

AD-A131 915

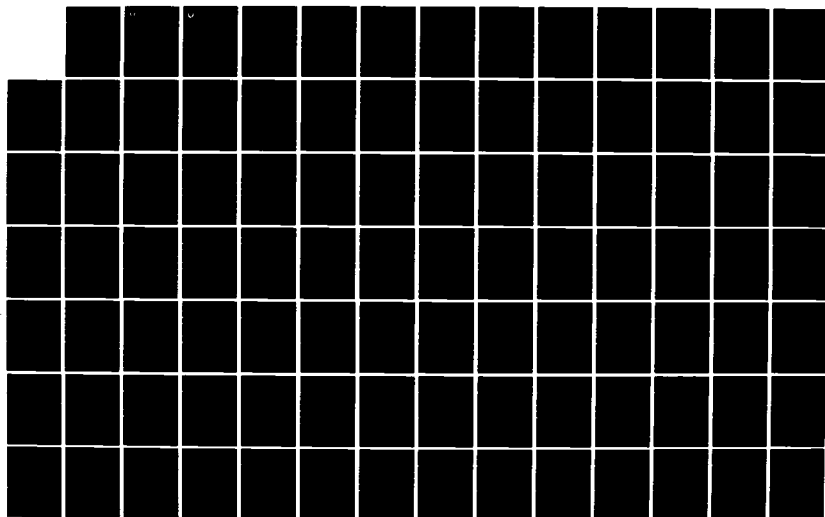
RESEARCH IN NETWORK MANAGEMENT TECHNIQUES FOR TACTICAL
DATA COMMUNICATION. (U) POLYTECHNIC INST OF NEW YORK
BROOKLYN R R BOORSTYN ET AL. 01 SEP 82 CECOM-80-0579-F

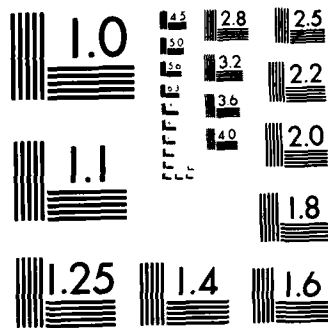
1/5

UNCLASSIFIED

DAK80-80-K-0579

F/G 17/2.1 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A



①

RESEARCH AND DEVELOPMENT TECHNICAL REPORT
CECOM

POLYTECHNIC INSTITUTE OF NEW YORK

RESEARCH IN COMPUTER COMMUNICATIONS

US ARMY (CECOM)

September 1980 to August 1982

Principal Investigators:

Robert Boorstyn

Aaron Kershenbaum

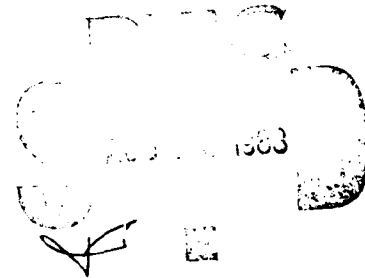
Basil Maglaris

Philip Sarachik

Polytechnic Institute of New York

333 Jay Street

Brooklyn, New York 11201



CECOM

U S ARMY COMMUNICATIONS-ELECTRONICS COMMAND
FORT MONMOUTH, NEW JERSEY 07703

for
dist

83 08 22 077

ALAJ 31915

DTIC FILE COPY



DEPARTMENT OF THE ARMY
HEADQUARTERS UNITED STATES ARMY COMMUNICATIONS
RESEARCH AND DEVELOPMENT COMMAND
FORT MONMOUTH, NEW JERSEY 07703

DRSEL-COM-RF-2

29 Nov 1982

Polytechnic Institute of New York
333 Jay Street
Brooklyn, NY 11201

R. Boerstyn

NETCOM 420199

Gentlemen:

The inclosed draft of the Final Technical Report under Contract DA AK-80-80-K-0579 has been reviewed and found acceptable. The draft, as amended, is approved for reproduction and distribution.

In accordance with contractual requirements, it is requested that the report be distributed according to:

☐ Attached distribution list	☒ Previously forwarded distribution list	☐ Attached changes to previous list

A copy of the distribution list should be inserted inside the back cover of each report distributed.

Upon depositing the requisite copies of the report, properly addressed and with adequate postage, in the U. S. Mail system, one copy of this letter of approval shall be endorsed with a certificate of mailing, dated, signed by the Contractor and returned to the undersigned as evidence that distribution has been completed. In addition, one (1) copy of the indorsement covering mailing of requisite copies of report will be submitted with the invoice to the Finance and Accounting Officer specified in the contract.

This letter is subject to the understanding that it does not authorize any increase in cost to the Government or change in delivery schedule, and no action taken by you thereunder will result in any such change. Prompt notification of any decrease in cost shall be furnished to the Contracting Officer.

Sincerely yours,

Charles J. Griff
Contracting Officer's Representative

Copy Furnished:
Contr Off, AMSEL—

Date _____

This is to certify that this organization on the above date mailed, in the manner prescribed above, the requisite copies of _____ Report under Contract DA _____ to all addressees on the distribution list.

Contractor

By _____

NOTICES

Disclaimers

The citation of trade names and names of manufacturers in this report is not to be construed as official Government indorsement or approval of commercial products or services referenced herein.

Disposition

Destroy this report when it is no longer needed. Do not return it to the originator.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER <i>CECOM-80-0579-F</i>	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Research in Network Management Techniques for Tactical Data Communications Networks		5. TYPE OF REPORT & PERIOD COVERED Final 1 Sept 80 - 31 Aug 82
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Dr. Robert Boorstyn, et al		8. CONTRACT OR GRANT NUMBER(s) DAAK-80-80-K-0579
9. PERFORMING ORGANIZATION NAME AND ADDRESS Polytechnic Institute of New York 333 Jay Street Brooklyn, NY 11201		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS CDR USA CECOM DRSEL-COM-RF-2 (CHARLES J. GRAFF) Fort Monmouth, NJ 07703		12. REPORT DATE 1 Sept 82
		13. NUMBER OF PAGES 285
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLAS
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Cleared for Public Release Distribution Unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Network Management, Network Control, Routing, Packet Switching, Network		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This is the final technical report for work performed on network management techniques for tactical data networks. It includes all technical papers that have been published during the control period. Research areas include Packet Network modelling, adaptive network routing, network design algorithms, network design techniques, switching techniques, and local area networks.		

TABLE OF CONTENTS

Summary of Report

Personnel

Activities

Research Reports

A. Packet Radio Networks

A.1 Throughput Analysis of Multihop Packet Radio Networks

Boorstyn, Kershenbaum, and Sahin

Accepted for Publication in IEEE Transactions on
Communications

A.2 Throughput Analysis of Multihop Packet Radio

Boorstyn and Kershenbaum

IEEE International Conference on Communications,
June 1980, Seattle

A.3 A New Acknowledgment Protocol for Analysis of Multi-
hop Packet Radio Networks

Boorstyn, Kershenbaum, and Sahin

IEEE COMPCON, September 1982, Washington

A.4 Extensions to the Analysis of Multihop Packet Radio
Networks

Maglaris, Boorstyn, and Kershenbaum

IEEE INFOCOM '83, April 1983, San Diego

A.5 Multiple Access Techniques with Arbitrary Packet
Length Distributions

Third Semiannual Technical Report, March 1982

A.6 Evaluation of Throughput in Multihop Packet Radio
Networks with Complex Topologies

Kershenbaum and Boorstyn

A.7 Optimal Transceiver Deployment in Multihop Packet
Radio Networks with Capture*

Maglaris and Kaminer

NSF Final Report, October 1981

B. Adaptive Routing

B.1 A Technique for Adaptive Routing in Networks

IEEE Transactions on Communications, April 1981

Boorstyn and Livne

B.2 A Simulation Study of a Dynamic Routing Scheme

Chu, Boorstyn, and Kershenbaum

IEEE National Telecommunications Conference, November
1981, New Orleans

B.3 Stable Routing Patterns

Third Semiannual Technical Report, March 1982

B.4 Decentralized Dynamic Clearing of Congested Multi-
Destination Networks*

Sarachik

Allerton Conference on Communication, Control, and
Computing, October 1981, Urbana

B.5 An Effective Local Dynamic Strategy to Clear Congested
Multi-Destination Networks*

Sarachik

IEEE Transactions on Automatic Control, April 1982

B.6 A Dynamic Alternate Route Strategy for Traffic Networks

Sarachik

IEEE Conference on Decision and Control, December 1982, Orlando

B.7 Optimal Control of M/M/2 Queues

Maglaris

C. Algorithms

C.1 A Shortest Path Algorithm for the Solution of the Simple Knapsack Problem and Extensions

Kershenbaum

Second Semiannual Technical Report, September 1981

C.2 A Network Shortest Path Approach to the Knapsack Problem

Kershenbaum

IEEE International. Conference on Circuits and Computers, October 1980, New York

C.3 Generalized Augmenting Paths for the Solution of Combinatorial Optimization Problems

Kershenbaum

Tenth IFIPS Conference on System Modeling and Optimization, August 1981, New York, Proceedings published by Springer-Verlag, 1982

C.4 A Note on Finding Shortest Path Trees

Kershenbaum

Networks Journal, 1981

C.5 Probabilistic Analysis of Algorithm Performance

First Semiannual Technical Report, March 1981

C.6 Probabilistic Analysis of Algorithms

Second Semiannual Technical Report, September 1981

D. Network Design

D.1 Second-Order Greedy Algorithms for Centralized Tele-
processing Network Design

Kershenbaum, Boorstyn, and Oppenheim

IEEE Transactions on Communications, October 1980

D.2 Centralized Teleprocessing Network Design

Kershenbaum and Boorstyn

To be published in Networks Journal

D.3 Network Design with an Objective Function Including
a Fixed Charge (Revised)

Kershenbaum

IEEE International Conference on Communications, June
1982, Philadelphia

D.4 An Algorithm for Designing Circuit Switched Networks*

Kershenbaum, Schneider, and Frisch

IEEE International. Conference on Circuits and
Computers, October 1980, New York

D.5 Optimization of Telephone Networks Using the New
WATS Tariff*

Kershenbaum and Kozicki

Journal of Telecommunication Networks, Summer 1982

D.6 Design of Survivable Circuit-Switched Communication
Networks*

Natarajan, Walters, and Maglaris

IEEE MILCOM, September 1982, Boston

E. Switching Techniques

E.1 Delay and Overhead in the Encoding of Data Sources

Hayes and Boorstyn

IEEE Transactions on Communications, November 1981

E.2 Delay and Overhead in the Encoding of Bursty Sources

Boorstyn and Hayes

IEEE International Conference on Communications,

June 1980, Seattle

E.3 An Optimal Strategy for Packetization

Tsao, Kershenbaum, and Boorstyn

E.4 Optimal Fixed Frame Multiplexing in Integrated Line-
and Packet-Switched Communication Networks*

Maglaris and Schwartz

IEEE Transactions on Information Theory, March 1982

E.5 Satellite Access Methods for a General Purpose Packet
Network with a Time Critical Traffic Component*

Maglaris and Lissack

To be published in Journal of Telecommunication Net-
works

F. Local Area Networks

F.1 End-to-End Delay Analysis on Local Area Networks:

An Office Building Scenario

Maglaris, Lissack, and Austin

IEEE National Telecommunications Conference, November
1981, New Orleans

F.2 Performance Evaluation of Interface Units for Broad-
cast Local Area Networks*

IEEE COMPCON, September 1982, Washington

Maglaris and Lissack

* Research supported by other agencies.

SUMMARY OF REPORT

The report consists of papers written in six research areas. Most of these papers have been published in journals or conference proceedings, or are soon to be published. A few are taken from earlier semi-annual reports.

The six research areas are Packet Radio Networks, Adaptive Routing, Algorithms, Network Design, Switching Techniques, and Local Area Networks.

In the area of Packet Radio Networks, our results on the throughput analysis of Multihop Packet Radio Networks was first presented at ICC'80 (A.1). An expanded version with extensions has been accepted for publication in the IEEE Transactions on Communications (A.2). Further extensions are presented in papers A.3, A.4, and A.5. In this work we developed an algorithm for finding the throughput of Multihop Packet Radio Networks with arbitrary topologies, sizes, connectivities, traffic requirements, and routings. Details of the algorithm are given in paper A.6.

Our work on Adaptive Routing was first presented at NTC'76. It was further reported on in the IEEE Transaction on Communications in an invited paper in April 1981 (B.1). Extensions to this work are presented in papers B.2, B.3, B.6, and B7. Our approach involved two levels for adaptive routing - a slowly varying global level and a dynamic local level. This partition produced a stable routing that was amenable to analysis. We showed that significant performance improvements were obtainable. A control theoretic approach applied to traffic networks is presented in papers B.4 and B.5.

Our work in the area of algorithms has focused on finding approaches to solving intractable problems which arise in the design and analysis of networks. Several of these papers (C.1, C.2, and C.4) deal with the solution of specific fundamental problems. Others (C.5, C.6) deal with the application of probabilistic methods to the solution of a broad class of problems. Finally, in C.3, we present a method of directly trading computational complexity for solution accuracy in the solution of combinatorial optimization problems.

Our work in the area of Network Design has led to solutions to specific problems in several areas including centralized networks (D.1, D.2), and circuit switched networks (D.4, D.5, and D.6). It has also resulted in the development of a facility location algorithm (D.3) which has been applied to the location of earth stations and packet radio repeaters in broadcast networks.

In section E, we present our work on an Information Theoretic approach to communications in networks (E.1 and E.2), on a comparison of packet switching and character switching, including an optimum packetization strategy (E.3), on optimal multiplexing in Integrated Switching schemes (E.4) and on Integrated Satellite Access (E.5).

In papers F.1 and F.2, we present some work on Local Area Networks for which we evaluate performance and assess the impact of user interface architecture.

Most of the work reported upon here was supported by the US ARMY (CECOM). Portions of this work were also supported by the National Science Foundation, by Network Analysis Corporation, by General Telephone and Electronics, and by IBM's Federal Systems Division. Research not supported under this grant is indicated by an

asterisk in the table of contents and is included to complement the other research reports and provide a more complete record of our activities.

PERSONNEL

Robert R. Boorstyn, Professor

Aaron Kershenbaum, Associate Professor

Basil Maglaris, Assistant Professor

Philip Sarachik, Professor

Students

Paul Chu, 1981

William Chuang, 1982

Veli Sahin, 1982

David Tsao, 1982

Mon-Song Chen

William Chen

H.K. Chao

Rachel Mendelsohn

ACTIVITIES

Papers published

Almost all of the papers published during this contract are included in this report and listed in the Table of Contents.

Talks presented

A paper on "Analysis of Multihop Packet Radio Networks," was presented by Professor Boorstyn at the URSI Conference, Boulder, January 1981. He also gave a talk on Dynamic Routing at Carleton University, Toronto at an all-day seminar on Networks co-sponsored by Carleton University, the IEEE Toronto Chapter, and Bell Northern Research. He also gave a talk on Analysis of Multihop Packet Radio Networks at Bell Laboratories. Professor Boorstyn was a panelist at IEEE INFOCOM'82, Las Vegas, April 1982 and spoke on Analysis of Packet Radio Networks. The panel discussed Analytic Techniques for Networks. Talks were also given at the University of Waterloo, Canada, CCNY, and Stevens Institute of Technology.

Professor Kershenbaum gave a talk on Multihop Packet Radio at the Packet Radio Symposium, UCLA, August 1982. He also gave talks on combinational optimization at the University of Rochester and at Stevens Institute of Technology.

Professor Sarachik gave talks on Decentralized Dynamic Clearing of Congested Multi-Destination Networks at an Engineering Foundation Conference-Workshop on "Issues in Control of Large Scale Urban

Traffic Systems," at Hennicker, N.H. in July 1981; and at a joint EE Department - IEEE Control System Society Colloquium held at Ohio State University in October 1981.

Professional activities

Professor Boorstyn has been the Chairman of the Computer Communications Committee of the IEEE Communications Society. He is on the Steering Committee of the IEEE INFOCOM Conferences. He is also Associate Editor of Networks Journal.

Professor Kershenbaum is the Editor for Network Analysis and Design Techniques for the Journal of Telecommunication Networks and is an Associate Editor of Networks Journal.

Theses completed

Paul Chu, 1981, A Dynamic Routing Scheme in Packet Switching Networks .

William Chuang, 1982, Probabilistic Models for Large Scale Optimization Problems.

Veli Sahin, 1982, Analysis of Multihop Packet Radio Networks.

David Tsao, 1982, A Comparison of Switching Techniques for Data Networks.

Research Reports

A. Packet Radio Networks

A.1 Throughput Analysis of Multihop Packet Radio Networks

Boorstyn, Kershenbaum, and Sahin

Accepted for Publication in IEEE Transactions on
Communications

*THROUGHPUT ANALYSIS OF MULTIHOP PACKET RADIO NETWORKS**

Robert R. Boorstyn and
Aaron Kershenbaum

Polytechnic Institute of
New York
333 Jay Street,
Brooklyn, New York 11201

Veli Sahin

Bell Laboratories
Holmdel, New Jersey 07733

ABSTRACT

We consider the problem of obtaining exact expressions for throughput and blocking probabilities in multihop packet radio networks operating under CSMA. We obtain exact results for a general class of message lengths, for general topologies, and for perfect capture. These results are obtained by assuming perfect acknowledgments.

I. INTRODUCTION

We consider the problem of obtaining exact expressions for throughput and blocking probabilities in multihop packet radio networks operating under carrier sense multiple access (CSMA). Procedures are developed which can be used to analyze general topologies for a general class of packet length distribution. Examples of chains, rings, and stars are presented.

II. THE NETWORK MODEL

We consider the problem of analyzing the throughput capability of a multihop packet radio network operating under carrier sense multiple access (CSMA). Thus, we assume that the network is comprised of terminals equipped with radio transponders suitable for broadcasting data over a limited distance. In general, the source and destination terminals

* This research was partially supported by USARMY CENCOMS under contract DAAK 80-80-K-0579, and by the National Science Foundation under grant ENG-79-08210.

cannot hear each other directly, and the data has to be relayed by one or more intermediate devices. A separate set of devices, called repeaters, may exist for this purpose, or the terminals themselves may relay messages for one another.

Control of the network is completely distributed, i.e., no station or central control mechanism is assumed to exist. Rather, we assume that each source terminal has prestored one or more routes to all destinations and includes all necessary routing information in the packets it transmits. These assumptions are made merely to simplify the presentation. In fact, the results presented are valid for networks using alternate routing as long as routing changes are not made over short time intervals. One of the motivations for this study came from a consideration of the design of routing procedures for such networks. It was necessary, however, to first develop an understanding for the throughput of various topologies.

Exogenous traffic is modeled as independent Poisson processes arriving at each source node, with appropriate rates and packet lengths. The topology is specified by a listing of which terminals (or repeaters) can hear each other. In the remainder we will not distinguish between terminals and repeaters and will refer to them collectively as either terminals or nodes. In general the transmissions of one terminal can be heard by many other terminals. The routing will specify which terminal is to repeat the packet, if necessary.

If two or more transmissions are simultaneously heard by a terminal (called a "collision") at least one, and possibly both, is "lost" and must be retransmitted. We assume retransmissions are scheduled at a random instant in time sufficiently far in the future so as to preserve the Poisson nature of the combined traffic stream, which now consists of exogenous traffic and rescheduled traffic. For this study, we assume that a packet can be retransmitted as many times as is necessary, i.e., that there is no maximum allowable number of retransmissions.

At any time, terminals may either transmit or receive, they cannot do both simultaneously. Before transmitting, a terminal senses the channel. If it detects that any of its

neighbors (i.e., terminals that it can hear) are transmitting (by, e.g., sensing a carrier) it reschedules the transmission as for collided packets above. If at the scheduled time for a transmission, the terminal is already engaged in transmitting a packet, the new packet is also rescheduled as above. Thus packets are continually rescheduled until they are successfully delivered to the next terminal on their route. We assume that the total stream of traffic scheduled by any terminal is a Poisson process. This includes originating traffic and packets rescheduled either due to collisions or due to the channel having been sensed busy. This scheme is called carrier sense multiple access (CSMA). The Poisson assumption is valid for the assumptions made above and will yield accurate results for throughput. Compromises will have to be made, however, if an accurate picture of time delays is to be considered.

It is possible, due to non-zero propagation delay, that collisions of transmissions from neighboring terminals may still take place despite the CSMA strategy. This will occur if a terminal senses the channel before another terminal's transmission is received. This effect is small if terminals are reasonably close or are not transmitting at high speed. We will ignore this phenomena here, and assume that all transmissions are instantly heard by their neighbors.

A passive acknowledgment is used for transmission to neighbors. The transmitting terminal listens to the channel to hear if a packet is being rebroadcast by a neighbor. If after a prespecified time interval, the transmitting node does not hear the packet rebroadcast, it retransmits the packet. But the packet may have been successfully received by the neighbor even though the originator does not hear the rebroadcast. Duplicate packets may be transmitted and deleted only at the final destination or they may be detected and deleted earlier. An end-to-end acknowledgment is returned to the originator from the final destination. In this paper we assume that passive acknowledgments are always heard and ignore the effect of end-to-end acknowledgments. Alternately, these acknowledgements could have been added to the required traffic. (In a sequel paper, the effect of passive acknowledgements will be studied).

We depict the topology of the network by a graph where terminals are represented

by nodes. The nodes are connected by a link if they can hear each other's transmissions, i.e., if they are neighbors. As an example see Figure 1. Node A can hear node B, but not node C. Node B can hear both nodes A and C. Node C can hear node B, but not node A. If node A is transmitting to node B and node C begins transmitting, then the transmission from A to B may be lost depending upon the "capture" assumptions we make. A conservative assumption is that the A to B transmission is lost - this is known as zero capture. Alternatively, perfect capture assumes that this transmission is successfully received. Half-amplitude capture assumes that the transmission is lost if C dominates A at B. This can happen if C is closer to B than A is to B, or has a greater signal strength perceived at B than A has. If A dominates C, then the transmission is successful. However, in all cases of a collision we assume the later transmission is lost. Thus if C is transmitting to B, this packet is lost in all cases. We will consider only perfect capture situations below. Note that under CSMA if node B is transmitting, neither A or C is allowed to transmit.

We assume that a routing has been specified. This takes the form of deciding which of the neighbors are to rebroadcast a packet from a particular source to a particular destination. Thus the amount of traffic that a terminal wishes to send to its neighbor can be computed. If these rebroadcast packets are scheduled at a random time far in the future the Poisson assumption for traffic streams is preserved. We assume that the traffic between neighbors is specified and form independent Poisson processes. We assume that the packet length is reassigned independently at each hop. This is analogous to the "independence assumption" in queuing networks.

The details of CSMA for a single hop network can be found in the papers by Tobagi and Kleinrock⁽¹⁾. Tobagi has also developed some simple models for two-hop networks⁽²⁾. Details of a packet radio network can be found in a paper by Kahn⁽³⁾. A discussion of routing in multihop packet radio can be found in the paper by Gitman, Van Slyke, and Frank⁽⁴⁾. An earlier version of this paper was presented at ICC'80⁽⁵⁾. More details can be found in the thesis of Sahin⁽⁶⁾.

III. GENERAL RESULTS

In this section we develop some expressions that are valid for the packet radio network we have modeled above using CSMA and with an arbitrary packet length distribution. Let i be a node, n_i one of its neighbors, N_i^* the set of all the neighbors of i , and N_i the set of all i 's neighbors, including i . Let g_i be the total rate (in packets/sec) of all scheduled traffic at node i . This includes originating traffic and all rescheduled traffic and is assumed to be Poisson. Let $1/\mu_i$ be the average length of packets transmitted by node i . Let $G_i = g_i/\mu_i$ be a normalized rate.

Node i is either busy (transmitting) or idle. It will transmit a scheduled packet at the instant it is scheduled all nodes in N_i are idle. Let A be a set of nodes. Let $P(A)$ be the probability that at a random instant all nodes in A are idle. The nodes not in A may or may not be idle. Similarly $P(i)$, $P(i,A)$, $P(A,B)$ are the probabilities that i is idle, node i and all nodes in A are idle, and all nodes in A and B are idle.

Since traffic is scheduled at node i with a Poisson rate g_i , will be transmitted if N_i is idle, and transmissions have average length $1/\mu_i$, the probability that i is busy is given by

$$1 - P(i) = G_i P(N_i)$$

If i is busy then under CSMA, n_i must be idle. Then since

$$P(n_i) = P(n_i, i) + P(n_i | i \text{ busy})[1 - P(i)]$$

and $P(n_i | i \text{ busy}) = 1$, we have

$$P(n_i, i) = P(n_i) + P(i) - 1 \quad (2)$$

Similarly, if $A \subseteq N_i^*$,

$$P(i|A) = P(i) - 1 \quad (3)$$

letting $A = N_i^*$ in equation (3), and using

$$P(N_i^*) = P(N_i)/P(i|N_i^*)$$

we get

$$P(i|N_i^*) = \frac{1}{1+G_i} \quad (4)$$

Equation (4) is often found in CSMA literature.

A packet from i to n_i will be transmitted when it is scheduled if N_i is idle. During the transmission all nodes in N_i^* will be idle. It will be successfully received at n_i if all neighbors of n_i not in N_i are also idle at the beginning of transmission. Otherwise a collision will occur. Let s_{i,n_i} be the rate in packets/sec. determined by the routing and assumed Poisson, of the traffic that i wishes to send to n_i . This is the required throughput or offered traffic. Let g_{i,n_i} be the rate of all scheduled traffic from i to n_i . We have also assumed that all these streams are Poisson and independent. Of these g_{i,n_i} packets per second, s_{i,n_i} must be successful. Thus

$$\frac{s_{i,n_i}}{g_{i,n_i}} = \frac{S_{i,n_i}}{G_{i,n_i}} = P(N_i, N_{n_i}) \quad (5)$$

where $S_{i,n_i} = s_{i,n_i}/\mu_i$ and $G_{i,n_i} = g_{i,n_i}/\mu_i$

The total scheduled traffic (normalized) at a node is given by

$$G_i = \sum_{n_i \in N_i^*} G_{i,n_i} \quad (6)$$

from equations (1) through (6) we will derive a relation between the S_{i,n_i} and G_i , and

determine the maximum $S_{i,n}$ the network can support. This we call the (maximum) throughput or capacity. In the next section we develop this relationship for exponential packet lengths and arbitrary topologies. Later we extend the analysis to a general class of packet length distributions.

IV. EXPONENTIALLY DISTRIBUTED PACKET LENGTHS

If the packet lengths are exponentially distributed, then the system can be viewed as a Markov process where the states are identified by which nodes are idle and which are busy. Let D be a set of busy nodes. Because of CSMA, no nodes in D may be neighbors of each other. Let $Q(D)$ be the probability that at an instant of time, all nodes in D are busy, and all nodes not in D , are idle. Then each set, D , represents a state in a Markov system, and $Q(D)$ is the state probability. In particular, the null set $D = \phi$, represents the state that all nodes are idle.

Assume the system is in state D . It will leave the state if any $i \in D$ stops transmitting. This happens with rate μ_i . Thus the transition to state $\{D-i\}$ occurs with rate μ_i . The only other way to leave state D is for one of the idle nodes that is not a neighbor of any $i \in D$ to begin to transmit. This occurs with rate g_j . Let N_D be the set of all neighbors of all nodes in D . Then the transition from D to $\{D+j\}$, $j \in N_D$, occurs with rate g_j . The global balance equations for this system are

$$\left(\sum_{i \in D} \mu_i + \sum_{j \notin N_D} g_j \right) Q(D) = \sum_{i \in D} g_i Q(D-i) + \sum_{j \notin N_D} \mu_j Q(D+j) \quad (7)$$

where D is one of the special sets defined above.

It is easy to see that these equations are satisfied by

$$Q(D) = \frac{g_i}{\mu_i} Q(D-i) = G_i Q(D-i), \quad i \in D \quad (8)$$

Thus

$$Q(D) = \left(\prod_{i \in D} G_i \right) Q(\phi) \quad (9)$$

where we adopt the convention that $\prod_{i \in \phi} G_i = 1$. Summing over all D, we get

$$\sum_{\text{all } D} Q(D) = \sum_{\text{all } D} \left[\prod_{i \in D} Q(\phi) \right] = 1 \quad (10)$$

In the previous section we found we were interested in quantities like $P(A)$, where A is any set of nodes, and $P(A)$ is the probability that all nodes in A are idle, and all nodes not in A may or may not be idle. This can be found by summing $Q(D)$ over all sets D that do not contain nodes in A. Thus

$$P(A) = \sum_{D \subset A^c} Q(D) = \frac{\sum_{D \subset A^c} \left(\prod_{i \in D} G_i \right)}{\sum_{D \subset N} \left(\prod_{i \in D} G_i \right)} \quad (11)$$

where $D \subset A^c$ refers to all such sets contained in the complement of A and N is the set of all nodes. We adopt the shorthand notation.

$$SP(B) = \sum_{D \subset B} \left(\prod_{i \in D} G_i \right) \quad (12)$$

where SP refers to sum of products. Thus

$$P(A) = SP(A^c)/SP(N) \quad (13)$$

Equations (5), (6), and (11) can be used for any topology to generate the solution to our problem. The equations relating the S_{i,n_i} , G_{i,n_i} , and G_i can be solved iteratively. For example, equation (5) now becomes

$$\frac{S_{i,n_i}}{G_{i,n_i}} = \frac{SP(\{N_i + N_{n_i}\}^c)}{SP(N)} \quad (14)$$

where by $A+B$ we mean the union of A and B .

Evaluation of sums of products in equation (12) are made easier by the following two rules. Consider two sets of nodes A and B such that no node in A can hear any node of B .

Then

$$SP(A+B) = SP(A) SP(B), A \cap B = \phi \quad (15)$$

Also,

$$SP(A) = SP(A-i) + G_i SP(A-N_i), i \in A \quad (16)$$

To prove these rules just consider all products. We have successfully evaluated many complex topologies with these procedures.

There are other relations which will be found useful in extending our model to more complex situations. We prove some of these below. Let C be a cut, i.e., a set of nodes that divides the network into three parts A , B , and C , where A and B have no neighbors in common as in equation (15). Let $A = A_1 + A_2$, $B = B_1 + B_2$ where $A_1 A_2 = B_1 B_2 = \phi$. Then

$$P(A_1|C, B_1) = \frac{P(A_1, C, B_1)}{P(C, B_1)} = \frac{SP(A_2 + B_2)}{SP(A + B_2)} = \frac{SP(A_2)}{SP(A)}$$

But

$$P(A_1|C) = \frac{P(A_1, C)}{P(C)} = \frac{SP(A_2 + B)}{SP(A + B)} = \frac{SP(A_2)}{SP(A)}$$

Thus

$$P(A_1|C, B_1) = P(A_1|C), C \text{ a cut} \quad (17)$$

we also have

$$P(A_1, B_1|C) = P(A_1|C)P(B_1|C), C \text{ a cut} \quad (18)$$

In particular if $C = N_i^*$, then

$$P(i|N_i^*, B) = P(i|N_i^*), B \subset N_i^c \quad (19)$$

V. A GENERAL CLASS OF PACKET LENGTH DISTRIBUTIONS

In this section we extend the results just proven to include a general class of distributions for the packet length. We will show that the procedures developed for perfect capture are independent of packet length distribution. To prove this we start with a simpler extension. In the above we assumed that all packet lengths are exponentially distributed and have the same mean when transmitted by a node to any of its neighbors. Different nodes may transmit different average length packets, however. Now assume that while all packet lengths are still exponentially distributed, the average length packet transmitted from a node may be different to each of its neighbors. This will be useful in analyzing different protocols (to be presented in a sequel paper) but is presented here as the first step in the desired extension.

Now the state of the network depends upon who is transmitting and to whom. We can keep the same structure by breaking every node into a set of "micronodes", one for each neighbor. These nodes may be indexed by (i, n_i) . If i is transmitting to n_i then this node is active, otherwise it is idle. Micronodes are connected in our topology if they can hear each other. Since CSMA still prevails, all micronodes for a given node are fully connected. Furthermore all micronodes for nodes that are connected in the original topology are also fully connected. The analysis now proceeds as above since the Markovian property has been maintained. For example, equation (14) still holds, but now N_i and N_{n_i} are collections of

miconodes. Note that N_i contains the full set of miconodes for i and all neighbors of i . S_{i,n_i} and G_{i,n_i} have the same meaning as before. However they are normalized by $1/\mu_{i,n_i}$, the average packet length for packets going from i to n_i . The terms in the sums of products are G_{i,n_i} for the miconodes.

Let A contain full sets of miconodes and include the node i . Then from equation (16)

$$SP(A) = SP[A - (i, n_i)] + G_{i,n_i} SP(A - N_i) \quad (20)$$

Here we have used the fact that $N_{(i,n_i)} = N_i$, and the notation that N_i is the set of all miconodes of i and neighbors of i . Since this is equivalent to the original set of node neighbors of i we keep the same notation. Repeating for all neighbors of i we get,

$$SP(A) = SP(A - i) + \left(\sum_{n_i} G_{i,n_i} \right) SP(A - N_i) \quad (21)$$

If we let

$$G_i = \frac{g_i}{\mu_i} = \sum_{n_i} G_{i,n_i} = \sum_{n_i} g_{i,n_i} / \mu_{i,n_i} \quad (22)$$

then equation (16) is preserved. In a similar manner all previously derived equations can be maintained where G_i takes on the definition in equation (22). Here $1/\mu_i$ is an average packet length, averaged over the different average packet lengths to different neighbors in proportion to their scheduled rate.

We now prove our main theorem for this section. Assume that the length of packets transmitted from i to n_i has the density

$$f_{i,n_i}(x) = \sum_j a_{i,n_i,j} \cdot \mu_{i,n_i,j} \cdot e^{-\mu_{i,n_i,j} x} \quad (23)$$

where

$$a_{i,n_i,j} \geq 0 \text{ and } \sum_j a_{i,n_i,j} = 1$$

Thus the length is distributed as a positive sum of exponentials. Another way of looking at this is that the a's are the probabilities of choosing the associated exponential density. Now create micronodes for each triple (i, n_i, j) . Here we use $S_{i,n_i,j} = s_{i,n_i,j} / \mu_{i,n_i,j}$ where $s_{i,n_i,j} = a_{i,n_i,j} S_{i,n_i}$. The micronodes for some i and any n_i, j are fully connected as are the micronodes for neighboring nodes. Equations (20) and (21) now become

$$SP(A) = SP[A - (i, n_i, j)] + G_{i,n_i,j} SP(A - N_i) \quad (24)$$

$$\text{and } SP(A) = SP(A - i) + \left[\sum_{n_i, j} G_{i,n_i,j} \right] SP(A - N_i) \quad (25)$$

In the same manner as above, we let

$$\begin{aligned} G_i &= \frac{g_i}{\mu_i} = \sum_{n_i, j} a_{i,n_i,j} = \sum_{n_i, j} \frac{g_{i,n_i,j}}{\mu_{i,n_i,j}} = \sum_{n_i} \left[\sum_j \frac{g_{i,n_i,j}}{\mu_{i,n_i,j}} \right] \\ &= \sum_{n_i} G_{i,n_i} \end{aligned}$$

These equations are used to find P(A) in terms of G_i and are identical in form to those derived for exponentially distributed packet lengths.

Equation (5) in turn comes from

$$S_{i,n_i,j} = G_{i,n_i,j} P(N_i + N_{n_i}). \quad (27)$$

Summing over j, we get equation (5)

$$S_{i,n_i} = \sum_j S_{i,n_i,j} = G_{i,n_i} P(N_i + N_{n_i}) \quad (28)$$

Thus all relations between the G's and the S's are preserved. The actual nature of the problem is taken into account by the relationship between the S's and the normalization by the μ 's.

We can now restate the above theorem. Let i be any node and $j \in N_i^*$, the neighborhood of i. Let $s_{i,j}$ be the successful (desired) rate from i to j, in packets/sec. Let the density of the packet lengths be

$$f_{i,j}(x) = \sum_k a_{i,j,k} \mu_{i,j,k} \exp^{-\mu_{i,j,k} x}, \text{ where } \mu_{i,j,k} > 0, a_{i,j,k} \geq 0,$$

$$\text{and } \sum_k a_{i,j,k} = 1.$$

Let $1/\mu_{i,j} = \sum_k a_{i,j,k} / \mu_{i,j,k}$ be the average packet length, in seconds. Let $S_{i,j} = s_{i,j} / \mu_{i,j}$. Then

$$\frac{S_{i,j}}{G_{i,j}} = P(N_i \cup N_j) = \text{a function of } (G_1, G_2, \dots)$$

where

$$G_i = \sum_{j \in N_i^*} G_{i,j}.$$

Proof:

$$\text{let } s_{i,j,k} = a_{i,j,k} s_{i,j}.$$

$$\text{Then } s_{i,j,k} = a_{i,j,k} s_{i,j} = P(N_i \cup N_j) g_{i,j,k}$$

$$\text{where } G_i = \sum_{j \in N_i^*} G_{i,j} = \sum_{j \in N_i^*} \sum_k g_{i,j,k} / \mu_{i,j,k}$$

Dividing $s_{i,j,k}$ by $\mu_{i,j,k}$ and summing over k , we get

$$S_{i,j} = P(N_i \cup N_j) G_{i,j}$$

VI. EXAMPLES OF THE PROCEDURE

As an example consider the chain of four nodes shown in Figure 2. We assume $S_{12} = S_{21} = S_{23} = S_{32} = S_{34} = S_{43} = S$ for simplicity, and perfect capture. Also note that $G_{12} = G_1$, $G_{43} = G_4$ and by symmetry $G_1 = G_4$, $G_{21} = G_{34}$, and $G_{23} = G_{32}$. Also from equation (6), $G_2 = G_{21} + G_{23} = G_3$. From equation (5), we have

$$\frac{S}{G_1} = P(1,2,3) = \frac{S}{G_{21}}, \frac{S}{G_{23}} = P(1,2,3,4)$$

But

$$\begin{aligned} SP(N) &= \sum_{i \in D} (\Pi G_i) = 1 + G_1 + G_2 + G_3 + G_4 + G_1 G_3 + G_2 G_4 + G_1 G_4 \\ &= 1 + 2G_1 + 2G_2 + 2G_1 G_2 + G_1^2 = \Delta \end{aligned}$$

and

$$P(1,2,3,4) = 1/\Delta, P(1,2,3) = (1+G_4)/\Delta = (1+G_1)/\Delta$$

Solving, we get

$$G_2 = G_1(2+G_1) \text{ and } S = G_1(1+G_1)/\Delta$$

or

$$S = G_1(1+G_1)/(1+6G_1+7G_1^2+2G_1^3).$$

We can now find the maximum value of S possible, the throughput of the chain, which is .128, obtained when $G_1 = 0.71$.

In general the equations cannot be solved as simply as for this four node example. Equation (11) is used to get expressions for $P(A)$ in terms of G_i . Then equation (14) is used to get expressions for S_{i,n_i} in terms of G_{i,n_i} and $P(A)$. Equation (6) provides the relation between G_i and G_{i,n_i} . The S_{i,n_i} are found from the offered traffic, the routing, and other assumptions and are considered as inputs. The equations are iterated until a solution of G 's for a set of S 's is found. The maximum set of S 's possible is considered the throughput, or capacity, of the network. For some modest size problems, as above, the equations can be solved directly.

As a second example consider the star topology shown in Figure 5. Here assume there are L legs of $N = 2$ nodes each. Denote the center node by 0, the nodes one hop out by 1, and the other nodes by 2. Further assume symmetrical traffic in the nodes and $S_{01} = S_{10} = S_{12} = S_{21} = S$. Then LS is the total traffic successfully transmitted by node 0. The equations are

$$\begin{aligned} \frac{S_{01}}{G_{01}} &= \frac{LS}{G_0} = \frac{S_{10}}{G_{10}} = \frac{S}{G_{10}} = P(\text{all but } L-1 \text{ \# 2 nodes}) = \frac{SP(L-1 \text{ \# 2 nodes})}{SP(N)} \\ &= (1+G_2)^{L-1}/\Delta \end{aligned}$$

$$\frac{S_{12}}{G_{12}} = \frac{S}{G_{12}} = \frac{S_{21}}{G_{21}} = \frac{S}{G_2} = P(0,1,2) = \frac{SP(L-1 \text{ legs})}{SP(N)} = (1+G_1+G_2)^{L-1}/\Delta$$

where

$$\Delta = SP(N) = (1+G_1+G_2)^L + G_0(1+G_2)^L$$

$$G_1 = G_{10} + G_{12}$$

But $G_{10} = G_0/L$ and $G_{12} = G_2$, so $G_1 = G_2 + G_0/L$. Thus we have two equations:

$$LS/G_0 = (1+G_2)^{L-1}/\Delta \text{ and } S/G_2 = (1+2G_2+G_0/L)^{L-1}/\Delta$$

For any $S < S_{\max}$, they can be solved for G_0 and G_2 . Alternatively for any G_0 we can find the corresponding G_2 by solving

$$G_0(1+G_2)^{L-1} = G_2(1+2G_2+G_0/L)^{L-1}.$$

Then the relation between S and G_2 can be studied. We then find the maximum S , $S_{\max}$, possible. $LS_{\max}$ is the maximum throughput of the star.

For larger problems we will get several equations of the form

$$G_0 = LS\Delta/(1+G_2)^{L-1}$$

$$G_2 = S\Delta/(1+2G_2+G_0/L)^{L-1}$$

For any S we solve these iteratively. Since $G_0 \geq LS$, $G_1 \geq 2S$, and $G_2 \geq S$, the lower bounds are good starting points for G_i . For S sufficiently less than $S_{\max}$ we have found the iteration converges monotonically and rapidly. As S approaches $S_{\max}$ from below the convergence is still monotonic but slows appreciably. For $S > S_{\max}$ the iteration does not converge and often diverges dramatically. We have uncovered no serious numerical problems with this procedure in the many examples we have evaluated.

VII. NUMERICAL EXAMPLES

We consider here three different topological structures with exponentially distributed packet lengths and perfect capture. We assume all $S_{i,n_i} = S$, for all i , and take full advantage of symmetry. The three topologies, shown in Figures 3, 4, and 5, are a chain, a ring, and a star, all with various lengths. In each case we find the maximum throughput, S . These are given in Table I.

The maximum one way throughput for a long chain is $S = .086$. This throughput is approached when the lengths of chains exceed 10. For smaller length chains the throughput is

higher. In CSMA transmissions of neighbors may not overlap in time. Since each node transmits successfully $2S$ packets per average packet transmission time, then we must have $S \leq 1/5$. The throughput is slightly ~~smaller~~ than half of this limit. The cause of the reduction is collisions from transmissions two hops away, the so-called "hidden terminals". The throughput of $S = .086$ for a chain, although the maximum possible, is not a useful operating point. As in ALOHA, this is the point at which delays become infinite and the system is unstable. The network would have to be operated at some lower level.

It is instructive to compare the performance of multihop CSMA with that of slotted ALOHA. Let p be the probability of transmission in one direction at a node. Then $S = p(1-2p)^2$ for a long chain. $S_{\max}$ here is .074 which is approximately 14% less than that for CSMA. There are two factors working here. CSMA will produce less collisions since neighbors will not interfere with each other. Hidden terminals will still produce collisions. (All terminals are hidden in ALOHA). But CSMA prohibits possible successful transmissions. For instance, node 3 can transmit successfully to node 2 while node 4 is transmitting successfully to node 5. This is possible in ALOHA but prohibited in CSMA. This is one of the prices paid to control collisions.

We note that for a ring greater than 7 nodes the maximum throughput is the same as that for a long chain. This is expected since the congestion is now in the middle and it is unimportant whether or not the chain is closed. A star with two legs is just a chain with N for the star replaced by $2N+1$ for the chain.

Consider the star configuration as representing the center node (0) trying to transmit to some node or nodes far away via many repeaters. For one leg, the maximum rate is .086. For two legs, the maximum rate is $2S$ or .172, exactly twice. The results are shown in Table II where the throughput of the center node is given by LS . We see from Table II that whereas the throughput doubles for $L = 2$, it increases only by 20% when $L=3$, by 4-5% further when $L=4$, and by 2% when $L=5$. Congestion at the central node is limiting its ability to increase its

throughput. Additional legs, beyond three, are not really helpful.

We have investigated ways of reducing the congestion at the center node. For larger L , the traffic in each leg is limited by congestion at the center. The collisions that cause most problems are for transmissions from the first level of nodes $(1, N+1, 2N+1, \dots, (L-1)N+1)$ to the center. These are collided with by other first level nodes. To reduce these collisions we considered connecting the first level nodes in a ring and then fully connecting them. These results are also summarized in Table I. When the first level is unconnected the throughput saturates at .229. When the first level is fully connected the throughput with 9 legs is .252, a 15% increase. For four legs, the ring connected topology is best, providing some compromise between reducing collisions and allowing simultaneous transmissions.

The best that we can expect in the fully connected case is $LS \leq 1/3$. This is because all transmissions from the center node and all first level nodes cannot overlap. We will discuss asymptotic results with even larger stars and chains in the next section.

VIII. ASYMPTOTIC RESULTS

We are interested in asymptotic results for several reasons. They provide us with the limiting behavior of the finite networks previously studied. Since the behavior of these networks seems to converge rapidly with their size, if asymptotic results are easier to obtain, they would be useful. We are also interested in very large networks. A final reason is to verify some of the bounding arguments on throughput made in the last section.

We first consider an infinitely long chain. We let $S_{i,i+1} = S_{i,i-1} = S$. Then all nodes are identical. Also $G_{i,i+1} = G_{i,i-1} = G_i/2$. Thus with $G_i = G$,

$$\frac{2S}{G_i} = \frac{2S}{G} = P(i-1, i, i+1, i+2) = \frac{SP(-\infty, \dots, i-2)SP(i+3, \dots, \infty)}{SP(-\infty, \dots, \infty)} \quad (29)$$

We can write the denominator as

$$SP(-\infty, \dots, \infty) = SP(-\infty, \dots, i-1)SP(i+1, \dots, \infty) + G, SP(-\infty, \dots, i-2)SP(i+2, \dots, \infty).$$

Now let $Q_k = \frac{SP(i-k, \dots, \infty)}{SP(i, \dots, \infty)} = \frac{SP(-\infty, \dots, i+k)}{SP(-\infty, \dots, i)}$. We observe that $Q_k \geq 1$ for $k \geq 0$ and if it converges is independent of i . Then equation (29) becomes

$$\frac{2S}{G} = \frac{1}{Q_1 Q_2 + G Q_1} \quad (30)$$

But

$$Q_k = \frac{SP(-\infty, \dots, i+k-1) + G SP(-\infty, \dots, i+k-2)}{SP(-\infty, \dots, i)} = Q_{k-1} + G Q_{k-2} \quad (31)$$

and $Q_0 = 1$. Thus $Q_2 = Q_1 + G$. Therefore

$$2S/G = \frac{1}{Q_1(Q_1 + 2G)}$$

or

$$S = \frac{G}{2Q_1(Q_1 + 2G)} \quad (32)$$

We note that since $Q_{-1} = \frac{1}{Q_1}$, from equation (31) we have

$$Q_1 = Q_0 + \frac{G}{Q_1} = 1 + \frac{G}{Q_1} \quad \text{or} \\ G = Q_1(Q_1 - 1) \quad (33)$$

Finally we have

$$S = \frac{Q_1 - 1}{2Q_1(Q_1 - 1)} \cdot Q_1 \geq 1 \quad (34)$$

The maximum value of S is .086, reached when $Q_1 = 1.7$ or $G = 1.2$.

IX. EXTENSIONS AND CONCLUSIONS

We have presented a simple but fairly realistic model of a multihop packet radio network and have obtained maximum throughputs for general topologies and packet lengths. We have assumed perfect reception of acknowledgments and have not included additional traffic due to end-to-end acknowledgments. Some aspects of acknowledgments can be included by increasing the required traffic. We are investigating the effect of imperfect acknowledgments and different retransmission strategies. The model should still be useful under these extensions.

REFERENCES

- [1] L. Kleinrock and F. Tobagi, "Packet switching in radio channels," parts I and II, IEEE Trans. Commun., vol. COM-23, pp. 1400-1433, December 1975.
- [2] F. Tobagi, "Analysis of a Two-Hop Centralized Packet Radio Network", Part II: CSMA IEEE Trans. Commun., COM-28, pp. 208-216, February 1980.
- [3] R. E. Kahn, "The organization of computer resources into a packet radio network," IEEE Trans. Communications, Vol. COM-25, pp. 169-178, January 1977.
- [4] I. Gitman, R. Van Slyke, and H. Frank, "Routing in packet-switching broadcast radio networks," IEEE Trans. Communications, vol. COM-24, pp. 926-930, September 1976.
- [5] R. Boorstyn and A. Kershenbaum, "Throughput Analysis of Multi-hop Packet Radio Networks," IEEE-ICC'80, pp. 13.6.1-13.6.6, Seattle, WA., June 1980.
- [6] V. Sahin, "Analysis of Multihop Packet Radio Networks", Ph.D Thesis, Polytechnic Institute of New York, June 1982.

Table 1

Maximum One-Way Throughput (S)

Number of Nodes, N	Star Number of Legs, L					
	Chain	Ring	L=2	L=3	L=4	L=5
1	1.000	1.000	.167	.103	.074	.058
2	.500	.500	.111	.076	.057	
3	.167	.167	.097	.072	.055	
4	.128	.073	.092	.070	.054	
5	.111	.100		.069	.054	.044
6	.102	.083				
7	.097	.087				
8	.094	.085				
9	.092	.086				
10	.091	.086				
∞	.086	.086	.086	.069	.054	.044

Table II

Maximum One-Way Total Throughput of the
Central Node in a Large Star Arrangement

Maximum Throughput, LS			
Center Arrangement			
<i>Number of Legs</i>	<i>Unconnected</i>	<i>Ring-Connected</i>	<i>Fully Connected</i>
1	.086		
2	.172		
3	.207	.198	.198
4	.216	.228	.216
5	.220	.230	.230
6	.220		.240
7			.245
8			.248
9			.252

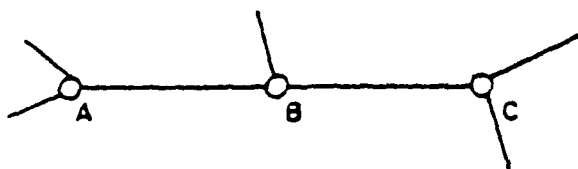


Figure 1. A part of a network

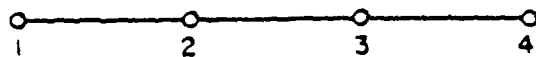


Figure 2. A four node chain

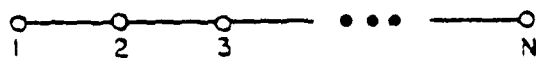


Figure 3. A chain

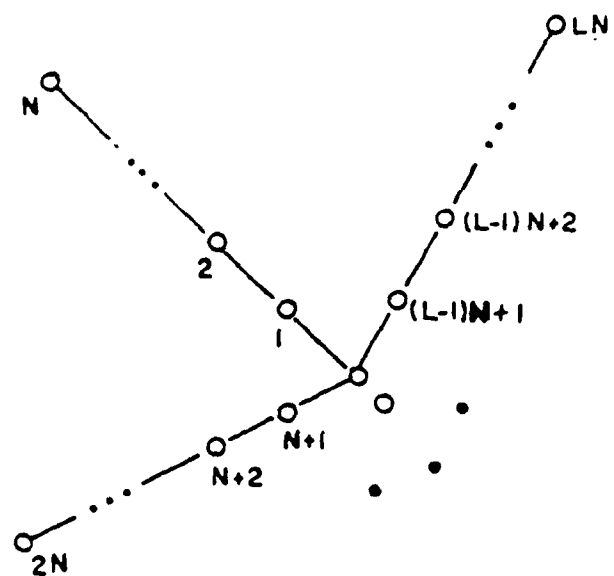


Figure 5. A star network (L legs)

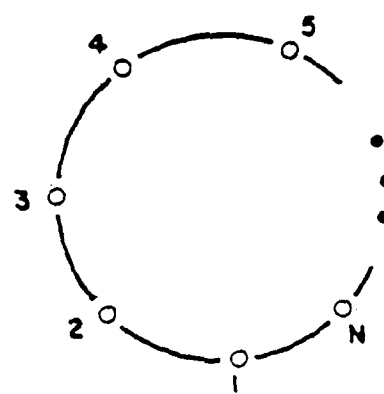


Figure 4. A ring

A.2 Throughput Analysis of Multihop Packet Radio

Boorstyn and Kershenbaum

IEEE International Conference on Communications,

June 1980, Seattle

THROUGHPUT ANALYSIS OF MULTIHOP PACKET RADIO

Robert R. Boorstyn and Aaron Kershenbaum

Polytechnic Institute of New York
333 Jay Street, Brooklyn, NY 11201

ABSTRACT

We consider the problem of obtaining exact expressions for throughput and blocking probabilities in multihop packet radio networks operating under CSMA. We obtain exact results for exponential message lengths for general topologies and for constant packet lengths for simple topologies. Both results are obtained by assuming perfect acknowledgments.

I. INTRODUCTION

We consider the problem of obtaining exact expressions for throughput and blocking probabilities in multihop packet radio networks operating under carrier sense multiple access (CSMA). By considering exponential packet lengths, procedures are developed which can be used to analyze general topologies. Examples of chains, rings, and stars are presented. Extensions to non-exponential and fixed packet lengths, and more complex models are discussed.

II. THE NETWORK MODEL

We consider the problem of analyzing the throughput capability of a multihop packet radio network operating under carrier sense multiple access (CSMA). Thus, we assume that the network is comprised of terminals equipped with radio transponders suitable for broadcasting data over a limited distance. In general, the source and destination terminals cannot hear each other directly, and the data has to be relayed by one or more intermediate devices. A separate set of devices, called repeaters, may exist for this purpose, or the terminals themselves may relay messages for one another.

Control of the network is completely distributed, i.e., no station or central control mechanism is assumed to exist. Rather, we assume that each source terminal has prestored one or more routes to all destinations and includes all necessary routing information in the packets it transmits. These assumptions are made merely to simplify the presentation. In fact, the results presented are valid for networks using alternate routing as long as routing changes are not made over short time intervals. One of the motivations for this study came from a consideration of the

design of routing procedures for such networks. It was necessary, however, to first develop an understanding for the throughput of various topologies.

Exogenous traffic is modeled as independent Poisson processes arriving at each source node, with appropriate rates and, initially, exponentially distributed packet lengths. The topology is specified by a listing of which terminals (or repeaters) can hear each other. In the remainder we will not distinguish between terminals and repeaters and will refer to them collectively as either terminals or nodes. In general the transmissions of one terminal can be heard by many other terminals. The routing will specify which terminal is to repeat the packet, if necessary.

If two or more transmissions are simultaneously heard by a terminal (called a "collision") at least one, and possibly both, is "lost" and must be retransmitted. We assume retransmissions are scheduled at a random instant in time sufficiently far in the future so as to preserve the Poisson nature of the combined traffic stream, which now consists of exogenous traffic and rescheduled traffic. For this study, we assume that a packet can be retransmitted as many times as is necessary, i.e., that there is no maximum allowable number of retransmissions.

Terminals may either transmit or receive, they cannot do both simultaneously. Before transmitting, a terminal senses the channel. If it detects that any of its neighbors (i.e., terminals that it can hear) are transmitting (by, e.g., sensing a carrier) it reschedules the transmission as for collided packets above. If at the scheduled time for a transmission, the terminal is already engaged in transmitting a packet, the new packet is also rescheduled as above. Thus packets are continually rescheduled until they are successfully delivered to the next terminal on their route. We assume that the total stream of traffic scheduled by any terminal is a Poisson process. This includes originating traffic and packets rescheduled either due to collisions or due to the channel having been sensed busy. This scheme is called carrier sense multiple access (CSMA). The Poisson assumption is valid for the assumptions made above and will yield accurate results for throughput. Compromises will have to be made, however, if an accurate picture of time delays is to be considered.

It is possible, due to non-zero propagation delay, that collisions of transmissions from neighboring terminals may still take place despite the CSMA strategy. This will occur if a terminal senses the channel before another terminal's transmission is received. This effect is small if terminals are reasonably close or are not transmitting at high speed. We will ignore this phenomena here, and assume that all transmissions are instantly heard by their neighbors.

A passive acknowledgment is used for transmission to neighbors. The transmitting terminal listens to the channel to hear if a packet is being rebroadcast by a neighbor. If after a prespecified time interval, the transmitting node does not hear the packet rebroadcast, it retransmits the packet. But the packet may have been successfully received by the neighbor even though the originator does not hear the rebroadcast. We assume that duplicate packets are transmitted and are deleted only at the final destination. We assume an end-to-end acknowledgment is returned to the originator from the final destination. In this paper we assume that passive acknowledgments are always heard and ignore the effect of end-to-end acknowledgments. Alternately, these acknowledgements could have been added to the required traffic.

We depict the topology of the network by a graph where terminals are represented by nodes. The nodes are connected by a link if they can hear each other's transmissions, i.e., if they are neighbors. As an example see figure 1. Node A can hear node B, but not node C. Node B can hear both nodes A and C. Node C can hear node B, but not node A. If node A is transmitting to node B and node C begins transmitting, then the transmission from A to B may be lost depending upon the "capture" assumptions we make. A conservative assumption is that the A to B transmission is lost - this is known as zero capture. Alternately, perfect capture assumes that this transmission is successfully received. Half-amplitude capture assumes that the transmission is lost if C dominates A at B. This can happen if C is closer to B than A is to B, or has a greater signal strength perceived at B than A has. If A dominates C, then the transmission is successful. We will consider these different capture situations below. Note that under CSMA if node B is transmitting, neither A or C is allowed to transmit.

We assume that a routing has been specified. This takes the form of deciding which of the neighbors are to rebroadcast a packet from a particular source to a particular destination. Thus the amount of traffic that a terminal wishes to send to its neighbor can be computed. If these rebroadcast packets are scheduled at a random time far in the future the Poisson assumption for traffic streams is preserved. We assume that the traffic between neighbors is specified and form independent Poisson processes. When considering exponentially distributed packet lengths we assume that the packet length is reassigned independently at each hop. Although this leads to some anomalies in interpret-

ing the results it is a key assumption in the model and is analogous to the "independence assumption" in queuing networks.

The details of CSMA for a single hop network can be found in the papers by Tobagi and Kleinrock⁽¹⁾. Tobagi has also developed some simple models for two-hop networks⁽²⁾. Details of a packet radio network can be found in a paper by Kahn⁽³⁾. A discussion of routing in multihop packet radio can be found in the paper by Gitman, Van Slyke, and Frank⁽⁴⁾.

III. GENERAL RESULTS

In this section we develop some expressions that are valid for the packet radio network we have modelled above using CSMA and with an arbitrary packet length distribution. Let i be a node, n_i one of its neighbors, N_i^* the set of all the neighbors of i , and N_i the set of all i 's neighbors and including i . Without loss of generality assume the average packet length at each terminal is unity. Let G_i be the total rate (in packets/sec) of all scheduled traffic at node i . This includes originating traffic and all rescheduled traffic and is assumed to be Poisson.

Node i is either busy (transmitting) or idle. It will transmit a scheduled packet if at the instant it is scheduled all nodes in N_i are idle.

Let A be a set of nodes. Let $P(A)$ be the probability that at a random instant all nodes in A are idle. The nodes not in A may or may not be idle. Similarly $P(i)$, $P(i,A)$, $P(A,B)$ are the probabilities that i is idle, i and nodes in A are idle, and all nodes in A and B are idle.

Since traffic is scheduled at node i with a Poisson rate G_i , will be transmitted only if N_i is idle, and transmissions have unity average length, the probability that i is busy is given by

$$1 - P(i) = G_i P(N_i) \quad (1)$$

If i is busy then under CSMA, n_i must be idle. Then since

$$P(n_i) = P(n_i, i) + P(n_i | i \text{ busy}) [1 - P(i)]$$

and $P(n_i | i \text{ busy}) = 1$, we have

$$P(n_i, i) = P(n_i) + P(i) - 1 \quad (2)$$

Similarly, if $A \subset N_i^*$,

$$P(A, i) = P(A) + P(i) - 1 \quad (3)$$

letting $A = N_i^*$ in equation (3), and using

$$P(N_i^*) = P(N_i) / P(i|N_i^*)$$

we get

$$P(i|N_i^*) = \frac{1}{1+G_i} \quad (4)$$

Equation (4) is often found in CSMA literature.

A packet from i to n_i will be transmitted when it is scheduled if N_i is idle. During the transmission all nodes in N_i^* will be idle. It will be successfully received at n_i if (1) all neighbors of n_i not in N_i are idle at the beginning of transmission and (2) depending upon the capture assumptions, none of these nodes begin transmitting until i 's transmission is ended. Otherwise a collision will occur. Let S_{i,n_i} be the rate in packets/sec, determined by the routing and assumed Poisson, of the traffic that i wishes to send to n_i . This is the required throughput or offered traffic. Let G_{i,n_i} be the rate of all scheduled traffic from i to n_i . We have also assumed that all these streams are Poisson and independent. Of these G_{i,n_i} packets (per second), S_{i,n_i} must be successful. Thus

$$\frac{S_{i,n_i}}{G_{i,n_i}} = P(N_i, N_{n_i}) P(X) \quad (5)$$

Let D_{i,n_i} be the nodes in N_{n_i} that dominate i at n_i . Then $P(X)$ is the probability that no nodes in D_{i,n_i} begin transmission during the transmission of the packet from i to n_i conditioned on node i transmitting, all nodes in N_i^* being idle throughout the transmission, and all nodes in N_{n_i} being idle, at least, initially. If D_{i,n_i} is empty or perfect capture is assumed then $P(X)=1$.

The total scheduled traffic at a node is given by

$$G_i = \sum_{n_i \in N_i^*} G_{i,n_i} \quad (6)$$

From equations (1) through (6) we wish to derive a relation between the S_{i,n_i} and G_i , and determine the maximum S_{i,n_i} the network can support. This we call the (maximum) throughput or capacity. In the next section we develop this relationship for exponential packet lengths and arbitrary topologies. Later we present some results for non-exponential lengths and simple topologies.

IV. EXPONENTIALLY DISTRIBUTED PACKET LENGTHS

If the packet lengths are exponentially distributed, then the system can be viewed as a

Markov process where the states are identified by which nodes are idle and which are busy. Let D be a set of busy nodes. Because of CSMA, no nodes in D may be neighbors of each other. We shall refer to such a set of non-neighbors (or strangers) as being "strange." Let $Q(D)$ be the probability that at an instant of time, all nodes in D , which is strange, are busy, and all nodes not in D , are idle. Then each strange set, D , represents a state in a Markov system, and $Q(D)$ is the state probability. The collection, $\mathcal{D}$, of all strange sets, $\mathcal{D}$, represents all the states in the system. In particular, the null set $D = \emptyset$, represents the state that all nodes are idle.

Assume the system is in state D . It will leave the state if any $i \in D$ stops transmitting. This happens with rate $\mu_i = 1$. Thus the transition to state $\{D-i\}$ occurs with rate μ_i . The only other way to leave state D is for one of the idle nodes that is a stranger to all $i \in D$ to begin to transmit. This occurs with rate $\lambda_j = G_j$. (We use μ_i and λ_j for the moment for clarity). Let N_D be the set of all neighbors of all nodes in D . Then the transition from D to $\{D+j\}$, $j \notin N_D$, occurs with rate λ_j . The global balance equations for this system are

$$\sum_{i \in D} (\sum_{j \notin N_D} \lambda_j) Q(D) = \sum_{i \in D} \mu_i Q(D-i) + \sum_{j \notin N_D} \mu_j Q(D+j) \quad (7)$$

where $D \in \mathcal{D}$, the collection of all strange sets.

It is easy to see that these equations are satisfied by

$$Q(D) = \frac{\lambda_i}{\mu_i} Q(D-i) = G_i Q(D-i), \quad i \in D \quad (8)$$

Thus

$$Q(D) = \left(\prod_{i \in D} G_i \right) Q(\emptyset) \quad (9)$$

where we adopt the convention that $\prod_{i \in \emptyset} G_i = 1$. Summing over all $D \in \mathcal{D}$, we get

$$\sum_{D \in \mathcal{D}} Q(D) = \sum_{D \in \mathcal{D}} \left(\prod_{i \in D} G_i \right) Q(\emptyset) = 1 \quad (10)$$

In the previous section we found we were interested in quantities like $P(A)$, where A is any set of nodes, and $P(A)$ is the probability that all nodes in A are idle, and all nodes not in A may or may not be idle. This can be found by summing $Q(D)$ over all sets D that do not contain nodes in A . Thus

$$P(A) = \sum_{D \subset A^c} Q(D) = \frac{\sum_{D \subset A^c} \left(\prod_{i \in D} G_i \right)}{\sum_{D \subset \mathcal{D}} \left(\prod_{i \in D} G_i \right)} \quad (11)$$

where $D \setminus A^c$ refers to all strange sets contained in the complement of A . We adopt the shorthand notation,

$$SP(B) = \sum_{D \setminus B \in D} (N_{G_1}) \quad (12)$$

where SP refers to sum of products. Thus

$$P(A) = SP(A^c)/SP(\pi) \quad (13)$$

where π is the set of all nodes.

As an example consider the chain of four nodes shown in figure 2. We assume $S_{12} = S_{21} = S_{23} = S_{32} = S_{34} = S_{43} = S$ for simplicity, and perfect capture. Also note that $G_{12} = G_1$, $G_{43} = G_4$ and by symmetry $G_1 = G_4$, $G_{21} = G_{34}$, and $G_{23} = G_{32}$. Also from equation (6), $G_2 = G_{21} + G_{23} = G_3$. From equation (5), we have

$$\frac{S}{G_1} = P(1,2,3) = \frac{S}{G_{21}} \cdot \frac{S}{G_{23}} P(1,2,3,4)$$

But

$$SP(\pi) = \sum_{D \setminus B \in D} (N_{G_1}) = 1 + G_1 + G_2 + G_3 + G_4 + G_1 G_3 + G_2 G_4 + G_1 G_4$$

$$= 1 + 2G_1 + 2G_2 + 2G_1 G_2 + G_1^2 = \Delta$$

and

$$P(1,2,3,4) = 1/\Delta, \quad P(1,2,3) = (1+G_4)/\Delta = (1+G_1)/\Delta.$$

Solving, we get

$$G_2 = G_1(2+G_1) \quad \text{and} \quad S = G_1(1+G_1)/\Delta.$$

or

$$S = G_1(1 + G_1)/(1 + 6G_1 + 7G_1^2 + 2G_1^3).$$

We can now find the maximum value of S possible, the throughput of the chain, which is .128, obtained when $G_1 = 0.71$.

We can also consider half-amplitude or zero capture. For example, consider zero capture. Then when 2 is transmitting to 3, 4 can interfere. But 1, 2 and 3 are idle, and 4 is initially idle. The transmission is successful if 2 finishes transmission before 4 starts. But these are events occurring in exponential time.

Thus $P(X) = \frac{1}{1+G_1} = \frac{1}{1+G_4}$ by symmetry. Now

$$\frac{S}{G_{23}} = P(1,2,3,4) = \frac{1}{1+G_4} \cdot \frac{S}{G_{21}} \text{ is unchanged.}$$

$\frac{S}{G_1}$ is a little harder to obtain, but is a straight-

forward calculation.

Equations (11) and (5) can be used for any topology to generate the solution to our problem. The equations relating the S_{i,n_i} and G_j can be solved iteratively. For example, equation (5) now becomes

$$\frac{S_{i,n_i}}{G_{i,n_i}} = \frac{SP([N_i + N_n])^c}{SP(\pi)} P(X) \quad (14)$$

where by $A+B$ we mean the union of A and B . For perfect capture $P(X)=1$. For other capture modes, the computations for $P(X)$ are straightforward but can become tedious for complex topologies.

Evaluation of sums of products in equation (12) are made easier by the following two rules. Consider two sets of nodes A and B such that no node in A can hear any node of B . These sets may be called non-communicating. Then

$$SP(A+B) = SP(A) SP(B), \quad \text{A \& B non-communicating} \quad (15)$$

Also,

$$SP(A) = SP(A-i) + G_i SP(A-N_i), \quad i \in A \quad (16)$$

To prove these rules just consider all products. We have successfully evaluated many complex topologies with these procedures.

There are other relations which will be found useful in extending our model to more complex situations. We prove some of these below. Let C be a cut, i.e., a set of nodes that divides the network into three parts A, B , and C , where A and B are non-communicating. Let $A = A_1 + A_2$, $B = B_1 + B_2$ where $A_1 \cap A_2 = B_1 \cap B_2 = \emptyset$. Then

$$P(A_1 | C, B_1) = \frac{P(A_1, C, B_1)}{P(C, B_1)} = \frac{SP(A_2 + B_2)}{SP(A + B_2)} = \frac{SP(A_2)}{SP(A)}$$

But

$$P(A_1 | C) = \frac{P(A_1, C)}{P(C)} = \frac{SP(A_2 + B)}{SP(A + B)} = \frac{SP(A_2)}{SP(A)}$$

$$\text{Thus} \quad P(A_1 | C, B_1) = P(A_1 | C), \quad C \text{ a cut} \quad (17)$$

We also have

$$P(A_1, B_1 | C) = P(A_1 | C) P(B_1 | C), \quad C \text{ a cut} \quad (18)$$

In particular if $C = N_i^*$, then

$$P(i|N_1^*, B) = P(i|N_1^*), \quad B \subset N_1^C \quad (19)$$

V Examples

We consider here three different topological structures with exponentially distributed packet lengths and perfect capture. We assume all S_{1,n_i} S , for all i , and take full advantage of symmetry. The three topologies, shown in figures 3, 4, and 5, are a chain, a ring, and a star, all with various lengths. In each case we find the maximum throughput, S . These are given in Table I.

Note that for a ring greater than 7 nodes the maximum throughput is the same as that for a long chain. This is expected since the congestion is now in the middle and it is unimportant whether or not the chain is closed. A star with two legs is just a chain with N for the star replaced by $2N+1$ for the chain. Consider the star configuration as representing the center node (0) trying to transmit to some node or nodes far away via many repeaters. For one leg, the maximum rate is .086. For two legs, the maximum rate is $2S$ or .172, exactly twice. The results are shown in Table II where the throughput of the center node is given by LS . We see from Table II that whereas the throughput doubles for $L=2$, it increases only by 20% when $L=3$, by 4-5% further when $L=4$, and by 2% when $L=5$. Congestion at the central node is limiting its ability to increase its throughput. Additional legs, beyond three, are not really helpful. We are evaluating other topologies to find ways to increase the throughput of a central node in such a configuration.

VI Non-exponential Packet Lengths

For non-exponential packet lengths, equations (1) through (6) are still valid but are insufficient to solve for the G_i and the S_{1,n_i} . For a three node chain and a star with any number of legs but $N=1$ we have been able to prove equation (19) for arbitrary packet length distributions. The equations for $P(A)$ are identical in form as for exponential lengths but the terms $P(X)$ for half-amplitude and zero capture are different. For example, for constant packet lengths instead of $1/(1+G)$ we now have e^{-G} . These two simple topologies can be completely analyzed for any capture mode. Further details will not be given here.

We have not been able to derive a similar relationship for other topologies. However, we feel that equations (17) through (19) are tempting approximations to try. Then if together with equations (1) through (6) they are sufficient to derive enough equations to find the throughput, the resulting approximate solution has some appeal. We have been able to do this for several simple topologies but have not verified the accuracy of the approximation. Consideration of other than perfect capture in these cases becomes very difficult.

VII Extensions and Conclusions

We have presented a simple but fairly realistic model of a multihop packet radio network and have obtained maximum throughputs for general topologies and exponential message lengths. The procedures are easiest to implement when perfect capture is assumed. Other capture modes can be handled in a straightforward but somewhat more tedious manner. We have assumed perfect reception of acknowledgments and have not included additional traffic due to end-to-end acknowledgments. Some aspects of acknowledgments can be included by increasing the required traffic. We are investigating the effect of imperfect acknowledgments and different retransmission strategies. The model should still be useful under these extensions.

We have been able to derive results for non-exponential packet lengths, but only for simple topologies. Our procedure suggests a simple, and probably good, approximation to be used for more complex topologies. However, imperfect capture becomes a more serious problem. We are investigating these extensions.

Finally, we are applying these techniques to evaluate different routing strategies and transmission schemes in a multihop packet radio network.

VIII Acknowledgments

The work described in this paper was started under a subcontract with Network Analysis Corporation, under a DARPA contract DAAK 80-79-C-0763, monitored and directed by USARMY CORADCOM. It has continued under a contract with the Post Doctoral Program, RADC, Task B-9-3513, funded by USARMY CORADCOM. The computations in Tables I and II were performed by Mr. Veli Sahin.

References

- 1) L. Kleinrock and F. Tobagi, "Packet switching in radio channels," parts I and II, IEEE Trans. Commun., vol. COM-23, pp. 1400-1433, December 1975.
- 2) F. Tobagi, "On the performance analysis of multihop packet radio systems: Parts I-IV," Packet Radio Temporary Notes #246-249, Comput. Sci. Dept., Univ. California, Los Angeles 1978.
- 3) R. E. Kahn, "The organization of computer resources into a packet radio network," IEEE Trans. Communications, Vol. COM-25, pp. 169-178, January 1977.
- 4) I. Gitman, R. Van Slyke, and H. Frank, "Routing in packet-switching broadcast radio networks," IEEE Trans. Communications, vol. COM-24, pp. 926-930, September 1976.

Table I. Maximum throughput (S)

Number of Nodes, N	Chain	Ring	Star			
			L=2	L=3	L=4	L=5
1	1.000	1.000	.167	.103	.074	.058
2	.500	.500	.111	.076	.057	
3	.167	.167	.097	.072	.055	
4	.128	.073	.092	.070	.054	
5	.111	.100		.069	.054	.044
6	.102	.083				
7	.097	.087				
8	.094	.085				
9	.092	.086				
10	.091	.086				
∞	.086	.086	.086	.069	.054	.044

Table II. Maximum Throughput of the Central Node in a Large Star Arrangement.

Number of Legs	Maximum Throughput
1	.086
2	.172
3	.207
4	.216
5	.220

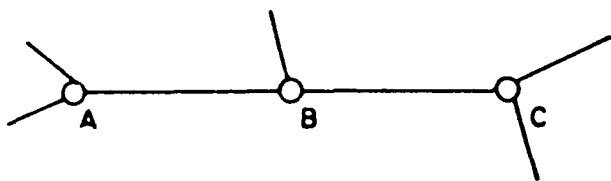


Figure 1. A part of a network

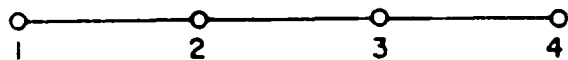


Figure 2. A four node chain

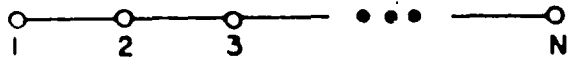


Figure 3. A chain

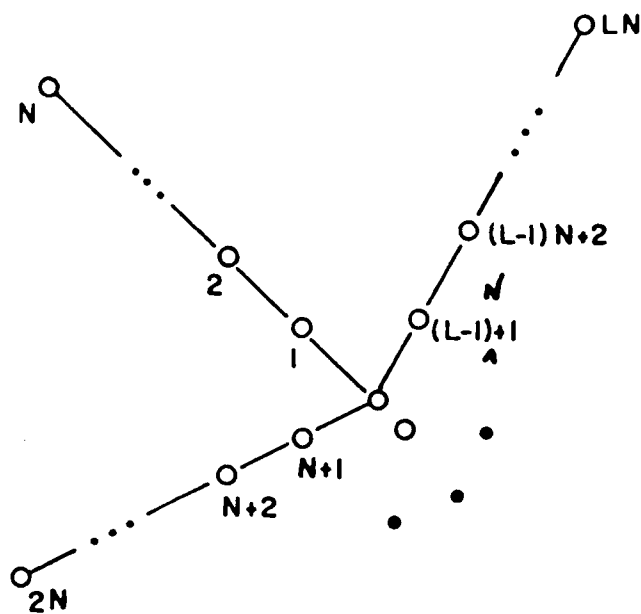


Figure 5. A star network (L legs)

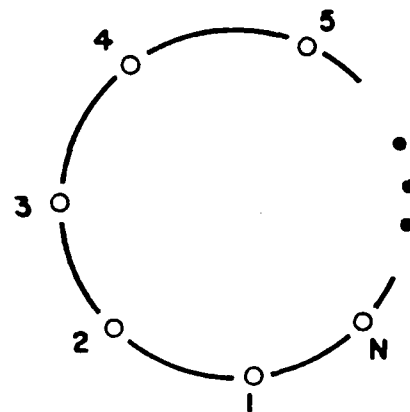


Figure 4. A ring

A.3 A New Acknowledgment Protocol for Analysis of Multi-
hop Packet Radio Networks

Boorstyn, Kershenbaum, and Sahin

IEEE COMPCON, September 1982, Washington

A NEW ACKNOWLEDGMENT PROTOCOL FOR ANALYSIS OF MULTIHOP PACKET RADIO NETWORKS*

Robert R. Boorstyn and Aaron Kershenbaum
Polytechnic Institute of New York, 333 Jay Street, Brooklyn, New York 11201

Veli Sahin
Bell Laboratories, Holmdel, New Jersey 07733

Abstract

We consider the problem of obtaining exact expressions for throughput and blocking probabilities in multihop packet radio networks operating under CSMA. We have obtained exact results for a general class of message lengths, for general topologies, and for perfect capture. These results were obtained by assuming perfect acknowledgments. Here, we extend these results to include situations where acknowledgments are not always heard. We also introduce and analyze a new acknowledgment protocol that significantly improves performance.

I. Introduction

We consider the problem of obtaining exact expressions for throughput and blocking probabilities in multihop packet radio networks operating under carrier sense multiple access (CSMA). Procedures have been developed to analyze general topologies for a general class of packet length distribution and perfect capture.⁴ That work assumed perfect acknowledgements. Here, we extend these results to include situations where acknowledgments are not always heard. We also introduce and analyze a new acknowledgment protocol that significantly improves performance.

II. The Network Model

We consider the problem of analyzing the throughput capability of a multihop packet radio network operating under carrier sense multiple access (CSMA). Thus, we assume that the network is comprised of terminals equipped with radio transponders suitable for broadcasting data over a limited distance. In general, the source and destination terminals cannot hear each other directly, and thus data has to be relayed by one or more intermediate devices. A separate set of

devices, called repeaters, may exist for this purpose, or the terminals themselves may relay messages for one another.

Control of the network is completely distributed, i.e., no station or central control mechanism is assumed to exist. Rather, we assume that each source terminal has prestored one or more routes to all destinations and includes all necessary routing information in the packets it transmits. These assumptions are made merely to simplify the presentation. In fact, the results presented are valid for networks using alternate routing as long as routing changes are not made over short time intervals.

Exogenous traffic is modeled as independent Poisson processes arriving at each source node, with appropriate rates and packet lengths. The topology is specified by a listing of which terminals (or repeaters) can hear each other. In the remainder we will not distinguish between terminals and repeaters and will refer to them collectively as either terminals or nodes. In general the transmissions of one terminal can be heard by many other terminals. The routing will specify which terminal is to repeat the packet, if necessary.

If two or more transmissions are simultaneously heard by a terminal (called a "collision") at least one, and possibly both, is "lost" and must be retransmitted. We assume retransmissions are scheduled at a random instant in time sufficiently far in the future so as to preserve the Poisson nature of the combined scheduled traffic stream, which now consists of exogenous traffic and rescheduled traffic. For this study, we assume that a packet can be retransmitted as many times as is necessary, i.e., that there is no maximum allowable number of retransmissions.

At any time, terminals may either transmit or receive, they cannot do both simultaneously. Before transmitting, a terminal senses the channel. If it detects that any of its neighbors (i.e., terminals that it can hear) are transmitting (by, e.g., sensing the carrier) it reschedules the transmission as for collided packets above. If at the scheduled time for a transmission, the terminal is already engaged in transmitting a packet, the new packet is also rescheduled as above. Thus packets are continually rescheduled until they are successfully delivered to the next terminal on their route. We assume that the total stream of traffic scheduled by any terminal is a

* This research was partially supported by USARMY CENCOMS under contract DAAK 80-80-K-0579 and by the National Science Foundation under grant ENG-79-08120

Poisson process. This includes originating traffic and packets rescheduled either due to collisions or due to the channel having been sensed busy. This scheme is called carrier sense multiple access (CSMA). This Poisson assumption is valid for the assumptions made above and will yield accurate results for throughput.

It is possible, due to non-zero propagation delay, that collisions of transmissions from neighboring terminals may still take place despite the CSMA strategy. This will occur if a terminal senses the channel before another terminal's transmission is received. This effect is small if terminals are reasonably close or are not transmitting at high speed. We will ignore this phenomena here, and assume that all transmissions are instantly heard by their neighbors.

A passive acknowledgement is used for transmission to neighbors. The transmitting terminal listens to the channel to hear if a packet is being rebroadcast by a neighbor. If after a prespecified time interval, the transmitting node does not hear the packet rebroadcast, it retransmits the packet. But the packet may have been successfully received by the neighbor even though the originator does not hear the rebroadcast. Duplicate packets may be transmitted and deleted only at the final destination or they may be detected and deleted earlier. An end-to-end acknowledgment is returned to the originator from the final destination. In our original work we assumed that passive acknowledgments were always heard and ignored the effect of end-to-end acknowledgments. The end-to-end acknowledgments could easily be added to the required traffic. In this paper we consider the situation when passive acknowledgments may not always be heard. We also introduce and analyze a new acknowledgment protocol.

We depict the topology of the network by a graph where terminals are represented by nodes. The nodes are connected by a link if they can hear each other's transmissions, i.e., if they are neighbors. As an example see Figure 1. Node A

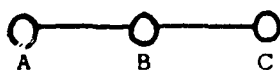


Figure 1
Network Topology

can hear node B, but not node C. Node B can hear both nodes A and C. Node C can hear node B, but not node A. If node A is transmitting to node B and node C begins transmitting, then the transmission from A to B may be lost depending upon the "capture" assumptions we make. A conservative assumption is that the A to B transmission is lost -- this is known as zero capture. Alternatively, perfect capture assumes that this transmission is successfully received. Half-amplitude capture assumes that the transmission is lost if C dominates A at B. This can happen if C is closer to B than A is to B, or has a greater signal strength perceived at B than A has. If A

dominates C, then the transmission is successful. However, in all cases of a collision we assume the later transmission is lost. Thus if C is transmitting to B, this packet is lost in all cases. We will consider only perfect capture situations below. Note that under CSMA if node B is transmitting neither A or C is allowed to transmit.

We assume that a routing has been specified. This takes the form of deciding which of the neighbors are to rebroadcast a packet from a particular source to a particular destination. Thus the amount of traffic that a terminal wishes to send to its neighbor can be computed. If these rebroadcast packets are scheduled at a random time far in the future the Poisson assumption for scheduled traffic streams is preserved. We assume that the packet length is reassigned independently at each hop. Although we have extended the analysis to include non-exponential packet length distributions, we consider only exponential packet lengths here. The results are similar.

The details of CSMA for a single hop network can be found in the papers by Tobagi and Kleinrock¹. Tobagi has also developed some simple models for two-hop networks². Details of a packet radio network can be found in a paper by Kahn³. Details of our analytic procedure were presented at ICC'80⁴. More details can be found in the thesis of Sahin⁵.

III. Analytic Procedure

In this section we review the analytic procedure for perfect acknowledgments⁴. Let i be a node, n_i one of its neighbors, N_i^* the set of all the neighbors of i , and N_i the set of all i 's neighbors, including i . Let g_i be the total rate (in packets/sec) of all scheduled traffic at node i . This includes originating traffic and all rescheduled traffic and is assumed to be Poisson. Let $1/\mu_i$ be the average length of packets transmitted by node i . Let $G_i = g_i/\mu_i$ be a normalized rate.

Node i is either busy (transmitting) or idle. It will transmit a scheduled packet if at the instant it is scheduled all nodes in N_i are idle. Let A be a set of nodes. Let $P(A)$ be the probability that at a random instant all nodes in A are idle. The nodes not in A may or may not be idle. Similarly $P(i)$, $P(i,A)$, $P(A,B)$ are the probabilities that node i is idle, node i and nodes in A are idle, and all nodes in A and B are idle.

A packet from i to n_i will be transmitted when it is scheduled if N_i is idle. During the transmission, because of CSMA, all nodes in N_i will be idle. It will be successfully received at n_i if all neighbors of n_i not in N_i are also idle at the beginning of the transmission. Otherwise

a collision will occur. Let s_{i,n_i} be the rate in packets/sec, determined by the routing of the traffic that i wishes to send to n_i . This is the required throughput or offered traffic. Let g_{i,n_i} be the rate of all scheduled traffic from i to n_i . We have assumed that all these streams are Poisson and independent. Of these g_{i,n_i} packets per second, s_{i,n_i} must be successful. Thus

$$\frac{s_{i,n_i}}{g_{i,n_i}} = \frac{S_{i,n_i}}{G_{i,n_i}} = P(N_i, N_{n_i}) \quad (1)$$

where $S_{i,n_i} = s_{i,n_i}/\mu_i$ and $G_{i,n_i} = g_{i,n_i}/\mu_i$.

The total scheduled traffic (normalized) at a node is given by

$$G_i = \sum_{n_i \in N_i^*} G_{i,n_i} \quad (2)$$

If the packet lengths are exponentially distributed, then the system can be viewed as a Markov process where the states are identified by which nodes are idle and which are busy. Let D be a set of busy nodes. Because of CSMA, no nodes in D may be neighbors of each other. Let $Q(D)$ be the probability that at an instant of time, all nodes in D are busy, and all nodes not in D are idle. Then each set, D , represents a state in a Markov system, and $Q(D)$ is the state probability. In particular, the null set $D = \phi$, represents the state that all nodes are idle.

We have shown⁴ that

$$Q(D) = \frac{\prod_{i \in D} G_i}{\sum_{D \subset N} \prod_{i \in D} G_i} \quad (3)$$

where N is the set of all nodes and we consider $\prod_{i \in \phi} G_i = 1$.

