j+1}
arrival of the jth packet
n_2 non-priority arrivals
ν_2 priority arrivals
T_1
departure of the jth packet
P_j
P_{j+1}
Server
X_s
τ_1
X_s
P_j
P_{j+1}
Queue
bulk arrival of packets $P_1, P_2, \ldots, P_k$
ν_1 priority arrivals
(a) The queue and server activity at the source node

Figure 4, Arrivals and Departures for the Queue and Server in the Priority Queueing Case where τ_1 is the time to service the ν_1 priority arrivals, with:

$$E[\nu_1] = \lambda_{ip}(X_s + E[\tau_1]), \tag{18}$$

λ_{ip} being the average arrival rate of interfering priority messages. Since the arrivals and service requirements are independent,

$$E[\tau_1] = E[\nu_1] \,.\, E[x \mid \text{mes. is priority}]$$

or:

$$E[\tau_1] = \lambda_{ip}(X_s + E[\tau_1])\bar{x}_p \tag{19}$$

Solving for $E[\tau_1]$ we obtain:

$$E[\tau_1] = \frac{\lambda_{ip}\ \bar{x}_1\ X_s}{1 - \lambda_{ip}\bar{x}_p} = \frac{\rho_p\ X_s}{1 - \rho_p} \tag{20}$$

where ρ_p is the traffic intensity for interfering priority messages.

$$\rho_p = \lambda_{ip}\ \bar{x}_p$$

The expected interdeparture time of packets P_j and P_{j+1} at the first node is then:

$$E[T_1] = X_s + \frac{\rho_p\ X_s}{1 - \rho_p} = X_s + E[\tau_1] \tag{21}$$

so that at the second node we can expect to find n_2 interfering arrivals of regular messages, where:

$$E[n_2] = \lambda_{inp}\ E[T_1] = \lambda_{inp}(X_s + E[\tau_1]) \tag{22}$$

Only the non-priority message arrivals during T_1 are considered to be interfering in the sense of adding to the interpacket gap. Any priority arrivals are either served prior to P_j or are considered in the ν_2 interfering priority arrivals, where ν_2 has an expected value of:

$$E[\nu_2] = \lambda_{ip}(X_s + E[\tau_2]) \tag{23}$$

Therefore, we have:

$$E[T_2] = X_s + E[n_2] \cdot \bar{x}_{np} + E[\nu_2] \cdot \bar{x}_p + \alpha E[\tau_1] \tag{24}$$

where the $\alpha E[\tau_1]$ term is due to the fraction, α, of the incoming interference that is routed along with the tagged message (as defined in equation 9).* Substituting for $E[n_2]$ and $E[\nu_2]$ from equations (22) and (23) respectively, and noting that:

$$E[\tau_2] = E[T_2] - X_s$$

we have:

$$E[\tau_2] = \rho_{np} (X_s + E[\tau_1]) + \rho_p (X_s + E[\tau_2]) + \alpha E[\tau_1] \tag{25}$$

Collecting terms and simplifying results in:

$$E[\tau_2] = \frac{1}{1 - \rho_p} \{\rho X_s + (\rho_{np} + \alpha) E[\tau_1]\} \tag{26}$$

which generalizes into the iterative result:

$$E[\tau_n] = \frac{1}{1 - \rho_p} \{\rho X_s + (\rho_{np} + \alpha) E[\tau_{n-1}]\} \quad n = 2, 3, \ldots \tag{27}$$

* The effect of the concurrently routed interference traffic is more complicated in the priority case, but on the average, will produce an interference component of $\alpha E[\tau_{in}]$.

A closed form representation of this equation is difficult to find by inspection of the forms of $E[\tau_2]$, $E[\tau_3]$, etc., but is expected from previous work to be of the form:

$$E[\tau_n] = \frac{X_s}{1-\gamma}\left\{\rho - K\left(\frac{\rho_{np}+\alpha}{1-\rho_p}\right)^n\right\} \tag{28}$$

This can be shown to be correct by an inductive proof, i.e., utilizing equations (27) and (28), such that we test for the equivalence of:

$$E[\tau_n] = \frac{1}{1-\rho_p}\left\{\rho X_s + (\rho_{np}+\alpha)\cdot\left(\frac{X_s}{1-\gamma}\right)\cdot\left\{\rho - K\left(\frac{\rho_{np}+\alpha}{1-\rho_p}\right)^{n-1}\right\}\right\}$$

$$\overset{?}{=} \frac{X_s}{1-\gamma}\left\{\rho - K\left(\frac{\rho_{np}+\alpha}{1-\rho_p}\right)^n\right\} \tag{29}$$

Simplification of the left-hand side shows that the equality holds. We can solve for the value, K, by evaluating the expression for a known condition, namely $E[\tau_2]$.

$$E[\tau_2] = \frac{1}{1-\rho_p}\left\{\rho X_s + (\rho_{np}+\alpha)\left(\frac{\rho_p X_s}{1-\rho_p}\right)\right\} = \frac{X_s}{1-\gamma}\left\{\rho - K\left(\frac{\rho_{np}+\alpha}{1-\rho_p}\right)^2\right\} \tag{30}$$

After considerable simplification and combination of terms, the value of K can be shown to be,*

$$K = \frac{\rho_{np}+\alpha\rho_p}{\rho_{np}+\alpha} \tag{31}$$

which completes the proof of Theorem 2.

* These simplifications in volve the usage of the equalities $\rho = \rho_p + \rho_{np}$ and $\gamma = \rho + \alpha$.

The expected gap size for the priority case can then be expressed as:

$$E[\tau_n] = \frac{X_s}{1 - \gamma}\left\{\rho - \left(\frac{\rho_{np} + \alpha\rho_p}{\rho_{np} + \alpha}\right)\left(\frac{\rho_{np} + \alpha}{1 - \rho_p}\right)^n\right\} \qquad n=2,3,\ldots \qquad (32)$$

which for ρ_p equal to zero, becomes the FCFS result of equation (12).

If we compare the FCFS and priority analysis results, we see that they have the same limiting values, i.e.,

$$\lim_{n\to\infty} E[\tau_n | \text{FCFS}] = \lim_{n\to\infty} E[\tau_n | \text{priority case}] \qquad (33)$$

This latter limit can be found by showing that:

$$\lim_{n\to\infty}\left(\frac{\rho_{np} + \alpha}{1 - \rho_p}\right)^n = 0 \qquad (34)$$

or equivalently, that

$$1 - \rho_p > \rho_{np} + \alpha$$

We know that,

$$1 > \gamma = \rho_p + \rho_{np} + \alpha$$

and therefore by subtracting ρ_p from each side of the equation,

$$1 - \rho_p > \rho_{np} + \alpha$$

which produces the limit as in equation (34). This limiting effect is also indicated in Figure 5, which shows both the priority and FCFS cases.