We are interested in quantities like $P(A)$, where A is any set of nodes, and $P(A)$ is the probability that all nodes in A are idle, and all nodes not in A may or may not be idle. This can be found by summing $Q(D)$ over all sets D that do not contain nodes in A . Thus

$$P(A) = \sum_{D \subset A^c} Q(D) = \frac{\sum_{D \subset A^c} \prod_{i \in D} G_i}{\sum_{D \subset N} \prod_{i \in D} G_i} \quad (4)$$

where $D \subset A^c$ refers to all such sets contained in the complement of A . We adopt the shorthand notation

$$SP(B) = \sum_{D \subset B} \left(\prod_{i \in D} G_i \right) \quad (5)$$

where SP refers to sum of products. Thus

$$P(A) = SP(A^c) / SP(N) \quad (6)$$

The above equations can be used for any topology to generate the solution to our problem. The equations relating the S_{i,n_i} , G_{i,n_i} , and G_i can be solved iteratively. For example, equation (1) now becomes

$$\frac{S_{i,n_i}}{G_{i,n_i}} = \frac{SP(N_i + N_{n_i})^C}{SP(N)} \quad (7)$$

We have also given many rules to make the writing of these equations easier⁴.

IV. Passive Acknowledgments

In this section we extend our previous work on throughput in multihop packet radio networks to include the effects of passive acknowledgments not always being heard. Let node j be a neighbor of node i . We have already seen that

$$S_{i,j} = G_{i,j} P(N_i | N_j) P(N_i) = G_{i,j} P(N_i, N_j) \quad (8)$$

When j transmits the packet to the next node in the route, i can hear the transmission, and thus receive a passive acknowledgment, if all of its neighbors (except j) are idle. If it does not hear the transmission, then after a suitable time-out period it retransmits the packet. Previously we assumed all passive acknowledgments were heard. Here we see that a passive acknowledgment is heard with probability $P(N_i | N_j)$.

We assume that the time-out period is selected so that all nodes have the opportunity of hearing exactly one transmission. If they fail to hear it, the packet is retransmitted. This is repeated until an acknowledgment is heard. The assumption that exactly one transmission may be heard in a time-out period is equivalent to assuming that the same time out period is used for all nodes and that it is long enough to insure one transmission.

The rate $S_{i,j}$ in equation (8) includes repeated packets, because of failure to hear acknowledgments. Let $S_{i,j}^*$ be the rate of packets for which passive acknowledgments are heard. Then

$$S_{i,j}^* = q_{i,j} S_{i,j} \quad (9)$$

where $q_{i,j} = P(N_i | N_j) = \frac{P(N_i, N_j)}{P(N_j)}$ (10)

We have already shown how to solve sets of equations like (8) for the $G_{i,j}$ as a function of the $S_{i,j}$, and found the maximum possible $S_{i,j}$ which we defined as the throughput⁴. For example, in a chain, all $S_{i,j} = S$ for $j = i$ and $i = 1$.

Here we see how to modify this procedure to account for passive acknowledgments. The $S_{i,j}^*$ are determined from the offered traffic and the routing.

We identify two situations. Packets are retransmitted if they are unsuccessfully transmitted or if the passive acknowledgment is not heard. In the latter case, a duplicate packet is sent. If the receiving node recognizes the duplicate packet and deletes it, then $S_{i,j}^*$ is the desired rate of traffic between i and j . If a simpler node does not have this capability, then duplicate packets are retransmitted, and after several hops many duplicates exist. In this latter case, $S_{i,j}^*$ includes these duplicate packets and must be related to the desired rate of traffic. We will first consider a chain.

Consider the chain shown in figure 2 consisting of n nodes. Assume that duplicate packets are not detected. We assume that node 1 and n wish to send S packets per second to each other. All other nodes are just repeaters. When node 2 relays a packet to node 3, node 1 must be idle, and all passive acknowledgments are always heard. Thus $q_{1,2} = 1$ and,

$$S_{1,2}^* = S_{1,2} = S \quad (11)$$

At the other end, node n does not repeat the packets, so passive acknowledgments are not possible. However every transmission must be successful since there are no interfering nodes. Thus

$$S_{n-1,n}^* = S_{n-1,n} \quad (12)$$

Node i attempts to transmit all $S_{i-1,i}$ successful packets to $i+1$. Thus

$$S_{i,i+1}^* = S_{i-1,i} \quad i = 2, \dots, n-1 \quad (13)$$

Similarly, in the reverse direction, we have

$$S_{n,n-1}^* = S_{n,n-1} = S$$

$$S_{2,1}^* = S_{2,1} \quad (14)$$

$$\text{and } S_{i+1,i}^* = S_{i,i-1} \quad i = 2, \dots, n-1$$

Equations (11 to 14) relate $S_{i,j}$ and $S_{i,j}^*$ to the desired traffic, S . Note, by symmetry, we obtain relations like equations (14) by replacing i and j by $n-i+1$ and $n-j+1$, respectively.

As an example we evaluate the throughput of a four node chain. First we assume perfect reception of acknowledgments and use only equation (8) with all $S_{i,j} = S$. Then

$$S = G_{12}P(1,2,3) = G_{23}P(1,2,3,4) = G_{21}P(1,2,3) \quad (15)$$

We let $G_{12} = G_1$, $G_{21} + G_{23} = G_2$, and use by symmetry that $G_3 = G_2$, $G_4 = G_1$, and $P(1,2,3) = P(2,3,4)$. From our previous work we have

$$P(1,2,3) = \frac{1+G_4}{D} = \frac{1+G_1}{D}$$

$$P(1,2,3,4) = \frac{1}{D} \quad (16)$$

$$D = 1+2G_1+2G_2+2G_1G_2+G_1^2 = (1+G_1)^2 + 2G_2(1+G_1)$$

Thus we get

$$G_2 = G_1 + G_1(1+G_1) = G_1(2+G_1)$$

$$S = \frac{G_1}{1+G_1+2G_2} = \frac{G_1}{(1+G_1)+2G_1(2+G_1)} \quad (17)$$

We can solve equation (17) for the maximum value of S , the throughput. We obtain $S_{\max} = .128$ when $G_1 = .71$.

To include the effects of passive acknowledgments we use equations (9 to 13) to obtain for arbitrary length chains,

$$S_{12} = S$$

$$S_{i,i+1} = \frac{S}{q_{2,3}q_{3,4}\dots q_{i,i+1}} \quad i=2, \dots, n-2 \quad (18)$$

$$S_{n-1,n} = S_{n-2,n-1}$$

Also,

$$S_{12}^* = S_{23}^* = S$$

$$S_{i,i+1}^* = \frac{S}{q_{2,3}\dots q_{i-1,i}} \quad (19)$$

Returning to the example for $n=4$ we use equations (18) and (10) in (8) to obtain

$$S = G_{12}P(1,2,3) = q_{23}G_{23}P(1,2,3,4)$$

$$= q_{23}G_{21}P(1,2,3)$$

$$= G_{12}P(1,2,3) = G_{23} \frac{[P(1,2,3,4)]^2}{P(1,2,3)} = G_{21}P(1,2,3,4) \quad (20)$$

$$\text{or } G_2 = G_1(1+G_1) + G_1(1+G_1)^2 = G_1(1+G_1)(2+G_1)$$

Thus

$$S = \frac{G_1}{1+G_1+2G_2} = \frac{G_1}{(1+G_1)+2G_1(1+G_1)(2+G_1)} \quad (21)$$

The throughput here is $S_{\max} = .098$ when $G_1 = .37$, a reduction of 23%.

In Table 1 we compare the effects of passive acknowledgments for various length chains. We see that as the chain becomes long the throughput vanishes. Also shown in that table is

Table 1. Effect of Passive Acknowledgments

Length of Chain	Throughput		
	Perfect Acknowledg.	Duplicates Detected	Duplicates Not Detected
4	.128	.106	.098
5	.111	.083	.069
6	.102	.072	.053
7	.097	.066	.044
8	.094	.063	.038
9	.092	.061	.034
10	.091	.060	.031
∞	.086	.057	0

the throughput for chains when duplicate packets are detected and not transmitted to the next node. We analyze this case now.

If duplicate packets are detected and not transmitted further, then $S_{i,j}^*$ is just the desired traffic to be sent from i to j . As before we find $S_{i,j}^*$ from the offered traffic and the routing.

For example, in a chain $S_{i,i+1}^* = S_{i+1,i}^* = S$ for $i = 1, \dots, n-1$. Thus equations (8 to 10) combine to give

$$S_{i,j}^* = G_{i,j} \frac{[P(N_i, N_j)]^2}{P(N_j)} \quad (22)$$

except at terminal nodes where no further transmission is necessary and $q_{ij} = 1$. This can be solved as before to yield the $G_{i,j}$ as a function of the $S_{i,j}^*$ and then the maximum $S_{i,j}^*$ or throughput can be found.

We present the analysis for the four node chain analyzed above. Here $S_{i,i+1}^* = S_{i+1,i}^* = S$ for $i = 1, 2, 3$. Again we use symmetry to obtain

$$S = G_1 \frac{(1+G_1)}{D} = G_{23} \frac{1}{(1+G_1)D} = G_{21} \frac{1+G_1}{D} \quad (23)$$

$$D = (1+G_1)^2 + 2G_2(1+G_1)$$

as before.

$$\text{But } G_2 = G_1 + G_1(1+G_1)^2, \text{ so}$$

$$S = \frac{G_1}{(1+G_1) + 2G_1[1+(1+G_1)^2]} \quad (24)$$

Solving we get $S_{\max} = .106$ when $G_1 = .42$

The throughput for various length chains are given in Table 1. We see here that there is a 34% reduction in throughput for long chains when the effects of passive acknowledg-

ments are included and duplicate packets are detected and not transmitted further.

We next analyze the effect of passive acknowledgments on throughput in star networks. We will consider two topologies - one where all legs are unconnected from each other, the second where the nodes closest to the center are fully connected. We have studied these networks in our previous work and assumed there that all

passive acknowledgments were heard.⁴ We assume further here that duplicate packets are detected and not retransmitted. See Figure 2 for the configuration of the star network with zero connectivity, L legs, and K nodes in each leg. For full connectivity the nodes $0, 1, K+1, 2K+1, \dots, (L-1)K+1$ are fully connected.

The equations used in the previous section are still valid here. When duplicate packets are detected, all $S_{i,j}^* = S$, the throughput in each leg. The end conditions are

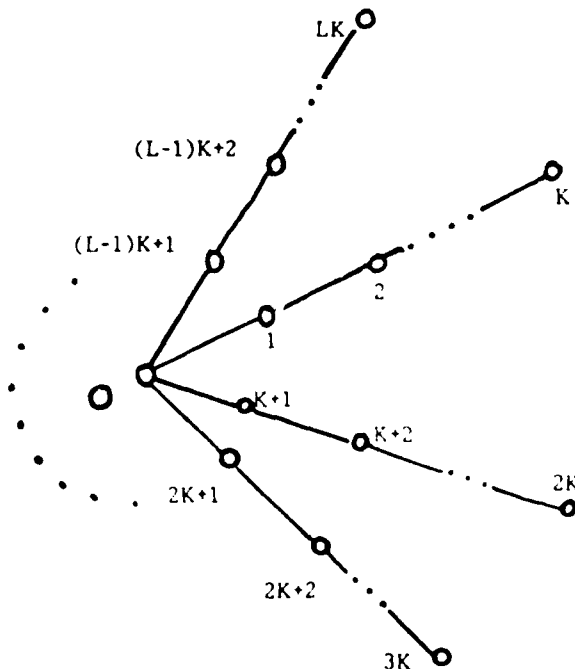
$$q_{2K+L-1, (L+1)K} = 1, \quad L = 0, \dots, L-1 \quad (25)$$

since transmissions to the end nodes are always heard and passive acknowledgments are not sent. This can be found from equation (10). We further assume that

$$q_{2K+1, 0} = 1, \quad L = 0, \dots, L-1 \quad (26)$$

Transmissions to the center node are not always heard. Retransmissions due to collisions are included in our analysis. However since the

Figure 2
A star network with L legs and zero connectivity.



center node does not retransmit, passive acknowledgments are not available. We assume a perfect end-to-end acknowledgment is operating.

Equations 8 to 10 have been solved with perfect passive acknowledgments and with imperfect passive acknowledgments and duplicate packet detection. Two topologies have been studied - zero connectivity and full connectivity around the center. The resultant maximum throughputs per leg are shown in Tables 2 and 3. The maximum throughput from the center node is given in Table 4. We see from these results that there is again a 34% reduction in throughput for long legs and zero connectivity. For full connectivity we find a 43 to 46% reduction in throughput due to the failure to hear passive acknowledgments.

From our previous work we found that except for short legs, the fully connected topology is better when passive acknowledgments are not considered. This is because inward transmissions from neighbors of the center are not collided with. With zero connectivity passive acknowledgments for transmissions from the first ring to the second are interfered with only by the center node. With full connectivity all other first ring nodes add to this interference. Thus the first ring will have to retransmit more. In Table 4 we compare the maximum throughputs from the center node for full and zero connectivity and see that indeed zero connectivity is better.

Table 2: Effects of Passive Acknowledgments on Star Networks. Zero Connectivity (Number of Legs, L=5) - Duplicates Detected

Number of Hops per Leg, K	Maximum throughput per Leg, S_{max}	
	Perfect Acknowledgments	Imperfect Acknowledgments
1	.058	.058
2	.045	.033
3	.044	.0290
4	.044	.0286
6	.044	.0286

Table 3: Effects of Passive Acknowledgments on Star Networks. Full Connectivity - Duplicates Detected

Number of Legs, L	Number of Hops per leg, K	Maximum throughput per Leg, S_{max}	
		Perfect Acknowledg.	Imperfect Acknowledg.
6	3	.04	.0233
	4	.04	.0230
	>5	.04	.0230
9	3	.0286	.0155
	4	.0280	.0154
	>5	.0230	.0154

Table 4: Effect of Passive Acknowledgments on Star Networks. Maximum Throughput from the Center

No. of Legs, L	No. of Hops per leg, K	Connectivity	Maximum throughput per Leg, S_{max}	
			Perfect Acknowledgments	Imperfect Acknowledgments
5	2	Zero	.225	.165
	>3		.220	.145
6	3	Full	.240	.140
	>4		.240	.138
9	3	Full	.257	.140
	>4		.252	.139

V. A New Protocol for Passive Acknowledgments

In the previous section we discussed the effect of imperfect acknowledgments on the maximum obtainable throughput for various topologies. We saw that the maximum obtainable throughput was reduced more than 30 percent for a chain, and more than 44 percent for the star network. The protocol that we used has the following disadvantages:

1. Each node has only one chance to hear acknowledgments from its neighbors.
2. Since the probability of a node hearing an acknowledgment from its neighbors is significantly less than one, some of the successful transmissions are duplicate transmissions.
3. The next node must recognize duplicate transmissions, otherwise the actual throughput will go to zero when the number of hops between source and destination nodes is large.
4. Since each node must wait until transmission of the packet by its neighbor to hear acknowledgments for that packet, the timeout period is large. Thus, delay will be large too.

We propose the following protocol. A node in general has many packets to transmit, in addition to repeating a packet just received. These packets are intended for all its neighbors. The old protocol requires one to wait until that particular packet is repeated. Here we enlarge the header of every packet so that it includes acknowledgment information for all recently received packets. Thus a "passive" acknowledgment can be received on the next packet transmitted. The acknowledgment for a particular packet can be included in the next several packets, so there are several attempts to hear the acknowledgment. At the expense of increased header and more legs in the nodes, the effect of passive acknowledgments can be made negligible.

The protocol that we have developed to overcome these problems is that a node recognizes a passive acknowledgment for its packet by the virtual transmission of that packet. A packet is virtually transmitted if its identifier is transmitted with another packet. Each node when transmitting a packet includes not only the packet's identifier but also the identifiers of previously received packets. A packet's identifier is contained in the next m packets. A node would then have m opportunities to hear if its transmission was received (we do not count the actual transmission). If a packet's identifier is transmitted m times then the acknowledgment for that packet is almost always heard for a sufficiently large value of m . If a packet's transmission is not received, then after a timeout period that packet is rescheduled for transmission.

VI. The Effect of the New Protocol on Throughput

We develop and analyze a new acknowledgment protocol which we refer to as the virtual acknowledgment protocol. In this protocol, if node j hears a packet transmitted to it by node i , node j includes the identifier of this packet in the next m packets it transmits to any node (in addition, of course, to later transmitting the packet itself). We refer to this as m virtual transmissions (and one actual transmission) of the packet. Node i then has m opportunities, rather than one, to hear that node j received its packet. Thus the probability of an acknowledgment being heard is greatly increased over what it was with passive acknowledgments. This increases throughput significantly, roughly 50% for the topologies we have studied.

In addition to this, node i can make the determination as to whether or not to transmit as soon as it hears any packet transmitted by node j . Thus, node i does not have to wait for its packet to be scheduled for retransmission and retransmitted by node j . This reduces delay significantly.

We present the analysis of throughput and overhead due to the increased length of the header and give a general procedure for computing these quantities. We also present analyses of several specific topologies of interest and compare the performance of this protocol with that of passive acknowledgments for these topologies.

Inserting n identifiers into a packet will increase a packet length by a factor (see Fig. 3)

$$\frac{h + d + (n+1)b}{h + d + b} \quad (27)$$

The value of n is related to m as will be shown later. Here n is the number of different identifiers added to the packet. For the moment, we will assume that the average value of n equals m .

The value of m is chosen such that the probability of hearing an acknowledgment, q_{ij} , is close to one. Since every node hears acknowledgments from its neighbors with probability close to one there are almost no duplicate transmissions. Thus nodes do not have to recognize duplicate transmissions for the sake of throughput.

If each node virtually transmits a packet m times then the probability of successfully hearing an acknowledgment can be recomputed as follows:

h bits b bits d bits

Header	ID		Data		
	1	2	(n+1)		
Header	ID	ID	...	ID	Data
h bits	(n+1)b bits				d bits

Figure 3

Packet Formats for the Old and New Protocols

Let q_{ij} be the probability of hearing an acknowledgment. Then the probability of not hearing an acknowledgment after m virtual transmissions is

$$q_{ij}^* = 1 - (1 - q_{ij})^m \quad (28)$$

For large values of m , q_{ij}^* goes to 1. When $q_{ij}^* = 1$ the throughput (in bits/sec) will increase to the results for perfect capture.

q_{ij}^* is substituted for q_{ij} in all throughput equations for chain and star networks which we presented in the previous sections. The maximum obtainable throughput for different topologies are given in Tables 5 and 6. The probability of

Table 5. Effect of imperfect acknowledgments on the throughput as a function of m for the chain network ($S_{ij} = S_{ji} = S$).

Length of Chain (N)	Throughput of a chain number of virtual transmissions, m								Perfect Acknowledgment
	1	2	3	4	5	6	7	8	
4	.106	.119	.124	.127	.127	.1275	.1276	.128	.128
5	.083	.1	.106	.108	.11	.1105	.1108	.1109	.111
6	.072	.09	.096	.1	.101	.102	.102	.102	.102
7	.066	.085	.091	.095	.096	.097	.097	.097	.097
8	.063	.081	.088	.091	.093	.094	.094	.094	.094
10	.06	.078	.085	.088	.09	.09	.0903	.0905	.091

Table 6. Effect of imperfect acknowledgments on the throughput as a function of m for five legs in a star network with zero connectivity ($S_{ij} = S_{ji} = 0$)

Length of Legs (K)	Throughput of a star with five legs number of virtual transmissions, m						Perfect Acknowledgment
	1	2	3	4	5	6	
1	.058	.058				.058	.058
2	.033	.039	.042	.043	.044	.0445	.045
3	.029	.037	.04	.042	.043	.0433	.044
≥ 4	.029	.037	.04	.0415	.042	.043	.044

hearing acknowledgments is also given in Table 7.

We see from Table 7 that the probability of hearing acknowledgments approaches 1 when $m = 5$ for zero connectivity. The throughput almost increases to the values of perfect acknowledgment for $m = 5$.

We compared the two protocols for acknowledgments for chain and star networks. The throughput of the center node increased by 50% for a ten node chain, by 48% for zero connectivity, and by more than 50% for full connectivity.

As we mentioned before the length of a packet is increased by a factor

$$f = \frac{h + d + (m+1) \cdot b}{h + d + b}$$

because of the virtual transmissions. We inserted $m \cdot b$ extra bits into a packet. Thus, the throughput is given by

$$S_{ij} = G_{ij} \cdot P(N_i, N_j) \cdot q_{ij}^* \cdot \frac{h + d + b}{h + d + (m+1) \cdot b} \quad (29)$$

If we choose $b = 12$, $h = 84$, $m = 5$, and $d = 960$ bits then

$$\frac{h + d + b}{h + d + (m+1) \cdot b} = .946 \quad \text{or} \quad \frac{m \cdot b}{h + d + b} \approx .057$$

This means that the throughput decreased 6% due to increased header when compared with perfect acknowledgments. But we know that the throughput increased much more than 6% because of the new protocol. The results in these tables were obtained by using

$$S_{ij} = G_{ij} \cdot P(N_i, N_j) \cdot q_{ij}^* \quad (30)$$

In order to incorporate Eq. (29), the results in Tables 5 and 6 are divided by Eq. (27).

Table 7. Probability of hearing acknowledgment (q_{ij}) as a function of m for five-legs of length 5 in a star network with zero connectivity

	number of virtual transmissions, m									
q_{ij}	1	2	3	4	5	6	7	8	10	
q_{01}	.530	.700	.810	.86	.91	.94	.95	.97	.99	
q_{12}	.670	.890	.960	.990	.997	.999	1			
q_{23}	.890	.980	.990	.999	1					
q_{34}	.910	.990	.994	.999	1					
q_{45}	1	1	1	1						
q_{10}	.910	.990	.999	1						
q_{21}	.930	.990	.999	1						
q_{32}	.940	.990	.999	1						
q_{43}	.970	.999	1							
q_{54}	1									

VII. Optimum m for the New Protocol

In the previous section we have studied the effect of the new protocol on throughput. We found that for $m \geq 4$ the star network with full connectivity retains its superiority with respect to the other connectivities. In this section we will more closely study the L-leg star network with full connectivity.

We know that increasing m will increase q_{ij}^* and hence throughput. There is, however, a tradeoff between the throughput and overhead (due to the increase in the packet's length) due to virtual transmissions. For several cases we find the optimum m . First we assume that at all nodes, if a packet's identifier is transmitted virtually m times, then we also insert m identifiers of received packets into the header of a packet which is ready for transmission. The next m transmissions, after receipt of the packet in question will include the identifier of that packet.

For this case the throughput is given by

$$S_{ij} = G_{ij} \cdot P(N_i, N_j) \cdot q_{ij}^* \cdot \frac{h + d + b}{h + d + (m+1) \cdot b} \quad (31)$$

where j is the one of i 's neighbors.

The results using Eq. (31) were computed for the 9-leg-star network for various values of m when $k = 4$. The optimum m is 10 for this case.

In Eq. (27) we assume that the increment in a packet's length is $m \cdot b$ bits. But we know that the hotspot (node 0) and nodes at the end of the legs do not have to send passive acknowledgments because the transmissions into the hotspot and nodes at the end of the legs will never have collisions. Thus there is no increment in the packet's length for these nodes. Equation (31) can be modified to obtain exact results as follows.

Again we assume that node 0 sends S units of traffic in each direction. Then the nodes that have $m \cdot b$ bits increment in their packet's length must send $1 + S$ units of traffic to their neighbors where

$$l = \frac{h + d + (m+1)b}{h + d + b}$$

For left to right transmissions the throughput equations are:

$$\begin{aligned} S_{01} &= S = G_{01} \cdot P(N_0, N_1) \\ S_{i,i+1} &= l \cdot S = G_{i,i+1} \cdot P(N_i, N_{i+1}) \\ &\quad \cdot q_{i,i+1}^* \text{ for } i = 1, 2, \dots, K-1 \end{aligned}$$

For right to left transmissions: (32)

$$\begin{aligned} S_{k,k-1} &= S = G_{k,k-1} \cdot P(N_k, N_{k-1}) \\ S_{i+1,i} &= l \cdot S = G_{i+1,i} \cdot P(N_{i+1}, N_i) \\ &\quad \cdot q_{i+1,i}^* \text{ for } i = 0, 1, \dots, k-2 \end{aligned}$$

We expect that the results from Eq. (32) are greater than the results from Eq. (31) because Eq. (31) is solved assuming that the packet's length is incremented by $m \cdot b$ bits at all nodes.

The results for Eq. (32) were computed. We found that the optimum m is 11 and the total throughput of the hotspot, S_T , is .2222. When $m > 11$ the throughput decreases because of overhead due to virtual transmissions.

We know that $q_{ij}^* \approx 1$ when $m = 4$ for all i and j except q_{12}^* . This means that nodes 1, 3, 4, ..., $k-1$ need transmit a packet's identifier only four times. Let us assume that node 2 transmits m times and all the other nodes transmit m times. Then the packet's length is incremented by

$$\begin{aligned} l_1 &= m \cdot b \text{ bits} & \text{for } i = 1, 3, 4, \dots, k-1 \\ l_2 &= m \cdot b \text{ bits} \\ l_0 &= l_k = l_{2k} = \dots = l_{LK} = 0 \end{aligned} \quad (33)$$

With these assumptions Eqs. (32) become

for left to right transmissions

$$\begin{aligned} S &= S_{01} = G_{01} \cdot P(N_0, N_1) \\ l \cdot S &= G_{i,i+1} \cdot P(N_i, N_{i+1}) \\ &\quad \cdot q_{i,i+1}^*, i = 1, 3, 4, \dots, k-1 \\ l' \cdot S &= G_{23} \cdot P(N_2, N_3) \cdot q_{23}^* \end{aligned}$$

for right to left transmissions we have: (34)

$$\begin{aligned} l \cdot S &= G_{i+1,i} \cdot P(N_{i+1}, N_i) \\ &\quad \cdot q_{i+1,i}^*, i = 0, 1, 2, 3, \dots, k-1 \\ l' \cdot S &= G_{21} \cdot P(N_1, N_2) \cdot q_{21}^* \\ S &= S_{k,k-1} = G_{k,k-1} \cdot P(N_k, N_{k-1}) \end{aligned}$$

where

$$l = \frac{h + d + (m+1)b}{h + d + b} \quad (35)$$

and

$$l' = \frac{h + d + (m'+1)b}{h + d + b} \quad (36)$$

For $h = 84$, $b = 12$, $d = 960$, and $m = 4$ the optimum m' is 19, and the total throughput of the hotspot, S_T , is .2390. With $m = 4$ and $m' = 19$ all $q_{ij}^* \approx 1$, therefore, the nodes will not send any significant number of duplicate transmissions to the next hop. Since for node 2, $m' = 19$, the increment in a packet's length there is 22% (228 bits). For nodes 1 and 3, the increment in the packet's length is 4.5% (48 bits).

We can further increase the throughput by reducing overhead as follows:

A node, i , receives $S_{R(i)}$ units of the successful transmission (throughput) from its neighbors where

$$S_{R(i)} = \sum_{j \in N_i} S_{ji} \quad (37)$$

For each unit of successful transmission node i sends m_i passive acknowledgments so that the total number of acknowledgments at node i is $m_i \cdot S_{R(i)}$ per unit of time. But node i transmits with a rate of $G_i \cdot P(N_i)$ and we know that $G_i \cdot P(N_i) \geq S_{R(i)}$. Therefore each transmission must carry r_i passive acknowledgments where

$$r_i = \frac{m_i \cdot S_{R(i)}}{G_i \cdot P(N_i)} \quad (38)$$

The overhead in a packet's length at node i is given by

$$l_i = r_i \cdot b, \quad \text{for } i = 0, 1, 2, \dots, k.$$

If we assume that the hotspot (node 0) and nodes at the end of each path (leg) want to send S packets/unit time then the throughput is given by

$$S_{ij} = \frac{h + d + r_i \cdot b}{h + d + b} \cdot S = G_{ij} \cdot P(N_i, N_j) \cdot q_{ij}^* \quad (39)$$

for $i = 0, 1, \dots, k$, and where j is a neighbor of node i .

For instance for the transmissions from left to right we have:

when $i = 0$

$$S_{01} = S = G_{01} \cdot P(N_0, N_1) \cdot q_{01}^*$$

$i = 1$

$$S_{12} = \frac{h + d + m \cdot \left(\frac{S_{01} + S_{21}}{G_1 \cdot P(N_1)} \right) \cdot b}{h + d + b}$$

$$\cdot S = G_{12} \cdot P(N_1, N_2) \cdot q_{12}^*$$

$i = 2$

$$S_{23} = \frac{h + d + m \cdot \left(\frac{S_{12} + S_{32}}{G_2 \cdot P(N_2)} \right) \cdot b}{h + d + b}$$

$$\cdot S = G_{23} \cdot P(N_2, N_3) \cdot q_{23}^*$$

$i = k-1$

$$S_{k-1,k} = \frac{h + d + m \cdot \left(\frac{S_{k-2,k-1} + S_{k,k-1}}{G_{k-1} \cdot P(N_{k-1})} \right) \cdot b}{h + d + b}$$

$$\cdot S = G_{k-1} \cdot P(N_{k-1}, N_k) \cdot q_{k-1,k}^*$$

Equation (39) was used to compute throughputs for $m = 4$, $L = 9$ and various m' in a star network. We found that the total throughput of the hotspot, S_T , is .2454 and the optimum m' is 27. Even though node 2 transmits a packet's identifier 27 times the increment in the packet's length is only 78 bits which is a 7% increment. This means that at node 2 each transmission carries 6.5 identifiers ($r_2 = 6.5$). We also found the $r_1 = 3.79$ and $r_3 = 3.82$ which are 4% increments. Previously we saw that there was a 46% reduction in throughput due to imperfect acknowledgments when we used the old protocol (no virtual transmissions). Here we see that with the new protocol there is only a 2.6% reduction in throughput with respect to the result for perfect acknowledgments (.252). Up to this point we assumed that the original length of a packet is 1056 bits (header = 96, data = 960).

The effect of the overhead depends also on the length of the data portion of a packet. If the data portion of a packet, d , is small the overhead will be large and the throughput will be reduced. We have solved Eq. (39) for different values of d . The results are given in Table 8. In Table 8 we also tabulate the reduction from the result for perfect acknowledgments. We see from Table 8 that when d increases the throughput also increases and the overhead gets smaller. When $d = 50$ bits the overhead at node two is 36% but when $d = 2000$ the overhead is only 4%. The reduction in throughput from the result of perfect acknowledgment is 18% when $d = 50$ but it is only .9% when $d = 2000$.

Table 8. Total throughput of the hotspot in a star network ($m = 4$, $L = 9$, $K = 4$) for different values of d .

d	m'	S_T	% Reduction from perfect acknowledgments
50	16	.2074	18
100	20	.2173	14
200	15	.2243	11
400	19	.2355	7
800	27	.2440	3
1600	35	.2489	1

VIII. Conclusions

We have shown that the performance of a multihop packet radio system is significantly reduced by the use of passive acknowledgments. We developed and analyzed a virtual acknowledgment protocol which results in near-perfect acknowledgments with very little overhead. Thus we are able to overcome most of the difficulty due to imperfect passive acknowledgments. We believe that this virtual acknowledgment scheme will also have a favorable impact on time delay and are currently in the process of analyzing this problem.

References

- 1) L. Kleinrock and F. Tobagi, "Packet switching in radio channels," parts I and II, IEEE Trans. Commun., vol. COM-23, pp. 1400-1433, December 1975.
- 2) F. Tobagi, "Analysis of a Two-Hop Centralized Packet Radio Network", Part II: CSMA IEEE Trans. Commun., COM-28, pp. 208-216, February 1980.
- 3) R.E. Kahn, "The organization of computer resources into a packet radio network," IEEE Trans. Communications, Vol. COM-25, pp. 169-178, January 1977.
- 4) R. Boorstyn and A. Kershenbaum, "Throughput Analysis of Multi-hop Packet Radio Networks," IEEE-ICC'80, pp. 13.6.1-13.6.6, Seattle, WA., June 1980.
- 5) V. Sahin, "Analysis of Multihop Packet Radio Networks", Ph.D. Thesis, Polytechnic Institute of New York, June 1982.

A.4 Extensions to the Analysis of Multihop Packet Radio
Networks

Maglaris, Boorstyn, and Kershenbaum

EXTENSIONS TO THE ANALYSIS OF MULTIHOP PACKET RADIO NETWORKS*

B. Maglaris, R. Boorstyn and A. Kershenbaum

Polytechnic Institute of New York
333 Jay Street, Brooklyn, New York 11201ABSTRACT

Multihop packet radio networks operating under CSMA with perfect capture have been analyzed with an exact Markov procedure for exponentially distributed packet lengths. In this paper we generalize this procedure and improve its computational complexity. We first prove that the exact expressions of the Markov model apply for arbitrary packet length distributions with rational Laplace transforms. The analysis is shown to depend on mean values only, and it is extended to networks where the average packet lengths depend upon the destination. Finally, it is demonstrated how certain properties of the analytic expressions permit the decomposition of large networks into computationally manageable segments.

I. INTRODUCTION

Packet radio networks consist of geographically dispersed Packet Radio Units (PRU's) broadcasting data over a limited distance. In case two units cannot have a direct connection, intermediate PRU's will act as relays, thus creating a multihop communication network. This is represented by a graph with nodes as sources, destinations, and relays, and arcs connecting nodes which are within each other's range. Several technologies and routing protocols have been proposed, e.g., [1] and [2], in order to establish reliable end-to-end paths for maximum network throughput. The need to evaluate different protocols and routing procedures initiated several analytic and simulation studies on packet radio network models. In [3] Kleinrock and Tobagi analyzed single-hop centralized networks employing Carrier Sense Multiple Access (CSMA). In this context they demonstrated the effect of propagation delay and of hidden terminals (PRU's transmitting to the central hub but not listening to each other). In [4] Tobagi considered a simple finite state model for two-hop networks. In [5] and [6] multihop packet radio networks using a slotted ALOHA protocol were analyzed. In [7] Boorstyn and Kershenbaum introduced an analytic procedure under which multihop CSMA networks with perfect capture can be analyzed efficiently in terms of their throughput performance. This technique yields exact results if

certain assumptions are made, the most important of which implies a Markov property as the underlying process. Namely, scheduled packet transmissions are assumed to form independent Poisson processes at each node and packet lengths are exponentially distributed. (A discussion of how these assumptions relate to a more realistic model is given in [7]. The most critical of these assumptions is a long rescheduling time.) This technique was used to evaluate deployments exhibiting various degrees of connectivity and was extended to investigate passive and active acknowledgment protocols [8].

In this paper we prove that the Markov analysis presented in [4] holds for arbitrary packet length distribution thus relaxing the exponential length assumption. Similarly it is demonstrated that PRU's transmitting (or relaying) different length packets to their neighbors can be easily handled via the same technique. These results prove to be insensitive to the packet length distribution under the perfect capture assumption. Finally, it is shown how certain properties of the probability measure on node sets decompose complex topologies and accelerate the computational procedure.

II. THE NETWORK MODEL

We assume that a large number of PRU's, dispersed within a geographical area, transmit packets of data addressed to some destination nodes. Due to the broadcast mode of transmission, all neighboring node PRU's within the transmitter's range hear the packet and depending on its address, either discard it or relay it until it reaches its eventual destination. This deployment is represented by a graph whereby neighboring PRU's are connected with an arc. All units transmit and receive on the same channel via a Carrier Sense Multiple Access (CSMA) protocol. Before initiating a transmission, a source (or relay) checks whether all its neighbors are idle. A collision may, however, result due to "hidden" terminals outside the range of a PRU initiating transmission but interfering with the receiving node (the receiver is a common neighbor to two non-neighboring transmitters). Collisions are detected via a perfect acknowledgment protocol whereby acknowledgments are assumed to be always heard and do not consume any network capacity. Packets encountering busy carrier condition or suffering collisions, are rescheduled for transmissions after a long randomized interval.

* This work has been supported in part by US ARMY CECOM under contract DAAK80-80-K-0579, and by IBM Federal Systems Division.

A FRU may not transmit and receive simultaneously. It can, however, lock its receiver to the first packet addressed to it and ignore subsequent colliding broadcasts. This is referred to as "perfect capture" in packet radio networks.

For networks operating as above, our primary objective is to assess the performance of a routing protocol, as defined by relaying decisions, under a given source-destination traffic profile. Formally, given the network topology (location of nodes and connectivity), the packet rates and packet length averages per source-destination requirement (end-to-end traffic) and the routing scheme, we can easily deduce s_{ij} , the rate in packets/sec that node i has to deliver successfully to its neighbor node j . Then, we want to evaluate g_{ij} , the rate at which node i schedules potential packet transmissions to its neighbor j . The ratio s_{ij}/g_{ij} denotes the probability of successful transmission of a scheduled packet and provides a measure for stability and delay since successful packets must retry after a random delay. Extensions to delay analyses and effects of acknowledgment schemes are not considered in this paper.

The analytic technique described below will yield an exact algorithm for g_{ij} under the following assumptions:

- 1) Zero propagation delay between neighbors. This implies perfect CSMA and does not allow any collisions other than those due to hidden terminals. It is a valid approximation for deployments within reasonable distances.
- 2) Packet lengths are distributed according to any distribution having a rational Laplace transform. This class includes almost all distributions used in practice, either directly or as limits from within the class.
- 3) Receivers capture the first transmission that reaches them (perfect capture).
- 4) Perfect acknowledgments.
- 5) The streams of scheduled packets at the nodes form independent Poisson processes. This assumption is consistent with Poisson offered (exogenous) traffic and long random rescheduling delays. Simulation studies for single-hop ALOHA [9] indicate that if rescheduling is delayed more than 10 times the packet length, this critical assumption is a valid one.
- 6) Packet lengths are assumed to be independently reassigned at each node in a path. This is consistent with the rescheduling delay assumptions above.

III. ANALYSIS FOR PACKET LENGTH DISTRIBUTIONS WITH ARBITRARY RATIONAL LAPLACE TRANSFORMS

For a given node, i , the total scheduling process out of i will be Poisson with rate g_i .

$$g_i = \sum_{j \in N_i} g_{ij} \quad (1)$$

where N_i is the set of neighbors of i . Node i generates packets with lengths having a distribution with rational Laplace transform. Let $\frac{1}{\mu_i}$ be the average packet length from node i .

Let V denote the set of all nodes. At any instant in time, the system will be characterized by the subset of nodes D which are simultaneously transmitting. Obviously nodes in D may not be neighbors.

The main theory of this section states that, under the assumptions above, the steady state probability of D , $Q(D)$ is given by:

$$Q(D) = Q(D-i) \frac{g_i}{\mu_i} \quad (2)$$

or

$$Q(D) = Q(\emptyset) \prod_{i \in D} \frac{g_i}{\mu_i} \quad (3)$$

where

$$Q(\emptyset) = \frac{1}{\sum_{D \subset V} \prod_{i \in D} \frac{g_i}{\mu_i}}$$

Here we define $\prod_{i \in D} \frac{g_i}{\mu_i} = 1$ if $D = \emptyset$.

These results are identical with those reported in [7] for exponential packet lengths. Thus, all results in [7] apply for general packet lengths. Namely, with N_i denoting the set of neighbors of node i and $P(A)$ the probability that all the nodes in a set A are idle, we have

$\Pr \{ \text{successful transmission from } i \text{ to } j \} =$

$$\frac{s_{ij}}{g_{ij}} = P(N_i \cup N_j) \quad (4)$$

Using the notation

$$SP(B) \triangleq \sum_{D \subset B} \prod_{i \in D} \frac{g_i}{\mu_i} \quad (5)$$

we have from (3) that

$$P(A) = \sum_{D \subset (V-A)} Q(D) = \frac{SP(V-A)}{SP(V)} \quad (6)$$

Subsets D in (5) and (6) consist of non-communicating nodes as in (3). Equations (4) and (6) form the basis of an iterative procedure, which along with definition (1), determines g_{ij} values, for given s_{ij} rates, if they exist, and diverges otherwise.

We now proceed to the proof of formulas (2) and (3) for general packet length distributions, which are different for each node. We use the method of stages as developed by Cox, [10], to decompose arbitrary service distributions to a combination of exponential services. Cox demonstrated that any distribution with rational Laplace transform, can be represented by the general Erlang-branching configuration of Figure 1. As in the figure, a node i which initiates a packet transmission, activates the first exponential server with rate μ_1^i . With a given probability $1-P_1^i$ the packet terminates transmission at this stage, otherwise it enters a second exponential server. Since only one packet is transmitted at a time, a busy node will correspond to one of the n stages being active. The average packet transmission time $1/\mu_i$, equals the average residency within the stage sequence of Figure 1.

$$\frac{1}{\mu_i} = \frac{1}{\mu_1^i} + \frac{P_1^i}{\mu_2^i} + \frac{P_1^i P_2^i}{\mu_3^i} + \dots + \frac{P_1^i P_2^i \dots P_{n-1}^i}{\mu_n^i} \quad (7)$$

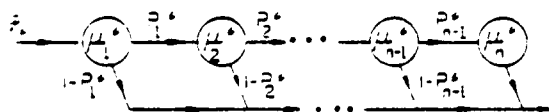


Fig. 1 Stage Decomposition for Node i

The rational Laplace transform class includes or approximates all distributions found in practice. As an example of the flexibility of the method, note that a single exponential server models exponential packet lengths and that a series of servers ($P_k^i=1$) approaches the constant packet length case as the number n increases.

The system with decomposed nodes as above, is a Markov process with Poisson scheduling and exponential servers. The state of the system must contain information on busy (transmitting) nodes and their respective current stage of transmission. Let D_k be the set of busy PRU's at stage k . The network state E is the collection of subsets D_k . The set D of busy PRU's regardless of stage is given by

$$D = D_1 \cup D_2 \cup \dots \cup D_n$$

Let N_D denote the set of neighbors of all nodes in D including D . We will make use of the following notation in order to write state transition (global balance) equations

$\{D_1, \dots, D_{k-1}, \dots, D_n\}$ denotes a state identical to E , with node i removed from the set D_k

$\{D_1, \dots, D_{k+1}, \dots, D_n\}$ similarly denotes the addition of node i to D_k

The global balance equations will equate flow into state E to flow out of state E . There are several possible ways of entering or exiting a state. For example, a new arrival to state $\{D_1-i, D_2, \dots, D_n\}$ with rate g_i will yield state E . A change of stage $k-1$ to k in $\{D_1, D_2, \dots, D_{k-1}+i, D_k-i, \dots, D_n\}$ with rate P_{k-1}^i/μ_{k-1}^i will result in E . Accounting for all possible transitions, and denoting the stationary probability of state E by $Q(E)$, we have:

$$\begin{aligned} & \sum_{i \in D_1} g_i Q(D_1-i, D_2, \dots, D_n) \\ & + \sum_{k=2}^n \left\{ \sum_{i \in D_k} \mu_{k-1}^i P_{k-1}^i Q(D_1, \dots, D_{k-1}+i, D_k-i, \dots, D_n) \right. \\ & \left. + \sum_{k=1}^n \sum_{i \notin N_D} (1-P_k^i) \mu_k^i Q(D_1, \dots, D_k+i, \dots, D_n) \right\} \\ & = \left\{ \sum_{i \in D_1} \mu_1^i + \sum_{k=2}^n \sum_{i \in D_k} \mu_k^i + \sum_{i \notin N_D} g_i \right\} Q(E) \quad (8) \end{aligned}$$

We observe that the global balance equations (8) can be decomposed into three sets of consistent local balance equations:

$$g_i Q(D_1-i, D_2, \dots, D_n) = \mu_1^i Q(E) \quad \forall i \in D_1 \quad (9)$$

$$\mu_{k-1}^i P_{k-1}^i Q(D_1, \dots, D_{k-1}+i, D_k-i, \dots, D_n) = \mu_k^i Q(E) \quad \forall i \in D_k, \quad 2 \leq k \leq n \quad (10)$$

$$\sum_{k=1}^n (1-P_k^i) \mu_k^i Q(D_1, \dots, D_k+i, \dots, D_n) = g_i Q(E) \quad \forall i \notin N_D \quad (11)$$

Equations (9) and (10) equate the flow into E due to arrivals at stage k of node i to the flow out of E due to departures from the same stage of the same node. Equations (11) are obtained by considering an additional stage, the idle stage

whereby node i is idle but permitted to transmit, $i \in N_D$. Then we equate the flow into E due to arrivals to the idle stage of node i to the flow out of E due to departure from this stage of node i . Equation (11) can be obtained from (9) and (10) by rearranging (9) and summing (9) and (10) for all k 's. Thus (9) and (10) constitute an independent set of local balance equations, consistent with (11) and summing up to the global balance equation (8). For similar reasoning on local balance equations, see [11] and [12]. In [12] similar processes are characterized within the context of Markov fields for spatial processes.

From the local balance equations, it easily follows that the stationary state probabilities $Q(E)$ possess product form solutions, namely

$$Q(D_1, D_2, \dots, D_n) = Q(\emptyset) \prod_{i \in D_1} \frac{g_i}{\mu_i} \times \prod_{i \in D_2} \frac{g_i P_1^i}{\mu_i^2} \times \dots \times \prod_{i \in D_n} \frac{g_i P_1^i \dots P_{n-1}^i}{\mu_i^n} \quad (12)$$

If $Q(D)$ denotes the steady-state probability that the nodes in D are busy regardless of the stage of transmission, $Q(D)$ will consist of the sum of all states $\{D_1, D_2, \dots, D_n\}$ which include a node $i \in D$ at some stage k , $i \in D_k$. Thus

$$Q(D) = \sum_{\substack{\text{All partitions such that} \\ D = D_1 \cup D_2 \cup \dots \cup D_n}} Q(D_1, D_2, \dots, D_n)$$

or

$$Q(D) = Q(\emptyset) \prod_{i \in D} g_i \left\{ \frac{1}{\mu_i} + \frac{P_1^i}{\mu_i^2} + \frac{P_1^i P_2^i}{\mu_i^3} + \dots + \frac{P_1^i P_2^i \dots P_{n-1}^i}{\mu_i^n} \right\}$$

and from (7)

$$Q(D) = Q(\emptyset) \prod_{i \in D} \frac{g_i}{\mu_i} \text{ q.e.d.}$$

Thus the steady state probabilities of the set of busy nodes has a product form solution, and depends on the average packet length (possibly different for each node) for packet length distributions with rational Laplace transform. These distributions need not be the same for each node. The number of stages n used in the proof corresponds to the maximum number of stages in the network. For nodes with smaller n , the branching probability P_k can be set to zero to truncate stages.

IV. NODES TRANSMITTING DIFFERENT PACKET LENGTHS TO EACH NEIGHBOR

In the previous section, we assumed that each node i schedules packet transmissions at an aggregate rate g_i and with average length $1/\mu_i$.

A slight modification extends the analysis to scenarios whereby a node i transmits packets of different average length to each of its neighbors j .

Let g_{ij} and $1/\mu_{ij}$ be the scheduling rate and average packet length for the i to j transmission. We keep the same structure as before by breaking the node i into a set of "micronodes," one for each neighbor. Micronodes are connected in our topology if they can hear each other. Obviously micronodes belonging to the same node i are fully connected and so are micronodes belonging to nodes connected in our initial topology. As an example, a five node chain will be decomposed as in Figure 2. Since nodes 2, 3 and 4 are transmitting to two neighbors they are decomposed into two micronodes. By applying the results of the previous section, we have that for a successful transmission from 2 to 3,

$$\frac{s_{21}}{s_{21}} = P(1, 2, 2', 3, 3') = \frac{SP(4, 4', 5)}{SP(V)}$$

where

$$SP(4, 4', 5) = 1 + \frac{g_{43}}{\mu_{43}} + \frac{g_{45}}{\mu_{45}} + \frac{g_{54}}{\mu_{54}}$$

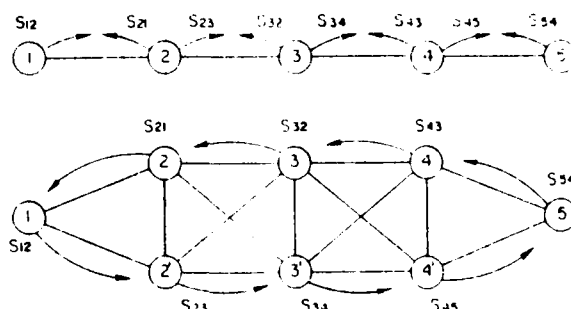


Fig 2 Micronode Decomposition of a Five-Node Chain

Now, define $G_4 = \frac{g_{43}}{\mu_{43}} + \frac{g_{45}}{\mu_{45}}$ and $G_5 = \frac{g_{54}}{\mu_{54}}$,

in other words let G_i be the average normalized scheduling rate of node i .

$$G_i = \sum_j \frac{g_{ij}}{\mu_{ij}} \quad (13)$$

It follows that for this example results are identical as in the original topology before decomposition but with g_i/μ_i replaced by (13).

In general consider $SP(B^*)$ where B^* is the set of micronodes associated with the set of nodes B . Let node $i \in B$ and denote its micronodes by ij where j are neighbors of i . We use the relation [5].

$$SP(B) = SP(B-1) + \frac{g_1}{\mu_1} SP(B-N_1) \quad (14)$$

For micronodes,

$$SP(B^*) = SP(B^* - \{1, j\}) + \frac{g_{1j}}{\mu_{1j}} SP(B^* - N_{1j}^*)$$

Note that because of the connectivity of micronodes, N_{1j}^* is the same for all j , namely the set of micronodes of N_1 , N_1^* . Let 1^+ be the set of micronodes of 1. Then proceeding recursively we have

$$SP(B^*) = SP(B^* - 1^*) + \sum_j \frac{g_{1j}}{\mu_{1j}} SP(B^* - N_{1j}^*) \quad (15)$$

Using (13) and repeating the above process we find that equation (6) still holds for the original nodes, but with g_{ij}/μ_{ij} replaced by G_i given in (13).

In the special case where all packets transmitted by 1 have average length $1/\mu_1$, then $G_1 = g_1/\mu_1$.

V. COMPUTATION OF SP(A)

As seen from equations (4), (5) and (6) evaluation of s_{ij}/g_{ij} requires an efficient way of computing SP(A) expressions for various sets of nodes. Recall that SP(A) stands for sums of products of rates G_i on all possible independent subsets of A. By an independent set we denote a set of nodes which do not communicate (i.e. are not neighbors).

$$SP(A) = \sum_{D \subseteq A} \prod_{i \in D} G_i$$

A straightforward algorithm to evaluate SP(A) would require identification of all independent subsets of A, a problem dual to identification of all cliques in a graph and thus NP-complete (a clique is a fully connected subset of a graph). It is however possible to handle considerable size networks by using several properties of the SP(A) function. We summarize some of them and demonstrate via an example how to decompose a network into smaller segments.

If two subsets of nodes A and B are isolated from each other, then in [7] it is shown that

$$SP(A \cup B) = SP(A) SP(B) \quad (16)$$

If C is an arbitrary subset of V, it can be shown that

$$SP(V) = SP(V-C) + \sum_{D \subseteq C} \left\{ \prod_{i \in D} G_i \cdot SP(V-C-N_D) \right\} \quad (17)^*$$

* This result is due to Mr. W. Chen, of the Bell Telephone Laboratories and the Polytechnic Institute of New York and can be proven by considering all terms not including nodes in C and then all terms involving different independent subsets of C.

As a special case with C containing one node 1

$$SP(V) = SP(V-1) + G_1 SP(V-N_1) \quad (20)$$

Sets N_1 and N_D above, denote the sets of neighbors of 1 and all nodes in D respectively. Finally, if C is a cut, i.e. a set of nodes which when removed decomposes the network into two isolated subsets A and B, it follows from (16) that for all independent sets $D \subseteq C$

$$SP(V-C-N_D) = SP(A-N_D) SP(B-N_D) \quad (19)$$

and

$$SP(V-C) = SP(A) SP(B)$$

As an example, consider the 10 node network illustrated in Figure 3. By choosing $C = \{5, 6\}$, the network is decomposed into three subsets, A, B and C with A and B being totally isolated from each other. Applying (17) and (19) we can reduce the computations needed to evaluate SP(V) as follows:

$$\begin{aligned} SP(V) &= SP(1, 2, 3, 4, 5, 6, 7, 8, 9, 10) = \\ &= SP(1, 2, 3, 4, 7, 8, 9, 10) + G_5 SP(1, 4, 9, 10) \\ &\quad + G_6 SP(1, 2, 7, 10) \\ &\quad + G_5 G_6 SP(1, 10) \\ &= SP(1, 2, 3, 4) SP(7, 8, 9, 10) + G_5 SP(1, 4) \cdot SP(9, 10) \\ &\quad + G_6 SP(1, 2) \cdot SP(7, 10) \\ &\quad + G_5 G_6 SP(1, 10) \end{aligned}$$

We can now proceed to direct evaluation of sums-of-products for subsets with 4 nodes at the most instead of 10 initially.

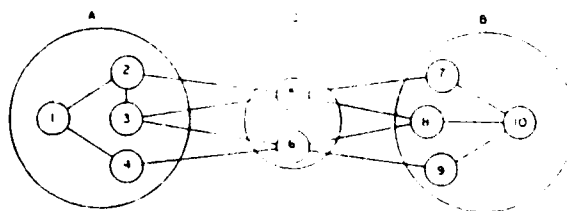


Fig 3 Decomposition Example

VI. CONCLUSIONS

We have shown how our work on an analysis of multihop packet radio networks, originally done for exponential packet lengths, can be extended to packet length distributions with rational Laplace transforms. Thus the results previously obtained can be used in this more general case with only

the average packet length being required. It was also shown that the packet length distributions transmitted by a single node to different neighbors need not be the same. Finally, a method was described to simplify the writing of the analytic expressions for large networks.

VII. REFERENCES

1. Kahn, R.E., "The Organization of Computer Resources into a Packet Radio Network," IEEE Transactions on Communications, Vol. COM-25, Jan. 1977, pp. 159-178.
2. Gitman, I., R. Van Slyke and H. Frank, "Routing in Packet-Switching Broadcast Radio Networks," IEEE Transactions on Communications, Vol. COM-24, Sept. 1976, pp. 926-930.
3. Kleinrock, L., and F. Tobagi, "Packet Switching in Radio Channels, Part I and II," IEEE Transactions on Communications, Vol. COM-23, Dec. 1975, pp. 1400-1433.
4. Tobagi, F., "Analysis of a Two-Hop Centralized Packet Radio Network, Part II: CSMA," IEEE Transactions on Communications, Vol. COM-28, Feb. 1980, pp. 208-216.
5. Kleinrock, L., and J. Silvester, "Optimum Transmission Radii for Packet Radio Networks or why Six is a Magic Number," Proceedings of the NTC '78, Birmingham, AL, Dec. 1978.
6. Kleinrock, L., and J. Silvester, "An Iterative Scheme for Performance Modeling of Slotted ALOHA Packet Radio Networks," Proceedings of the ICC '82, Philadelphia, PA, June, 1982, pp. 60.1-60.5.
7. Boorstyn, R., and A. Kershenbaum, "Throughput Analysis of Multi-Hop Packet Radio Networks," Proceeding of the ICC '80, Seattle, WA, June 1980, pp. 13.6.1-13.6.6. (An expanded version of this paper has been accepted for publication in the IEEE Transactions on Communications.)
8. Boorstyn, R., A. Kershenbaum, and V. Sahin, "A New Acknowledgment Protocol for Analysis of Multihop Packet Radio Networks," Proceedings of the COMCON '82, Washington, DC, Sept. 1982, pp. 355-392.
9. "Queueing Systems," Volume 2, L. Kleinrock, John Wiley & Sons, New York, 1976, p. 374.
10. "The Theory of Stochastic Processes," D.R. Cox, and H.D. Miller, Chapman and Hall, London, 1965.
11. Baskett, F., M. Chandy, R. Muntz, and J. Palacios, "Open, Closed and Mixed Networks of Queues with Different Classes of Customers," Journal of the ACM, Vol. 22, No. 2, April 1975, pp. 248-260.
12. "Reversibility and Stochastic Networks," F.P. Kelly, John Wiley & Sons, New York, 1979.

Multiple Access Techniques with Arbitrary Packet

or File Distributions

Final Technical Report, March 1982

B. Multiple Access Techniques with Arbitrary Packet Length Distributions

B.1 Introduction

In our original multihop packet radio analysis, we assumed exponentially distributed packet lengths. We have been able to generalize the analysis for packet lengths having densities formed by the positive sum of exponential terms (see Appendix C). In our analysis we assumed that propagation delays among neighboring PRU's are negligible. Thus in a Carrier Sensing Multiple Access (CSMA) mode of operation, collisions may occur due to the "hidden terminal" phenomenon only (i.e., two non-communicating PRU's schedule packet transmissions to a common neighbor simultaneously).

CSMA analyses incorporating the effects of propagation delays have been reported extensively in the literature for single-hop networks (i.e., all PRU's hear each other) and fixed packet sizes. As a first step in generalizing these results, we studied single-hop multiple access protocols with non-fixed packet lengths. Although our main thrust is on CSMA packet radio, we also derived formulas for pure ALOHA and CSMA with collision detection (CSMA/CD). The former was a necessary step in order to demonstrate the impact of packet length distribution on the simplest multiple access method, whereas the latter is a straightforward extension of pure CSMA and is especially popular in local networking environments. Note that the pure ALOHA case was studied previously [5], whereas no extension has been reported on CSMA to our knowledge. The CSMA/CD result is so simple that it may already be known.

In what follows, we summarize the variable packet length analyses in pure ALOHA, CSMA and CSMA/CD. In all cases we assumed infinitely many Poisson sources and Poisson aggregate scheduling processes, with rates s and g packets/sec respectively. Packet lengths are distributed arbitrarily.

B.2 Pure ALOHA

Referring to Figure 9, we consider a transmission of length Y (shaded). This transmission will be successful if a) no other packet is transmitted in Y seconds and b) no previously transmitted packet is still transmitting. We are assuming zero capture. Calling these probabilities P_a and P_b , we obtain $s = gP_aP_b$. But

$$P_a = \int_0^{\infty} e^{-gy} fY(y) dy = MY(-g)$$

where

$$MY(g) = \int_0^{\infty} e^{gy} fY(y) dy$$

is the moment generating function of Y and $fY(y)$ is its density. P_b is found by considering the T second interval prior to the transmission in question. Assume transmissions in that interval occur T_i seconds before the start of our test transmission and have length Y_i . Then

$$P_b = \lim_{T \rightarrow \infty} P(\text{all } T_i > Y_i)$$

But the number of transmissions in T is Poisson and all are identically distributed and independent. Therefore

$$\begin{aligned}
 P(\text{all } T_i \geq Y_i) &= \sum_{k=0}^{\infty} [P(T_i \geq Y_i)]^k \frac{(gT)^k}{k!} e^{-gT} \\
 &= e^{-gT[1-P(T_i \geq Y_i)]} = e^{-gTP(T_i < Y_i)}
 \end{aligned}$$

Here T_i is uniform in the interval $(0, T)$ and Y_i is distributed as Y . They are independent. Thus

$$P(T_i < Y_i) = \frac{1}{T} \int_0^T [1 - F_Y(t)] dt$$

and

$$TP(T_i < Y_i) \rightarrow \int_0^{\infty} [1 - F_Y(t)] dt = E(Y)$$

Here $F_Y(y)$ is the distribution function of Y and $E(Y)$ its expectation. Finally we have

$$s = gM_Y(-g) e^{-gE(Y)}.$$

Note that if Y is fixed then $s = ge^{-2gy}$ as it should.

But Y is the length of the transmitted packets. Condition (b) above does not involve the length of the transmitted packet Y . But condition (b) does! Longer packets are more likely to suffer a collision. Let X be the length of the offered (or successful) packets. Y should in a sense be larger since longer packets are retransmitted more often. Due to condition (a) alone a packet of length x will be successfully transmitted with probability e^{-gx} and requires an average of e^{gx} transmissions to be successful. Thus

$$f_Y(y) = e^{gy} f_X(y)/M_X(g).$$

Also

$$E(Y) = \frac{dM_X(g)}{dg} / M_X(g)$$

and $P_a = 1/M_X(g)$.

Thus $s = \frac{g}{M_X(g)} e^{-gM_X'(g)/M_X(g)}$

This also reduces to $s = ge^{-2gx}$ when $M_X(g) = e^{gx}$. This result has been already established [5] but is derived here in a different manner.

B.3 CSMA

Refer to Figure 10 for CSMA. After an idle period a packet scheduled with rate g is transmitted. The packet lasts for X seconds. In the propagation time a after transmission any other scheduled packet can also be transmitted thus causing a collision. We assume $X \geq a$. Again we assume zero capture. If no such packet is transmitted, then the original transmission is successful, lasts for X seconds, and is followed by an a second period to clear the channel and the idle state resumes. Thus the successful rate is

$$s = ge^{-ga} P(\text{channel is idle}).$$

But $P(\text{channel is idle}) = \frac{1/g}{1/g + a + E(Z)}$

where $1/g$ is the average idle time and Z is the busy period exclusive of the last a seconds.

To evaluate $E(Z)$ we denote the times of the transmissions of interfering packets as T_i and their lengths as X_i . The number of such packets is exponential. Collisions depend only upon the propagation time a and not on the length of the transmitted packet as in ALOHA. Thus

$$Z = \max \{X, T_i + X_i\}.$$

$$\text{and } F_Z(z) = F_X(z) \sum_{k=0}^{\infty} [F_W(z)]^k \frac{(ga)^k}{k!} e^{-ga}$$

where $W = T_i + X_i$, T_i is uniform in $(0,a)$ and is independent of X_i , which is distributed as X . Thus

$$F_Z(z) = F_X(z) e^{-ga[1 - F_W(z)]}$$

$$\text{and } E(Z) = \int_0^{\infty} [1 - F_Z(z)] dz.$$

The last two equations can be used to find $E(Z)$, although not easily. Finally

$$s = \frac{ge^{-ga}}{1 + ga + gE(Z)}.$$

B.4 CSMA/CD

Refer to Figure 11 for CSMA/CD (unslotted). Here collisions are detected and transmissions aborted. Again packets are scheduled with a rate g . After an idle period a packet is transmitted. If no packets are transmitted in the next a seconds that packet is successful. After a seconds to clear, the channel returns to idle. A scheduled packet can be transmitted in the first a seconds and will cause a collision. But it will be aborted a seconds after the original transmission. The original transmission will be aborted a seconds after the start of the colliding packet. Thus, as before,

$$s = ge^{-ga} P(\text{channel idle}).$$

$$P(\text{channel idle}) = \frac{1/g}{1/g + a + E(Z)}$$

and $Z =$

$$X, \text{ with prob. } e^{-ga}$$

$$\min\{T_i\} + a, \text{ with prob. } 1 - e^{-ga}.$$

Here we assume, for convenience, that $X \geq 2a$. Solving we get

$$E(Z) = E(X)e^{-ga} + \sum_{k=1}^{\infty} \left(\frac{a}{k+1} + a \right) \frac{(ga)^k}{k!} e^{-ga}$$

$$= E(X)e^{-ga} + \frac{1}{g} [1 - (1+ga)e^{-ga}] + a(1 - e^{-ga})$$

or

$$s = \frac{ge^{-ga}}{1 + gE(X)e^{-ga} + (1+2ga)(1-e^{-ga})}$$

Note than only $E(X)$ appears in the above equation.

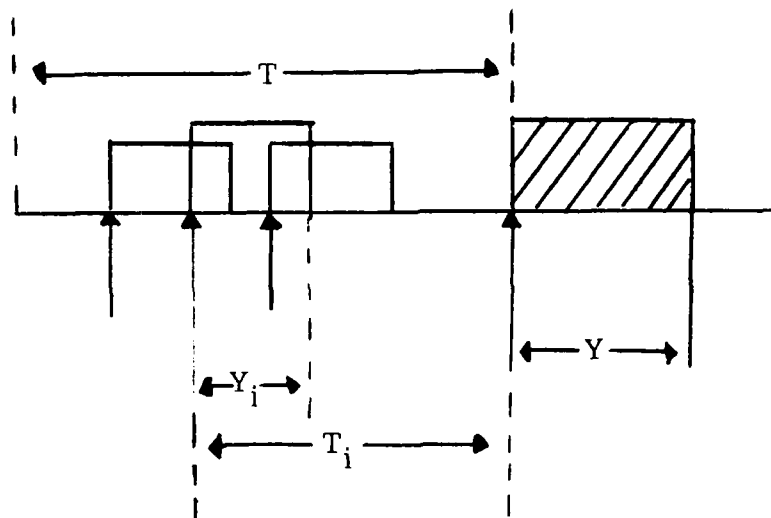


FIGURE 9
PURE ALOHA

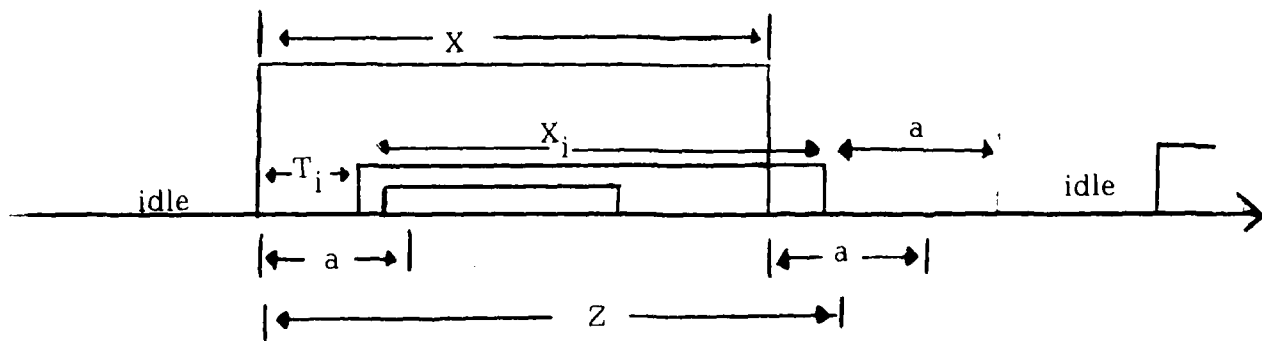


FIGURE 10

CSMA

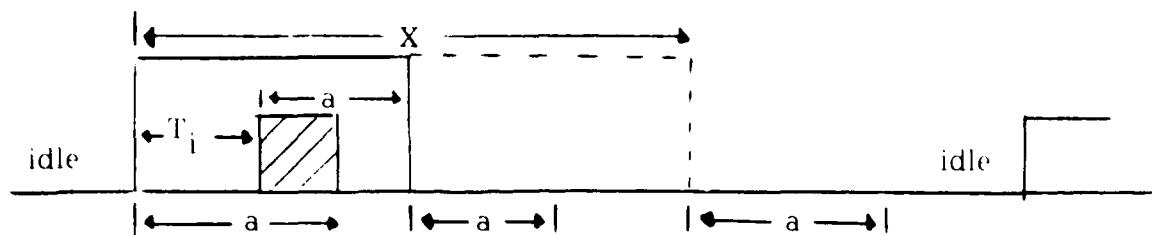


FIGURE 11
CSMA/CD

REFERENCES

5. S. Bellini and F. Borgonovo, "On the Throughput of an ALOHA channel with Variable Packet Lengths," IEEE Transactions on Communications, November 1980.

A.6 Evaluation of Throughput in Multihop Packet Radio

Networks with Complex Topologies

Kershenbaum and Boorstyn

Evaluation of Throughput in Multihop Packet Radio

Networks with Complex Topologies

Aaron Kershenbaum and Robert Boorstyn

Polytechnic Institute of New York

In a previous paper [1] we developed an analytical model of multihop packet radio networks operating under CSMA (Carrier Sense Multiple Access), Poisson arrivals, perfect acknowledgements (i.e. acknowledgements always heard and not taking any channel capacity), perfect capture, and zero propagation delay. This yields a Markov model of the system and the relationship

$$\frac{s_{ij}}{g_{ij}} = P(N_i \cup N_j) \quad (1)$$

where s_{ij} is the required successful rate (in packets per unit of transmission time) of transmissions from node i to node j , g_{ij} is the scheduled rate of traffic from node i to node j , $P(A)$ is the probability that all nodes in set A are idle (and says nothing about other nodes), N_i is the neighborhood of node i (i.e., node i together with all nodes which can hear its transmissions), and N_j is the neighborhood of node j . Some of the above assumptions about the network model can be relaxed to include a more general class of networks, but we use the simple model above for the sake of clarity. It is unlikely that such generalizations would significantly affect the computational procedure described below.

We are given r_{ij} , the required end-to-end traffic from i to j , for all nodes i and j . We assume a routing has been done yielding hop-by-hop requirements s_{ij} . We are also given the connectivity for each

node. Equation 1 states that a transmission will be successfully received once it is scheduled if it is transmitted and if no neighbor of the receiver is transmitting at the time transmission begins. (Our assumption of perfect capture allows us to ignore interference from any subsequent transmissions.)

In our previous work [1] we have shown that the probability of a set of nodes, A , being idle is given by

$$P(A) = \frac{\sum_{I \subset A^c} \prod_{i \in I} G_i}{\sum_{I \subset V} \prod_{i \in I} G_i} \quad (2)$$

where A^c is the complement of A , (i.e. all nodes not in A), I is an independent set (i.e. nodes which all cannot hear each other), and G_i is the total scheduled rate from node i (i.e. G_i is the sum on j of g_{ij}). We refer to the quantities in Equation 2 as sums of products on a set A and denote it by $SP(A)$. We can thus rewrite (1) as

$$\frac{s_{ij}}{g_{ij}} = \frac{SP(V - (N_i \cup N_j))}{SP(V)} = F_{ij}(\vec{G}) \quad (3)$$

where $\vec{G} = (G_1, G_2, \dots, G_N)$ and $G_i = \sum_j g_{ij}$. Then we can write,

$$g_{ij} = \frac{s_{ij}}{F_{ij}(\vec{G})} \quad (4)$$

We then can iteratively compute new estimates of the g_{ij} given current estimates of these quantities and hence the G_i . Our previous experience, reported in [1] showed that if the network can support the given s_{ij} then the iteration will converge if we start with $g_{ij} = s_{ij}$. Note that $g_{ij} \geq s_{ij}$.

Conceptually, then, the problem is solved. One need only set up the expressions for the $F_{ij}(\vec{G})$ and iterate (4) for each i and j until successive estimates for the g_{ij} converge to within some given

tolerance. This was done for small networks and for larger networks with symmetric traffic and connectivity; results using this procedure are reported in [1]-[4].

For larger networks without symmetry, however, several significant problems are encountered. Foremost among these is that, in general, the number of terms in $SP(V)$ grows exponentially with the number of nodes in V . Specifically, the number of terms in $SP(V)$ equals the number of independent sets in V . In the worst case, when all nodes are independent of one another, $SP(V)$ includes a term corresponding to each of the 2^N subsets of V . While such a case corresponds to an unrealistic situation where no nodes can communicate with one another, many real networks are loosely connected and thus contain a very large number of independent sets. Indeed, this number will grow exponentially with N if connectivity (average nodal degree) does not increase with N . This can be seen by examining any regular topology; e.g., a grid.

Even if the number of terms in $SP(V)$ can be controlled, say to grow quadratically with N , there are still several problems. First, $SP(A)$ must be generated not only for $A=V$, but for many other sets as well. In particular, each F_{ij} will have a different $SP(A)$ in the numerator. Also, F_{ij} must be evaluated not once but many times before the iteration converges. Finally, the process of generating the F_{ij} must be automated in order to avoid tediously having to manually input the individual expressions.

We now describe a procedure which overcomes all these problems. We first observe that in the pathological case of all nodes independent, $SP(V)$ can be written as

$$SP(V) = \prod_{i=1}^N (1+G_i) \quad (5)$$

If this expression were expanded directly into a sum of products form it would indeed contain 2^N terms as was remarked above. As it is written, however, only N additions and $N-1$ multiplications are required for its evaluation. While, again, the situation where all nodes are independent of one another is unrealistic, the observation is nevertheless important. It is possible to make the evaluation of $SP(A)$ tractable by grouping together common subexpressions and using the distribution of multiplication over addition to reduce the number of arithmetic operations. This is the basis for the procedure described below.

Every independent set either contains a given node, i , or it does not. If the set contains node i then it does not contain any neighbors of i . We can thus evaluate $SP(A)$ by

$$SP(A) = SP(A-i) + G_i \cdot SP(A-N_i) \quad (6)$$

$$SP(\phi) = 1$$

where i is any member of A .

This recursive expansion of $SP(A)$ can generate up to $2^{|A|}$ terms, where $|A|$ is the cardinality of A , but fewer terms will be generated in general since $A-N_i$ will eventually become ϕ and terminate the recursion. A more dramatic reduction in the number of generated terms will occur if we recognize sets, A , which have already appeared and, hence, for which $SP(A)$ is already known. In this case, the known value of $SP(A)$ is used directly and no further recursive expansion is required. This corresponds exactly to factoring out a common subexpression.

As an example of how this procedure works, consider the network shown in Figure 1. We begin by computing $SP(V)$. The recursive expansion of $SP(V)$ is shown in Figure 2. Each node in Figure 2 corresponds to an expansion using (6). Thus, for example, the root of the tree corresponds to

$$SP(\{12345\}) = SP(\{12345\} - \{1\}) + G_1 \cdot SP(\{12345\} - N_1) \quad (7)$$

where $N_1 = \{123\}$. The G_i represent the same term appearing in (6). In each case we expanded $SP(A)$ about the lowest numbered node in set A . This was done for simplicity. In fact, the selection of which node to expand about is significant and will be discussed later. Nodes followed by an asterisk in Figure 2 correspond to terms which have already been generated and therefore need not be expanded. As can be seen, there are 6 distinct nodes in Figure 2.

The first six rows of Table 1 correspond to the six distinct nodes in Figure 2. The seventh row of Table 1 and the column labeled NEXT will be explained below, as will be the procedure for generating Table 1. The first five columns of the first six rows in Table 1 represent the recursive expansion of $SP(V)$ as shown in Figure 2. For example, Row 1 in Table 1 corresponds to Equation (7) above where Terms 1 and 2 are the appropriate $SP(A)$ and $GMULT = G_1$.

If the column labeled VALUE is used to hold numerical values of SP for the sets given in the column labeled SET, and $GVAL(i)$ contains the current numerical value of G_i , the current estimate of the scheduled rate from node i , then Table 1 can be used to obtain the numerical value of one set given values of others. Specifically, $VALUE(5)$ corresponds to $SP(\phi)$ and is equal to 1 by definition. This value can then be used to obtain $VALUE(4)$:

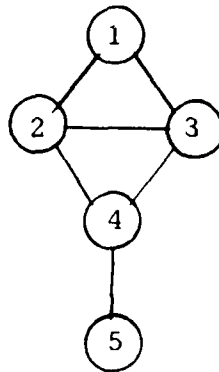


Figure 1 - A multihop network

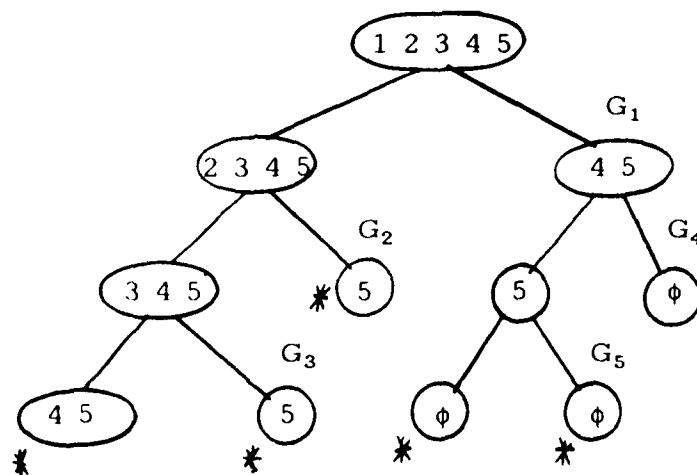


Figure 2 - Recursive expansion of SP(V)

Row	Set	Term 1	Term 2	GMULT	NEXT	VALUE
1	{12345}	2	3	1	0	-
2	{2345}	6	4	2	1	-
3	{45}	4	5	4	6	-
4	{5}	5	5	5	3	-
5	ϕ	-	-	0	7	1
6	{345}	3	4	3	2	-
7	{1}	5	5	1	4	-

Table 1 - Code Table for all $SP(A)$

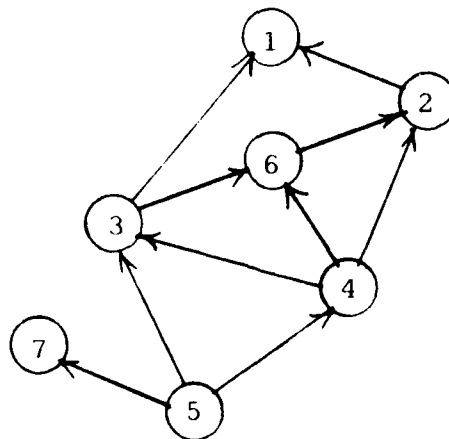


Figure 3 - Precedence graph

$$\begin{aligned}
\text{VALUE}(4) &= \text{VALUE}(\text{TERM1}(4)) + \text{GVAL}(\text{GMULT}(4)) \\
&\quad \cdot \text{VALUE}(\text{TERM2}(4)) \\
&= \text{VALUE}(5) + \text{GVAL}(5) * \text{VALUE}(5) \\
&= 1 + G_5
\end{aligned}$$

which is, of course, $\text{SP}(\{5\})$.

It is necessary that the rows of Table 1 be evaluated in the correct order so that the appropriate VALUE's are available when they appear as terms in larger expressions. While this ordering is not unique, there is a precedence of rows imposed by Table 1. This precedence is shown in Figure 3. An arc leads from node i to node j in Figure 3 if row i must be evaluated before row j . This graph is obtainable directly from Table 1 using the columns labeled ROW, TERM1, and TERM2. Physically, we represent Figure 3 by the outward adjacency of each node (i.e. a list of the endpoints of arcs leading out of it) and the inward degree of each node (the number of arcs leading into it). With this information one can do a topological sort of the nodes (c.f. Knuth[5]), i.e., we sort the nodes by current inward degree, continually reducing the inward degree by eliminating nodes already sorted. Topological sorting is a simple procedure whose running time is proportional to the number of arcs in the precedence graph and which produces a list of nodes in precedence order. In this case, the resulting list would be 5,7,4,3,6,2,1. Note that each row in Table 1 can be dependent upon at most two others and so the number of arcs in the precedence graph, the size of the adjacency list, and the run time of the topological sort are all linear in the number of rows in Table 1.

The column labeled NEXT gives the next row to evaluate as given by the topological sort. Row 5, corresponding to ϕ is of course first; this fact is recorded in a variable called FIRST. Thus, Table 1 allows us to evaluate $SP(V)$ simply by making a single pass through Table 1 and doing one addition and one multiplication per row.

We have thus solved the problems of how to automatically generate a functional expression for $SP(V)$, how to avoid having to reevaluate common subexpressions, and how to simplify the iterative calculation of the G_i . With respect to the last point, note that Table 1 is set up only once. The evaluation of the $SP(A)$ is then straightforward, requiring only K additions and K multiplications, where K is the number of rows in Table 1. Indeed, one generation of Table 1 suffices for the evaluation of the G_i for many values of s_{ij} , which would be useful in analyzing variations in routing and offered traffic.

We require not only $SP(V)$ but also $SP(V-(N_i \cup N_j))$ for all i and j which are neighbors and have $s_{ij} > 0$. We represent all these other $SP(A)$ in Table 1 in exactly the same way as we do $SP(V)$ and we take full advantage of common subexpressions among all $SP(A)$ and $SP(V)$. Thus, for example,

$$SP(V-(N_2 \cup N_3)) = SP(\{5\})$$

which is already in Table 1, so we do not add a row for it. Indeed, as it turns out in this case, the only addition to Table 1 is necessitated by $SP(V-(N_4 \cup N_5)) = SP(\{1\})$. This expands directly in terms of $SP(\phi)$ and hence necessitates the addition of only one row, Row 7, to Table 1. Note that the topological sort and evaluation of VALUE is done for the entire table. Thus, for each iteration on the G_i , only one pass through Table 1 need be made to obtain all necessary $SP(A)$.

The evaluation of F_{ij} in Equation (3) amounts simply to dividing the VALUE in the row corresponding to $SP(V - (N_i \cup N_j))$ by the VALUE in the row corresponding to $SP(V)$. The appropriate row numbers are, of course, recorded in a table once during the procedure which generates Table 1.

The actual procedure for generating Table 1 is now outlined. A stack of sets, A , for which $SP(A)$ is to be recursively expanded is maintained. Initially, the stack contains V , the complete node set. It is convenient to maintain such sets and the neighborhood sets N_i as bit vectors. At each stage, the set, A , at the top of the stack is popped and the sets $A-i^*$ and $A-N_{i^*}$ are formed, where i^* is the most desirable node in A to expand upon. A single ordering of nodes in V is produced according to criteria defined below and is used to determine i^* for all A . Assume therefore that the nodes in the network have been numbered according to their desirability; i^* then corresponds to the leftmost 1 in the bit vector representing A . This value should be recorded along with the set A so that it can be used as the starting point for the search for the best node in the sets $A-i^*$ and $A-N_{i^*}$. This value in fact already appears in the GMULT entry in each row.

Initially, Table 1 contains only one row, corresponding to V . The SET entry in row 1 contains a vector of N 1's. As a set, A , is popped from the top of the stack, its TERM1 and TERM2 entries are set. These should correspond to the rows in Table 1 whose SET entries contain $A-i^*$ and $A-N_{i^*}$. A search of the SET entries currently in Table 1 is made to determine if these sets are already present. If so, the row containing each set is recorded in the TERM1

and TERM2 entries of the current row. If not, a row is added to Table 1 for each new set, the SET and GMULT entries are filled in, and the new sets are added to the stack of sets to be evaluated. In practice, the row number in Table 1 rather than the set itself is placed in the stack in order to avoid having to search for it. Also a binary search tree or AVL tree [6] structure on the SET values should be kept to facilitate the search for existing sets. An alternative to this is to hash these entries as described in [6].

To generate entries corresponding to $SP(V-(N_i \cup N_j))$, these sets are treated exactly like those arising above; i.e. a search is made for them and if necessary they are added to Table 1 and to the stack. The search for i^* in such sets starts at node 1.

We thus see that both the code generation procedure and evaluation procedures are proportional to the number of rows in Table 1. There are other factors such as the search time for existing sets, the search for i^* in each row, and the generation of sets $V-\{N_i \cup N_j\}$. These latter factors are minor, at worst polynomial in the number of nodes. Our primary concern is then with the number of rows in Table 1, which can theoretically grow exponentially with N , the number of nodes.

The number of rows in Table 1 will be kept down if sets arise repeatedly in the expansion process. This is more likely to happen if a consistent ordering of the nodes is used in expanding all sets. This is in fact done, as was mentioned above. If network connectivity is related to distance, a node ordering based on location will result in similar subsets being generated during the expansion process. More generally, any ordering based on connectivity is likely to help.

Alternatively, an ordering based upon nodal degree may be advantageous. Expanding on nodes with large degree will cause $A-N_{i*}$ to shrink rapidly to $\emptyset$ and thereby reduce the number of generated terms. Conversely if node i has no neighbors in A , then $A-i = A-N_i$ and only one subexpression instead of two is created.

Another possibility for controlling the size of Table 1 is to decompose a large network by numbering all the nodes in a cut (separating set) first, then numbering all the nodes on one side of the cut, and finally numbering all the nodes on the other side, as in shown in Figure 4. This guarantees that once expansion of the nodes in the cut (nodes 1 through K) is done, the remaining sum of products will factor since

$$SP(A \cup B) = SP(A) \cdot SP(B) \quad (7)$$

if A and B are completely disconnected from one another. In particular, it can be shown that

$$SP(V) = \sum_{I \subseteq C} [(\prod_{i \in I} G_i) SP(V-N_I)] \quad (8)$$

where C is any subset of V , I is any independent subset of C , and N_I is the union of the neighborhoods of nodes in I .

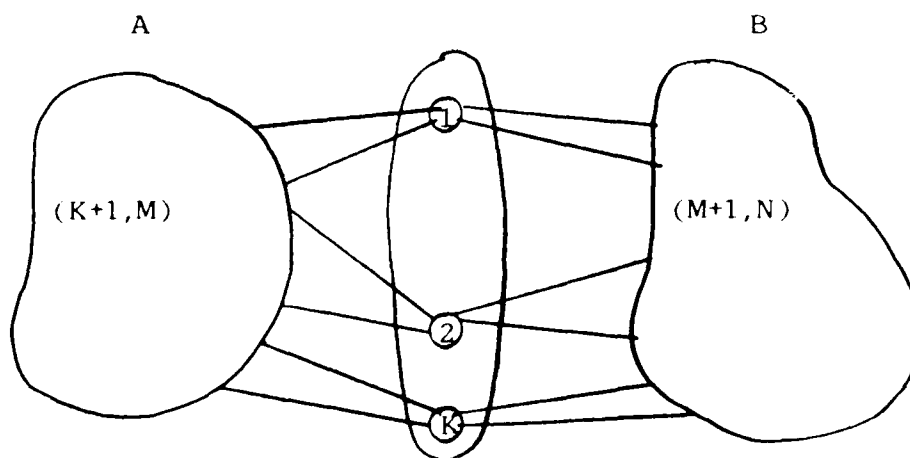


Fig. 4 - Decomposition of a network using a cut

If C is a cut, dividing V into disjoint sets A and B with no links between them, as in Figure 4, then Equation (8) becomes

$$SP(V) = \sum_{C \in \mathcal{C}} \left\{ \prod_{i \in I} G_i \right\} SP(A - N_I) SP(B - N_I) \quad (9)$$

Thus, the sum of products "factors". The numbering scheme will ensure that the nodes in $B - N_I$ are left for last and hence become a common subexpression for all subsets of $SP(A - N_I)$. Thus, the factors in (9) will be evaluated separately and multiplied together at the end. This technique can be applied recursively to decompose very large networks. It appears to have the property that Table 1 will grow at worst exponentially in the size of the cuts.

We have coded a preliminary version of this procedure in order to perform experiments to determine the growth rate of Table 1 as a function of N using various types of connectivity and node orderings. Thus far, the following conclusions and observations have been made:

1. Random X and Y coordinates were generated for N nodes in a unit square. Euclidean distances were computed. A threshold, T , was varied from 0 to 2. Nodes at distance T or less were said to be connected. Experiments for $N=15$ and $N=30$ were run. Nodes were ordered based on the sum of their X and Y coordinates. For $N=15$, Table 1 typically had about 50 rows, for $SP(V)$ only. For $N=30$, Table 1 typically had about 120 rows. The conclusion we draw on the basis of this limited data is that Table 1 does not appear to be growing exponentially. Indeed, it seems to be growing only slightly faster than linearly.

2. For random connectivity and node ordering based on nodal degree (largest first) roughly 80 terms were generated for $N=15$ and roughly 600 terms for $N=30$. These results, based on few runs, are inconclusive.
3. Grid networks (nodes evenly spaced over a unit square) of 25 nodes and varying connectivity were examined using distance based connectivity as in 1 above. Table size varied with connectivity, with the worst case being 130 terms at a nodal degree around 6. Again, the conclusion is that table size does not seem to be growing very rapidly.

We are currently implementing a more complete version of the procedure in order to do more extensive testing.

References

- [1] R. Boorstyn and A. Kershenbaum, "Throughput Analysis of Multihop Packet Radio", Proceedings of IEEE International Conference on Communications, June 1980.
- [2] R. Boorstyn, A. Kershenbaum, and V. Sahin, "A New Acknowledgement Protocol for Analysis of Multihop Packet Radio Networks", Proceedings of IEEE COMPCON, September 1982.
- [3] B. Maglaris, R. Boorstyn, and A. Kershenbaum, "Extensions to the Analysis of Multihop Packet Radio Networks"
- [4] R. Boorstyn, A. Kershenbaum, and V. Sahin, "Throughput Analysis of Multihop Packet Radio Networks," accepted for publication in IEEE Transactions on Communications.
- [5] D. Knuth, Fundamental Algorithms, Addison Wesley, 1973.
- [6] A.V. Aho, J. Hopcroft, and J. Ullman, The Design and Analysis of Computer Algorithms, Addison Wesley, 1974.

A.7 Optimal Transceiver Deployment in Multihop Packet

Radio Networks with Capture

Maglaris and Kaminer

NSF Final Report, October 1981

AD-A131 915

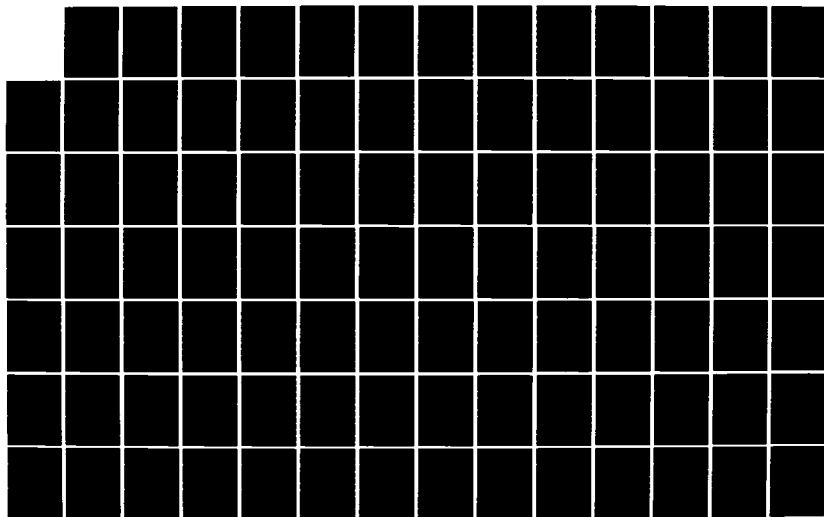
RESEARCH IN NETWORK MANAGEMENT TECHNIQUES FOR TACTICAL
DATA COMMUNICATION. (U) POLYTECHNIC INST OF NEW YORK
BROOKLYN R R BOORSTYN ET AL. 01 SEP 82 CECOM-80-0579-F
DAAK80-80-K-0579

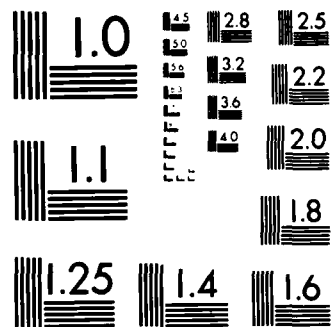
2/5

UNCLASSIFIED

F/G 17/2.1

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

NETWORK ANALYSIS CORPORATION

TECHNICAL NOTE - 277

**OPTIMAL TRANSCETVER DEPLOYMENT IN MULTIHOP PACKET-RADIO NETWORKS
WITH CAPTURE**

June 1981

Prepared by:

- Basil Maglaris
- Gregory Kaminer

This work was supported by the National Science Foundation under grant
ENG.7908120.

1. INTRODUCTION

Packet radio has recently emerged as a viable technology for both fixed and mobile computer communications. It utilizes packet-switched communications, which became highly important in computer communication networks, over broadcast radio channels.

The original packet switching concept was based on point-to-point data circuits interconnecting network packet switches. With technology advances in satellite and ground radio broadcasting, packet radio networks have become an area of major interest [6]. During the early 1970's, the ALOHA project at the University of Hawaii demonstrated the feasibility of using packet broadcasting for bursty computer traffic [1].

In a broadcast radio network, channels are shared by a set of nodes and cannot be dedicated to specific pairs. Numerous studies ([1], [6], [7], [5], and [9]) have shown that "fixed capacity assignments" are wasteful for many applications compared to "random access schemes" which result in a dynamic sharing of channel capacity without centralized control. These studies focused on a single-hop network, whereby Packet Radio Units (PRU's) attempt to transmit data to a single central station. In conjunction with the fact that there is no control over access to the channel, destructive errors result when several packets are received simultaneously or "collide." These collisions reduce the channel throughput under random access.

The analysis of multihop packet radio networks involving packet routing via relays, is extremely difficult. So one must use simulation [9], [10] or study simple models to measure network performance, mainly the network capacity or network throughput [8], [14], and [15]. The throughput of multihop slotted ALOHA network was studied recently, [12], and it was found that one of the most important factors affecting it, is the transmission radius (or transmission power) used by the nodes. To increase the throughput, one may even want to reduce the transmission power to a level lower than the maximum possible.

Techniques which are used to increase the network throughput include Carrier Sensing Multiple Access (CSMA) [13], and resolution of collisions via packet capture mechanisms. The effect of capture in increasing the network throughput was studied in [2] and [11] for single-hop networks. In [4] the effect of capture on the throughput was investigated for multiple transmitters and receivers in a slotted ALOHA environment.

The problem of obtaining exact expressions for throughput and blocking probabilities in multihop packet radio networks using CSMA access method was considered in [3]. Exact results were obtained assuming exponential packet lengths for general topologies. This analysis has been extended to constant packet length for selected simple topologies.

In this work, we consider multihop packet radio networks using slotted and unslotted ALOHA access methods with and without capture. We consider a set of identical PRU's randomly (uniformly) distributed in the plane. We find, via analytic techniques and computer experimentation, that the optimal density of radio units which maximizes the throughput of the network is the same for all access methods considered and corresponds to the previous found value for the slotted ALOHA case without capture [12].

2. SLOTTED CASE MODEL

The nodes of the network are assumed to be uniformly distributed with density λ .

Two access methods are considered:

- No capture (slotted ALOHA).
- Capture (closest neighbor).

We consider a multihop packet radio network, with transceivers uniformly distributed in a large geographical area. Packet Radio Units (PRU's) have equal transmission range. Any node within the circle of radius r around a transmitting node will hear the transmission and can respond to it. Each node always has a message to send and will do so whenever permitted. (see Figure 1).

Let:

A_i - Range of reception for terminal i .

N_i - Number of terminals in the area A_i

G - Probability of a packet transmission during a slot by a terminal.

q_i - Probability of successful reception of a packet from a transmitting terminal of the area A_i .

$$q_i = \Pr \{ \text{a neighbor transmits successfully to } i / A_i \} = P_1 \cdot P_2 \cdot P_3;$$

where P_1 - probability that i does not transmit.

$$P_1 = 1 - G$$

P_2 - Probability that a neighbor from area A_i transmits to i and i can receive the packet.

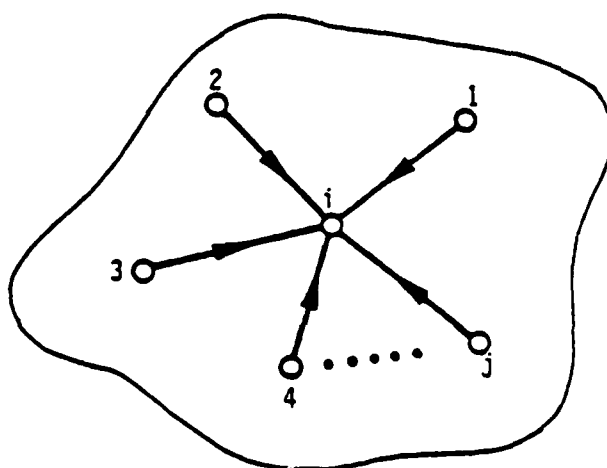


FIGURE 1: PRU i RECEIVES FROM j PRU'S WITHIN ITS REGION

$$P_2 = \binom{N_i}{1} G (1 - G)^{N_i - 1} \quad \text{- no capture (one transmission)}$$

$$P_2 = 1 - (1 - G)^{N_i} \quad \text{- capture (one or more attempts made)}$$

P_3 - Probability that captured packet is destined to i , while source has N_i options to address.

$$P_3 = \frac{1}{N_i}$$

Therefore,

a. No capture (slotted ALOHA):

$$q_i = G(1 - G)^{N_i}$$

b. Capture:

$$q_i = \frac{1 - G}{N_i} \left[1 - (1 - G)^{N_i} \right]$$

Assuming that the number of terminals in an area A is a Poisson random variable with density λ , we have that S (the throughput to a node) is given by:

$$S = \sum_{i=1}^{\infty} \frac{(\lambda A)^i e^{-\lambda A}}{i!}$$

$\lambda A = N$ - Average degree.

Therefore,

a. No capture:

$$S = Ge^{-NG} - Ge^{-N}$$

b. Capture:

$$S = (1 - G) e^{-N} \sum_{i=1}^{\infty} \frac{N^i}{i! i} \left[1 - (1 - G)^i \right]$$

The maximum throughput, will be achieved with $G = G_{opt}$, by setting:

$$\frac{dS}{dG} = 0$$

The total network throughput S_{net} , will be:

$$S_{net} = n \cdot S$$

Where n is the total number of nodes. The average number of hops in an end-to-end path $\bar{h}$ is given in [12] :

$$\bar{h} = \frac{128}{45\pi} \left(\frac{n}{N} \right)^{\frac{1}{2}} \frac{1}{1 + e^{-N} - \int_{-1}^1 e^{-\frac{N}{\pi}} \left[\cos^{-1}(t) - t \sqrt{1-t^2} \right] dt} ;$$

Therefore, the end-to-end throughput γ is:

$$\gamma = \frac{S_{net}}{\bar{h}} ;$$

To find G_{opt} :

a. No capture: it is given in [12] .

b. Capture:

$$S = (1 - G) e^{-N} \sum_{i=1}^{\infty} \frac{N^i}{i! i} \left[1 - (1 - G)^i \right] ;$$

$$\frac{dS}{dG} = e^{-N} \sum_i \frac{N^i}{i! i} \left[1 - (1-G)^i \right] + (1-G) e^{-N} \sum_i \frac{N^i}{i! i} i (1-G)^{i-1} = 0$$

or

$$e^{N(1-G)} = 1 + \sum_i \frac{N^i}{i! i} \left[1 - (1-G)^i \right]$$

Therefore,

$$G_{opt} = 1 - \frac{1}{N} \ln \left[1 + \sum_i \frac{N^i}{i! i} \left[1 - (1-G_{opt})^i \right] \right]$$

3. UNSLOTTED CASE MODEL

For the unslotted case, the following schemes are considered:

- Pure ALOHA (All PRU's transmit at random. Collided receptions result into loss of all colliding packets).
- Perfect Capture (In case of collision, the receiver keeps tracking the packet it received first. This is possible under a broad range of spread spectrum technologies).
- Half-Amplitude Capture (In case of collision, the receiver will listen to the closest transmitter).

As in the unslotted case, we assume that PRU's are randomly located in the network area. The rate of new packets and total rate of packets (including retransmissions) between PRU's separated by a distance d , are given by S_d and G_d respectively.

The following areas are defined (see Figure 2):

- A_1 - Inside the contour D-E-F-C-H-D.
- A_2 - Inside the contour D-K-C-F-G-E-D.
- A_3 - Inside the contour D-J-F-G-E.
- A_4 - Inside the contour E-G-F-A-E.
- A_5 - Inside the contour D-E-A-F-C-H-D.

These areas can be expressed as functions of communication radius (r) and variable angle θ_1 :

$$A_1 = \pi r^2$$

$$A_2 = \pi r^2 - \frac{r^2}{2} (2 \theta_1 - \sin 2 \theta_1) \cdot 2$$

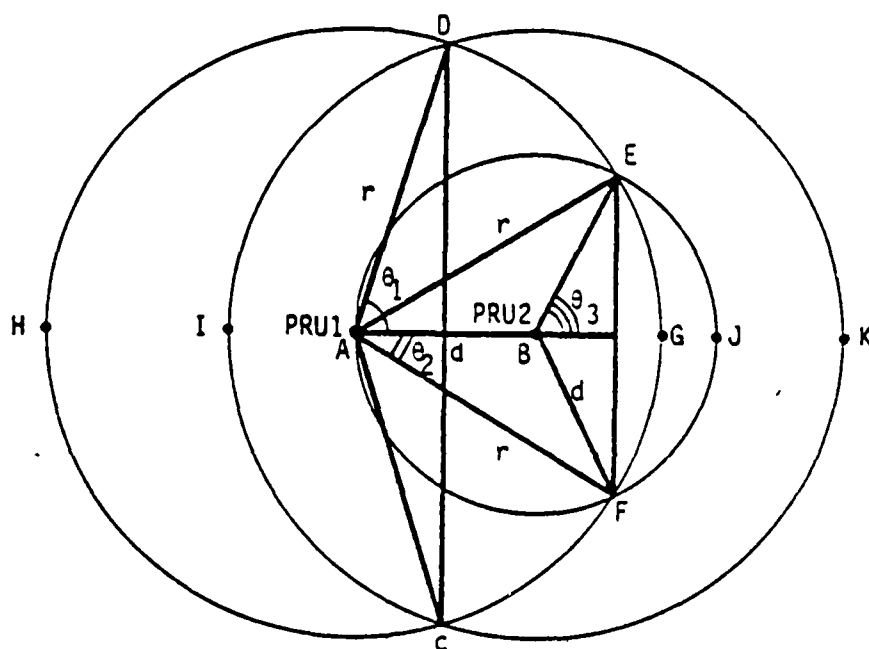


FIGURE 2: REGIONS DEFINED WITH RESPECT TO PRU1 - PRU2 TRANSMISSIONS

$$\begin{aligned}
 A_3 &= \begin{cases} 0, \text{ if } d \leq r/2 \text{ or } \theta_1 \geq \cos^{-1} \frac{1}{4} \\ \frac{d^2}{2} (2\theta_3 - \sin 2\theta_3) - \frac{r^2}{2} (2\theta_2 - \sin 2\theta_2) = \\ r^2 \left\{ 2 \cos^2 \theta_1 \left[4 \cos^{-1} \left(\frac{1}{4 \cos \theta_1} \right) \right] - \sin \left[4 \cos^{-1} \left(\frac{1}{4 \cos \theta_1} \right) \right] \right. \\ \left. - \cos^{-1} \frac{1}{4 \cos \theta_1} + \frac{1}{2} \sin \left[2 \cos^{-1} \left(\frac{1}{4 \cos \theta_1} \right) \right] \right\}; \end{cases}
 \end{aligned}$$

Since,

$$d = 2r \cos \theta_1;$$

$$\theta_2 = \cos^{-1} \frac{r}{2d}; \text{ and } \theta_3 = 2\theta_2;$$

$$A_4 = \begin{cases} \pi d^2 = 4 \pi r^2 \cos^2 \theta_1, \text{ if } \theta_1 \geq \cos^{-1} \frac{1}{4} \\ \pi d^2 - A_3 = \pi r^2 \left[4 \cos^2 \theta_1 - \frac{1}{\pi} \frac{A_3}{r^2} \right]; \end{cases}$$

$$A_5 = \pi r^2 - A_4 = \begin{cases} \pi r^2 \left[1 - 4 \cos^2 \theta_1 \right]; \text{ if } \theta_1 \geq \cos^{-1} \frac{1}{4} \\ \pi r^2 \left[1 - 4 \cos^2 \theta_1 + \frac{1}{\pi} \frac{A_3}{r^2} \right]; \end{cases}$$

Therefore, the joined probability of the number of nodes in the areas above is:

$$P(n_1, n_2) = \frac{(\lambda A_1)^{n_1} (\lambda A_2)^{n_2}}{n_1! n_2!} e^{-\lambda(A_1 + A_2)}$$

$$P(n_1, n_3) = \frac{(\lambda A_1)^{n_1} (\lambda A_3)^{n_3}}{n_1! n_3!} e^{-\lambda(A_1 + A_3)}$$

$$P(n_3, n_4, n_5) = \frac{(\lambda A_3)^{n_3} (\lambda A_4)^{n_4} (\lambda A_5)^{n_5}}{n_3! n_4! n_5!} e^{-\lambda(A_3 + A_4 + A_5)}$$

$$P(n_4, n_5) = \frac{(\lambda A_4)^{n_4} (\lambda A_5)^{n_5}}{n_4! n_5!} e^{-\lambda(A_4 + A_5)}$$

Where n_i = population of i^{th} area.

Using the average degree definition:

$$N = \lambda \pi r^2$$

we obtain:

$$P(n_1, n_2) = \frac{N^{(n_1 + n_2)}}{n_1! n_2!} \left(1 - \frac{2 \Theta_1}{\pi} + \frac{\sin 2 \Theta_1}{\pi} \right)^{n_2}$$

$$\cdot \exp \left\{ N \left(-2 + \frac{2 \Theta_1}{\pi} - \frac{\sin 2 \Theta_1}{\pi} \right) \right\};$$

$$P(n_1, n_3) = \frac{N^{(n_1 + n_3)}}{n_1! n_3! \pi} \left(\frac{A_3}{r^2} \right)^{n_3} \cdot \exp \left\{ -N \left(1 + \frac{1}{\pi} \frac{A_3}{r^2} \right) \right\};$$

$$P(n_3, n_4, n_5) = \frac{N^{(n_3 + n_4 + n_5)}}{n_3! n_4! n_5!} \frac{1}{\pi} \left(\frac{A_3}{r^2} \right)^{n_3} \left[4 \cos^2 \Theta_1 - \frac{1}{\pi} \left(\frac{A_3}{r^2} \right) \right]^{n_4}$$

$$\cdot \left(1 - 4 \cos^2 \Theta_1 + \frac{1}{\pi} \frac{A_3}{r^2} \right)^{n_5} \cdot \exp \left\{ -N \left(1 + \frac{1}{\pi} \frac{A_3}{r^2} \right) \right\};$$

$$P(n_4, n_5) = \frac{N^{(n_4 + n_5)}}{n_4! n_5!} \left(4 \cos^2 \theta_1\right)^{n_4} \cdot \left(1 - 4 \cos^2 \theta_1\right)^{n_5} \cdot e^{-N};$$

In order to compute the throughput (successful transmission rate S_d) between PRU 1 to PRU 2 separated by a distance d , the following observation is made: For PRU 2 to receive correctly a packet (of duration normalized to 1), the packet should be transmitted from:

- Area ($A_1 UA_2$) during $\{t - 1, t + 1\}$ for pure ALOHA, (t is the time of packet transmission).
- Area ($A_1 UA_2$) during $\{t - 1, t\}$ for perfect capture.
- Area ($A_3 UA_4 UA_5$) during $\{t - 1, t + 1\}$ for half amplitude capture.

The probability of successful transmission S_d/G_d , will be given as follows:

Pure ALOHA:

$$\frac{S_d}{G_d} = \sum_{\substack{n_1 \geq 2 \\ n_2 \geq 0}} \frac{1}{n_1} e^{-2G(n_1 + n_2)} P(n_1, n_2)$$

Perfect Capture:

$$\frac{S_d}{G_d} = \sum_{\substack{n_1 \geq 2 \\ n_2 \geq 0}} \frac{1}{n_1} e^{-G(n_1 + n_2)} P(n_1, n_2)$$

Half Amplitude Capture:

$$\frac{S_d}{G_d} = \sum_{\substack{n_4 \geq 2 \\ n_5 \geq 0 \\ n_3 \geq 0}} \frac{1}{n_4 + n_5} e^{-2G(n_4 + n_3)} P(n_3, n_4, n_5);$$

if A_3 exists.

$$\frac{S_d}{G_d} = \sum_{\substack{n_4 \geq 2 \\ n_5 \geq 0}} \frac{1}{n_4 + n_5} e^{-2G n_4} P(n_4, n_5);$$

if $A_3 = 0$.

Using the expression for probabilities, transformation $d \rightarrow \theta_1$ (see below), and unconditioning with respect to θ_1 :

Pure Aloha:

$$S = \sum_{\substack{n_1 \geq 2 \\ n_2 \geq 0}} G e^{-2G(n_1+n_2)} \frac{N^{(n_1+n_2)}}{n_1! n_2! n_1} \int_{\pi/3}^{\pi/2} 2 \sin \theta_1 \left(1 - \frac{2\theta_1}{\pi} + \frac{\sin 2\theta_1}{\pi} \right)^{n_2} \\ \cdot \exp \left\{ N \left(-2 + \frac{2\theta_1}{\pi} - \frac{\sin 2\theta_1}{\pi} \right) \right\} d\theta_1;$$

Perfect Capture:

$$S = \sum_{\substack{n_1 \geq 2 \\ n_2 \geq 0}} G^{-G(n_1+n_2)} \frac{N^{(n_1+n_2)}}{n_1! n_2! n_1} \int_{\pi/3}^{\pi/2} 2 \sin \theta_1 \left(1 - \frac{2\theta_1}{\pi} + \frac{\sin 2\theta_1}{\pi} \right)^{n_3} \\ \cdot \exp \left\{ N \left(-2 + \frac{2\theta_1}{\pi} - \frac{\sin 2\theta_1}{\pi} \right) \right\} d\theta_1;$$

Half-Amplitude Capture:

$$S = \sum_{\substack{n_4 \geq 2 \\ n_3, n_5 \geq 0}} G e^{-2G(n_3+n_4)} \frac{N^{(n_3+n_4+n_5)}}{n_3! n_4! n_5! (n_4+n_5)} .$$

$$\cos^{-1} \frac{1}{4} = 1.32$$

$$\cdot \int 2 \sin \theta_1 \left(\frac{1}{\pi} \frac{A_3}{r^2} \right)^{n_3} \left(4 \cos^2 \theta_1 - \frac{1}{\pi} \frac{A_3}{r^2} \right)^{n_4} \left(1 - 4 \cos^2 \theta_1 + \frac{1}{\pi} \frac{A_3}{r^2} \right)$$

$$\frac{\pi}{3} = 1.043$$

$$\cdot \exp \left\{ -N \left(1 + \frac{1}{\pi} \frac{A_3}{r^2} \right) \right\} d\theta_1 +$$

$$+ \sum_{\substack{n_4 \geq 2 \\ n_5 \geq 0}} G e^{-2Gn_4} \frac{N^{(n_4+n_5)}}{n_4! n_5! (n_4+n_5)} \int_{\cos^{-1} \frac{1}{4} = 1.32}^{\pi/2 = 1.571} 2 \sin \theta_1 \left(4 \cos^2 \theta_1 \right)^{n_4} .$$

$$\cdot (1 - 4 \cos^2 \theta_1)^{n_5} e^{-N} d\theta_1;$$

To find G_{opt} , we set:

$$\frac{dS}{dG} = 0;$$

This equation is solved numerically using Newton's method:

$$G^{(i+1)} = G^{(i)} - \frac{f(G^{(i)})}{f'(G^{(i)})} ;$$

for given error value allowed.

Using h , [12], we obtain the throughput:

$$\gamma = \frac{S_{net}}{h} ; \text{ where } S_{net} = n \cdot S$$

Transformation $d \longrightarrow \theta_1$

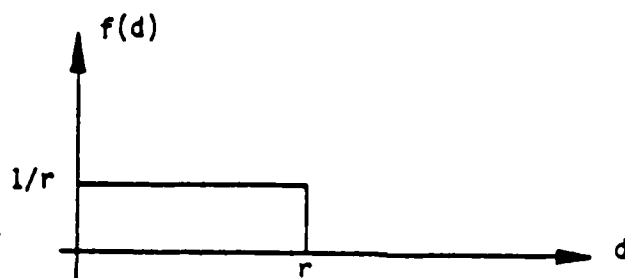
Since d is uniformly distributed (see Figure 3) its distribution will be:

$$f_d(d) = \frac{1}{r} ;$$

$$f_{\theta_1}(\theta_1) = \frac{f_d(d)}{|\theta_1'|} = \frac{f_d(2r \cos \theta_1)}{\left| -\frac{1}{2r \sqrt{1 - \frac{d^2}{4r^2}}} \right|} =$$

$$= \frac{1/r}{\frac{1}{2r \sqrt{1 - \frac{4r^2 \cos^2 \theta_1}{4r^2}}}} = 2 \sin \theta_1 ;$$

$$\text{for } \frac{\pi}{3} \leq \theta_1 \leq \frac{\pi}{2} ;$$



$$0 \leq d \leq r$$

$$d = 2r \cos \theta_1$$

FIGURE 3: PROBABILITY DENSITY FUNCTION OF THE DISTANCE BETWEEN PRU1 - PRU2

4. NUMERICAL RESULTS

The analyses described above were implemented in computer programs in order to compute the maximum throughput for various average degree values. These programs for the unslotted cases involve considerable computational complexities and excessive run times. We have been, however, able to obtain the basic results which are portrayed in Figure 4.

In Figure 4 we show the normalized throughput $\frac{\gamma}{\sqrt{n}}$ as function of the average degree N . For slotted capture case, the value of N , which maximizes the throughput, is 9 or 10, at which point the optimal network throughput is $0.19 \sqrt{n}$. The maximizing value of N for the four other cases is approximately 6, which is the same result as in [12].

Note that the significance of the optimal degree can be interpreted as what should be the density of PRU's, in a deployment scenario in order to achieve maximum throughput.

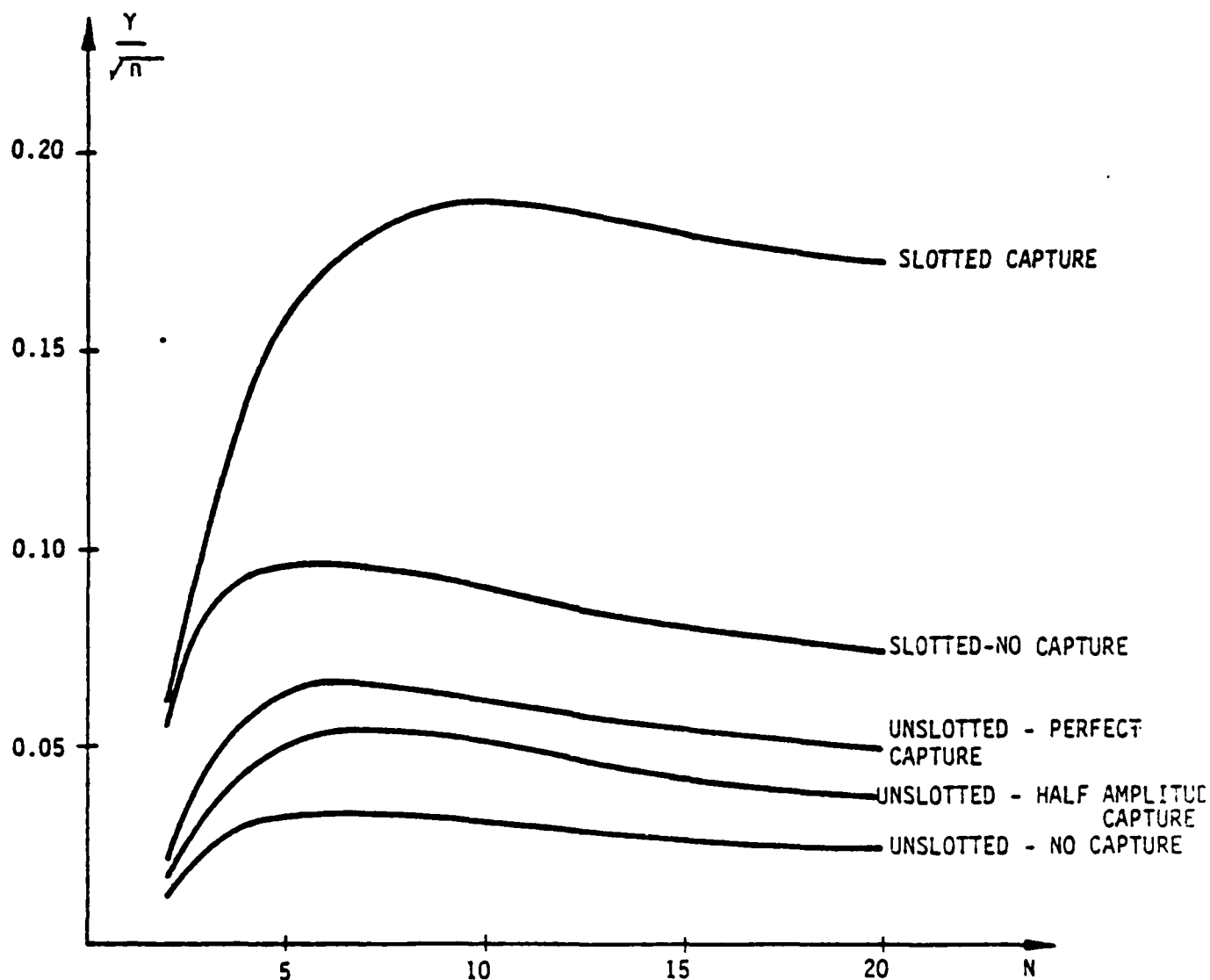


FIGURE 4

5. FURTHER RESEARCH

The approximate nature of the models studied was justified from the need to obtain a tractable solution to the optimal density of radio units in a deployment scenario. The following extensions can be performed using similar assumptions on the random topology and disregarding boundary effects:

- Model Carrier Sensing Multiple Access (CSMA) with and without capture. No exact models for CSMA have been so far obtained, due to the non-Markovian nature of the resulting packet transmission process (see [3]). Approximate methods can be devised or additional assumptions on exponential packet lengths can force a Markovian birth-death approach as in [3].
- Model flood routing, whereby every PRU relays all received packets, regardless of their ultimate destination.

Further research should address the impact of specific topology and routing instead of the random dispersed PRU's over an infinite area. More specifically, in [3], Boorstyn and Kershenbaum proposed a methodology for modeling CSMA packet radio networks with or without capture, under the assumption of exponential packet length. The generalization of the method to fixed or general packet length distribution is still an open issue.

Other areas of interest are:

1. Flow control techniques to achieve maximum throughput.
2. Feasibility of using packet radio networks for packetized voice.

References

1. Abramson, N., "The ALOHA System - Another Alternative for Computer Communications," AFIPS Conference Proceedings, Vol. 37, Fall Joint Computer Conference, Las Vegas, November 1970, pp. 281-285.
2. Abramson, N., "The Throughput of Packet Broadcast Channels," IEEE Transactions on Communications, Vol. COM-25, January 1977, pp. 117-128.
3. Boorstyn, R.R. and A. Kershenbaum, "Throughput Analysis of Multihop Packet Radio."
4. Fratta, L. and D. Sant, "Some Models of Packet Radio Networks with Capture," Proceedings of ICC-80, Atlanta, Georgia, October 1980, pp. 155-161.
5. Gitman, I., R. Van Slyke, and H. Frank, "Routing in Packet-Switching Broadcast Radio Networks," IEEE Transactions on Communications, Vol. COM-24, August 1976, pp. 926-930.
6. Kahn, R.E., "The Organization of Computer Resources into a Packet Radio Network," IEEE Transactions on Communications, Vol. COM-25, January 1977, pp. 169-178.
7. Kleinrock, L. and F. Tobagi, "Packet Switching in Radio Channels," Parts I and II, IEEE Transactions on Communications, Vol. COM-23, December 1975, pp. 1400-1433.
8. Kleinrock, L., "On Giant Stepping in Packet Radio Networks," UCLA Packet Radio Temporary Note #5, PRT 136, March 1975.
9. Network Analysis Corporation, "Sixth Semi-Annual Technical Report: Local, Regional, and Large Scale Integrated Networks: Vol. 2 - Recent Advances in Ground Packet Radio Systems," February 1976 (NTIS Order #AD/A035950).
10. Network Analysis Corporation, "Packet Radio Deployment Study," performed for the USA CORADCOM, Final Report, April 1980.

11. Roberts, L.G., "ALOHA Packet Systems With and Without Slots and Capture," Computer Communication Review, Vol. 5, No. 2, April 1975, pp. 28-42.
12. Silvester, J., "On the Spatial Capacity of Packet Radio Networks," Ph.D Dissertation, Computer Science Department, UCLA, May 1980.
13. Tobagi, F. and L. Kleinrock, "Packet Switching in Radio Channels: Part II - The Hidden Terminal Problem in CSMA and the Busy Tone Solution," IEEE Trans. on Communications, Vol. COM-23, December 1975, pp. 1417-1433.
14. Tobagi, F., "On the Performance Analysis of Multihop Packet Radio Systems: Parts I - IV," Packet Radio Temporary Note #246-249, Computer Science Department, UCLA, 1978.
15. Yemini, Y. and L. Kleinrock, "On a General Rule for Access Control, or Silence is Golden," Flow Control in Computer Networks, Proceedings of the International Symposium on Flow Control in Computer Networks, Versailles, February 1979 (North Holland, Amsterdam, 1979), pp. 335-347.

B. Adaptive Routing

B.1 A Technique for Adaptive Routing in Networks

IEEE Transactions on Communications, April 1981

Boorstyn and Livne

A Technique for Adaptive Routing in Networks

ROBERT R. BOORSTYN, MEMBER, IEEE, AND ADAM LIVNE, MEMBER, IEEE

(Invited Paper)

Abstract—A two-level adaptive routing scheme for packet-switched computer communication networks is proposed and investigated. The first level is quasi-static and based on the global network status. The second level is dynamic with decisions being made at each node in an attempt to obtain the savings in average delay predicted by a multiserver model of the node. Simulations confirm the predicted improvement.

I. INTRODUCTION

MOST adaptive routing schemes perform about as well as nonadaptive schemes when evaluated in a fixed environment [1]. However, they do adapt to changes in network topology and input statistics. The reason they do not show a significant improvement is that they are actually quasi-static in that they sense and respond to the above changes slowly. Their goal is to select good paths [2].

We focus our attention on the node, which may be viewed as a multiple server queueing system. Most adaptive routing schemes operate the node as a collection of single server queues. If the node is operated as a multiple server queue, an advantage at the node of a factor approximately equal to the number of servers (outgoing branches) can be obtained. However, the control over good paths may be lost. We show that we can simultaneously obtain both good paths and improved node performance. These improvements in time delay exist for the network as well. Consequently, network bandwidth can be significantly reduced. Furthermore, the approach lends itself to analysis. We also discuss limits and extensions.

Many computer communication networks (ARPANET, TELENET, TYMNET, etc.) use dynamic routing schemes to compensate for input traffic variations, to respond to changes in topology, and to take advantage of temporary changes in loading in different paths. During the design of a network, they are replaced by analytically tractable nondynamic (static) schemes. We present here a routing scheme for which we have been able to derive approximate analytical models. Furthermore, we can establish the efficiency of this scheme, especially in heavily loaded situations.

A typical static routing scheme would operate as follows. Consider as separate commodities the message originating at a particular node and destined for a second node in the network. The static routing scheme would specify the optimum propor-

tion of traffic to be routed over each path. Efficient algorithms exist for design of this type of routing [3].

We can identify one particular problem with this approach. Although good paths are indeed found, any node essentially operates as a collection of single server queues—one queue for each outgoing branch. Considering the node as a queue with several potential servers, this is not an efficient manner of operation. Indeed, if a node had k outgoing branches and were operated as a queue with k servers, then the time delay would be reduced by a factor of approximately k . Conversely, the throughput can be increased.

If the node were operated as suggested above, messages would wander aimlessly through the network and the total performance would be abysmal. Our approach is to retain the good paths for commodities and yet still get the benefit of the faster performance at the node.

Briefly our scheme is as follows [4]. Consider a node as a single queue with several servers (output channels). For a particular commodity, i.e., a message with a certain destination, the use of some of these servers would cause the messages to be sent along "bad" paths—either too long or too congested. Thus, for each commodity and at each node we specify a subset of the output channels as allowable and permit the message to use any allowable channel according to some discipline. Each commodity appearing at the node has its own allowable set of channels. These restrictions force messages to use "good" paths. It has already been noted [5] that an advantage can be obtained if a selection between relatively good paths is not forced, and the adaptive routing scheme continues to allow each of these to be used.

The assignment of allowable branches at each node for each commodity is one level of our adaptive routing scheme. These assignments are based on essentially global information of topology, flows, and long term averages, and may be adaptive in a quasi-static way. The second, and truly dynamic, level of our routing strategy is local and involves the queue discipline at each node. As an example, consider a node with two outgoing channels (servers). All message commodities fall into three classes. Two of these must use only one of the servers and have no choice. Messages join the appropriate queue and are served in turn. The third class of messages may use either of the two servers and join a third queue.

Another strategy at a node is to give priority to the messages that have no choice, i.e., are dedicated to one of the servers. Yet another strategy is to allow messages in the third category (nondedicated) to join the shorter of the two dedicated queues. We have evaluated the performance of both strategies, and other similar strategies, for this simple two server model, for more complex node models, and for simple

Manuscript received June 18, 1980; revised October 20, 1980. This work was supported in part by U.S. Army CORADCOM under Task H-9-2513 of the Post Doctoral Program, RADC. This paper was presented at the National Telecommunications Conference, Dallas, TX, December 1976.

R. R. Boorstyn is with the Department of Electrical Engineering, Polytechnic Institute of New York, Brooklyn, NY 11201.

A. Livne is with the Scientific Department, Ministry of Defence, Tel-Aviv, Israel.

networks. The measure of performance was the average time delay for all messages traversing the node.

II. NETWORK MODEL

We consider a network, actually the backbone of a distributed (store-and-forward) packet-switched data communications network, as a collection of nodes connected by full duplex links—comprising the topology. The input traffic is described by a matrix of rates (packets/second) of traffic originating at one node and destined for another. Index the nodes by $1, \dots, N$ and denote the links between nodes i and j by (i, j) . Alternately, index the links by $1, \dots, L$. (We assume N nodes and L links.) Denote the input traffic (from source i to destination j) by γ_{ij} (packets/second). Assume (for simplicity) that the average length of all packets is $1/\mu$ bits. Assume further that all input streams, for each of the $N(N-1)$ node pairs, are independent and Poisson. Let the link capacities be given by C_l .

A nonadaptive routing is given by a set of paths for each commodity (node-pair traffic) and the proportion of input traffic using each path. From this, the flow in each link in each direction, $\lambda_{(i,j)} = \lambda_l$ (packets/second), can be found. Note that $\gamma = \sum_{i,j} \gamma_{i,j}$ is the network throughput, or total offered load, and $\lambda = \sum_{i,j} \lambda_{(i,j)}$ is the total internal traffic. Furthermore, note that $\bar{l} = \lambda/\gamma \geq 1$ is the average path length in the network.

We assume exponential packet lengths. We further make the "independence" assumption [6] that at each node in a path, a new random assignment of the packet length, with the same distribution, is made. This idealization leads to the property that each node can be modeled independently as simple $M/M/1$ or $M/M/k$ queues. It has been shown that this approximation yields valid results for many network situations [6]. Some care must be taken in situations, such as adaptive routing, where dependence between nodes is a key factor. We will ignore this warning until the end, when we comment upon its implications, and assume the above type of independence at each node.

Many streams of traffic converge at a node—from incoming network links and from outside the network. Some of this traffic is destined for this node, leaves the network, and will not concern us at this node. The remaining traffic must be routed to an appropriate outgoing link. A nonadaptive strategy decides what proportion of such traffic uses each link on a commodity-by-commodity basis. Thus, each link can be modeled as an $M/M/1$ queue and has link average time delay $T_l = 1/(\mu C_l - \lambda_l)$. The average time delay in the network is $T = (\sum_l \lambda_l T_l)/\gamma$. If we define $T_{link} = (\sum_l \lambda_l T_l)/\lambda$ as the time delay of a typical link, then $T = \bar{l} T_{link}$.

III. NODE MODEL

A node in a network has several outgoing links, say k ; it is said to be of degree k . In nonadaptive routing, the node operates as a collection of single server queues which we can model as $M/M/1$ queues. Thus the time delay for link l is $T_l = 1/(\mu C_l - \lambda_l)$. The time delay for node n is $T_{(n)} = \sum_l \lambda_{n,l} T_{(n,l)}/\lambda_{(n)}$ where $\lambda_{(n)} = \sum_l \lambda_{n,l}$ is the rate of traffic

that node n sends onto the network. The network time delay is $T = \sum_n \lambda_{(n)} T_{(n)}/\gamma$ which gives the same result as above, but expressed in terms of nodal delay.

Ignoring routing for a moment, the node can be viewed as a multiple server queue (with k servers). If we assume all $C_l = C$ at this node, and if all packets join one large queue and are served by any available server, then we have an $M/M/k$ queue with average waiting time at node n given by

$$W_{(n)} = \frac{1}{\mu C} \frac{P_B}{k} \frac{1}{1-\rho} \quad (1)$$

where $\rho = \lambda_{(n)}/\mu k C$ and P_B is the probability that all servers are busy. Although P_B can be enumerated given ρ and k , the only property we need here is that $P_B \rightarrow 1$ as $\rho \rightarrow 1$.

Contrast the above result to that for a simple server queue where, with $\rho = \lambda_l/\mu C$, the waiting time at link l is

$$W_l = \frac{1}{\mu C} \frac{\rho}{1-\rho} \quad (2)$$

We see that the multiple server operation has a waiting time advantage of at least k . The time delay is the waiting time plus the service time ($1/\mu C$). As $\rho \rightarrow 1$, the waiting time dominates, $P_B \rightarrow 1$, and the multiple server operation has an advantage in time delay of a factor of k . Although we concentrate in this paper on packet time delay as a performance measure, many other network situations focus on waiting time. In that case, greater reductions in delay may be achieved since P_B is of the order of ρ^k .

In making the above comparison, we assume that the link and node utilizations are not increased when adaptive routing is used to obtain multiple server performance at the nodes. Assume first that only minimum hop paths are used for the nonadaptive routing, resulting in single server queues at the links. Allow the adaptive routing to choose between all minimum hop paths for each commodity. This will produce the same utilization in both cases. If other paths are used in the nonadaptive routing, similar, but less precise, arguments can be made. It will be seen in the next sections that the adaptive routing performs better when more of the traffic has choice. Since there is a significant improvement in time delay, when the utilization is moderate it is possible to trade off a higher utilization by using longer paths with the resulting greater choice. In the examples discussed below, we have not found this to be necessary. In some cases where minimum hop paths are not used in the optimum nonadaptive routing, the adaptive routing could still provide a choice of similar length paths. Then the utilization will not be increased.

These observations are summarized in Figs. 1 and 2. In Fig. 1 we compare waiting times for queues with two servers and a single stream of input traffic. Multiple server operation is best. In the other three cases, a queue is provided for each server and the input stream is divided into two in different ways. In order of decreasing performance, an arrival joins the shorter queue, arrivals alternate between the two queues, and arrivals

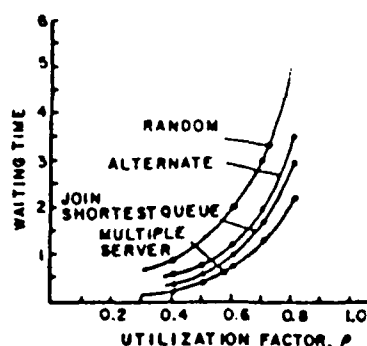


Fig. 1. Waiting time in a node with two servers for different routing decision rules.

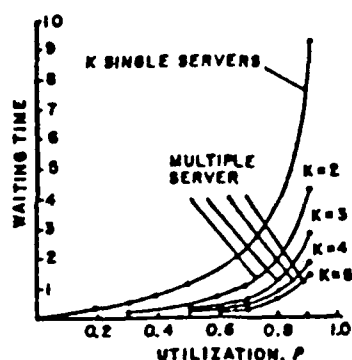


Fig. 2. Waiting time in a multiple-server node.

randomly select a queue. The latter is equivalent to two single server queues. In Fig. 2 we contrast multiple server and single server performance in a node.

Our adaptive routing procedure will attempt to obtain these performance advantages.

IV. ADAPTIVE ROUTING PROCEDURE

First consider routing all packets at a node by allowing them to use any outgoing link. The node can now be modeled as a multiple server queue and achieves the already described performance advantage over single server operation. Path lengths, however, would become extremely long because of this random routing, the total traffic in the network would increase catastrophically, and total network performance would be destroyed. We will show how to achieve the time delay advantage without this deterioration in total network performance.

We propose a two-level adaptive routing procedure. The first level is global, and while still adaptive, is expected to be slowly varying, responding to average statistics of congestion and traffic and alarms due to link failure, onset of congestion, new traffic, etc. We assume that some mechanism exists for making adjustments, and that these will be made relatively infrequently compared to the rate of second level adaptivity. We will assume in this paper that the first level of adaptation does not change over the period of interest.

The first level consists of assigning a set of "good" paths to each commodity or node pair that wish to communicate. Instead of finding the "best" path, we look for as large a set as possible of paths that are good in some sense—a small number

of hops, small congestion, etc. (A similar approach was presented in [5].) We do not force ourselves at this level to choose among alternative paths that are of the same or similar quality. That is done in the second level. The second level performs better if there are more alternative paths. There is an inherent tradeoff here, since choosing longer paths places more traffic on the network. We assume that a selection of paths is possible and, although we do not investigate it here, that it can be made slowly adaptive.

As a result of the first level procedure, there will be nodes where the paths for a particular commodity intersect, including the originating node. Packets at such a node have a choice of which outgoing link they can use (two or more links will be in the various allowable paths). There are certain links which may not be used by a particular commodity—they lead to bad paths. We do not force the choice on the packet upon arrival at the node, but operate the node as a (constrained) multiple server facility. If all packets could choose any of the links, then it would indeed be a multiserver facility and we would get the k -fold improvement in delay. We will show that most of this improvement is still obtained, even when some servers cannot service some commodities.

This two-level adaptive procedure will then give us the desired improvement in time delay while maintaining good paths.

V. ADAPTIVE NODE MODEL

First, as shown in Fig. 3, consider a node with two outgoing links (a and b). Some commodities have assigned paths that only use one of these links. They have no choice and will be routed directly to a dedicated queue which can only be served by one of the servers. Denote the total rate of traffic joining each of the queues as λ_a and λ_b . Other commodities (of total rate λ_c) have paths which use either link a or b . We assign them to a third queue (c). Server a , for example, may choose to serve a packet from a or c , but not b . We will discuss different strategies for the server.

The only way this constrained multiple server queueing system differs from $M/M/2$ behavior is when queues a and c are empty, for example, while queue b is not. One of the servers (a) is idle while the node still has some work. To reduce the possibility of this happening, we could keep queue c as large as possible. Thus, an excellent strategy, which appears to be optimum, is to give priority to the dedicated queues, serving queue c only when one of the dedicated queues is empty. Another strategy is to allow packets from the stream λ_c to join the shorter of the two queues (thus eliminating the need for the third queue). We have [4] evaluated these and other strategies for this node model and more complex node models, and have found the performance of the node as measured by average time delay to be fairly insensitive to the strategy adopted by the servers. Although different classes of traffic may suffer radically different time delays, the average delay of all traffic in a node remains roughly the same for many good strategies. We will consider below only two strategies—the one which gives priority to the dedicated queues and the one in which arrivals join the shortest queue.

In Fig. 4 we present the average queue length $E(N)$ versus utilization ρ for a two-server node under various traffic pat-

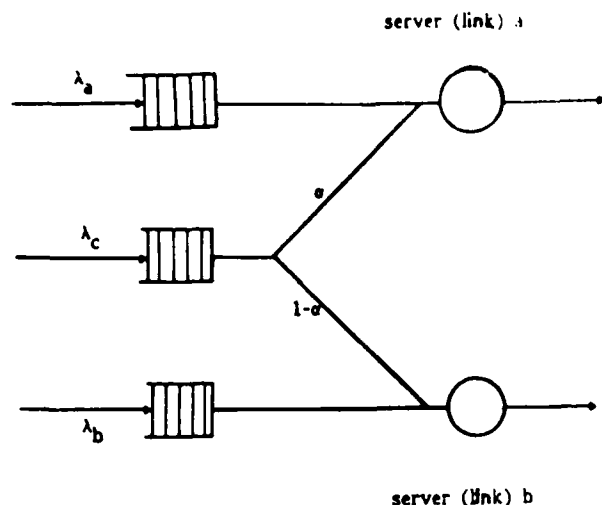


Fig. 3. A node with two servers.

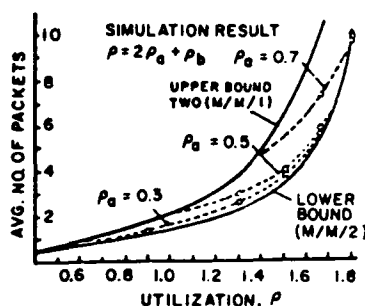


Fig. 4. Estimated average number of packets in a node with two links. Simulation results.

terns. We assume that the two link capacities are equal to C , $\lambda_a = \lambda_b$, $\rho_a = \lambda_a/\mu C$, $\rho_c = \lambda_c/\mu C$, $\rho = 2\rho_a + \rho_c$. The lower bound is for an $M/M/2$ queue or all nondedicated traffic ($\lambda_a = 0$). The upper bound is for all dedicated traffic ($\lambda_c = 0$) and represents two $M/M/1$ queues. In general, all of our results lie between those two bounds of performance. The circled points are simulation results for different traffic mixes and the strategy that gives priority to dedicated queues. From this figure and from similar results for three-server nodes and different traffic patterns (as discussed below and in Fig. 5), the following rule of thumb has emerged. As long as 15-20 percent of the traffic is not dedicated and the choice of traffic is not apportioned to disjoint sets of servers, then the node behavior will achieve more than half of the improvement of an $M/M/k$ queue. It has been shown [7], [8] that in the limit of heavy traffic (high utilization) a "join the shortest queue" policy tends towards the performance of an $M/M/k$ queue. This is true even if some of the arrivals cannot join some of the queues, but there must be some linkage between all the queues. This is just our node model. Since we believe that our priority strategy is better than a join the shortest queue policy, we have that in the limit of heavy traffic, $M/M/k$ behavior is achieved.

The above remarks hold for nodes with any number of links. Assume that a node has k links. Then there are $2^k - 1$ classes of input traffic: k classes of dedicated traffic, one to each server; $k(k-1)/2$ classes of traffic assigned to specific

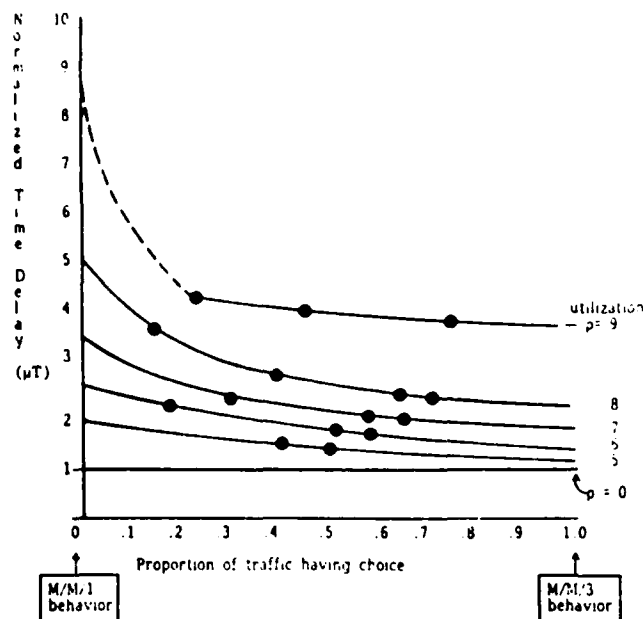


Fig. 5. Performance of a node with three links. Simulation results.

pairs of servers; ...; and finally one class of traffic that can use all servers (links). Many of these classes may be empty. But if they span all servers, and if the total traffic that is not dedicated is roughly 15-20 percent of the total traffic, then nodal delay is reduced by a factor of order k over a single queue model. If the traffic having choice is too light or if the servers are in isolated groups, then this advantage is reduced.

In Fig. 5 the normalized time delay (in units of service time or message length) is given as a function of the amount of traffic having choice at a three-server node and for various utilizations. These results were obtained by nodal simulations. The left limit of the graph represents $M/M/1$ or nonadaptive behavior. The right limit represents $M/M/3$ or the limiting adaptive behavior. There are two types of traffic that have choice at a three-server node. One type may be served by any of the three servers. The other type may be served by only two servers—there are three different members of this type. A variety of different combinations were simulated. The resulting time delay showed only slight variations for different traffic patterns and depended only upon the total traffic having choice. It is seen from Fig. 5 that if 15-20 percent of the traffic has choice, then the behavior is significantly closer to $M/M/3$ than to $M/M/1$.

For high utilizations, the time delay is dominated by the waiting time and the maximum improvement expected is by a factor equal to the degree of the node. For moderate utilizations, $M/M/k$ operation would result in even greater relative savings.

We have been able to develop upper bounds on node performance that exhibit some of the above behavior. For a more accurate understanding, we have resorted to the simulation results already described.

Consider the two-server node of Fig. 3 and assume that the dedicated queues have priority. Furthermore, let the capacities $C = 1$ for convenience. Then we can derive the following ex

pressions:

$$\mu E(W_a) = \rho_1 + \lambda_a E(W_a) \quad (3)$$

$$\mu E(W_b) = \rho_2 + \lambda_b E(W_b) \quad (4)$$

$$\mu E(W_c) \leq \frac{1}{2} P_B + \frac{1}{2} \lambda_c E(W_c) + \min_{i=a,b} \{\lambda_i [E(W_i) + E(W_c)]\}. \quad (5)$$

In these equations W_i is the waiting time of queue i , $E(W_i)$ is its expectation, $\rho_1 = (\lambda_a + \alpha \lambda_c)/\mu$, $\rho_2 = [\lambda_b + (1 - \alpha)\lambda_c]/\mu$, and α is the proportion of λ_c traffic served by server 1. P_B is again the probability that the system is busy. Equation (3) is derived as follows. An arrival to queue a must wait for server 1 to finish service, if it is occupied, and then wait for all messages that it found in queue a to be served. Recall that queue a has priority. The probability of server 1 being busy is ρ_1 , there are $\lambda_a E(W_a)$ messages already on queue a on the average, and each message has an average service time of $1/\mu$.

Equation (5) is more difficult to derive. An arrival to queue c waits for the next completion of service if the system is busy. The probability of this is P_B and the average time to the next completion is $1/2\mu$. It must also wait for all the messages that it found on queue c when it arrived. Again, the average number of these is $\lambda_c E(W_c)$ and each goes into service, when allowed, with a rate 2μ . But queue c does not have priority. Thus, it must also wait for the first queue to put into service all messages found in that queue upon arrival, $\lambda_a E(W_a)$ on the average, and for all other messages that arrived to that queue while the message on queue c was waiting $-\lambda_b E(W_b)$, on the average. The last term in (5) should be the expectation of the minimum of two random variables—the total number of messages in queues a or b while our message waits in queue c . We upper bound this with the minimum of the two expectations.

If we let $\lambda_a = \lambda_b$, then $\alpha = 1/2$. Furthermore, we upper bound P_B by $\rho_1 = \rho_2$. Equations (3)–(5) can now be solved to obtain for the average waiting time for all messages:

$$\mu E(W) \leq \frac{\rho}{1 - \rho} (1 - \frac{1}{2} \beta) \quad (6)$$

where $\beta = \lambda_c/(\lambda_a + \lambda_b + \lambda_c)$, the fraction of traffic having choice. Note that for $M/M/1$ behavior, $\mu E(W) = \rho/(1 - \rho)$, and for $M/M/2$ behavior, $\mu E(W) = (\rho/(1 - \rho))/(\rho/(1 + \rho))$. As $\beta \rightarrow 0$, our upper bound converges to $M/M/1$ behavior. As $\beta \rightarrow 1$, our upper bound yields a reduction of $1/2$, where $M/M/2$ behavior has a better improvement of $\rho/(1 + \rho)$. Our upper bound on P_B is causing the discrepancy here. In any event, we note the linear behavior with β in (6).

A similar expression, with similar results, can be derived for a three-server node. Let the arrival rate to the dedicated queues be equal, and assume only one other class of traffic which can choose between all three servers. Then we obtain

$$\mu E(W) = \frac{\rho}{1 - \rho} (1 - \frac{2}{3} \beta) \quad (7)$$

where β has the same meaning as above. Here, $\beta = 1/\mu E(W)$

is a third of that obtained with $M/M/1$ queues, whereas an $M/M/3$ queue would have an even smaller waiting time

VI. NETWORK PERFORMANCE

In general, we can say that if we adopt this adaptive routing procedure, and if at each node there is a sufficient amount of nondedicated traffic, then at high utilization the time delay at each node is reduced by almost the degree of that node from the time delay using nonadaptive routing. The effect on the network will be to reduce the time delay by a factor on the order of the average degree of the nodes. For moderate utilizations, a large part of that advantage will still be achieved. Assume, for simplicity, that using nonadaptive routing, all links have the same average time delay T_1 . Then $T = \bar{l}T_1$. If our adaptive strategy is used, then $T_{(n)} = 1/k_{(n)}T_1$ where $k_{(n)}$ is the degree of node n . For this adaptive routing, the network delay becomes at best $T \cong \bar{l}T_1 * \sum_n (\lambda_{(n)}/k_{(n)})/\lambda$. If all $k_{(n)} = k$, then the reduction is of the order of $1/k$.

The accuracy of the approximate performance at a node improves as more of the traffic has choice and as the utilization increases. These results serve as a quick estimate of the benefits obtainable from adaptive routing and are useful as a design tool.

VIII. EXAMPLES

We have evaluated many examples of networks [4]. Most of these had some symmetry and a modest number of nodes (at most 20). Our evaluation procedure was as follows. We first obtained the optimum nonadaptive routing and evaluated its performance. In most of the examples, this resulted in using shortest paths. We then allowed the adaptive routing to use all shortest paths for each commodity. We then found the amount of traffic that had choice at each node. For a wide variety of situations, there was sufficient nondedicated traffic at each of the nodes to justify our approximation.

There are two steps in a careful evaluation—the computation of the amount of traffic of each class (with respect to the nature of choice) in each node and the evaluation of node performance. We have already discussed good estimates for the latter. The former step can be quite complex, since it depends upon how traffic is split at each node at which it has choice. We have developed several approximate techniques to perform this step [4]. For example, consider Fig. 3 again. Let α be the fraction of λ_c traffic that is served by a . The total utilization of server a is $(\lambda_a + \alpha \lambda_c)/\mu = \rho_1$. Similarly, define $\rho_2 = [\lambda_b + (1 - \alpha)\lambda_c]/\mu$. Queue a , including the server, is empty with probability $1 - \rho_1$. A message on queue c will be served by a only if the entire queue is empty. Thus, approximate α by $\alpha = (1 - \rho_1)/(2 - \rho_1 - \rho_2)$. This can be solved for α to obtain $\alpha = (1 - \rho_a)/(2 - \rho_a - \rho_b)$ where $\rho_a = \lambda_a/\mu$, $\rho_b = \lambda_b/\mu$. This approximation has been verified by simulation [4].

We summarize one example of a six node network shown in Fig. 6. Each of the nodes has degree 3. We allow only shortest paths (over one of our adaptive procedures). Thus, all traffic from 1 to 2 or 4 or 5 or 6 has a choice of two paths, but traffic from 1 to 3 or 6 has a choice of three two-hop paths. We assume that all links have the same capacity, $\mu C = 10$, and all $\gamma_{ij} = 1$. It is fairly easy to see that $\rho = 0.6$, $\beta = 0.5$, and $\lambda = 42$. A non-

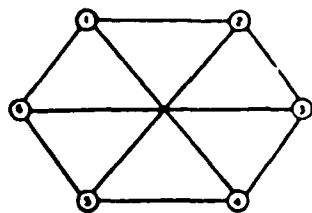


Fig. 6. A six-node network.

adaptive optimum routing can be found so that the flows in each link are identical, $\lambda_i = 7/3r$. Thus, for nonadaptive routing, $T = 1.4/(10 - 7/3r)$. As $r \rightarrow 30/7 = 4.29$, the adaptive routing will give an advantage of a factor of 3.

We now compare the optimum nonadaptive routing with our adaptive routing. For nonadaptive routing, we used the above results for single server queues. For the adaptive routing, we used the results of nodal simulations for three-server nodes and $T = \bar{T}(n)$ as discussed in Section VI. For $r = 1.84$, $\rho = 0.43$, the nonadaptive routing had a time delay of 0.245, while the adaptive routing had a time delay of 0.197. For $r = 3.06$, $\rho = 0.71$, the results were 0.490 for nonadaptive routing and 0.295 for adaptive routing. Thus, we have reduced the time delay by a factor of 1.66 in the latter case. In this example, the amount of traffic having choice at each node was 2/7 of the total or 29 percent. If full multiple server performance was attainable, the time delay would have been 0.217. Thus, 70 percent of this improvement was attained. The improvement increases if more traffic has choice. As the utilization increases, the full benefit of multiple server performance is achieved. The comparison can also be made with respect to waiting times since the service times are constant and cannot be reduced. Here the service time for the network is 0.14. Thus, for 43 percent utilization, the waiting time has been reduced from 0.105 to 0.057 by use of adaptive routing—a factor of 1.84. At 71 percent utilization, the reduction is from 0.350 to 0.155—a factor of 2.26.

A number of other examples with as many as 20 nodes have been investigated [4]. Most had some amount of symmetry, but not nearly as much as the example cited here. One 12-node example was generated arbitrarily. In all cases, the above procedure was followed. The optimum nonadaptive routing was found. In almost all of the examples, shortest paths were used. The adaptive routing was created by giving a choice among all shortest paths for each commodity. There was sufficient choice at most nodes.

The effects of asymmetry—of the topology, the traffic matrix, and link capacities—have not yet been adequately studied. For very small networks or pathological cases, longer paths may have to be used by the adaptive strategy to achieve sufficient choice at some nodes. This will increase the utilization and reduce, and perhaps eliminate, the performance advantage. However, the performance advantage seems significant enough to leave some margin for this tradeoff.

Asymmetries in link capacities and traffic patterns create another problem. The second level of adaptive routing is based on local information only. This may lead to saturation at other nodes. The first level of the adaptive routing, as described above, may not have enough flexibility to circumvent this. We

have begun to study a variation that shows some promise. Consider traffic between a pair of nodes that has two or more good paths. Previously we assumed that all that traffic was to be given choice. Here we dedicate some of that traffic to specific paths in order to balance the load. For example, we may wish to produce a flow distribution similar to that used in the nonadaptive routing scheme. The remainder of the traffic is allowed to have choice as before. Since the amount of traffic having choice at a node is not critical, it is expected that it will still be possible to achieve a performance advantage. Several simple examples that have been tried have been encouraging.

A brief report on two further examples illustrates these ideas. Details can be found in [4]. Fig. 7 shows an asymmetrical eight-node network. We assume equal channel capacities and equal requirements between pairs of nodes. Four of the nodes have two servers; the other four have three servers. Hence, we expect the maximum improvement to be by a factor between 2 and 3. The optimum nonadaptive routing uses shortest paths. The adaptive routing allows choice among these shortest paths, thus retaining the same utilization. The most utilized links were the two connecting clusters of four nodes. When these were 80 percent utilized, the adaptive routing scheme outperformed the optimum nonadaptive scheme by a factor of 1.6 for time delay and 2.6 for waiting time. When the utilization was 90 percent, these factors were 2.2 and almost 3, respectively.

Consider another asymmetrical eight-node network, shown in Fig. 8. Here we assumed an asymmetrical traffic matrix. If only shortest paths were allowed, little traffic has choice, a nonadaptive and our adaptive scheme are almost identical, and the time delay is 6.24 for both. The optimum nonadaptive routing had a time delay of 4.80. An adaptive routing can be found using the same length paths for each node-pair traffic. This resulted in a time delay of 4.08—a 15 percent improvement. There was not enough choice to get more improved performance in the nodes. When more choice was allowed, increasing the utilization, the time delay increased to 4.62. The service time here was equal to 2. Hence, the waiting time for shortest paths was 4.24, the optimum nonadaptive routing had a waiting time of 2.80, and the adaptive routing with equal length paths had a waiting time of 2.08—a 25 percent improvement.

More study is needed to fully understand the influence of these asymmetries on this type of adaptive routing.

VIII. CONCLUSIONS AND FURTHER WORK

We have presented a two-level adaptive routing procedure that exhibits marked improvement in performance over nonadaptive routing. Moreover, we have presented some analytic approximations to estimate this improvement. In heavy traffic, we have shown the improvement to be by a factor equal to the nodal degree (or average nodal degree). In general, for moderate utilization, a large part of this improvement is still achieved.

A more elaborate simulation of a network is underway. In that work, the independence assumption is not made. Although it confirms our model, it has shed some light on the "independence" assumption. We found that at high utiliza-

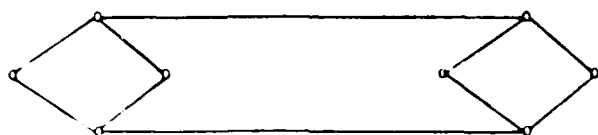


Fig. 7. An eight-node asymmetrical network.

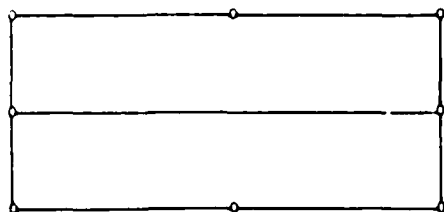


Fig. 8. An eight-node asymmetrical network used with an asymmetrical traffic matrix.

tion we were getting most, but not all, of the expected improvement. Many different nodal strategies had only a modest effect on resolving this gap. We found that it was easy to generate strategies that produced multiple server behavior. But some of them had a profound effect on the output process from the server, and thus affected the Poisson assumption for inputs to the next node. The priority scheme was notorious in this regard. Usually any traffic one hop away from its destination will be placed in a dedicated queue. Thus, only continuing traffic will be found in nondedicated queues. These have lower priority and will be transmitted only when the dedicated queue is empty. But then a burst of several packets is likely to be sent. A bursty arrival process results in poorer queue performance than a Poisson process. This contributed significantly to the delay performance gap. The design of a nodal strategy must consider not only multiserver performance achievement at a node, but the resultant output process as well. This is the direction of our current research.

The usual way such a burst could occur is when the packets are short. But these should have made the next node's job easier. Unfortunately, the second aspect of the independence assumption now comes into play—packet lengths are reassigned at each node. This accentuates the effect of burstiness. When the simulation was altered to take into account dependence between nodes, hence dispensing with the independence assumption, the difficulty disappeared.

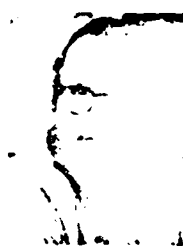
ACKNOWLEDGMENT

The simulation work described in the last section has been performed by P. Chu.

REFERENCES

- [1] M. Gerla, "Deterministic and adaptive routing policies in packet switched computer networks," in *Proc. 3rd Data Commun. Symp.*, St. Petersburg, FL, Nov. 1973, pp. 23-28.
- [2] R. G. Gallager, "A minimum delay routing algorithm using distributed computation," *IEEE Trans. Commun.*, vol. COM-25, pp. 23-35, Jan. 1977.
- [3] D. G. Cantor and M. Gerla, "Optimal routing in a packet switched computer network," *IEEE Trans. Comput.*, vol. C-23, pp. 1067-1068, Oct. 1974.

- [4] A. Livne and R. R. Boorstyn, "On a technique for dynamic routing," in *Proc. Nat. Telecommun. Conf.*, Dallas, TX, Nov. 1976, p. 42.7.1. Also, A. Livne, "Dynamic routing in computer communication networks," Ph.D. dissertation, Polytechnic Inst. New York, Brooklyn, NY, July 1976. Also, R. R. Boorstyn, "A model for efficient routing in S.F. computer communication networks," in *Proc. EUROHOP-79*, P. A. Samet, Ed., Amsterdam, North Holland, 1979, pp. 343-350.
- [5] H. Rudin, "On routing and delta-routing: A taxonomy and performance comparison of techniques for packet switched networks," *IEEE Trans. Commun.*, vol. COM-24, pp. 43-59, Jan. 1976.
- [6] L. Kleinrock, *Queueing Systems*, New York: Wiley, 1975.
- [7] G. J. Foschini and J. Salz, "A basic dynamic routing problem and diffusion," *IEEE Trans. Commun.*, vol. COM-26, pp. 320-327, Mar. 1978.
- [8] G. J. Foschini, "On heavy traffic diffusion analysis and dynamic routing in packet switched networks," *Computer Performance*, Chandy and Reiser, Ed., Amsterdam, North Holland, 1977, pp. 499-513.



Robert R. Boorstyn (M'58) was born in New York, NY, on May 28, 1937. He received the B.E.E. degree from the City College of New York, New York, NY, in 1958, and the M.S. and Ph.D. degrees, both in electrical engineering, from the Polytechnic Institute of Brooklyn, Brooklyn, NY, in 1963 and 1966, respectively.

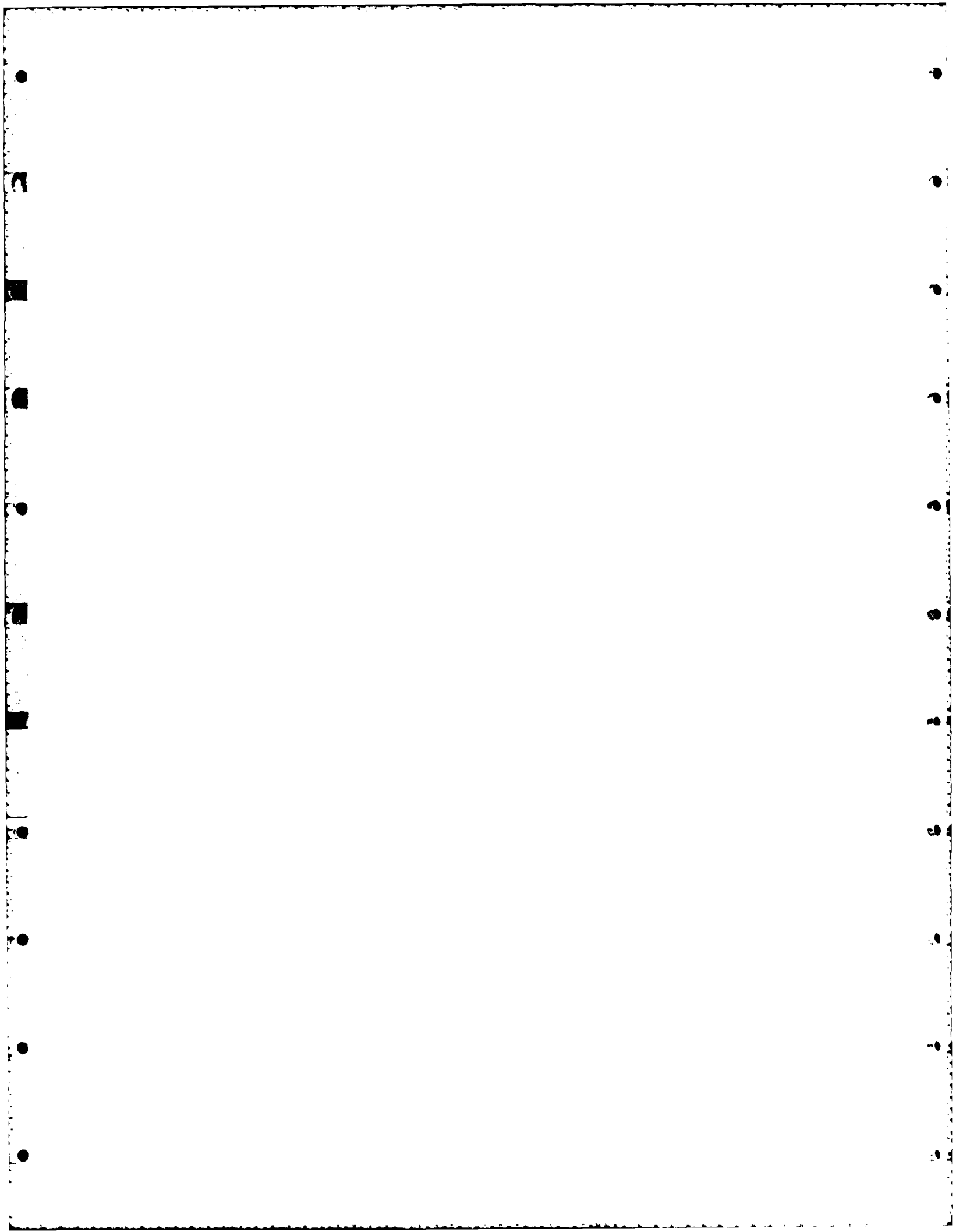
From 1958 to 1961 he worked as an Engineer for the Advanced Studies Department of the Sperry Gyroscope Company. In 1961 he joined the staff of the Department of Electrical Engineering, Polytechnic Institute of Brooklyn, now the Polytechnic Institute of New York, where he is currently a Professor. From 1977 to 1978 he was on leave at Bell Laboratories where he conducted research in computer communications networks. His current research interests are in computer communication networks, data communications, and communication theory.

Dr. Boorstyn has been Secretary of the IEEE Information Theory Group and Editor for Computer Communication of the IEEE TRANSACTIONS ON COMMUNICATIONS. He is Chairman of the Computer Communication Committee of the IEEE Communications Society and Associate Editor of the journal *Networks*. He was a member of the delegation to the first joint USSR-IEEE Workshop on Information Theory, Moscow, December 1975.



Adam Livne (M'72) received the B.Sc. and M.Sc. degrees in electrical engineering from the Technion—Israel Institute of Technology, Haifa, in 1959 and 1963, respectively, and the Ph.D. degree in electrical engineering from the Polytechnic Institute of New York, Brooklyn, NY.

From 1959 to 1965 he was an Electronic Design Officer in the Israeli Navy, where he conducted communications and radar systems designs. In 1965 he joined the Scientific Department of the Ministry of Defence as a Research Engineer in the Communication Department. He took part and directed advanced R&D projects in signal processing and communication systems. Between 1969 and 1971 he worked at the Lockheed Electronics Company, Plainfield, NJ, as a Principal Engineer in the design and simulation of digital signal processing systems. During 1974-1976 he completed his studies for and the Ph.D. degree at the Polytechnic Institute of New York. At present he is a Senior Research Engineer at the Electronic Division, Scientific Department, Ministry of Defence, Tel Aviv, Israel.



B.2 A Simulation Study of a Dynamic Routing Scheme

Chu, Boorstyn, and Kershenbaum

IEEE National Telecommunications Conference, November
1981, New Orleans

A SIMULATION STUDY OF A DYNAMIC ROUTING SCHEME

P. H. N. Chu,*
Bell Telephone Laboratories
Holmdel, New Jersey 07733

R. R. Boorstyn,[†]
Polytechnic Institute of New York
Brooklyn, New York 11201

A. Kershenbaum[†]

ABSTRACT

A dynamic routing scheme is studied. Instead of assigning fixed paths or randomly selecting paths, permissible paths (e.g., minimum hop paths) are pre-assigned to each node pair. A packet arriving at a node, and destined for another node, is assigned to one of two types of queues - a dedicated one or a shared one. The objective of the routing scheme is to reduce both nodal as well as network average packet delay. We have conducted a series of simulation studies to evaluate the nodal as well as the network delay under the proposed dynamic routing scheme. Several important and encouraging results are obtained and are presented in this paper.

1. INTRODUCTION

A dynamic routing scheme has been previously proposed (References [1], [2], [3], and [4]). Instead of assigning fixed paths or randomly selecting paths, permissible paths (e.g., minimum hop paths) are pre-assigned to each node pair. A packet arriving at a node, and destined for another node, is assigned to one of two types of queues. A dedicated queue is a queue whose packets are served by

one and only one dedicated server. A shared queue is a queue whose packets may be served by two or more specified servers. A server is equivalent to an output link. The purpose of pre-selecting paths is to avoid long paths and resultant increased network traffic that would occur in a random routing scheme. The purpose of the shared queue is to let a portion of the nodal traffic experience the delay of a multiple server queue and therefore reduce the average packet delay at a node. The objective of the model is therefore to reduce both nodal as well as network average packet delay.

We have conducted a series of simulation studies to evaluate the nodal as well as the network delay under the proposed dynamic routing scheme. A more detailed report can be found in Reference [5]. Section 2 describes the nodal and network models as well as the different service disciplines used. Section 3 gives the results of nodal and network delay performance. A summary is given in Section 4.

2. THE MODEL

2.1 The Model Model

The traffic from each pair is preassigned to one or more pre-selected paths (and therefore server(s) at a node). Two types of queue can be constructed at a node. A dedicated queue can only be served by one dedicated server. There are at most n such queues at a node with n links. A shared queue can be served by any of several preassigned servers. There could be as many as n shared queues at a node. A two server, three queue nodal model is shown in Figure 1.

* This research has been partially supported by Bell Laboratories through the TRP program while the author is pursuing the Ph.D. degree at PINY.

[†] This research has been partially supported by US. ARMY, CORADCOM, under contract DAAK 80-80-K-0579.

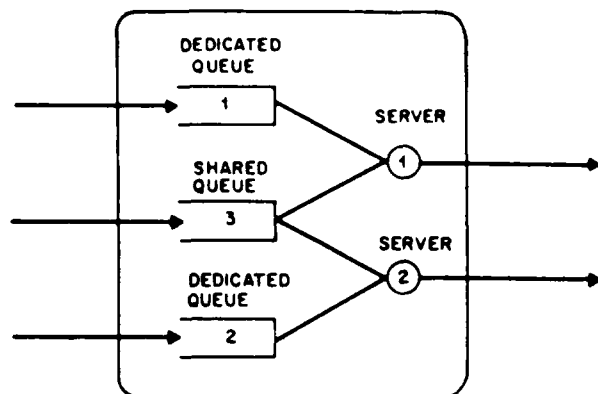


FIGURE 1 A NODAL QUEUEING MODEL WITH TWO SERVERS

In Figure 1, traffic arrives at the different queues with arrival rates λ_i ($i = 1, 2, 3$) packets/sec. Queue 1 and queue 3 are competing with server 1 for service while queue 2 and queue 3 are competing for service by server 2. The service rate is μ packets/sec. If one of the competing queues (e.g., queue 1 or queue 3) is empty (e.g., queue 1), and the server for these queues is idle, a packet in the non-empty queue is served. If both queues are not empty when the server has just completed service, a pre-defined service discipline will determine which queue to serve next. When both competing queues are empty (e.g., queue 1 and queue 3) and the server for these queues is idle, the server will stay idle until there is a new arrival to one or both of these queues. The idle server will not serve the queue which is not pre-assigned to it for service. This is the major difference between our nodal model and a multiple server queueing system. The operation of a node with more communications links is basically the same as that of a two link node. Shared queues will be served by a subset of the servers.

2.2 Service Disciplines

The nodal model described in the previous section allows different types of queues to compete for service. A service discipline must therefore be specified. We will describe four different service disciplines for the nodal model of Figure 1.

1. Priority Service Discipline (PSD)

A server always searches the dedicated queue for service prior to its service of the shared queue.

For example, server 1 in Figure 1, after completing a service, will always look to queue 1 for more service. Packets in queue 3 will be served by server 1 only if queue 1 is empty when server 1 has just completed a service. Therefore, server 1 is busy as long as there are packets to be served in queue 1 or queue 3. Conversely, server 1 is idle and stays idle when both queue 1 and queue 3 are empty. The instantaneous queue length of queue 2 does not effect any of the decision rules at server 1 described above. A packet entering the shared queue when both servers are idle is served randomly by either server 1 or server 2.

2. Alternate Service Discipline (ASD)

A server at the node in Figure 1, whenever possible, alternatively serves n packets from the dedicated queue and m consecutive ones from the shared queue. The server is forced to serve more than n (or m) packets from the dedicated (or the shared) queue if the other queue happens to be empty. Under this situation, alternation of service will begin whenever there is a new arrival to the empty queue. The process is started over again at this point.

3. Serve The Longer Queue Discipline (SLQD)

The queue with the longer queue among the two competing queues (the shared and the dedicated queues) gets service first. A server, upon a service completion compares the queue lengths of the two competing queues, and serves a packet from the longer queue. The SLQD tries to equalize all the queue lengths at a node.

4. Random Service Discipline (RSD)

A server (e.g., server 1), upon completing a service, serves packets from one of two competing queues according to the following algorithm: It serves queue 1 if queue 3 is empty and vice versa; it randomly selects a queue if both queue 1 and queue 3 are non-empty. This is a totally "uncontrolled" service discipline.

2.3 The Network Model

In order to describe the dynamics of a packet switching employing the proposed dynamic routing scheme, one

must (1) construct nodal queueing models similar to the one in Figure 1 for every node in the network, (2) pre-select a set of "permissible" paths for each source-destination node pair, and (3) assign the data flow according to the preselected paths between adjacent nodes. In order to simplify description of the network model, we have chosen a four node network to illustrate the dynamics of our network model.

A four node network is shown in Figure 2. Every node is connected to two links, i.e., is of degree two. Therefore, the nodal queueing model at each node is exactly the same as that of the two server three queue model described previously. The permissible paths between sources and destinations are selected based on a shortest path (minimum hop) algorithm. Adjacent nodes will only use the one hop path between them. Non-adjacent nodes will use either one of the two-hop paths. Therefore, traffic between adjacent nodes is pre-assigned to the dedicated queues. Traffic between non-adjacent nodes, on the other hand, is pre-assigned to the shared queue at the source node and the appropriate dedicated queue at the intermediate node.

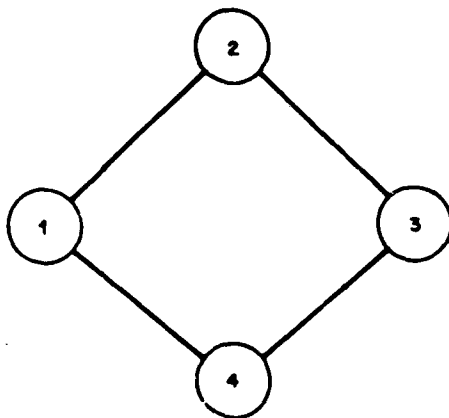


FIGURE 2 A FOUR NODE NETWORK

Each server at the node serves the two competing queues (the dedicated queue and the shared queue) according to a pre-specified discipline. We have simulated the network for the four service disciplines discussed previously. A service discipline where the condition of the next node is "known" to all servers at the current node and used to determine routing is also studied.

3. RESULTS

Simulation programs in PL/1 have been written to evaluate the delay of the nodal model of Figure 1 and the network model of Figure 2 operating under the

different service disciplines. All simulation results are based on statistics collected over a sufficient time interval so as to represent the steady state. The statistics collected for the PSD scheme are based on at least one million (sometimes four million) simulation events - including both arrivals and departures. In order to conserve computing time, the statistics collected for other service disciplines are based on at least two hundred thousand simulation events. In our simulation program, all arrival processes to source nodes are assumed to be Poisson and all service distributions are assumed to be exponential and Independent of each other. No assumption is made about arrivals processes to intermediate nodes in the network model.

3.1 Model Delay Performance

The percentage of the traffic that is shared at a node is an important parameter affecting the nodal delay. We have studied nodal delay based on three sets of nodal traffic mixes. They are low (approximately 10 percent to 15 percent), moderate (25 percent), and high (50 percent) percentages for the shared traffic at a node respectively. Fairly extensive simulation runs were conducted throughout the whole range of nodal utilization (0 to 1) for the PSD and a moderate percentage of shared traffic. We assumed that the input processes to the node were Poisson.

3.1.1 Simulation Results Of The Priority Service Discipline

The effects of the percentage of the traffic that is shared on nodal delay are tabulated in Table 1. Notice that the higher the percentage, the closer the nodal delay is to that of an M/M/2 queueing system.

Figure 3 plots the delay-throughput relationship of the PSD for different percentages of shared traffic. The delay for the M/M/2 and the M/M/1 queueing systems are also plotted for reference.