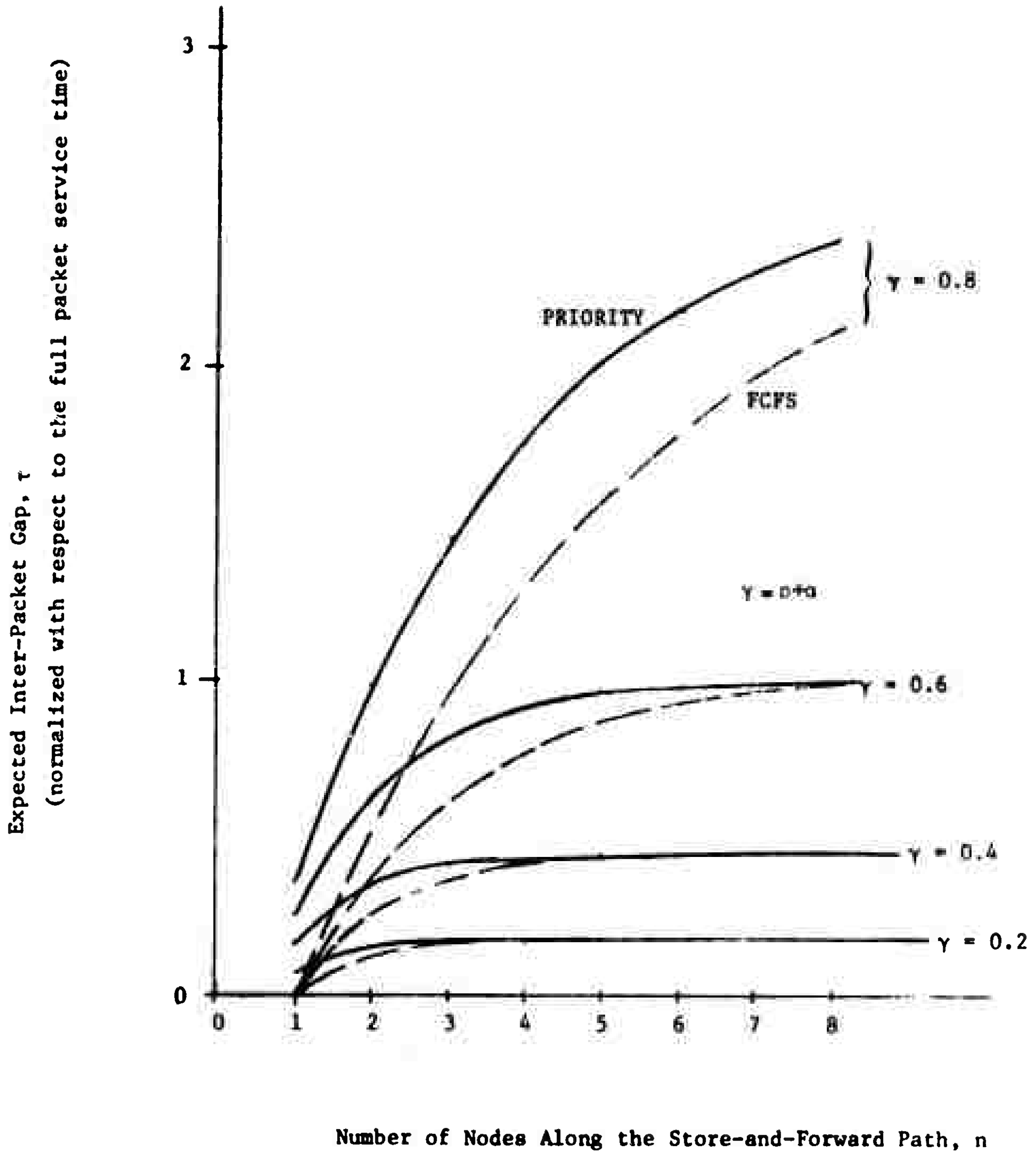


FIGURE 5, THE EXPECTED INTERPACKET GAP FOR THE PRIORITY SYSTEM AFTER GOING THROUGH n NODES

4. THE ANALYSIS FOR NON-BULK ARRIVALS

The preceding analysis was based on the assumption that all of the packets of a message arrive at the source IMP simultaneously, i.e., as a bulk arrival. If we assign a finite rate to these HOST-to-IMP transfers, such that a full packet requires X_1 seconds to cross this boundary, then the analysis must be modified to include this effect. The interpacket gap at the output of the originating node then has an additional term:

$$E[\tau_1] = \lambda_{1p}\,\bar{x}_p(X_s + E[\tau_1]) + \lambda_{1np}\,\bar{x}_{np}X_1 \tag{35}$$

which is a modification to Equation (19) to include the non-priority interference. Solving for the average gap time, we obtain:

$$E[\tau_1] = \frac{\rho_p X_s + \rho_{np} X_1}{1 - \rho_p} \tag{36}$$

The delays at subsequent nodes can be determined by the previous iterative equation (27), with the result being an additional term beyond that of equation (32),

$$E[\tau_n] = \frac{X_s}{1-\gamma}\left\{\rho - K\left(\frac{\rho_{np} + \alpha}{1 - \rho_p}\right)^n\right\} + \frac{\rho_{np}(\rho_{np} + \alpha)^{n-1}}{(1-\rho_p)^n}\;X_1 \qquad n = 2,3,\ \ldots \tag{37}$$

where K is as defined in equation (31), and the additive term is the effect of the time required for a HOST-to-IMP packet transmission, X_1. However, it is interesting to note that this additive effect does not change the asymptotic values as n becomes very large, since the X_1 term of equation (37) approaches zero as $n\to\infty$.

5. EXPERIMENTAL VERIFICATION OF THE PACKET SEPARATION MODEL

A complete experimental verification of the packet separation model would require the ability to generate artificial traffic at each node along the store-and-forward path of a test message. Since such facilities are currently available at only one site, the UCLA Network Measurement Center, the verification test was much less extensive than would normally be desired, and was conducted only for the case of n = 1. Equation (37) is valid only for $n \geq 2$, so the test was based on equation (36),

$$E[\tau_1] = \frac{\rho_p X_s + \rho_{np} X_1}{1 - \rho_p}$$

The interference traffic was generated by the pseudo-random artificial traffic generator (approximately an exponential distribution of message lengths and Poisson arrivals) with the traffic intensity, ρ, being determined by selection of the average message length. The number of active message generators was a compromise between, (1) the desire to avoid alternate routing, and (2) the desire to minimize the RFNM dependency of the arrivals.* Eight generators were found to meet both desires satisfactorily.

The generation of the multi-packet test traffic was planned utilizing two different techniques to obtain a range of values of the parameter, X_1, (the time required for the serial "arrival" of a packet). The "fake HOST" pseudo-message generator facility within the IMP was selected to provide a minimal value of X_1 of 3.3 msec., which is the typical time required for the background program to generate a full packet of data. Similarly, the "fake HOST" generated traffic at a neighboring IMP was utilized for the maximal value of X_1 = 23.0 msec. which is the time required for the serial arrival of a full packet at the UCLA IMP.

The theoretical values** of the inter-packet gap time, τ, for various message lengths and values of X_1 are plotted in Figure 6 along with the test data from the experiment. The increase in the value of τ for very short messages is due to the

* The RFNM (Request For Next Message) is a flow control mechanism which does not allow the next message to be sent on a given logical link, until a request is received in reply to the present message.