In Figure 4, a measure of the "closeness" of the PSD to M/M/2 queueing system

performance, $\frac{T_{M/M/1} - T_{PSD}}{T_{M/M/1} - T_{M/M/2}}$, is plotted

against the percentage of shared traffic for a node operated under the PSD scheme. The utilizations are studied. The ordinate is the percentage of multiple server performance that is obtained by the PSD. At high traffic utilization ($p=0.875$), only 25 percent

TABLE 1 Nodal Delay Of The PSD With Low, Moderate, And High Percentage Of Traffic Being Shared

$$(P = \frac{\lambda_1 + \lambda_2 + \lambda_3}{2\mu}, \mu = 4 \text{ packets/sec})$$

Shared Traffic	Arrival Rate to Queue (packets/sec)			Utilization ρ	Nodal Delay (seconds)		
	λ_1	λ_2	λ_3		PSD	M/M/2	M/M/1
Low	2.8	2.8	1	0.825	1.01	0.78	1.43
	3.0	3.0	1	0.825	1.33	1.07	2.00
	3.2	3.2	1	0.825	1.65	1.38	2.67
Moderate	2.475	2.475	1.65	0.875	0.91	0.78	1.43
	2.625	2.625	1.75	0.875	1.13	1.18	2.00
	2.775	2.775	1.85	0.925	1.37	1.37	2.33
High	1.65	1.65	3.3	0.825	0.80	0.78	1.43
	1.75	1.75	3.5	0.875	1.09	1.07	2.00
	1.85	1.85	3.7	0.925	1.38	1.38	2.67

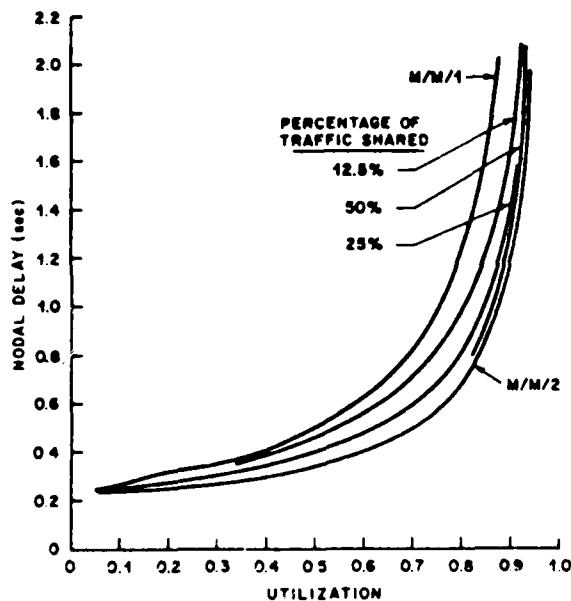


FIGURE 3 DELAY-THROUGHPUT RELATIONSHIP OF NODAL MODEL

of the nodal traffic need be shared to achieve 90 percent. At higher nodal utilization, the same effect is reached with less shared traffic. At moderate nodal traffic utilization ($\rho=0.5$), the PSD achieves 65 percent of multiple server performance with only 25 percent of the traffic being shared.

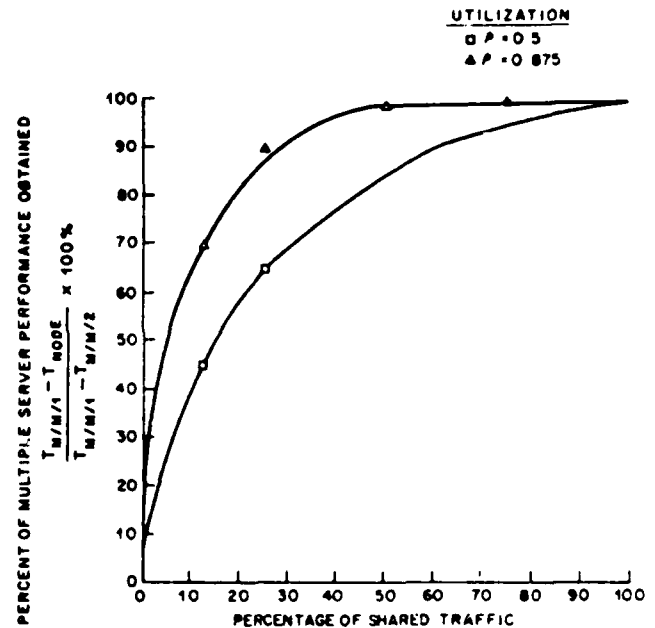


FIGURE 4 COMPARISON OF PSD AND MULTIPLE SERVER QUEUES

In Figure 5, we have redrawn the plot of Figure 3 for the average nodal delay in the range from 0 to 0.5 seconds for the case of 25 percent shared traffic. For a fixed delay of 0.5 second, the utilization of the PSD scheme is about 25 percent greater than that of the M/M/1 model (a commonly used fixed routing nodal model). Thus for the same throughput a significant decrease in required channel capacity is obtained.

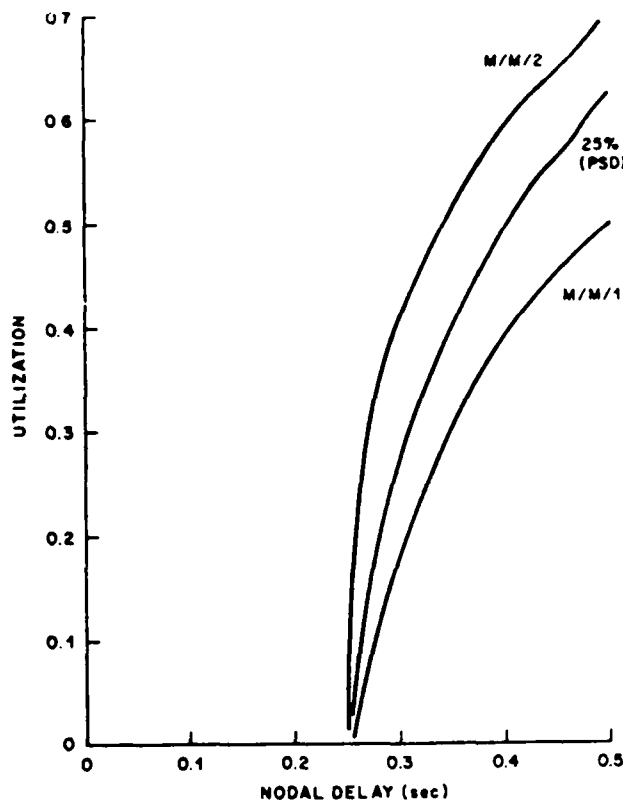


FIGURE 5 DELAY-THROUGHPUT RELATIONSHIP FOR NODAL MODEL

3.1.2 Results For Other Service Disciplines

In this section, we discuss the nodal delay for other service disciplines. For these disciplines, we only studied the nodal performance for $p = 0.875$ and 50 percent of the traffic being shared.

Table 2 tabulated the nodal delay from our nodal simulation for the ASD, the SLQD, and the PSD. We can make the following observations from Table 2:

1. A general characteristic of the results in Table 2 is that; at high utilizations ($p=0.875$ in this case) with a high percentage of the traffic shared (50 percent), the nodal delay is closer to that of the M/M/2 than the M/M/1 queueing system regardless of the service disciplines.
2. The PSD has the best nodal delay among all service disciplines.

3. The nodal delay of the SLQD, with the modification of not serving (whenever possible) two packets in a row from the shared queue, is the closest to that of the PSD.
4. The ASD with parameter $x=1$ and the threshold at the shared queue set to zero, has almost the worst nodal delay performance.
5. The RSD, being that it is an "uncontrolled" service discipline, does perform the worst among all service disciplines. However, the PRSD performs better than the M/M/1 queueing system by almost 40 percent.

Table 2 Nodal Simulation Results
 $\lambda_1 = \lambda_2 = 1.75$ packets/sec,
 $\lambda_3 = 3.5$ packets/sec, $\mu = 4$ packets/sec.
 $p = 0.875$

Service Discipline		Nodal Delay (seconds)	
PSD		1.03	
SLQD Serve the Longer Queue but not <u>y</u> consecutive "shared queue" packets	y = 1	1.04	
	y = 2	1.09	
	y = 3	1.08	
	y = ∞	1.15	
ASD Alternate Service: one from shared queue and x from dedicated queue but do not serve shared queue when its queue length ≤ TH	TH = 0	x = 1	1.11
		x = 2	1.12
		x = 3	1.05
	TH = 1	x = 1	1.09
		x = 2	1.05
		x = 3	1.07
	TH = 2	x = 1	1.08
		x = 2	1.05
		x = 3	1.04
	TH = 3	x = 1	1.06
		x = 2	1.07
		x = 3	1.05
RSD Random		1.13	
M/M/2		1.01	
M/M/1		2.00	

3.2 Network Delay Performance

3.2.1 Analysis

In this section, we study the average network delay performance. In our routing strategy, instead of treating each communications line separately, we focus our interest on the whole node, including all its communications links. As a result, our focus is on nodal delay as opposed to link delay which is usually studied.

Let us assume that we have an expression for nodal delay at node j , which is denoted by T_j . The average number of

packets N_j , at node j can therefore be written as $N_j = \lambda - T_j$, where λ is the total traffic (from both external and internal traffic sources) at node j . The average number of packets in the network (N) is therefore the sum of the average number of packets at each node. That is, $N = \sum_{j=1}^N N_j$. The average network delay ($\bar{T}$) is therefore

$$\bar{T} = \frac{1}{\gamma} \sum_{j=1}^N \lambda - T_j \quad (1)$$

where $\gamma = \sum_{j,k} \gamma_{j,k}$ is the total external network traffic demand and $\gamma_{j,k}$ is the offered rate of traffic between node pair (j,k) . Equation (1) is therefore a general expression for the average network delay.

To analyze the network delay we make the following assumptions. We assume the node models used previously. The traffic is found by applying the routing implied by the preassigned paths. We can then calculate the rate in packets per second of arrivals to the shared and dedicated queues. In our network analysis we assume these arrivals to be a Poisson process and compute the nodal delay for each node using the techniques described above. Equation (1) gives us the network delay. Below we compare this technique with the results of two simulations. The first dispenses with the Poisson assumption and accurately models arrivals to intermediate nodes in paths. To make the simulation simpler, packet lengths are independently reassigned at each node in a path. The second simulation dispenses with this approximation and retains the initial packet length for a throughput its path.

3.2.2 Simulation

In this section, we will focus our discussion on two PL/1 simulations which were written to evaluate the network delay performance. The interarrival time of arrivals to network nodes is assumed to be exponentially distributed. The packet length is assumed to be exponentially distributed. All servers are assumed to have the same service capacity (for convenience as unity). Therefore, the average service rate of each server is the same. In most of our simulation studies, we have used $p=0.875$, 50 percent shared traffic at the nodes, and a symmetrical network. Other traffic utilizations, nodal traffic mixes, and asymmetric network traffic

demands are also studied but not as extensively as the above.

The first simulation allows the packet lengths to be reassigned at the next node. It does not make any assumptions about the output process from a server. A server serves its competing queues according to a pre-specified service discipline. After the server finishes service of a packet, the packet either exits from the network or joins the next queue for more service. Once it arrives at the next queue (in the next node), a new packet length from the same exponential distribution is assigned to it. It then is put on the appropriate queue and waits for service. We compare the results of our network simulation with our analysis based upon nodal simulation results. In the latter, we assume all inputs to the node to be Poisson.

The second simulator simulates the true service situation in an operating packet network. That is, a particular packet when served by the next server will require the same amount of service time. When the results obtained from the second simulation are compared with our analysis by using our nodal simulation results, we can evaluate the effect of both the Poisson output stream and the packet length reassignment assumptions.

3.2.2.3 The First Network Simulation

3.2.2.1.1 Local Service Disciplines

All four service disciplines are studied. The simulation results in Table 3 and in previous work (Reference [5]) reveals the following:

1. The average network delay from the network simulation is significantly closer to that of the M/M/2 nodal model than to the M/M/1 nodal model.
2. The difference between the analytical results using the nodal model and those obtained from the network simulation, for the PSD, increases as the percentage of shared traffic increases at a node.
3. With the same traffic mix at a node the difference between the network delay found analytically and those obtained from the network simulation for the PSD increases as the traffic load increases.
4. The network delay obtained from the network simulation for the Alternate Service Discipline (ASD) and the Serve the Longer Queue Discipline (SLQD) are lower than that of the

PSD scheme. The difference in these delay are at most 11 percent. The network delay of the Random Service Discipline (RSD) scheme is approximately the same as that of the PSD scheme.

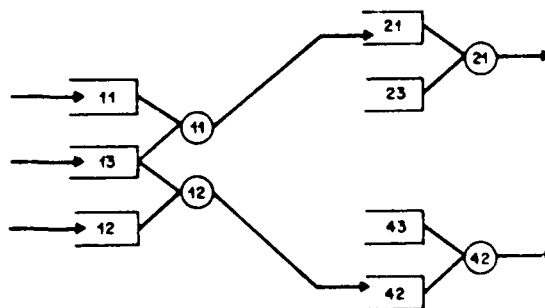
5. The difference between the results obtained analytically and from our network simulation for other service disciplines is substantially large (12 to 26 percent difference).
6. The parameter n (the number of packets to be served from the dedicated queue before the server alternate its service) in the ASD has an effect on network delay.
7. In the ASD, the threshold parameter set at the shared queue to avoid the shared queue becoming empty before the dedicated queues, also has an effect on network delay.
8. The network delay of the RSD - an "uncontrolled" and "unpredictable" service discipline - is about the same as that of the PSD and the "worst cases" of the ASD and the SLQD.

Although we are operating at high utilization, we are getting most, but not all, of the expected improvement. Many different nodal strategies have only a modest effect on resolving this gap. We find that it is was easy to generate strategies that produced multiple server behavior. But some of them have a profound effect on the output process from the server, and thus affect the Poisson assumption for inputs to the next node. The PSD is notorious in this regard. Usually any traffic one hop away from its destination will be placed in a dedicated queue. Thus, only continuing traffic will be found in nondedicated queues. These have lower priority and will be transmitted only when the dedicated queue is empty. But then a burst of several packets is likely to be sent. A bursty arrival process results in much poorer queue performance than a Poisson process. This contributes significantly to the gap in delay. The design of a nodal strategy must consider not only multiserver performance at a node, but the resultant output process as well. The usual way such a burst could occur is when the packets are short. But these should have made the next node's job easier. Unfortunately, the second aspect of the independence assumption now comes into play - packet lengths are reassigned at each node.

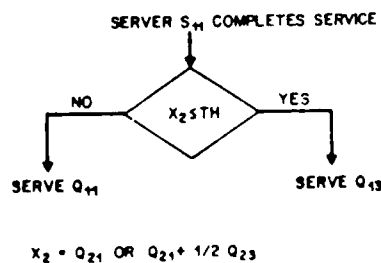
This accentuates the effect of burstiness. When the simulation is altered to take into account dependence between nodes, hence dispensing with the independence assumption (as described in the next section), the difficulty disappeared.

3.2.2.1.2 A Look Ahead Service Scheme

A common situation of all the "local" service disciplines (The PSD, the ASD, the SLQD, and the RSD) is that a packet may be waiting to be served in the shared queue at one node while there is an idle server at the next node. Conversely, a packet may wait in the dedicated queue for service when there is no need to serve a shared queue packet because the next server has plenty of packets to serve at the next node. Both phenomena cause degradation in network performance. Both are due to the fact that each server, upon completing a service, does not know what the next server wants. Consequently, the server may serve a "wrong" packet from time to time. Therefore, it is not the selection of a queue but the timing of this selection process which influences the service decision. In all of the "local" service disciplines which we have discussed so far, the timing factor was not considered. We have designed a look ahead service discipline and have examined its performance. The basic algorithm is shown in Figure 6.



(a) A PART OF THE FOUR NODE NETWORK



(b) THE LOOK AHEAD ALGORITHM

FIGURE 6 A LOOK AHEAD SERVICE SCHEME

Table 3 Network Simulation Results
 $(\lambda_{11}=\lambda_{12}=0$ packet/sec, $\lambda_{13}=3.0$ packets/sec, $\mu_{11}=\mu_{12}=4.0$ packets/sec, $\rho=0.875)$

Service Discipline			Network Delay (seconds)		% Difference*
			Nodal Model	Simulation	
PSD			2.06	2.75	33%
ASD Alternate Service: one from shared queue and n from dedicated queue but do not serve shared queue when its queue length $\leq$ TH	TH = 0	n = 1	2.22	2.65	20%
		n = 2	2.24	2.51	12%
		n = 3	2.10	2.58	23%
	TH = 1	n = 1	2.18	2.50	15%
		n = 2	2.10	2.50	19%
		n = 3	2.14	2.53	18%
	TH = 2	n = 1	2.16	2.59	20%
		n = 2	2.10	2.52	20%
		n = 3	2.08	2.62	26%
	TH = 3	n = 1	2.12	2.51	18%
		n = 2	2.14	2.49	17%
		n = 3	2.10	2.55	21%
SLQD Serve the Longer queue but not y consecutive shared queue packets					
	y = 1	2.08	2.45	18%	
	y = 2	2.18	2.58	19%	
	y = 3	2.16	2.67	24%	
			2.26	2.67	19%
RSD			2.26	2.71	20%
M/M/2			2.02	-	-
M/M/1			4.00	-	-

$$* \% \text{ Difference} = \frac{T_{\text{NETWORK}} (\text{Simulation}) - T_{\text{NETWORK}} (\text{Nodal Model})}{T_{\text{NETWORK}} (\text{Nodal Model})}$$

Table 4 show our results. We make the following observations:

1. The network delay of the best look ahead scheme has an improvement of about 11 percent over that of the best local scheme.
2. The look ahead service scheme performs consistently better than the local service disciplines.
3. When both servers are empty, it is better to serve an arrival to the shared queue on the basis of downstream information, than to do it randomly.

3.2.2.2 The Second Network Simulation - No Packet Length Reassignment

The analytically obtained network delay and our first simulation differ not only in the PSD but also in the ASD and the SLQD where the output stream should be somewhat smoother. Furthermore, as was demonstrated by Kleinrock (Reference [5]), the packet length reassignment assumption does not work well in a small network like our four node network with very few traffic streams flowing at each

node. This motivated us to build the second network simulator whereby the packet length reassignment assumption is not made.

3.2.2.2.1 Dynamic Routing Scheme With Local Node Service Disciplines

We have used our second simulation to evaluate the network delay performance under the various service disciplines (PSD, ASD, SLQD, and RSD) discussed previously. Each node, operates independently, serves its queues according to the pre-specified service discipline. Traffic information at each node is not exchanged between adjacent nodes. Therefore, only "local" node service disciplines are discussed in this section. We have also examined the average network delay under two fixed routing strategies (Reference [5]).

Table 5 tabulates our results. We make the following observations:

1. Removing the packet length assumption greatly improves the accuracy of our analytic nodal method.

Table 4 Network Simulation Results Of A Look Ahead Routing Strategy
 $(\lambda_{11}=\lambda_{12}=0$ packet/sec, $\lambda_{13}=3.5$ packets/sec, $\mu_{13}=\mu_{12}=4$ packets/sec, $\rho=0.875$)

Threshold	Network Delay (seconds)		
	$x_2 = Q_{21}^*$	$x_2 = Q_{21}^{**}$	$x_2 = Q_{21} + \frac{1}{2} Q_{23}^{**}$
0 (PSD)	2.75	2.75	--
1	2.33	2.26	2.27
2	2.36	2.29	2.19
3	2.37	2.32	2.26
4	2.35	2.42	2.32
5	-	-	2.35
6	-	-	2.41

* When both S_{11} and S_{12} are not busy and there is an arrival to Q_{13} , this packet will be served by either S_{11} or S_{12} .

** Under the same situation as above, x_2 will be compared with x_4 (the same parameter at node 4).

The packet in Q_{13} will be served by S_{11} (of S_{12}) if

$$x_2 < x_4 \text{ (or } x_2 > x_4 \text{)}$$

2. The packet length reassignment assumption has more effect on the network delay performance for the PSD than any other service disciplines.
3. The PSD is the best "local" service disciplines among all other local service disciplines.
4. The network delay of our dynamic routing scheme under any one of the service disciplines is significantly better than either of the benchmark fixed routing strategies.

3.2.2.2 Dynamic Routing With Look Ahead Service Scheme

The "clock driven" network simulation keeps track of how much time a packet requires from a server. This enables a server to know how much work (in terms of service time) the next server has at any given instant of time. A server can make an accurate decision to keep the network from losing its service capacity. A server, upon completing a service, compares the total amount of work at the next server (the remaining service time for the packet still in service at the next server, the total service time in the dedicated queue at the next node, the total service time in the shared queue at the next node, and a threshold parameter which incorporates the fact

that the shared queue packets are served by both servers at the next node and there are new arrivals to the next node during the service time) to the sum of the service times of the first packet in each of its competing (shared and dedicated) queues. The server serves the shared queue packet if the total amount of work at the next server is less than the sum of the service times of the first packet in each of its competing queue. Should the server not serve the shared queue packet immediately, the next server could be idle for a period of time equal to the difference between these two quantities. The server would otherwise serve the dedicated queue packet. This look ahead service scheme would be ideal in our dynamic routing if the threshold parameter could be correctly set. The network delay with both exponentially distributed and constant packet lengths are evaluated. The following observations can be made (see Table 6):

1. The PSD is equivalent to setting the threshold value to negative infinity.
2. The network delay is insensitive to the threshold values near its "optimal" setting.
3. The difference between the network delay of the "best" local service discipline (the PSD) and the look ahead service scheme with the "best"

choice of the threshold value is about 10 percent.

Table 5 Network Simulation Results For Different Service Disciplines - Local Decision.

($\lambda_{11} = \lambda_{12} = 0$ packet/second, $\lambda_{13} = 3.5$ packets/second, $\mu_{11} = \mu_{12} = 4$ packets/second, $\rho = 0.875$)

Service Discipline	Network Delay (seconds)		
	First Simulation	Second Simulation	Analytic
PSD	2.75	2.16	2.06
ASD	2.56	2.38	2.23
SLQD	2.48	2.32	2.08
RSD	2.73	2.56	2.26
Fixed Routing #1	--	3.84	--
Fixed Routing #2	--	3.26	--

Table 6 Network Simulation Results For A Look Ahead Routing Strategy ($\lambda_{11} = \lambda_{12} = 0$ packet/second, $\lambda_{13} = 3.5$ packets/second, $\mu_{11} = \mu_{12} = 4$ packets/second, $\rho = 0.875$)

Routing Strategy		Network Delay (seconds)	
		Exponential Packet Lengths	Constant Packet Lengths
Local	PSD	2.16	1.23
	RSD	2.56	1.31
Look Ahead	TH	-0.25	1.86
		0.00	1.94
		0.05	1.96
		0.10	1.94
		0.15	1.95
		0.25	1.96
		0.50	2.01
		0.75	2.07
		1.00	2.13

3.2.2.2.3 Asymmetric Traffic Demands And Service Rates

We have illustrated significant improvements in network delay using our adaptive techniques for the four node network. However, only symmetric traffic loads have been studied so far. In order to demonstrate the robustness of our routing strategy, a certain degree of asymmetry was introduced into the network. We have evaluated the network delay of the four node symmetric network with asymmetric traffic and with different link capacities.

We make the following observations from the results of Tables 7 and 8:

The RSD performs the worst among all service disciplines; the FFI performs the best among all routing strategies

based on local traffic information; the "best" look ahead scheme performs better than the PSD in terms of average network delay; and the threshold value set in the look ahead scheme is insensitive to the network delay performance over a wide range.

4. CONCLUSION

We have studied a dynamic routing scheme for packet switching networks. Its network delay performance is superior to that of fixed routing. We have also developed and evaluated an analytic technique for evaluating adaptive routing performance. Extensive simulations confirm these results. A locally adaptive scheme performs almost as well as one that has more information.

Table 7 Network Delay With Asymmetric Traffic Demands

($\lambda_{12}=2$ packets/sec, $\lambda_{12}=1$ packet/sec, $\lambda_{13}=2$ packets/sec,
 $\mu_{1j}=4$ packets/sec, $\rho=0.875$)

Routing Strategy		Network Delay (seconds)	
Local	PSD	1.91	
	SLQD	1.96	
	RSD	2.67	
Look Ahead	TH	-0.25	1.81
		-0.125	1.81
		0.05	1.82
		0.10	1.81
		0.15	1.83
		0.25	1.84
		0.50	1.87
		0.75	1.91
		1.00	1.95

Table 8 Network Delay With Asymmetric Service Rates

($\lambda_{11}=\lambda_{12}=0$ packets/sec, $\lambda_{13}=3.5$ packets/sec, $\mu_{11}=\mu_{22}=\mu_{33}=\mu_{42}=5$
 packets/sec, $\mu_{12}=\mu_{21}=\mu_{32}=\mu_{41}=3$ packets/sec, $\rho=0.875$)

Routing Strategy		Network Delay (seconds)	
Local	PSD	2.23	
Look Ahead	TH	-0.25	2.07
		0	2.03
		0.25	2.04
0.75		2.15	
		1.00	2.22

REFERENCES

- [1] Livne, A., "Dynamic Routing in Computer Communication Networks," Ph.D. Dissertation, Polytechnic Institute of New York, June, 1977.
- [2] Livne, A. and R. R. Boorstyn, "On a Technique for Dynamic Routing," Proc. National Telecommunications Conf., Dallas, Nov. 1976, p. 42.2-1
- [3] Livne, A. and R. R. Boorstyn, "A Model for Efficient Routing in S/F Computer Communications Networks," EURO IFIP 79, P. A. Samet, ed., North-Holland Publishing Company, 1979, pp. 343-350.
- [4] Boorstyn, R. R. and A. Livne, "A Technique for Adaptive Routing in Networks," IEEE Trans. on Communications, Vol. COM-29, Number 4, April, 1981.
- [5] Chu, P. H. N., "A Dynamic Routing Scheme in Packet Switching Networks," Ph.D. Dissertation, June, 1981.
- [6] Kleinrock, L., "Communication Nets - Stochastic Message Flow and Delay," McGraw-Hill, New York, 1964, and Dover, New York 1972.

B.3 Stable Routing Patterns

Third Semiannual Technical Report, March 1982

RESEARCH SUMMARIES

A. Studies in Adaptive Routing

A.1. Stable Routing Patterns

As part of a general investigation in the area of dynamic routing in computer communication networks we consider the problem of finding stable global routing patterns. Specifically, we have proposed [1] a two-level routing procedure where the lower (local) level adapts dynamically to instantaneous variations in the congestion of the network in the immediate vicinity of each node and the higher (global) level ensures stability by keeping the average load across the entire network in some sense globally balanced. We now consider this latter problem.

We consider as a measure of global balance the utilization of the most heavily utilized network element (node or link) and seek to minimize this quantity. For simplicity, we will speak only of link utilizations. (Node utilizations can be included in a straightforward manner.) Thus, we are given a network containing N nodes and L (directed) links. Each link, l , has a capacity C_l . There are (directed) requirements r_{ij} between nodes i and j . Each r_{ij} is satisfied by routing it on one or more paths $P_{ij}^{(K)}$ from i to j . (In the two level adaptive routing scheme, these paths (or links within them) will be the alternatives open to each requirement.) A routing pattern is defined by the paths $P_{ij}^{(K)}$ and the fraction, $f_{ij}^{(K)}$, of r_{ij} using each $P_{ij}^{(K)}$. The utilization of each link is equal to the total flow (sum of fractions of requirements) on the link divided by its capacity.

The maximally utilized link is in a sense the most vulnerable part of the network and the most likely cause for the dynamic routing mechanism to break down (e.g. loop) due to congestion. By minimizing the

utilization of the maximally utilized link we seek to minimize the chance of congestion leading to such failure. It should be noted that we are dealing with the global level of the routing procedure here and as such consider only long term average utilizations, not instantaneous measures. The local level of the routing procedure concerns itself with making decisions instantaneously on the basis of the local state of the network in the vicinity of a node. Even within the constraints of a given $P_{ij}^{(K)}$ and $f_{ij}^{(K)}$, defined by the global strategy, the local level has considerable flexibility in choosing when to use each route and as such can obtain substantial reductions in delay when compared with static routing policies.

We now turn to the problem of actually finding the optimal global routing pattern as defined above. The technique resembles the Flow Deviation Method of Cantor and Gerla [2] and will be described in similar terms. Cantor and Gerla sought to minimize the average delay whereas we seek to minimize the maximum utilization of a link. Both functions are convex functions over a convex region and as such, the same type of procedure can be proven to yield an optimal routing pattern. The proof is given in [2].

Our function is only piecewise differentiable and as such, the gradient search used in [2] is not appropriate. In fact, the alternative described here takes the special nature of the objective function, a minimum of linear functions, into account and not only overcomes its non-differentiability but also is considerably more efficient and easier to implement than a gradient search.

We now give an outline of the optimization procedure. A high level flowchart of this procedure is given in Figure 1. As mentioned

above, at this level the procedure is almost identical to the Flow Deviation Algorithm. The key difference, which is only evident in a more detailed description, is how the optimal superposition of flows is found.

We define the length of a link to be its utilization. Initially, we set all link lengths to 0. We define the length of a path to be the length of the longest link in the path plus a small constant times the number of links in the path. This latter term is added to break ties among paths with equally utilized links in favor of a path with the smallest number of links. Note that this definition of path length is different from the conventional one but it serves our purpose. Shortest paths using this metric are computable using conventional shortest path algorithms.

The routing pattern found at each stage in the optimization procedure is a single shortest path for each requirement r_{ij} . (In general, this path is not unique, but this poses no problem.) A flow pattern is defined as the total flow in each link and is found by loading the requirements onto the links specified in the current flow patterns.

An optimal superposition of the current flow pattern with all previous flow patterns is then found. This is the key step in the procedure and is done by finding the value of λ between 0 and 1 which minimizes V , the maximum link utilization, where λ represents the fraction of all previous flow patterns used. The new optimal superposition of flows is then λ times the previous superposition plus $(1-\lambda)$ times the current flow pattern. A new superposition, and hence link utilizations, is then obtained. This in turn yields a new value for $V(K)$ and the link lengths. We can now start another iteration. If, however, no improvement in $V(K)$ has been observed, the iteration has converged

and we terminate the procedure with an optimal flow pattern. By saving the routing patterns and $\lambda^{(K)}$, the values of λ for each K , the optimal routing pattern can be obtained. In particular, if $P_{ij}^{(K)}$ is the (i,j) path first used in the routing pattern in iteration K then the fraction of commodity (i,j) using $P_{ij}^{(K)}$ is

$$(1-\lambda^{(K)}) \prod_{j=K+1}^M \lambda^{(j)}$$

where M is the number of iterations and the product is defined equal to 1 for $K = M$.

We now turn to the problem of how to find the optimal superposition of flows. Consider two flow patterns, $F^{(1)}$ and $F^{(2)}$. Each flow pattern assigns a flow to each link. Thus $f_{ij}^{(1)}$ and $f_{ij}^{(2)}$ are the flows assigned to link (i,j) in $F^{(1)}$ and $F^{(2)}$, respectively. For any λ between 0 and 1, the flow assigned to link (i,j) by superposing $\lambda F^{(1)}$ and $(1-\lambda)F^{(2)}$ is then

$$\lambda f_{ij}^{(1)} + (1-\lambda)f_{ij}^{(2)}$$

which equals

$$f_{ij}^{(2)} + \lambda(f_{ij}^{(1)} - f_{ij}^{(2)})$$

which is a linear function of λ . Dividing by C_l , to obtain utilizations, there will be, in general, a different function, $a_l + b_l \lambda$ for each link l . (We will for simplicity refer to links by a single index, l , rather than endpoints (i,j) .)

We seek the value of λ which minimizes the maximum of these functions over all l . Several simple observations allow us to find this value of λ in an efficient and straightforward manner. First, if for two

links, l and m , $a_l \geq a_m$ and $b_l \geq b_m$ then link l is said to dominate link m and link m can be ignored as it clearly does not participate in the maximum since $a_l + b_l\lambda \geq a_m + b_m\lambda$ for all values of λ . Indeed, if $a_l + b_l\lambda > a_m + b_m\lambda$ for all λ between 0 and 1 then link m may be ignored. Note that this latter condition is not equivalent to the former, for example if $a_l = 10$, $b_l = 0$, $a_m = 2$, and $b_m = 3$.

We can then arrange the links l in descending order of a_l and examine the b_l . Any link m for which b_m does not exceed b_l , where l is the predecessor of m in the order, can be eliminated. We now have an ordering of links which is descending in a_l and ascending in b_l . We then compute, for each adjacent pair of links l and m , the value λ_l for which $a_l + b_l\lambda_l = a_m + b_m\lambda_l$. The λ_l should form an ascending sequence. A value of λ_l which is less than the value of its predecessor in the sequence corresponds to a link l which can be eliminated from further consideration. In this case, link l is eliminated and m has a new predecessor. The λ_p is then recomputed for link p the new predecessor of m and this process is repeated. For link n , the last link in the sequence, $\lambda_n = 1$.

We now compute for each remaining link l $v_l = a_l + b_l\lambda_l$ and select the minimum of these values. The resulting λ_l and v_l are the desired values yielding the optimal superposition.

This entire process is illustrated in Figure 2. The links have been sorted so that the a_l are descending. Links with nonascending b_l have already been eliminated. Thus the b_l form an ascending sequence. This is evident in Figure 2 by the fact that the lines form a sequence increasing in slope. The intersection of lines 1 and 2 (i.e. the lines starting at a_1 and a_2) defines λ_1 . Similarly, the intersection of lines

2 and 3 defines λ_2 and $\lambda_2 > \lambda_1$. So thus far no line is dominated. The intersection of lines 3 and 4 takes place between $\lambda = 1$ so line 4 is dominated by line 3 and line 4 is thus eliminated from further consideration. The intersection of lines 3 and 5 defines a value of λ_3 (dotted line), but when λ_5 is computed we find it to be less than λ_3 . So, line 5 is dominated by line 6 and removed from further consideration. λ_3 is then recomputed from the intersection of lines 3 and 6. λ_6 is computed from the intersection of lines 6 and 7. Finally $\lambda_7 = 1$. This leaves us with $\lambda_1, \lambda_2, \lambda_3, \lambda_6$, and λ_7 (also $\lambda_0 = 0$). We search among the corresponding v_1 and find v_3 is minimum. It and λ_3 define the desired superposition.

The entire optimization process is illustrated in Figures 3, 4, and 5. The network consisting of 3 nodes, 6 links and 6 requirements is shown in Figure 3. For simplicity, we assume *symmetric requirements* and link capacities. We can thus assume a symmetric solution, i.e., routes for r_{ij} the reverse of routes for r_{ji} and equal utilization of each link in both directions. This allows us to only consider 3 links and 3 requirements in the example. This is done to simplify the example. The actual procedure works with directed links and requirements. It can also be used with undirected links and requirements but such a situation is rarely physically meaningful.

Initially, all requirements are routed directly since the initial shortest paths by our definition would be the paths with the minimum number of links. This is illustrated in Figure 4a. The link lengths are then recomputed -- link 1 has a utilization of .3 and hence a length of .3, etc. The shortest paths are then recomputed and are shown in Figure 4b. Note that the shortest path from B to C is now B-A-C.

The requirements are loaded onto these paths. The flow pattern is shown in Figure 4c.

Now a superposition of the flow patterns in Figures 4a and 4c is done. Figure 5 illustrates the dynamics of this. Note that r_{AC} dominates r_{AB} and that the optimal λ is .9 and $v = 4.5$. Figure 4d shows the resultant routing pattern formed by using the first routes for 90% of the traffic and the second routes for the remaining 10%. Figure 4e shows the flow pattern resulting from the superposition. Note that the maximum utilization is .45 (in links (A,C) and (B,C)) which is less than the maximum in either of the patterns in Figures 4a and 4c.

We now recompute the link lengths and the shortest paths. The resultant routes are the same as in Figure 4a. An optimal superposition between this flow pattern and the one in Figure 4e is then done. The optimal value of λ is 1, no improvement in v is found and we conclude that the routing and flow pattern in Figures 4d and 4e are optimal. Note that the links (A,C) and (B,C) are both maximally utilized. They form a cut which is analogous to the saturated cut in Gerla's Cut Saturation Method. (The existence of such a cut is a necessary condition for the optimality of a flow pattern.)

We thus have developed a simple and efficient algorithm for obtaining stable flow patterns for use globally as the higher level in our 2 level adaptive routing procedure. In the coming months we hope to implement this procedure and experiment with it.

In an allied study we investigated a pattern for placing virtual calls on a network. A simulation program was written to directly observe the dynamic performance of an algorithm which loads calls on alternate routes according to the following algorithm:

1. Load each incoming call onto the route currently carrying the smallest number of calls. (The number of calls carried by a route is defined for the purposes of this algorithm to be the number of calls on the first link in the route.)
2. If there is a tie among several routes in a set, S , select route i with probability $P_i(S)$.

The simulation was written to provide us with a first glimpse of the dynamic performance of such a procedure as a guide for further research in this area. We thus wanted to keep it as simple as possible and considered a 3 node 6 link network as shown in Figure 5a with symmetric requirements. The program can easily be expanded to consider more general cases but we chose this simple one initially in order not to obscure the basic results.

Calls arrive at each node at a rate λ (Poisson) and are served at rate μ (exponential) by the links, i.e., have exponential duration with average length $1/\mu$. Each call has a choice of a 1 hop path or a 2 hop path. A call arriving at a node is equally likely to be destined for either other node. Thus, there is total symmetry in the system. It should be noted that a call taking a 2 hop path occupies 2 links but remains in the system for time $1/\mu$ on average (not $2/\mu$).

The simulation is straightforward. Call arrivals are generated randomly and arriving calls are routed according to the algorithm given above. The number of calls taking the 1-hop and 2-hop routes were recorded for each run. A parameter, α , determined the probability of taking the 1-hop route when there was a tie between the 2 routes ($\alpha = \text{Prob \{using the 1-hop route in case of a tie\}}$).

For $\alpha = .5$ the fraction of calls taking the 1-hop route was, not surprisingly, very close to $\frac{1}{2}$. For $\alpha = 0$, however, the fraction varied. For $\lambda/\mu = 1$, 80% of the calls took the 2-hop path. For $\lambda/\mu = 10$, 66% of the calls took the 2-hop path. For $\lambda/\mu = 50$, 65% of the calls took the 2-hop path. For $\alpha = 1$, the results (fraction on 1-hop versus fraction on 2-hop paths) reversed relative to the results for $\alpha = 0$.

We thus conclude that we have some control but not total control over the routing via α which only operates during a tie. The control gets greater for systems with a smaller number of calls in progress, as evidenced by the results for smaller values of λ/μ (which is directly related to the number of calls in the system). Observations of the number of calls in the system at various points in a simulation run led to the conclusion that the system is 'stable', i.e., that the link loads reach a stable level and remain close to that point and close to one another.

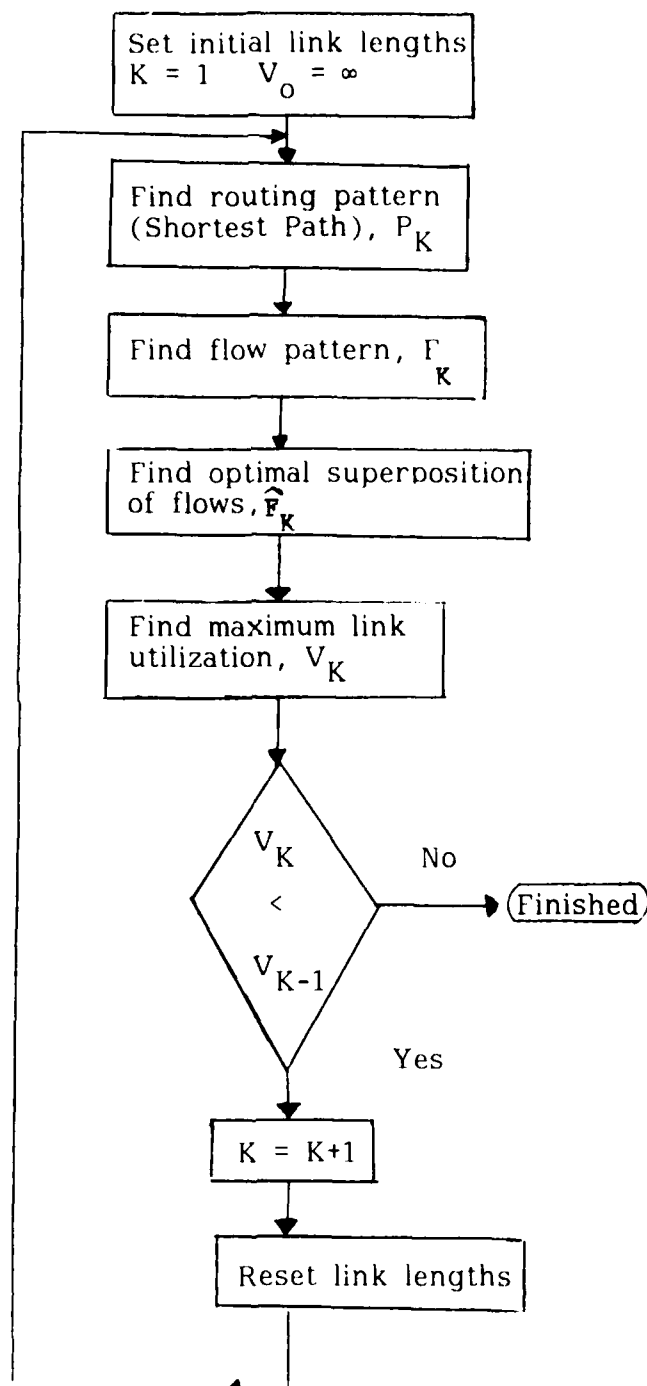
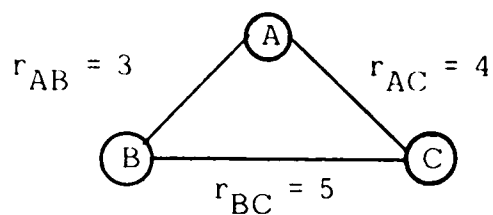


FIGURE 1
OPTIMIZATION PROCEDURE



$$C_{AB} = C_{AC} = C_{BC} = 10$$

FIGURE 3

NETWORK AND REQUIREMENTS

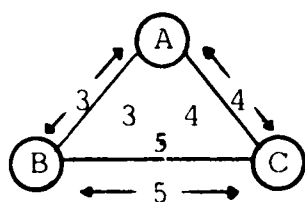


FIGURE 4a
Routing Pattern 1
= Flow Pattern 2

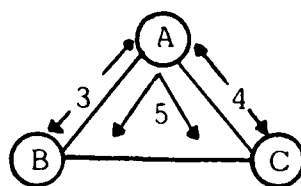


FIGURE 4b
Routing Pattern 2

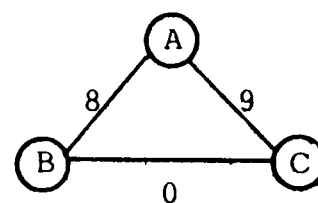


FIGURE 4c
Flow Pattern 2

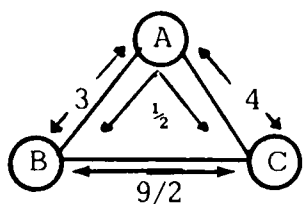


FIGURE 4d
Superposition
of Routing
Patterns

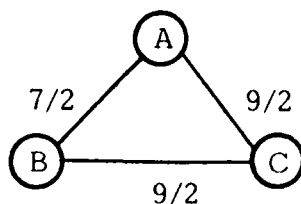


FIGURE 4e
Superposition
of Flows

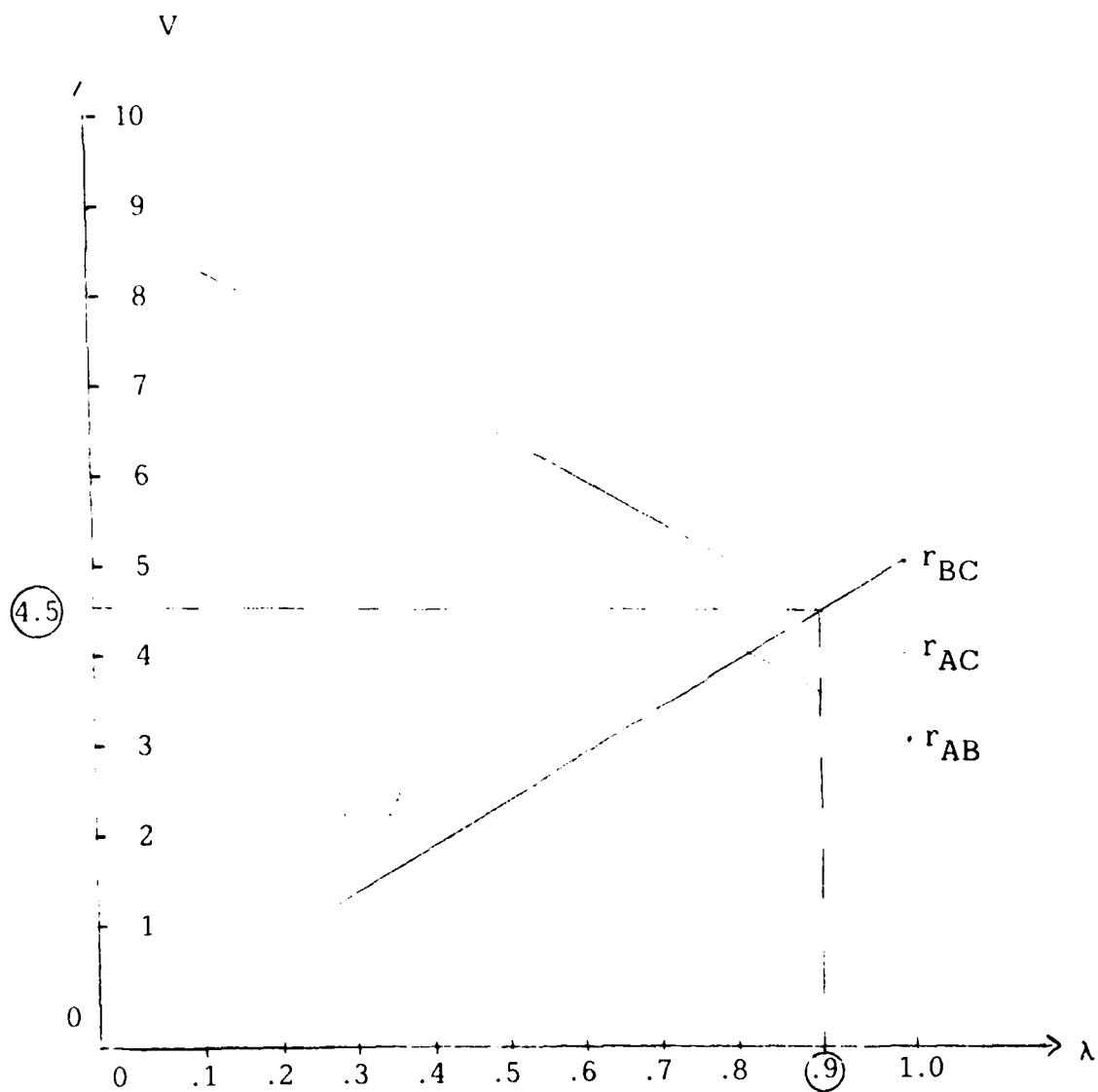


FIGURE 5
MAXIMUM LINK FLOW (V) VERSUS λ

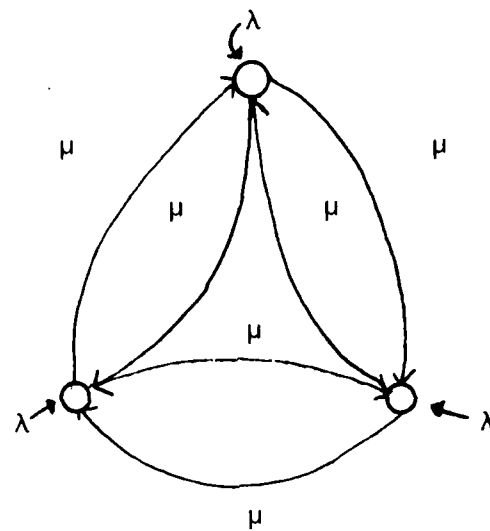


FIGURE 5a
A SYMMETRICAL NETWORK

B.4 Decentralized Dynamic Clearing of Congested Multi-
Destination Networks

Sarachik

Allerton Conference on Communication, Control, and
Computing, October 1981, Urbana

DECENTRALIZED DYNAMIC CLEARING OF CONGESTED MULTI-DESTINATION NETWORKS*

P. E. SARACHIK
Department of Electrical Engineering
Polytechnic Institute of New York
Brooklyn, N.Y. 11201

ABSTRACT

This paper considers the problem of routing traffic with many destinations through a congested network. The problem is formulated as a very general dynamic multicommodity flow problem. The optimal control problem which results is a very difficult one to solve analytically (even for open loop solutions) because of the presence of both state variable constraints and delays in the system dynamics.

The difficulties inherent in the centralized problem can be avoided by using a decentralized approach to obtain dynamic routing strategies. Some of the results obtained using this approach are presented.

INTRODUCTION

Consider a network made up of N nodes connected by L directed links along which the rate of traffic flow can be controlled. M of the nodes are destination nodes toward which varying amounts of traffic must be routed in accordance to the traffic demand.

When the traffic demand exceeds the capacity of the network to meet this demand for service, congestion builds in the network. In the model used here (first suggested by Gazis^[1]) the congested network is viewed as a store and forward network in which all delays due to congestion are treated by assuming that queues can build inside the network at the network nodes. Traffic is assumed to move between nodes at a constant velocity (this corresponds a fixed travel time between nodes) and is stored at the nodes in queues until it can proceed onward toward its destination.

The degree of congestion is measured by the aggregate of all queues in the network. The problem considered here is that which occurs after a "rush hour" or period of heavy demand when the demand has fallen below the capacity but congestion is still present.

Using the store and forward model the queue dynamics can be expressed

*The research was supported by the National Science Foundation under grant ENG-77-14898, and partially by NATO grant #1425.

as

$$\dot{q}_n^m(t) = r_n^m(t) - \sum_{k=1}^N u_{nk}^m(t) + \sum_{j=1}^N u_{jn}^m(t - s_{jn}^m) \quad (1)$$

for $n=1, \dots, N$
 $m=1, \dots, M$
and $n \neq m$.

where q_n^m denotes a continuous state variable which approximates the length of the traffic queue at node n having node m as its destination; u_{nk}^m denotes the rate of traffic flow to destination m on link (nk) ; r_n^m is the traffic demand entering at n with destination m and s_{jn}^m is the delay on link (j,n) encountered by traffic to m . For deterministic input demand, the objective is to clear the congestion (so that $q_n^m(T)=0$ for all n and m) at a specified time T which is large enough to achieve this objective while minimizing the aggregate cost of congestion

$$J\{u\} = \int_{t_0}^T \sum_{m=1}^M \sum_{n=1}^N [q_n^m(t) + \sum_{k=1}^N s_{nk}^m u_{nk}^m(t)] dt \quad (2)$$

In this integral the q_n^m terms give the aggregate delay due directly to the congestion while the flow terms produce a cost or aggregate delay associated with the link flows. The constraints to be met

$$q_n^m(t) \geq 0; u_{nk}^m(t) \geq 0 \text{ and } \sum_{m=1}^M u_{nk}^m(t) \leq C_{nk} \quad (3)$$

for all m, n, k and t

impose the requirements that queue lengths and flows must be non-negative and that each link has a finite capacity C_{nk} .

D'Ans and Gazis^[2] have shown, that an optimal open loop control for this centralized control problem can be computed by discretizing the time appropriately. When this is done, the problem can be expressed as a linear programming problem and an optimal open loop solution can be computed for any initial conditions.

This problem with all s_{nk}^m equal to zero was considered by Moss and Segall^[3]. They present a set of necessary conditions via the maximum principle but they offer a general procedure for finding the flows in feedback form, only for single destination networks with zero input demand.

In seeking a closed loop solution to this centralized optimal control problem, one runs into considerable difficulty because even necessary conditions are not available for problems having both state variable constants and time-delay systems.

A DECENTRALIZED APPROACH

Most of the difficulties inherent in the centralized formulation can be avoided by adopting a decentralized approach to this problem. In general the decentralized approach to controlling large scale systems is to use a number of low level controllers operating on local information to control portions of a large system. This local information is supplemented by some interchange of information between local controllers or between local controllers and a supervisory controller. How well the overall system operates, depends on what information is exchanged. The local controllers are designed to perform their limited tasks optimally. For the dynamic routing problem considered here, the local controllers are node level controllers which must assign traffic flow on the links leaving the node based on the information available locally.

The Local Optimization Problem

The optimization problem faced at a typical node is based on the queue dynamics

$$\dot{q}^m(t) = R^m(t) - \sum_{\ell=1}^{\bar{L}} u_{\ell}^m(t) \quad (4)$$

where R^m is the total rate of traffic arrivals (from both upstream nodes and from outside the network) which have the destination m and there are $\bar{L}$ links which exit from the node. We wish to find the link flows $u_{\ell}^m(t)$ as a function of the local state vector $q(t)$, to minimize

$$J\{u\} = \int_{t_0}^T \sum_{m=1}^M [q^m(t) + \sum_{\ell=1}^{\bar{L}} S_{\ell}^m u_{\ell}^m(t)] dt \quad (5)$$

where S_{ℓ}^m denotes the cost or delay per unit of flow on link ℓ to reach destination m . The constraints are as in (3) and $q(T) = 0$. It is apparent that since all incoming demand has been consolidated into one term, the local controller is not attempting to control a time-delay system. A feedback solution can be thus be sought using Hamilton-Jacobi theory. The minimum cost due to congestion can be expressed as

$$V(q(t_0)) = \min_u \{J\{u\} - J^*\} = \min_u \int_{t_0}^T \sum_{m=1}^M [q^m(t) + \sum_{\ell=1}^{\bar{L}} S_{\ell}^m (u_{\ell}^m(t) - u_{\ell}^{m*}(t))] dt \quad (6)$$

where u_{ℓ}^{m*} are the optimal non-congested flows (i.e. the flows which minimize J in (5) when $q(t_0) = 0$) and J^* is the corresponding minimum non-congested cost on $[t_0, T]$. Note that $R^m(t) = \sum_{\ell=1}^{\bar{L}} u_{\ell}^{m*}(t)$. When the

R^m are constants, then the minimum cost function $V(q)$ satisfies the Hamilton-Jacobi equation

$$\min_{\underline{u}} \sum_{m=1}^M [q^m(t) + \sum_{\ell=1}^{\bar{L}} (S_{\ell}^m - \frac{\partial V}{\partial q^m})(u_{\ell}^m(t) - u_{\ell}^{m*})] = 0 \quad (7)$$

where $V(0) = 0$ and the minimization is subject to constraints as in (3). If the $V(q)$ which satisfies (7) and boundary conditions imposed by state constraints can be found then this would produce necessary and sufficient conditions for the optimal flows.

A complete solution to the local optimum control problem is not yet available. A method for obtaining an approximate solution was presented by Özgüner and Sarachik^[4]. However the exact solution is known only for two special cases.

Two Special Cases

The Single Destination Case ($M=1$). Chu^[5] first solved this local dynamic routing problem using a geometric approach. Later Sarachik and Özgüner^[6] obtained the same solution using the approach above. Since (7) must be minimized at each time instant, this is equivalent to choosing the $u_{\ell}(t)$ to minimize

$$\sum_{\ell=1}^{\bar{L}} (S_{\ell} - \frac{\partial V}{\partial q}) u_{\ell}(t) \quad (8)$$

subject to the constraint $0 \leq u_{\ell}(t) \leq C_{\ell}$. This says that we should choose

$$u_{\ell}(t) = \begin{cases} C_{\ell} & \text{for } S_{\ell} < \frac{\partial V}{\partial q} \\ 0 & \text{for } S_{\ell} \geq \frac{\partial V}{\partial q} \end{cases} \quad (9)$$

Thus if the links are numbered according to the ordering $S_1 \leq S_2 \leq \dots \leq S_{\bar{L}}$ and if K is the largest index ℓ satisfying $S_{\ell} < (\partial V / \partial q)$ we see that on links $\ell=1, 2, \dots, K$ the full capacity should be used to carry traffic flow whereas on links $\ell=K+1, \dots, \bar{L}$ no traffic should flow. Substituting this result in (8) gives

$$u_{\ell}(t)_{\text{opt}} = \begin{cases} C_{\ell} & \text{for } q(t) > Y_{\ell} \\ 0 & \text{for } q(t) \leq Y_{\ell} \end{cases} \quad (10)$$

where the thresholds Y_{ℓ} are given by

$$Y_{\ell} = \sum_{i=1}^{\ell} (C_i - u_i^*)(S_{\ell} - S_i) \quad (11)$$

for $\ell=1, \dots, \bar{L}$.

Multi-destination with $L=2$. Sarachik^[7] obtained a solution to (7) with arbitrary M , for a node with only two exit links. The method of solution involved finding the solutions along the state space axes (these are one dimensional problems) and using the results as boundary conditions for the admissible state space $q \geq 0$. The derivation can be found in [7]. The results show that the admissible state space is divided into non-intersecting regions by planes. In each region the optimal flows must satisfy a necessary and sufficient condition. The regions can be found by simple threshold tests on aggregate queue lengths. Specifically with $\beta_m \triangleq S_2^m - S_1^m$ (the additional cost of using link 2) the destinations are ordered according to $\beta_1 \geq \beta_2 \geq \dots \geq \beta_K \geq 0 \geq \beta_{K+1} \geq \dots \geq \beta_M$ (note that traffic to destinations 1 and K prefer to use link 1). Thresholds are calculated using

$$X_1(k) = X_1(k+1) + \gamma_1(k) [\beta_k - \beta_{k+1}] \text{ for } k = K, \dots, 1 \quad (12)$$

with $X_1(K+1) \triangleq \beta_{K+1} \gamma_1(K)$ and

$$X_2(k) = X_2(k-1) + \gamma_2(k-1) [\beta_{k-1} - \beta_k] \text{ for } k = K+1, \dots, M \quad (13)$$

with $X_2(K) \triangleq -\beta_K \gamma_2(K)$. The $\gamma_i(k)$ denotes the capacity available on link i for emptying queues when the demand R^m for destinations $m=1$ to k is routed on link 1 and the rest on link 2 (note that for $k \neq K$ this means some traffic will be diverted away from its preferred link). The composite queue lengths are next defined as

$$Q_1(k) \triangleq \sum_{m=1}^k q^m ; \quad Q_2(k) \triangleq \sum_{m=k+1}^M q^m \quad (14)$$

and the test variable $T_1(k) = (\gamma_2(k)Q_1(k) - \gamma_1(k)Q_2(k))/\gamma_2(k)$ is found for all those $k \leq K$ which also make $\gamma_2(k) > 0$. For $k \leq K$ if $X_1(k+1) < T_1(k) \leq X_1(k)$ and $Q_1(k), Q_2(k) > 0$ the optimum flows satisfy

$$\begin{aligned} \text{a) } U_1(k) &= C_1; u_1^m = 0 \text{ for } m > k \\ \text{b) } U_2(k) &= C_2; u_2^m = 0 \text{ for } m \leq k \end{aligned} \quad (15)$$

where the composite flows are defined as

$$U_1(k) \triangleq \sum_{m=1}^k u_1^m ; \quad U_2(k) \triangleq \sum_{m=k+1}^M u_2^m \quad (16)$$

whereas if $T_1(k-1) \leq X_1(k) < T_1(k)$ the optimum flows must satisfy (15a) and

$$U_2(k) + u_2^k = C_2 ; \quad u_2^m = 0 \text{ for } m \leq k-1. \quad (17)$$

The only difference between (15b) and (17) is that (17) permits a non-zero u_2^k . For a state on the boundary of the admissible region (i.e. when $Q_1(k)=0$ for some k or i) the conditions (15) and (17) must be modified slightly. In this case the $U_1(k)$ must equal the corresponding aggregate demand.

If the above tests fail then for $k > K$ the test variable $T_2(k) = (\gamma_1(k) Q_2(k) - \gamma_2(k) Q_1(k)) / \gamma_1(k)$ is formed for all those k for which $\gamma_1(k) > 0$. For $Q_1(k), Q_2(k) > 0$ if $X_2(k) < T_2(k) \leq X_2(k+1)$ the optimum flows must satisfy (15) whereas if $T_2(k) \leq X_2(k) < T_2(k+1)$ the optimum flows satisfy (15a) and (17). It should be noted that (15) or (17) do not specify the flows uniquely so many optimal implementations are possible.

Coordination of Local Control

In order for the node level controllers to implement the local control strategies, information must be supplied to it about the parameters S_ℓ^m . This can be done for each link exiting the node by forming $S_\ell^m = s_\ell^m + \phi_j^m$ where s_ℓ^m is the delay (or cost) of traversing link ℓ to the first downstream node j and ϕ_j^m denotes an estimate of the delay (or cost) of reaching destination m from node j . These ϕ_j^m cost-to-go terms must be computed at each node and transmitted to the upstream nearest neighbor. The way these terms are calculated greatly affects the performance of the overall network. Chu^[5] using a capacity weighted average for a single destination network ($M=1$) defined

$$\phi_j^m(t) = \frac{q_j^m(t) + \sum_{\ell=1}^{\bar{L}} S_\ell^m C_\ell}{\sum_{\ell=1}^{\bar{L}} C_\ell} \quad (18)$$

and had to restrict the application to non-cyclic networks. Whereas if C_ℓ is replaced by the excess capacity $\gamma_\ell \triangleq C_\ell - u_\ell^*$ the strategy can be extended to networks containing cyclic paths since it can be proven that no cyclic flows will occur^[6]. Sarachik^[8] using flow weighted averages (i.e. replace C_ℓ in (18) by u_ℓ^m) for the multi-destination case required the non-cyclic restriction. This restriction was removed in [4] by using a certain kind of excess capacity average. The question of what is the best coordination scheme is still open.

CONCLUSION

This paper has described the general problem of using dynamic routing to clear congestion from a congested network. A decentralized strategy was described which utilizes local node level controllers. The optimal local strategies were presented for 2 special cases when traffic demand is constant. Research is being done to extend the local results for traffic demand modeled by a Poisson arrival process and to obtain the exact optimum routing for M destinations with L exit links. More research is also needed to determine whether an optimal coordination strategy can be found.

REFERENCES

- [1] Gazis, D.C., "Modelling and optimal control of congested transportation systems", Networks, Vol. 4, pp. 113-124, (1974).
- [2] G.C. D'ans and D.C. Gazis, "Optimal Control of Oversaturated Feed - Forward Transportation Networks," IBM Research Report, #RC4543; 1973.
- [3] F.H. Moss and A. Segall, "An Optimal Control Approach to Dynamic Routing in Data-Communication Networks," Report EE312. Faculty of EE, Technion; 1977 (To appear in IEEE Trans. Aut. Control).
- [4] Özgüner, U. and Sarachik, P.E., "Decentralized Routing in Congested Multi-Destination Traffic Networks" Proc. of 1981 ICCS Conference, pp. 1006-1009; 1981.
- [5] Chu, K.C., "Decentralized real-time control of congested traffic networks", IBM Research Report, RC-6337, Dec. 1976.
- [6] Sarachik, P.E. and Özgüner, U., "On decentralized routing in traffic networks," Univ. of Toronto, Dept. of Electrical Engineering, Systems Control Report #8003, April 1980. To appear in IEEE Trans. Aut. Cont.
- [7] Sarachik, P.E., "Optimal local dynamic routing for clearing multi-destination networks," EE Dept. Report. Polytechnic Inst. of N.Y.; Jan. 1981.
- [8] Sarachik, P.E., "Clearing of congested multi-destination networks," Proc. of Engr. Found. Conf. on Research Dev. in Comp. Control of Urban Traffic Syst., Feb. 1979.

B.5 An Effective Local Dynamic Strategy to Clear Congested
Multi-Destination Networks

Sarachik

IEEE Transactions on Automatic Control, April 1982

An Effective Local Dynamic Strategy to Clear Congested Multidestination Networks

PHILIPPE SARACHIK

Abstract—The solution of a local routing problem for a congested node with two alternative paths to M destinations of a traffic network is presented. The solution shows that the admissible state space is divided into a finite number of regions within which a different set of optimal flow conditions must be satisfied. A specific realization of these optimal conditions is presented which can be utilized by local node controllers in decentralized control strategies for larger networks.

I. INTRODUCTION

The problem of routing traffic through networks from nodes of origin to destination nodes is an important problem common to the field of traffic control and data-communication networks. Much of the early work dealing with the traffic assignment problem in both fields was restricted to finding an optimum steady-state routing (static strategies) under non-congested conditions. In 1974 Gazis [1] proposed that a store-and-forward network could be used to model congested transportation systems. In this model the delay due to congestion is associated with the waiting time on queues. For a single destination node and using the aggregate delay in the network as the performance criterion, D'Ans and Gazis [2] obtain an open-loop solution by discretizing in time and Chu and Gazis [3] obtain a dynamic feedback solution for a simple network with two alternate routes. At about the same time, Segall and Moss [4]–[6] used a similar model to study the dynamic routing problem in congested multidestination data-communication networks. They assumed that the delays encountered in transiting network links are zero, so the resulting queue dynamics are governed by ordinary differential equations. Using the maximum principle, they obtain a set of necessary conditions for the optimal solution and propose a constructive algorithm for finding the optimal centralized control.

When transit delays are present, the queue dynamics are governed by differential difference equations and the general centralized optimal dynamic routing problem becomes formidable, since state variable constraints are also present. Most of the difficulties inherent in the centralized problem can be avoided by seeking a decentralized control solution.

The decentralized approach to clearing congestion from traffic networks has been discussed in three recent papers [7]–[9]. In these papers the problem of finding real-time dynamic feedback routing for traffic is formulated as an optimal control problem and a decentralized structure for the solution is utilized. Chu [7] considered a single destination acyclic network and presented a solution based on the interconnection of local controllers which implement route selections at each node by choosing among K alternate routes to the single destination. Sarachik and Orguner [9] considered single destination cyclic networks. They show that the optimal local routing algorithm, when used in their decentralized strategy, leads to loop-free flows in the network even though cyclic paths are present. Sarachik [8] considered multidestination acyclic networks and presented a decentralized strategy for that case. Local routing strategies for two specific simple subnetworks were utilized in the decentralized strategy to illustrate the control of a larger network.

In order to make the decentralized routing strategy of [8] for multidestination networks more generally useful, this correspondence considers the problem of designing a local feedback controller for a typical node of a multidestination network when the node has two exit links (see Fig. 1). An optimization problem is formulated and solved for this local problem

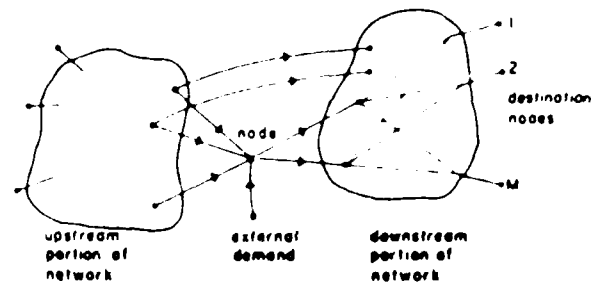


Fig. 1 A typical node with two exit links.

when the demand for service is constant. It is shown that the state space is divided into regions separated by linear manifolds. The closed form conditions for the optimal control are given for each region. These conditions do not specify a unique optimal control so a specific optimal realization is presented. This realization is in the form of a feedback control which can be easily implemented in real time.

In using any decentralized control strategy to route traffic through a complex network, decisions must be made locally at each node on how to best allocate traffic to branches leaving the node. These decisions will affect the traffic levels at downstream nodes and thus must be based on information, provided to the local node controllers, about conditions at other nodes of the network. Questions concerning how the communication between nodes affects the overall behavior of large networks which use decentralized routing are extremely important but are beyond the scope of this paper. References [7]–[9] address some of these questions and indicate how local node controllers can be utilized in a decentralized dynamic routing strategy.

II. THE LOCAL ROUTING PROBLEM

The demand for service is the net traffic flow arriving at a typical node along inbound links and from outside the network. This demand is met by routing the incoming traffic to the exit links. When traffic demand exceeds the capacity of the exit links, queues build in front of the node. When the capacity exceeds the demand, the queue lengths will be decreased. In this paper attention is restricted to the situation which exists toward the end of "rush hours." That is, congestion exists in the form of long queues but the demand is less than the capacity of the exit links so the queues can be emptied and the congestion cleared out.

At a congested node containing queues of traffic for M destinations and having only two exit links, the queue dynamics can be represented by [1], [5]

$$\dot{q}^m(t) = r^m - u_1^m(t) - u_2^m(t) \quad \text{for } m = 1, 2, \dots, M \quad (1)$$

or in vector form

$$\dot{q}(t) = r - u_1(t) - u_2(t)$$

where q^m denotes a continuous state variable which approximates the length of the traffic queue (measured in vehicles, bits, messages, etc.) having node m as its destination; u_i^m is the flow rate on link i destined for m ; and r^m is a constant flow of traffic entering the node destined for m (r is the demand vector).

The problem faced by the node controller is to choose the control vectors u_1 and u_2 at each time instant in order to empty the queues while optimizing some performance measure. A useful measure of this performance [7]–[10] is the aggregate cost (or delay) given by

$$J(u) = \int_0^T \sum_{m=1}^M [q^m(t) + S_1^m u_1^m(t) + S_2^m u_2^m(t)] dt \quad (2)$$

Manuscript received February 21, 1981; revised August 7, 1981. This work was supported in part by the National Science Foundation under Grant ENG-77-148-98 and in part by NATO under Grant 1425.

The author was on leave at the Department of Electronic Communication, Control and Computer Systems, Tel Aviv University, Tel Aviv, Israel. He is with the Department of Electrical Engineering, Polytechnic Institute of New York, Brooklyn, NY 11201.

where S_i^m denotes the cost (or delay) per unit of flow on link i to reach destination m . The time T is any time greater than the time t^* required to empty all queues at the node with minimal cost. Note that the contribution to J due to the q^m terms is the aggregate delay associated with clearing the queues [1], [5], while the other terms represent the aggregate cost (or delay) associated with reaching the destinations from the entrance of the exit links [10].

It should be noted that when the local controller is used in a decentralized setting, the cost parameters S_i^m will be composed of two portions: $S_i^m = r_i^m + \phi_j^m$ where r_i^m is the cost (or delay) of traversing link i to the first downstream node j and ϕ_j^m is a cost (or delay) associated with reaching destination m from node j . The node controller will receive the information ϕ_j^m from its downstream neighbors and update the values S_i^m . In operation the node controller must be able to adapt to these updates. Thus, a real-time feedback solution of the local optimization problem is essential.

In this problem the queues can be emptied, since we consider only the case for which the total traffic demand $R = \sum_{m=1}^M r_i^m$ entering the node is less than the combined capacity of the two exit links ($C_1 + C_2$). The optimal noncongested cost J^* can be defined as the minimum of (2) when the initial state $q(t_0)$ is zero. Thus, for the same T as in (2),

$$J^* = \int_{t_0}^T \sum_{m=1}^M (S_1^m u_1^m + S_2^m u_2^m) dt \quad (3)$$

where u_i^m are the optimal noncongested flows and $y_1^* + y_2^* = r$. The minimum cost due to the congestion at the node is obtained by subtracting J^* from the minimum of (2). This gives

$$\begin{aligned} V(q(t_0)) &= \min_{y_1, y_2} [J(y) - J^*] \\ &= \min_{y_1, y_2} \int_{t_0}^T \sum_{m=1}^M [q^m(t) + S_1^m(u_1^m(t) - u_1^m) + S_2^m(u_2^m(t) - u_2^m)] dt \end{aligned} \quad (4)$$

where the minimization is carried out subject to the constraints

$$q(t) \geq 0; \quad y_i(t) \geq 0; \quad \sum_{m=1}^M u_i^m(t) \leq C_i \quad \text{for } i=1,2; \quad t_0 \leq t \leq T \quad (5)$$

and

$$q(T) = 0.$$

These constraints impose the requirements that queue lengths and flows must be nonnegative, that each link has a finite capacity, and that the queues must be empty at the terminal time. Note that for a given $q(t_0)$ and any $T \geq t^*$ (T could be ∞) the same minimum cost V is obtained, so V does not depend on the time-to-go ($T - t_0$). Also, system (1) is time invariant and the coefficients in (2) are constant. Therefore, V is independent of the starting time t_0 . Applying Hamilton-Jacobi theory, we know that the minimum cost $V(q)$ for system (1) satisfies the partial differential equation

$$\begin{aligned} \min_{y_1, y_2} \left\{ \sum_{m=1}^M [q^m(t) + S_1^m(u_1^m(t) - u_1^m) + S_2^m(u_2^m(t) - u_2^m)] \right. \\ \left. + \frac{\partial V(q)}{\partial q} [r - y_1(t) - y_2(t)] \right\} = 0 \quad (6) \end{aligned}$$

where the minimization is subject to constraints (5). Using $r = y_1^* + y_2^*$ this equation can be rewritten as

$$\min_{y_1, y_2} \left\{ \sum_{m=1}^M \left[q^m - \left(\frac{\partial V}{\partial q^m} - S_1^m \right) (u_1^m - u_1^m) - \left(\frac{\partial V}{\partial q^m} - S_2^m \right) (u_2^m - u_2^m) \right] \right\} = 0 \quad (7)$$

(the dependence of q^m and u_i^m on t is not indicated explicitly). Note that since (7) is linear in the y_i 's, the minimum lies on the constraint boundaries, and since two of these boundaries involve sums, an optimal solution need not be unique.

By finding the $V(q)$ which satisfies (7) and the boundary conditions imposed by the state constraints, we can obtain explicit necessary and sufficient conditions for the optimal flows.

III. OPTIMAL FLOW CONDITIONS

In this section the conditions for optimal flow are presented. A detailed description of how these conditions are derived is given in [11]. To simplify the presentation, the destinations are numbered so that for $\beta_m \triangleq S_2^m - S_1^m$

$$\beta_1 > \beta_2 > \beta_3 \cdots \beta_K > 0 > \beta_{K+1} > \beta_{K+2} \cdots \beta_{M-1} > \beta_M.$$

With this numbering, traffic to destinations 1 through K prefers link 1 while link 2 is preferred by traffic to destinations $K+1$ through M . Also, if it becomes necessary to divert traffic from link 1 to link 2, traffic to k should not be diverted until all traffic to $m > k$ has been diverted. Conversely, when traffic must be diverted from link 2 to link 1, traffic to k should not be diverted until all traffic to $m < k$ has been diverted.

It is convenient to define the excess capacities as

$$\gamma_1(K) \triangleq C_1 - \sum_{m=1}^K u_1^m; \quad \gamma_2(K) \triangleq C_2 - \sum_{m=1}^M u_2^m \quad (8)$$

and

$$\gamma_2(k) \triangleq \max \left[0, \gamma_2(K) - \sum_{m=k+1}^K u_1^m \right]; \quad \gamma_1(k) = \gamma - \gamma_2(k) \quad \text{for } k=0, \dots, K-1 \quad (9)$$

$$\gamma_1(k) \triangleq \max \left[0, \gamma_1(K) - \sum_{m=k+1}^M u_2^m \right]; \quad \gamma_2(k) = \gamma - \gamma_1(k) \quad \text{for } k=K+1, \dots, M \quad (10)$$

where $\gamma \triangleq \gamma_1(K) + \gamma_2(K)$. Note that (8) defines the capacity available on each link for clearing queues after the steady-state demand is accommodated, when no steady-state flow is diverted. When C_1 and C_2 are each large enough to accommodate the preferred demands, then $u_1^m = r^m$, $u_2^m = 0$ for $m \leq K$ and $u_1^m = 0$, $u_2^m = r^m$ for $m > K$, and both $\gamma_1(K)$ and $\gamma_2(K)$ are greater than zero. If one exit link has insufficient capacity C_i to accommodate its preferred traffic, then as much preferred traffic as possible is routed on this link (with priorities governed by the β_m) giving $\gamma_i(K) = 0$ and the rest is diverted to the other link. It is thus quite simple to determine the values u_i^m for all m and i for this problem and to form the sets $\gamma_i = [\gamma_i(0), \dots, \gamma_i(M)]$ for $i=1,2$ using (8), (9), and (10). The $\gamma_1(k)$ and $\gamma_2(k)$ values from (9) are the excess capacities available for emptying queues when demand for destinations $(k+1)$ to K is diverted from link 1 to 2, whereas the values from (10) are the excess capacities when the demand for destinations $K+1$ to M is diverted from link 2 to 1. Diversion of traffic from one link to another can eventually use up the capacity of the link receiving the diverted traffic. To keep track of this, we define

$$\underline{k} \triangleq \text{the smallest value of } k \text{ such that } \gamma_2(k) > 0$$

$$\bar{k} \triangleq \text{the largest value of } k \text{ such that } \gamma_1(k) > 0$$

$$\bar{K} \triangleq \min[\bar{k}, \bar{k}], \quad \underline{K} \triangleq \max[\underline{k}, \underline{k}] \quad (11)$$

A set of threshold constants can now be defined recursively by

$$X_1(k) = X_1(k+1) + \gamma_1(k)[\beta_k - \beta_{k+1}] \quad \text{for } k = K, \dots, 1 \quad (12)$$

with $X_1(K+1) \triangleq \beta_{K+1}$, $\gamma_1(K)$ (so $\gamma_1(k) \geq 0$ is nondecreasing as k decreases from K to 1) and

$$X_2(k) = X_2(k-1) + \gamma_2(k-1)[\beta_{k-1} - \beta_k] \quad \text{for } k = K+1, \dots, M \quad (13)$$

with $X_2(K) \triangleq -\beta_K \gamma_2(K)$ (so $X_2(k) \geq 0$ is nondecreasing as k increases from $K+1$ to M). Composite queues are defined as

$$Q_1(k) \triangleq \sum_{m=1}^k q^m; \quad Q_2(k) \triangleq \sum_{m=k+1}^M q^m = Q - Q_1(k) \quad (14)$$

where Q is the sum of all queue lengths. The composite flows on each link are

$$U_1(k) \triangleq \sum_{m=1}^k u_1^m; \quad U_2(k) \triangleq \sum_{m=k+1}^M u_2^m. \quad (15)$$

Note that $k=0$ means $Q_1(0)=0$ and $U_1(0)=0$ while $k=M$ means $Q_2(M)=0$ and $U_2(M)=0$; for notational convenience the explicit dependence of these quantities on r has been omitted. The upper flow bounds are

$$\bar{C}_1(k) \triangleq \min \left[C_1, \sum_{m=1}^k r^m \right]; \quad \bar{C}_2(k) \triangleq \min \left[C_2, \sum_{m=k+1}^M r^m \right]. \quad (16)$$

The solution of the Hamilton-Jacobi equation (7) shows [11] that the admissible state space $q \geq 0$ is divided into disjoint regions which fill the space. In each region $V(q)$ is given by a different quadratic function, but it is continuous across the region boundaries. These regions can be expressed conveniently in terms of variables

$$\begin{aligned} T_1(k) &\triangleq Q_1(k) - \frac{\gamma_1(k)}{\gamma_2(k)} Q_2(k) \quad \text{defined for } k \leq \bar{k} \\ T_2(k) &\triangleq Q_2(k) - \frac{\gamma_2(k)}{\gamma_1(k)} Q_1(k) \quad \text{defined for } k \leq \bar{k}. \end{aligned} \quad (17)$$

The regions are

$$R_{k,k} = \left\{ q: \begin{aligned} &X_1(k+1) < T_1(k) < X_1(k) \quad \text{for } k \leq k < \min[K-1, \bar{k}] \\ &X_2(k) < T_2(k) < X_2(k+1) \quad \text{for } \max[K+1, \underline{k}] < k \leq \bar{k} \end{aligned} \right\} \quad (18)$$

$$R_{K,K} = \left\{ q: \beta_{K+1} < \frac{Q_1(K)}{\gamma_1(K)} - \frac{Q_2(K)}{\gamma_2(K)} < \beta_K \right\}. \quad (19)$$

In (19) (recall that K is the number of destinations which prefer link 1), if $K=0$ the upper bound is zero but if $K=M$ the lower bound is zero.

$$R_{k,k-1} = \left\{ q: \begin{aligned} &T_1(k) > X_1(k) && \text{for } k = \underline{k} < K \\ &T_1(k-1) < X_1(k) < T_1(k) && \text{for } \underline{k} < k < \bar{k} \\ &T_1(k-1) < 0 && \text{for } k = \bar{k}+1 < K \\ &T_2(k) < 0 && \text{for } k = \underline{k} > K+1 \\ &T_2(k) < X_2(k) < T_2(k-1) && \text{for } \underline{k} < k < \bar{k} \\ &T_2(k-1) > X_2(k) && \text{for } k = \bar{k}+1 > K+1 \end{aligned} \right\} \quad (20)$$

The region in which $q(t)$ lies and the index value k is obtained by determining which of the inequalities (18), (19), (20) the state vector $q(t)$ satisfies

When the state vector q lies in a region $R_{k,k}$ optimal flows must satisfy the condition

$$U_1(M) = U_1(k) = \begin{cases} \bar{C}_1(k) & \text{if } Q_1(k) = 0 \\ C_1 & \text{otherwise} \end{cases} \quad (21a)$$

and

$$U_2(0) = U_2(k) = \begin{cases} \bar{C}_2(k) & \text{if } Q_2(k) = 0 \\ C_2 & \text{otherwise} \end{cases} \quad (21b)$$

and the restriction

$$u_1^m + u_2^m \leq r^m \quad \text{when } q^m = 0. \quad (21c)$$

Note that $U_1(M) = U_1(k)$ means that $u_1^m = 0$ for $m > k$ and $U_2(0) = U_2(k)$ means $u_2^m = 0$ for $m \leq k$.

If the state vector q lies in a region $R_{k,k-1}$, then optimal flows must satisfy (21a) and

$$U_2(0) = U_2(k-1) = \begin{cases} \bar{C}_2(k-1) & \text{if } Q_2(k-1) = 0 \\ C_2 & \text{otherwise} \end{cases} \quad (21d)$$

as well as restriction (21c). Note that actually the only difference between these conditions is that (21d) allows u_2^k to be nonzero (this is required only when $Q_2(k-1) = 0$) so traffic to destination k could be routed via both links 1 and 2. Since these conditions do not specify the optimal flows uniquely any implementation which is convenient will be optimal. These conditions are necessary and sufficient for optimality because they were obtained from $V(q)$ which satisfies the Hamilton-Jacobi equation (7) and boundary conditions imposed by the state constraint $q \geq 0$ (Reference [11] gives expressions for the minimum cost $V(q)$ in each region and a proof that these regions fill the admissible state space $q \geq 0$).

The region in which q lies could be determined by using a microprocessor or a special chip which implements the conditions (18)–(20) in parallel by comparing the vector of test variables against the vector of threshold values.

IV. A SPECIFIC OPTIMAL ROUTING

In the preceding section the general requirements (21) permit many optimal routing strategies. In this section a specific optimal routing is presented which satisfies these general requirements

For states lying in the regions $R_{k,k}$ consider the flows

$$\begin{aligned} a) \quad u_1^m &= u_1^{m*} + u_2^{m*} + \gamma_1(k) \frac{q^m}{Q_1(k)}; \quad u_2^m = 0 \quad \text{for } 1 \leq m \leq k \\ b) \quad u_1^m &= 0; \quad u_2^m = u_1^{m*} + u_2^{m*} + \gamma_2(k) \frac{q^m}{Q_2(k)} \quad \text{for } k+1 \leq m \leq M \end{aligned} \quad (22)$$

where for $Q_1(k) = 0$ or $Q_2(k) = 0$ the indeterminate term $0/0$ is defined to be zero (thus (21c) is satisfied when $q^m = 0$). We now verify that (22) satisfies the requirements (21a)–(21b). For $Q_1(k) > 0$ and $Q_2(k) > 0$, summing (22) gives

$$\begin{aligned} a) \quad U_1(k) &= \sum_{m=1}^k u_1^{m*} + \sum_{m=1}^k u_2^{m*} + \gamma_1(k) \\ b) \quad U_2(k) &= \sum_{m=k+1}^M u_1^{m*} + \sum_{m=k+1}^M u_2^{m*} + \gamma_2(k). \end{aligned} \quad (23)$$

Now $\gamma_1(k) > 0$ and $\gamma_2(k) > 0$ (this is necessary for $R_{k,k}$ to exist) so for $k < K$ we have $u_2^m = 0$ for $m \leq k$ and $u_1^m = 0$ for $m > K+1$. Thus,

$$a) \quad U_1(k) = \sum_{m=1}^k u_1^{m*} + \gamma_1(k) = \gamma_1(K) + \sum_{m=1}^k u_1^{m*} = C_1$$

$$b) \quad U_2(k) = \sum_{m=k+1}^K u_1^m + \sum_{m=k+1}^M u_2^m + \gamma_2(k) = \gamma_2(K) + \sum_{m=k+1}^M u_2^m = C_2 \quad (24)$$

For $k > K$ this is demonstrated in a similar manner. When $Q_1(k) = 0$ the $\gamma_1(k)$ term does not appear so (23a) gives

$$U_1(k) = \sum_{m=1}^k (u_1^m + u_2^m) = \sum_{m=1}^k r^m = \bar{C}_1(k)$$

since $\gamma_1(k) > 0$. Similarly when $Q_2(k) = 0$ the $\gamma_2(k)$ does not appear in (23b)

$$U_2(k) = \sum_{m=k+1}^M r^m = \bar{C}_2(k).$$

For states lying in the regions $R_{k,k-1}$ the situation is complicated by degeneracies which can occur.

A composite expression which gives optimal flows for q in $R_{k,k-1}$ is

$$\begin{aligned} a) \quad u_1^m &= u_1^m + u_2^m + \Gamma_1(k) \frac{q^m}{Q_1(k)}; \quad u_2^m = 0 \quad \text{for } 1 \leq m \leq k-1 \\ b) \quad u_1^k &= u_1^k + \Gamma_1(k) \frac{q^k}{Q_1(k)} + \Delta(k) \\ c) \quad u_2^k &= u_2^k + \Gamma_2(k) \frac{q^k}{Q_2(k-1)} - \Delta(k) \\ d) \quad u_1^m &= 0; \quad u_2^m = u_1^m + u_2^m + \Gamma_2(k) \frac{q^m}{Q_2(k-1)} \quad \text{for } k+1 \leq m \leq M \end{aligned} \quad (25)$$

where the indeterminate term $0/0$ is defined to be zero when either $Q_1(k)$ or $Q_2(k-1)$ is zero so again (21c) is satisfied when $q^m = 0$. The symbols used in (25) are defined as

$$\left. \begin{aligned} \Gamma_1(k) &\triangleq \gamma_1(k-1); \quad \Gamma_2(k) \triangleq \gamma_2(k-1) \\ \Delta(k) &\triangleq -\gamma_2(k); \quad \Delta(k) \triangleq -u_1^k \quad \text{for } k \neq \bar{k} \end{aligned} \right\} \quad \text{for } k \leq K$$

$$\left. \begin{aligned} \Gamma_1(k) &\triangleq \gamma_1(k); \quad \Gamma_2(k) \triangleq \gamma_2(k) \\ \Delta(\bar{k}+1) &\triangleq \gamma_1(\bar{k}); \quad \Delta(k) \triangleq u_2^k \quad \text{for } k \neq \bar{k}+1 \end{aligned} \right\} \quad \text{for } k > K$$

We now verify that (25) gives $U_1(k) = C_1$ and $U_2(k-1) = C_2$ which satisfies the conditions (21a), (21d) for an optimum. When $k = \bar{k} > K$ or $k = \bar{k}+1 \leq K$ or k is in the range $\bar{k} < k < \bar{k}+1$, the $\Gamma_i(k)$ are positive so the result can be shown exactly as before. When $k = \bar{k} \leq K$ and $Q_1(k) > 0$ we find by summing (25) that

$$\begin{aligned} a) \quad U_1(\bar{k}) &= \sum_{m=1}^{\bar{k}} u_1^m + \sum_{m=1}^{\bar{k}-1} u_2^m + \gamma_1(\bar{k}-1) - \gamma_2(\bar{k}) \\ b) \quad U_2(\bar{k}-1) &= \sum_{m=\bar{k}+1}^M u_1^m + \sum_{m=\bar{k}}^M u_2^m + \gamma_2(\bar{k}-1) + \gamma_2(\bar{k}). \end{aligned} \quad (26)$$

Since $\gamma_2(\bar{k}) > 0$ we have $u_1^m = 0$ for $k > K+1$ and since $\gamma_2(\bar{k}-1) = 0$ we have $\gamma_1(\bar{k}-1) = \gamma > 0$ and $u_2^m = 0$ for $m \leq \bar{k}-1$. Thus using (8) and (9)

$$\begin{aligned} U_1(\bar{k}) &= \sum_{m=1}^{\bar{k}} u_1^m + \gamma - \gamma_2(\bar{k}) = C_1 \\ U_2(\bar{k}-1) &= \sum_{m=\bar{k}+1}^K u_1^m + \sum_{m=\bar{k}}^M u_2^m + \gamma_2(\bar{k}) = C_2 \end{aligned} \quad (27)$$

This result can be obtained in a similar manner for $k = \bar{k}+1 > K$.

To show that (25) is optimum when $Q_1(k-1) = 0$ we will show that when this happens the state q can only be in $R_{k,k-1}$ with $\bar{k} \leq K$ and we

have just demonstrated that in this case the composite flows (26) are optimum. By definition (14) for the composite states note that if $Q_1(k-1) = 0$, then $Q_1(m) = 0$ and $Q_2(m) = Q_2(k-1)$ for all $m \geq k-1$. For $k > K$, (20) requires that $\gamma_2(k-1)Q_2(k-1) \leq -\gamma_1(k-1)X_2(k)$ which is clearly impossible since $Q_2(k-1)$, $\gamma_2(k-1)$, and $X_2(k)$ cannot be negative. For $\bar{k} < k \leq K$, (20) requires the impossible condition $Q_1(k-1) \leq X_1(k) < Q_1(k-1)$. We conclude therefore that a state q with $Q_2(k-1) = 0$ can satisfy the conditions (20) for $R_{k,k-1}$ only when $k = \bar{k} \leq K$. A similar argument shows that when $Q_1(k) = 0$ the state can be in $R_{k,k-1}$ only when $k = \bar{k}+1 > K$.

V. CONCLUSIONS

The problem of finding an optimal routing strategy for local node controllers in a multideestination network has been solved for the case of nodes with two exit links and constant demand. It has been shown that the admissible state space is divided into control regions separated by linear manifolds and explicit expressions for the boundaries of these regions are presented. To each region there exists a set of controls which are optimal for the states of that region. A specific feedback realization of the optimal control has also been presented. The implementations (22) and (25) for the optimal flows are not the only ones possible. They are, however, convenient for traffic control purposes because they provide some exit capacity to traffic for each of the destinations and the longer the queue, the larger will be its share of the capacity. They can also be readily implemented using a microprocessor since they consist merely of the sum of steady-state terms and a linear feedback on the ratio of a state variable q^m to the composite state Q_i of which q^m is a part. The steady-state values must be available at the local controller either by local computation or they must be supplied by a central supervisory controller. Note also that if either the demand or the exit capacity changes to new constant values then the γ_i vectors change, so that the boundaries of the control regions and the feedback gains change. This merely requires a redetermination of the region in which q lies. The feedback routing given is therefore adaptive to changes not only in demand, but in capacity as well.

It should be noted that the solution for γ_i constant may not be as restrictive as it appears. For example γ_i could represent the average demand over some interval so it would be constant on that interval. Furthermore, the routing (22) gives constant flows $u_i^m(t)$ as long as the state remains in one region. The routing (25) gives almost constant flows in one region. Thus even in decentralized operation, when all nodes use this local strategy the incoming demand can be considered as piecewise constant over intervals. The feedback routing could track the changes by updating the γ_i vectors.

For data communication networks, the problem solved here could be applicable when there are delays or costs associated with sending messages to the destinations. Of course, it is usually not desirable to mix the messages for the various destinations on a single link, so a different specific implementation would be preferable in such networks.

REFERENCES

- [1] D. C. Gazis, "Modeling and optimal control of congested transportation systems," *Networks*, vol. 4, pp. 113-124, 1974.
- [2] G. C. D'Ans and D. C. Gazis, "Optimal control of oversaturated feed-forward transportation networks," IBM Res. Rep. RC4543, 1973.
- [3] K. C. Chu and D. C. Gazis, "Dynamic allocation of parallel congested traffic channels," in *Trans. and Traffic Theory (Proc. 5th Int. Symp. Trans. Traffic Theory)*, D. J. Hultsch, Ed., Sydney: Reed and Reed, 1974.
- [4] A. Segall and F. H. Moss, "Application of optimal control theory to dynamic routing for computer networks," in *Proc. Joint Automat. Contr. Conf.*, 1976, pp. 541-546.
- [5] A. Segall, "The modeling of adaptive routing in data-communication networks," *IEEE Trans. Commun.*, vol. COM-25, pp. 95-99, Jan. 1977.
- [6] F. H. Moss and A. Segall, "An optimal control approach to dynamic routing in data-communication methods, Part I: Principles, Part II: Geometrical interpretation," *Facultis Elec. Eng. Technol. Reps.* EE 312, EE 319, 1977, 1978.
- [7] K. C. Chu, "Decentralized real-time control of congested traffic networks," IBM Res. Rep. RC 6337, Dec. 1976.
- [8] P. E. Sarachik, "Clearing of congested multi-destination networks," in *Proc. Conf. Res. Directions in Comput. Contr. of Urban Traffic Syst.*, Feb. 1979, pp. 231-250.
- [9] P. E. Sarachik and U. Orguner, "On decentralized routing in traffic networks," *IEEE Trans. Automat. Contr.*, to be published.
- [10] P. E. Sarachik, "Dynamic allocation for clearing a congested multi-destination transportation network," IBM Res. Rep. RC 4777, Apr. 1977.
- [11] —, "Optimal local dynamic routing for clearing congested multi-destination networks," *Dep. Elec. Eng. Polystech. Inst.*, Jan. 1981.

B.6 A Dynamic Alternate Route Strategy for Traffic Networks

Sarachik

IEEE Conference on Decision and Control, December
1982, Orlando

A DYNAMIC ALTERNATE ROUTE STRATEGY FOR TRAFFIC NETWORKS

Philip E. Sarachik

Polytechnic Institute of New York
333 Jay Street
Brooklyn, New York 11210

Abstract

A dynamic alternate route strategy is developed for a typical node of a traffic network. The strategy compares the queue length at the node against an optimal threshold value. When the threshold is exceeded, some of the traffic is diverted along an alternate more costly route to its destination.

The local node level strategy is easily implemented in real time and is also adaptive to changes in route capacities, input rates or the relative delays along the alternate route. It is also suitable for use in a decentralized routing strategy for large networks.

Introduction

The store and forward network model has been found to be very useful for problems concerned with moving traffic through congested networks from origin to destination nodes. With this model, whenever the instantaneous traffic demand exceeds the capacity of the network to use the demand for service, congestion builds in the network in the form of queues which are stored at the network nodes. For a fixed network structure, the problem faced at each node is how to choose the routes along which to send traffic on toward its destination while minimizing some measure of cost or delay and satisfying the capacity constraints of the network links. For stochastic traffic demand most of the recent work has emphasized dynamic or adaptive routing schemes [1-6] which use information about the actual state of the node or network to determine the best routing. In almost all this work a node with L routes to the destination is modelled as L parallel $M/M/1$ queues. The routing decision is made at the time a customer arrives by placing him in one of the L queues. Foschini and Salz [2] show that for high traffic demand a join-the-shortest-queue rule gives average queueing delays approaching that of the $M/M/L$ queue (i.e. average delay is reduced by a factor of L compared to equal splitting). For two alternate routes, Boorstyn and Livne [6] also get performance approaching the $M/M/2$ case by adding a third queue which can use either of the two servers when the dedicated queues become empty.

It would seem that a more direct approach to achieving such reduced queueing delays at the a node is to operate the node using a single queue with L possible servers. The routing decision can then be put off until the moment a customer gets to the head of the queue and must be put into service. In Fig. 1 the box D denotes a decision rule which allocates traffic to the route servers. The decision can then be made on the basis of the latest state information and also on the delays to be encountered along the different routes [1].

This problem has been solved for the case of deterministic input demand [7,8], where it was found that the optimal routing strategy is a thresh-

hold strategy. The purpose of this paper is to investigate the threshold strategy for a discrete queue dynamic model with random Poisson arrivals.

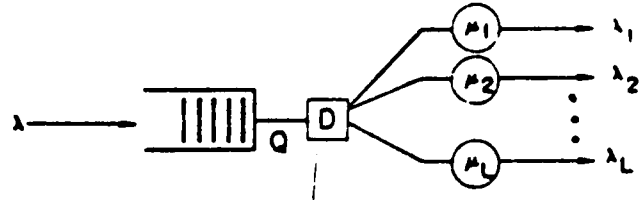


Fig.1 A single queue alternate route model.

A Threshold Strategy for Local Alternate Routing

Consider any source node and a single destination node with L routes between them indexed by $i = 1, \dots, L$. The problem is to assign traffic to these routes in order to minimize the average delay in reaching the destination.

We assume that traffic arrives at an average rate λ with Poisson interarrival times and that the length of each unit of traffic is exponentially distributed with mean $1/\mu$. (For messages the units are in bits and for vehicles the units are in feet). All traffic arriving at the node is assigned to a single queue and the decision on which route to use is made only when it reaches the front of the queue. We have then the situation of a single queue with many potential servers as shown in Fig. 1. Letting Q denote the total number of customers (messages or vehicles) waiting in the queue and in service, Little's formula $PQ = E(Q)/\lambda$ gives the average time spent on the queue and in service. Furthermore all traffic using the i th route will encounter an additional average delay of S_i before reaching the destination. This additional delay includes both transit delays if any and queueing and service delays at nodes along the route to the destination. The average total delay to reach the destination is thus given by

$$T = \frac{1}{\lambda} E(Q) + \frac{1}{\lambda} \sum_{i=1}^L \lambda_i S_i \quad (1)$$

where λ_i is the average rate at which traffic is sent along route i . For a deterministic version of this problem with constant arrival rate, the optimal routing strategy was found to be a threshold strategy [7]. That is to say when the routes are numbered so that

$$S_1 \leq S_2 \leq \dots \leq S_L \quad (2)$$

then the optimal strategy is to use route 1 at full

capacity if and only if Q exceeds a threshold K_i (where $0 = K_1 \leq K_2 \leq \dots \leq K_n$). This means that for $K_i < Q \leq K_{i+1}$ only the first i routes should be used. For very small Q only route 1 (the one with smallest delay) is used and as Q increases the next best alternate routes are added. Work is presently being carried out to prove the optimality of this type of strategy for the stochastic problem considered here. On a heuristic basis one would expect a threshold strategy to be optimal since the first term of (1) is small when all routes are used and the second term is minimized when only route 1 is used. For very small values of Q it should be best to use only route 1 but when Q reaches a level at which the increased cost of using the next best route 2 is offset by the amount that Q can be reduced, then route 2 should also be used. The same rationale for larger values of Q leads to the threshold strategy outlined above. This is the type of routing policy which is considered in the following discussion.

$$J = \lambda T = E\{Q\} + \sum_{i=1}^L S_i \mu_i P_i \quad (3)$$

Two Alternate Routes



$$\lambda P(k-1) = (\mu_1 + \mu_2) P(k) \text{ for } K+1 \leq k$$

$$\bar{g} > \frac{\ln \left[\left((S_2 - S_1)(\lambda - \mu_1) + \frac{\lambda}{\lambda - \mu_1} + \frac{\mu_1 + \mu_2}{\mu_1 + \mu_2 - \lambda} \right) \frac{\lambda - \mu_1}{\lambda \sigma} \right]}{\ln y_1} \quad (12)$$

For the NS model where swapping is not allowed we find from Fig. 3 that

$$\begin{aligned} \text{(a)} \quad & \lambda P(0) = \mu_1 p_a(1) + \mu_2 p(1) \\ \text{(b)} \quad & \lambda [p(k-1) + p_a(k-1)] = \mu_1 [p(k) + p_a(k)] + \mu_2 p(k) \\ & \text{for } 2 \leq k \leq K \end{aligned} \quad (13)$$

$$(c) \quad \lambda p(k) + \mu_2 \sum_{j=1}^k p(j) = \mu_1 p(k+1) \text{ for } 1 \leq k \leq K$$

(d) $\lambda p(k) = (\mu_1 + \mu_2) p(k+1)$ for $K+1 \leq k$

$$(e) \quad \lambda [p(K) + p_n(K)] = (\mu_1 + \mu_2) p(K+1)$$

where

$$p(k) = \text{Prob } \{Q = k \text{ and server 2 is busy}\}$$

$$p_-(k) = \text{Prob} \{Q = k \text{ and server 2 is not busy}\}$$

Eq. (13d) can be solved to give

$$p(k) = \gamma_1^{(k-K-1)} p(K+1) \text{ for } k \geq K+1 \quad (14)$$

where $p(K+1)$ is obtained from (13e) once $p(K)$ and $p_s(K)$ are known. Equations (13a-c) give $2K$ equations for the remaining $(2K+1)$ unknown $P(0)$, $p(1)$, ..., $p(K)$, $p_s(1)$, ..., $p_s(K)$.

The additional condition needed to solve is as usual

$$\sum_{k=0}^{\infty} P(k) = 1 \quad (15)$$

An analytical solution of these equations leads to complicated expressions. However a computational solution is easily obtained, since (13c) can be solved iteratively to give $p(2) \dots p(K-1)$ in terms of $p(1)$. That is if we define the coefficients b_k by

$$p(k) \leq b_k p(1) \quad \text{for } 1 \leq k \leq K+1 \quad (16)$$

then using this in (13c) gives

$$b_{k+1} = \gamma_1 b_k + \gamma_2 \sum_{j=1}^k b_j \text{ for } 1 \leq k \leq K \quad (17)$$

where $b_1 = 1$ and $\gamma_3^A = \mu_2/\mu_1$. Or for easier calculation define

$$d_k \leftarrow \sum_{j=1}^k b_j \quad \text{and iterate}$$

$$d_k = d_{k-1} + b_k \quad (18)$$

$$b_{k+1} = \gamma_1 b_k + \gamma_2 d_k \quad \text{for } 1 \leq k \leq K$$

starting with $d_0 = 0$, $b_1 = 1$. Also since $P(k) = p(k) + p(k)$ (Note $P(k) = p(k)$ for $k \geq K+1$) eq. (13b) with (16) gives

$$P(k) = \gamma_1 P(k-1) - \gamma_1 b_k p(1) \text{ for } 2 \leq k \leq K \quad (19)$$

where from (13a)

$$P(1) = \gamma_1 P(0) + (1 - \gamma_2) p(1) \quad (20)$$

The solution of (19) can be written as

$$P(k) = \gamma_1^{(k-1)} P(1) - \gamma_3 \left[\sum_{j=1}^{k-1} \gamma_1^{(k-1-j)} b_{j+1} \right] P(1) \\ \text{for } k = 2, \dots, K. \quad (21)$$

so by defining

$$a_k \prod_{j=1}^{k-1} \gamma_1^{(k-1-j)} b_{j+1} \quad \text{for } 1 \leq k \leq K \quad (22)$$

we can iterate

$$a_{k+1} = \gamma_1 a_k + b_{k+1} \text{ for } 1 \leq k \leq K-1 \quad (23)$$

starting with $a_1 = 0$. Thus we see that (20) and (21) become

$$P(k) = \gamma_1^k P(0) + \gamma_1^{(k-1)} (1-\gamma_3) P(1) - \gamma_3 \alpha_k P(1) \\ \text{for } 1 \leq k \leq K. \quad (24)$$

Thus all the $P(k)$ can be expressed in terms of $p(1)$ and $P(0)$. But $p(1)$ itself can be expressed in terms of $P(0)$ because (13e) and (16) gives

$$y_2, P(K) = p(K+1) = b_{K+1} p(1) \quad (25)$$

so using $P(K)$ from (24) in (25) gives

$$P(1) = \frac{Y_1^K}{(b_{K+1}/Y_2) + Y_3 a_K - Y_1^{K-1}(1-Y_3)} P(0) \triangleq \beta P(0) \quad (26)$$

Thus all the $P(k)$ are expressible in terms of $P(0)$. When these are inserted into (15) we find that for $\gamma \neq 1$,

$$P(0) = \frac{1}{\frac{1-\gamma_1^K}{1-\gamma_1} + \beta \left[\frac{1-\gamma_1^{(K-1)}}{1-\gamma_1} (1-\gamma_3) - \gamma_3 \sum_{k=1}^{K-1} a_k + \frac{b_{K+1}}{\gamma_2(1-\gamma_2)} \right]} \quad (27)$$

where β is defined by (26), so all other probabilities are determined. Here

$$P_1 = 1 - P(0) - p(1) = 1 - (1+\beta) P(0) \quad (28)$$

$$P_2 = 1 - P(0) - \sum_{k=1}^K p_s(k) = 1 - \sum_{k=0}^K P(k) + \sum_{k=1}^K p(k)$$

and as before

$$E(Q) = \sum_{k=0}^{\infty} k P(k) \quad (29)$$

$$R = S_1 \mu_1 P_1 + S_2 \mu_2 P_2$$

can then be calculated.

Example 1

An example with $\mu_1 = 30$, $\mu_2 = 10$, $\lambda = 25$ and $S_1 = 1$, $S_2 = 2$ was analyzed. The results are shown in Fig. 4 for both the S and NS models. In this example $E(Q)$ increases monotonically with threshold K for both models. For most values of K the NS model gives slightly lower values because the second server operates more of the time than for the S model. The apparent inconsistency for $K=1$ is readily explained by the fact that in the NS model it is possible for the faster server (in this case server 1) to be idle while the slower server is busy. For low values of threshold this happens more often with the result that $E(Q)$ is raised. Example 2 will present a more striking illustration of this effect in which $E(Q)$ will not be monotonic but will actually display a minimum. As $K \rightarrow \infty$ both models approach the M/M/1 case since the second server is not used, so $\lim E(Q) = \lambda/(\mu_1 - \lambda) = 5$. The routing costs R decrease monotonically with threshold K as the more costly route 2 is used less frequently. As expected NS has larger values for R than S does because NS cannot switch a customer being served by server 2 to server 1 when Q falls below the threshold value. As $K \rightarrow \infty$, $P_1 \rightarrow 1$, $P_2 \rightarrow 0$ so $R \rightarrow 25$ for both models. The total cost J^2 is shown in Fig. 4b. NS gives $J_{\min} = 29.197$ at $K^* = 9$ and S gives $J_{\min} = 29.144$ at $K^* = 7$ but the minima are fairly broad. In this example using the simpler S model to find the minimum would give performance imperceptibly different than using the more realistic NS model.

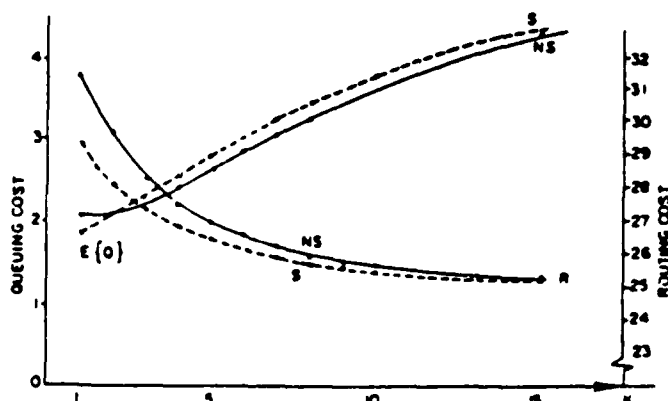


Fig. 4a. Queuing and Routing Costs for Example 1

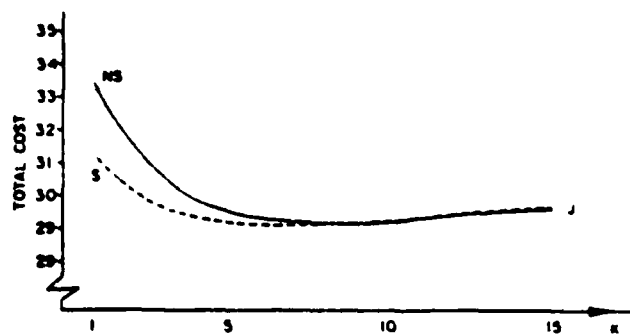


Fig. 4b. Total Cost for Example 1

As λ decreases we expect that K^* will increase since server 1 can more easily serve the demand by itself and at some point K^* will be infinite (i.e. it is best to use only route 1). As λ increases server 1 requires more help in holding $E(Q)$ down so K^* decreases. In fact for $\mu_1 \leq \lambda < \mu_1 + \mu_2$ server 2 must be used some of the time to keep $E(Q)$ finite.

Fig. 5 shows the total delay $T = J/\lambda$ for the NS server of Example 1 with $\lambda = 15, 25$ and 35 . For $\lambda = 15$, T decreases monotonically with K giving $K^* = \infty$ and $T_{\min} = 16/15 = 1.067$. For $\lambda = 35$ we see that $T \rightarrow \infty$ as $K \rightarrow \infty$ since $\lambda > \mu_1$. A minimum occurs at $K^* = 4$ giving $T_{\min} = 50.2/35 = 1.434$. The $\lambda = 25$ curve is the NS curve of Fig. 4b divided by 25.

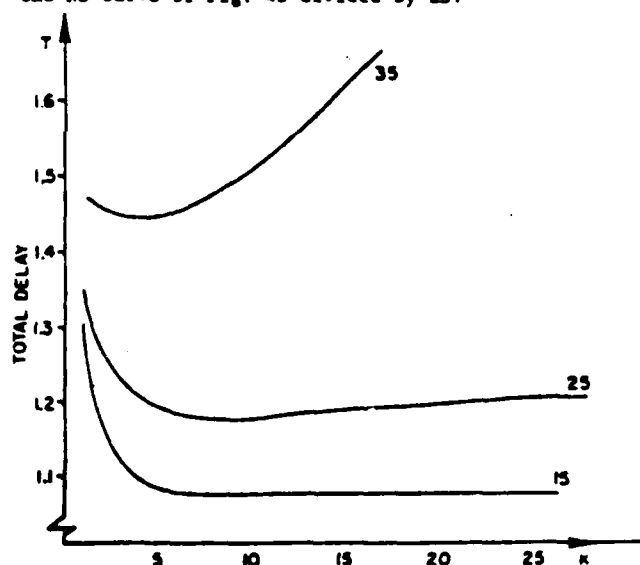


Fig. 5. Total Delay of Example 1 for different λ

In Example 1, the route with smaller routing delay S_1 also had the faster server ($\mu_1 > \mu_2$) so the delay in service using server 1 and the delay to reach the destination using both server 1 and the delay to reach the destination both favor route 1. It is evident that situations can arise where these two effects oppose each other so that if μ_1/μ_2 is large enough then even if $S_2 < S_1$ route 1 can still be the preferred route since the large delay in using server 2 outweighs the shorter delay in reaching the destination along route 2.

Example 2

Here $\mu_1 = 35$, $\mu_2 = 5$, $S_1 = 1.1$, $S_2 = 1$ and $\lambda = 20$. Fig. 6a shows $E(Q)$ and R for the NS and S

models. It can be seen that for the NS model $E(Q)$ has a minimum as explained above. This occurs at $K = 4$ and gives $\min E(Q) = 1.243$. The routing costs R increase monotonically with K here because $S_1 > S_2$. Fig. 6b shows that the NS model has a minimum at $K^* = 2$ of $J \min = 23.114$ while the S model has a minimum of $J \min = 22.933$ at $K^* = 1$.

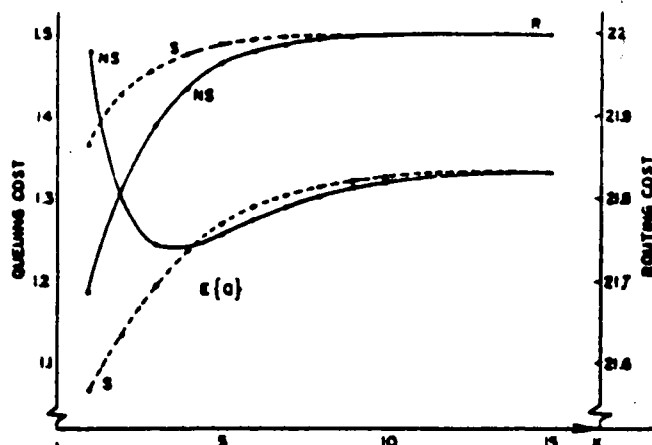


Fig. 6a Queueing and Routing Costs for Example 2

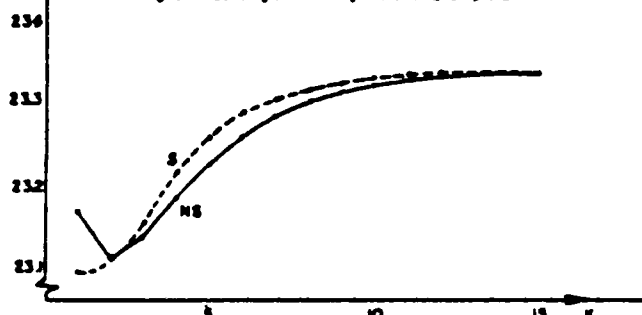


Fig. 6b Total Costs for Example 2

Example 2 illustrates that the relative sizes of S_1 and S_2 alone cannot be used to determine the preference ordering of the alternate routes. An exact determination can be made by finding $E(Q) + R$ from (29) with $K = 1$ when first one route and then the other is assumed to be preferred. The route yielding the smaller cost will then be the preferred route 1. An approximate and more easily applied test can be used. Suppose only one customer is in the system. If route 1 is chosen then the average delay to reach the destination is $1/\mu_1 + S_1$, since the first term is the average delay in service and the second is the additional average delay encountered along the route. Thus the most preferred route labeled #1 is the one with smallest $S_i + 1/\mu_i$ and all alternate routes can then be ordered by this quantity.

For the values of Example 2 we find $S_1 + 1/\mu_1 = 1.13$ and $S_2 + 1/\mu_2 = 1.2$.

Decentralized Routing

The local routing strategy just discussed is well suited for use in a decentralized routing scheme for larger networks. Since each node n is minimizing the average time T that it takes an entering customer to reach the destination, if this information is exchanged periodically between neighboring nodes it gives each node the information needed to update its

S_i values. Thus for example if t_i is the propagation delay from a node along route 1 to neighboring node j then $S_j = t_i + T_j$ will be used in the local routing strategy. With each update a node would recalculate the threshold values on which its real time routing decision is based using the latest value of S_i , μ_i and λ_i . These information updates would be made at a rate much slower than required for the local routing decisions.

Conclusion

This paper suggests an alternate route strategy which simply compares the number of customers Q at a node against an optimal threshold value K^* . When the threshold is exceeded then some traffic is diverted along an alternate route. The real time strategy is dynamic because it adapts its routing to the size of Q . It is also adaptive in the sense that K^* can be recomputed readily when operating parameters such as μ_i , λ_i or S_i change. Thus it seems ideally suited for use in a decentralized routing procedure for large networks where a low rate exchange of information between neighboring nodes can be used to update the operating thresholds.

REFERENCES

1. C.E. Agnew "On Quadratic Adaptive Routing Algorithms", *Comm. of ACM*, vol. 19, No. 1 pp. 18-22; Jan. 1976.
2. G. Foschini, J. Salz "A Basic Dynamic Routing Problem and Diffusion", *IEEE Trans. Comm.*, vol. COM-26, No. 3, pp. 320-327; March 1978.
3. A. Ephremedes, P. Varaiya, J. Walrand, "A Simple Dynamic Routing Problem," *IEEE Trans. Aut. Control*, vol. AC-25, No. 4, pp. 690-693; Aug., 1980.
4. T.S. Yum "The Design and Analysis of a Semi-dynamic Deterministic Routing Rule," *IEEE Trans. Comm.*, vol. COM-29, No. 4; pp. 498-504; April, 1981.
5. T.S. Yum, M. Schwartz "The Join-Biased-Queue Rule and its Application to Routing in Computer Communication Networks," *IEEE Trans. Comm.*, vol. COM-29, No. 4; pp. 505-511; April 1981.
6. R. Boorstyn, A. Livne "A Technique for Adaptive Routing in Networks," *IEEE Trans. Comm.*, vol. COM-29, No. 4, pp. 474-480, April 1981.
7. P.E. Sarachik and U. Ozgürer, "On Decentralized Routing in Traffic Networks," *IEEE Trans. Aut. Control*, (to appear 1982).
8. P.E. Sarachik, "Decentralized Dynamic Clearing of Congested Multi-Destination Networks," *Proc. of 19th Allerton Conference*, pp. 797-803; Oct. 1981.

B.7 Optimal Control of M/M/2 Queues

B. Maglaris

OPTIMAL CONTROL OF M/M/2 QUEUES

Basil S. Maglaris

Polytechnic Institute of New York

December, 1982

Abstract

The optimization of two-server Markovian queues with unequal rates of service is analyzed via the theory of Markov decision processes using an average expected cost criterion. It is shown that the optimal policy, that minimizes the steady-state average time delay experienced by customers in the system, is a deterministic threshold. Under this policy, the slower server is used whenever the number of waiting customers exceeds the threshold value.

1. INTRODUCTION

Stochastic optimization of queueing systems has been extensively studied due to its direct applicability in diverse engineering and economic disciplines. Early works on single server systems dealt with optimal control of the service rate in Markovian queues as in [CRABILL - 72], [ZACKS - 70] and [BLACKBURN - 72]. Optimality criteria involved service costs/rewards, service rate switching penalties, customer delays and start-up/shut-down costs. In [NAOR - 69] the queue size was controlled by imposing tolls to arriving customers and in [KNUDSEN - 72], this was generalized to multiple server systems. In several queueing control problems the optimal strategies depend on thresholds on the queue size. For excellent reviews on the methodology involved, see [PRABHU - 73] and [STIDHAM - 82], where specific hints are presented on how to apply the general semi-Markov decision theory, [HOWARD - 71], [ROSS - 69] to queueing processes.

A wealth of applications requiring optimal control of queues emerged within the context of store-and-forward data communication networks. Switching nodes in the network can be modeled as queueing systems with various streams of incoming data and several output options. It was found that adaptive routing at the node level significantly improves the delay performance of data messages as reported in [BOORSTYN - 81], [YUM - 81]. In [FOSCHINI - 78] a diffusion approximation was used to analyze node models where newly arriving messages had the option to join queues dedicated to alternate routes. In [EPHREMEDES - 80], techniques from the optimal control of queues were employed to derive optimal strategies for the above node model. In [SARACHIK - 82], the adaptive routing problem was formulated as

an M/M/2 queue with different service rates, and based on previous results, [SARACHIK - 81] pertaining to deterministic input models, a threshold policy was anticipated to minimize delay. Note that an M/M/2 queue exhibits lower delay than two dedicated queues since arriving messages in the latter case decide on a route by estimating a priori which server will become available first. In figure 1 we illustrate an M/M/2 queue with two unequal service rates, $\mu_a > \mu_b$. If the two rates are not substantially asymmetric, immediate use of any available server on a FCFS basis appears to be an efficient way to empty the system. At the other extreme, however, with server a many times faster than b, it is reasonable to defer use of b unless the number of messages in queue is large enough to feed the fast server for the duration of service in b. Obviously, the "customer" in b receives unfair service but the average system performance improves. This is an illustration of the discrepancy between "individual" and "social" optimization as reported in [KNUDSEN - 72]. In what follows, we formulate the M/M/2 optimization as a Markov decision process and prove that the optimal policy under certain conditions is of the threshold type.

2. MARKOV DECISION PROCESS FORMULATION

The optimization problem involves the minimization of the average number of customers (messages) in the system, including those in service. This is, by Little's formula, equivalent to minimizing the average message delay.

The state must indicate the number of customers in the system and denote whether each of the servers a and b is busy servicing a customer. Let the state be

- 0 : Empty system
 $ia, i \geq 1$: i customers in the system and only server a busy
 $ib, i \geq 1$: i customers in the system and only server b busy
 $i, i > 2$: i customers in the system and both a and b busy.

Actions may be taken whenever an arrival finds one or both servers empty and a choice must be made whether or not to serve or which server to use, and whenever a departure creates the opportunity for a waiting customer to proceed to a server. In order to "discretize" the continuous time process defined on the state space above, we uniformize with respect to R , the maximum total rate out of all states. Obviously $R = \lambda + \mu_a + \mu_b$ corresponding to a state i , with both servers busy. For states ia , ib and 0 we now consider transitions at rate R some of which represent arrivals or departures and others represent "self loops" or transitions not triggered by external events. As an example, from state ia , $i > 2$, we may have the following transitions:

Upon an arrival, with rate λ	$ia \rightarrow (i + 1)a$	OR
	$ia \rightarrow (i + 1)$	
Upon a departure, with rate μ_a	$ia \rightarrow (i - 1)a$	OR
	$ia \rightarrow (i - 1)$	OR
	$ia \rightarrow (i - 1)b$	
Remaining transitions with rate μ_b	$ia \rightarrow ia$	OR
	$ia \rightarrow i$	

This uniformization technique is suggested in [STIDHAM, 82] as a means to simplify the algebra involved with the optimality equations without changing the steady state statistics of the process.

The possible transitions out of a state depend on the actions taken at transition points, whenever there is a choice. The action space is finite; either serve the customer at the head of the queue with a specific available server or not. No swapping is allowed i.e. a message already in server a may not be switched to server b . Furthermore, we assume that if there are customers in the system, at least one server must be active. This assumption may be proved but we omit the formalism to provide a clear picture of the state transitions. In Figure 2 we depict the state diagram with all candidate transitions. Solid lines and circles identify a set of actions leading to a threshold policy at $i = 3$. Under that policy all dotted states are transient and do not influence the steady state averages. Rates such as μ'_a , μ''_a denote transitions opposing to the threshold policy but valid candidates for an optimal policy.

The optimal policy π , would minimize the average expected cost g defined as the average number of customers j under π :

$$g = \min_{\pi} \{E_{\pi}(j)\}$$

If $C(X_n)$ denotes the cost of state X_n at the n^{th} period between the n^{th} and the $(n + 1)^{\text{st}}$ transition (with average dwell time $1/R$), then

$$E_{\pi}(j) = \lim_{N \rightarrow \infty} \frac{1}{N+1} E_{\pi} \left\{ \left[\sum_{n=0}^N C(X_n) \right] / X_0 \right\}$$

The one period cost $C(X_n)$ is exactly the number in system at X_n

$$C(i) = C(ia) = C(ib) = i$$

In order to use the results of the Markov Decision theory directly, we first limit the queue capacity (maximum number of customers)

to a large but finite number M . Arrivals into a full system are blocked and cleared. The minimization criterion involves customers already in the system and ignores the effect of blocking. This truncation is a standard technique, [PRABHU - 73] to alleviate problems induced by unbounded one period costs and to handle ergodicity requirements for average expected cost minimization. Certainly if M is large enough and the process stable, truncation is a good approximation. Furthermore, it will be shown that the form of the optimal policy does not depend on M and thus we may postulate that if a stable optimal policy exists it is of the same form as in the truncated problem.

Under the modeling assumptions stated above, all conditions for the existence of a deterministic stationary optimal policy are satisfied, [ROSS - 69, p. 149] and thus the policy π minimizing the average number of customers in the system is a deterministic assignment of actions to states.

3. FORM OF THE OPTIMAL POLICY

In order to determine the form of the optimal policy, we examine the equivalent discounted cost problem with a discount factor $\alpha < 1$. As shown in [ROSS - 69] this problem can be transformed into an average expected cost minimization by letting $\alpha \rightarrow 1$.

Let $V(i)$, $V(ia)$, $V(ib)$ denote the α -optimal costs starting from states i , ia , and ib respectively. With X_n representing the state during the n^{th} period. We have

$$V(X_0) = \min_{\pi} E_n \left\{ \left[\sum_{n=0}^{\infty} \alpha^n C(X_n) \right] / X_0 \right\}$$

$$X_0, X_n \in \{i, ia, ib\}, C(i) = C(ia) = C(ib) = i$$

The recursive optimality equations for the V's are given below:

$$(\lambda + \mu_a + \mu_b) V(0) = \alpha [\lambda \cdot \min\{V(1a), V(1b)\} + (\mu_a + \mu_b)V(0)] \quad (1)$$

$$(\lambda + \mu_a + \mu_b) V(1a) = 1 + \alpha [\lambda \cdot \min\{V(2a), V(2)\} + \mu_a V(0) + \mu_b V(1a)] \quad (2)$$

$$(\lambda + \mu_a + \mu_b) V(1b) = 1 + \alpha [\lambda \cdot \min\{V(2b), V(2)\} + \mu_b V(0) + \mu_a V(1b)] \quad (3)$$

$$(\lambda + \mu_a + \mu_b) V(2a) = 2 + \alpha [\lambda \cdot \min\{V(3a), V(3)\} + \mu_a \min\{V(1a), V(1b)\} + \mu_b \min\{V(2a), V(2)\}] \quad (4)$$

$$(\lambda + \mu_a + \mu_b) V(2b) = 2 + \alpha [\lambda \cdot \min\{V(3b), V(3)\} + \mu_b \min\{V(1a), V(1b)\} + \mu_a \min\{V(2b), V(2)\}] \quad (5)$$

$$(\lambda + \mu_a + \mu_b) V(2) = 2 + \alpha [\lambda \cdot V(3) + \mu_a V(1b) + \mu_b V(1a)] \quad (6)$$

$$(\lambda + \mu_a + \mu_b) V(i, a) = i + \alpha [\lambda \cdot \min\{V(i+1, a), V(i+1)\} + \mu_a \min\{V(i-1, a), V(i-1, b), V(i-1)\} + \mu_b \min\{V(i, a), V(i)\}], \quad i > 2 \quad (7)$$

$$(\lambda + \mu_a + \mu_b) V(i, b) = i + \alpha [\lambda \cdot \min\{V(i+1, b), V(i+1)\} + \mu_b \min\{V(i-1, b), V(i-1, a), V(i-1)\} + \mu_a \min\{V(i, b), V(i)\}], \quad i > 2 \quad (8)$$

$$(\lambda + \mu_a + \mu_b) V(i) = i + \alpha [\lambda \cdot V(i+1) + \mu_b \min\{V(i-1), V(i-1, a)\} + \mu_a \min\{V(i-1), V(i-1, b)\}], \quad i > 2 \quad (9)$$

If M is the maximum (blocking) state :

$$(\lambda + \mu_a + \mu_b) V(M, a) = M + \alpha [\lambda \cdot V(M, a) + \mu_a \min\{V(M-1, a), V(M-1, b), V(M-1)\} + \mu_b \min\{V(M, a), V(M)\}] \quad (10)$$

$$(\lambda + \mu_a + \mu_b) V(M, b) = M + \alpha [\lambda \cdot V(M, b) + \mu_b \min\{V(M-1, b), V(M-1, a), V(M-1)\} + \mu_a \min\{V(M, b), V(M)\}] \quad (11)$$

$$(\lambda + \mu_a + \mu_b) V(M) = M + \alpha [\lambda \cdot V(M) + \mu_b \min\{V(M-1), V(M-1, a)\} + \mu_a \min\{V(M-1), V(M-1, b)\}] \quad (12)$$

Let R denote the uniformization rate:

$$R = \lambda + \mu_a + \mu_b$$

LEMMA 1

The functions $V(i)$, $V(ia)$, $V(ib)$ are increasing in i and $V(1a) > V(0)$, $V(1b) > V(0)$, $V(2) > V(1a)$, $V(2) > V(1b)$

Proof

We will use induction on the recursive equations (1)-(9). First assume that we start with a set of V 's satisfying the Lemma. A logical choice is to set the cost at step 0 equal to the one period costs:

$$V_0(i) = V_0(ia) = V_0(ib) = i$$

Suppose now that the lemma holds after n subsequent applications of the recursive equations to V_0 :

$$V_n(i), V_n(ia), V_n(ib) \text{ are increasing on } i$$

To complete the proof we must show that the lemma will hold at the $(n+1)$ st step. We will demonstrate this step for $V_{n+1}(i)$, $i > 2$. From (9) we have:

$$RV_{n+1}(i+1) = i+1 + \alpha[\lambda V_n(i+2) + \mu_b \min\{V_n(i), V_n(i,a)\} + \mu_a \min\{V_n(i), V_n(i,b)\}]$$

$$RV_{n+1}(i) = i + \alpha[\lambda V_n(i+1) + \mu_b \min\{V_n(i-1), V_n(i-1,a)\} + \mu_a \min\{V_n(i-1), V_n(i-1,b)\}]$$

Now, $i+1 > i$, $V_n(i+2) > V_n(i+1)$ (by assumption), $\min\{V_n(i), V_n(i,a)\} > \min\{V_n(i-1), V_n(i-1,a)\}$ and $\min\{V_n(i), V_n(i,b)\} > \min\{V_n(i-1), V_n(i-1,b)\}$ since the minimum of two increasing functions is increasing. Thus $RV_{n+1}(i+1) > RV_{n+1}(i)$. Using the same reasoning it can be easily seen that all costs $V_{n+1}(i)$, $V_{n+1}(ia)$, $V_{n+1}(ib)$ are increasing on i up to its maximum M and that $V_{n+1}(2) > V_{n+1}(1a) > V_{n+1}(0)$, $V_{n+1}(2) > V_{n+1}(1b) > V_{n+1}(0)$.

LEMMA 2

If $\mu_a > \mu_b$, $V(1a) < V(1b)$, and $V(i) < V(ib)$, $i \geq 2$

Proof

We use the same inductive argument as in the proof of Lemma 1. Assuming that the inequalities hold at step 0 and step n, we have at the (n+1)st step

$$RV_{n+1}(1a) = 1 + \alpha[\lambda \cdot \min\{V_n(2a), V_n(2)\} + \mu_a V_n(0) + \mu_b V_n(1a)]$$

$$RV_{n+1}(1b) = 1 + \alpha[\lambda V_n(2) + \mu_b V_n(0) + \mu_a V_n(1b)]$$

In order to show that $V_{n+1}(1a) < V_{n+1}(1b)$, since $V_n(2) \geq \min\{V_n(2a), V_n(2)\}$, it suffices to show that

$$\mu_a V_n(0) + \mu_b V_n(1a) < \mu_b V_n(0) + \mu_a V_n(1b) \quad \text{or}$$

$$\mu_a V_n(1b) - \mu_b V_n(1a) > (\mu_a - \mu_b) V_n(0)$$

Since $V_n(1b) > V_n(1a)$ (by inductive assumption), $\mu_a > \mu_b$ and $V_n(1a) > V_n(0)$ (Lemma 1), it follows that

$$\mu_a V_n(1b) - \mu_b V_n(1a) > (\mu_a - \mu_b) V_n(1a) > (\mu_a - \mu_b) V_n(0)$$

This proves the first argument, $V(1a) < V(1b)$, assuming $V_{n+1}(i) < V_{n+1}(ib)$. From (5) and (6), applying the inequalities at step n, we have

$$RV_{n+1}(2) = 2 + \alpha[\lambda V_n(3) + \mu_a V_n(1b) + \mu_b V_n(1a)]$$

$$RV_{n+1}(2b) = 2 + \alpha[\lambda V_n(3) + \mu_b V_n(1a) + \mu_a V_n(2)]$$

From Lemma 1, we have $V_n(2) > V_n(1b)$, thus $V_{n+1}(2) < V_{n+1}(2b)$

From (8) and (9), we have that for $i \geq 2$

$$RV_{n+1}(i) = i + \alpha[\lambda V_n(i+1) + \mu_b \min\{V_n(i-1), V_n(i-1, a)\} + \mu_a V_n(i-1)]$$

$$RV_{n+1}(ib) = i + \alpha[\lambda V_n(i+1) + \mu_b \min\{V_n(i-1, a), V_n(i-1)\} + \mu_a V_n(i)]$$

and since $V_n(i) > V_n(i-1)$ it follows that $V_{n+1}(i) < V_{n+1}(ib)$.

Note that we make use of arguments from Lemma 1 in the inductive proof. This is valid since the two Lemmas are consistent and may start from the same initial set of V 's. Alternatively, one can combine the two Lemmas into a single inductive proof.

COROLLARY 1

States ib , $i > 1$ are transient under the optimal policy.

Proof

Observing the optimality equations, we notice that under Lemma 2, there is no path from states (i) and (ia) into any state (ib) , $i > 1$. State $(1b)$ may be reached upon departure from 2. This statement conforms with the intuitive argument that there is no reason to use the slow server when the fast server is idle.

LEMMA 3

If $V(i-1,a) > V(i-1)$, $i-1 > 1$, then $V(j,a) > V(j)$ for all $j \geq i$

Proof

Assume that in the optimal policy, there exists a state $i-1$, such as $V(i-1,a) > V(i-1)$. In this case, we will use the inductive procedure again but now keeping the costs of states $(i-1,a)$ and $(i-1)$ fixed to their optimal values. The iterative application of the optimality equations to the remaining costs, will converge by contraction to their optimal values. The a-priori knowledge of $V(i-1,a)$ and $V(i-1)$ will reduce the number of unknowns, and if our guess is correct, it just creates two redundant consistent equations.

At step 0 we set $V_0(i-1,a) = V(i-1,a)$ and $V_0(i-1) = V(i-1)$. The remaining V_0 's are set to arbitrary initial values satisfying the Lemma

B.6 A Dynamic Alternate Route Strategy for Traffic Networks

Sarachik

IEEE Conference on Decision and Control, December
1982, Orlando

and the monotonicity property of Lemma 1, in reference to $V(i-1,a)$ and $V(i-1)$. It can be easily seen that the monotonicity property will hold throughout the subsequent operations, by translating the proof of Lemma 1, to incorporate the fixed values $V(i-1,a)$ and $V(i-1)$.

Assume that at step n the Lemma holds. Remember that no change is allowed for the cost of states $(i-2,a)$ and $(i-1)$. From (7) and (9), we have

$$RV_{n+1}(i,a) = i + \alpha[\lambda V_n(i+1) + \mu_a V_n(i-1) + \mu_b V_n(i)], \quad V_n(i-1) = V(i-1)$$

$$RV_{n+1}(i) = i + \alpha[\lambda V_n(i+1) + \mu_b V_n(i-1) + \mu_a V_n(i-1)]$$

and since $V_n(i) > V_n(i-1)$, it follows that $V_{n+1}(i,a) > V_{n+1}(i)$. The same equations apply for all $j \geq i$. At $j=M$, $V_n(j+1)$ is replaced by $V_n(M)$ and similar inequalities hold. This completes the proof of the Lemma.

THEOREM 1a

The α -optimal policy for the finite M/M/2 queue, defers service from the slow server until the number of waiting customers exceeds a threshold value.

Proof

From Lemma 3, it follows that for the least $K \geq 1$ for which $V(K+1,a) > V(K+1)$, all states (ia) , $i > K$, will exhibit greater costs $V(ia)$ than $V(i)$. Thus new arrivals and departures will favor states (ia) , for $i \leq K$ and states (i) for $i > K$. This threshold K will render all states (ia) , $i > K$, transient along with states (ib) , $i > 1$ (Corollary 1). The threshold value, depends on the parameters λ, μ_a, μ_b . It may be that for $\mu_a \gg \mu_b$, the threshold K approaches the limit of the queue size in which case, we never use the slow server b . Equivalently if μ_a/μ_b is close to unity, the threshold moves to 1 which is exactly an M/M/2 queue with arrivals on empty state routed to server a .

THEOREM 1b

The optimal average expected cost policy for the finite M/M/2 queue is a deterministic threshold as in Theorem 1a.

Proof

For the finite state problem, state (0) is recurrent for all deterministic policies considered in the α -optimal recursive equations (1)-(12). Furthermore the one period cost i is bounded by its maximum M and hence all conditions for the existence of optimal average expected cost policies are satisfied as in [ROSS - 70], Corollary 6.20. The structure of the optimal policy π will be the same as in the discounted case and the optimal average expected cost, $g = E_{\pi}(i)$ is given by

$$g = \lim_{\alpha \rightarrow 1} (1-\alpha)V_{\alpha}(0)$$

The truncation of the state space was introduced to guarantee that the one period costs are bounded, and that state 0 is recurrent under all policies. However, since the threshold structure does not depend on the maximum queue size M , it can be concluded that if a stable optimal policy exists to the infinite state problem, it is of the threshold type.

4. DISCUSSION AND NUMERICAL EXAMPLES

In the previous section we proved that the optimal policy minimizing the average queue length (or average delay) is of the threshold type. Obviously the criterion involves "social optimality" from the system point of view. Individual customers (messages) would never chose the slow server regardless of the state of the system unless their service time on the slow server is less than the sum of their

service time in the fast server and the waiting time before being served by the fast server. The optimum threshold value depends on the parameters of the system, namely the ratio of the service rates and the total system utilization. In order to illustrate this dependence we solved the equilibrium equations in an infinite size system for various thresholds, service rate ratios and traffic intensities. In Figure 3, we depict the average queue length at traffic intensity $\lambda/(\mu_a + \mu_b) = 0.5$. As expected if server a is much faster than b (39 to 1), the optimal threshold becomes very large, eventually leading to a single server queue with the slow server never used. Conversely for relatively balanced rates (2 to 1 or equal), both servers are always used as in an uncontrolled M/M/2 system (preference is given to a if both a and b are idle). For rate ratios 3 to 1 and 7 to 1 there exists clearly an optimal threshold at 2 and 4 respectively. A consequence of using thresholds is that we obtain unbalanced traffic volumes routed via a and b. In Figure 4, we illustrate the percentage of the traffic routed via a, $100 \times \lambda_a/\lambda$. Notice, that even with balanced rates 1 to 1 and $K=1$ (pure M/M/2 system), more traffic is routed to a than b, due to decisions taken upon arrivals to an empty system. As the threshold increases, certainly the λ_a/λ ratio approaches 100%. This observation led to the suggestion of threshold policies in adaptive routing schemes, where the traffic served via each outgoing link (server) is constrained to a value determined by some global criterion. In [MAGLARIS - 83], we formulated this problem and proved that threshold policies minimize average queueing delays with constrained output flows.

Finally, in Figure 5, we demonstrate the effect of the traffic intensity on the optimum threshold value. As intuitively expected, queues close to saturation do not have the luxury of threshold policies and must operate both servers at full capacity. Lightly utilized queues do, however, exhibit improved performance under threshold values which increase as the utilization drops. Depending on the $\mu_a:\mu_b$ ratio, a system utilized at low levels would involve thresholds approaching a positive integer close to $\mu_a/\mu_b - 1$. As an example, for $\mu_a:\mu_b = 7$, and very low traffic intensity, the optimal policy uses the slow server only if 7 messages have been accumulated, i.e. the threshold value is 6.

5. REFERENCES

1. Blackburn, J.D., "Optimal Control of a Single Server Queue with Balking and Reneging," *Management Science*, Vol. 19, No. 3, November 1972, pp. 297-313.
2. Boorstyn, R.R. and A. Livne, "A Technique for Adaptive Routing in Networks," *IEEE Transactions on Communications*, Vol. COM - 29, No. 4, April 1981, pp. 474-480.
3. Crabill, T.B., "Optimal Control of a Service Facility with Variable Exponential Service Times and Constant Arrival Rates," *Management Science*, Vol. 18, No. 9, May 1972, pp. 560-566.
4. Ephremedes, A., Varaya, P., and J. Walrand, "A Simple Dynamic Routing Problem," *IEEE Transactions on Automatic Control*, Vol. AC - 25, No. 4, August 1980, pp. 690-693.
5. Foschini, G. and J. Salz, "A Basic Dynamic Routing Problem and Diffusion," *IEEE Transactions on Communications*, Vol. COM - 26 No. 3, March, 1978, pp. 320-327.
6. "Dynamic Probabilistic Systems; Vol. II, Semi-Markov and Decision Processes," R.A. Howard, John Wiley & Sons, New York, 1971.
7. Knudsen, N.C., "Individual and Social Optimization in a Multi-server Queue with a General Cost-Benefit Structure," *Econometrica*, Vol. 40, 1972, pp. 515-528.
8. Maglaris, B., "Optimal Control of M/M/2 Queues and Adaptive Routing in Computer Networks," to appear in MELECOM - 83 Proceedings, Athens, Greece, May 1983.
9. Naor, P., "The Regulation of Queue Size by Levying Tolls," *Econometrica*, Vol. 37, 1969, pp. 15-23.
10. *Applied Probability Models with Optimization Applications*, S.M. Ross, Holden - Day, San Francisco, 1970.
11. Prabhu, N.U., and S. Stidham, Jr., "Optimal Control of Queueing Systems," TR 186, Department of Operations Research, Cornell University, Ithaca, N.Y., June 1973.
12. Sarachik, P.E., "Decentralized Dynamic Clearing of Congested Multi-Destination Networks," *Proceedings of the 19th Allerton Conference*, October 1981, pp. 797-803.
13. Sarachik, P.E., "A Dynamic Alternate Route Strategy for Traffic Networks," *Proceedings of the 21st IEEE Conference on Decision and Control*, Orlando, Florida, December 1982.
14. Stidham, S. Jr., "Optimal Control of Arrivals to Queues and Network of Queues," *Proceedings of the 21st IEEE Conference on Decision and Control* Orlando, Florida, December 1982.

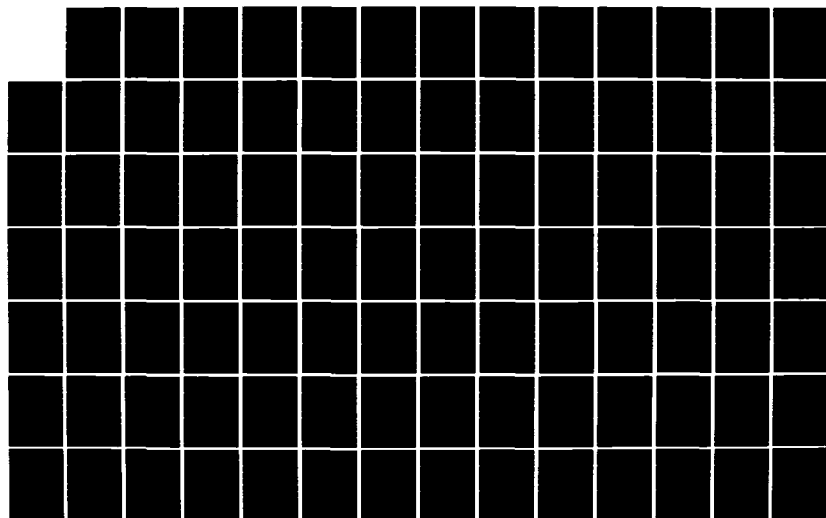
15. Yum, T.S., and M. Schwartz, "The Join-Biased-Queue Rule and its Application to Routing in Computer Networks," IEEE Transactions on Communications, Vol. COM - 29, No. 4, April 1981, pp. 505-511.
16. Zacks, S., and M. Yadin, "Analytic Characterization of the Optimal Control of a Queueing System," Journal of Applied Probability, No. 7, 1970, pp. 617-633.

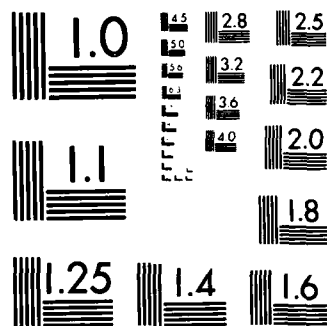
AD-A131 915

RESEARCH IN NETWORK MANAGEMENT TECHNIQUES FOR TACTICAL
DATA COMMUNICATION. (U) POLYTECHNIC INST OF NEW YORK
BROOKLYN R R BOORSTYN ET AL. 01 SEP 82 CECOM-80-0579-F
UNCLASSIFIED DAAK80-80-K-0579

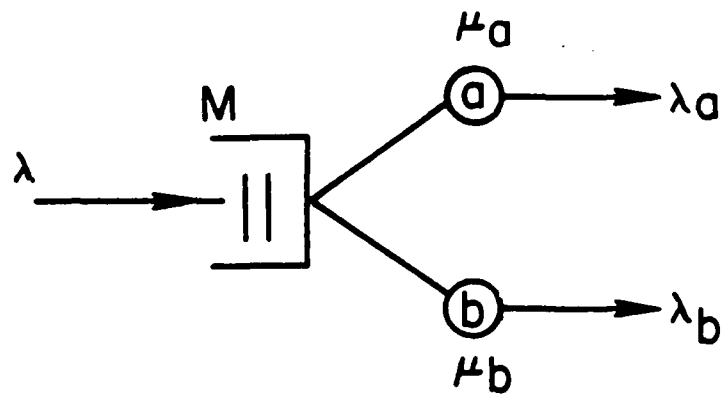
3/5

F/G 17/2.1 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963 A

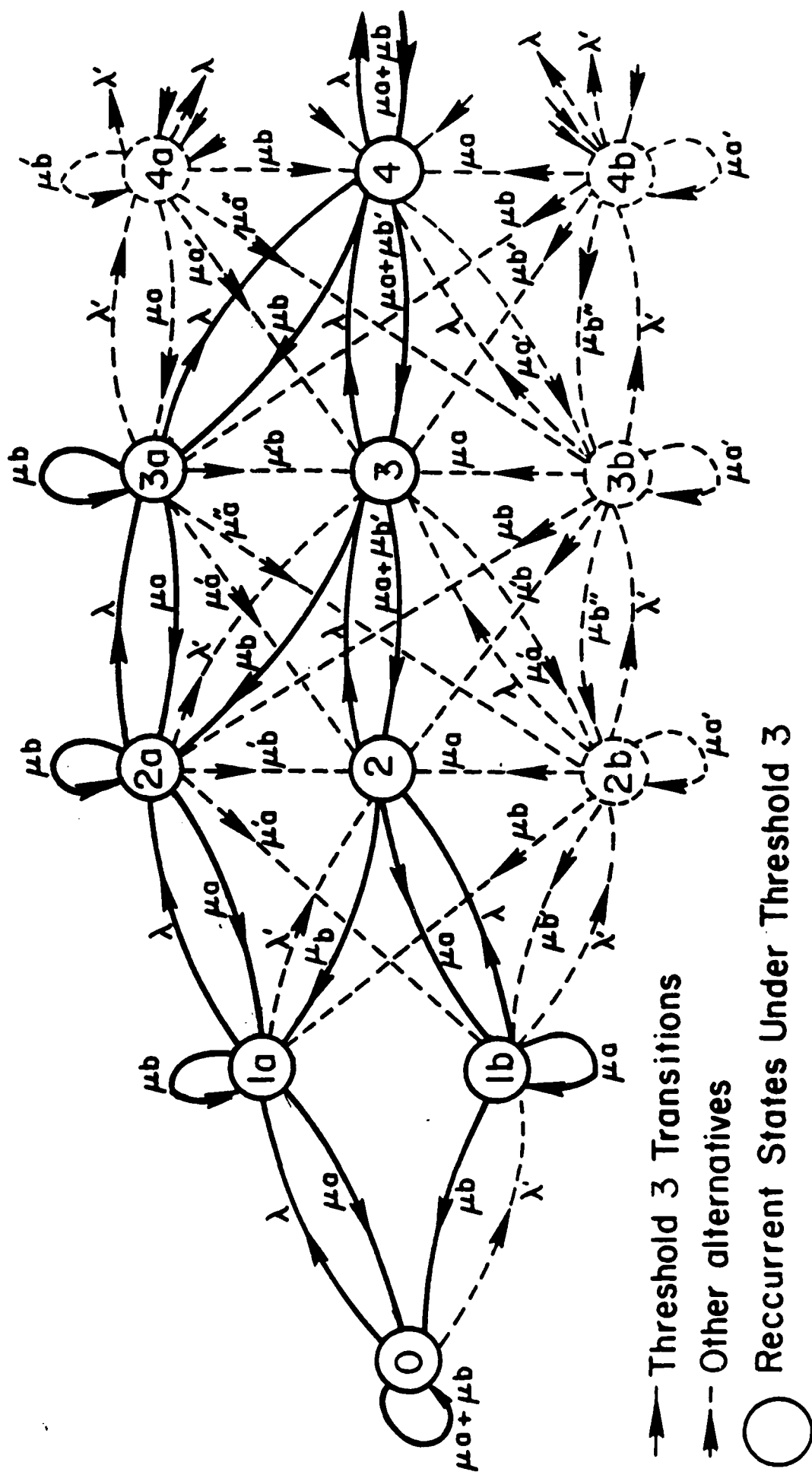


$$\mu_a > \mu_b, \quad \frac{\lambda}{\mu_a + \mu_b} < 1$$

Finite Queue Size M.

Fig.1 The M/M/2 Queueing Model.

Fig2 State Transition Diagram



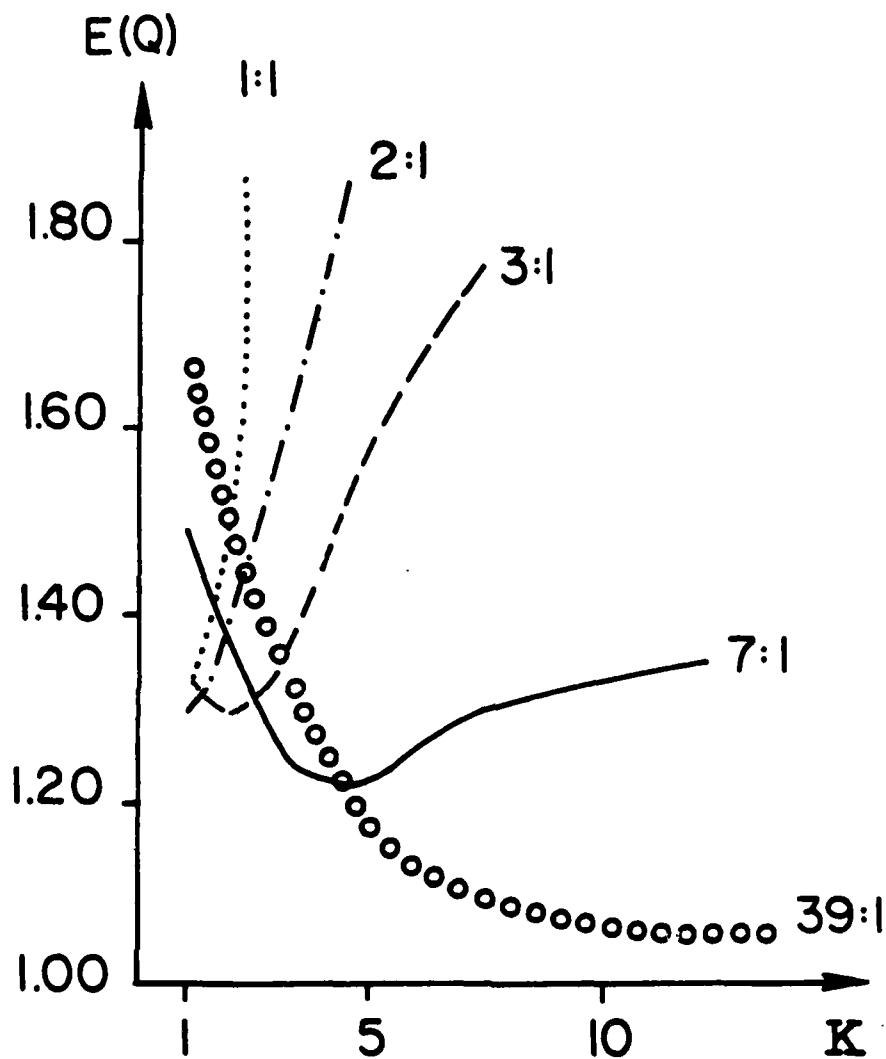


Fig.3 Average Queue Length as Function of the Threshold K for Various $\mu_a : \mu_b$ Ratios.
Traffic Intensity 50%

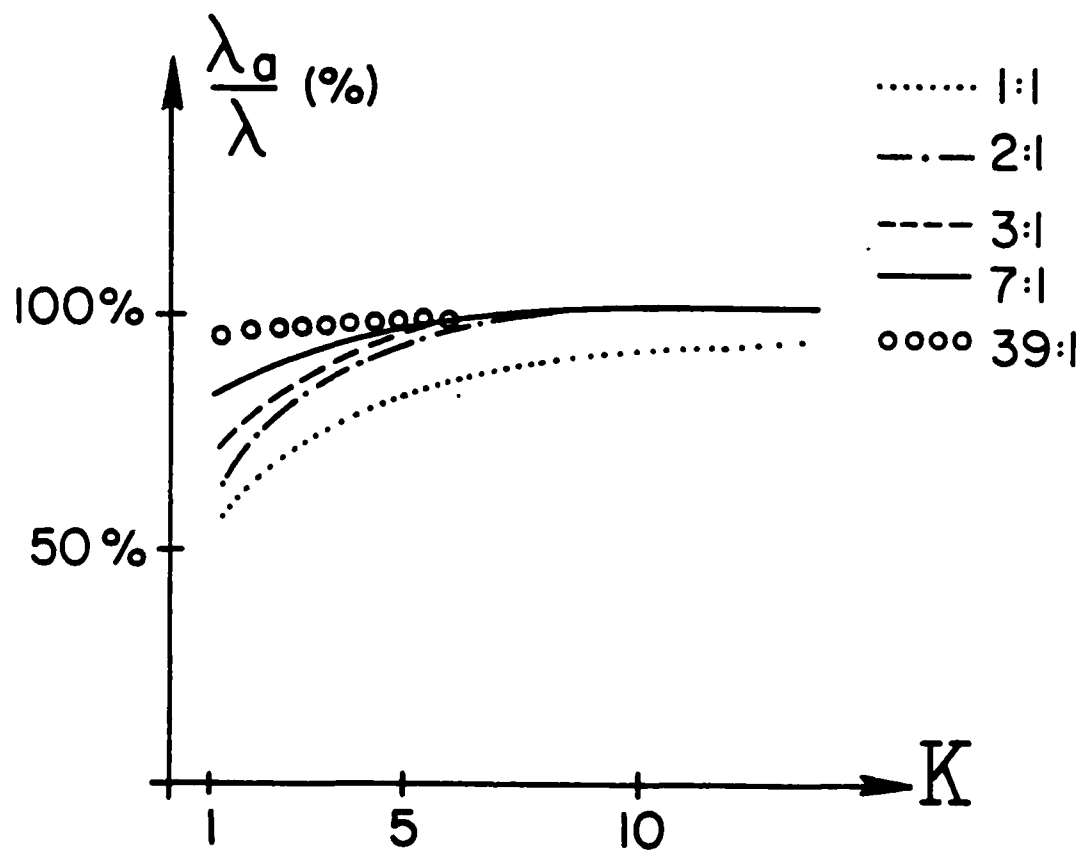


Fig.4 Percentage of Traffic Served at A as
Function of the Threshold K for
Various $\mu_a : \mu_b$ Ratios.
Traffic Intensity 50%

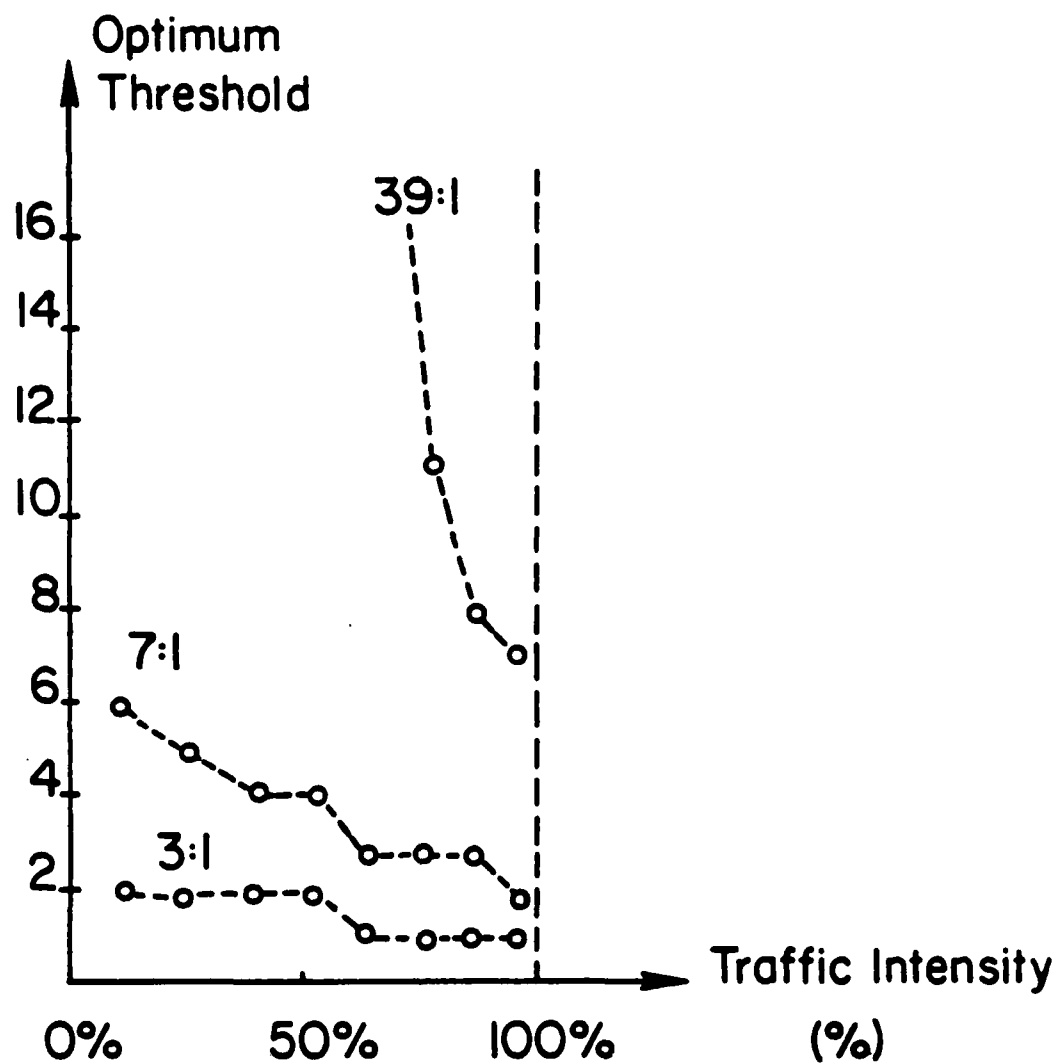


Fig5 Optimum Threshold as Function of the Traffic Intensity, for various $\mu_a : \mu_b$ Ratios.

C. Algorithms

C.1 A Shortest Path Algorithm for the Solution of the
Simple Knapsack Problem and Extensions

Kershenbaum

Second Semiannual Technical Report, September 1981

A SHORTEST PATH ALGORITHM FOR THE SOLUTION
OF THE SIMPLE KNAPSACK PROBLEM AND EXTENSIONS

Aaron Kershenbaum*

Department of Electrical Engineering and Computer Science
Polytechnic Institute of New York
Brooklyn, NY 11201

ABSTRACT

We consider several versions of the Knapsack Problem and show that they can be solved using a shortest path algorithm requiring both storage and running time which are polynomial functions only of the number of types of object and the size (weight) of a single object. This is a significant improvement over previous algorithms for the solution of the Knapsack Problem using shortest paths, which typically require storage and runtime which are functions of the total knapsack size.

I. INTRODUCTION

The Knapsack Problem, which is interesting in its own right, has also received much attention recently^[3] as a means of solving integer programming problems via a relaxation technique using group theory. Formally, the Knapsack Problem can be stated as:

$$\text{Maximize } z = \sum_{i=1}^M p_i x_i$$

*This work was partially supported by NSF under Grant ENG 7908120 and by U.S. Army CECOM under Contract DAAK-80-80-K-0579.

$$\begin{aligned} \text{subject to } &= \sum_{i=1}^M w_i x_i = W \\ &x_i \geq 0 \text{ for } i = 1, 2, \dots, M \\ &x_i \text{ integer} \end{aligned}$$

Thus, we are given M types of object with weight w_i and profitability p_i per object of type i . We are required to maximize the total profit subject to a constraint that the sum of the weights of the objects selected equals W . In the simple (unbounded) version of the problem there is no restriction on the number of objects of each type which may be included in the solution.

Sometimes an inequality constraint is used in place of the equality constraint, i.e., one seeks objects whose aggregate weight does not exceed W . Such a problem can be transformed into a problem with an inequality constraint by including an additional object with unit weight and zero profitability. Here we will concentrate primarily on problems with equality constraints.

We assume that W and the w_i are non-negative integers (or, more generally, that they are commensurate) and that the p_i are non-negative.

Previous approaches to the solution of this problem include those reported by Horowitz and Sahni^[1] using dynamic programming and implicit enumeration. These approaches have been found to be effective for a wide range of problems but have exhibited excessive run-times for problems with a large number of nearly equally valuable (in terms of cost per unit weight) objects. Also, their worst case running times are exponential in the number of object types. Denardo

and Fox^[2] have developed an approach which overcomes these objections by solving the problem as a shortest path problem in a network with W nodes and $W \times M$ arcs. Thus, their approach requires storage proportional to W and runtime proportional to $W \times M$. If W and M are sufficiently small, their approach is extremely attractive. If W is large, however, the storage, and to a lesser extent the runtime, become prohibitive.

Denardo and Fox present their algorithm in the context of using a Knapsack Problem as an integer programming relaxation. In such cases, W is generally of the same order as the w_i . We extend their approach to the case where W is very large but the individual w_i are not, and exploit the underlying cyclic group structure of the problem to develop an algorithm with both runtime and storage requirements a function only of the weight of a single object and the number of object types. Garfinkle and Nemhauser^[4] point out the relationship to the group structure but do not discuss application of the structure to develop an algorithm as efficient as the one presented here.

II. BASIC PROCEDURE

For the sake of clarity, we begin by describing the basic procedure and justifying it. In later sections, extensions and refinements of this procedure are considered.

We begin by defining a network $K = (N, A, L)$ corresponding to the Knapsack Problem (KP). The node set, N , contains $W + 1$ nodes numbered 0 through W corresponding to the aggregate weight of the objects selected thus far. At each node, i , there are outgoing arcs

corresponding to feasible objects which may be selected to augment a partial solution with weight i . Thus, at node i there will be arcs $(i, i + w_k)$ for all k such that $i + w_k \leq W$. The length of an arc corresponding to an object of type k is p_k . We thus have:

$$N = \{0, 1, \dots, W\}$$

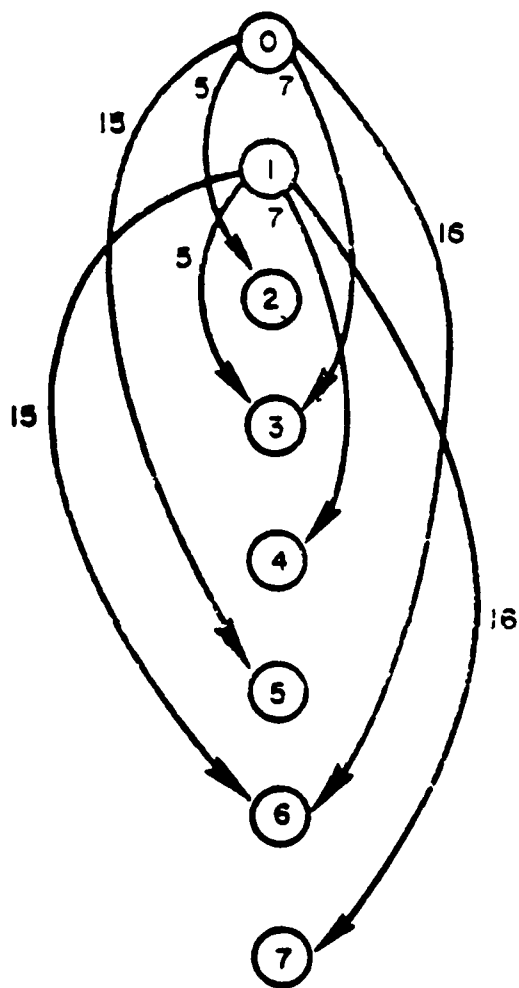
$$A = \{(i, i + w_k) \mid i = 0, 1, \dots, W, \text{ for all } k \text{ such that } i + w_k \leq W\}$$

$$L = \{l_{ij} \mid (i, j) \in A, \text{ where } l_{ij} = p_k \text{ for arcs corresponding to type } k \text{ objects}\}$$

The network thus contains approximately MW arcs. We will assume that $w_{k_1} \neq w_{k_2}$ for $k_1 \neq k_2$ as if two types of objects have the same weight the one with the smaller profit can be ignored. We also assume $w_k > 0$ and $p_k \geq 0$ for all k . We will refer to arcs corresponding to a type k object simply as type k arcs. Figure 1 illustrates part of a network corresponding to a KP with 4 types of object, described in the table accompanying Figure 1. For the sake of clarity, only nodes 0 through 7 and arcs from nodes 0 and 1 are shown in Figure 1.

Paths from node 0 to node W in such networks correspond to solutions of the associated KP. The length of a path, defined as the sum of the lengths of the arcs in the path, is the total profit of the solution. Thus, the longest path corresponds to the optimal solution. Note that, by the assumption that $w_k \geq 0$ for all k , the network is acyclic and so the problem of finding the longest path from 0 to W is solvable using Bellman's shortest path algorithm suitably modified to find longest paths. Defining d_i to be the length of the current estimate of the longest path from node 0 to node i , this amounts simply to:

$$\text{that } w_{k_1} \neq w_{k_2} \text{ for } k_1 \neq k_2 \text{ as}$$



i	P_i	W_i
1	2	5
2	3	7
4	5	15
4	6	16

Figure 1- Standard Network Representation Knapsack Problem

Algorithm 1

Step 0: $d_0 \leftarrow 0$

$d_i \leftarrow \infty \quad i = 1, \dots, W$

Step 1: For $i = 1$ to W

For all j such that $(i, j) \in A$

$\{d_j = \max(d_j, d_i + l_{ij})\}$

One can also keep track of the solution itself by setting $a_j \leftarrow i$ whenever the value of d_j is updated. At the end of the algorithm one can trace the path back from node W to node 0 using the a_j . We refer to d_j as the label on node j and to a_j as the predecessor of node j . Step 1 is known as a scan of each node, i .

Algorithm 1 is the simplest possible shortest path algorithm. The nodes are scanned in numerical order and each node is scanned only once. The algorithm has a running time on the order of MW operations and a storage requirement on the order of W . If W is very large, the approach becomes unattractive. The approach described below avoids this problem by solving the problem using a network which is, in general, much smaller. The key to reducing the size of the network is embodied in the following lemmas. Define $\hat{w} = \max_k (w_k)$ and $w^* = w_j$ such that $\frac{p^*}{w^*} = \max_k \frac{p_k}{w_k}$. Thus $\hat{w}$ is the weight of the "heaviest" type of object and w^* is the weight of the "best" type of object.

The proofs of some of the lemmas rely upon elementary properties of cyclic groups. The interested reader is referred to [6] for more details.

Lemma 1: An optimal KP solution exists containing no set of objects whose aggregate weight is a multiple of w^* .

Proof: Given an optimal KP solution containing a set of objects of aggregate weight nw^* , we can replace these objects by n "best" objects without altering the total weight. The net change in profit, Δp is defined by:

$$\Delta p = np^* - \sum_j p_j$$

$$\sum_j \left(\frac{w_j}{w^*} p^* - p_j \right)$$

By the definition of p^* , each term of the sum is non-negative and hence $\Delta p \geq 0$. We can thus replace the set without destroying optimality and the lemma is proven.

The terms in the above sum can be interpreted as the amount of profit "wasted" by using a type j object instead of a best object. We can in fact redefine the arc lengths in the network corresponding to a given KP to be precisely these quantities. The optimal KP solution then corresponds to the shortest path from 0 to W . This can be seen to be true by observing that the above transformation of the arc lengths affects the lengths of all paths from 0 to W in precisely the same way. If the length of a 0 to W path in the original network was p the length of the same path in the new network is $p^* \frac{W}{w^*} - p$. Thus the relative length of all 0 to W paths are reversed. Alternatively, one can observe that the optimal path is the one which wastes the least profit. In the sequel, we use this form of the length function as it will allow us to improve the efficiency of the algorithm. We refer to the network using this length function as K_1 .

In Figure 1, the new lengths of arcs of types 1 through 4 are 1, 2, 0, and 2, respectively. Note that type 3 arcs are best and that best arcs have 0 length. All arcs have non-negative length. If the best type of arc is not unique (e.g. if $w_2 = 9$ in the above example), it is possible for the length of more than one type of arc to be 0. To avoid confusion, we assume that the best type of object is unique. The extension of the algorithms to this case is straight forward.

Lemma 2: An optimal solution exists containing no more than $w^* - 1$ non-best objects.

Proof: Any set of w^* or more objects must contain a subset of weight nw^* for some $n > 0$. To see this, consider a set of w^* objects, $i_1, i_2, \dots, i_{w^*}$. From the sums

$$S_j = \left(\sum_{k=1}^j W_{i_k} \right) \text{ modulo } w^*$$

$$\text{If } S_j = 0 \text{ then } \sum_{k=1}^j W_{i_k} = nw^*.$$

$$\text{If } S_j = S_\ell \text{ for } \ell > j \text{ then } \sum_{k=j}^{\ell} W_{i_k} = nw^*.$$

But there are w^* S_j 's and only $w^* - 1$ nonzero values of modulo w^* . Hence, either $S_j = 0$ or $S_j = S_\ell$ for some ℓ and the set contains a subset of weight nw^* . From the proof of Lemma 1 we see that we can replace any such subset by n best elements without decreasing the profit. Thus, we need only consider sets containing $w^* - 1$ or fewer non-best elements together with zero or more best elements. An optimal solution can always be obtained in this way.

But since all elements have weight no greater than $\hat{w}$, the aggregate weight of the non-best elements in any such optimal solution

will not exceed $\hat{w}^*(w^* - 1)$. If $W \geq \hat{w}w^*$, we need not consider W explicitly as a constraint. We need simply apply the above algorithm to get d_j for $j \leq \hat{w}w^*$, and then complete the solution by adding the appropriate number of best objects.

Indeed, we are only interested in the weight modulo w^* of a partial solution since any two solutions whose weights differ by a multiple of w^* differ only by one or more best objects. We thus have:

Lemma 3: If $W \geq \hat{w}w^*$, the optimal KP solution can be found by adding zero or more best objects to the set of objects defining the shortest path in the network formed by merging all nodes with the same weight modulo w^* as described above.

Proof: Consider the original network before the nodes were merged. Consider the 0 to W path corresponding to an optimal solution. By Lemma 1, we may assume this path contains no set of arcs (correspond to objects) of weight nw^* except for best arcs. Since the network contains paths corresponding to picking these objects in all possible orders, we may assume that the path under consideration has all the best arcs at the end. Consider only the initial subpath corresponding to the non-best arcs.

Now, consider what the transformation does to this initial subpath. All the nodes in this path have different weights modulo w^* . Thus the transformation changes this optimal 0 to W path in K_1 to a 0 to W' path in the network with merged nodes, where $W' = W$ modulo w^* . Note that other paths, including some optimal ones, become complex paths containing cycles because they contain subsets whose aggregate weights are a multiple of w^* . The path we are focusing on

however remains a simple path and can thus be found using an ordinary shortest path algorithm.

No new paths are created by the transformation. Thus, an optimal KP solution can be obtained by appending 0 or more best arcs to the shortest 0 to W' path in the new network, and the lemma is proven.

It is thus possible to define a new network

$K_2 = (N, A, L)$:

$$N = \{0, 1, \dots, w^* - 1\}$$

$$A = \{(i, j) \mid j = (i + w_k) \text{ modulo } w^* \text{ for some } k\}$$

$$L = \{\ell_{ij} \mid (i, j) \in A, \ell_{ij} = \frac{w_k}{w^*} p^* - p_k \text{ for}$$

$$k \text{ defining } (i, j)\}$$

This network has only w^* nodes instead of W . Furthermore, we note that if $(w_{k_1} = w_{k_2}) \text{ modulo } w^*$, then type k_1 and type k_2 arcs are parallel and the arc corresponding to the object with the smaller ℓ_{ij} can be ignored. Thus, there is at most 1 outgoing arc from each node to each other. The total number of arcs is thus at most $w^*(w^* - 1)$. We have thus removed dependency on both M and W and have a procedure whose running time and storage are a polynomials only in w^* .

ALGORITHM DESCRIPTION

We now describe the actual procedure in more detail and analyze its storage and runtime.

The following is the essence of the procedure:

Step 0: (Initialization)

$$d_0 = 0$$

$$d_j = \infty \quad j = 1, \dots, w^*-1$$

where d_j is the current estimate of the length of the shortest path from 0 to j .

Step 1: Find i , the best node to scan next. This is discussed in detail below.

Step 2: (Scan node i)

For $k = 2, \dots, M$

Let $j = (i + w_k) \text{ modulo } w^*$

If $d_j > d_i + \ell_{ij}$ then $d_j = d_i + \ell_{ij}$ and $PR_j = k$

where ℓ_{ij} is the length of an arc corresponding to a type k object ($\ell_{ij} = w_k(p^*/w^* - p_k)$ as defined above) and PR_j records the type of the arc used to label node j and hence keep track of the optimal solution.

Step 3: Return to Step 1 if any nodes remain to be scanned; otherwise stop.

It is clear that the storage required for this procedure is linear in $w^* + M$. If Step 1 is sufficiently simple and each node is only scanned once, the procedure's runtime will be dominated by Step 2 and will be proportional to Mw^* . This is in fact the case, as we will see.