** See the appendix for the calculation of these values.

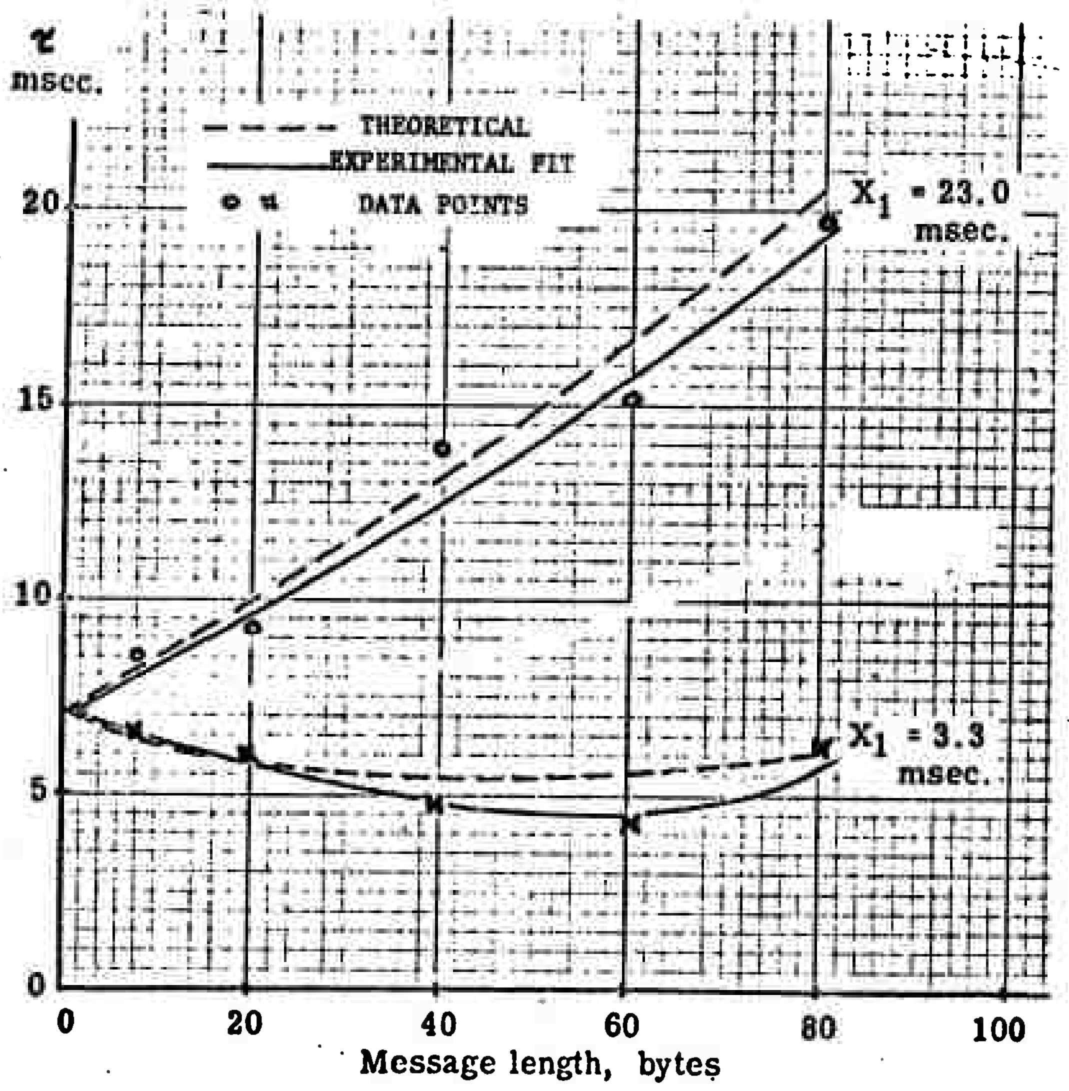


Figure 6, A Comparison of Measured and Theoretical Inter-Packet Gap Times Due to Interference Traffic fact that most of these messages are of the priority class, and therefore cause an increased amount of interference. Equation (37) indicates that for the special case in which all of the interference traffic is of the priority class, the value of τ will not be a function of X_1, and this effect was also verified by the experimental data. This effect is due to the fact that the only arrivals that will contribute to the inter-packet gap are those that occur during the X_s seconds that a packet requires for service, and the subsequent arrivals during their service, etc., as expressed in equation (19).

The experimental and theoretical results agree reasonably well for both cases. Both the shape of the curves and the magnitude of the interpacket gap (as measured by trace data at the destination) indicate that the model represents the actual system system behavior satisfactorily.*

This agreement between the experimental and theoretical results was not obtained in the initial experiments, but came about as a result of several improvements to the original model. These improvements were made as a result of new insight gained from the measurement work, and included the correction of an algebraic error, as well as correcting a more fundamental oversight of having neglected the effect of acknowledgement traffic (as described in the Appendix). The measurements were thereby found to significantly aid in the development and verification of the theoretical model, and conversely, the modeling studies were of great value in determining which system values were of interest in the measurement efforts. The two areas of investigation are quite complementary and can be utilized together to gain insight and confidence in the correspondence between the behavior of models and the actual system being studied.

* The ability to trace messages is one of the set of measurement facilities at the NMC. The trace data includes the time of arrival of each packet, from which the value of τ can be computed (see reference 6).

APPENDIX -- THE THEORETICAL TRAFFIC INTENSITY INCLUDING THE EFFECTS OF ACKNOWLEDGEMENTS AND PRIORITY TRAFFIC

The theoretical curves of Figure 6 include the effects of the acknowledgements, the priority messages, and the non-priority messages by considering the expected service time to be:

$$\begin{aligned} E[x] = {} & E[x \mid \text{an ACK}] \,.\, \Pr[\text{an ACK}] \\ & + E[x \mid \text{a pri. mes.}] \,.\, \Pr[\text{a pri. mes.}] \\ & + E[x \mid \text{non-pri. mes.}] \,.\, \Pr[\text{non-pri. mes.}] \end{aligned}$$

The acknowledgements are in response to the RFNM's (Request For Next Message), and therefore occur with the same frequency as messages, such that:

$$\Pr[\text{an ACK}] = 0.5$$

Messages are considered to be in the priority class if they are less than or equal to 10 bytes (80 bits) in length, which for the exponential density of message length, gives the following expression for β, the fraction of messages that are in the priority class.

$$\beta = 1 - e^{-10/\bar{\ell}}$$

where $\bar{\ell}$ is the average message length in bytes. (The ACK's will also be in the priority class). The expected service time for a priority message can be found from the truncated exponential density such that,

$$\bar{x}_p = E[x_m | x_m \le X_T] = \bar{X}\left[1 - \frac{(X_{T/\bar{X}})\ \bar{e}^{\,X_{T/\bar{X}}}}{1 - \bar{e}^{\,X_{T/\bar{X}}}}\right]$$

and the expected service time for a non-priority message can be shown to be:

$$\bar{x}_{np} = E[x_m \mid x_m > X_T] = X_T + \bar{X}$$

Using these results, one can write the expected service time (of messages and ACK's) as:

$$E[x] = 0.5X_a + 0.5\beta\,[X_0 + \bar{x}_p] + 0.5(1 - \beta)\,[X_0 + X_T + \bar{X}]$$

where the first two terms represent the priority service times, and the last term considers the non-priority service times. (The value X_0 is the fixed overhead service time of a message.) The equation is of the form:

$$E[x] = E[x \mid \text{priority}] + (1 - \beta)E[x \mid \text{non-priority}]$$

and if multiplied by the net arrival rate (of messages and ACK's), produces a traffic intensity, ρ, given by:

$$\rho = \lambda E[x] = \lambda\{0.5X_a + 0.5\beta\,[X_0 + \bar{x}_p]\} + \lambda\{0.5(1 - \beta)[X_0 + X_T + \bar{X}]$$

where:

$$\lambda = \lambda_a + \lambda_m = 2\lambda_m$$

with the subscripts referring to ACK's and messages. The values of the priority and non-priority interference traffic intensities can then be written as:

$$\rho_p = 2\lambda_m\{0.5X_a + 0.5\beta\,[X_0 + \bar{x}_p]\}$$

and:

$$\rho_{np} = 2\lambda_m\{0.5(1 - \beta)[X_0 + X_T + \bar{X}]\}$$

with the composite traffic intensity being:

$$\rho = \rho_p + \rho_{np}$$

If we consider an example in which the messages have an average length of 20 bytes, we find that 40% of the messages are in the priority class, and such messages have an average service time of 3.44 msec.*, compared to an average of 7.5 msec. for the non-priority messages (including the overhead characters for each). The service time of an acknowledgement is 3.0 msec., and when combined with the priority messages, produces a traffic intensity of:

$$\rho_p = \lambda_m[3.0 + 0.4(3.44)] = 0.175$$

where λ_m is 1/(25 msec.) for the artificial traffic being generated. The traffic intensity for the non-priority traffic can be found in a similar manner, since:

$$\rho_{np} \approx \lambda_m\{(1 - \beta)[X_0 + X_T + \bar{X}]\}$$

which becomes:

$$\rho_{np} = \frac{0.6}{25}[2.9 + 1.6 + 3.2 = 0.185$$

such that:

$$\rho = \rho_p + \rho_{np} = 0.36$$

The expected value of τ can then be calculated for the known value of X_s equal to 23.0 msec., and for the desired value of X_1. These results are tabulated in Table 1 along with the associated values of ρ_p and ρ_{np}. (These are the values utilized for the theoretical curves of Figure 6.)

* Based on a 20 μsec./bit transmission delay.

TABLE 1

THEORETICAL VALUES OF THE INTER-PACKET GAP, τ

Avg. Mes. Length (bytes)	ρ	β	ρ_p	ρ_{np}	Expected Value of τ	
					$X_1=3.3$	$X_1=23.0$
0	.23	1.00	.23	.00	6.9	6.9
8	.28	.72	.22	.06	6.7	8.3
20	.36	.39	.18	.18	5.8	10.1
40	.49	.22	.15	.34	5.3	13.3
60	.63	.15	.14	.49	5.6	16.8
80	.77	.12	.14	.63	6.0	20.4

BIBLIOGRAPHY

1. Roberts, L. G. and B. D. Wessler, "Computer Network Development to Achieve Resource Sharing," Proc. of Spring Joint Computer Conference, Vol. 36, AFIPS Press, Montvale, N. J., 1970, pp. 543-549.

2. Heart, F., et al., "The Interface Message Processor for the ARPA Computer Network," ibid., pp. 551-568.

3. Kleinrock. L., "Analytic and Simulation Methods in Computer Network Design," ibid., pp. 569-580.

4. Frank, H., I. Frisch & W. Chou, "Topological Considerations in the Design of the ARPA Computer Network," ibid, pp.581-588.

5. Carr, S., S. Crocker & V. Cerf, "HOST-HOST Communication Protocol in the ARPA Network," ibid., pp. 589-597.

6. Cole, G.D., Computer Network Measurements: Techniques and Experiments, Report #UCLA-ENG-7165, CSMA-1, University of California, Los Angeles, Calif., 1971.

7. Kleinrock, L., Communication Nets: Stochastic Message Flow and Delay, McGraw-Hill, New York, N. Y., 1964.

8. Fultz, G., Adaptive Routing Techniques for Store-and-Forward Message Switching Computer Communications Networks, Ph.D. Dissertation, Dept. of Computer Science, University of Calif., Los Angeles, 1972.

APPENDIX E

PERFORMANCE MEASUREMENTS ON THE ARPA COMPUTER NETWORK

by Gerald D. Cole

ACM/IEEE Second Symposium on Problems in the Optimization of Data Communications Systems, Palo Alto, California, Oct. 20-22, 1971

PERFORMANCE MEASUREMENTS ON THE ARPA COMPUTER NETWORK*

by
Gerald D. Cole
Computer Science Department
University of California
Los Angeles, California

Abstract

An extensive measurement capability has been implemented in the Interface Message Processors (IMP's) of the ARPA network of computers. These measurement facilities were implemented by the network contractor, Bolt Beranek and Newman, Inc., to meet the measurement needs of the UCLA Network Measurement Center, and consist of software routines within the store-and-forward IMP's to accumulate statistics, periodically record snap-shots, trace selected messages through the net, and to be able to generate artificial traffic. These capabilities have been augmented by routines at the Network Measurement Center to provide selective control over the data gathering functions, as well as, providing data reduction and print-out routines, and more extensive artificial traffic generation facilities. The purpose of the measurement effort is to provide insight into network behavior, and to support the analytic and simulation modeling work being done at UCLA. In this latter role, the measurements provide a means of verifying, or correcting, models of network behavior by performing appropriate tests in the actual network environment. The result of several such efforts will be discussed.

1. Introduction

The ARPA computer network interconnects approximately twenty different HOST computer systems at sites across the country for the purpose of sharing resources.(1) Several of these sites have developed expertise in selected areas such as graphics, artificial intelligence, and man-machine communications, and serve as specialization centers within the net to provide particular skills and hardware/software resources related to their particular areas of specialization. The UCLA node in the network serves as one such site, being the Network Measurement Center (NMC), as well as being heavily involved in analytic and simulation modeling of network behavior. This concern for analytic and empirical evaluation of network performance has been an integral part of the network development, and an early responsibility of the NMC was to define the necessary measurement capabilities to be implemented in the Interface Message Processors or IMP's.** Subsequent usage has been made of these measurement facilities to evaluate the network operational status, to indicate areas where redesign may be required, and to provide insight into the cause and effect relationships of the network behavior. This latter activity has proven to be particularly fruitful in conjunction with the network modeling efforts since the validity of the models can thereby be tested against actual observed network behavior.

The IMP-generated statistics routines are utilized in conjunction with programs at the NMC which control the data collection, process the resulting measurement data, and generate desired levels of artificial traffic. These measurement tools will be described in the following section, after which some applications of the measurement facilities will be discussed.

2. The Set of Measurement Tools

The IMP's can gather statistics on the network usage and performance by means of several data gathering routines which can be selectively enabled at appropriate sites. These routines include:[3]

- accumulated statistics (counts and histograms)
- snap-shot statistics (queue lengths and routing tables)
- trace data (event timing for message flow)
- status reports (activity and condition information)

The first three of these routines have been utilized extensively by the network measurement center, while the fourth has been utilized exclusively by the network contractor, Bolt Beranek and Newman.

Accumulated Statistics

The accumulated statistics routine has been utilized more frequently than any of the other measurement tools, since it provides a summary report of the activity at each node where such statistics have been enabled. These data reports include histograms of the lengths of HOST-to-IMP or IMP-to-HOST messages, and of packets which were transmitted on the modem output lines, as well as

* This research was supported by the Advanced Research Projects Agency of the Department of Defense under Contract Number DAHC-15-69-C-0285.

**The IMP's are small computers (modified Honeywell DDP-516s) which handle the store-and-forward communications of the network, and also have the capability of collecting message handling statistics.

counts of ACK's*, RFNM's,** input errors, retransmissions, total words sent, and other data of concern. Each such data message represents the activity over a 12.8 second interval, this period being determined primarily by the concern for counter overflow. The data is sent to the Network Measurement Center as a 390 byte message, which is then inspected and either discarded, printed, or put in a file for further data compacting. When printed, the data is annotated, reformatted, and in some instances additional values are computed, with a resulting print-out as shown in Figure 1. The various fields of this data format are described below.