A node, j , must be scanned if its label, d_j , is improved (reduced) as it may then improve the labels of other nodes. If we are to ensure that each node is scanned at most once, we must defer

scanning it until its label cannot be improved. When all arc lengths are positive, as they are here, this objective is achieved by scanning the nodes in ascending order of d_j . This is Dijkstra's well known shortest path algorithm.

In the general case, Dijkstra's algorithm requires that in Step 1 we find the as yet unscanned node with the smallest label. If this were done naively, it would require an examination of w^* labels and the overall procedure would require $(w^*)^2 + Mw^*$ operations. In this case, however, we can implement Step 1 carefully and do much better.

We form "buckets" of nodes to be scanned and place nodes to be scanned together in the same bucket if they have similar labels. If the width of the buckets is q , then we place a node with label d , where $(k-1)q \leq d < kq$ in the k^{th} bucket. We then simply work our way through the buckets in ascending order of bucket number scanning the nodes in each bucket in arbitrary order. Thus, there is no explicit search in Step 1. If the width of the bucket is no greater than the length of the shortest arc in the graph, then any node labeled by the node currently being scanned will reside in a higher numbered bucket and hence, each node will be scanned at most once.

The only problem with this procedure, which is well known, is that the number of buckets itself may become very large, in particular much larger than the number of nodes, if the longest arc is many times greater than the shortest. In this case, a great deal of storage and running time may be wasted in dealing with empty buckets. We now show that in this case fewer than w^* buckets are required in all cases.

We begin by reindexing the arcs out of each node corresponding to the $M - 1$ remaining types of object in increasing order of length. (Recall that the arc corresponding to the best type of object was removed from explicit consideration.) Thus, L_1 , is the length of the arc which corresponds to the second best type of object, etc. The w_i and p_i are reindexed correspondingly. We will refer to arcs of length L_i , weight w_i , and profit p_i after this reindexing as arcs of type i (or as arcs corresponding to objects of type i). For simplicity, we will assume $L_i \neq L_j$ for $i \neq j$ although the procedure does not require this.

Consider the case where w^* is relatively prime to w_1 . By the definition above, arcs of length L_1 are the shortest arcs in the graph. Since w_1 and w^* are relatively prime, a path containing at most $w^* - 1$ arcs comprised entirely of arcs of type 1 exists from 0 to each other node. Thus, we know that no node need have a label greater than $L_1(w^* - 1)$. So, $w^* - 1$ buckets of width L_1 suffice since the shortest path will be no longer than the aforementioned one.

In the case where w^* and w_1 are not relatively prime, the above path will loop back to node 0 before reaching many of the other nodes. In this case, the following preliminary computation is carried out before setting up the buckets:

$$r = w^*$$

For $j = 1, M-1$

$$g_j = r / \text{GCD}(r, w_j)$$

$$r = \text{GCD}(r, w_j)$$

where $\text{GCD}(i, j)$ is the greatest common divisor of the integers i and j and can be found using Euclid's Algorithm, whose running time is bounded by the following Lemma:

Lemma 4: Euclid's Algorithm has a worst case running time of order $\log N$ when applied to find the GCD of N and M , for $N > M$.

Proof: For $N > M$, Euclid's Algorithm replaces a problem on N and M by one on M and $N - qM$. The worst case for this, i.e., the one which converges most slowly, is when $q = 1$ at all stages. This results in a Fibonacci sequence. It is well known [1] that the K^{th} Fibonacci number, F_K is given by:

$$F_K = \frac{1}{\sqrt{5}} \left\{ \left(\frac{1 + \sqrt{5}}{2} \right)^K - \left(\frac{1 - \sqrt{5}}{2} \right)^K \right\}$$

$$\frac{1}{\sqrt{5}} \left(\frac{1 + \sqrt{5}}{2} \right)^K$$

Thus, the runtime of the algorithm is of order K for N of order F_K , i.e., K is of order $\log N$.

We observe that, corresponding to the paths comprised entirely of type 1 arcs above, there is now a tree rooted at 0, spanning all the nodes, and containing paths with at most $g_j - 1$ arcs of length L_j . An illustration of such a tree is given in Figure 4. In this case only nodes 0, 8, and 4 can be reached using only arcs of type 1 before the path cycles back to node 0. The number of nodes so reachable is g_1 (three in this case). The original set of w^* nodes is partitioned into g_1 parts which are then each partitioned into g_2 smaller parts corresponding to the nodes which are now reachable via arcs of type 2. The partitioning continues until all nodes are reached. The above can be proven rigorously in the general case using the properties of cyclic groups.

We now define buckets of variable width corresponding to the longest path in the tree defined above. Thus, there are $g_1 - 1$

buckets of width l_1 followed by $g_2 - 1$ buckets of width l_2 , etc. A total of $B = \sum (g_i - 1)$ buckets is required. Since the product of the g_i 's is w^* , $B \leq w^*$.

We have shown that less than w^* buckets are required. We now show that no node is scanned more than once. To do so, we show that the width of each bucket is no greater than the length of the shortest arc still permitted in a path.

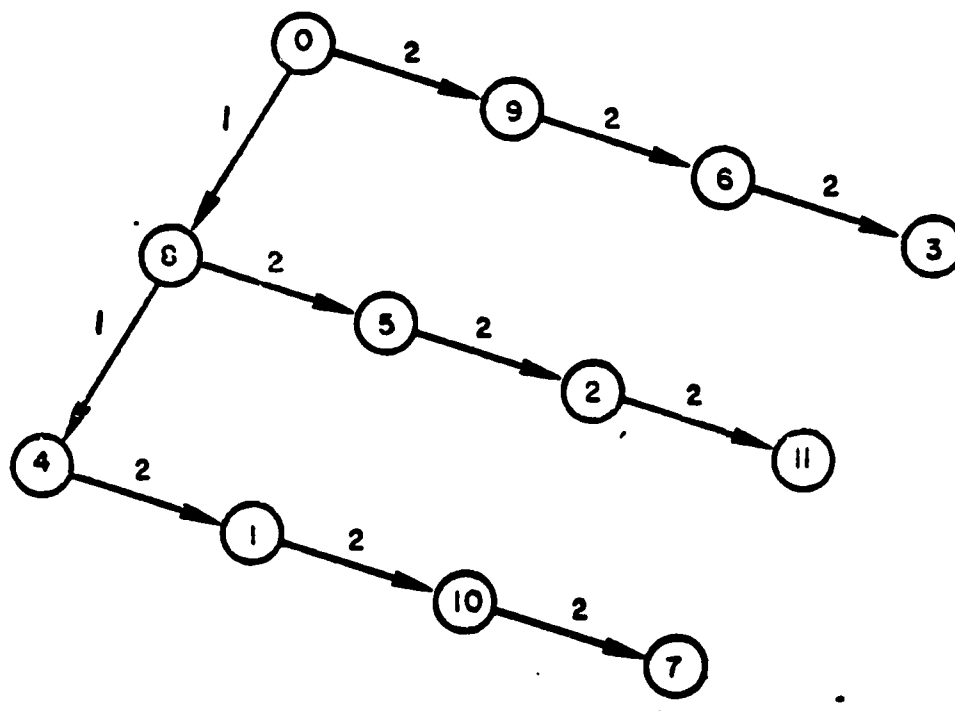


Figure 4: A spanning tree bounding the lengths of shortest paths

Note that the paths from 0 to j correspond to ordered selections of objects of aggregate weight j ; i.e., there are distinct paths corresponding to the selection of the same objects in different orders. We can, and should, consider only one path corresponding to each distinct collection of objects. This is easily done by keeping track of the last type of arc in the path (which we are doing with the variable

PR_j) and only considering arcs with the same or higher index (object type). Thus, we will find solutions corresponding to a selection of objects in non-decreasing order of type.

Now, consider a node in bucket i of width w_j . This node was either reached via a path in the tree considered above or via an arc whose length is greater than that of any in that tree. In latter case, the length of the smallest arc under consideration exceeds the width of any bucket. In the former case, since we are considering arcs in increasing order of length, again a node can only enter a bucket via an arc at least as long as the width of the bucket.

Extensions

If $W < w^*\hat{w}$ then it is possible for w_i to be equal to w_j and for objects i and j not to dominate one another. In particular, if $w_i > w_j$ and $\ell_i < \ell_j$ neither i nor j may dominate. This is because the constraint on W may prohibit the use of a sufficient number of objects with the smaller length. As an example of this, consider a problem with $W = 24$ and (w_i, ℓ_i) of $(5,50)$, $(6,58)$, $(11,109)$, $(1,0)$. Here, the ℓ_i are 0,2,1, and 10, and $w_2 = w_3$ modulo 5. Neither object 2 nor object 3 dominates the other, however. In fact, the optimal solution is 4 type-2 objects in this case. For $W = 21$ or $W = 22$, however, type-3 objects would be used in the optimal solution. For $W = 23$, both type-2 and type-3 objects would be used. Lemma 3 can thus not be extended directly to the case whose $W < w^*\hat{w}$.

Thus, it is not possible to make the final reduction to a network with w^* nodes in this case. However, when the individual w_i are small, which is the principal case of interest here, $w^*\hat{w}$ will be small as well. The above algorithm can then still be applied effectively *

the network with W nodes. Alternatively, one can view the following procedure as working on the reduced graph with w^* nodes but allowing multiple labels on each node. We adopt this latter point of view in describing the modified procedure below.

A label on a node, i , is now a couple (d_i, c_i) where d_i is the amount of wasted profit to get to node i , $i = 1, 2, \dots, w^*-1$, and c_i is the cycle number of this label. The c_i are defined by the relation

$$i = j - w^*c_i$$

where j is the node in the network K_1 (with W nodes) which would have received the same label. A node i in K_2 can thus have several labels corresponding to different c_i , but for any two labels (d_i, c_i) and (d'_i, c'_i) on the same node i ,

$$d'_i > d_i \rightarrow c'_i < c_i$$

since as we noted above the only reason for considering a larger d_i would be to obtain a smaller c_i . Similarly, for any pair of objects with weights having the same residue modulo w^* the object with the larger weight must have the smaller length.

There are several limitations on the maximum number of objects of any given type. In particular, m_i , the maximum number objects of type i , is bounded by

$$m_i \leq \min \left\{ \left\lceil \frac{W}{w_i} \right\rceil, \frac{w^*}{\text{GCD}(w^*, r_i)} - 1 \right\}$$

where r_i is the residue of w_i modulo w^* . The first term on the right hand side follows from the fact that any larger number of type i objects have weight greater than W . The second term follows by the some reasoning is Lemma 2.

Proceeding along the lines similar to those in [3] we create $(\log_2 M_i) + 1$ objects corresponding to $1, 2, 4, 8, \dots$ and 2^b objects of type i , where b is largest power of 2 that is less than M_i . We thus now have defined a new problem with at most $M \log_2 W$ objects. At most one object of each type is permitted in the solution.

Proceeding along lines similar to those in [3], we can create $(\log_2 M_i) + 1$ objects corresponding to $1, 2, 4, 8, \dots, 2^b$ objects of type i , where b is the largest power of 2 that is less than M_i . We then have a problem with at most $M^1 = M \log_2(w^*)$ objects where at most one object of each type need be considered. Thus, we could create M^1 buckets and proceed with the algorithm as above.

One can refine this procedure by eliminating dominated objects from consideration. In particular, if two objects i and j , have the same residue modulo w^* , and $w_i > w_j$ and $\ell_i \leq \ell_j$ then object j may be eliminated from further consideration. One must be careful, however, not to allow one copy of an object to eliminate another copy of the same object.

For example suppose there were initially 4 types of object with (W_i, P_i) equal to $(10, 20)$, $(3, 5)$, $(7, 4)$ and $(1, 0)$ $W = 10$ and $W < \hat{w}w^*$, respectively. If $W = 37$ we must consider the modified procedure. We thus create objects with (W_i, P_i) equal to $(6, 10)$, $(12, 20)$, and $(6, 10)$ corresponding to 2, 4, and 2 objects of type 2. These three new objects together with the original object of type 2 can be used to select anywhere from 0 to 9 copies of the type 2 object. The first new object is identical to the third and would dominate it unless we specifically prohibited this.

If the W_i and P_i are drawn independently from uniform distributions, this latter refinement will reduce the average number of objects to no more than $w^* \ln(M)$ objects and buckets as we see from Lemma 5.

Lemma 5: There will be on the average on the order of no more than:

$$w^* \ln \left[\frac{M \log_2(W^*)}{w^*} \right]$$

objects left undominated after dominated objects are removed in the above procedure.

Proof: As observed above, there are approximately $M \log_2(W^*)$ or fewer objects before dominance is checked. Suppose these objects divide evenly into residue classes. There would then be $M \log_2(w^*)/w^*$ objects in each residue class.

An object can be dominated by any other object in the same class. If the objects are ordered in increasing order of w_i , an object will be dominated unless its w_i is less than the w_i of all objects preceding it on the list since the weights are uniformly distributed, the K^{th} item in the list will be undominated with probability $\frac{1}{K}$. Thus, given a list of length L the expected number of undominated elements remaining on the list is

$$\sum_{j=1}^L \frac{1}{j} \ln(L)$$

The result follows directly by substituting for L . If the residue classes do not all contain the same number of elements, the result is still true since we observe that the quantity

$$\ln(L_1) + \ln(L_2) \text{ for } L_1 + L_2 = 2L$$

is maximized for $L_1 = L_2 = L$, that is, equal sized residue classes yield the maximum number of objects.

Thus, even for $W < w^*\hat{w}$, we have a number of buckets polynomial in w^* , not W . The running time is thus at worst of order $Ww^* < (w^*)^2\hat{w}$.

It is possible to extend the procedure still further to take into account restrictions on the number of permissible objects of each type, for some or all of the objects. First, if one is given an a priori restriction, U_i , on the number of objects of type i , one need only set

$$M_i = \text{Min} (M_i, U_i)$$

where M_i is the maximum number of objects of type i permitted and the M_i on the right hand side is computed as above. If the a priori maximum permissible number of best objects is less than W/w^* than a fictitious best object with $w^* = W + 1$ must be added to the problem. This maintains the validity of the procedure but may reduce its efficiency considerably if the original w^* was much smaller than W .

CONCLUSIONS

We have presented an algorithm for the solution of the simple (unbounded) Knapsack Problem whose running time and storage are functions only of the size of the "best" object weight, w^* in the case where the overall weight constraint is sufficiently large. In the cases where the overall weight constraint is smaller or where restrictions exist on the number of objects of a given type, we present an extension of this procedure which in practice is often very efficient and whose worst case running time is no greater than $(w^*)^2\hat{w}$ where $\hat{w}$ is the weightiest object.

C.2 A Network Shortest Path Approach to the Knapsack
Problem

Kershenbaum

IEEE International. Conference on Circuits and
Computers, October 1980, New York

A NETWORK SHORTEST PATH APPROACH TO THE KNAPSACK PROBLEM

Aaron Kershenbaum*

Department of Electrical Engineering and Computer Science
Polytechnic Institute of New York
Brooklyn, NY 11201

ABSTRACT

We consider the simple (unbounded) Knapsack Problem and show that it can be solved using a shortest path algorithm requiring both storage and running time which are polynomial functions only of the number of types of object and the size (weight) of the single largest object. This is a significant improvement over previous algorithms for the solution of the Knapsack Problem using shortest paths, which typically require storage and runtime which are functions of the total knapsack size.

INTRODUCTION

The Knapsack Problem, which is interesting in its own right, has also received much attention recently as a means of solving integer programming problems via a relaxation technique using group theory. Formally, the Knapsack Problem can be stated as:

$$\begin{aligned} \text{Maximize } z &= \sum_{i=1}^M p_i x_i \\ \text{subject to } \sum_{i=1}^M w_i x_i &= W \\ x_i &\geq 0 \text{ for } i=1,2,\dots,M \\ x_i &\text{ integer} \end{aligned}$$

Thus, we are given M types of object with weight w_i and profitability p_i per object of type i . We are required to maximize the total profit subject to an constraint that the sum of the weights of the objects selected equals W . In the simple (unbounded) version of the problem which we consider here, there is no restriction on the number of objects of each type which may be included in the solution.

Sometimes an inequality constraint is used in place of the equality constraint considered here; i.e., one seeks objects whose aggregate weight does not exceed W . Such a problem can be transformed into a problem with an inequality constraint by including an additional object with unit weight and zero profitability. Here we will concentrate primarily on problems with equality constraints.

We assume that W and the w_i are non-negative integers (or, more generally, that they

* This work was supported by NSF Grant ENG7908120.

are commensurate) and that the p_i are non-negative.

Previous approaches to the solution of this problem include those reported by Horowitz and Sahni [1] using dynamic programming and implicit enumeration. These approaches have been found to be effective for a wide range of problems but have exhibited excessive runtimes for problems with a large number of nearly equally valuable (in terms of cost per unit weight) objects. Also, their worst case running times are exponential in the number of object types. Denardo and Fox [2] have developed an approach which overcomes these objections by solving the problem as a shortest path problem in a network with W nodes and $W \times M$ arcs. Thus, their approach requires storage proportional to W and runtime proportional to $W \times M$. If W and M are sufficiently small, their approach is extremely attractive. If W is large, however, the storage, and to a lesser extent the runtime, become prohibitive.

Denardo and Fox present their algorithm in the context of using a Knapsack Problem as an integer programming relaxation. In such cases, W is generally of the same order as the w_i . We extend their approach to the case where W is very large but the individual w_i are not, and exploit the underlying cyclic group structure of the problem to develop an algorithm with both runtime and storage requirements a function only of the weight of the largest single object and the number of object types.

BASIC PROCEDURE

For the sake of clarity, we begin by describing the basic procedure informally. Extensions, refinements and a more formal description are given in later sections.

Given a particular Knapsack Problem (i.e., given values for W , M and the pairs (p_i, w_i) , $i=1,\dots,M$) we can construct a graph, G , with nodes $j=0,1,\dots,W$ and directed arcs $e=(i,m)$ where $m=i+w_k$ for some k . There is an arc in G corresponding to each $i < W$ and each $k < M$ provided $m < W$. Figure 1 shows such a graph for $M=3$, $W=7$, and the $(p_i, w_i) = (3,2), (5,3), (10,5)$. The length of each arc (shown next to the arc) is the profit obtained by selecting the type of object corresponding to the arc. Node numbers correspond to weight used up by the

objects already selected. a path from node 0 to node j corresponds to a selection of objects with aggregate weight j and the length of the path (the sum of the lengths of the arcs in the path) corresponds to the profit obtained. There exists a path corresponding to any feasible selection of objects and hence, the longest path from 0 to W in G corresponds to the optimal solution to the Knapsack Problem. This is the essence of the shortest path technique described by Denardo and Fox among others.

It is possible to accomplish the same optimization, however, using a graph containing a much smaller number of nodes. Johnson [3] observed that it is not in general necessary to label all nodes in order to obtain the optimum solution, but rather, that one may stop when node $j = w^* \hat{w}$ has been reached, where $\hat{w} = \max_j (w_j)$ and $p^*/w^* = \max_j (p_j/w_j)$; i.e., $\hat{w}$ is the weight of the "heaviest" object and w^* is the weight of the "best" object. We will explore this observation and exploit it in developing a new, more efficient algorithm.

The key to proving the validity of the above observation is that if an arbitrary solution contains one or more objects whose aggregate weight is a multiple, say q , of w^* , then a new solution no worse than the first can be obtained by replacing these objects by q best objects. Such a replacement leaves the total weight unaltered and cannot decrease the total profit. Thus, we need only consider partial solutions which do not contain any objects whose aggregate weight is a multiple of w^* . The appropriate number of best objects can be added to complete any such solution at the end of the procedure.

There can be at most w^*-1 objects in any such partial solution. To see this, consider a partial solution containing b objects with weights $w(1), w(2), \dots, w(b)$. Form the sums $s_j = (\sum_{i=1}^j w(i))$ modulo w^* . The s_j must all be different and must all be integers in the range 1 to w^*-1 . Otherwise, the b objects contain a subset of aggregate weight qw^* since:

$$s_j = 0 + \sum_{i=1}^j w(i) = qw^*$$

and

$$s_j = s_k + \sum_{i=j+1}^k w(i) = qw^* \text{ for } j < k$$

Note that it is in fact possible for such a partial solution to contain w^*-1 objects, for instance w^*-1 objects with weight w_1 when w_1 and w^* are relatively prime (i.e., contain no common factors greater than 1), so this bound

on the maximum number of objects is tight at least in some cases.

Since $\hat{w}$ is the weight of the largest object, Johnson's observation is proven. Thus, one could replace G by another graph with a similar structure but with only $\hat{w}(w^*-1)+1$ nodes and thus improve both the memory and runtime performance of the procedure if the new number of nodes is less than W .

Using similar reasoning, we go substantially further. First, we redefine the lengths of the arcs in G above to represent the amount of wasted profit relative to a (not necessarily realizable) path comprised entirely of best arcs. Note that this redefinition of arc lengths reverses the previous ordering of the lengths of different paths between nodes 0 and any other j . The optimal solution is thus now the shortest path from 0 to W in the new G , with arc lengths given by $w_k (p^*/w^* - p_k)$ for arcs which correspond to type k objects. Note that all arcs have non-negative length and best arcs are the only ones that have zero length. For simplicity, we will assume that the best arc type is unique; the case of multiple best arc types is in fact easy to handle in the procedure.

We now note that every pair of nodes j and $j+w^*$ in G are connected by a zero length arc and hence that if a path of length L from 0 to j exists then a path of length L from 0 to $j+w^*$ exists too. For W sufficiently large, in particular, for $W > w^* \hat{w}$, it is sufficient to consider a new graph, H , with only w^* nodes corresponding to the original node numbers modulo w^* . This gives rise to a graph of the type shown in Figure 2 for the same example given in Figure 1 except that W is now assumed to be much larger. (This graph is in fact a Cayley color graph for the cyclic group of order w^* .)

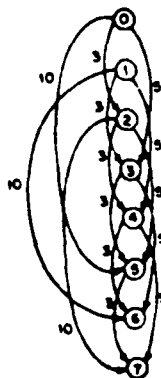


Figure 1. A graph for solving the Knapsack Problem

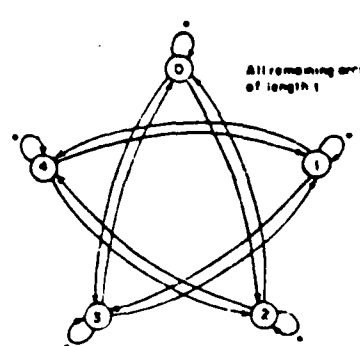


Figure 2. Cyclic graph for Knapsack Problem

Note that this graph contains arcs both from higher numbered nodes to lower numbered nodes and vice versa, unlike G which contained only arcs from lower numbered nodes to higher numbered nodes. Also, note that arcs appear as zero length self loops and can thus be ignored. Since all remaining arcs are of positive length, all cycles are of positive length and the shortest path problem is well defined. Thus, we have almost reduced the problem to one of finding a shortest path in a graph with w^* nodes instead of W nodes, the resultant path augmented by the appropriate number of best arcs to form the optimal solution.

The only loose end to tie up is what to do in the case where $W < w^*\bar{w}$. In this case, it is possible that the shortest path found in H above may correspond to a path to a node greater than W ; i.e., the "appropriate" number of best arcs is negative. Put another way, the reduction from G to H was based on the assumption that node j is dominated by node $j-w^*$ in G since the lower numbered node could always extend its paths through the higher numbered node via a zero length arc. This is not true, however, if $j-w^* > W-\bar{w}$ since a path to W may exist from the lower numbered node but not from the higher numbered node.

We can handle this problem by adding an additional $\bar{w}$ nodes to H , corresponding to the nodes $W-\bar{w}+1, \dots, W$ in G and adding arcs from appropriate nodes $0, \dots, w^*-1$ to these new nodes corresponding to each type of object. Finally, we add arcs between lower numbered nodes and higher numbered nodes corresponding to each type of object, exactly as in G . This entire transformation, which at first appear complex, amounts simply to collapsing G by removing the nodes in the range w^* through $W-\bar{w}$ since the removed nodes are not essential to finding the optimal solution.

Figure 3 shows part of such a graph in a case where $w^*=5$, $\bar{w}=6$, and $W=22$. Shown are only the arcs corresponding to an object with weight 2. Thus, even with this complication, both the runtime and storage have been reduced to a polynomial in M , w^* , and $\bar{w}$; i.e., there is no functional dependence on W . Note that if $W < w^*\bar{w}$ then we simply solve the problem using G ; no reduction is necessary.

In the sequel, we will limit the discussion to the case where $W > w^*\bar{w}$. The case where $W < w^*\bar{w}$ follows by the above transformation, adding storage and runtime proportional to $\bar{w}$. The final phase of the procedure in this case is an ordinary shortest path algorithm where nodes w^* through $w^*\bar{w}$ are each scanned sequentially.

ALGORITHM DESCRIPTION

We now describe the actual procedure in more detail and analyze its storage and runtime.

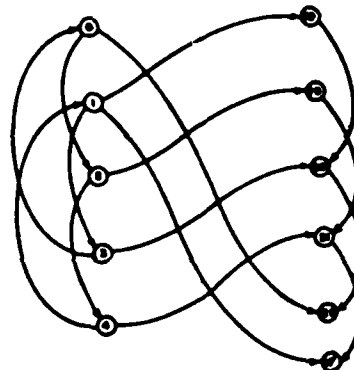


Figure 3. A graph for solving the Knapsack Problem when $W < w^*\bar{w}$

The following is the essence of the procedure:

Step 0: (Initialization)

$$d_0 = 0$$

$$d_i = \infty \quad i=1, \dots, w^*-1$$

where d_j is the current estimate of the length of the shortest path from 0 to j .

Step 1: Find i , the best node to scan next. This is discussed in detail below.

Step 2: (Scan node i)

For $k=2, \dots, M$

Let $j=(i+w_k)$ modulo w^*

If $d_j > d_i + L_k$ then $d_j = d_i + L_k$ and $PR_j = k$

where L_k is the length of an arch corresponding to a type k object ($L_k = w_k(p^*/w^* - p_k)$ as defined above) and PR_j records the type of the arc used to label node j and hence keep track of the optimal solution.

Step 3: Return to step 1 if any nodes remain to be scanned; otherwise stop.

It is clear that the storage required for this procedure is linear in w^*+M . If Step 1 is sufficiently simple and each node is only scanned once, the procedure's runtime will be dominated by Step 2 and will be proportional to Mw^* . This is in fact the case, as we will see.

A node, j , must be scanned if its label, d_j , is improved (reduced) as it may then improve the labels of other nodes. If we are to ensure that each node is scanned at most once, we must defer scanning it until its label cannot be improved. When all arc lengths are positive, as they are here, this objective is achieved by scanning the nodes in ascending order of d_j . This is Dijkstra's well known shortest path algorithm.

In the general case, Dijkstra's algorithm requires that in Step 1 we find the as yet unscanned node with the smallest label. If this were done naively, it would require an examination of w^* labels and the overall procedure would require $(w^*)^2 + Mw^*$ operations. In this case, however, we can implement Step 1 carefully and do much better.

We form "buckets" of nodes to be scanned and place nodes to be scanned together in the same bucket if they have similar labels. If the width of the buckets is q , then we place a node with label d , where $(k-1)q < d < kq$ in the k^{th} bucket. We then simply work our way through the buckets in ascending order of bucket number scanning the nodes in each bucket in arbitrary order. Thus, there is no explicit search in Step 1. If the width of the bucket is no greater than the length of the shortest arc in the graph, then any node labeled by the node currently being scanned will reside in a higher numbered bucket and hence, each node will be scanned at most once.

The only problem with this procedure, which is well known, is that the number of buckets itself may become very large, in particular much larger than the number of nodes, if the longest arc is many times greater than the shortest. In this case, a great deal of storage and running time may be wasted in dealing with empty buckets. We now show that in this case fewer than w^* buckets are required in all cases.

We begin by reindexing the arcs out of each node corresponding to the $M-1$ remaining types of object in increasing order of length. (Recall that the arc corresponding to the best type of arc was removed from explicit consideration.) Thus, L_1 is the length of the arc which corresponds to the second best type of object, etc. The w_i and p_i are reindexed correspondingly. We will refer to arcs of length L_i , weight w_i , and profit p_i after this reindexing as arcs of type i (or as arcs corresponding to objects of type i). For simplicity, we will assume $L_i \neq L_j$ for $i \neq j$ although the procedure does not require this.

Consider the case where w^* is relatively prime to w_1 . By the definition above, arcs of

length L_1 are the shortest arcs in the graph. Since w_1 and w^* are relatively prime, a path containing at most w^*-1 arcs comprised entirely of arcs of type 1 exists from 0 to each other node. Thus, we know that no node need have a label greater than $L_1(w^*-1)$. So, w^*-1 buckets of width L_1 suffice since the shortest path will be no longer than the aforementioned one.

In the case where w^* and w_1 are not relatively prime, the above path will loop back to node 0 before reaching many of the other nodes. In this case, the following preliminary computation is carried out before setting up the buckets:

$$r = w^*$$

For $j=1, M-1$

$$g_j = r / \text{GCD}(r, w_j)$$

$$r = \text{GCD}(r, w_j)$$

where $\text{GCD}(i, j)$ is the greatest common divisor of the integers i and j and can be found using Euclid's Algorithm.

We observe that, corresponding to the paths comprised entirely of type 1 arcs above, there is now a tree rooted at 0, spanning all the nodes, and containing paths with at most g_1-1 arcs of length L_1 . An illustration of such a tree is given in Figure 4. In this case only nodes 0, 8, and 4 can be reached using only arcs of type 1 before the path cycles back to node 0. The number of nodes so reachable is g_1 (three in this case). The original set of w^* nodes is partitioned into g_1 parts which are then each partitioned into g_2 smaller parts corresponding to the nodes which are now reachable via arcs of type 2. The partitioning continues until all nodes are reached. The above can be proven rigorously in the general case using the properties of cyclic groups.

We now define buckets of variable width corresponding to the longest path in the tree defined above. Thus, there are g_1-1 buckets of width L_1 followed by g_2-1 buckets of width L_2 , etc. A total of $B = \sum_i (g_i - 1)$ buckets is required. Since the product of the g_i 's is w^* , $B \leq w^*$.

We have shown that less than w^* buckets are required. We now show that no node is scanned more than once. To do so, we show that the width of each bucket is no greater than the length of the shortest arc still permitted in a path.

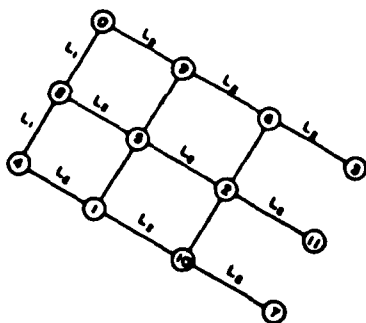


Figure 4: A spanning tree bounding the lengths of shortest paths

Note that the paths from 0 to j correspond to ordered selections of objects of aggregate weight j ; i.e., there are distinct paths corresponding to the selection of the same objects in a different order. We can, and should, consider only one path corresponding to each distinct collection of objects. This is easily done by keeping track of the last type of arc x in the path (which we are doing with the variable PR_x) and only considering arcs with the same or higher index (object type). Thus, we will find solutions corresponding to a selection of objects in non-decreasing order of type.

Now, consider a node in bucket i of width w_i . This node was either reached via a path in the tree considered above or via an arc whose length is greater than that of any in that tree. In latter case, the length of the smallest arc under consideration exceeds the width of any bucket. In the former case, since we are considering arcs in increasing order of length, again a node can only enter a bucket via an arc at least as long as the width of the bucket.

CONCLUSIONS AND REFINEMENTS

The basic procedure as stated is seen to have a storage requirement proportional to $w^* + w + M$ in the worst case and runtime proportional to $(w^* + w)M$ in the worst case. This is a significant improvement over procedures of this type currently in use. Several refinements can be made to the basic procedure, however, to improve its performance still further in specific cases.

If $w_i = w_j$ and $p_i < p_j$, object i can be eliminated from consideration. In fact, if w_i and w_j are congruent to each other modulo w^* and $p_i < p_j$, object i can be eliminated from consideration in paths between nodes i and j in the range 0 to $w^* - 1$ (but not other nodes in the case $w < w^* - w$.)

Note that at most $w^* - 1$ types of arcs can thus remain between the nodes 0 and $w^* - 1$ and both the storage and running time become functions only of w^* , not w or M . (In the case of inequality constraints, this dominance can be pushed still farther and object i can be eliminated if $w_i \geq w_j$ and $p_i \leq p_j$.)

Since the shortest path to any node is bounded by $v = (g_i - 1)L_i$, when scanning a node with label b , arcs of length greater than $v - b$ can be eliminated from consideration. Since arcs are scanned in ascending order of length, this is easily implemented by terminating the scan when the first such arc is encountered.

We are currently carrying out computational experiments to compare the efficiency of this procedure with others currently in use.

- [1] Horowitz, E. and S. Sahni Fundamentals of Computer Algorithms, Computer Science Press, 1978.
- [2] Denardo, E. and B.L. Fox, "Shortest Route Methods: 2 Group Knapsacks, Expanded Networks, and Branch-and-Bound" Operations Research vol. 27, no. 3 May-June 1979.
- [3] Johnson, E.L. "Integer Programming: Facets, Subadditivity, and Duality for Group and Semi-group Problems" IBM Research Report RC 7450, 12/78.

C.3 Generalized Augmenting Paths for the Solution of
Combinatorial Optimization Problems

Kershenbaum

Tenth IFIPS Conference on System Modeling and Optimization, August 1981, New York, Proceedings
published by Springer-Verlag, 1982

Generalized Augmenting Paths for the Solution of Combinatorial
Optimization Problems

Aaron Kershenbaum

Polytechnic Institute of New York

Abstract

Alternating chain procedures can be thought of as generalizations of the greedy algorithm in that instead of accepting the best remaining element, they seek to obtain a better augmentation by examining a wider range of alternatives. It is possible to generalize the notion of an augmenting sequence to include augmentations which are in effect trees as opposed to simply paths such that these augmentations are sufficient to guarantee optimality. Unfortunately, in the worst case, these trees are of exponential size. We examine the application of such generalized augmenting sequences to the solution of NP-complete problems and examine their effectiveness and efficiency.

Introduction

The theory of NP-completeness, which was first expounded by Cook [1], has led to a search for a unified treatment of combinatorial optimization problems. Cook was able to characterize a very large class of interesting and important problems as being equivalent in the sense that an efficient algorithm capable of finding an optimal solution to any one of these problems can be used to obtain optimal solutions to all of the others. Many papers by many authors and an excellent compendium [2] of problems in this class (as well as techniques for proving that a problem is in this class) have been published since Cook's seminal paper. Problems in this class are called NP-complete problems (or, more properly, NP-hard when they are optimization problems as opposed to decision problems).

Cook's results can be interpreted in several ways. One of these is to say that many clever people have spent many years trying and failing to find efficient algorithms for individual problems in this class. Surely one of them would have succeeded if, in fact, such algorithms existed. Hence, it is unlikely that such an algorithm will be found and it is tempting to stop looking for one. This leads to the development of heuristics for the solution of such problems [3] and to probabilistic methods [4].

An alternate interpretation is that this pessimistic view is justified only with respect to algorithms which guarantee optimal solutions and reasonable runtimes for all instances (input data sets) of a problem. In this paper we speak of an algorithm's runtime being reasonable if it grows polynomially rather than exponentially with the size of the problem. This does not preclude the existence of algorithms with guaranteed reasonable runtimes and which yield optimal (or near-optimal) solutions with high probability. Nor does it preclude the existence of algorithms

which guarantee optimal solutions and which have reasonable runtimes with high probability. There are many examples of both types of algorithms which are used in practice to solve specific NP-complete problems. Most important, the theory of NP-completeness does not preclude or even lessen the likelihood of the existence of algorithms which solve specific (nontrivial) instances of a problem and guarantee both an optimal solution and reasonable runtime.

In this paper, we explore this second, more optimistic, point of view and present a family of algorithms for the solution of an NP-complete problem. Some algorithms in this family have guaranteed reasonable runtimes. Others guarantee optimal solutions. While the algorithms are presented for the solution of a specific problem, the technique can be extended to the solution of other problems as well.

Matroid Theory

A specific way of approaching the solution of many combinatorial optimization problems is via matroid theory. The excellent book by Lawler [5] gives a complete treatment of this. Here we outline the fundamentals of this theory which are necessary for the presentation which follows.

A matroid is a couple (E, F) where E is a finite set of m elements:

$$E = \{e_j \mid j = 1, 2, \dots, M\}$$

and F is a family of independent subsets of E . The notion of independence is quite general. We require, however, that it satisfy two properties:

P1: Every subset of an independent set is independent, i.e., if

$$I \in F \text{ and } J \subset I \text{ then } J \in F$$

P2: If I_P and I_{P+1} are independent subsets of E containing P and $P + 1$ elements, respectively, then there exists an element, $e \in I_{P+1}$ ($e \notin I_P$) such that $I_P \cup \{e\}$ is an independent set containing $P + 1$ elements.

Given two matroids, (E, F_1) and (E, F_2) , defined on the same set of elements, but using two different notions of independence, we define an intersection of them to be any subset, $I \subset E$, such that $I \in F_1$, and $I \in F_2$. This definition can be extended to cover three or more matroids as well.

Many combinatorial optimization problems can be thought of as finding the best independent set in a matroid or the best intersection of two or more matroids. If weights, w_j , are associated with the elements, e_j , in E , then one can speak of the best set as being the one with largest total weight. The maximal (or minimal) spanning tree problem can be thought of as finding the maximum (or minimum) weight independent set in a matroid (E, F) where E is the set of edges in the graph and F is the family of forests. A forest is defined to be a set of 0 or more

edges which do not contain a circuit. As another example, Lawler [5, p. 304] shows that the Traveling Salesman Problem can be thought of as finding the best intersection of three matroids. The problem of finding the maximum weight intersection of three matroids has been shown to be NP-complete [2]. Lawler shows [5, p. 364] that the problem of finding intersections of four or more matroids can be reduced to that of finding intersections of three. There are many other combinatorial optimization problems which can be naturally thought of as matroid intersection problems. The theory of NP-completeness assures us that all problems can be thought of in this way.

We will consider one of the simplest possible 3-Matroid Intersection Problems in the sequel for the sake of clarity. The problem considered is the Three Dimensional Assignment Problem (TDAP). In this problem, we are given N people, N jobs, and N days. There is a cost, C_{ijk} of having person i doing job j on day k . Each person is to do only one job, each job is to be done only once, and only one job is to be done on a day. Formally the problem is:

$$\text{Minimize } Z = \sum_{i,j,k} C_{ijk} X_{ijk}$$

such that

$$\sum_{i,j} X_{ijk} = \sum_{i,k} X_{ijk} = \sum_{j,k} X_{ijk} = 1 \text{ for } i,j,k = 1,2, \dots, N \quad X_{ijk} \in \{0,1\}$$

Thus, setting X_{ijk} to 1 corresponds to having person i do job j on day k . This problem can be viewed as an intersection of three partition matroids. Given a set of elements, E (in this case, the X_{ijk}), a partition matroid can be defined by a partition of E and a vector, A , constraining the number of elements of E which may be selected from any part of the partition. Formally, we have the partition of E into subsets E_j , $j = 1, \dots, k$, where

$$\bigcup_j E_j = E \quad \text{and} \quad E_i \cap E_j = \emptyset \text{ for } i \neq j$$

and an integer vector $A = \{a_j \mid j = 1, \dots, k\}$

A matroid (E, F) is then defined where F consists of all subsets, I , of E formed by selecting no more than a_j elements of E_j .

In the case of the TDAP, the first partition of the X_{ijk} is by person, i.e.,

$$E_i = \{X_{ijk} \mid j = 1, \dots, N; k = 1, \dots, N\}$$

and $a_i = 1$ for all i . The independent sets in this first matroid correspond to assigning each person at most one job. Similarly, two more matroids can be defined to constrain jobs and days. Intersections of these three matroids correspond to feasible partial assignments and intersections of maximum cardinality correspond

to feasible complete assignments. If we define weights w_{ijk} associated with the x_{ijk} :

$$w_{ijk} = C - C_{ijk}$$

where C is larger than any C_{ijk} , then the maximum weight intersection corresponds to the optimal solution to the TDAP.

Augmenting Paths

We now define a family of algorithms for the solution of matroid intersection problems. These are generalizations of the basic procedure given in [6].

Given a matroid (E, F) (and hence a notion of independence) and a (not necessarily independent) subset S , of E , we define the span of S , denoted $sp(S)$, as S together with all elements of E not independent of the elements in S , that is

$$sp(S) = \{ e \mid I \cup \{e\} \notin F \text{ where } I \text{ is any independent subset of } S \}$$

If S is an independent set and $e \in sp(S)$ then e forms a unique cycle, which we denote by $C(e)$, with S . A cycle is a dependent set which becomes independent if any element is removed from it.

If the matroid intersection problem only involves two matroids, we can obtain a maximum weight intersection by producing a sequence of intersections, $I^{(K)}$, containing K elements, for $K = 1, 2, \dots, m$. Each $I^{(K)}$ is the maximum weight intersection containing K elements. The algorithm which produces the $I^{(K)}$ is called an augmenting path procedure because it augments $I^{(K)}$ to produce $I^{(K+1)}$ by finding the longest path in the graph $G^{(K)}$ defined below.

We define $G^{(K)}$ to be a bipartite graph with nodes corresponding to the elements, e_j , of E plus distinguished start and finish nodes, a and z . Directed arcs are defined as follows:

$$\begin{array}{ll} (a, i) & i \in E - sp_1(I^{(K)}) \\ (i, j) & i \in E - I^{(K)}, j \in C^{(2)}(i) \\ (i, z) & i \in E - sp_2(I^{(K)}) \\ (j, i) & i \in E - I^{(K)}, j \in C^{(1)}(i) \end{array}$$

Paths from a to z correspond to augmentations of $I^{(K)}$, that is, to sets of elements to be added or deleted from $I^{(K)}$ to produce an intersection with $K + 1$ elements. Notice that all a to z paths go alternately through nodes not contained in $I^{(K)}$ (which are to be added to $I^{(K)}$) and nodes in $I^{(K)}$ (which are to be deleted). Note also that there is one more node of the former type than there is of the latter and hence an augmentation results. If we associate lengths with the arcs equal to the weights of the elements which the nodes correspond to (positive for elements to be added and negative for elements to be deleted), then the length of a path corresponds to the incremental weight of the augmentation. The longest

path results in an optimal augmentation. Such a path can be found using a shortest path algorithm suitably modified to find longest paths. $G^{(K)}$ contains no positive cycles and so the algorithm converges.

These augmentations do, indeed, result in intersections. As one passes through nodes from a to z we see that an element is added preserving independence in the first matroid but not the second. An element is then deleted restoring independence in the second matroid and hence the intersection. A node is then added which, because of the deleted node, maintains independence in the first matroid. This process continues until the added element maintains independence in the second matroid as well as the first, thus completing the augmentation.

As an example, consider a two dimensional assignment problem (involving, say, only people and jobs.) The W_{ij} 's for this problem are given in Figure 1. $I^{(2)}$ is clearly 11,22, i.e., person 1 assigned to job 1, and person 2 assigned to job 2. $G^{(2)}$ is shown in Figure 2. The arc lengths are shown as are the lengths of the longest paths to each node from node a . The longest a to z path is $a, 11, 12, 22, 23, z$ which corresponds to deleting 11 and 22 from the intersection and adding 31, 12, and 23. The length of this path, 7, is the difference between the weight of $I^{(3)}$ and $I^{(2)}$. A complete description of this process and a proof of its validity is given in [5].

Generalized Augmenting Paths

In the graph shown in Figure 2, one can obtain an optimal augmentation (i.e., one which takes us from an optimal assignment of K elements to an optimal assignment of $K + 1$) because:

1. If the current intersection is not maximal then an augmenting path exists.
2. The labels given to the nodes during the longest path algorithm completely summarize the augmenting paths.

We now wish to generalize the notion of an augmenting path, and hence the entire procedure, to the problem of the intersection of three matroids. One way of doing this is to "freeze" one of the matroids and only consider alternating sequences within the other two. In this case the first node, s_1 in an augmenting path would be independent of $I^{(K)}$ in two of the three matroids (or in all three, in which case it is the only node in the augmenting path). Say s_1 is independent of $I^{(K)}$ in the first and third matroids. We could then freeze the third matroid and maintain the same span within the third matroid throughout the augmenting path. Thus, the deletion of s_i for i even reduces this span and the addition of s_i for i odd restores it. We thus reduce the search space to two matroids and the same polynomial bounded procedure will work. Note that, alternatively, we could have considered the first matroid frozen. Indeed, it is so frozen in the two matroid intersection algorithm. Thus, there are three types of augmenting paths, one for each matroid

within which s_1 is dependent. Unfortunately, while this procedure is polynomial bounded, it does not guarantee optimal solutions as there are augmentations which have no such corresponding augmenting path.

In order to guarantee that all augmentations are explored, we must relax the definition of an augmenting path still further to include cases where independence is not necessarily restored by the deletion of s_i for i even. Thus, an augmenting path may start with any element, s_1 , which is independent of $I^{(K)}$ in at least one of the matroids. Unlike the procedure given for two matroids, one may begin with independence in any matroid. Consider the graph shown in Figure 3 corresponding to two augmenting paths, Path 1 and Path 2, for the partial assignment 111,222,333 (i.e., person 1 to job 1 on day 1, etc.) in a TDAP. These paths are not strictly comparable in that they exclude different elements along the way. Thus in Figure 2, when node 22 is labeled using the path a,32,22 it is equivalent (in terms of how the path can continue, not necessarily in terms of the numerical value of the label) to being labeled using the path a,31,11,12,22. In Figure 3, however, when node 111 is labeled using the path a,411,111 it is different from labeling 111 using the path a,154,111 because different continuations of these paths are possible. Thus starting with a,411,111 we can continue to 152 but not 215 and, conversely, starting with a,154,111 we can continue with 215 but not 152. Thus, Path 1 and Path 2 are not comparable in terms of their lengths only.

Such paths must also be compared in terms of their spans. We note that if two paths from a to some node i result in sets having identical spans then the same continuations of both paths are possible. (This was the case for intersections of two matroids.) Indeed, it is possible for paths to have slightly different spans and still have the same set of possible continuations. In particular, if the only difference in the intersections of the spans of two paths are nodes outside the intersection of the spans of $I^{(K)}$, then the paths are comparable. We can thus generalize the augmenting path procedure to consider all undominated a to z paths where one path dominates another only if it has the same continuations and a larger length.

The notion of a path itself, however, must be generalized as well. In the case of 3 matroids, not all augmentations correspond to paths. We see an example of this for a TDAP. The augmentation $\{412,234,341,123\} - \{111,222,333\}$ does not correspond to any path in the conventional sense. It is possible however, to extend the augmenting path procedure to include such augmentations by extending the notion of a path.

We define a generalized augmenting path with respect to an intersection $I^{(K)}$ to be a sequence of nodes $S = (s_1, s_2, \dots, s_m)$ where $s_i \in E - I^{(K)}$ for odd i and $s_i \in I^{(K)}$ for even i . As before, $I^{(K)} + s_1 - s_2 + s_3 - \dots + s_m$ is an intersection. Also, the even s_i are deleted in order to remove dependencies created by the inclusion

Job Person	1	2	3
1	10	9	5
2	5	10	9
3	9	5	1

FIGURE 1 - Cost Matrix

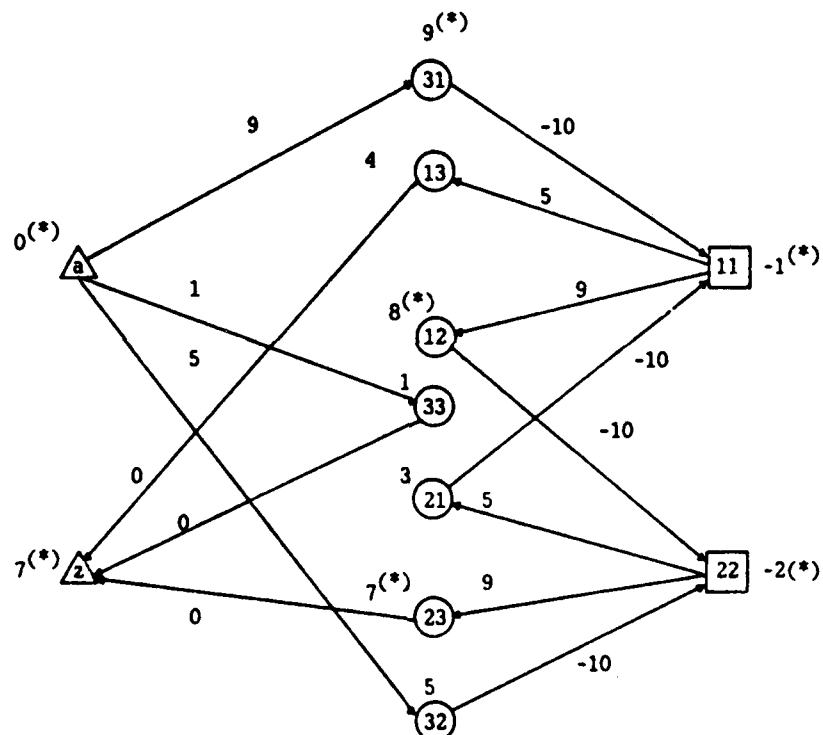
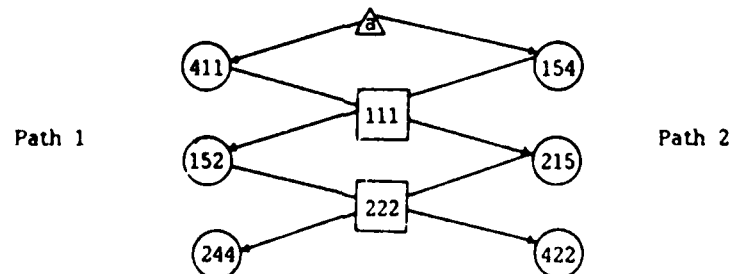
FIGURE 2 - Bipartite Graph $G^{(2)}$ 

FIGURE 3

of the odd s_j . Now, however, the subsequences $I^{(K)} + s_1 - s_2 + \dots - s_j$ for even j need not correspond to intersections.

One can thus guarantee an optimal intersection as in the case of two matroids. The number of generalized augmenting paths one may need to consider, however, may grow exponentially with K . In practice, however, the number of such paths can be controlled at the expense of optimality. First, the length of any path, $(s_1, s_2, \dots, s_j)$, can be reduced by a penalty to account for the nodes which still must be deleted to restore the intersection. In the case of arbitrary matroids, this may be complex to compute. In the case of the TDAP, however, where 3 partition matroids are involved, and all cycles contain 2 elements, it is easily computed.

In some cases the above may keep the computations reasonable. In others, it may be necessary to reduce the number of paths considered by relaxing the definition of dominance. This will also result in a heuristic rather than an optimal solution. In the case of the TDAP, one such relaxation is to ignore differences in the spans outside the intersection of the span of $I^{(K)}$. This is motivated by the fact that we consider deleting elements in $I^{(K)}$ in order to include elements blocked by them.

We can thus consider a hierarchy of generalized augmenting path procedures with increasingly stringent dominance criteria and increasing runtime. A tradeoff between optimality and runtime is then available. We are currently investigating this tradeoff using the TDAP as an example.

Acknowledgment

This research has been supported by U.S. Army, CORADCOM, under contract DAAK-80-80-K-0579 and the National Science Foundation under Grant ENG7908120.

References

1. Cook, S.A., "The Complexity of Theorem-Proving Procedures," Proc. of Third Ann. ACM Symposium on Theory of Computing, 1971, p. 151-158.
2. Garey, M.R. and D.S. Johnson, Computers & Intractability. W.H. Freeman, 1979.
3. Sahni, S. and E. Horowitz, "Combinatorial Problems: Reducibility and Approximation," Operations Research 26(4), 1978.
4. Karp, R.M., "The Probabilistic Analysis of Some Combinatorial Search Algorithms," in Algorithms and Complexity, Academic Press 1976.
5. Lawler, E., Combinatorial Optimization: Networks and Matroids, Holt, Rinehart & Winston, 1976.
6. Edmonds, J., "Matroid Intersection," in Discrete Optimization I, North Holland Publishers Co. p. 39-49.

C.4 A Note on Finding Shortest Path Trees

Kershenbaum

Networks Journal, 1981

A Note on Finding Shortest Path Trees

Aaron Kershenbaum

*Department of Electrical Engineering, Polytechnic Institute of New York,
333 Jay Street, Brooklyn, NY 11201*

Two shortest path algorithms are compared and it is shown that, while one outperforms the other in practice, the former's running time is exponential in the worst case while the latter's is polynomial. A procedure which constructs such worst case examples is given.

In the excellent paper "A Computational Analysis of Alternate Algorithms and Labeling Techniques for Finding Shortest Path Trees" by Dial, Glover, Karney, and Klingman [1] the statement is made "that the label-setting algorithm has a worst case bound, that is an order of magnitude better than the label-correcting algorithm." It is, in fact, easily shown that this statement is true for the performance of Algorithm C1 relative to the performance of a label setting method. In Algorithm C1, since the sequence list is managed in a FIFO fashion, paths are generated in order of the number of arcs in them. Each node can thus be scanned at most N times (creating paths with K arcs, $K = 1, \dots, N$), where N is the number of nodes. Since nodes are scanned only once in label setting algorithms, the result follows.

The situation is surprisingly different with respect to Algorithm C2. I have used this algorithm to find routes in very large, very sparse real networks (thousands of nodes and average nodal degree between 2 and 3) with a variety of length functions (generally distance-related) and have found it to outperform all others. This is exactly as the authors indicated. The worst case behavior of this algorithm, however, is exponential!

Consider, for example, the network in Figure 1. Suppose that the list of nodes adjacent to node 1 (the root) is in ascending order of node number and that the list of nodes adjacent to all other nodes is in descending order. We find that every time a node is scanned, node 2 is relabeled and scanned. In fact, node 2 takes all labels between 20 and 5 and is scanned every time it is relabeled. Here $N = 6$ and node 2 is scanned 2^{N-2} times. Networks with this characteristic and of arbitrary size can be formed as follows.

Step 1: Start with a network with two nodes, 1 and 2, and two arcs (1, 2) and (2, 1) with $L(1, 2) = 1$ and $L(2, 1) = 1$, where $L(i, j)$ is the length of the arc from i to j .

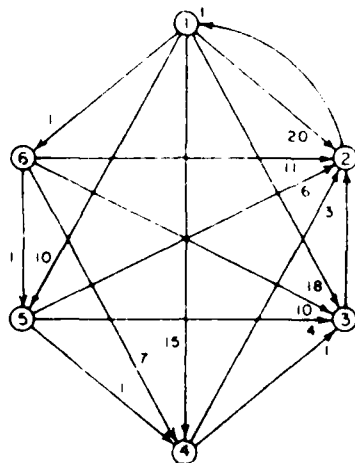


FIG. 1.

Step 2: For $K = 3$ to N alter the network as follows:

- (a) Add node K .
- (b) Add arcs (K, i) for $i = 2, \dots, K-1$ with $L(K, i) = L(1, i)$.
- (c) Set $L(1, i) = L(1, i) + 2^{N-3} + 1$.
- (d) Add an arc $(1, K)$ with $L(1, K) = 1$.

During the execution of algorithm C2, the node 2 takes all labels from $N-1$ to $2^{N-2} + N - 2$ and is scanned 2^{N-2} times.

It should be noted that this pathological situation is a product not only of the unusual arc lengths but also of the ordering of the adjacency lists, in particular that lists of adjacent nodes are ordered with the nearest node first for all nodes except the root. If the adjacency lists were ordered oppositely, each node would be scanned only once. Thus, we see a strong dependence of the performance of Algorithm C2 on a preordering of the links. The general idea would be to order links in such a way that they would be likely to be scanned in ascending order of their labels. Alternatively, adjacency lists could be ordered nearest node first and the sequence list management could be altered only slightly to add the entire list of relabeled nodes to top of the sequence list in forward order (rather than the reverse order implied in Algorithm C2) of the adjacency list of the node being scanned.

In summary, we note that Algorithm C2 has excellent behavior in practice and preprocessing is probably warranted only if multiple applications of the algorithm are being done (e.g., from multiple sources).

Reference

- [1] R. Dial, F. Glover, D. Karney, and O. Klingman, "A Computational Analysis of Alternate Algorithms and Labeling Techniques for Finding Shortest Path Trees," *Networks*, 9, 215 (1979).

Received January 1980

Accepted February 4, 1981

C.5 Probabilistic Analysis of Algorithm Performance

First Semiannual Technical Report, March 1981

IID. Probabalistic Analysis of Algorithm Performance

The problem of finding an optimal set of repeater locations covering a given set of terminal sites, or set of potential repeater locations and a covering matrix specifying which terminal sites can be covered by each repeater location, is an important problem in the design of multihop packet radio networks. The problem is an instance of the classic set covering problem which has many other applications as well. The set covering problem is known to be NP-complete and as such it is unlikely that an algorithm will be found which can guarantee an optimal solution and also guarantee a reasonable running time, that is a runtime which grows polynomially in the number of terminals and repeaters.

The theory of nondeterministic polynomial completeness (NP-completeness) basically states that there is a large class of problems, which includes almost all optimization problems of interest in the area of network design, which have the property that a solution to one could be used to obtain a solution to all the others in a reasonable amount of time. It has been shown that one problem, the Satisfiability Problem, has the property that all problems in this class can be transformed into an instance of it. Thus, an algorithm which could obtain an optimal solution to the Satisfiability Problem, could be used to solve all the others. Techniques have been developed for proving that many other problems are NP-complete. Proofs have been given for the NP-completeness of over 200 other well-known problems.

In our previous work we reported on results obtained using a new set covering algorithm. The algorithm obtained optimal solutions for moderate sized problems with up to 300 node networks and cover-

age related to distance. That is, if the average matrix reflected the fact that a repeater was far more likely to cover a nearby terminal than one farther away, then the algorithm worked very well, converging to an optimal solution very quickly.

If on the other hand, the coverage matrix was random, the algorithm ran much more slowly and only problems with up to 50 terminals could be treated efficiently. In examining why the algorithm had difficulty treating problems with random data, we observed that the difficulty stemmed from the large number of optimal and near optimal solutions available. The algorithm had no difficulty finding an optimal solution but, rather, had difficulty in verifying the solution was optimal in a reasonable amount of time due to the presence of a large number of alternate solutions of comparable quality. This led us to conjecture that it might be possible to develop algorithms which had a reasonable running time and which could find optimal time or near-optimal solutions with high probability. It is known that asymptotically, as the size of the problem becomes infinite, there exists algorithms which give optimal results almost always. These results are an outgrowth of the same symmetry which makes it difficult to verify the optimality of the solutions produced by our set covering algorithm.

These results are encouraging but of no direct use in solving network design problems since real problems are of finite size and nothing was said about how fast the algorithms converge probabilistically to an optimum; i.e., what the probability is of their finding the optimum, or a solution within some bound of the optimum. We thus sought to investigate what could be said about the probabilistic

performance of known heuristics for problems of moderate size. We chose the set covering problem mentioned above as the first problem to investigate. The probabilistic model of this problem follows.

We are given a set $T = \{t_1, t_2, \dots, t_N\}$ of N terminals, a set $R = \{r_1, r_2, \dots, r_M\}$ of M potential repeater sites, and an $N \times M$ covering matrix, C , where c_{ij} is 1 if t_i can be covered by r_j . We seek a subset of R containing as few repeaters as possible and covering all terminals. The elements of C are chosen randomly and independently. Specifically, we assume $\Pr\{c_{ij}=1\}=p$ and $\Pr\{c_{ij}=0\}=1-p=q$.

We define R_j , the set of terminals covered by r_j , by:

$$R_j = \{t_i | c_{ij} = 1\}$$

Similarly, we define T_i , the set of repeaters covering t_i , by:

$$T_i = \{r_j | c_{ij} = 1\}$$

A cover of the terminals is then defined as a subset $S \subset R$ satisfying:

$$\bigcup_{r_j \in S} R_j = T$$

A minimum cover, S^* , is then a cover containing as few elements as possible, i.e.:

$$\bigcup_{r_j \in S^*} R_j = T$$

and $|S^*| \leq |S|$ for all S such that $\bigcup_{r_j \in S} R_j = T$.

Note that the minimal cover is not in general unique. Indeed our approach rests on the existence of a large number of minimal and near minimal covers.

In order to analyze the probabilistic behavior of a heuristic we must do two things. First, for a given set of values of N , M , and p , we must find the probability that $|S^*| = K$, that is, the probability

that there exists a minimal covering containing exactly K repeaters. Let

$$P(N,M,K) = \Pr\{|S^*| = K\} \text{ for given } N, M, \text{ and } p.$$

Next we must find $P_{\text{ALG}}(N,M,K)$, defined as the probability that the algorithm will find a solution with exactly K repeaters. A figure of merit for evaluating the probabilistic behavior of the algorithm is then

$$F_{\text{ALG}} = \frac{\sum_{K=0}^M KP_{\text{ALG}}(N,M,K) - \sum_{K=0}^M KP(N,M,K)}{\sum_{K=0}^M KP(N,M,K)} \quad (\text{HD.1})$$

This is simply the relative error made by the algorithm, i.e., the difference between the average number of repeaters in a covering found by the algorithm and the minimum number required, normalized by the minimum number required. We believe that $\lim_{N \rightarrow \infty} F_{\text{ALG}} = 0$ for any M , and p , and any reasonable algorithm where a reasonable algorithm is defined as one which stops when it has a cover. In particular, we believe $F_{\text{ALG}} \rightarrow 0$ for an algorithm which picks repeaters randomly until a covering is obtained.

We begin by evaluating $Q(N,M,K)$, the probability that a randomly chosen set of K repeaters covers all N terminals. Notice that this is not the same as the probability that an algorithm picking repeaters at random will stop after K repeaters since in some cases fewer than K will also suffice and the algorithm will find them. Also, $Q(N,M,K)$ is a cumulative distribution, that is it relates to the probability that K or fewer repeaters are required for a covering rather than exactly K . We do have, however, that

$$Q(N,M,K) \leq \sum_{L=0}^K P(N,M,K) \quad (\text{IID.2})$$

that is, the probability of a randomly chosen set of K repeaters covering all N terminals is less than or equal to the probability of the existence of a covering containing K or fewer terminals. Thus we have,

$$P(N,M,K) > Q(N,M,K) - Q(N,M,K-1), \quad K > 1,$$

that is, the difference between two successive Q 's is a lower bound on the corresponding P .

$Q(N,M,K)$ is easy to evaluate because repeaters picked randomly are picked independently of one another and our random model assumed their original characteristics are independent. The probability of a single terminal not being covered by a single repeater is $q(=1-p)$. The probability of a single terminal not being covered by any of the K repeaters is q^K . The probability of at least one repeater of K covering a given terminal is then $1-q^K$. Finally, the probability that all N terminals are covered by the K repeaters is $(1-q^K)^N$. We thus have that

$$Q(N,M,K) = (1 - q^K)^N \quad (\text{IID.3})$$

Note that this is not a function of M except in the trivial sense that M must be no smaller than K . Figure IID.1 shows values of $Q(N,M,K)$ for $p=.5$. Figure IID.2 shows values of $Q(N,M,K) - Q(N,M,K-1)$ for the same examples. Both show a tendency for solutions to cluster about a narrow range of K . This is encouraging in that it implies that probabilistically, simple algorithms should do well.

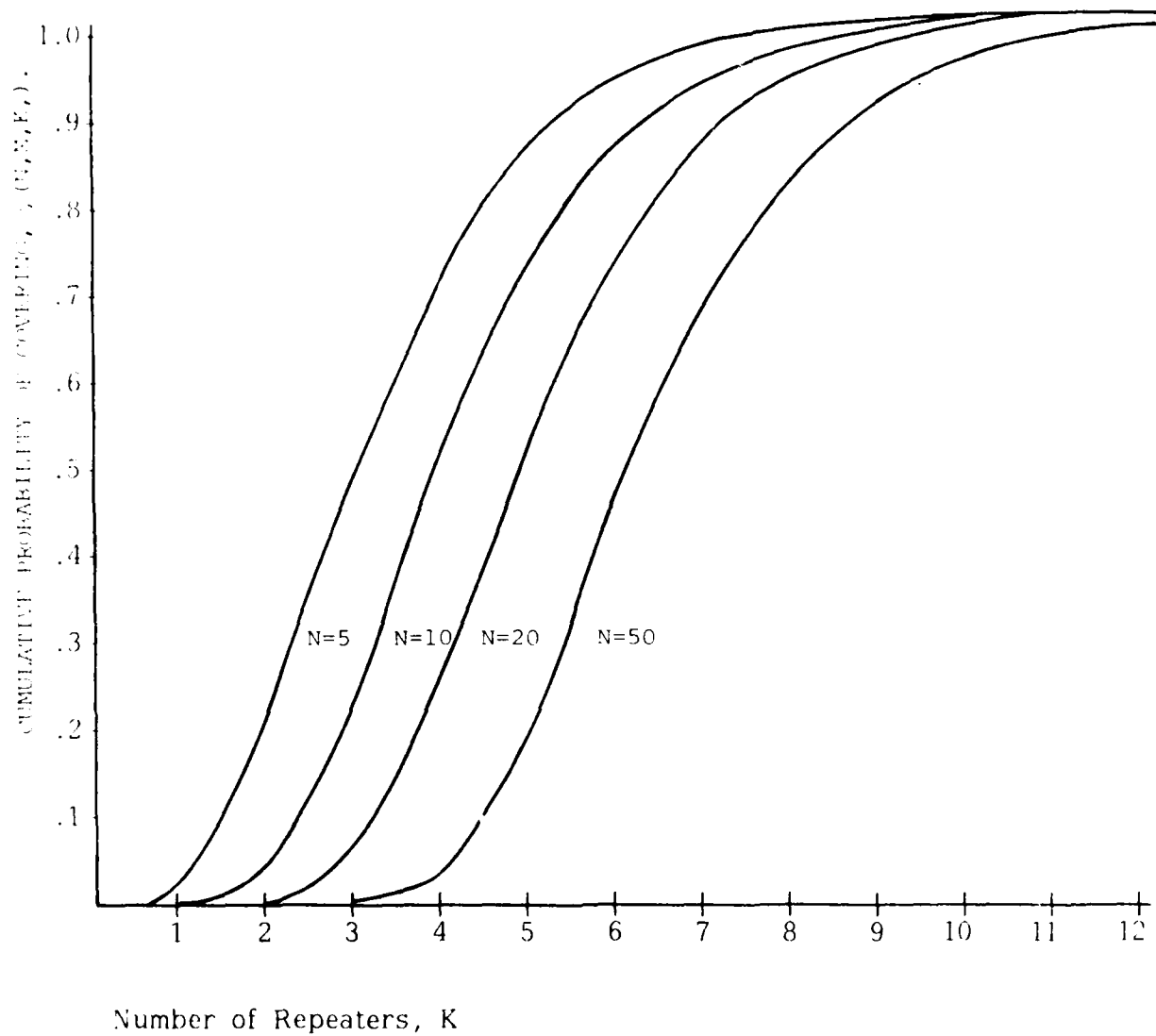


FIGURE HD.1 CUMULATIVE PROBABILITY OF A COVERING FOR
RANDOM ALGORITHM ($p=0.5$).

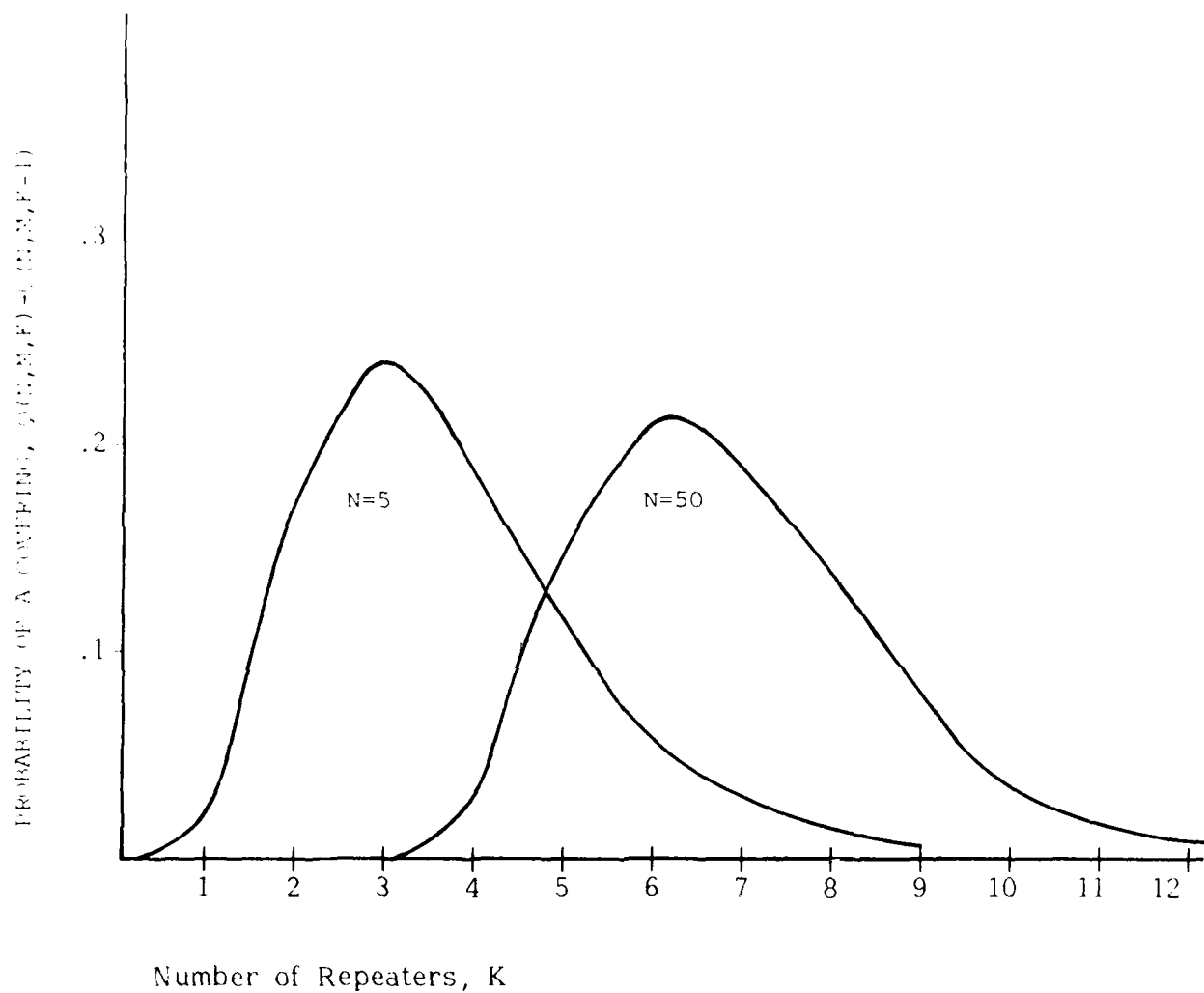


FIGURE HD.2 PROBABILITY OF A COVERING FOR RANDOM ALGORITHM ($p=0.5$).

As yet, we do not have an exact analytic expression for $P(N,M,K)$. We can get a tighter lower bound on it by evaluating the distribution of solutions from increasingly sophisticated algorithms. A simple minded algorithm would be to pick repeaters randomly until one obtained a covering. The probability of obtaining a solution with K or fewer repeaters using this algorithm is $P_1(N,M,K)$ given by:

$$P_1(N,M,K) = \sum_{i=0}^N \binom{N}{i} p^i q^{N-i} P_1(N-i, M-1, K-1) \quad (HD.4)$$

since the K^{th} repeater will cover i terminals with probability $\binom{N}{i} p^i q^{N-i}$ and the remaining $K-1$ of $M-1$ repeater must cover the remaining $N-i$ terminals. Note that this is, again, not a function of M except that M must be no smaller than K . The probability of obtaining a covering with exactly K repeaters is given by $P_1(N,M,K) - P_1(N,M,K-1)$.

A somewhat more sophisticated algorithm is to consider repeaters in a random order and to select a repeater only if it improves the chances of obtaining a covering more by picking it than by not picking it. Letting $P_2(N,M,K)$ be the probability of obtaining a covering containing K or fewer repeaters using this algorithm, we have

$$P_2(N,M,K) = \sum_{i=0}^N \binom{N}{i} p^i q^{N-i} \{ \max (P_2(N-i, M-1, K-1), P_2(N, M-1, K)) \} \quad (HD.5)$$

$P_2(N,M,K)$ is plotted for $N=5$ and $N=10$ in Figure HD.3. Along with it, the probability of obtaining a covering using K randomly chosen repeaters is plotted. Notice that this algorithm's performance is a substantial improvement over randomly picking repeaters. Notice also, that $P_2(N,M,K)$ is a function of M . This algorithm is very simple and has a reasonable running time even for very large N and

M. Our next objective is to compare it with an upper bound on $P(N,M,K)$ to see if it already has converged to near optimal performance probabalistically.

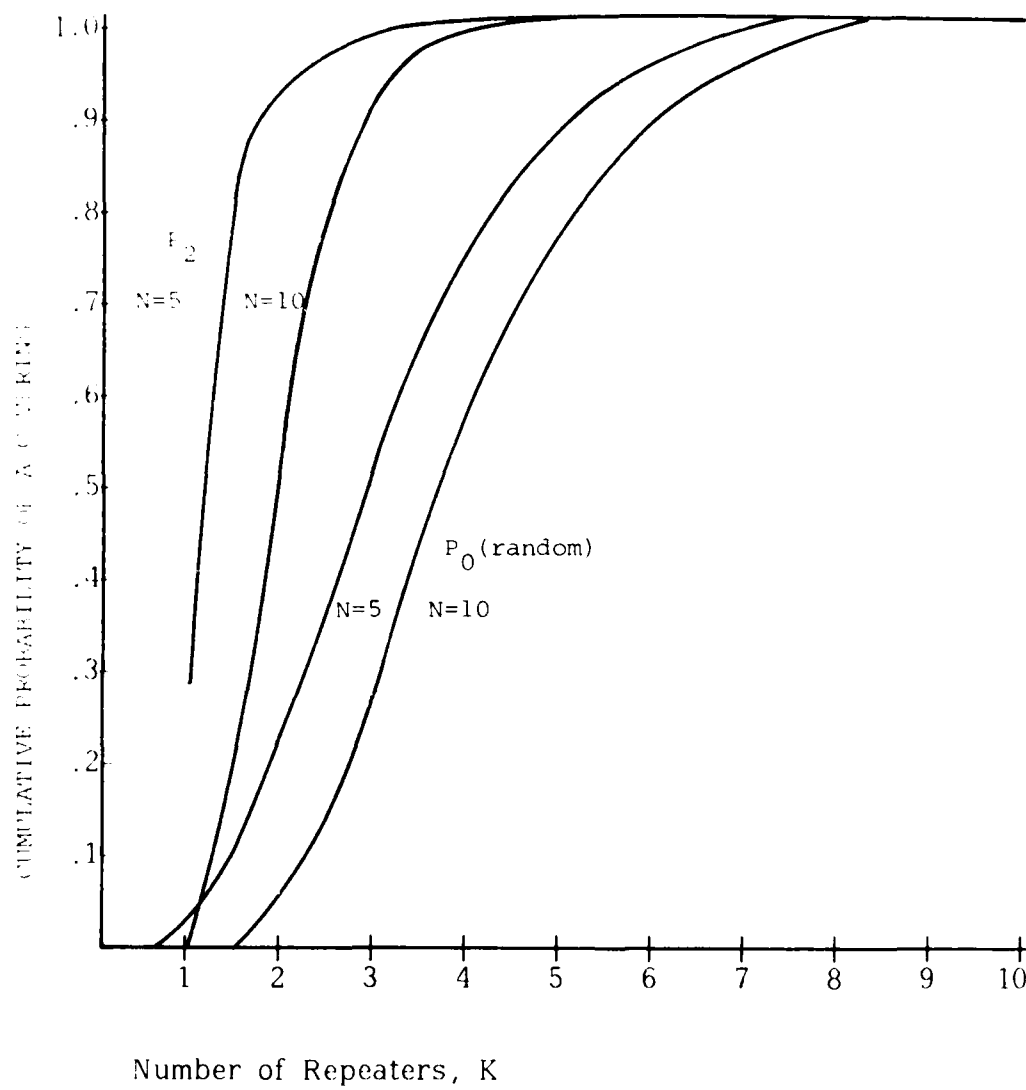


FIGURE IID.3 CUMULATIVE PROBABILITY OF A COVERING FOR MORE SOPHISTICATED ALGORITHM ($p=0.5, M=10$).

C.6 Probabilistic Analysis of Algorithms

Second Semiannual Technical Report, September 1981

C. PROBABILISTIC ANALYSIS OF ALGORITHMS

We have obtained results in this area indicating that even relatively simple algorithms will, with high probability, obtain near optimal solutions to certain difficult optimization problems. This is encouraging in that it implies that problems, previously considered intractable, may in fact be solvable, at least with high expectation of success. The results are also useful in guiding heuristics towards potentially fruitful areas within the solution space and in avoiding unnecessary effort in areas where little further progress can be expected.

Specifically, we have analyzed the intrinsic probabilistic behavior of the 3-Satisfiability Problem and found that, independent of the size of the problem, there is only a limited region of uncertainty, i.e., only a finite set of values where even the simplest sensible algorithm can make an error. This problem, which is described in detail below, is representative of a very large class of difficult combinatorial problems and can, in fact, be used as a vehicle for solving many other problems. Thus, we believe these results can be extended to other problems as well, most notably, to problems in network design such as facility location and link topology optimization. We are currently in the process of investigating such extensions and of examining the probabilistic performance of several specific heuristic algorithms.

C.1. Probabilistic Behavior of the 3-Satisfiability Problem

The 3-Satisfiability Problem has a simple and homogeneous structure which makes it easy to develop and study a probabilistic model. It also does not involve any numerical data which would otherwise obscure the nature and generality of the results which we obtain. These results can, however, be directly extended to optimization problems which do involve numerical data, as we will see in the following discussion.

In the ordinary satisfiability problem, we are given a Boolean expression, E , over m Boolean variables $v_1, v_2, \dots, v_m$ and we ask if there exists a set of truth values for the variables which will result in E taking the value True. Thus, each of the v_i can take either the value True or False (alternatively denoted by 1 or 0) and E will then take either the value True or False based on the values of the v_i . E is usually given in disjunctive normal form, i.e., as a set of clauses all of

which must be True in order for E to be true. Each clause contains one or more variables and is said to be true if at least one of the variables in the clause is assigned the value it takes in the clause. Thus for example, $E = (v_1 + v_2)(\bar{v}_1 + v_3)$ has 2 clauses. The first, $(v_1 + v_2)$ is true if either v_1 is True or v_2 is True. The second is True if either v_1 is False or v_3 is True. Thus E is satisfiable as the values $v_1 = \text{True}$, $v_2 = \text{True}$, and $v_3 = \text{True}$ satisfy both clauses and hence E. There are several other assignments of truth values v_1 , v_2 , and v_3 which will satisfy this E. The expression $(v_1)(\bar{v}_1 + \bar{v}_2)(v_2)$ on the other hand is not satisfiable.

The 3-Satisfiability Problem is a version of the ordinary Satisfiability Problem where all clauses contain exactly 3 variables. Garey and Johnson, in their book Computers and Intractability, show this problem is NP-complete by showing that any ordinary Satisfiability Problem can be transformed into a corresponding 3-Satisfiability Problem of roughly the same size. Thus the two problems are equivalent.

Many other problems can also be transformed into corresponding 3-Satisfiability Problems. In particular, combinatorial optimization problems such as the Traveling Salesman Problem or the problem of locating earth stations in a satellite network can be so transformed. Thus, we can produce a Boolean expression which corresponds to a particular optimization problem in the sense that if the expression is satisfiable then the optimization problem has a solution with a value less than or equal to a given constant (for minimization problems) or greater than or equal to a given value (for maximization problems). Furthermore, the satisfying truth assignment can be used to obtain the solution to the corresponding optimization problem directly. By altering the

value of the constant in the above transformation the optimal solution to the corresponding optimization problem can be found via binary search; i.e., if we know the optimum lies between values c_1 and c_2 , we try the value $c_1 + c_2/2$. This technique is known as thresholding.

One can determine if an expression is satisfiable by assigning all 2^m possible truth values to its variables. This approach is, of course, not practical for large values of m . There are many sophisticated techniques for answering the question whether or not a given expression is satisfiable, but all have running times which ultimately grow exponentially with the number of variables and thus are limited to problems of modest size.

We obtain here a technique which yields the a priori probability of an expression with a given structure being satisfiable given a probabilistic model of the space from which the problem is drawn. For problems where this probability, P_s , is very close to 1 or very close to zero, we need seek no further. Only for problems where P_s is significantly different from 0 or 1 need the question be investigated any further. In some cases, the P_s itself is all that we need. For example, if we have found, using a heuristic algorithm, a solution of value c_1 to a given maximization problem, and have determined that P_s is less than .01 for an expression corresponding to the existence of a solution of value greater than or equal to $c_1 + a$ (for a much smaller than c_1), then we are reasonably certain that there is not much to be gained from an attempt at further optimization. This is very important because it has often turned out that it is much harder to verify the optimality (or near-optimality) of a solution than to find an optimal solution especially when the solution space is rich and there exist many alternate near-optimal solutions.

We consider a uniform probabilistic model for the 3-Satisfiability Problem where all clauses are equally likely. Thus a problem is totally characterized by v and b , the number of variables and clauses, respectively. We will consider two models which are nearly equivalent. In one case, clauses are picked without replacement, i.e., a given clause can appear at most once in an expression. In the other case, clauses are picked with replacement.

Duplicate clauses do not affect the satisfiability of an expression. We can thus relate problems where the clauses are chosen with replacement to problems where the clauses are chosen without replacement by saying that a problem with v variables and b clauses chosen with replacement is the same as a problem with v variables and b' clauses chosen without replacement if the number of distinct clauses in the first problem equals b' . The relationship between the number of clauses chosen with replacement, b , and the average number of distinct clauses is given by the following argument.

Let X_b be the number of distinct clauses when b clauses are selected with replacement. Then $X_b \leq b$ and $X_b = X_{b-1} + \alpha$, where $\alpha = 1$ with probability P_{new} and $\alpha = 0$ with probability $1 - P_{\text{new}}$. P_{new} is the probability that the b^{th} clause is different from all the first $b - 1$. P_{new} is simply the ratio of the number of unchosen clauses to the total number of possible clauses. The number of possible clauses is

$$B = 8\binom{v}{3}$$

since each clause contains 3 variables each of which can take either of two values. Thus P_{new} is given by:

$$P_{\text{new}} = \frac{B - X_{b-1}}{B} \quad (C-1)$$

Let $f(b)$ be the average number of distinct clauses. Then

$$E(X_b | X_{b-1}) = X_{b-1} + P_{\text{new}} \quad (C-2)$$

and

$$f(b) = E(X_b) = f(b-1) + \frac{B - f(b-1)}{B} \quad (C-3)$$

Thus

$$f(b) = f(b-1) \times \frac{B-1}{B} + 1 \quad (C-4)$$

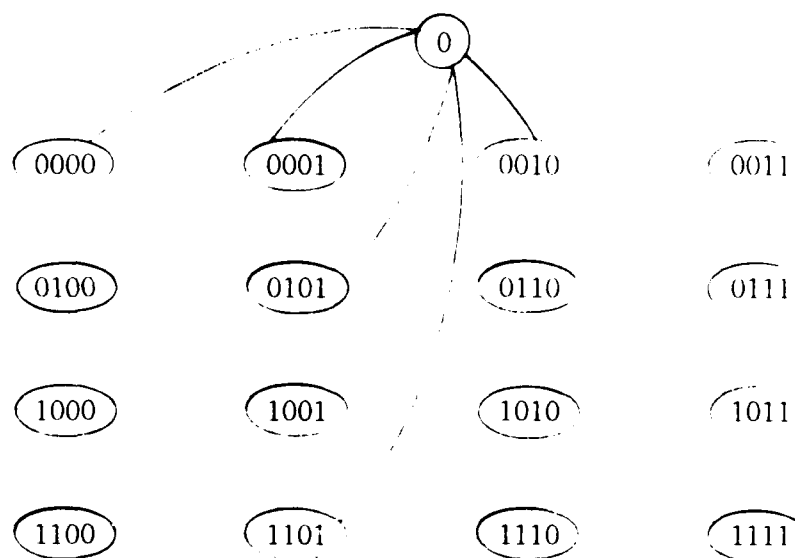
As an example, if $v = 10$, then $B = 960$ and $f(50) \approx 48.7$. As we will see, we are most interested in values of b near $5v$. For $v = 20$ and $b = 100$, $f \approx 99.5$. As v increases, f/b approaches 1 very quickly. Thus, there is little difference between choosing clauses with or without replacement.

C.2. Analytical Method 1

We now turn to the problem of estimating $P_s(v, b)$, the probability of expression E , with v variables and b clauses, being satisfiable. Given a problem with v variables and b clauses, we define the truth assignment graph, G , associated with that problem to be a graph with $2^v + 1$ nodes. Nodes 1 through 2^v correspond to the possible truth values for the v variables and node 0 is a distinguished pseudonode. Each clause in E is inconsistent with one eighth of the truth assignments, specifically those in which all three variables in the clause have values opposite to their values in the clause. G contains an arc between node 0 and any node corresponding to a truth assignment which does not satisfy E . We thus proceed through E clause by clause and add arcs between node 0 and nodes which do not satisfy the clause. There are 2^{v-3} arcs corresponding to each clause. In general, arcs

"belonging" to different clauses overlap, i.e., there will be truth values inconsistent with more than one clause. We do not show the multiplicity of these arcs. Thus, G will in general contain fewer than $b \times 2^{v-3}$ arcs. We are interested in G because P_s , the probability E is satisfiable, is precisely equal to the probability that G is not connected. Figure C.1 shows a truth assignment graph.

The problem of whether a random graph is connected or not has been studied extensively by Erdos and Renii. We will proceed along similar lines.



$$E = (v_1 + v_2 + v_3) (v_1 + v_2 + v_4) (\bar{v}_2 + v_3 + \bar{v}_4)$$

FIGURE C.1. A TRUTH ASSIGNMENT GRAPH

We define $A(v, k, b)$ to be the number of Boolean expressions whose corresponding truth assignment graphs contain k or more isolated nodes (i.e., nodes with no incident links). $P_s(v, b)$ is then given by

$$P_s(v, b) = 1 - \sum_{k=0}^{2^v} (-1)^k \frac{A(v, k, b)}{\binom{2^v}{b}} \quad (C-5)$$

where $B = 8\binom{V}{3}$.

This relation follows from the fact that if an expression is satisfiable then its corresponding truth assignment graph must contain at least one isolated node. Note that the alternating sum is required in order to account for the situations where graphs with more than k isolated points are included in terms with k or more isolated points. We seek the number of expressions with graphs with one or more isolated points. The alternating sum gives us the number with exactly zero isolated points.

We can write $A(v, k, b)$ as

$$A(v, k, b) = \sum_{i=0}^v \binom{v}{i} 2^i NM(v, i, k) B(v, i, k, b) \quad (C-6)$$

where $NM(v, i, k)$ is the number of ways of selecting k clauses which match in i specific variables (i.e., k clauses in which i specific variables take the same truth value in all k clauses) and $B(v, i, k, b)$ is the number of Boolean expressions whose corresponding truth assignment graphs have k or more isolated points and the isolated points match in i variables.

$$NM(v, i, k) = \sum_{m=0}^{v-i} (-1)^m \binom{2^{v-i-m}}{k} \binom{v-i}{m} 2^m \cong \frac{1}{k!} (2^k - 2)^{v-i} \quad (C-7)$$

where the approximation is made by replacing

$$\binom{2^{v-i-m}}{k} \text{ by } \frac{(2^{v-i-m})^k}{k!}$$

Now, $B(v, i, k, b) = \binom{c(v, i, k)}{b}$ where $c(v, i, k)$ is the total number of clauses which are permissible in the sense that they give

rise to graphs that have the requisite number of isolated points and matched variables. $c(v, i, k)$ can be estimated by

$$c(v, i, k) \cong 8 \binom{v}{3} \exp\left[-\frac{2^k k(v-i)}{(2^k - 2)8v}\right] \quad (C-8)$$

The details of this approximation, which are somewhat lengthy, are given in W. Chuang's thesis.

Using the above approximations and Sterling's approximation

$$n! \cong \sqrt{2\pi} N^{N+1/2} e^{-N}$$

we find by algebraic manipulation that

$$P_s(v, b) \cong 1 - \exp[-\exp(-c)] \quad (C-9)$$

where $b = 8(\ln 2^v + c)$

and c is a constant.

Thus, only for $b \cong 5.5 v$ does $P_s(v, b)$ take values significantly different from 0 or 1 since for any value of c less than -2, P_s is nearly 1 and for any value of c greater than +4, P_s is nearly 0. Figure C.2 summarizes the relationship between P_s and c . Note that this is not a function of v or b directly. The approximations used in obtaining this analytic form are, however, functions of v as we will see. The approximation turns out to be reasonable for modest values of v (e.g., v in the range 10 to 30) which we are interested in. Note especially the sharp drop from 1 to 0 over the interval $c = -2$ to $c = +4$.

C.3. Analytic Method 2

Because of the approximations made in the previous analysis, we sought a second method to independently verify the approach. We

consider the case now where clauses are selected with replacement. We note that all clauses are equally likely and that a given clause eliminates $1/8$ of the possible truth assignments. Indeed, because of this symmetry, on the average the b^{th} clause in an expression eliminates $1/8$ of the remaining truth assignments which satisfy the first $b-1$ clauses. We say a truth assignment is eliminated if it does not satisfy an expression. Thus, if we define $N_S(v, b)$ as the number of truth assignments satisfying an expression with v variables and b clauses, we have the recurrence relation

$$\begin{aligned}\bar{N}_S(v, b) &= \bar{N}_S(v, b-1) - 2^{v-3} + \frac{2^v - \bar{N}_S(v, b-1)}{2^v} 2^{v-3} \\ &= \frac{7}{8} \bar{N}_S(v, b-1) = \left(\frac{7}{8}\right)^b \bar{N}_S(v, 0)\end{aligned}\quad (C-10)$$

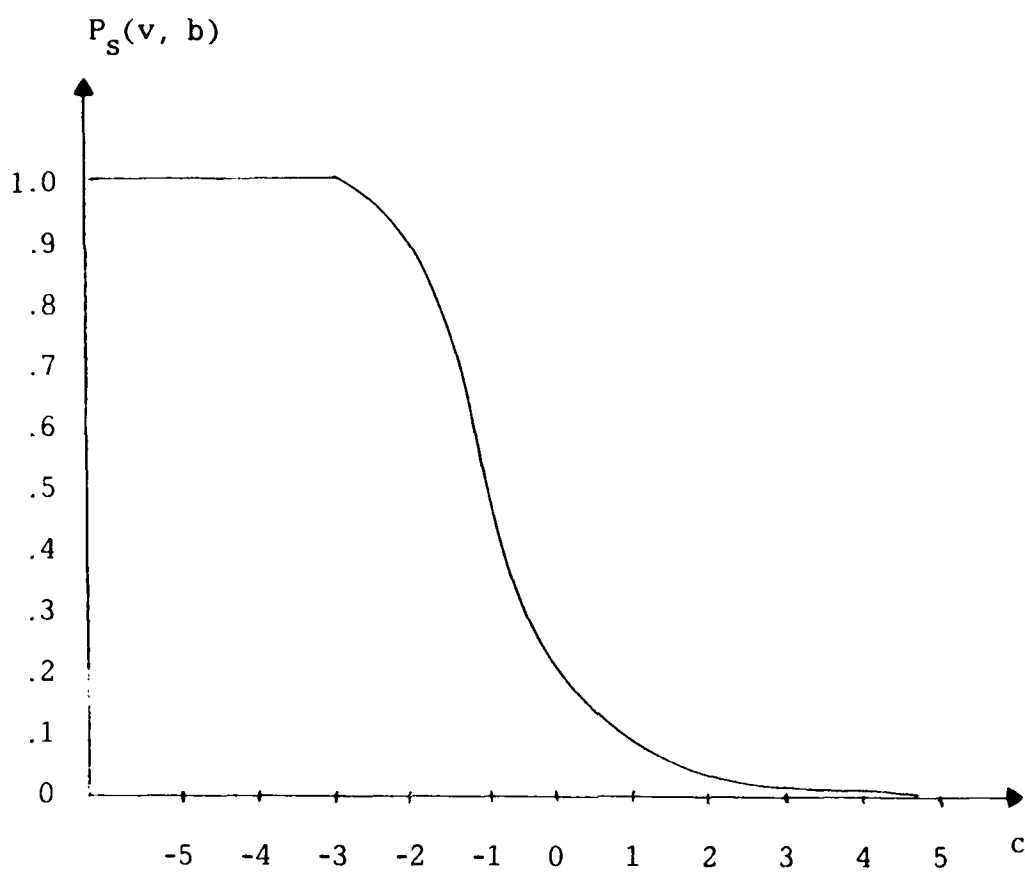
where $\bar{N}_S$ is the expected value of N_S .

Since $\bar{N}_S(v, 0) = 2^v$, i.e., all truth assignments satisfy an expression with no clauses, we have

$$\bar{N}_S(v, b) = \left(\frac{7}{8}\right)^b 2^v \quad (C-11)$$

$P_S(v, b)$ is by definition the probability that $N_S(v, b)$ is greater than zero. Unfortunately, we only have an expression for $\bar{N}_S$, the average value of N_S and not for the value of N_S itself. If $\bar{N}_S(v, b) \gg 1$, then we would expect $P_S(v, b) \cong 1$. If $\bar{N}_S(v, b) < 1$, then we may assume that $N_S(v, b)$ is either 0 or 1. Then $P_S(v, b) = P(N_S(v, b) = 1) = \bar{N}_S(v, b)$. Thus we use the approximation

$$P_S(v, b) = \begin{cases} 1 & , \quad \bar{N}_S(v, b) \geq 1 \\ \bar{N}_S(v, b) & , \quad \bar{N}_S(v, b) \leq 1 \end{cases} \quad (C-12)$$



c	P_s
-3	.999999998
-2	.9994
-1	.934
0	.632
1	.308
2	.127
3	.049
4	.018
5	.007

FIGURE C.2. $P_s(v, b)$ AS A FUNCTION OF c .

This approximation is reasonably good. We first find b such that

$$N_S(v, b) = 1:$$

$$\left(\frac{7}{8}\right)^b 2^v = 1$$

$$b = \frac{v}{\log_2 8 - \log_2 7} \cong 5.2 v \quad (C-13)$$

We then have

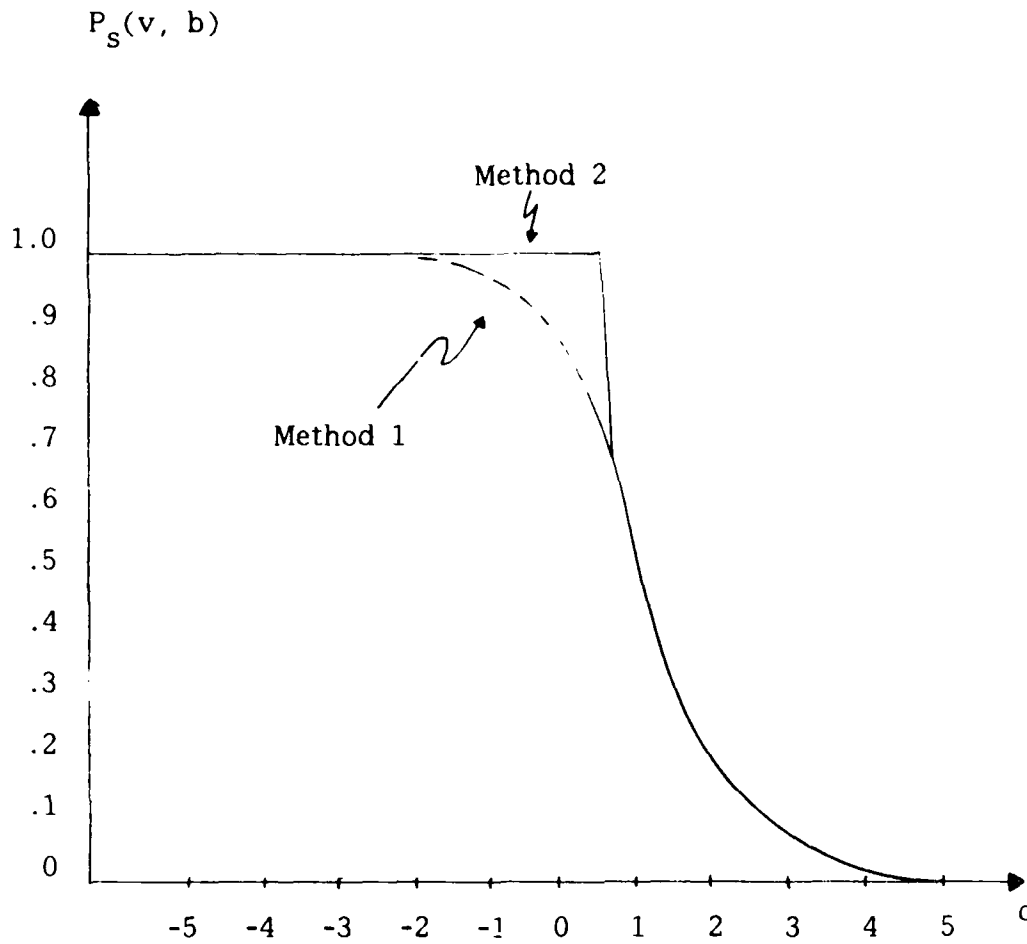
$$P_S(v, b) = \begin{cases} 1 & b \leq 5.2v \\ \left(\frac{7}{8}\right)^{b-5.2v} & b \geq 5.2v \end{cases} \quad (C-14)$$

This function is very similar to the one obtained by the first analytic method and again the region over which $P_S(v, b)$ drops effectively from 1 to 0 is limited again to a small range and is independent of v and b . Figure C.3 illustrates the relationship between P_S and c (where $c = \frac{b - 5.2v}{8}$).

The dotted line of Fig. C.3 shows P_S as found by Method 1 for the same values of c . As can be seen, for $c \geq 1$ the curves are virtually identical. There is, however, a 5% difference between the constants relating v and b in the two approximations. Nevertheless, the two analyses corroborate each other and give rise to the same analytical behavior of P_S ; i.e., v and b are linearly related at the point where P_S drops and P_S decays exponentially.

C.4. Simulation

Finally, in order to compare the accuracy of the two analytic methods, we performed a simulation to measure $P_S(v, b)$ directly. In the simulation, clauses were generated at random with replacement, all clauses being equally likely. Each time a clause was generated, the truth values it eliminated were eliminated. This procedure was con-



c	P_S
0	1.632
1	.344
2	.118
3	.041
4	.014
5	.005

FIGURE C.3. A COMPARISON OF THE FIRST AND SECOND METHODS FOR $P_S(v, b)$

tinued until no truth values remained. For each value of v , the value of b at which the last remaining truth value was eliminated was recorded and became a sample, b_i . An estimate of $P_S(v, b)$ is then formed by

$$P_S(v, b) \cong \frac{(\# \text{ of } b_i > b)}{(\text{total } \# \text{ of } b_i)} \quad (\text{C-15})$$

Figure C.4 gives the results of this simulation for $v = 10, 20$ and 30 . As can be seen, the true value of $P_S(v, b)$ lies between the N values predicted by the two analytic methods and is somewhat closer to that predicted by the second method. The important property of rapid decay from 1 to 0 is borne out.

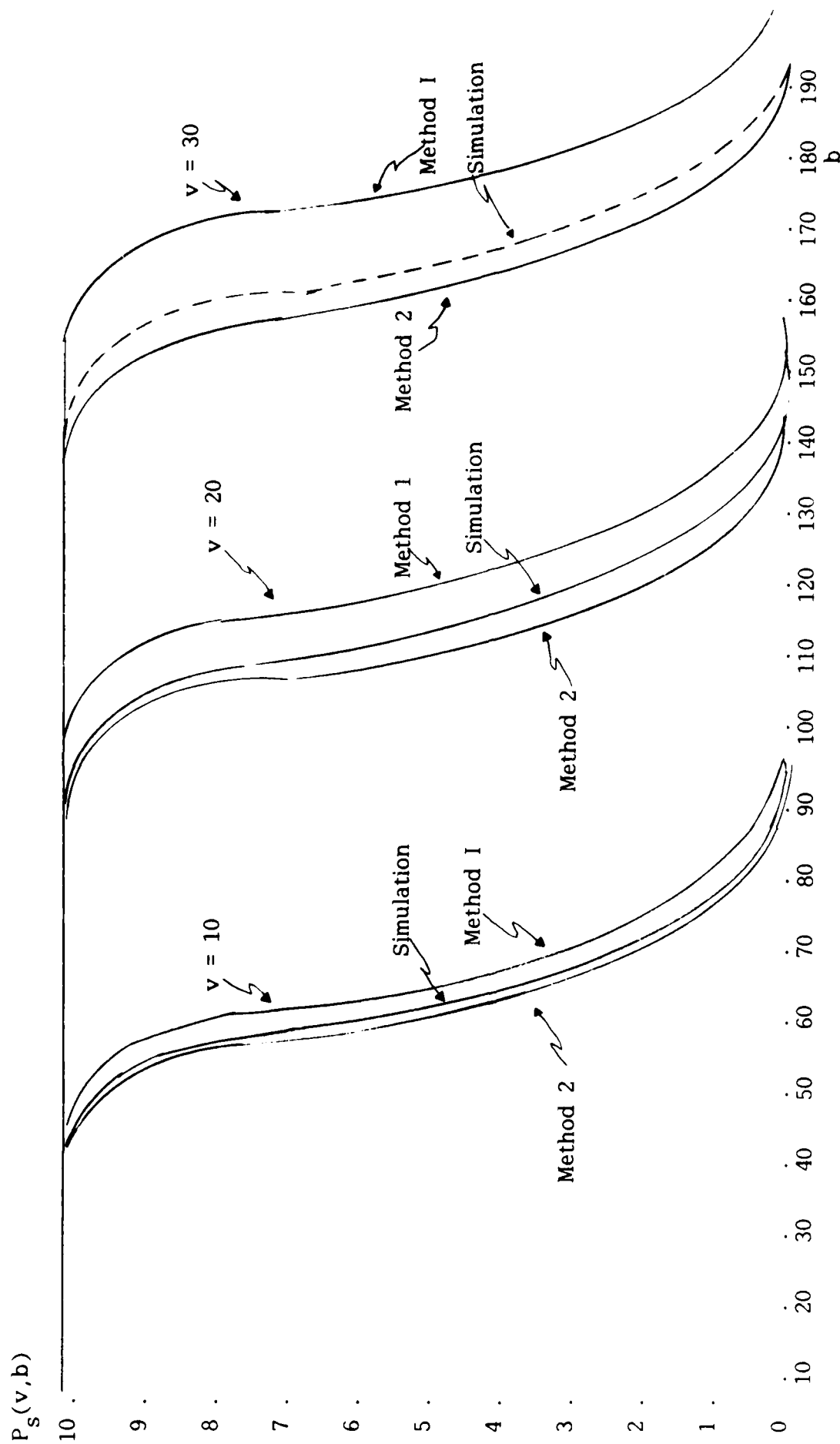


FIGURE C.4 COMPARISON OF SIMULATION AND TWO ANALYTICAL METHODS

D. Network Design

D.1 Second-Order Greedy Algorithms for Centralized Tele-
processing Network Design

Kershenbaum, Boorstyn, and Oppenheim

IEEE Transactions on Communications, October 1980

Second-Order Greedy Algorithms for Centralized Teleprocessing Network Design

A. KERSHENBAUM, MEMBER, IEEE, R. BOORSTYN, MEMBER, IEEE, AND R. OPPENHEIM

Abstract—We consider the problem of designing a centralized telecommunication network comprised of multipoint lines given a set of terminal locations, traffic requirements, and a common central site. The optimal solution to this problem is a capacitated minimal spanning tree. We develop a class of heuristic algorithms for the solution of this problem by imbedding existing heuristics, referred to as first-order greedy algorithms, inside a loop where small, carefully chosen sets of arcs are alternately forced in and out of the solution. The resultant procedure is shown to be superior to existing techniques, producing solutions typically 2 percent better, while requiring only a modest amount of additional computer time.

INTRODUCTION

The problem considered is that of finding an optimal (minimum cost) design for a centralized telecommunication network given a set of terminal locations, traffic magnitudes between these locations, and single common source or destination (central site). In order to retain simplicity and low cost in the terminal hardware, such networks are configured as trees comprised of communication facilities of a single capacity. Thus, the optimal solution to this problem is a capacitated minimal spanning tree (CMST), i.e., a tree of minimum total length satisfying a constraint or set of constraints that limit the total traffic and/or number of nodes in any subtree rooted at the central site. In centralized networks, such subtrees are called multipoint lines.

More formally, we are given a set of N locations (nodes), in addition to the central site; a symmetric distance measure $D = \{d_{ij} | i, j = 0, 1, \dots, N\}$, giving the cost between any pair of locations; and a constraint M on the total number of nodes or traffic in a multipoint line. We seek a spanning tree T rooted at the center, satisfying the constraint, and of minimum total length.

This formulation is quite general. All we require of the cost function is that the cost of a link d_{ij} not depend upon what other links are present in the solution. The constraint is, likewise, quite general. One can, in fact, have more than one constraint, and the constraints may be on total traffic, number of nodes, nodal degree, number of links in cascade, or any other quantity associated with the design. We require only that if some subtree S does not satisfy the constraints, then no other subtree S' , containing S , satisfies the constraints. We also assume that a star solution, i.e., all nodes connected directly to the center, is feasible. This can always be satisfied by splitting a location into two or more nodes.

Several heuristics and optimal techniques have been devel-

oped for the solution of this problem. The optimal techniques [1]–[3] are branch and bound procedures which, in general, have running times which are exponential in the number of nodes. While they are, thus, not practical as design procedures, results obtained using them (in particular, a recently developed procedure [7]) have led to some insight into the refinement of heuristic procedures. The procedure described below is an outgrowth of this work.

Currently, the most widely used procedures for the solution of the CMST problem are heuristics [3], [4], [8], [10] which produce solutions within 50 percent of the optimum (in cases where the optimum is known) and which have running times which are a low-order polynomial, generally between quadratic and cubic, in the number of nodes. Karnaugh [6] has developed a family of second-order greedy algorithms (SOGA's) which iterate the above heuristics (which he refers to as First-Order Greedy Algorithms (FOGA's)), and have longer running times, but produce results generally 2–3 percent better than the above heuristics. The procedures described below are variants of the general SOGA procedures described by Karnaugh, produce results of comparable quality, and are considerably faster than his. In many cases their running times are competitive with the simpler procedures (FOGA's) previously used.

PROCEDURAL DESCRIPTION

The most often used heuristic solutions to the CMST problem share the following properties: 1) their running time is a polynomial in the number of nodes; 2) in the absence of constraints, they will yield a minimum spanning tree (MST); and 3) the quality of the solution (i.e., the amount by which it differs from the optimum) is not controllable and, except very loosely, is not known.

The basic heuristics used to solve CMST problems can be divided into two categories—primal procedures which seek to improve a feasible starting solution or partial solution, and dual procedures which seek to make a low cost (infeasible) starting solution feasible. We concentrate on primal procedures, having found them to be generally more flexible. It has been shown [8] that most such procedures fall within the framework of the following scheme.

1) Start with each node on a separate multipoint line directly connected to the center. Associate a weight w_i with each node i by applying a given rule (w -rule). For each potential link L_{ij} interconnecting a pair of nodes i and j , define a trade-off function t_{ij} as $d_{ij} - w_i$.

2) Consider the (not previously considered) L_{ij} for which t_{ij} is minimum. If nodes i and j are in separate multipoint lines and the subtree formed by merging these lines does not violate any constraint, then add L_{ij} to the network, replacing L_{i0} or L_{j0} , whichever is more costly. If not, reject L_{ij} .

3) Update the w_i and t_{ij} and return to Step 2) until no further gain can be obtained. Usually, $w_i = w_j$ for nodes i and j in the same multipoint line.

The implementation of such a procedure has been considered in detail [9]. A careful implementation is shown to be of order $N^2 \log_2 N$ where N is the number of locations and if one wishes to consider all branches. It is of order $N \log_2 N$ if one only examines the K nearest neighbors of each node and if the w -rule itself is not too computationally complex. In practice, the most widely used w -rule is to set the weight of each node in a subtree to be the minimum distance between any node in the subtree and the center. This is known as the Esau-Williams

Paper approved by the Editor for Computer Communication of the IEEE Communications Society for publication after presentation at the 1st International Symposium on Policy Analysis & Information Systems, Durham, NC, June 1979. Manuscript received May 10, 1978; revised May 8, 1980. This work was supported in part by ARPA Contract DAH0-15-73-CO-135, in part by NSF Grant NSF ENG 7908120, and in part by U.S. Army CORADCOM.

A. Kershenbaum and R. R. Boorstyn are with the Department of Electrical Engineering and Computer Science, Polytechnic Institute of New York, Brooklyn, NY 11201.

R. Oppenheim is with the Graduate School of Management, Rutgers University, Newark, NJ 07102.

algorithm. Karnaugh [5] described an alternate implementation of the Esau-Williams procedure which has a different computational complexity, generally between quadratic and cubic, but which produces identical results. While the SOGA's described below can work with any imbedded FOGA, for the sake of comparison with previous work, we performed experiments with an imbedded Esau-Williams algorithm.

In order to implement a SOGA, one must decide which arcs one will attempt to force in or out of the solution. Karnaugh suggested two possibilities which are briefly described below. The reader is referred to [6] for a complete description.

1) **Inhibit**—At each stage in its execution, the FOGA brings in one arc connecting two previously unconnected sets of nodes. The Inhibit loop successively prevents each of the mergers which took place in the previous iteration from taking place. Thus, one iteration of the Inhibit loop involves up to N (where N is the number of terminals) iterations of the FOGA, each of which prevents a single pair of node clusters from merging. At the end of each iteration of the Inhibit loop, the best of the generated solutions (and its associated cluster inhibitions) is kept and used in place of the original (uninhibited) FOGA solution, and the Inhibit loop is repeated until no further progress is made.

2) **Join**—For each node a_i , find its nearest neighbor b_i , and nearest neighbor among those nodes which are closer to the center than a_i, c_i . Successively, one arc at a time, force arcs (a_i, b_i) and (a_i, c_i) into the solution if they are not already present. One Join loop consists of a sequence of executions of the FOGA, each with a single forced arc. The Join loop is iterated, starting from the best solution obtained during the previous loop, until no further progress is made. The number of FOGA iterations in a Join loop is bounded from above by $2N - 1$ but is, on the average, somewhat less than N , the exact number being a function of the particular problem.

Thus, both the Inhibit and Join procedures have running times of order CN times the running time of the embedded FOGA, the Join procedure being somewhat faster. The factor C is the number of iterations required for convergence, i.e., to reach the point of no further progress. It would be possible to greatly improve the running time of the SOGA if one could restrict one's attention to a small (i.e., $\ll N$) subset of forced inclusions or exclusions. Also, Karnaugh mentioned that the Join procedure, while somewhat faster than Inhibit, is weaker because it restricts itself to local transformations involving nearest neighbors.

The procedure described in [7] is a branch-and-bound algorithm for obtaining optimal solutions to CMST problems. While it does not converge to an optimal solution quickly enough to be useful as a design procedure, it was sufficiently effective to be useful in obtaining optimal solutions to a number of problems of sufficient size and constraint tightness to be able to make observations about characteristics of optimal solutions. This led to the characterization of arc subsets which are small and at the same time reasonably effective for use as candidates in a SOGA. In particular, it was found that arcs present in optimal solutions but not present in heuristically generated solutions were almost always MST arcs. While it is not necessarily so that all arcs present in the optimal solution and not in the Esau-Williams solution are MST arcs, in all the experiments run and checked to this end, invariably at least one arc in the optimum, but not in the Esau-Williams solution, was an MST arc.

This is not to say that it is conjectured that one must always be able to find an optimum solution which contains an

MST arc not part of the Esau-Williams solution (when the Esau-Williams solution itself is not optimal). It merely strengthens the point that MST arcs are reasonable arcs to examine when seeking to improve a heuristic solution.

Even without this evidence, however, it is intuitively appealing to consider such arcs. Most primal heuristics share the property that they proceed by connecting nodes to their nearest feasible neighbors [8]. The algorithms differ only in the order in which they consider the nodes. Thus, the solutions differ only in that the distance to a node's nearest feasible neighbor may differ depending upon when the node is considered during the execution of the algorithm. If such a heuristic fails to find the optimum, it will be because the algorithm erred by waiting too long before consideration of some critical node (or nodes), and hence missed being able to include some desirable arc (or arcs) in the solution. It is quite likely that one or more of these arcs will be MST arcs, indeed, as mentioned above, this was always the case with the experiments carried out.

Thus, a heuristic is proposed which attempts to improve solutions generated by some primal heuristic by forcing the inclusion of one or more MST arcs which the primal heuristic left out. This gives rise to the following procedure.

Step 1: Generate an MST of arcs $T_m = \{a_j | j = 1, \dots, N-1\}$. Generate a feasible tree of arcs $T_f = \{b_j | j = 1, \dots, N-1\}$. Find $S = T_m - (T_m \cap T_f)$. (For problems of interest, S is nonempty.)

Step 2: For each subset $S_1 \subset S$, set $S_2 = S - S_1$. Remove all $a_j \in S_2$ from the network. Start a solution by including all $a_j \in S_1$. Apply the primal heuristic to complete the generation of a solution.

It should be noted that an alternative procedure would be to include a_j in S_1 , but not to exclude a_j in S_2 . It was felt, however, that if elements in S_2 were retained as candidates for inclusion, duplicate solutions would be likely to result. The exclusion of elements in S_2 guarantees unique solutions, and hence should increase the likelihood of generating improvements.

COMPUTATIONAL EXPERIENCE

In order to assess the effectiveness and efficiency of the procedural scheme described in the previous section, a series of experiments was run. Nodes were generated with random locations over a unit square and unit traffic; problems with $N = 40, 60, 90$, and 120 nodes were generated, and constraints on the number of nodes in subtrees varying between $m = N/20$ and $m = N/4$ were examined.

Alternatively, a constraint on traffic could have been used, but we did not feel that significantly different results would have been obtained. Our first objective was to examine the running time of our implementation of the Esau-Williams Algorithm within the context of the Unified Algorithm [8].

Karnaugh [6] described an alternate implementation of the Esau-Williams procedure, and reports on its running time on an IBM 370/158. Our experiments were done on a DEC KL 20/50 which is somewhat slower than a 370/158. In order to establish a basis for the run time comparisons with Karnaugh's procedures, shown in Fig. 2 and Table I, we first compared our implementation of the FOGA (cf. [8]) with his. We found that both procedures had comparable run times and computational complexities (between quadratic and cubic) for the first FOGA iteration (i.e., for obtaining an initial solution), but that our implementation was of a lower order of complexity for later iterations. The difference in running times

for later iterations was roughly a factor of $5/2$ for $N = 120$. Since most of the execution time of the overall procedure is in iteration of the FOGA, we conclude that this coincides with the differences in running times between the two versions of Inhibit in Fig. 2.

Next, by running a sequence of 120 problems for N between 10 and 100, we examined the size of the subsets S generated, i.e., the average number of MST links not present in the FOGA solution. Most primal heuristics generate subtrees which are MST's on the set of nodes they contain along with the center. Most of the arcs in these subtrees (with the exception of the arc directly connected to the center) will be MST arcs. Thus, one might expect that the cardinality of S , $|S| \approx N/m$, and a vast improvement can be made over a blind branch exchange or SOGA procedure. In practice, we found that $|S|$ ranged between N/m and $N \log N/m$.

There are, in fact, $2^{|S|}$ subsets of S . Thus, for moderately large tightly constrained problems, S could grow large enough to make evaluating all subsets impractical. Furthermore, it is reasonable to assume that not all arcs in S interact with one another i.e., improvements in separate parts of the network can be found and justified independently of one another. Thus, the heuristic was modified to only consider subsets $S_1 \subset S$ such that $|S_1| \leq K$ for some given K . The best subset S_1^* is found and permanently forced into the solution. We then set $S = S - S_1^*$ and repeat the procedure until no further improvement can be made. In practice, $K = 2$ worked well. Experiments were run with larger values of K ; only in isolated cases was any improvement over $K = 2$ obtained. (Even $K = 1$ worked well in many cases.) Thus, it was decided in all the remaining experiments to use the extended procedure and restrict the examination to subsets of cardinality less than or equal to two.

Using this modification of the new heuristic, it was observed that forcing arcs between nodes which are close to the center seemed to have the greatest effect on the value of the solution. This was probably a consequence of the fact that the Esau-Williams algorithm starts with nodes from the center, and hence, dealing with nodes near the center first radically changes the solution value.

In particular, it was found that for small networks ($N \leq 20$), there was little difference (usually ≤ 1 percent) in performance between the new heuristic and the Esau-Williams solution. This is almost certainly because both procedures were generating near-optimal solutions. However, for larger networks, particularly for tightly constrained problems, the new heuristic performed noticeably better with improvements averaging about 1.5 percent. Fig. 1 shows the improvement relative to the Esau-Williams algorithm. As can be seen, the improvement increases with the problem size. Results were even more encouraging for large networks with the central site in the corner. Such networks may be viewed as one-fourth of a network of $4N$ nodes with the central site in the center. For such problems, improvements averaging 4 percent and as high as 8 percent were observed. Thus, the procedure appears to be of value for many realistically sized problems with tight constraints.

A straightforward implementation of the Esau-Williams procedure has a computational complexity of order $N^2 \log_2 N$. A more careful implementation [9] can reduce the complexity to order $N \log_2 N$. As discussed previously, $|S|$ was found to range between N/m and $N \log N/m$. Since we examine subsets of cardinality at most 2, $|S|$ subsets are examined on each iteration. At worst, $|S|/2$ successive subsets of cardinality 2

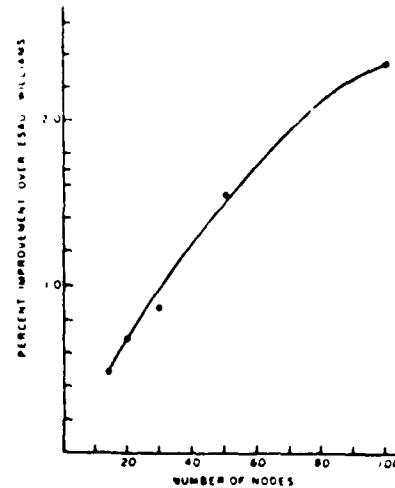


Fig. 1. Improvements obtained with new heuristic.

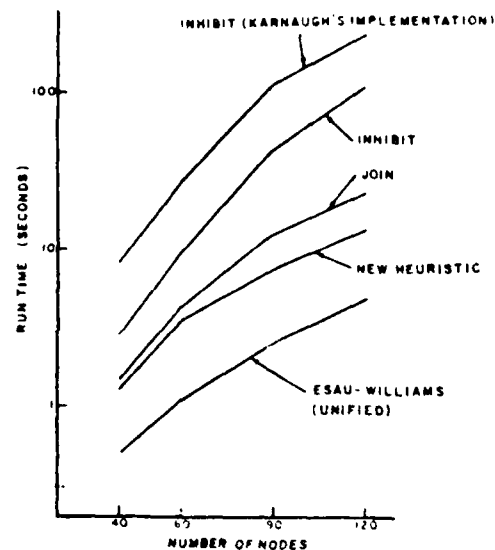


Fig. 2. Comparison of run times.

could be introduced, and thus, the procedure could iterate at most $|S|/2$ times. In practice, the number of iterations grows more slowly than $|S|$.

Thus, a careful implementation of the procedure has a complexity of $|S|^3 N \log N$.

Finally, a version of the Inhibit and Join procedures (as described in [6], but using the Unified Algorithm in place of the alternate implementation of Esau-Williams) was coded and run on another set of problems, and the heuristic described in this paper was then run on the same set of problems. Thus, we were able to directly compare the quality of the obtained solutions, as well as the running times of the new heuristic with Inhibit and Join. The running times are summarized in Fig. 2 and Tables I and II. As can be seen the new heuristic is much faster than Inhibit and faster than Join. Indeed, it is only two-three times slower than the Unified Algorithm.

The quality of the solutions obtained varied. Overall, the results were consistent with the experiment described above, although a drop in effectiveness is noted for the 120-node problems. We do not consider this latter phenomenon significant since there was a large variation in individual runs, and the first experiment, which was based on a much larger num-

TABLE I
A COMPARISON OF RUN TIME (IN SECONDS)

Algorithm	Number of Nodes			
	40	60	80	120
Inhibit (Karnaugh's Implementation)	9.0	26.7	118	240
Inhibit	2.8	9.5	46	113
Join	1.4	4.3	12	24
New Heuristic	1.2	3.6	8	13
Esau-Williams (unified)	0.5	1.1	2.6	4.7

TABLE II
PERCENT IMPROVEMENTS IN PERFORMANCE OVER
ESAU-WILLIAMS

Algorithm	Number of Nodes				
	40	60	80	120	Average
Inhibit	2.2	2.0	3.6	2.6	2.6
New Heuristic	2.4	1.5	2.6	0.8	1.9
Join	2.1	1.5	2.0	0.9	1.6

ber of problems, exhibited no similar behavior. In some cases, the new heuristic outperformed Inhibit, in others, Inhibit performed better. On the whole, the quality of the solutions obtained with Inhibit was 2.6 percent better than those obtained using the Esau-Williams procedure, while those obtained using the new heuristic were 1.9 percent better than Esau-Williams. Both Inhibit and the new heuristic outperformed Join fairly consistently, Join averaged a 1.6 percent improvement over Esau-Williams.

CONCLUSION

We found the new heuristic to be of definite value as it obtained solutions roughly 2 percent better than the Esau-Williams procedure without greatly increasing the running time. In comparison with Karnaugh's SOGA's, the new heuristic is much faster and obtains solutions somewhat better than Join and somewhat worse than Inhibit. We conclude that the new heuristic is useful as a practical design procedure, even when imbedded in a larger procedure which solves more global problems.

REFERENCES

- [1] K. M. Chandy and R. A. Russell, "The design of multipoint linkages in a teleprocessing tree network," *IEEE Trans. Comput.*, vol. C-21, pp. 1062-1066, Oct. 1972.
- [2] K. M. Chandy and J. Lee, "The capacitated minimum spanning tree," *Networks*, vol. 3, no. 2, pp. 173-182, 1973.
- [3] R. Elias and M. L. Ferguson, "Topological design of multipoint teleprocessing networks," *IEEE Trans. Commun.*, vol. COM-22, pp. 178-179, Nov. 1974.
- [4] R. Elias and K. C. Williams, "Teleprocessing network design," Part 2, *IBM Syst. J.*, vol. 8, no. 3, pp. 442-443, 1969.
- [5] R. Frank, L. E. Frisch, W. Chu, and K. Van Arman, "Optimal

design of centralized computer networks," *Networks*, vol. 1, pp. 43-57, 1971.

- [6] M. Karnaugh, "A new class of algorithms for topology optimization," *IEEE Trans. Commun.*, vol. COM-24, pp. 546-553, May 1976.
- [7] A. Kershenbaum and R. Horstyn, "Centralized teleprocessing network design," in *Proc. IEEE Telecomm. Conf.*, Dec. 1974, pp. 27-31, 27-34.
- [8] A. Kershenbaum and W. Chu, "A unified algorithm for design of multidrop teleprocessing networks," *IEEE Trans. Commun.*, vol. COM-22, pp. 1762-1772, Nov. 1974.
- [9] A. Kershenbaum, "Computing capacitated minimal spanning trees efficiently," *Networks*, vol. 4, no. 4, 1974.
- [10] N. V. Reinfeld and W. R. Vogel, *Mathematical Programming*, Englewood Cliffs, NJ: Prentice Hall, 1958.

D.2 Centralized Teleprocessing Network Design

Kershenbaum and Boorstyn

To be published in Networks Journal

Centralized Teleprocessing Network Design

By

Aaron Kershenbaum

and

Robert R. Boorstyn
Polytechnic Institute of New York

Abstract

The problem considered is that of finding an optimal (minimum cost) design for a centralized processing network given a set of locations, traffic magnitudes between these locations, and a single common source or destination. Several heuristics, which are efficient (in terms of their execution time and memory requirements on a digital computer) and which produce seemingly good results, have already been developed and are currently accepted techniques. Some work has also been done on finding optimal solutions to this problem both as a design tool and as a means of verifying the effectiveness of proposed heuristics. We focus in this latter area. Currently known techniques for the optimal solution of this problem via integer programming have fallen short of the desired objectives as they require too much memory and running time to be able to treat problems of realistic size and complexity. We develop an improved technique which is capable of handling more realistic problems.

This work was supported in part by the U.S. Army CORADCOM, Contract No. DAAK 80-80-K-0579, and by the National Science Foundation, Grant No. ENG-7908120

1. INTRODUCTION AND PROBLEM STATEMENT

The problem considered is that of finding an optimal (minimum cost) design for a centralized telecommunication network given a set of locations, traffic magnitudes between these locations, and a single common source or destination. The vast majority of telecommunication networks currently in existence are of this type. Thus, this problem has been much studied (2,3,4,5,8,9,12,16,24,28,31,32).

Several heuristics, which are efficient (in terms of their execution time and memory requirements on a digital computer) and which produce seemingly good results, have already been developed and are currently accepted techniques. Some work has also been done on finding optimal solutions to this problem as a means of verifying the effectiveness of proposed heuristics. Currently known techniques for the optimal solution to this problem via integer programming have fallen short of the desired objective as they require too much memory and running time to be able to treat problems of realistic size and complexity. We develop an improved technique which is capable of handling problems of realistic size.

More formally, the problem considered here is that of finding a minimum spanning tree subject to one or more constraints which in general are equivalent to demanding that the sum of the traffic associated with the nodes in any subtree must not exceed some predetermined maximum.

A minimum spanning tree is a loop-free collection of arcs joining a set of nodes such that the sum of the lengths of the arcs is minimal. In the case of a communication network, these collections of arcs are called multidrop lines.

It should be noted that this constraint form is quite general and encompasses many real-world constraints which arise in the design of centralized telecommunications networks. Thus, for example, in addition to treating the obvious constraint imposed by line capacity, it is possible to treat a restriction on the number of terminals on a multidrop line by associating a uniform traffic with each terminal. Also, the length (cost) functions which can be treated are quite general. Any function which is not a function of the tree chosen is permissible.

Formally, we seek to solve the following problem:

Given

1. A vertex (node) set $V = \{v_i | i=0,1,\dots,n\}$ representing the terminal locations in the network. Node v_0 is a distinguished node which we will refer to as the center.
2. A symmetric function giving the length (cost) d_{ij} of an arc between any pair of locations.
3. A constraint, m , on the number of nodes which may share a multidrop line. This constraint can be generalized to allow a weight or traffic, c_i , to be associated with each node and to require that the sum of the weights associated with the nodes on any multidrop line not exceed m .

We define the set of nodes in the j^{th} multidrop line to be V_j and the multidrop line itself to be a minimal spanning tree T_{V_j} on $V_j \cup \{v_0\}$. Thus, the constraint can be stated in terms of the cardinality of V_j as $|V_j| < m \forall j$. In the more general form, the constraint would be $\sum_{v_i \in V_j} c_i < m \forall j$. We wish to find a tree, T_V^* of minimum total length satisfying the constraint in 3 above. That is, we wish to

minimize $\sum_{i=1}^N d_{ip_i}$ subject to 3, where v_{p_i} is the immediate predecessor of v_i , i.e., the node closest to v_i on the path between v_i and v_0 in T , and T is any spanning tree. We consider exact (optimal) solutions to this problem. The primary motivation for the work is to develop an exact algorithm capable of permitting study of the performance of heuristics on a broader class of problems than was previously studied, to gain insight into the performance of both exact and heuristic procedures and, in particular, to pinpoint where and why they fail.

II. OUTLINE OF A NEW OPTIMAL SOLUTION TECHNIQUE

There currently exist several techniques which will yield optimal solutions to the CMST problem. These techniques can be divided into two classes - branch exchange methods (as proposed by Lin (22) and Frank (9)) and branch and bound methods (10,22). We concentrate on the latter class of techniques.

The specific application of branch and bound techniques to the solution of the CMST problem was proposed by Chandy and Russell (3) and was subsequently refined (2) so that it could treat somewhat more meaningful problems. Subsequently, Elias and Ferguson (4) proposed further refinements and thereby expanded the range of applicability of the technique. Gavish (34) recently developed a bound using Lagrangean relaxation.

The basic technique is, as has already been mentioned, a branch and bound algorithm. The original problem considered has all branches in the category "permissible," i.e., any branch may or may not be part of the final solution. Subproblems are generated by selecting a permissible branch and making it "prohibited" in one subproblem or "required" in another.

The relaxation used is simply to generate a modified MST by including all "required" branches, excluding all "prohibited" branches, and forming the tree of minimum total length by connecting (as yet unconnected) nodes using remaining ("permissible") branches.

Clearly, a solution obtained in this manner is a lower bound on the value of a feasible solution to the subproblem as it is the tree of minimum length. Note also that in the case where all arcs are specified (prohibited or required), the lower bound and solution are

identical and the subproblem fathoms. In general, the subproblem fathoms when

1. No feasible solution exists to the subproblem. This occurs, when the required branches form a loop, when the required branches create a subtree violating the constraints, or when the prohibited branches disconnect the network. Other criteria exist but are difficult to test for.
2. The lower bound equals or exceeds the value of the best solution found thus far.
3. The lower bound solution is feasible.

When all subproblems have fathomed, the current best solution is the global optimum.

A number of observations have been made, which can be used to accelerate a basic branch and bound technique. One of these, which is used in the sequel, is given in Theorem 1 below:

Theorem 1: (3)

If branches (v_0, v_{j_1}) , (v_0, v_{j_2}) , ..., (v_0, v_{j_K}) , are part of some MST, T , on V then there exists a CMST including these edges.

Corollary:

If arcs (v_0, v_{j_1}) , (v_0, v_{j_2}) , ..., (v_0, v_{j_K}) are present in the modified MST produced in any subproblem, then (if any CMST's exist in the subproblem) there exists a CMST on the subproblem containing these arcs.

The preceding theorem and corollary allow one to avoid considering subproblems with such arcs prohibited. The techniques developed in the sequel make explicit use of both observations as well as others made in the references cited.

The inherent problem with the existing procedures lies in the relaxation method used. At each step, the problem is relaxed to a modified MST. Unfortunately, this bound is often too loose to eliminate a sufficiently large percentage of the subproblems to make the procedure practical. This is particularly evident when the constraints are tight; it is for such problems that the relaxation is loosest. Unfortunately, it is also for that class of subproblems that the known heuristics display the widest variation in the quality of solutions.

Note that any optimal solution to the CMST problem has the property that all subtrees are MST's on the set of nodes contained in the subtree and the center. Thus, it suffices to find the optimal partition of the nodes into subtrees. The technique which is developed in the following sections will thus generate partitions of the nodes.

The technique works within the framework of branch and bound algorithms, as did the techniques referred to above. We develop two algorithms, one based on generating subproblems by restricting nodes, and the other based on generating subproblems by restricting arcs. These techniques differ from previous ones in that the relaxation used here is tighter and thus, a smaller number of subproblems need be examined.

We begin by restricting the problem slightly. We seek a CMST subject to the constraint that the number of nodes (rather than the sum of the weights of the nodes) in any subtree not exceed a pre-specified maximum. Since our primary intent here is to study the performance of CMST algorithms, this modification would not, in general, have a significant effect. Indeed, if one preferred, a node

of weight K could be replaced by K nodes of weight 1, providing one is willing to allow the original node of weight K to be split among more than one subtree.

To find a partition of the nodes $V = \{v_i \mid i = 1, 2, \dots, n\}$ into subtrees, we begin by making n copies of each node corresponding to the possibilities of the node being in any of n possible subtrees. Thus, v_{ij} corresponds to node v_i being in subtree j .

The problem of obtaining an optimal partition of nodes into subtrees can be thought of as one of selecting an optimal subset from the set $E = \{v_{ij} \mid i=1, 2, \dots, n; j=1, 2, \dots, n\}$. Feasible subsets of E , i.e., those corresponding to partitions satisfying the capacity constraint, will contain 1 v_{ij} for each i and at most m v_{ij} 's for each j . If we associate a weight, w_{ij} , with each v_{ij} , the optimal subset of E (hence the optimal partition of V) is defined as the feasible subset of minimum total weight.

An efficient algorithm (see 14, 21, 35) exists for the solution of the problem of finding the optimal subset of E given the values of w_{ij} . The algorithm, which can be thought of as a matroid intersection [1,17,19,20,29] algorithm or alternatively as a series of shortest path problems in appropriately defined graphs, has a worst case running time of order n^3 and in practice has a running time closer to order n^2 (see 35). Unfortunately, the set of weights, w_{ij} , which correspond directly to the "cost" (contribution to the overall length of the CMST) of v_i in subtree j can be specified only when the problem solution is already known. We can, however, define a set of weights, w_{ij} , which have the property that the optimal partition found using these weights will have a value (sum of weights) which is a

lower bound on the length of the optimal CMST. Thus, we can relax the CMST problem to the problem of finding an optimal partition. This, together with generating subproblems by successively restricting either nodes or arcs, gives rise to an optimum CMST algorithm within the branch and bound context.

An appropriate set of weights, w_{ij} , can be defined as follows. Suppose we are given, for each subtree j , the set $\hat{V}_j$ of nodes permitted in the subtree; a procedure for obtaining the $\hat{V}_j$ will be given below. One can then find T_j , the minimum spanning tree on the nodes in $\hat{V}_j \cup \{v_0\}$. Let d_{ij} be the length of the arc connecting v_i to its predecessor in T_j (i.e., d_{ij} is the length of the last arc in the path from v_0 to v_i in T_j). The following theorem, which is proven in (35), allows us to obtain appropriate w_{ij} :

Theorem 2: The weight of the optimal partition using $w_{ij} = d_{ij}$ is a lower bound on the length of the CMST for the same V and M .

Furthermore, it is proven in (35) that other similarly defined w_{ij} 's also preserve this lower bound. In particular, suppose T_j contains a path $(v_0, \dots, v_p, v_q, \dots, v_x, \dots, v_y, \dots)$ as shown in Figure 1. Let S be the set of nodes $\{v_q, \dots, v_k, \dots, v_s\}$ and let w_k be the largest weight of any node in S . Suppose $w_{pj} > w_{kj}$. Define $\Delta = w_{pj} - w_{kj}$. Then the following theorem holds:

Theorem 3: If a set of weights $w_{ij} = d_{ij}$ is modified by transferring weight Δ from w_{pj} to w_{sj} where Δ , v_p and v_s are defined as above, the weight of the optimal partition is still a lower bound on the length of the CMST for the same V and m .

Theorem 3 allows us to transfer weight from a node to its successors in T_j in order to guarantee that the lower bound obtained

from the partitioning problem is at least as tight as the bound obtained using an MST, as is done in (2), (3), and (4). A proof that this can always be done is given in (35). As an example of how this works, consider the network shown in Figure 2a. The MST for this network and the node weights corresponding to it are shown in Figure 2b. These weights correspond to the bound obtained using an MST. Suppose, however, that we restrict v_3 from being part of a given subtree j . The weights shown in Figure 2c would then be obtained if we simply set $w_{ij} = d_{ij}$. Note in particular that w_{2j} has been reduced from 5 to 1. This reduction in w_{2j} could result in a loosening of the lower bound. Theorem 3 allows us to transfer up to $\Delta = w_{1j} - w_{2j}$, i.e., 7 units of weight, from w_{1j} to w_{2j} and obtain the weights shown in Figure 2d. Note that the w_{ij} in Figure 2d are at least as great as the w_{ij} in Figure 2b. Thus, the lower bound obtained using the w_{ij} in Figure 2d will be at least as tight as that obtained using an MST. In fact, the bound so obtained is significantly tighter, as is shown by the computational experience given in Section V.

We now turn to the question of the branching rule within the branch and bound procedure. Little was said by Chandy and Russell, Chandy and Lo, and Elias and Ferguson on the order in which subproblems are considered in the branch and bound procedure. Classically, two approaches are available. The first is to always consider the subproblem with the least lower bound. Alternatively, one can use depth first search, where one always solves most recently generated subproblems before returning to older subproblems. There are advantages and disadvantages to both approaches.

The first approach allows one to proceed without any good feasible solutions to guide the process. The assumption is that subproblems with the lowest lower bounds will give rise to the best feasible solutions. Hence, one prefers to explore these subproblems first in the hope that they will give rise to low cost feasible solutions which will eliminate other subproblems (with higher lower bounds) from consideration. Also, by examining subproblems in this order, one is continually narrowing the range between the upper and lower bounds, and hence, has the option of terminating the algorithm when the interval shrinks to some prespecified width.

There are, however, two major drawbacks to this approach. First, one must keep (a potentially large number of) subproblems around in order to select the next one. Thus, the storage required for the procedure is potentially exponential. In practice, it was storage, not running time, which was the active constraint on problem size in previously developed techniques. One could temporarily store subproblems in secondary storage, but this would complicate and slow down the procedure.

Second, by considering problems in ascending order of lower bound, one will, in general, be sequentially considering dissimilar subproblems. Thus, one cannot easily take advantage of information obtained in the solution of one problem for the solution of another. For example, in the Elias and Ferguson technique, the similarity between modified MST's for related subproblems cannot be easily exploited if this first procedural outline is adopted.

Using depth first search overcomes both of these objections. Indeed, a great deal of simplification is obtainable both in the genera-

tion of subproblems and in obtaining solutions owing to the similarity of successively considered subproblems.

The maximum number of subproblems which need be kept around at any time is bounded by the number of nested specifications it is possible to make. Thus, if one is restricting nodes, the bound is n ; if one is including or excluding arcs, the bound is $\binom{n}{2}$. This essentially eliminates storage as an active constraint on the size of the problem which can be considered.

A further reason for using depth first search is that the major reasons one would ordinarily choose the first procedure are not present here. Any of the existing heuristics can be used to quickly generate a good upper bound. Furthermore, the procedures developed in the sequel lend themselves to generating feasible solutions for all subproblems. Thus, a good upper bound is always available.

Hence, depth first search is used in developing the techniques in the sequel. It should be further noted that the philosophy used in developing these techniques was to create the simplest, most flexible framework within which to work so that a variety of acceleration techniques could be developed and tested. The concentration is on restricting the number of subproblems examined (which is exponential in n) rather than the amount of work spent on each subproblem (which is a low order polynomial in n).

III. NODE PARTITIONING

The first exact technique built around the above relaxation is one which generates subproblems by restricting the subtrees a node is allowed into. The procedure is described below. We begin by describing the initialization procedure.

Step 0: (Initialize)

0.1) Find an upper bound, z^* , using a heuristic to generate a good, feasible solution.

0.2) Find an MST, T , and identify arcs $(v_0, v_{i_1}), (v_0, v_{i_2}), \dots, (v_0, v_{i_k})$.

0.3) Reorder the nodes so that $v_{i_1}, v_{i_2}, \dots, v_{i_k}$ are now $v_1, v_2, \dots, v_k$.

0.4) For $1 \leq n$, set $R_{ij} = \begin{cases} \text{TRUE} & i \neq j \\ \text{FALSE} & i = j \end{cases}$

[R_{ij} is a logical variable which is set to TRUE if v_{ij} has been removed from consideration in this subproblem. Observations made above allow us to remove some v_{ij} immediately].

0.5) For $k < i \leq n$, set $R_{ij} = \begin{cases} \text{TRUE} & i > j \\ \text{FALSE} & i \leq j \end{cases}$

0.6) For $i \leq j \leq n$, find an MST, T_j , on $\hat{V}_j \cup v_0$, where $\hat{V}_j = \{v_i | R_{ij} = \text{FALSE}\}$ i.e., $\hat{V}_j$ is the set of nodes permitted in subtree j .

0.7) Set $w_{ij} = d_{ij}$, where d_{ij} is the distance from v_i to its predecessor in T_j .

0.8) For $j = 1, 2, \dots, k$, exchange weight between w_{ij} for

different values of i so that the resulting modified weights, w'_{ij} , satisfy:

$$w'_{ij} \geq w_{i0}$$

where $w_{i0} = d_{ip_i}$ in the unconstrained MST, T , generated in Step 0.2 above.

0.9) For $j = k+1, \dots, n$, exchange weight between w_{ij} so that the resulting w'_{ij} satisfy

$$w'_{ij} \geq w_{ij} - 1$$

The justification for all of these steps was given in Section 2.

Steps 0.8 and 0.9 guarantee that the individual w'_{ij} will all be at least as great as the weights assigned using unconstrained MST. Hence, this is a realization of the statement that the lower bound obtained using this procedure must be at least as great as the lower bound obtained using an MST. This also holds true for subproblems. Thus, we have initialized a subproblem with nodes $1, 2, \dots, k$ forced into subtrees $1, 2, \dots, k$, respectively, since, for $i \leq k$, $R_{ij} = \text{TRUE}$ for $i \neq j$. In the course of the depth first search, we keep track of the following variables:

d = The depth of the search, i.e., the number of nodes which have been forced. d is initialized to k .

d_{MIN} = The minimum allowable depth. d_{MIN} is initialized to k since at least k nodes should will be forced.

IW_d = The subtree which the d^{th} node is forced into.

The depth first search proceeds by forcing node $k+1$ into subtree 1; i.e., d is set to $k+1$ and IW_{k+1} is set to 1. It continues either by increasing d (to force another node) changing IW_d (to force

a node into a different subtree) or decreasing d (to release a node after forcing it successively through all subtrees). The depth first search procedure follows.

Depth First Search Procedure

Step 0: Initialize problem (Steps 0.1 through 0.9 above).

Set $d = k+1$

Set $d_{\text{MIN}} = k+1$ Set $IW_d = 1$

Step 1: Solve the currently defined subproblem; i.e., find a lower bound z_L , and a feasible solution z_F .

Step 2: If the current subproblem fathoms; i.e., $z_L \geq z_F$, go to Step 3; otherwise go to Step 5.

Step 3: Set $IW_d = IW_d + 1$

If $IW_d > NMAX_d$ go to Step 4; otherwise set up a new subproblem and go to Step 1. $NMAX_d$ is the highest indexed subtree which the node at depth d may be forced into. In section 3 we observed that one should not skip over subtrees. Thus:

$$NMAX_d = \max (K, \max [IW_i]) \quad K < i < d$$

Step 4: Set $d = d+1$

If $d < d_{\text{MIN}}$ stop; otherwise set up a new subproblem and go to Step 1.

Step 5: Set $d = d+1$

Set $IW_d = 1$

Set up new subproblem and go to Step 1.

To set up a new subproblem, one need only modify the values of a few R_{ij} to impose (or remove) the restriction implied by the alteration of d and IW_d . Thus, after d is set to $d+1$ and IW_d is set to 1:

$$R_{dj} = \begin{cases} \text{FALSE} & j = 1 \\ \text{TRUE} & j > 1 \end{cases}$$

After IW_d is set to $IW_d + 1$:

$$R_{d,IW_d} = \text{FALSE}$$

$$R_{d,IW_d-1} = \text{TRUE}$$

After d is set to $d - 1$:

$$R_{d+1,j} = \begin{cases} \text{FALSE} & j \leq NMAX_d \\ \text{TRUE} & j > NMAX_d \end{cases}$$

The search space can be further pared using the corollary to Theorem 1. If, in any subproblem, one finds that two nodes, v_i and v_j , both forced into the same subtree appear in separate subtrees in the MST formed on the set of permissible nodes in that subtree, then the subproblem may be discarded.

As was mentioned, this optimal technique based on generating a partition lends itself simply to obtaining a feasible solution to each subproblem. The partition generated at each step is feasible. One need only generate MST's on each group of nodes to obtain a feasible solution. A simple acceleration technique, which proved to be quite effective in practice, was to reorder the nodes $v_{k+1}, \dots, v_n$ by distance from v_0 , nearest first. This tended to increase the lower bound most rapidly. Such nodes, when restricted to a single subtree, were absent from all others, and the "deprived" subtrees were often forced to connect to v_0 over longer arcs.

The R_{ij} are used in the optimal partitioning procedure in a straight forward fashion; any element v_{ij} , with its corresponding $R_{ij} = \text{TRUE}$, is considered to be removed from the problem.

AD-A131 915

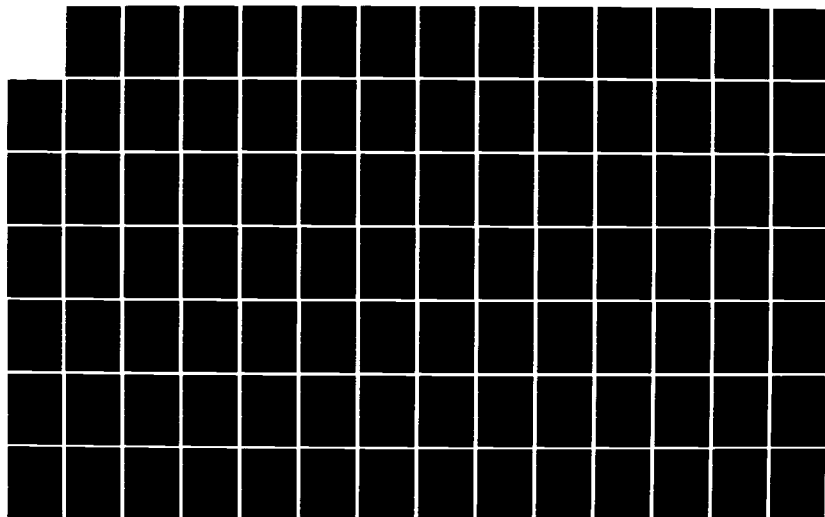
RESEARCH IN NETWORK MANAGEMENT TECHNIQUES FOR TACTICAL
DATA COMMUNICATION. (U) POLYTECHNIC INST OF NEW YORK
BROOKLYN R R BOORSTYN ET AL. 01 SEP 82 CECOM-80-0579-F

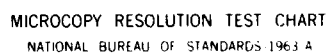
4/5

UNCLASSIFIED

DAK80-80-K-0579

F/G 17/2.1 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

The weight exchange procedure described as part of the initialization, which guarantees that the weight on each node will be at least as great as its contribution to the length of an MST, is used here as well. At each level, K , in the decision tree, we save the values of the w'_{ij} in a variable referred to as w_{ij}^K . We then demand that $w'_{ij} = w_{ij}^K \geq w_{ij}^{K-1}$ i.e., we exchange weights to enforce the restriction. The justification for doing so is identical to that used in the initialization procedure. Note that this exchange guarantees not only that the lower bound will remain tighter than an MST, but also that the lower bound will be monotone with the depth in the decision tree. Neither of these things is true without the exchange.

IV. ARC RESTRICTIONS

Another method of applying this relaxation technique to the solution to the CMST problem is to restrict arcs; i.e., to force arcs to be either "prohibited" or "required" as was done by Chandy and Russell and Elias Ferguson. Thus, the initialization and subproblem solution are essentially the same as they were in the technique described in the previous chapter, but the method of generating subproblems is different. The solution order is still a depth first search.

Some differences exist in the initialization procedure. Instead of forcing a node into a subtree, we simply "require" the arcs $(i_1, 0)$, ..., $(i_K, 0)$ which are part of the unconstrained MST. This is, of course, equivalent to what was done in the previous case. It is implemented in a slightly different way, however.

The entire procedure, both during initialization and during subsequent subproblem generation, restricts itself to dealing with established arcs, i.e., arcs which connect a node directly to v_0 or to other nodes connected to v_0 by established arcs. Thus, each "required" arc forces a node into a given subtree and each "prohibited" arc forces a node out of a given subtree. As a new subtree is encountered (i.e., ..., when an arc of the form (v_0, v_i) is made "required"), we simply assign the next available subtree number to the subtree.

Subproblems are generated by successively restricting (requiring or prohibiting) established arcs. We again use d to represent the depth of the search. Here, however, d refers to the number of forced arcs rather than the number of forced nodes. Note that while

the number of required arcs is limited to n , the number of prohibited arcs is not. We thus have a different type of decision tree than we did in the previous section.

When forcing nodes into subtrees, we dealt with a tree of depth n but with nodes of degree sometimes as great as n . Here, we deal with a binary tree of depth as great as $\binom{n}{2}$. It is not clear, however, especially with the paring techniques being used, which decision tree is actually larger.

The arc chosen for inclusion is, in each case the next arc to be brought in by Prim's MST Algorithm (25); i.e., the shortest arc connecting a node to some node connected to v_0 by required arcs. This has several advantages:

1. The arc chosen, if excluded, will tend to raise the lower bound. This is important as it helps control the size of the decision tree. Since a potentially large number of successive arc exclusions is possible, it is important that an arc exclusion result in an increase of the lower bound as often as possible so that the fathoming process will limit the depth of the search.
2. If the arc is of the form (v_0, v_i) , it need only be considered as "required" and not "prohibited." This is a direct consequence of Theorem 1.
3. If the arc (v_i, v_j) is prohibited, and hence, v_i is excluded from the subtree containing v_j , then so are all arcs of the form (v_i, v_k) , where v_k is forced into the same subtree as v_j . This is a direct consequence of an Elias and Ferguson result.

We omit the details of the remainder of the implementation of the arc restricting procedure as they are similar to the node restricting procedure described above.

V. COMPUTATIONAL EXPERIENCE

The procedures described in the previous two sections were coded in FORTRAN and run a PDP-10. In this section, we discuss the results of experiments run to test the behavior of their run time and effectiveness as a function of problem size and constraint tightness.

Problems were generated by reading in n and m and generating random X and Y coordinates for the nodes within a unit square. The location of the center was, in various problems, either random, centered, or in the corner. Euclidean distances were used. Most experiments were run with the center at the geographic center of the unit square; in this way, larger problems could be examined.

Several series of problems were run with identical values of n and m (and, of course, different randomly generated points) to see how stable the running time is from one problem to another. The standard deviation was found to be close to the mean for the problems run. This essentially says that we should not pay close attention to exact run times or the exact number of subproblems examined.

Series of problems were run varying n and m and using both the node restricting and arc restricting procedures. Both procedures were run with the identical problems and, furthermore, the same set of nodes was used (with new nodes added as the problem size grew) for all problems in this series. This results of this experiment are shown in Table 1. As can be seen, the running times for both procedures were comparable and run time grows exponentially with problem size. (It was gratifying to find that the optimal solution values found by both procedures always matched!)

It has already been mentioned that these procedures yield lower bounds which are at least as great as those obtainable using unconstrained MST's. This was verified empirically by actually generating lower bounds with MST's as the algorithm proceeded. The program was to print any exceptions, i.e., any times where the MST generated a higher lower bound; none occurred. Figure 3 shows some typical lower bound values obtained using partitioning and MST's. As can be seen, not only are the partitioning lower bounds greater, but they grow more quickly with depth. This is significant, as a linear increase in lower bound value will reduce the run time exponentially.

To measure the impact of the difference in lower bounds between the MST and partitioning methods, several problems were run first with the partitioning method and then with the MST method of lower bounding. The results of this experiment are shown in Table 2. As can be seen, the partitioning algorithm examines a much smaller number of subproblems, and apparently, its effectiveness increases as the problems grow larger. Thus, although it is somewhat more difficult to evaluate the lower bound using the partitioning algorithm than it is using an MST, it is less than n times as hard to do so. The reduction in the number of subproblems which must be examined appears to be sufficiently great to warrant the use of the partitioning technique. Indeed, as the problem size grows, its attractiveness seems to increase.

VI. SUMMARY AND CONCLUSIONS

The purpose of this study was to develop improved exact techniques for the solution of the CMST problem (a model of the multidrop line problem) so that known heuristics for the solution to that problem could be examined on a broader class of problems and so that new heuristics could be developed on the basis of what was learned. Much of this happened. An improved exact technique, based upon generating lower bounds using partitioning instead of MST's, was developed and computational experiments were run using it. The bounds yielded by these techniques were tighter than those yields by the MST based techniques, and hence, the number of subproblems which had to be examined in order to obtain a solution was smaller. Indeed, the decrease in the number of subproblems examined more than compensated for the increased effort required for the examination of each subproblem. Thus, the new techniques served their purpose in that they permitted the examination of problems not carefully examined before. In particular, it was possible to examine problems with very tight constraints, although it was not possible to examine problems of substantially greater size than had been previously examined.

Even with the improved technique, the growth of run time with respect to problem size was found to be exponential, albeit of a lower order than previously know exact techniques and of a much lower order than the solution space. Thus, one cannot use the technique for large problems. A number of acceleration techniques were developed and incorporated into the procedure.

Thus, it was possible to examine sufficiently interesting problems using the exact technique to make several insights into the problem. The first is that the performance of the known heuristic degrades as the constraint tightness increases and improves as the problem size increases. An improved heuristic [33] was also developed on the basis of this study.

REFERENCES

1. Bruno, J., Weinberg, L., "Generalized Networks: Networks Embedded on a Matroid, Part I," NETWORKS, Vol. 6, No. 1, January 1976, pp. 53-94.
2. Chandy, K.M., Lo, T., "The Capacitated Minimum Spanning Tree," NETWORKS, Vol. 3, No. 2, 1973, pp. 173-182.
3. Chandy, K.M., Russell, R.A., "The Design of Multipoint Linkages in a Teleprocessing Tree Network," IEEE TRANS. ON COMPUTERS, Vol. C-21, 1972, pp. 1062-1066.
4. Elias, D., Ferguson, M.J., "Topological Design of Multipoint Teleprocessing Networks," IEEE TRANS. ON COMM., Vol. C-22, 1974, pp. 1753-1761.
5. Esau, L.R., Williams, K.C., "On Teleprocessing System Design, Part II," IBM SYST. J., Vol. 5, No. 3, 1966, pp. 142-147.
6. Fisher, M.J., "Efficiency of Equivalence Algorithms," COMPLEXITY OF COMPUTER COMPUTATIONS, Plenum Press, 1972.
7. Ford, L.R., Fulkerson, D.R., FLOWS IN NETWORKS, Princeton University Press, Princeton, New Jersey, 1962.
8. Frank, H., Chou, W., "Topological Optimization of Computer Networks," PROC. IEEE, Vol. 60, Nov., 1972, pp. 1385-1397.
9. Frank, H., Frisch, I.T., Chou, W., Van Slyke, R., "Optimal Design of Centralized Computer Networks," NETWORKS, Vol. 1, No. 1, 1972, pp. 43-57.
10. Geoffrion, A., PERSPECTIVES ON OPTIMIZATION, Addison-Wesley, Reading, Mass., 1972, pp. 137-162.
11. Johnson, E., "On Shortest Paths and Sorting," PROC. ACM ANNUAL CONF., August, 1972, pp. 510-517.
12. Karnaugh, M., "Multipoint Network Layout Program" International Business Machine Corp., REPORT #RC3723, 1972.
13. Kershenbaum, A., "Computing Capacitated Minimal Spanning Trees Efficiently," NETWORKS, Vol. 4, No. 4, October 1974, pp. 299-310.
14. Kershenbaum, A., Boorstyn, R., "Centralized Teleprocessing Network Design," PROC. NATIONAL TELECOM. CONF., December 1976.
15. Kershenbaum, A., Chou, W., "A Unified Algorithm for Designing Multidrop Teleprocessing Networks," IEEE TRANS. ON COMM., Vol. C-22, 1974, pp. 1762-1772.

16. Kershenbaum, A., Van Slyke, R., "Computing Minimum Spanning Trees Efficiently," PROC. ACM ANNUAL CONF., August 1972, pp. 518-527.
17. Krogdehl, S., "A Combinatorial Base for Some Optimal Matroid Intersection Algorithms," TECH. REPORT SAN-CS-74-468, Nov. 1974, Computer Science Department, Stanford University.
18. Kruskal, J.B., "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem" PROC. AMER. MATH. COS., Vol. 7, 1956.
19. Lawler, E., "Matroid Intersection Algorithms," MATHEMATICAL PROGRAMMING, Vol. 9, No. 1, 1975.
20. Lawler, E., "Matroids with Parity Conditions: A new Class of Combinatorial Optimization Problems," MEMORANDUM No. ERL-M334, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, Nov. 1971.
21. Lawler, E., COMBINATORIAL OPTIMIZATION: NETWORKS AND MATROIDS, Holt, Reinhart and Winston, 1976.
22. Lin, S., "An Effective Heuristic Algorithm for the Traveling Salesman Problem" OPERATIONS RESEARCH, 1972, pp. 498-516.
23. Little, J.D.C., Murty, K.C., Sweeney, D.W., Karcl, C., "An Algorithm for the Traveling Salesman Problem" OPERATIONS RESEARCH, Vol. 11, November 1976, pp. 972-989.
24. Martin, J., SYSTEM ANALYSIS FOR DATA TRANSMISSION, Englewood Cliffs, New Jersey, Prentice-Hall, 1972.
25. Prim, R.C., "Shortest Connection Networks and Some Generalizations," BELL SYST. TECH. J., Vol. 36, 1957, pp. 1389-1401.
26. Reinfeld, N.V., Vogel, W.R., MATHEMATICAL PROGRAMMING, Printice-Hall, Englewood Cliffs, New Jersey, 1958.
27. Rosenstiehl, P., "L'arbre Minimum d'un Graphe," THEORY OF GRAPHS, P. Rosenstiehl, Ed., New York: Gordon and Breach, 1967.
28. Sharma, R.L., El-Bardai, M.T., "Suboptimal Communications Network Synthesis," PROC. INTERNATIONAL CONF. ON COMM. June 1970, pp. 19.11-19.16.
29. Tutte, W.T., INTRODUCTION TO THE THEORY OF MATROIDS, American Elseview, 1971.
30. Whitney, H., "On the Abstract Properties of Linear Dependence," AMER. J. MATH., Vol. 56, 1935, pp. 509-533.

31. Whitney, V.K.M., "Comparison of Network Topology Optimization Algorithms," PROC. 1972 ICCG, 1972, pp. 332-337.
32. Papadimitriou, C.H., "The Complexity of the Capacitated Tree Problem," NETWORKS, Vol. 8, No. 3, 1978, pp. 217-230.
33. Kershenbaum, A., Boorstyn, R., and Oppenheim, R. "Second Order Greedy Algorithms for Centralized Teleprocessing Network Design" IEEE TRANS. ON COMM., October 1980, p. 1835-1838.
34. Gavish, B., "New Algorithms for the Capacitated Minimal Directed Tree Problem", Proceedings of ICCG 80, Port Cester, 'Y, October 1980, p. 996-1000.
35. Kershenbaum, A., "Centralized Teleprocessing Network Des", Ph.D. Thesis, Polytechnic Inst. of NY, 1976.

Table 1: Number of Subproblems Examined

<u>n</u>	<u>m</u>	<u>Node Restricting</u>	<u>Arc Restricting</u>
8	2	16	34
8	3	13	24
10	2	120	199
10	3	57	68
10	4	67	73
12	2	298	696
12	3	375	362
12	4	171	138
14	2	766	723
14	4	526	379
16	3	not run	1085
16	4	818	737
18	3	not run	6832

TABLE 2: COMPARISON OF NUMBER OF SUBPROBLEMS EXAMINED
USING MST AND PARTITIONING AS LOWER BOUNDS

n	m	NUMBER OF SUBPROBLEMS (MST)	NUMBER OF SUBPROBLEMS (PARTITIONING)
8	3	87	24
12	3	2,767	362
20	7	4,205	146

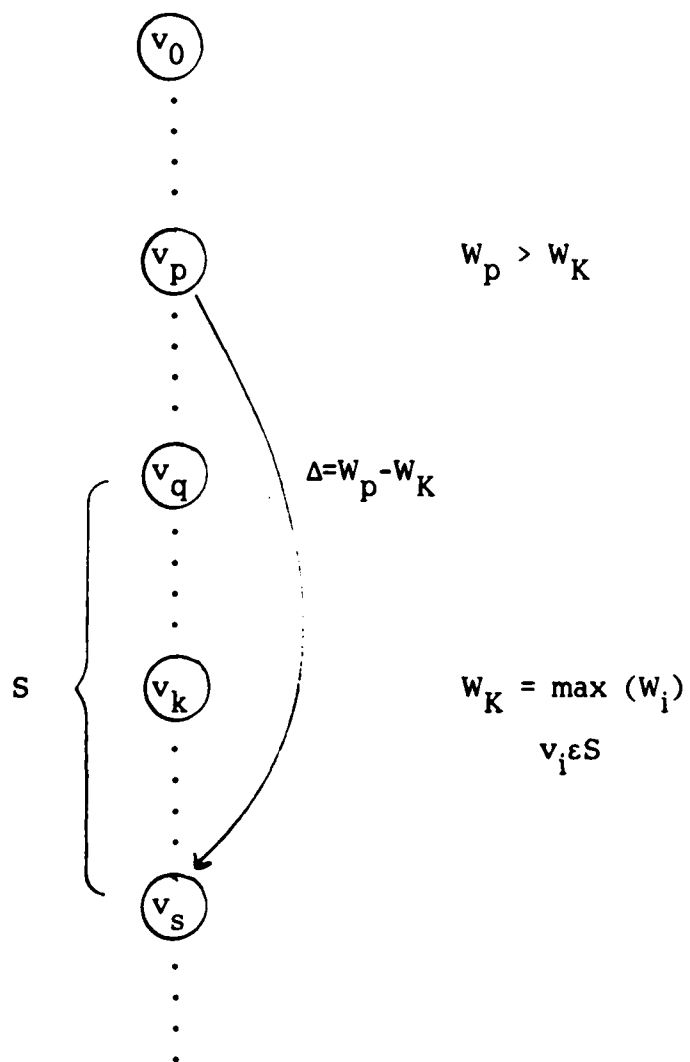
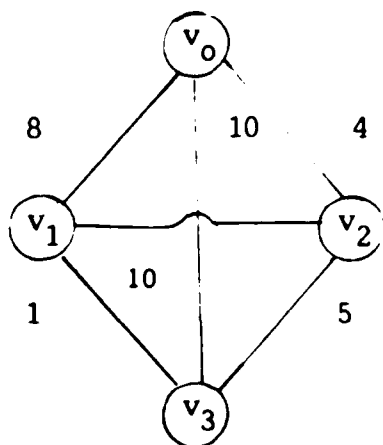
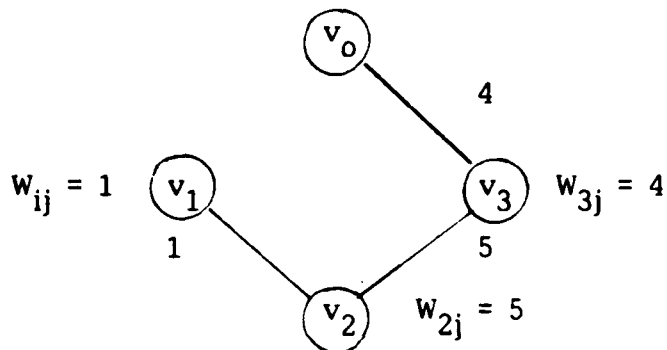


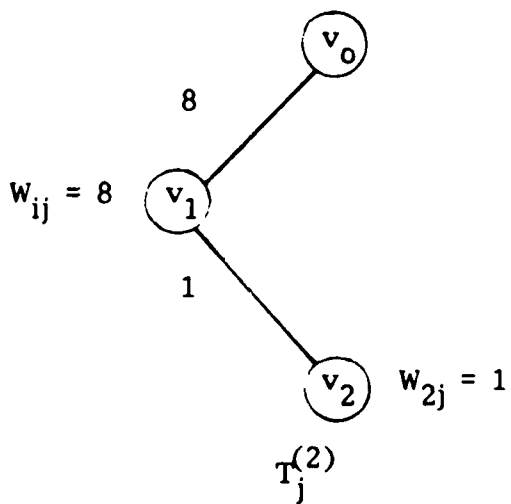
Figure 1



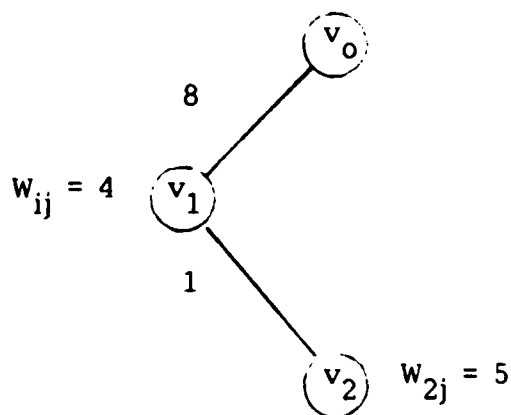
2a



2b



2c



2d

Figure 2

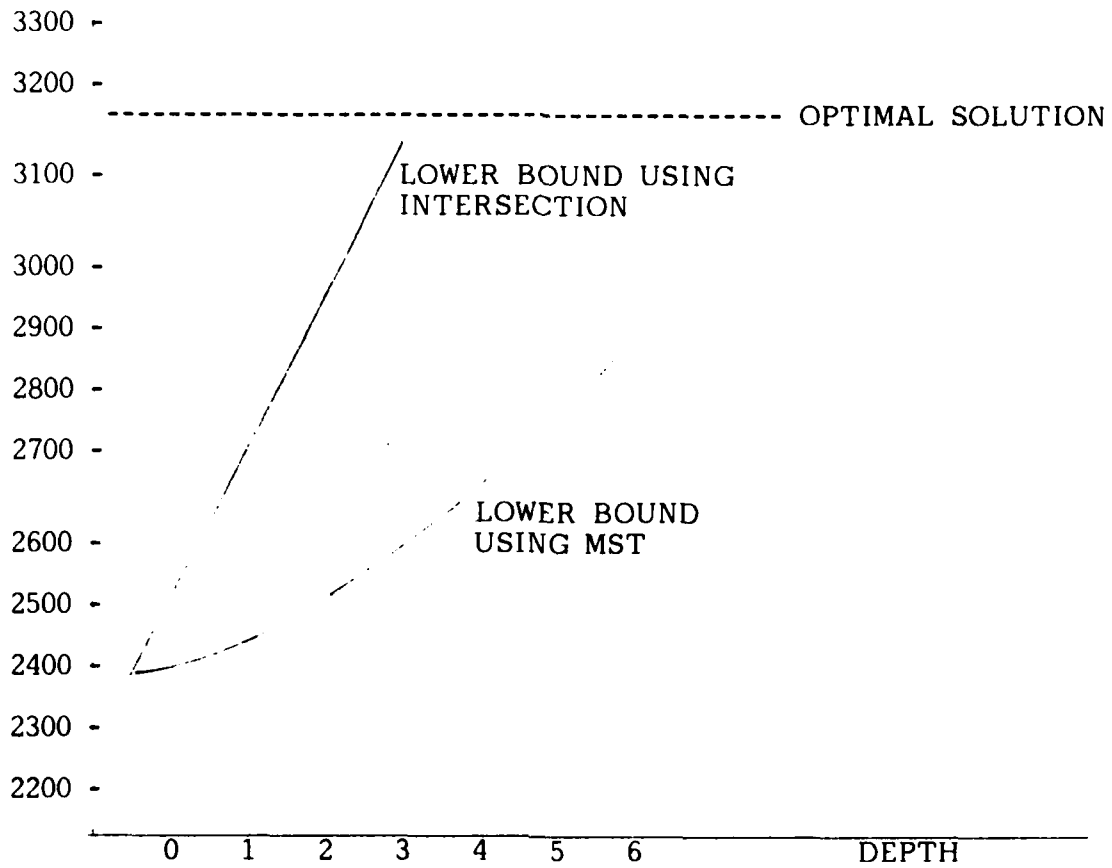


FIGURE 3: COMPARISON OF LOWER BOUND VALUES USING MST AND INTERSECTION

D.3 Network Design with an Objective Function Including
a Fixed Charge (Revised)

Kershenbaum

IEEE International Conference on Communications, June
1982, Philadelphia

NETWORK DESIGN WITH AN OBJECTIVE FUNCTION INCLUDING A FIXED CHARGE (Revised)

A. Kershenbaum*
Polytechnic Institute of New York

I. Introduction

There are many network design problems which would be easily solvable except for the presence of a fixed charge in the cost of each facility. The presence of the fixed charge however, which cannot be ignored without destroying the validity of the model, makes these problems much more difficult to solve. We begin by presenting several such problems and a general technique which may be of use in obtaining solutions to all of them. We then focus on one such problem and actually apply the technique to its solution.

II. Problem Statements

The most basic problem, and the one which we will ultimately focus on, is that of locating earth stations in a satellite network. We are given a fixed set of N terminal locations and a set of M potential earth station location sites. Terminals communicate with one another via the satellite and they reach the satellite via direct terrestrial connection to an earth station (see Figure 1). The cost of connecting terminal i to earth station j is given by c_{ij} . The cost of an earth station at the j^{th} potential site is denoted by d_j . The cost of a satellite link (or of the satellite itself) is assumed to be distance independent and linearly related to its capacity. Thus, the cost of the satellite links is not affected by the selection of earth stations and can therefore be ignored. We assume initially that there are no capacity constraints on the earth stations.

The problem described above is a well-known uncapacitated facility location problem (UFLP):

$$\text{Minimize } Z = \sum_{i=1}^N \sum_{j=1}^M c_{ij} X_{ij} + \sum_{j=1}^M d_j Y_j$$

* This work was supported in part by the US Army Comadcom under Grant DAAK-80-80-K-0579 and by the National Science Foundation under Grant ENG-7908120

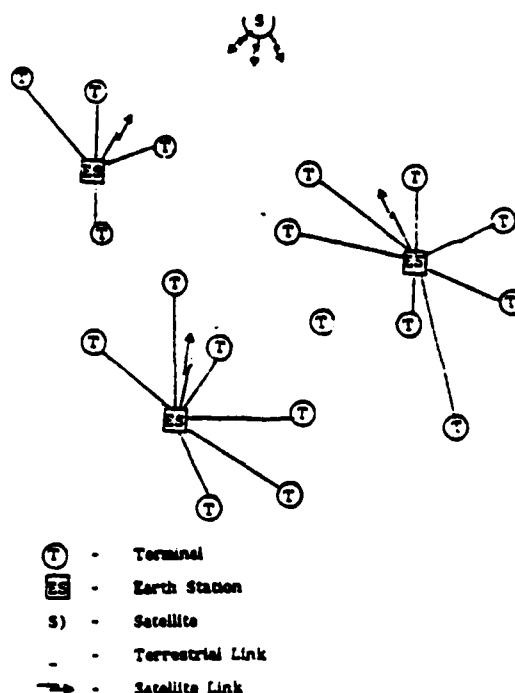


Figure 1. NETWORK MODEL

subject to:

$$\sum_j X_{ij} = 1 \text{ for all } i$$

$$X_{ij} \leq Y_j \text{ for all } i \text{ and } j$$

$$X_{ij}, Y_j \in \{0, 1\} \text{ for all } i \text{ and } j$$

where X_{ij} corresponds to the fraction of terminal i assigned to facility j any y_j is 1 if facility j is included in the solution.

This problem has been much studied [1-10]. The book by Handler and Mirchandani [5] is a compendium of related problems and solutions. The procedure of Erlenkotter [3] has been shown to be successful in obtaining optimal solutions to many significant uncapacitated facility location problems. The procedure we will describe in the following section is equivalent to Erlenkotter's procedure in the case of the uncapacitated facility location problem but differs from his in that it is based directly in combinatorics (rather than in a linear programming dual) and is directly extensible to many other problems described below, among them the capacitated facility location problem.

We note that the UFLP reduces to a trivial problem where each terminal is connected to its nearest (least-cost connection) earth station if we ignore the fixed cost of the earth stations. This is the property which unifies all the problems we will examine.

There are many extensions to the UFLP which are also part of this class of problems. If a capacity constraint is imposed on the earth stations, the problem reduces to an assignment problem if the cost of the earth stations is ignored and we assume either that the capacity constraint is on the number of terminals or we allow a terminal to be associated with more than one earth station and the cost of connection is linear with traffic.