Part 1 of the data gives message size statistics for both HOST-to-IMP and IMP-to-HOST data transfers. The statistics on single packet messages are accumulated in a logarithmic scale histogram, while multipacket traffic is recorded as a uniform interval histogram. The average number of words in the last packet of a message is also given. In the example shown, there were a total of 200 artifically generated messages received from the HOSTS, of which 208 were single packet messages and twelve were two packets in length.

The second part of Figure 1 shows that these 220 messages were split fairly evenly between six destinations. These sites were selected as destinations for the artificial traffic to show the difference in delays when messages are sent through several store-and-forward nodes.

The round trip times are measured from the receipt of the first packet of the message at the originating IMP, until the receipt of the RFNM by this IMP. The data values are accumulated as a sum of round trip times and a count of the number of round trips, with the average being computed at the NMC.

Part 3 lists the activity for each HOST, both real and fake,*** the latter being referred to as GHOST's. The table helps to determine which HOST's contributed to the composite HOST-to-IMP behavior as recorded in Parts 1 and 2.

The activity on the individual modem channels is recorded in Part 4. HELLO and IHY (I Heard You) messages are sent periodically, and proper counts indicate that the lines are active. (The HELLO message is combined with the routing table update message.) The "#PKTS RECVD" total includes these HELLO messages, and any ACK's and RFNM's that have been received, as well as the number of actual message packets that have been received. Other data include the number of RFNM's sent, the total number of data words sent, the number of times an arrival found the free storage list empty, and in such cases, the number of times that an unacknowledged store-and-forward packet was discarded to free an input buffer.

The final section of Figure 1 shows logarithmic scale histograms of packet lengths on each modem channel. Only the message content of the transmissions are recorded in these histograms, i.e., they do not include the HELLO, IHY, ACK, or RFNM traffic. The choice of the logarithmic scale was due to a priori expectations on the length distribution.

The various portions of the accumulated data can be utilized together to develop a reasonably good picture of the activity over the 12.8 second interval. In the example of Figure 1 we know the total number of messages from the HOST, their final destinations, and their average round trip times. The HOST-IMP histograms give us information of the message size distribution, and in the modem channel histograms give similar data for the packet size distribution as well as channel utilization information. In this example, the channel statistics show that more packets were sent via SRI than was expected from the routing table at the UCLA IMP, which would normally send about half of these packets via RAND. Such data can lead the experimenter to probe further into the cause of such behavior, but to do so, he needs the routing table information which is contained in another set of measurements called snap-shots.

Snap-Shot Statistics

Each snap-shot statistics message contains the queue lengths and the routing table information for that particular IMP and that particular instant of time. These values are recorded and sent at 0.82 second intervals, this time period being a reasonable compromise between the desire to see a time-sequence of state changes, and the conflicting desire to reduce the artifact* caused by sending the statistics too frequently. Like all of the other measurement tools, the snap-shots can be enabled or disabled at each individual IMP.

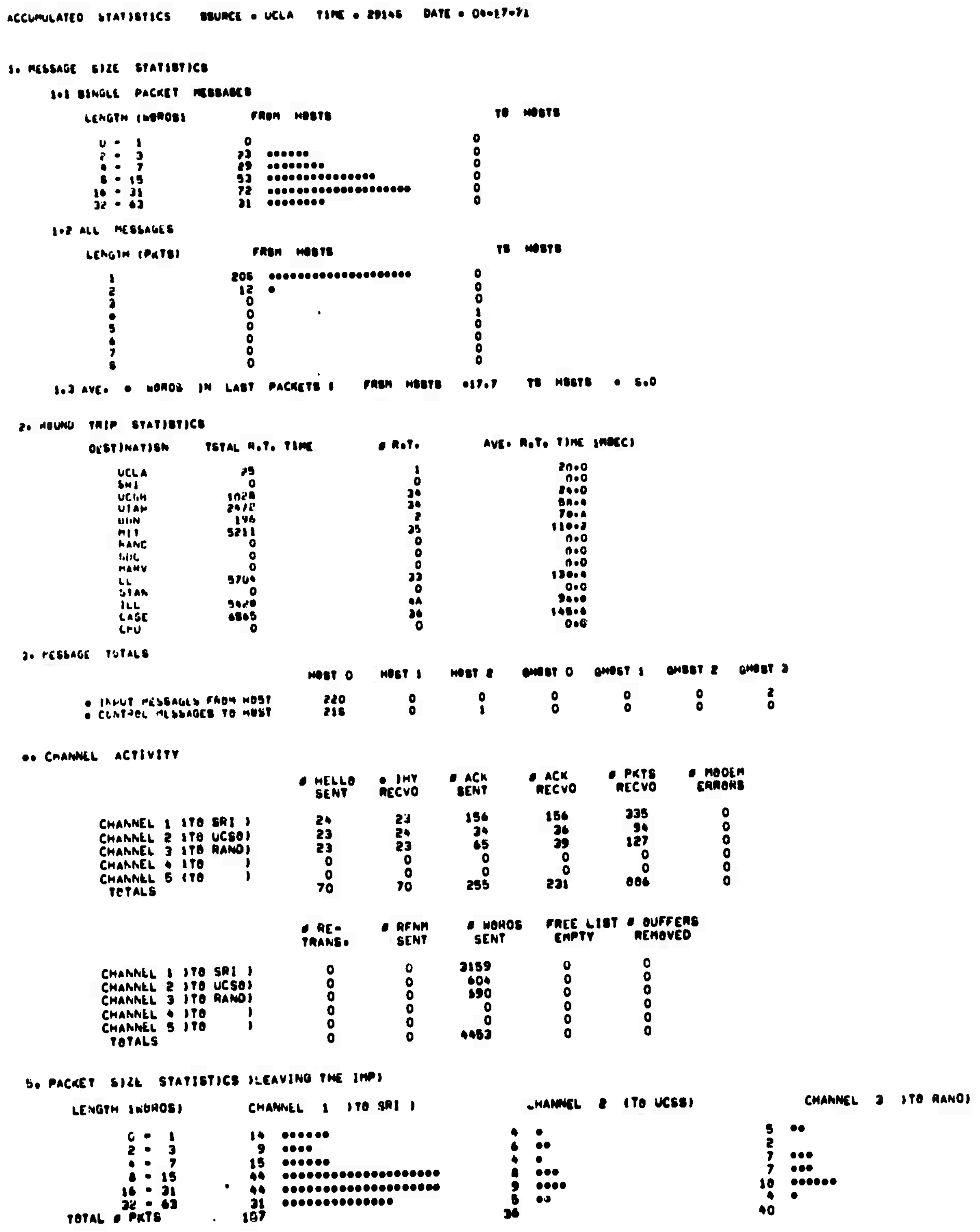
* An ACK is an acknowledgement, i.e., a short transmission between IMPs to indicate the successful reception of a message segment (referred to as a packet).

** A RFNM (Request For Next Message) is a short transmission which is sent back to the source of a message from the destination IMP. The RFNM is part of a flow control mechanism which involves logical links between source-destination pairs, with a given link becoming blocked after use until the receipt of a RFNM.

***A Fake HOST is a background program in the IMP. These programs include the statistics generation routines, the debug facilities, and the IMP console Teletype handler.

* Artifact in measurement data refers to any effect that results in a difference between the actual system variables (as they would occur without any measurements) and the observed data.

150



```
ACCUMULATED  STATISTICS    SOURCE = UCLA   TIME = 29146   DATE = 04-17-71


1. MESSAGE  SIZE  STATISTICS
     1.1 SINGLE  PACKET  MESSAGES
          LENGTH (WORDS)          FROM  HOSTS                         TO  HOSTS

            0 -  1               0                                   0
            2 -  3              23  ......                           0
            4 -  7              29  ........                         0
            8 - 15              53  ...............                  0
           16 - 31              72  ....................             0
           32 - 63              31  ........                         0
     1.2 ALL  MESSAGES
          LENGTH (PKTS)           FROM  HOSTS                         TO  HOSTS

             1                 205  ....................             0
             2                  12  .                                0
             3                   0                                   0
             4                   0                                   1
             5                   0                                   0
             6                   0                                   0
             7                   0                                   0
             8                   0                                   0
     1.3 AVE. # WORDS  IN  LAST  PACKETS :    FROM  HOSTS  =17.7    TO  HOSTS   =  5.0

2. ROUND  TRIP  STATISTICS
          DESTINATION    TOTAL R.T. TIME          # R.T.          AVE. R.T. TIME (MSEC)

             UCLA              25                    1                   20.0
             SRI                0                    0                    0.0
             UCSB            1028                   34                   24.0
             UTAH            2472                   34                   68.4
             BBN              196                    2                   70.A
             MIT             5211                   35                  110.2
             RAND               0                    0                    0.0
             SDC                0                    0                    0.0
             HARV               0                    0                    0.0
             LL              5704                   33                  130.4
             STAN               0                    0                    0.0
             ILL             5420                   4A                   94.0
             CASE            6865                   36                  148.6
             CMU                0                    0                    0.0

3. MESSAGE  TOTALS
                                          HOST 0   HOST 1   HOST 2   GHOST 0   GHOST 1   GHOST 2   GHOST 3

       # INPUT MESSAGES FROM HOST           220       0        0        0         0         0         2
       # CONTROL MESSAGES TO HOST           215       0        1        0         0         0         0

4. CHANNEL  ACTIVITY
                                      # HELLO   # IHY    # ACK     # ACK    # PKTS    # MODEM
                                        SENT    RECVD     SENT     RECVD    RECVD     ERRORS

       CHANNEL 1 (TO SRI )               24       23      156       156      335        0
       CHANNEL 2 (TO UCSB)               23       24       34        36       94        0
       CHANNEL 3 (TO RAND)               23       23       65        39      127        0
       CHANNEL 4 (TO     )                0        0        0         0        0        0
       CHANNEL 5 (TO     )                0        0        0         0        0        0
        TOTALS                           70       70      255       231      006        0

                                      # RE-    # RFNM    # WORDS  FREE LIST # BUFFERS
                                      TRANS.    SENT      SENT      EMPTY    REMOVED

       CHANNEL 1 (TO SRI )               0        0      3159        0         0
       CHANNEL 2 (TO UCSB)               0        0       604        0         0
       CHANNEL 3 (TO RAND)               0        0       690        0         0
       CHANNEL 4 (TO     )               0        0         0        0         0
       CHANNEL 5 (TO     )               0        0         0        0         0
        TOTALS                           0        0      4453        0         0

5. PACKET  SIZE  STATISTICS (LEAVING THE IMP)
      LENGTH (WORDS)       CHANNEL  1  (TO SRI )            CHANNEL  2  (TO UCSB)          CHANNEL  3  (TO RAND)

           0 -  1           14  ......                       4  .                            5  ..
           2 -  3            9  ....                         6  ..                           2
           4 -  7           15  ......                       4  .                            7  ...
           8 - 15           44  ....................         8  ...                          7  ...
          16 - 31           44  ....................         9  ....                        10  ......
          32 - 63           31  ..............               5  ..                           4  .
     TOTAL # PKTS          187                              36                              40
```

Figure 1. A Typical Accumulated Statistic Print-Out.

race Statistics

The trace capability allows one to literally
ollow a message through the network, and to learn
the route which it takes and the delays which
encounters. However, it is one of the more
mplicated of the measurement data formats to
scribe because the interpretation of the data
pends on the function of the IMP handling the
ssage, i.e., either source, store-and-forward,
destination, as well as depending on the possi-
lity of retransmissions.

A typical store and forward situation will,
en traced, result in a record of (1) the time
arrival, (2) the time at which the message is
ocessed, i.e., put on an output queue, (3) the
me at which transmission is initiated, and (4)
e time when the ACK is received. In the case of
retransmission, this latter time is the time of
transmission, and is so indicated by a tag bit
the data block. At the destination IMP, this
urth word is the time at which the IMP-to-HOST
ansmission was completed. All times are
corded in terms of a clock with a 100 μsec.
solution.

The other information in the trace data mes-
ge includes the output channel (if on a modem
annel), the HOST number, or the "fake HOST"
nber, as well as the entire header of the packet.
e header contains the source, the destination,
e link number, the message number, the packet
nber, and the priority/non-priority status.

tificial Traffic Generation

The measurement facilities of the network are
gmented by artificial traffic generation rou-
nes both in the IMPs and at the NMC. Such
tificial traffic was helpful in earlier "shake-
wn" tests of the network operation, and more
cently has been a necessary tool in our work
lating measurements and modeling.

Each IMP can generate artificial traffic
nsisting of fixed length messages that are
riodically sent on a single link at a selected
ne interval, but such traffic is not sufficient
simulate congestion at the IMP's due to the
ow control mechanism which is built into the
nk/RFNM operation. A more powerful artificial
affic generator was therefore developed as part
the NMC capabilities, and includes the ability
send messages on up to 64 links, to multiple
stinations, with either fixed or random message
ngths and inter-transmission times. The traffic
ich was utilized in the example of Figure 1 was
nulated by this latter facility utilizing the
ndom variable features as indicated by the
ssage length histograms.

. Measurements Related to Network Applications

Applications of the network were initially
nited by the lack of an established protocol
r HOST-to-HOST transmission, although several
tes made use of the network by establishing
special purpose protocol subsets for handling
their particular functions of interest such as
interactive work, file transmissions, or graphics
display functions. Rather than summarizing the
measurement facilities relative to a number of
these application areas, we will present a more
detailed accounting of the measurements that were
taken on the activity of one set of users.

The activity of interest involved the use
of the DEC PDP-10 at the University of Utah and
the XDS 940 at SRI, with the general nature of
the interaction being as follows. A file would
be sent from SRI to Utah, and then a number of
interactive debug operations would be performed,
with transmission from the XDS 940 to the PDP-10
being on a character-by-character basis, and
responses being handled on a line at a time basis.
The files were transmitted in blocks of 4608 bits,
which was a convenient multiple of the 24-bit
word length of the XDS 940 and the 36-bit word
length of the PDP-10, and also provided the
desired block size of 128 words for the PDP-10.

The established protocol involved several
special characters to indicate the message type
and length, such that a single character message
required five IMP-words of transmitted data.
Due to these character-by-character and file
transmission protocol conventions, almost all of
the SRI-to-Utah transmissions consisted of either
5-word messages or 5-packet messages respectively,
as shown in the sample print-out of Table 1.
Each line corresponds to one 12.8 second statis-
tics message, with the time-of-day being deter-
mined by adding 12.8 second increments to the
starting time since the data values were taken
consecutively. The sequence of events which is
shown includes a 10-word "file name" message,
the file transmission of 67 5-packet messages
plus a 4-packet file fragment, an "end of file"
message, and a sequence of interactive traffic.
This particular file transmission consisted of
approximately 39,300 bytes and occurred over a
25.6 second period for an average bandwidth usage
of about 12.5 Kbits/second. The maximum file
transmission bandwidth observed during any 12.8
second interval was one in which 73 5-packet
messages were transmitted. Each message was
the standard 576 byte record, for a total band-
width usage of:

```
26.3 Kb/sec for data ("effective" data rate")
 0.3 Kb/sec for marking, padding, and protocol
       format
_3.9 Kb/sec for network header, checksum, etc.
30.5 Kb/sec total bandwidth used.
```

Transmission occurred during the entire
interval since the file transmission was measured
over three 12.8 sec. intervals, and this was the
center interval. The average round trip time
for a message was 159 msec. (measured from the
time the source IMP receives the first packet
until it received the RFNM). The minimum round
trip time would be about 138 msec. based on the
time for serial transmissions and propagation
delays. The extra delay of 159 minus 138, or

TABLE 1

A SUBSET OF THE ACCUMULATED STATISTICS MEASUREMENT OF THE SRI-UTAH TRAFFIC

			single packet messages*						multi-packet messages**						
time-of-day	avg. RT time	avg. # words	0-1	2-3	4-7	8-15	16-31	32-63	2	3	4	5	6	7	8
21:59•990	153•7	39•0	0	0	0	1	0	0	0	0	0	29	0	0	0
22:00•203	118•1	32•4	0	1	9	(file name)				0	1	38	0	0	0
22:00•417	26•4	5•0	end of		32	0	0	0	0	0	0	0	0	0	0
22:00•630	0•0	0•0	file		0	0	0	0	0	0	0	0	0	0	0
22:00•843	82•3	5•0	0	0	11	interactive				0	0	0	0	0	0
22:01•057	23•9	5•0	0	0	6	traffic					file				0
22:01•270	25•3	5•0	0	0	12	0	0	0	0		transmission				0
22:01•483	27•6	5•0	0	0	19	0	0	0	0						0
22:01•696	0•0	0•0	0	0	0	0	0	0	0	0	0	0	0	0	0
22:01•910	0•0	0•0	0	0	0	0	0	0	0	0	0	0	0	0	0
22:02•123	0•0	0•0	0	0	0	0	0	0	0	0	0	0	0	0	0
22:02•336	133•8	37•8	0	1	0	1	0	0	0	0	0	34	0	0	0
22:02•550	0•0	0•0	0	0	0	0	0	0	0	0	0	0	0	0	0
22:02•763	0•0	0•0	0	0	0	0	0	0	0	0	0	0	0	0	0
22:02•976	29•4	5•0	0	0	3	0	0	0	0	0	0	0	0	0	0
22:03•190	76•2	5•0	0	0	11	0	0	0	0	0	0	0	0	0	0
22:03•403	55•4	5•0	0	0	14	interactive				0	0	0	0	0	0
22:03•616	24•8	5•0	0	0	8	traffic				0	0	0	0	0	0
22:03•830	60•8	5•0	0	0	13					0	0	0	0	0	0
22:04•043	0•0	0•0	0	0	0	0	0	0	0	0	0	0	0	0	0

* MEASURED IN TERMS OF IMP WORDS
** MEASURED IN TERMS OF THE NUMBER OF PACKETS PER MESSAGE

21 msec., was due to the delays in the PDP-10 acceptance of messages. The effective round trip cycle must include an additional 12 msec. for HOST processing of the RFNM and for the leader and first packet transmission from the HOST to the IMP. Therefore, the maximum single link data rate is approximately one 576 byte record every 150 msec. or 30.7 Kb/sec. of actual data, compared with the observed maximum effective data rate of 26.3 Kb/sec. A similar calculation using the measured average round trip time and the resulting effective data rate indicates that about 4 to 5 msec. are required for HOST processing of the RFNM and setting up the next transmission.

The round trip time required for messages between the two HOST's was plotted in Figure 2, as a function of message length L. Each point of this scatter plot represents one interactive message, i.e., a one-packet message. The scattering is an indication of the frequency of occurrence of both message length variations and round trip delay variations. A definite lower limit of delay was observed as indicated by the dashed line, and this limiting delay function is just what would be expected based on estimated values of the various components of delay. In this minimum delay case, there is no queueing time such that the expected delay consists of:

• Serial trans. at source IMP	= 2.0 + 0.32L	msec.
• Propagation & modem delay	= 7.5	msec.
• Processing time at dest.	= 0.5	msec.
• Serial IMP-HOST trans.	= 0.3 + 0.16L	msec.
• Serial transmission of RFNM	= 2.0	msec.
• Propogation and modem delay	= 7.5	msec.
• Processing of RFNM	= 0.2	msec.
Total	20.0 + 0.48L	msec.

This estimated total delay agrees very well with the experimental minimum delay values as shown in Figure 2. The other points in the figure represent messages whose delay values exceeded the minimum due to queueing delays and/or HOST delays in taking the message from the IMP.

4. Measurements to Determine Network Performance and Limiting Values

Several experiments were conducted to evaluate

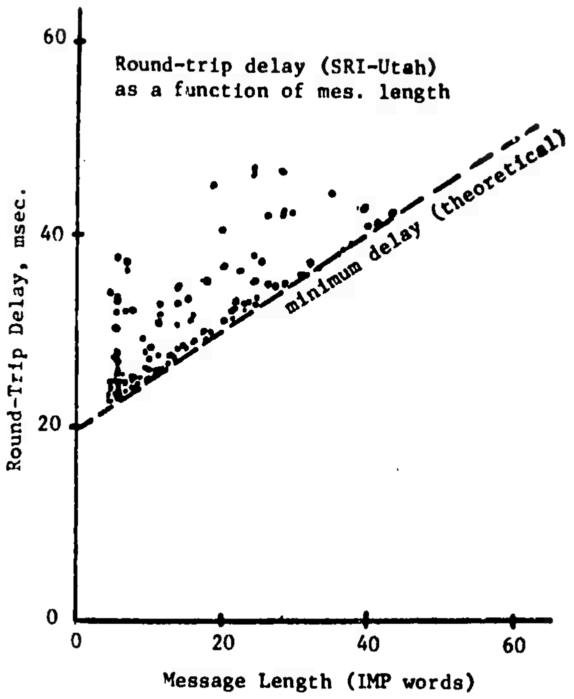


Figure 2. Round-Trip Delay Measurements of the SRI-Utah Traffic.

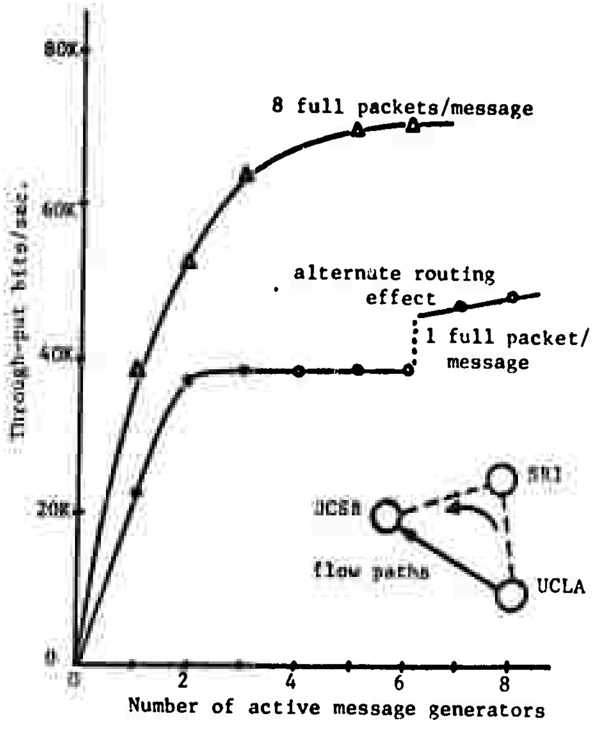


Figure 3. Through-put Measurements Between UCLA and the Neighboring UCSB IMP

various aspects of the network performance at different traffic levels, and to determine parameter sensitivities and limiting values. The following example is typical of these experiments, and involves the measurement of the through-put between two nodes with the effect of alternate routing being considered by both analytic and measurement techniques. The purpose of the example is to show how the two techniques are complementary, with the measurements providing the insight needed to improve and correct the analytic model of the network behavior.

The initial through-put tests were conducted by sending artificial traffic from the NMC at UCLA to the discard "fake HOST" in the UCSB IMP, and measuring the resulting through-put as a function of the number of links utilized in the transmission.

The communication lines between the IMP's have a bandwidth of 50 Kbits/second, which establishes an upper bound on the single path through-put. However, the overhead associated with each message limits the total through-put to some lesser value as shown in Figure 3. The same figure also indicates the measured through-put for the case of alternate routing which exceeds the 50 Kbit/second value due to the bifurcation of the traffic flow. In all cases the messages were sent at the RFNM driven rate, i.e., as soon as a link would become unblocked.

Two different message sizes were included in the above test. In the first case, the messages were one full packet in length so that the effective packet overhead was minimized (relative to the packet size). The maximum through-put for such messages was measured to be 38.6 Kbits/second, which agrees quite well with the theoretical value of 39.0 Kbits/second (based on 20.0 msec. of useful data transfer and 5.7 msec. for the packet overhead and the acknowledgement for the RFNM). This maximum through-put is not reached for the case of a single link due to the quiescent period while waiting for the RFNM to return. This delay is about 13.5 msec.,* which adds to the "non-productive" transmission period, and results in an expected single link through-put of 25.5 Kbits/second. (The measured value was 22.2 Kbits/second.) The through-put for the one-packet message case can be seen to increase abruptly when the transition is made from six to seven traffic generators, due to alternate routing via the SRI path.

The single link through-put for the full 8-packet message case was seen to be approximately equal to the saturation value of the single packet case, i.e., 38.5 Kbits/second. This equality is due to the canceling effects of fewer RFNM's to acknowledge, and the short quiescent delay for the single link RFNM. No alternate routing was observed until the two generator test, above which the traffic split evenly between the direct UCLA-UCSB and the UCLA-SRI-UCSB routes.

* Based on a measured round trip time from UCLA to UCSB of 22.8 msec. for a 320 bit message and the known service time of 9.3 msec. for a message of that length.

The saturation effect for the 8-packet message case occurred at a lower bandwidth than was expected, since the direct and alternate route paths should each be able to deliver messages with an effective rate of about 43.5 Kbits/second (which is higher than the single-packet through-put since there are fewer RFNM's to acknowledge). The measured limiting value of about 70 Kbits/second was therefore investigated further to try to determine the cause of this behavior. Several possible reasons for this discrepancy were considered and described below, since they represent the possible limiting factors in such transmissions and indicate the manner in which the statistics gathering tools can be utilized to determine the cause of anomalies.

a. Lack of store-and-forward buffers: In order to obtain the maximum through-put, there must be a sufficient quantity of store-and-forward buffers to be able to create a queue that is long enough to keep the channel busy during the entire routing update interval (which was thought to be about 520 msec.). Therefore, with a packet service time of 23.0 msec., the queue would have to build up to a total of about 23 buffers. Since we understood that there was a total of 25 store-and-forward buffers, this did not appear to be the limiting factor. Snap-shot measurements also showed that typically 18 to 20 store-and-forward buffers were in use, which seemed to substantiate this conclusion.

b. Possible bandwidth reduction effects: The effective transmission bandwidth can be reduced by a number of effects including line errors, lack of reassembly buffers, and routing loops. However, measurements indicated that none of these effects were causing a significant through-put reduction.

c. Interface and HOST computer system limitations: The transmission capabilities of the HOST and HOST-IMP interface were verified by directing the artificial traffic to the local (UCLA) discard "fake HOST" with an observed through-put of 91.5 Kbits/second, which far exceeded the UCLA-to-UCSB saturation value.

d. Trace measurements to determine the detailed behavior: Since the accumulated statistics and the snap-shots both indicated that the limitation had to be in the UCLA IMP, a detailed investigation of the packet handling behavior was made by tracing all of the packets at the UCLA IMP for a brief instant of time. (At the traffic rates involved, about 100 packets/second were generated and each packet resulted in a separate trace message.) This trace data indicated two errors in the values of the IMP parameters that had been used in the calculations of part (a) above. First the routing updates were found to occur at approximately 640 msec. intervals instead of the 520 msec. that this parameter was previously understood to be, and secondly, the upper limit on the store-and-forward buffers on an output queue must be approximately 20 (instead of 25.)* If we reconsider the effect of the finite buffering with these new parameter values, we find that the first channel will have approximately 18 packets to serve when the routing change occurs, and at 23.0 msec. of service time per packet, that channel will remain busy for about 420 msec.** Since the next routing update will occur after 640 msec., the first channel will be inactive for 220 of the 640 msec., for an effective through-put contribution of:

$$\text{through-put} = \frac{420}{640} \cdot (43.5 \text{ Kbits/sec.})$$
$$= 28.5 \text{ Kbits/sec.}$$

for a total estimated through-put of 72 Kbits/second, compared to the measured 70 Kbits/second, which is a reasonable match between the theory and measurement.

This test activity was described in some detail to indicate how the various measurement tools can be utilized to reconcile differences between the expected and experimental data values. However, it also points out an equally important aspect of the experimentation; namely that those people involved in the network measurement and modeling efforts must be aware of any changes in the network parameters if they are to be able to properly interpret the test results.

6. Summary

The measurement facilities which have been included in the design of the network IMPs have been very useful for network performance evaluation, and for gaining insight to determine the sensitivities and limiting effects of the network behavior. This insight has also been valuable for improvement of analytic models, by comparing the model behavior to the actual measured network behavior.

References

1. Roberts, L. G. and B. D. Wessler, "Computer Network Development to Achieve Resource Sharing," Proc. of SJCC, Vol. 36, AFIPS Press, pp. 543-549.

2. Heart, F. E., et al., "The Interface Message Processor for the ARPA Computer Network," Proc. of SJCC, Vol. 36, AFIPS Press, pp. 551-567.

3. BBN Report No. 1822, "Specification for the Connection of a HOST and an IMP," Bolt, Beranek and Newman, Inc., Cambridge, Mass., Feb. 1970.

* A subsequent check with BBN showed that the actual routing update interval is 655 msec. The upper limit on store-and-forward buffers is 25 (octal) which is 21 (decimal).

** The other two store-and-forward packets are assumed to be required for the SENT queue.