Another problem is that of designing a network as above where not all terminals need be included. In this version of the problem, there is a value, V_i , associated with including terminal i in the network and we seek to maximize the profit, P , where P is given by

$$P = \sum_{i=1}^N V_i W_i - \sum_{i=1}^N \sum_{j=1}^M C_{ij} X_{ij} - \sum_{j=1}^M d_j Y_j$$

subject to

$$\sum_{j=1}^M X_{ij} = W_i \text{ for all } i$$

$$X_{ij} \leq Y_j \text{ for all } i \text{ and } j$$

$$X_{ij}, W_i, Y_j \in \{0, 1\} \text{ for all } i \text{ and } j$$

A capacitated version of this problem exists by adding the constraints

$$\sum_i X_{ij} t_i \leq S_j \text{ for all } j$$

where t_i is the traffic associated with terminal i and S_j is the capacity of earth station j .

This problem can be reduced to the preceding one, with fixed terminal locations, by the fol-

lowing transformation. A bogus potential earth station location, e_{m+1} , is introduced with $d_{m+1}=0$ and $C_{i,m+1}=V_i$ for $i=1, \dots, N$. The corresponding capacitated or uncapacitated problem is then solved to find the optimal solution including all terminals. The bogus earth station and all terminals associated with it are then removed from the optimal solution to yield an optimal solution to the problem with a choice of which terminals to include [4]. Thus, we will limit considerations to the fixed terminal problems in the sequel.

Another, closely related, problem is that of locating repeaters in a radio based network. Here we assume we are given a set of N terminal locations and M potential repeater locations. Terminals communicate via radio links to the repeaters. The repeaters communicate directly with one another, again via radio links, relaying messages between terminals. A terminal can or cannot communicate with a repeater depending on the distance between them and the nature of the intervening terrain. Thus, we have a UFLP with all $c_{ij} \in \{0, \infty\}$. As we will see, the nature of the cost matrix makes this problem different from the previously stated UFLP's in several important ways which will alter the effectiveness of the proposed solution techniques.

II. Basic Procedure

We now present a general procedure for the solution of all of the above problems. For the sake of clarity, we describe application of the procedure to the solution of the UFLP. Similar procedures have been proposed by Held & Karp [b], and Camerini [1] for the solution of a broad class of problems. Those procedures have been shown to be effective in some cases and ineffective in others. One of our objectives is to shed light on inherent differences among problems which differentiate them in terms of the effectiveness of these procedures and others like them.

Let $\hat{C}_{ij} = C_{ij} + P_{ij}$ where P_{ij} is a penalty added to the cost, C_{ij} , of connecting terminal i to earth station j . If $P_{ij} \geq 0$ for all i and j and

$$\sum_i P_{ij} = d_j$$

then the solution to the UFLP using $\hat{C}_{ij}$ in place of C_{ij} and $d_j=0$ is a relaxation of the original UFLP for any P_{ij} satisfying the above constraints. Here the facility costs are distributed over the terminals. The solution to the problem with $d_j=0$ is trivially obtained by selecting the minimum $\hat{C}_{ij}$ in each row. We note that the solution value thus obtained is indeed a lower bound to a feasible solution since

$$\sum_{i \in I_j} (\bar{C}_{ij} - C_{ij}) \leq d_j$$

where I_j is the set of terminals associated with earth station j in the solution to the relaxed problem.

Ideally, we would choose the P_{ij} to maximize the value of the relaxed optimum and thus provide as tight a bound as possible, i.e., solve:

Maximize over P_{ij}

$$[Z = \sum_i \min_j (C_{ij} + P_{ij})]$$

This problem has been successfully approached using subgradient optimization techniques [6,10]. For the moment, however, we defer further discussion of how best to set the P_{ij} and note that relatively simple heuristics often suffice in many cases while in other cases even the optimal P_{ij} (i.e. those maximizing the lower bound) leave a lower bound significantly lower than the optimal solution to the unrelaxed problem.

For the specific case of the UFLP, a linear programming problem can be solved to find the optimal P_{ij} :

$$\text{Maximize } Z = \sum_j V_j$$

$$C_{ij} + P_{ij} \geq V_i \text{ for all } i \text{ and } j$$

$$\sum_i P_{ij} = d_j \text{ for all } j$$

$$P_{ij} \geq 0$$

This is the basis for Erlenkotter's procedure but, again, he points out that often a simple heuristic suffices to set the P_{ij} .

The solution to the relaxed problem and the values of (slack) variables

$$S_j = \sum_i \text{MAX}(0, V_j - C_{ij})$$

can be used as the basis for a branch and bound procedure to find the optimum UFLP solution. Specifically, we begin by setting the P_{ij} and solving the relaxed problem. We then obtain a feasible solution to the original problem by selecting the earth stations, j , for which $S_j = 0$.

These correspond to earth stations which justify their cost by virtue of the savings obtained for terminals associated with them. To see this, note that, during the procedure, we try to make V_i (or equivalently, the minimum C_{ij} in row i) as large as possible in order to make the lower

bound as large (tight) as possible. Thus, S_j must equal 0 for some j with $C_{ij} \leq V_j$; or else we would make V_j larger.

Let $J^* = \{j | S_j = 0\}$. If, for all i , $C_{ij} \geq V_j$ for all $j \in J^*$ except for a single j the bound is tight. If $C_{ij} < V_j$ for two or more $j \in J^*$, there is a gap between the lower bound and the feasible solution. Erlenkotter suggests a procedure for adjusting the V_i to sometimes eliminate such situations. If a gap remains, one such j is selected and alternately forced in and out of the solution. The procedure is continued with the newly formed subproblems in place of the problem which spawned them. A subproblem is said to fathom, and hence is eliminated from further consideration, when its lower bound equals or exceeds the currently best feasible solution.

A similar procedure can be defined for the capacitated facility location problem. Again the fixed costs, d_j , are replaced by penalties, P_{ij} , added to the C_{ij} satisfying $P_{ij} \geq 0$ and $\sum_i P_{ij} \leq d_j$. An assignment problem is then solved to obtain a lower bound on the original problem. Heuristics for setting the P_{ij} and obtaining a feasible solution are less obvious here and very little work has previously been done on these problems.

III. A More General Procedure

We now describe the procedure in a more general context. Define a matroid, M , to be a 2-tuple (E, F) where E is a finite set of elements $\{e_j | j=1, \dots, N\}$ and F is a family of independent subsets of E . Our notion of independence is very general. We require only that independent sets, I , satisfy:

1. If $I \in F$ and $I' \subseteq I$ then $I' \in F$; i.e., all subsets of an independent set are independent.

- Given two independent sets, I_p and I_{p+1} , containing p and $p + 1$ elements respectively, there exists some $e_j \in I_{p+1}$, $e_j \notin I_p$ such that $I_p \cup \{e_j\} \in F$. That is, we can always find an element in the larger independent set (not already in the smaller one) which can be added to the smaller independent set without destroying independence.

The book by Welsh [9] is an excellent treatment of matroid theory and the book by Lawler [7] gives applications of matroid theory to the solution of combinatorial optimization problems including much of the discussion in this section. Camerini [1] gives much of the rest.

There are many types of matroids. Three types are of particular interest to us:

- Matric Matroids** - Let E be the set of columns of a matrix and F the family of sets of columns which are linearly independent.
- Graphic Matroids** - Let E be the set of edges of an undirected graph and F be the family of edge sets that contain no cycles (i.e., members of F are trees and forests).
- Partition Matroids** - Let E be an arbitrary set and B_i , $i = 1, \dots, K$ be a partition of E ; i.e., $\bigcup_i B_i = E$ and $B_i \cap B_j = \emptyset$ for $i \neq j$. Let F be the family of subsets of E containing no more than A_i elements from B_i , i.e.,

$$I \in F \text{ iff } |I \cap B_i| \leq A_i \text{ for all } i$$

It is fairly easy to see these three notions of independence satisfy the two required properties and thus that the three 2-tuples are indeed matroids.

Matroids are of interest in the context of the solution of combinatorial optimization problems because of the existence of very efficient procedures for the solution of problems which can be shown to be equivalent to finding the "best" independent set in a matroid.

Specifically, suppose a weight, w_j , is associated with each element $e_j \in E$ for some matroid (E, F) . We may wish to find the set $I^* \in F$ satisfying

$$W(I^*) = \max_{I \in F} (W(I))$$

$$\text{where } W(I) = \sum_{j \in I} w_j$$

Without the structure imposed by the matroid properties, the solution of this problem could entail examining all $I \in F$, potentially on the order of 2^N subsets if $|E|=N$. We can prove, how-

ever, using the matroid properties that the algorithm [7] solves the problem.

Greedy Algorithm

Step 0: $I^* \leftarrow \emptyset$
 $i \leftarrow 0$
 Reindex the e_j so that $w_j \geq w_k$ for $j < k$

Step 1: $i \leftarrow i + 1$
 If $i > N$, stop; I^* is the optimal set

Step 2: If $I^* \cup \{e_i\} \in F$ set $I^* \leftarrow I^* \cup \{e_i\}$
 Return to Step 1

Thus, we need only sort the elements of E and examine each element once, testing for independence with the previously chosen elements. Assuming the test for independence is not too difficult (and it is quite simple in the three cases above), the overall procedure is of the order of a polynomial in N . Thus, large problems can be solved.

Furthermore, even if w_j is replaced by any monotone (increasing) $f(w_j)$ the greedy algorithm will still find an optimal solution; that is, the optimal solution found using w_j above is also optimal for any monotone $f(w_j)$. Thus, f need only be evaluated for the e_j actually chosen.

Sometimes we wish to find an independent set of minimum total weight subject to the restriction that it also be of maximum cardinality. An example of this is a minimal spanning tree (i.e. finding a tree spanning all nodes and of minimum total length). The greedy algorithm can be used to solve this problem by setting $w_j = A - l_j$ where the e_j are edges of the given graph, l_j is the length of e_j , and $A > l_j$ for all j .

Unfortunately, many problems cannot be phrased as finding maximum weight independent sets in a matroid because the sets satisfying the constraints do not conform to the two properties given above. One may, however, be able to phrase the problem as the intersection of two or more matroids as follows.

The intersection of two matroids, $M_1 = (E, F_1)$ and $M_2 = (E, F_2)$, defined over the same set of elements but with two different notions of independence, is the 2-tuple (E, F) where $I \in F$ iff $(I \in F_1 \text{ and } I \in F_2)$. Unfortunately, the intersection of two matroids is not a matroid and thus many properties and algorithms do not carry over. There is, however, an efficient algorithm (c.f. Lawler) for finding the maximum weight set $I \in F$ for an intersection of two matroids.

The intersection of K matroids can similarly be defined as a 2-tuple (E, F) where $I \in F$ iff $I \in F_j$ for all $j=1, 2, \dots, K$. There is no known algorithm with a running time polynomial in the number of elements in E to find the maximum weight $I \in F$ for $K \geq 2$. Indeed, the problem for $K \geq 3$ has been shown to be NP-complete. It has also been shown that the problems for $K \geq 3$ are essentially equivalent to problems for $K=3$, i.e., it is only slightly more difficult to solve problems for general K than it is to solve problems for $K=3$.

We now turn to the detailed solution of one of the specific problems described in the preceding sections.

IV. The Repeater Location Problem

The techniques described in the preceding sections is applied to the solution of the repeater location problem in a broadcast radio network. In this problem, we are given a set of N terminal locations, t_i , a set of M potential radio repeaters, r_j , and a coverage matrix C_{ij} defined by:

$$C_{ij} = \begin{cases} 1 & \text{if } t_i \text{ can communicate with } r_j \\ 0 & \text{otherwise.} \end{cases}$$

A feasible network design corresponds to a selection of repeaters which "cover" all the terminals in the sense that each terminal can communicate with at least one repeater. (We assume the repeaters can always communicate with one another.) An optimal solution is a feasible set of repeaters of minimum cardinality. In this simple form of the problem, all repeaters are assumed to have identical cost (say, 1) and unlimited capacity. Thus, the problem is:

$$\begin{aligned} \text{Minimize } Z &= \sum_{j=1}^M Y_j \\ \text{s.t. } \sum_{j=1}^M C_{ij} Y_j &\geq 1 \text{ for } i = 1, \dots, N \\ \text{and } Y_j &= 0 \text{ or } 1 \text{ for } j = 1, \dots, M \end{aligned}$$

The variables Y_j correspond to selecting the repeater in potential location j ($Y_j=1$) or not selecting it ($Y_j=0$). This is, of course, a set covering problem (c.f. Garfinkle and Nemhauser).

We could also consider this problem as an uncapacitated facility location problem (UFLP). In this latter case, the coverage matrix corresponds to the matrix of costs of connecting location i to facility j in the UFLP as follows:

Coverage Matrix

$$C_{ij}=1$$

$\Leftrightarrow$

$$C_{ij}=0$$

$\Leftrightarrow$

Cost Matrix

$$C_{ij}=0$$

$$C_{ij}=1$$

One could thus consider solving the repeater location (RLP) problem as a UFLP. In practice this turns out to be a bad idea because of the symmetry of the RLP. In particular, all repeaters have identical costs and all terminals connect with identical cost to repeaters which cover them. An algorithm for the general UFLP would have to wade through a large number of alternate optima. We can eliminate a large part of this problem and in fact use the symmetry to advantage by applying the following dominance rules (c.f. Garfinkle and Nemhauser) to eliminate potential solutions:

$$\text{Let } R_i = \{j \mid C_{ij} = 1\}$$

$$\text{and } T_j = \{i \mid C_{ij} = 1\}$$

Thus, R_i is the set of repeaters which can cover terminal i and T_j is the set of terminals covered by repeater j .

Rule 1: If $R_i \subset R_k$ then t_i dominates t_k and t_k can be eliminated from the problem.

Rule 2: If $T_j \subset T_k$ then r_j dominates r_k and r_k can be eliminated from the problem.

It is easy to see that it suffices to consider only undominated terminals and repeaters in seeking an optimal solution to the RLP since any solution containing a dominated repeater could have included an undominated repeater instead and any solution which covers an undominated terminal also covers all terminals dominated by it.

To get the full benefit of dominance among terminals and repeaters, rules 1 and 2 should be applied repeatedly until no further reduction in the problem can be effected, because it may be that r_j dominates r_k after some dominated terminals are removed from consideration. Furthermore, in the course of a branch and bound procedure, where repeaters are forced into and out of the solution to define disjoint subproblems, rules 1 and 2 may be successfully applied to find further dominance in subproblems.

A version of the above procedure was coded up and run on a variety of problems. The results of these runs are shown in Tables I & II. As can be seen, the procedure is quite effective for many problems even though only relatively naive branching and bounding heuristics were used. In particular, the branching rule used was to find the repeater which covers the largest number of currently uncovered and undominated terminals and alternatively forcing it in and out of the solution. The lower bound, Z_{LB} , was generated for each subproblem using the following procedure.

Step 0: Z_{13} = number of repeaters forced into the solution in this subproblem

$$W_{ij} = \begin{cases} 0 & \text{if } C_{ij} = 1 \\ = & \text{if } C_{ij} = 0 \end{cases}$$

Step 1: For each repeater, r_j , let N_j = the number of uncovered and undominated terminals covered by r_j . Also, let $S_j = 1$. S_j is the remaining weight to be assigned in column j of the cost matrix

Step 2: Find the column (repeater), K , with maximum S_j/N_j . We only consider j such that $N_j > 0$. Let $b = S_K/N_K$

Step 3: For $i = 1, N$
 [For $j = 1, M$
 [If $C_{ik} = 1$ and $C_{ij} = 1$, $\{W_{ij} = W_{ij} + b$
 $N_j = N_j - 1$
 $S_j = S_j - b$
 $\}$
 $\}$
 $\}$

Step 4: If there is any j with $N_j > 0$, return to Step 2. Otherwise go to Step 5

Step 5: $Z_{LB} = Z_{LB} + \sum \min(W_{ij})$

Note that the above procedure simply assigns b to all elements in a row at each step and this is of the general form described above.

Research is currently in progress to identify more effective branching and bounding rules. We see (see Table I) that for problems with geographic coverage, i.e. when the values of C_{ij} are based on the distance between t_i and r_i , the procedure works very well. This is because the dominance is very effective in these cases. For random coverage, however, the dominance is less effective. The overall procedure is nonetheless sufficiently effective to solve problems with 50 locations.

TABLE 1 (Geographic Coverage)

NT	NR	D	SOL'N	COUNT
30	30	.024	18	1
30	30	.10	7	1
30	30	.46	2	1
50	50	.027	24	1
50	50	.484	3	1
100	100	.071	18	1
100	100	.109	11	3
150	150	.032	36	1
150	150	.103	11	3
150	150	.204	6	15
300	50	.210	7	1
300	50	.483	4	1

NT = Number of terminals

NR = Number of repeaters

D = Density, percentage of 1's in coverage matrix

SOL'N = Number of repeaters in optimal solution

COUNT = Number of subproblems examined

TABLE II (Random Coverage)

NT	NR	D	SOL'N	COUNT
50	20	.10	9	1
50	20	.20	7	31
50	20	.25	6	51
50	20	.30	5	15
50	20	.35	4	63
50	20	.50	3	15
50	20	.70	3	27
50	20	.80	2	1
50	50	.05	17	9
50	50	.10	11	157
50	50	.30	5	373
50	50	.50	3	41

References

1. Camerni, P. and F. Maffioli, "Heuristically Guided Algorithm for K-parity Matroid Problem", Discrete Math. 21 (1978) p. 103-116.
2. Coravajols, G., M. Fisher, and G.L. Newhouser, "On the Uncapacitated Location Problem" Annals of Discrete Math. 1 (1977) p. 163-167.
3. Erlen Kotter, D., "A Dual-Based Procedure for Uncapacitated Facility Location", Operations Research, 26 (1978) p. 992-1009.
4. Erlen Kotter, D., "Facility Location with Price-Sensitive Demands: Private, Public, and Quasi-Public", Management Science, 24 (1978) p. 378-386.
5. Handler, G. and P. Mirchandani, Location on Networks, MIT Press, 1979.
6. Held, M. and R.M. Karp, "The Traveling Salesman Problem and Minimum Spanning Trees: Part II", Math. Programming 1 (1971) p. 6-25.
7. Lawler, E., Combinatorial Optimization: Networks and Matroids, Holt, Rinehart, and Winston, 1976.
8. Spielberg, K., "Algorithms for the Simple Plant Location Problem with Some Side Conditions" Operations Research 17 (1969) p. 85-111.
9. Welsh, D.S.A., Matroid Theory, Academic Press, 1976.
10. Wolfe, P., M. Held, and E. Crowder, "Validation of Subgradient Optimization", Math. Programming 6 (1974) p. 62-88.

D.4 An Algorithm for Designing Circuit Switched Networks

Kershenbaum, Schneider, and Frisch

IEEE International. Conference on Circuits and
Computers, October 1980, New York

AN ALGORITHM FOR DESIGNING CIRCUIT SWITCHED NETWORKS

Kenneth S. Schneider*

Aaron Kershenbaum[#]

Ivan Frisch[†]

*SIGCOM, Inc., 134 Birchwood Pk. Dr., Jericho, New York

[#]PINY, Brooklyn, New York

[†]Network Analysis Corp., Great Neck, New York

ABSTRACT

An algorithm is described for designing circuit switched networks. In contrast to previous work, the algorithm considers both voice and data traffic. The network design is cost efficient. It provides topological design as well as routing doctrine. Performance analysis for both voice and data traffic is carried out.

INTRODUCTION

Circuit switching is a technology which has long been employed in voice communication networks. In such networks, prior to any communication taking place between two users, a connecting route is established between them. A sequence of channels or trunks is reserved for their use. The route is dedicated to their communication for its duration. If a route cannot be reserved, due to traffic congestion, the user initiating the request for connection is "turned away." His call is "lost." Circuit switched networks are thus characterized by an overhead delay in the establishment of the connecting route. However, they are also characterized by a low delay during the actual period of communication. This is due to the absence of any contention after a route has been set-up. These are conditions well-suited to voice communications. The duration of the typical telephone conversation is normally much greater than the time for route set-up. The overhead can be easily tolerated.

The present paper provides an algorithm for designing circuit switched communication networks. In contrast to previous work, attention is directed to combined voice + data traffic, rather than to voice traffic alone. There is great interest in designing networks for such integrated communications. Traffic volumes are much higher than in separate networks. Consequently, savings can be achieved through economy of scale.

In most integrated communications voice traffic (at least at the outset) is much higher than data traffic. Circuit switching provides a reasonable connection technology. This is the networking alternative, which from the delay point of view, is most well-matched to the dominant traffic element. If data traffic grows relative to voice traffic, arguments can well be made for using other methods of connection, such as packet switching or hybrid switching. However, in a combined environment voice traffic will always be significant. Circuit switched networks, with their mat-

ure switching technology, will always be attractive.

The algorithm presented is unified. It provides a circuit switched network design for a pre-specified end-to-end performance criterion -- the probability that a call is lost. The design specifies the network topology -- the connectivity of network nodes and the dimension of links in terms of number of trunks. A routing doctrine is specified with both primary and alternate routes. The average cross-network delay performance for data traffic is provided. Network cost is given, having been computed from both link and node requirements. It should be emphasized that only the backbone network is considered. Local traffic is assumed to be concentrated at the backbone nodes using standard teleprocessing techniques such as multiplexing and concentration. The authors have drawn from a wealth of previous work in the area of circuit switching. Specific acknowledgement is paid to the work of Katz and Kneplev [2] and [3]. However, it is believed that this is the first algorithm which considers both voice and data traffic carried on the same circuit switched network. Furthermore, the voice and data traffic are not multiplexed on the same channels, (in a TASI type arrangement), rather they contend for available link capacity.

DESCRIPTION OF THE ALGORITHM

High Level Description

Before proceeding with a detailed description of the design algorithm it would be worthwhile to take a look at a high level block diagram, illustrated in Figure 1. The major functional segments of the design procedure are shown here. The PRELIMINARY SEGMENT provides the necessary design inputs, e.g. node locations, traffic, desired performance, etc. It also "massages" the inputs, putting them in the form necessary for later algorithmic procedures. The TOPOLOGICAL DESIGN SEGMENT carries out the optimization of the network design. It provides a cost effective connectivity of the nodes for the desired traffic. It also dimensions the network links to meet the desired performance goal. The ROUTING SEGMENT gives the doctrine to be used on this topological design. It defines the routes through the network used to connect desired end-user nodes. The block labelled NETWORK SPECIFICATIONS summarizes the network design up to the point in the algorithm; that is, the topology, trunk dimensions and routing doctrine. The VOICE PERFORMANCE ANALYSIS SEGMENT analyzes the performance of

the network stored in the NETWORK SPECIFICATION with respect to voice traffic. It estimates the average probability of a lost call for the given input traffic. This is the probability that a voice communicator desiring connection to a remote node is turned away. Nothing is done if the average probability of a lost call is within a given tolerance of the goal in the PRELIMINARY SEGMENT. Otherwise, instructed adjustments are fed back to the NETWORK SPECIFICATION. The topological design is not changed. However, the link dimensions in terms of trunks are changed to bring the measured average probability of a lost call close to the desired goal. The DATA PERFORMANCE ANALYSIS SEGMENT measures the average cross network delay for data traffic carried in the resulting network stored in the NETWORK SPECIFICATION block. The desired goal for delay does not impact network design. This block is merely a post-processor. The COSTING SEGMENT block computes the transmission and switch cost for the resulting network. Finally, the OUTPUTS block summarizes the circuit switched network design.

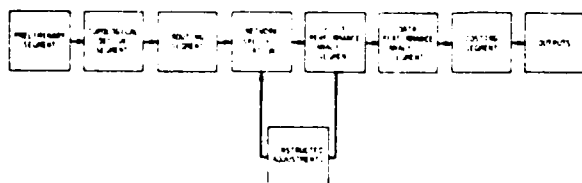


Fig. 1 High Level Block Diagram of Algorithm

Detailed Description

A detailed description of each portion of the algorithm structured in Fig. 1 will now be given.

Preliminary Segment Consists of three separate blocks shown in Fig. 2. The INPUT BLOCK stores the constraints which the network is designed to satisfy. Here are the backbone nodes and their locations. These are labelled $1, 2, \dots, i, \dots, j, \dots, NN$. The voice traffic is stored as a matrix $[V_{ij}]$ and the data traffic as a matrix $[D_{ij}]$. Voice traffic units are Erlangs and data traffic units are bits per second. The i - j entries of each of these matrices constitute the traffic originating at node "i" which is destined for node "j". Voice traffic is taken to be symmetric. A party initiating a voice call must automatically provide an answer-path. An Erlang of voice traffic originating at node "i" with destination "j" automatically implies an equal flow from "j" to "i". The design algorithm assumes a particular "architecture" for handling data traffic. Parameters associated with this architecture are provided in the INPUT BLOCK. Specifically, two different types of data messages are allowed to be handled by the desired circuit switched network, a bulk file transfer (Type A) and a short interactive message (Type B). A fixed fraction of the data traffic pattern, $RHOA$ is assumed to correspond to Type A traffic and a fixed fraction, $RHOB$ is assumed to correspond to Type B traffic. Type A file transfer messages are taken to consist of IA bits and are handled in the fol-

lowing way. Upon the start of a Type A message generation, a connecting route to the destination is obtained by dial-up. It turns away relatively is carried out every ten seconds until a connection is obtained. The Type A message is buffered and fed into the connecting route at a rate of R bits per second. The route is held until the end of a single message transfer. The connection is then broken. IA and R are listed in the INPUT BLOCK. Typical values are 10^7 and 9600. Type B interactive traffic is assumed to have the following characteristics. An interactive message is LB bits long. A stream of interactive messages generated at one node with another as the destination have time separations of T seconds. T is the "think" time which depends upon the speed and attention of the terminal user generating the messages. Upon the start of a Type B message generating session (the generation of the first message) a route is dialed up, as for a Type A message. However, after the first message is sent, the route is not closed down. Rather, it is taken to be held open until $K-1$ additional Type B messages are sent over it. Messages are assumed buffered and fed to the connecting route at a rate of R bits per second. The parameters LB , T and K are provided in the INPUT BLOCK. Typical values are $LB=1000$ -- a standard teletype line, $T=10$ sec. and $K=200$ or 1. $K=200$ corresponds to classical circuit switching of an interactive data stream. Here, after dial-up a terminal user holds a connecting route to a computer for an entire computing session. $K=1$ corresponds to "fast circuit switching." Here, dial-up and routing are carried out for each interactive message.

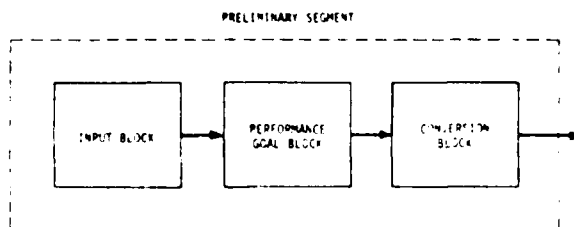


Fig. 2 Details of Preliminary Segment

The PERFORMANCE GOAL BLOCK stores the end-to-end performance desired for the circuit switched network being designed. This is the average end-to-end loss probability, P_L , averaged over all node pairs and weighted by traffic. It also includes the average cross network delay desired for Type A and Type B messages -- again averaged over all node pairs but weighted only by data traffic.

Before the actual network design procedure can begin some processing of the parameters stored in the INPUT BLOCK must be carried out. This is done in the CONVERSION BLOCK. Specifically, the inter-node distances are computed using the node locations. The inter-node distances are stored in the matrix $[L_{ij}]$. The i - j entry of this matrix is a measure of the cost of a link directly connecting node "i" to node "j". The network design procedure operates with a combined traffic matrix of voice

and data traffic. The combined entry has units of Erlangs. Hence, the data traffic has to be converted to equivalent Erlang values and added to the voice traffic. This is carried out in this block, by computing the average number of data calls per second and the call holding time for the assumed data traffic characteristics and handling strategy. The following formula is used to compute ED_{ij} , the Erlang value of the data traffic originating at node "i" with node "j" as destination.

$$ED_{ij} = \left\{ (RHOA) \left(\frac{D_{ij}}{R} \right) \right\} + \left\{ (RHOB) (D_{ij}) \left[\frac{1}{R} + \frac{(1-1)T}{K LB} \right] \right\}$$

With this, the Total Traffic Matrix $[T_{ij}] = [V_{ij}] + [ED_{ij}]$ is computed.

Topological Design Segment is the heart of the network design algorithm. Optimization of the network is carried out in this segment. Details of the Topological Design Segment are illustrated in Fig. 3. Certain parameters must be defined before the optimization procedure of this segment can commence. This definition takes place in the INITIALIZATION BLOCK. Specifically, the value of a parameter "E" is set equal to $\frac{1}{\sqrt{NN}} P_L$. This cor-

responds to the link blocking probability "guess-timate" required to achieve the desired end-to-end average loss probability. P_L was given in the PRELIMINARY SEGMENT. This "formula" for E is derived from a "union" bound assuming that on the average there are $\sqrt{NN}$ links in each end-to-end route. A parameter "TEST" is set initially equal to ∞ . This parameter is used in the test for convergence of the optimization procedure. Finally, a matrix $[W_{ij}]$ is defined. This is a $NN \times NN$ matrix. It is referred to as the "weight matrix." Initially, it is defined as having all entries equal to 1.

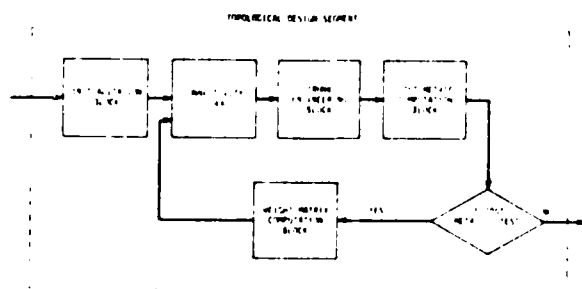


Fig. 3 Details of Topological Design Segment

The TOPOLOGICAL DESIGN SEGMENT operates by connecting nodes with "short" high usage routes. With any "rational" cost measure this should be economic network. With parameters defined in the INITIALIZATION BLOCK, the CONNECTIVITY BLOCK is entered. Here, nodes are connected by routes using the following procedure. The CONNECTIVITY BLOCK computes a matrix $[Length_{ij}]$. The i-j component of this matrix is given by the formula

$Length_{ij} = W_{ij} L_{ij}$. This is an $NN \times NN$ matrix. $Length_{ij}$ is a weighted distance between node "i" and node "j". With the matrix $[Length_{ij}]$ defined, the CONNECTIVITY BLOCK considers each successive node pair $i-j, i \neq j$. It uses a "shortest path" algorithm to connect the nodes in each pair. Each ordered node pair $i-j$ is connected by a route of directed links from i-to-j. However, lengths in the shortest path algorithm are taken relative to the weighted distances given by matrix $[Length_{ij}]$.

In the first iteration the network put out by the CONNECTIVITY BLOCK will be fully connected.

With topological connectivity defined the TRUNK ENGINEERING BLOCK is entered. Each link of the connected network is considered. All of the end-to-end shortest path routes, determined in the CONNECTIVITY BLOCK, utilizing a given link are enumerated. The end-to-end traffic requirement for each route is obtained from the matrix $[T_{ij}]$. For

a given link between nodes "i" and "j", the sum end-to-end traffic, A_{ij} , of all "weighted" shortest path routes traversing this link is obtained. If circuit switched "calls" are routed on these paths, A_{ij} upperbounds the traffic that will be offered to directed link (i,j). The TRUNK ENGINEERING BLOCK solves the equation $E = E(S_{ij}, A_{ij})$ for S_{ij} and quantizes it. $E(\cdot, \cdot)$ is the blocking probability given by the Erlang B formula. S_{ij} provides dimension of directed link (i,j) in trunks.

The COST METRIC BLOCK computes a measure of the cost of the connected and trunk dimensioned network which has been produced by the two previous blocks. $COST METRIC = \sum W_{ij} L_{ij} S_{ij}$. The sum is

over all ordered node pairs i,j which are adjacent in the network. The value of COST METRIC is compared to the value of the parameter TEST. If it is greater than or equal to TEST, the TOPOLOGICAL DESIGN SEGMENT is exited. If COST METRIC is less than TEST the optimization procedure continues. Since TEST = ∞ initially, the procedure continues for at least two iterations. When it continues, TEST is set = COST METRIC and a new weight matrix $[W_{ij}]$ is computed from the following formula,

$$W_{ij} = \frac{\partial E(S,A)}{\partial A} / \left[-\frac{\partial E(S,A)}{\partial S} \right]$$

The denominator in this formula is the rate of change of $E(S,A)$ over the interval $[S, S+1]$, i.e. $E(S+1, A) - E(S, A)$. This can be reduced to

$$W_{ij} = \left\{ \frac{E(S-1, A) - E(S, A)}{E(S, A) - E(S+1, A)} \right\} \left\{ \frac{1}{[(A/S) E(S-1, A)] + 1} \right\}$$

The above two formulas are evaluated at $S=S_{ij}$, $A=A_{ij}$. The TOPOLOGICAL DESIGN SEGMENT then goes the next iteration. The CONNECTIVITY BLOCK now computes the matrix $[Length_{ij}]$ using the new weight matrix. This iterative procedure continues until a $COST METRIC \geq TEST$ is obtained. By studying the formula for W_{ij} one may note that low values are obtained when dealing with a link which in response to an incremental increase in offered flow can achieve the desired blocking probability, E, with as few additional trunks as possible. The TOPOLOGICAL DESIGN SEGMENT achieves a connectivity of the nodes by employing short, highly utilized directed links. This provides a cost-efficient circuit switched

network. Note that the use of the weight matrix and the specific formula for the weight are similar to the convex branch elimination method used to design packet switched networks. [1]

Routing Segment In circuit switched networks when a communicator at "i" wishes to "talk" to one at node "j", a connection is first attempted by reserving trunks on a path termed the "primary route." If due to blocking the connection cannot be completed an attempt is made at reserving trunks along an alternate route. The ROUTING SEGMENT of the algorithm, shown in Fig. 4, provides both primary and alternate routing doctrines for the network derived in the TOPOLOGICAL DESIGN SEGMENT. The primary route from node "i" to node "j" is taken as the shortest directed path between these two nodes. "Shortness" here is taken to mean length relative to $[Length_{ij}]$, as computed in the last iteration of the TOPOLOGICAL DESIGN SEGMENT. The alternate routing doctrine defined is a form of "progressive routing." A detailed description of it is given in Ref. [4]. Basically, at each of the interior nodes of the primary route connecting "i" to "j", there is a table which lists routes to node "j". These routes diverge from the primary route at the interior node. The routes in each table are ordered in a hierarchical manner. In attempting to reserve the primary route, if blocking is encountered at interior node "k", the routing table residing at this node is consulted. An attempt at completing the route is then made along the first entry in the table. If there is blocking along this route the second is tried, etc. If there is blocking along all of these routes the "caller" is turned away from the network. The primary route and possible alternate routes in one situation are illustrated in Fig. 5. In the version of the algorithm which has been implemented, diverging alternate routes are taken to be the second shortest route, the third shortest route, etc. Shortness is taken according to the last values of $[Length_{ij}]$.

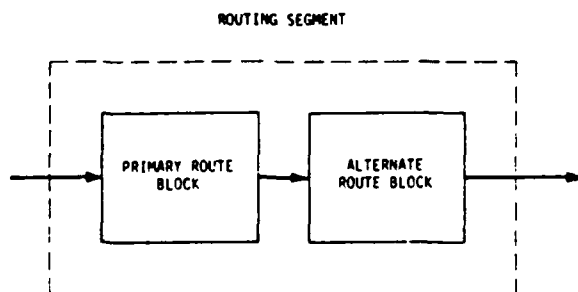


FIG. 4 DETAILS OF ROUTING SEGMENT

Network Specification In this module of the algorithm the network design is stored; that is, the topology, link trunk dimensions and routing doctrine are summarized.

Voice Performance Analysis Segment and Instructed Adjustments The goal of the algorithm is

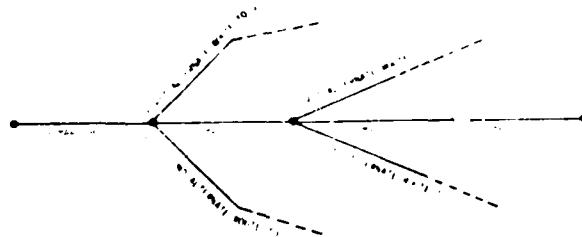


Fig. 5 The Primary Route From Node "i" to Node "j" with Diverging Alternate Routes

to design a cost-efficient network satisfying a constraint on P_L . The TOPOLOGICAL DESIGN SEGMENT attempts to meet this constraint by sizing the trunk dimensions of each link to correspond to the trunk engineering by utilizing the Erlang B formula. However, the data traffic and alternate routed traffic may not have the Poisson arrival statistics assumed by this formula. As a result, the network initially stored in the NETWORK SPECIFICATION BLOCK may not meet the desired P_L . The VOICE PERFORMANCE ANALYSIS SEGMENT and INSTRUCTED ADJUSTMENT BLOCK correct this deficiency. The VOICE PERFORMANCE ANALYSIS SEGMENT analyzes the network stored in the NETWORK SPECIFICATION BLOCK and obtains an estimate, called $\hat{P}_L$, of P_L . Details of the analysis procedure are given in Ref. [4]. The estimate $\hat{P}_L$ is compared to the desired value of P_L . If it is within Δ , (typically $=10^{-3}$), the network stored in the NETWORK SPECIFICATION BLOCK is accepted as the design and the algorithm proceeds to the DATA PERFORMANCE ANALYSIS SEGMENT. If $\hat{P}_L$

is not within Δ the algorithm proceeds to the INSTRUCTED ADJUSTMENTS BLOCK, which computes a new value of link blocking probability from the previous one for trunk engineering. The new value, $E(\text{new})$, is obtained from the following formula. $E(\text{new}) = E + 1/\sqrt{NN} (P_L - \hat{P}_L)$. The INSTRUCTED ADJUSTMENTS BLOCK then takes the network topology stored in the NETWORK SPECIFICATION BLOCK. It carries out the trunk engineering of each link to satisfy the blocking probability, $E(\text{new})$. A new network is then produced; the routing doctrines stay the same. This new network replaces the previous one stored in the NETWORK SPECIFICATION BLOCK. Its loss probability is analyzed in the VOICE PERFORMANCE ANALYSIS SEGMENT. The procedure continues as described above until the trunk engineering produces a $\hat{P}_L$ within Δ of the desired P_L .

Data Performance Analysis Segment This block

computes $\bar{D}_A$ and $\bar{D}_B$, the average cross-network delay for Type A and Type B data messages, respectively. The average is taken, among other things, with respect to all node pair data traffic requirements. These quantities are computed by obtaining $\bar{D}_{ij}$ for each traffic type. This is the average cross-network delay for a data message (either Type A or B) which has its source at node "i" and destination at node "j". $\bar{D}_{ij}$ is computed by the algorithm using, $\bar{D}_{ij} = C_L + C_P + C_t + C_r$. C_L is

the average time taken in waiting to redial until a route is finally granted. It has been assumed that if a data communicator is turned away he redials after 10 seconds. Hence, $C_L = 10 P_L / (1 - P_L)$. C_p is the signal propagation time, which is determined by the actual length of the i-to-j primary route. Most traffic will go along this route. C_t is the transmission time for a message, given by the formula $C_t = \text{message length} / R$. Message length is L_A or L_B as specified in the PRELIMINARY SEGMENT. C_r is the average time taken to set-up the route connecting node "i" to "j" or to turn the communicator away because of inability to reserve a route. Let this actual route set-up time be designated as T_r . For the Type A file transfer message, or for a Type B interactive traffic with classical circuit switching, the entire set-up delay is borne by the first message. The remaining K-1 messages have zero set-up time. In this case the average $C_r = T_r / K$. The value of T_r (or C_r) depends upon the type of signalling employed by the circuit switched network in order to set-up a route -- the methods used to transmit routing information through the network in reserving trunks. The design algorithm assumes that Common Channel Inter-switch Signalling is employed. Separate digital links are assumed to connect the nodes of the network and are reserved strictly for signalling information. Routes are reserved by sending small addressing packets on these signalling links. Transmission occurs at the rate of R_s bits per seconds. Typically, $R_s = 9600$. A circuit switch model is incorporated into this block of the algorithm in order to account for delays in the switch. These delays correspond to queuing delays in having the switch deal with the routing requests and finding a path through the switch.

Costing Segment In this block the transmission cost and the switch cost of the circuit switched network design are computed. Transmission cost of the network is taken to be the sum of the separate transmission costs of each of the links in the network. Switch cost is taken to be the sum of the costs of the individual switches. Because of the modularity of this block it can be programmed to model any tariff and circuit switch cost. For example, in our experiments we have used the DDS line rates.

Outputs In this block the final circuit switch network design with performance and cost is stored.

EXAMPLE NETWORK DESIGNS

The topologies of typical network designs obtained with the algorithm are shown in Figs. 6a, b, & c. These are eight node networks spanning the continental United States. They were designed for total voice traffic = 5400 Erlangs, total data traffic = 361 kbps, $RHOA = RHOB = 0.5$, $K = 200$ (classical circuit switching), $LB = 1000$, $T = 10$. The designs were carried out for different values of P_L . Costs are noted. These designs were obtained by implementing the algorithm in Fortran. The running time was approximately five seconds on

a DEC 20/50 with the TOPS 20 operating system. The memory required was close to 65,000 (36 bit) words.

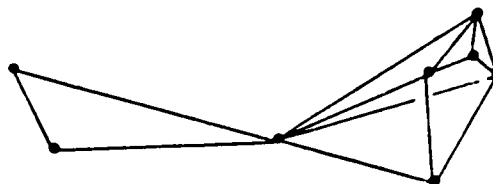


Fig. 6a $P_L = .002$, Cost = \$2,476,025 per month

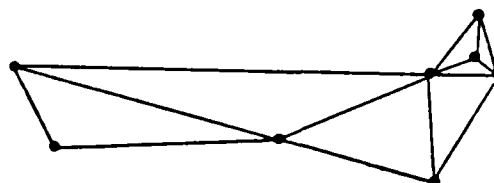


Fig. 6b $P_L = .01$, Cost = \$1,903,429 per month

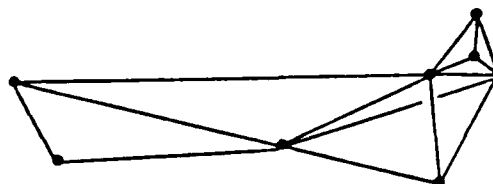


Fig. 6c $P_L = .1$, Cost = \$1,832,319 per month

REFERENCES

1. Cantor, D. G. and Gerla, M., "Optimal Routing in a Packet Switched Computer Network," IEEE Trans. on Computers, C-23, no. 10, pp. 1062-69, Oct. 1974.
2. Katz, S., "Statistical Performance Analysis of a Switched Communication Network," Proceedings of the International Teletraffic Congress, 1967.
3. Knepley, J. E., "Minimum Cost Design for Circuit Switched Networks," Technical Note No. 36-73, Defense Communication Agency System Engineering Facility, Reston, Virginia, July 1973.
4. Schneider, K. "An Algorithm for Computing Average Loss Probability in a Circuit Switched Communication Network", IEEE Trans. on Communications, COM-25, no. 1, pp. 27-32, Jan. 1980.

D.5 Optimization of Telephone Networks Using the New

WATS Tariff

Kershenbaum and Kozicki

Journal of Telecommunication Networks, Summer 1982

Optimization of Telephone Networks Using the New WATS Tariff

Aaron Kershenbaum

Polytechnic Institute of New York

Zvi Kozicki

ConTel Information Systems*

ABSTRACT The problem of determining the optimal number of WATS lines given an hourly profile of outbound calls from a given location is considered. It is shown that the relationship between cost and carried load under the new WATS tariff with a very small fixed charge for lines is relatively smooth and unimodal. It is thus possible to obtain globally optimal solutions to most problems by using a very efficient local optimization procedure which only considers addition and deletion of single lines and the exchange of one line for another. Computational experience with this procedure and a comparison with exhaustive search are presented.

1. Introduction

Wide Area Telephone Service (WATS) is a service offered by telephone companies to provide volume discounts to larger users. Until recently, WATS offerings included both full and measured service. With full service WATS, the user was permitted essentially unlimited calling for a fixed monthly fee. Measured service provided for a lower fixed fee, 10 hours of free service and an hourly fee for usage in excess of 10 hours per month. The new WATS tariff eliminates full period WATS service and offers a somewhat modified form of measured service.

There are actually many WATS offerings for both inward and outward callers to and from a given location on either an intrastate or interstate basis. We will focus here on the interstate out-WATS service offered by AT&T

Long Lines within the continental U.S. In this case the continental U.S. is divided up into 5 bands. Band 1 consists of area codes close to the caller but outside his state. Band 2 consists of these area codes plus others somewhat farther away. This continues until Band 6, which includes all area codes in the U.S., including Alaska, Hawaii, Puerto Rico and the Virgin Islands. In this paper, we will deal primarily with the 5 WATS bands which cover the continental U.S. outside the caller's state. Thus, each WATS band includes all the lower numbered bands and represents a roughly fixed percentage of the telephones in the U.S. regardless of where the caller is. The specific area codes involved are a function of which state the caller is in. The service applies only to interstate calls. Most states, if not all, have intrastate WATS service but may have different characteristics and are not the subject of this paper. The service applies only to outward calls from a single location. The in-

* This work was supported in part by a grant from ConTel Information Systems

interested reader is referred to [1] for all the details of this tariff. Its salient numerical parameters are described in the following sections.

In order to use WATS service, the user orders one or more lines, which we refer to as a group, in one or more of the 6 bands. A line in a given band may be used to place an interstate call to any area code in that band. Note that if the user has both Band 1 and Band 2 groups, for example, a call to an area code in Band 1 may use a line in either group. In addition to ordering lines, users must decide on how to route calls. That is, they must decide whether to allow an outgoing call to use a higher numbered band if no lines in its band are available, or overflow the call to DDD (Direct Distance Dialing; i.e., the call is placed as an ordinary long distance call), or block the call entirely.

2. The New WATS Tariff

The new tariff, as it is used in this paper, is described in detail on the 8th revised page 14.1 of AT&T Tariff 259 [1]. The significant change from the old tariff is that full period lines are eliminated and only measured lines remain. Specifically, the cost of a WATS line carrying H_d hours of traffic in the daytime, H_e hours of traffic in the evening, and H_n hours of traffic at night costs:

$$C = 27.50 + C_d(H_d) + C_e(H_e) + C_n(H_n)$$

where

$C_d(H_d)$ is the cost for H_d hours of daytime traffic, computed from a piecewise linear function, i.e. so much per hour for the first 15 hours, so much per hour for the next 25 hours, etc. For example, for a Band 5 line in New York City,

$$\begin{aligned} C_d &= 19.91 \times [\min(H_d, 15)] \\ &+ 17.71 \times [\min(H_d - 15, 25)]^+ \\ &+ 15.53 \times [\min(H_d - 40, 40)]^+ \\ &+ 13.13 \times [H_d - 80]^+ \end{aligned}$$

where $[X]^+$ is X if $X \geq 0$ and 0 otherwise

C_e is computed in a similar manner. C_n is simply $6.92 \times H_n$. The constants used to compute C_e are exactly 65% of those to compute C_d . This, presumably, reflects the fact that DDD costs in the evening are 65% of the day rates. The night time hourly cost is, approximately 40% of an average day rate, again paralleling DDD costs. Table 1 gives the WATS and DDD costs (estimates) for New York City. These are used in the computational experiments and examples presented in the rest of this paper.

Table 1A. WATS Line Daytime Cost (Hourly)

Band	1st 15 Hours	Next 15 Hours	Next 40 Hours	All Other Hours
1	15.13	13.46	11.79	9.98
2	17.32	15.42	13.52	11.43
3	18.14	16.15	14.15	11.97
4	18.89	16.81	14.74	12.47
5	19.91	17.71	15.53	13.13

Table 1B. WATS Line Evening Cost (Hourly)

Band	1st 15 Hours	Next 15 Hours	Next 40 Hours	All Other Hours
1	9.83	8.75	7.66	6.49
2	11.26	10.02	8.79	7.43
3	11.79	10.50	9.20	7.78
4	12.29	10.93	9.58	8.11
5	12.94	11.51	10.09	8.53

Table 1C. WATS Line Nighttime Cost (Hourly)

Band	All Hours
1	5.26
2	6.02
3	6.31
4	6.56
5	6.92

Table 1D. Daytime Average Cost From New York City

Band	DDD Cost (Hourly)
1	19.96
2	22.96
3	23.94
4	24.94
5	26.26

Thus, we see that WATS and DDD costs vary in similar ways. It will be useful for us to consider a model of how cost varies with band as it will allow us to make a quantitative trade-off between traffic in different bands and in particular to approach the problem of deciding on the size of each WATS group. Table 2 gives the relative costs in each of the bands as seen from New York City. These factors apply to all hourly costs; e.g., DDD costs, the first 15 hours of daytime WATS traffic, etc. The factors are exact for WATS costs and reasonable approximations for DDD costs. These factors are different for different states, but they exist for all states and are easily determined from the tariff.

Indeed, the entire WATS tariff may be modelled as a bulk discount on DDD. Given a vector, D_b , the average per hour DDD cost in Band b , the constants used to compute C_d , C_e , and C_n can be thought of as percentages of D_b . For example, if the average hourly daytime DDD cost for Band 5 calls from NY City were \$26.26 during the day, we could write C_d as

$$\begin{aligned} C_d = & 26.26 \times \{.758 \times \min(H_d, 15) \\ & + .674 \times [\min(H_d - 15, 25)] + \\ & + .591 \times [\min(H_d - 40, 40)] + \\ & + .5 \times (H_d - 80)\}. \end{aligned}$$

That is, the WATS tariff offers a discount of between 25% and 50% relative to DDD. The discount is, of course, relative to the 26.26 figure. If this figure were lower, the discount is, relatively, smaller. Based upon an assumed 4 minute holding time, this rate appears to be a reasonable estimate. Shorter holding times would give, comparatively, higher discounts.

Table 2. Relative Costs

Band	Factor
1	1.000
2	1.145
3	1.199
4	1.249
5	1.316

The significant thing is that the same model with the same discounts applies to all bands, all states, and all times of day.

The usage of a WATS line in a band is defined to be the average usage over the group. Thus, if there are M lines in a Band b WATS group and together they carry H hours of daytime traffic during a month, then $H_d = H/M$. A larger group will block less traffic and, hence carry more traffic. Thus, if we add a line to the group both H and M will increase. Since the efficiency of lines within a group decreases as the number of lines increases, however, H will increase more slowly than M and H_d will decrease. The cost/hour for traffic carried by the group will therefore increase as the size of the group increases. It will not, however, exceed the DDD cost until the \$27.50 fixed charge becomes a significant portion of the total cost of the line.

The choice of how to optimally route calls is itself an interesting problem involving trade-offs between cost and quality of service. In practice, a user's routing choices are limited by the capabilities of his local equipment (PABX) and may include overflow, queueing for lines in some bands, blocking, and even time-of-day sensitive routing. This is, however, beyond the scope of this paper. The simple assumption is made that a call is offered to the lowest numbered band capable of handling it and if no lines are available in this band then the call overflows to each higher numbered band and finally to DDD, if necessary.

Thus, the focus is on the problem of determining the optimal number of lines in each WATS band group. The larger the number of lines in a group, the more traffic the group carries and hence, the less traffic overflows to higher numbered groups and to DDD. The base cost of a line is very small, only \$27.50 per month. At first glance one might expect this would lead to solutions with many lines. However, as we will see in the following section, the tariff provides that the cost per hour decreases as the number of hours of usage increases and that the number of hours of usage is averaged over the lines in a band. Thus, adding a line to a group in a band will decrease the number of hours of usage per line and hence, the average cost per hour of carried calls increases. This limits the number of lines

desired in a band and makes the problem interesting

3. Optimization Procedure

The optimization procedure is implemented in FORTRAN and runs on a DEC 20 60. The software package includes an interactive front-end which permits the user to edit a WATS configuration, traffic, tariff, and parameter values. Finally, it includes a full spectrum of design and analysis modules which allow him to design configurations using a variety of options for queueing, blocking and overflowing traffic. For the sake of clarity of presentation, we will concentrate on the overflow version of the problem in the remainder of this paper. Thus, we will assume that traffic offered to a group overflows to each of the higher numbered groups and eventually to DDD, if necessary.

The procedure takes as input a description of the tariff as given above and a description of the traffic to be carried by the network. This traffic description may take one of several forms depending on what information is available. Ideally, the user will process billing information supplied by the local telephone company and get at least one month's data giving the actual number of minutes, calls and the actual DDD (Direct Distance Dial) cost over the given time period. Alternatively estimating procedures may be used if this information is not available. In either case, the user develops a traffic matrix which gives the number of erlangs of traffic* to each band. DDD cost per minute in each band, which is also an input to the program, may also be developed as an average derived from actual call records or alternatively may be derived from an average figure for the band. Usually, this will not be a critical issue as DDD costs do not differ widely from one part of a band to another.

If, however, a large amount of traffic is known to go to a location near the edge of the band, then the DDD cost of this traffic will not be typical for the band and should be computed using the actual cost of calling that location. Also, if the length of all calls, or calls to

certain bands is known to be unusually short, then the one minute minimum charge which is applicable for most long distance calls will significantly increase the DDD cost. In this case, one should obtain the actual number of calls so that a more accurate estimate of the true DDD cost can be computed.

The major outputs from the procedure are the number of lines in each group and the total WATS and DDD cost. Auxiliary reports are also produced, including the cost per band, the traffic offered to and carried by each band, details on an hour by hour basis and summaries of activity at major locations (when input supporting this is available). This is useful as part of more global optimization procedures which determine the placement of tie lines, FX (foreign exchange) lines, and tandem switches.

Figure 1 is a flowchart of the global optimization procedure. It begins by initializing the configuration to an all band 5 solution, i.e., no lines in bands 1 through 4 and a sufficient number of lines in band 5 to satisfy a given grade of service during the peak hour.* As will be seen, the procedure is not sensitive to the starting configuration used. In particular, we have not observed any significant variation in the running time of the procedure or the quality of the solution it produces when the starting configuration is changed.

The procedure is constrained to produce solutions with no fewer than LO_i lines and no more than HI_i lines in band i . The values of LO_i and HI_i , which are inputs to the procedure can be set in several ways. First, if one wishes simply to find an optimal solution, LO_i can be set to 0 and HI_i can be set to a large number (e.g., a number of lines sufficient to carry 99% of the load offered to band i and all lower bands). If the expansion of an already existing network or a local modification of a larger network (containing private tandem switches, FX lines, etc.) is being considered, LO_i and HI_i may be set to values very close to current values. Finally, by setting $LO_i = HI_i = V_i$, a specific configuration containing V_i lines per band can be evaluated.

*An erlang is a measure of traffic intensity. One erlang is equivalent to 60 minutes of call holding time per hour (i.e., sufficient traffic to occupy a single line full time).

*Grade of service is defined as the fraction of calls blocked (not carried) by the system.

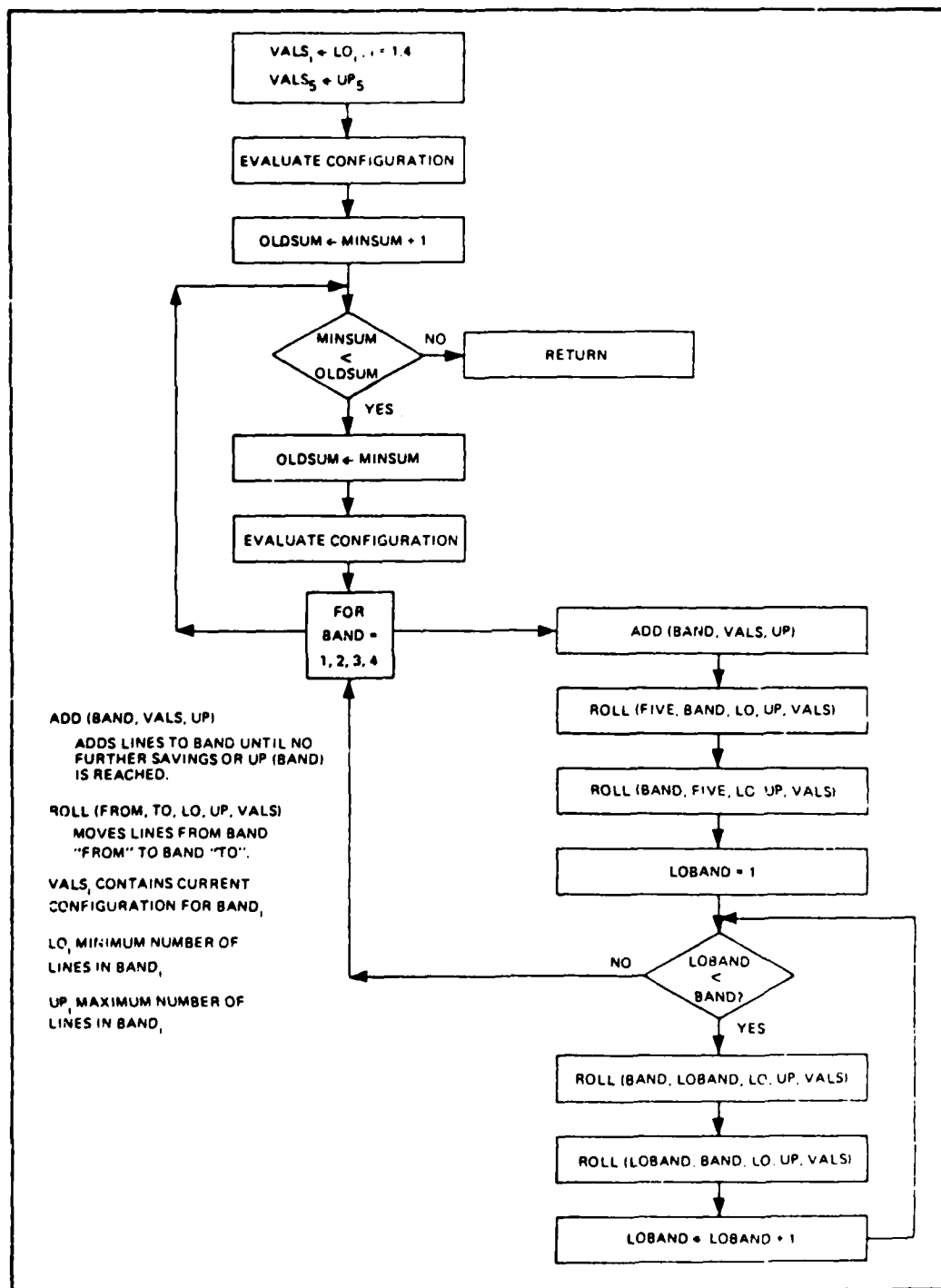


Fig. 1. OPTSUB.

The current solution being examined is maintained in the array VAL. Thus, VAL_{*i*} contains the number of lines in band *i* in the current solution. Local perturbations to the current solution are explored by incrementing and decrementing individual VAL_{*i*}.

The procedure EVAL takes VAL, the traffic requirements, and the tariff description as input. It then calculates the total amount of traffic carried by each WATS band (and by DDD) and costs out the configuration. In the case considered here, this is done by using the Erlang-B formula and allowing traffic to overflow freely from each band through the higher numbered bands and eventually to DDD, if necessary. In other versions of the procedure, overflow and blocking are computed on the basis of Rapp's approximation to Wilkinson's Equivalent Random Method [4,5] and Barrer's equations [6] for estimating overflow in networks with queueing. EVAL also compares the total cost of the current solution with MINSUM, the lowest cost for any configuration thus far evaluated. If the current cost is less than MINSUM, it replaces MINSUM and the current configuration is retained.

The routines ADD and ROLL explore local modifications to the current configuration and comprise the heart of the optimization procedure. ADD considers the addition of one or more lines to band BAND. It proceeds by adding single lines to the group in band BAND until it finds the cost of the configuration no longer decreases.

ROLL, whose flowchart is shown in Figure 2, proceeds along similar lines attempting to exchange lines between two bands, thus effectively rolling traffic from one band to another. ROLL begins by removing single lines from the group in band FROM until the cost increases. It then attempts to reduce the cost by adding a line to the group in band TO. Thus, ROLL considers both the possibility of removing one or more lines from band FROM and exchanging one or more lines in band FROM for the same number of lines in band TO. Finally, ROLL considers the possibility of exchanging one line in band FROM for two lines in band TO.

It should be noted that the function of ADD is actually contained in ROLL. ADD is used, however, because it is roughly three times as

fast as ROLL. ADD and ROLL are called repeatedly to improve the current solution via local modifications until no further progress is made. The best value prior to the current cycle of local modification is recorded in OLD. If after attempting all local modifications, MINSUM is still unchanged, the procedure terminates. A similar technique is used to terminate procedure ROLL.

An earlier version of this algorithm only considered ROLLing between adjacent bands. The procedure was not as much faster as one might have expected (typically, the difference was less than a factor of two) and the results obtained were sometimes slightly worse.

Note in Figure 1 that band 5 is treated somewhat differently from the other bands. Variable BAND only takes values between 1 and 4 and ADD and ROLL are called accordingly. For each value of BAND, the proper size of the band 5 group is first determined before other local modifications are attempted. This is because the tradeoffs between other bands are generally less dramatic than tradeoffs involving band 5. With other bands the WATS costs, which are generally similar, are involved. When band 5 is involved part of the tradeoff is with DDD, which is somewhat more significant.

In both the flowcharts in Figures 1 and 2, it can be seen that the algorithm proceeds in an orderly fashion to consider all reasonable local exchanges (i.e., the addition of a line in a band, the deletion of a line from a band, and the exchange of a line in one band for a line in another). The variable FIVE in Figure 1 is simply the constant five. The variable BAND takes the values one through four and is the current band being considered in a local exchange. LOBAND takes values between one and BAND-1 and is the other band involved in the current local exchange. At each stage, EVAL is called to evaluate the current configuration and update MINSUM if this configuration is the best yet encountered.

The significant feature about this approach is that only local changes, i.e., the adding or deleting of single lines, are considered and movement in a given direction is halted when no further progress is encountered. This greatly improves its running time as compared with a procedure considering a complete spectrum of alternatives, which would

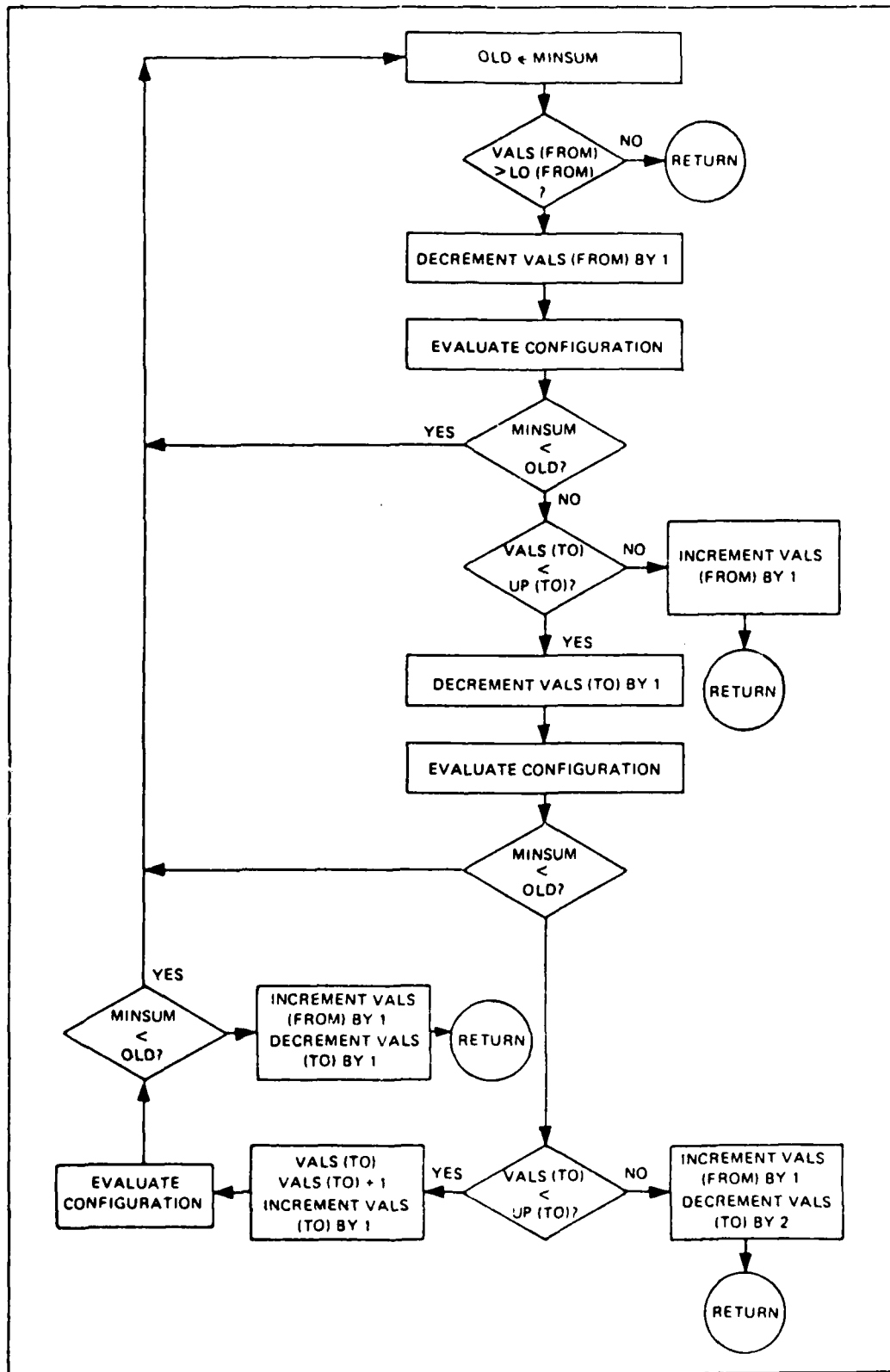


Fig. 2. ROLL (FROM, TO, LO, UP, VALS).

involve considering a number of alternatives equal to the product over i of $(H_i - L_i)$.

The running time of this procedure is essentially linear in the number of perturbations (attempts to move a line from one band to another) examined. The time to evaluate a perturbation is essentially linear in the number of lines in the network. The number of perturbations varies from one network to another, but appears roughly linear in the number of lines. The procedure ran for less than one minute for a network with a total of 75 lines in it.

Theoretically, in order to guarantee that such a procedure will converge to a global optimum, the function we seek to minimize should be unimodal, i.e., it should possess a unique minimum which is reached by descending along any sequence of local exchanges. In an absolute sense, the function we seek to minimize is not unimodal. However, we have found it to be nearly so in practice.

An argument can be made for the validity of the procedure based upon the average cost per hour in each group. The total offered load is fixed. In all cases—queueing, blocking, and overflow—the total carried load is also essentially fixed. Thus, to minimize total cost, one seeks to minimize the cost per hour of carried traffic. The procedure described above can be thought of as one which constantly seeks to move traffic from a group with a high cost per hour to one with a low cost per hour and stops when all alternatives have roughly the same incremental cost per hour.

Such a procedure would work perfectly, yielding an optimal solution every time, if as traffic was moved from one group to another, the cost per hour changed in a consistent direction. Unfortunately, in this case, it need not. As traffic is moved to a group, the group operates more efficiently and the cost per hour for traffic carried by the group goes down. This is good as it reinforces our decision to move traffic into the group. Unfortunately, as the traffic offered to a group increases, so does the amount that overflows from it. If the place it overflows to has a higher cost per hour, then it makes the decision to offer traffic less desirable. Furthermore, the group from which traffic is removed will become less efficient initially but if one then decreases the size of the group, it will become

more efficient. Thus, there are many factors working against one another and the net result is not, in general, clear. All this is compounded by the fact that we are working with different amounts of traffic in different hours and so the effect of a change is further confused.

If we examined the situation more closely, however, we will see that there is a cause for optimism. The actual procedure does not directly offer more or less traffic to the group; it either increases or decreases the size of the group. It does not calculate costs per hour; it calculates the actual cost of the network. Also, it makes the smallest possible changes, working with increments and decrements of a single line. Thus, it makes decisions based upon the total net effect of a change. If adding a line to a group decreases the total cost, it will keep adding lines to the group until the net effect reverses. The changes are made gradually so that the net effect can be watched as closely as possible. The entire process can be viewed as one of equalizing potentials in an electrical network or as rolling water among partitions within a container. The problem is that whenever you roll some water, somebody comes along and alters the height of the partitions.

If the costs per hour in two adjacent groups are nearly equal, there is little to be gained from rolling traffic between them. Once we get to this point, we can only make small mistakes. It is relatively easy to get to this point with the WATS groups since their costs per hour were close to begin with. The situation with respect to DDD is different. The whole point of using WATS is that it is significantly cheaper than DDD. The procedure takes this into account by only rolling traffic between groups which do not overflow directly to DDD and reoptimizing the last group, which does, directly. Thus, for example, if we are considering the effect of removing a line from the band 2 group, we will also reoptimize the band 5 group in the case where band 2 traffic overflows to all higher numbered groups. Recall that this "optimization" generally amounts to little more than checking if it pays to add a line to the band 5 group.

There is only one remaining cause for concern. Suppose we had a problem with identical traffic in each band and we began with a solution with equal sized groups in each band

It might turn out that it doesn't pay to increase or decrease the size of any group by a single line, but that a total consolidation into a properly sized band 5 group is the optimal solution. This is the general problem of an economy of scale existing but not manifesting itself until a critical size is reached. This has not turned out to be the case in any of the specific problems examined so far, but it is something to watch out for. One protection against this is to start the optimization off first with a balanced solution and then with an all band 5 solution. This is what is done.

4. Experimental Results

In order to better understand the optimization procedure and the nature of the solutions to the WATS optimization problem under the new tariff, a set of experiments was run. The purpose of these experiments was to test the behavior of the cost function directly. Thus, instead of simply running the optimization procedure, we selectively enumerated solutions in order to more thoroughly examine the solution space. Subsequently, we ran the optimization procedure and verified that it did indeed find the same solutions.

Table 3 gives the salient input data common to all experiments. The time-of-day profile is assumed to be identical for all bands (not a realistic assumption, but simple to input) and is given in Table 3A. Only daytime traffic is considered, again for simplicity. Table 1B gives the cost for WATS lines; it is taken from the tariff for New York City. The DDD costs, also estimates for New York City, are given in

Table 3. Time-of-Day Profile

Hour	Fraction
8	.02
9	.08
10	.12
11	.10
12	.14
1	.18
2	.16
3	.11
4	.06
5	.03

Table 1C. Table 1D gives the relative costs by band. One can observe that these factors relate all WATS costs and DDD costs. All experiments were run with 22 engineering days/month.

The first experiment had 1 Erlang/day of traffic in each band. Table 4 summarizes some of the configurations explored. The configurations in Table 4 are the number of lines in each band. Commas between the numbers are omitted when they are single-digit numbers. First note that the pure band 5 solutions are unimodal; i.e., they decrease in cost initially and then steadily increase in cost as the number of lines increases. This is not particularly surprising. It is, however, significant as an incremental procedure like the one described here would not work if the cost were not unimodal.

Starting from the best pure band 5 solution—00002 (Configuration 3), the best solution (#9) is reachable through local transformations via the configurations 3, 6, and 9. Suppose, however, that after Configuration 6, Configu-

Table 4. 1 Erlang/Band

Number	Config.	Cost	Bands					D
			1	2	3	4	5	
1	00000	\$2,889	0	0	0	0	0	2809
2	00001	2,297	0	0	0	0	1217	1080
3	00002	2,133	0	0	0	0	1816	317
4	00003	2,153	0	0	0	0	2080	73
5	00004	2,194	0	0	0	0	2181	14
6	00011	2,075	0	0	0	1017	745	313
7	00101	2,100	0	0	822	0	853	425
8	00111	2,042	0	0	822	596	506	118
9	10011	2,041	316	0	0	873	640	213
10	11111	2,074	316	386	410	428	450	84
11	11011	2,053	316	386	0	683	534	133
12	10111	2,043	316	0	652	506	473	97

ration 8 was reached. This would happen if we tried adding a line in band 3 before adding a line in band 1. Configuration 9 could still be reached via the path 8, 9, i.e. by rolling one line from band 3 to band 1. Configuration 9 could also be reached from a balanced solution like Configuration 10 through the path 10, 11, and 9. Thus we see that Configuration 9 is reachable in a variety of ways. This is encouraging. We do not know if Configuration 9 is optimal.

While we cannot be sure that the procedure will, in fact, find the optimum, the data in Table 4 is very encouraging. First, we see that the paths to the best solution do exist. Second, we see that the solution space is very flat. Most dead ends, if any exist, are likely to be within 1% of the optimum.

Similar conclusions can be drawn from Table 5A, which is the result of an experiment with 10 Erlangs per day per band. Here again, the best solution, (#17), is reachable from the band 5 solution and balanced configuration (#15). Here again, both the band 5 and balanced solutions, and hence, any dead ends reachable from them, are within 1% of the optimum. The carried load and cost/hour are given in Table 5B for Configuration 17. They indicate an increasing cost/hour as the band increases with the most substantial jump between band 5 and DDD. Other experiments with 3 Erlangs/day/band and 30 Erlangs/day/band were run. The results were similar to those reported here.

Table 5A. 10 Erlang/Band

Number	Configuration	Cost
1	00000	\$28.886
2	00005	21.354
3	0000.10	18.858
4	0000.11	18.851
5	0000.12	18.940
6	00066	17.991
7	00336	17.740
8	01236	17.670
9	01245	17.618
10	10236	17.582
11	11136	17.582
12	11226	17.526
13	11235	17.488
14	22222	18.271
15	22225	17.511
16	21226	17.477
17	21235	17.461

Table 5B. Carried Load and Cost/Hour

Band	Carried Load	Cost	Cost/hour
1	7.47	\$ 2,169	13.20
2	5.04	1,565	14.11
3	9.95	3,242	14.81
4	12.74	4,463	15.92
5	13.84	5,467	18.28
DDD	.96	556	26.11
TOTAL	50.00	\$17,461	

The procedure was run on several actual problems using data derived from actual call tapes. An experienced designer then attempted to improve the solution manually using the interactive system within which the procedure was embedded. In two cases an enumeration was run, automatically stepping through all possible solutions. In no case was any improvement observed. Thus, while we have no formal proof that the procedure converges to an optimal solution, this empirical evidence, together with the essentially unimodal nature of the cost function, leads us to believe that it does in fact yield globally optimal solutions in most cases.

5. Extensions and Further Conclusions

The multiple hour nature of the problem does not seem to have much of an effect on the procedure. If an off-peak hour has traffic similar to the peak hour, the optimization should be unaffected. If an off-peak hour has much less traffic, most of its traffic will be carried in-band. Neither situation seems to cause a problem.

FX and tie lines could be handled by increasing the number of rows in the traffic matrix to include candidate FX and tie lines. Similarly, the procedure could be extended to handle a more general class of overflow rules where traffic from one band is restricted to overflow only to specific other bands. In some cases, where the number of overflow options is limited, an explicit routing table would have to be entered.

Queueing and blocking solutions could be produced by replacing the Erlang B formula by one appropriate to the situation. The conclusions drawn from the experiments above are principally results of the cost function, not

the Erlang B function. They are thus likely to hold in the other situations as well.

References

1. AT&T Tariff 259 (effective June 1, 1981)
2. R. Mina, "Introduction to Teletraffic Engineering," Telephony, 1974
3. Jewett, "Designing Optimal Voice Networks for Business, Government and Telecommunications," Telephony, 1980
4. R. I. Wilkinson, "Theories for Toll Traffic Engineering in the U.S.A.," *BSTJ* Volume 35, pp. 412-514, 1956
5. Y. Rapp, "Planning of Junction Networks in a Multi-Exchange Area," *Ericson Technics* No. 1, 1964
6. D. Y. Barrer, "Queueing with Impatient Customers and Ordered Service," *Operations Research*, Volume 5, 1957

About the Authors



Aaron Kershenbaum was born in Brooklyn, New York on October 9, 1948. He received the B.S.E.E. and M.S.E.E. simultaneously in 1970 from the Polytechnic Institute of Brooklyn, Brooklyn, New York. In 1976, he received the Ph.D.E.E. from that same institution. He is currently an Associate Professor in the Department of Electrical Engi-

neering and Computer Science at the Polytechnic Institute of New York, Brooklyn, New York. His major areas of research are computer communications and the design and analysis of algorithms. From 1969 to August 1978, he was with Network Analysis Corporation, Great Neck, New York and was Vice President for Software Development. While at NAC, he developed many of the algorithms and much of the program packaging used there for the analysis and design of large scale computer communications networks.



Zvi E. Kozicki, as Manager of Applications Programming for Network Analysis Corporation, has major responsibilities in the development of computer-aided network analysis and design tools. In this capacity, Mr. Kozicki has technical responsibility for NAC's network design services (NDS) including MIND (Modular Interactive Network Designer) and is involved in ongoing research to support and develop new tools. He is also currently developing a simulation model to evaluate a major military communications system (JTIDS). In the past, Mr. Kozicki applied his talents in a variety of projects including the development of a comprehensive telecommunications data base system for the Department of Agriculture, the design and implementation of computer-aided design software for cable television systems, and the development of an analysis tool to design and manage an international Telex system. He received his MS (CS) in 1976 and his B.S. (Math)—Magna Cum Laude—in 1973, from the City College of New York.

D.6 Design of Survivable Circuit-Switched Communication
Networks

Natarajan, Walters, and Maglaris

IEEE MILCOM, September 1982, Boston

DESIGN OF SURVIVABLE CIRCUIT-SWITCHED COMMUNICATION NETWORKS

Kadathur S. Natarajan - ConTel Information Systems, Inc.
David H. Walters - ConTel Information Systems, Inc.
Basil Maglaris - Polytechnic Institute of New York

ConTel Information Systems, Inc.
130 Steamboat Road
Great Neck, NY 11024

ABSTRACT

We address the problem of designing a circuit-switched network for voice communications operating in a military environment. The circuit-switched network design problem may be briefly stated as: given the topology, route tables and control discipline, end-to-end offered traffic and performance requirements, determine trunk group sizes such that the requirements are satisfied. One of the key requirements of a design is that the network be survivable, where survivability is based on different destruction scenario conditions. Our objective is to guarantee an acceptable level of performance for every node pair and under each of the different anticipated damage scenarios. The main contributions of our present work are the development of approaches for designing networks that simultaneously satisfy performance requirements for different destruction scenarios.

We describe the architecture of a survivable, circuit-switched network. The key characteristics of the survivable network design problem are highlighted and differences with respect to classical trunk sizing problem are pointed out. An important aspect of our work is that the sizing is based on the logical topology of the network rather than its trunk group topology. One design approach, which we have used successfully, is presented in detail.

1. INTRODUCTION

In this paper, we address the problem of designing a circuit-switched network for voice communications operating in a military environment. The circuit-switched network design problem may be briefly stated as: given the topology, route tables and control discipline, end-to-end offered traffic and performance requirements, determine trunk group sizes such that the requirements are satisfied. One of the key requirements of a design is that the network be survivable, where survivability is based on different destruction scenario conditions. Survivability constraints on a network design include: a) connectivity requirements, and b) end-to-end performance requirements.

Satisfying connectivity requirements is a classical and relatively well studied problem. The main contributions of our present work are the development of approaches for designing networks that simultaneously satisfy performance requirements for different destruction scenarios. Our objective is to guarantee an acceptable level of performance for every node pair and under each of the different anticipated damage scenarios. This objective is more general, and in our opinion more appropriate, than the goal of ensuring connectivity

requirements. We avoid the pitfall of designing networks that satisfy connectivity constraints, but exhibit unacceptably poor performance. An important aspect of our work is that the design is based on the logical topology of the network rather than its trunk group topology.

The optimal design of circuit-switched networks for voice communications is complicated due to a variety of reasons. In order to appreciate this fact, we first review the salient characteristics of military networks that distinguish it from commercial telephony. They typically have a mesh connected topology (switch interconnections) and use non-hierarchical alternate routing schemes. Other important operational features that are unique to military networks include the presence of multiple levels of traffic priority, preemption of calls, provision for multiple route search methods (normal, preemptive, and hotline). The main End-To-End (ETE) performance measures of interest to a network designer are: Grade Of Service (GOS), Probability Of Preemption (POP), and Speed Of Setup (SOS) of calls.

In this paper, we focus attention on effective design of networks that satisfy ETE GOS performance requirements for multiple scenarios (prespecified by the user). In the next section, we introduce relevant terminology and explain the topological representation of the network. In Section 3, the key characteristics of the survivable network design problem are pointed out. In the final section, we develop approaches for designing survivable networks. One such approach, which we have used successfully, is presented in detail.

It is worth noting that recent literature [1] and [2] contain approaches to the dimensioning of networks for more than one busy hour. These multi-hour sizing algorithms address a problem conceptually similar to ours, in the sense that requirements of each busy hour may be thought of as a scenario. However, the methods are largely inapplicable to us because they consider networks (commercial telephony) with hierarchical routing, while our interest is in dimensioning networks with non-hierarchical routing.

2. TOPOLOGICAL REPRESENTATION

In this section, we fix relevant terminology and concepts. We also elaborate briefly on the architecture of a typical survivable voice communication network and set the framework for subsequent sections of the paper. Our discussion will focus on the backbone non-hierarchical switched part of the overall communications system. Users are allowed to be connected via dedicated local loops to the switch sites and routed via the backbone switched network to the destination handset.

The backbone topology of a survivable network is dealt at three levels of detail. These are: 1) physical topology, 2) logical topology, and 3) trunk group topology. A physical topology is comprised of a set of physical sites (e.g., switch sites, radio relay stations), and physical links (e.g., microwave links) as depicted in Figure 1. The physical elements (sites and links) are the targets of destruction under enemy attack.

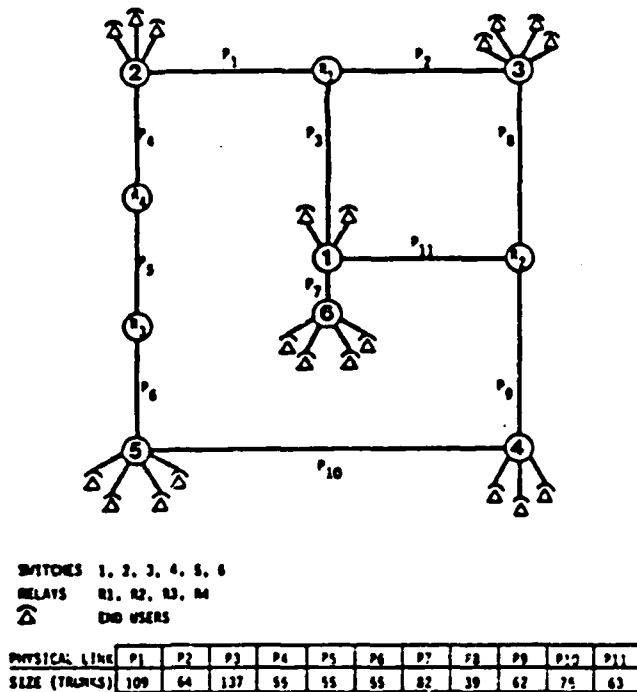
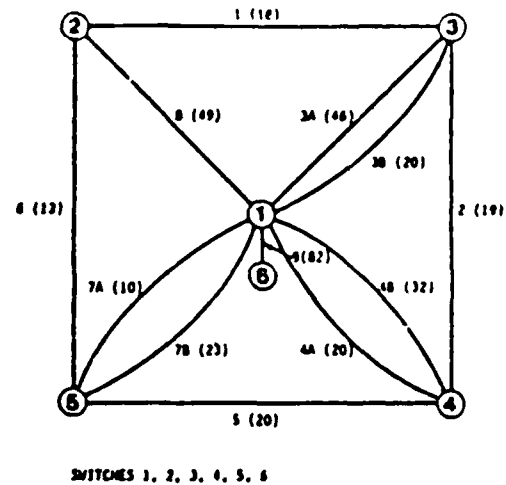


FIGURE 1: PHYSICAL TOPOLOGY

A logical link in a network is a path in the physical network between two switch sites without intervening switching. A logical link is considered operational (i.e., not destroyed) if, and only if, all the physical elements composing it are operational. A pair of switches may be connected by more than one logical link. If such is the case, the logical links represent conceptually different communication paths between the two switches. If multiple logical links exist between a switch pair, there may be physical elements (other than the terminal switch sites) common to two or more of the logical links. The logical topology of a network consists of a set of switch sites and logical links connecting site pairs.

In Figure 2, we show a logical topology of the network whose physical topology is given in Figure 1. There are several potential logical topologies one can define with respect to the given physical network topology, and Figure 2 shows just one of the choices.

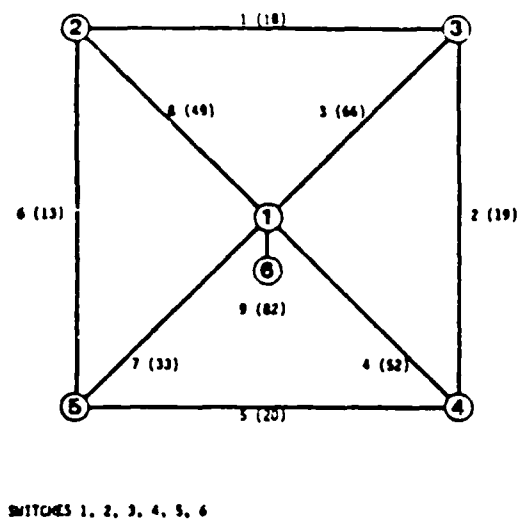


LOGICAL LINK	SIZE	LOGICAL LINK COMPOSITION
1	18	2 - R ₁ -3
2	19	3 - R ₂ -4
3A	46	1 - R ₁ -3
3B	20	1 - R ₂ -3
4A	20	1 - R ₂ -4
4B	32	1 - R ₁ -2-R ₄ -R ₃ -5-4
5	20	4 - 5
6	13	2 - R ₄ -R ₃ -5
7A	10	1 - R ₁ -2-R ₄ -R ₃ -5
7B	23	1 - R ₂ -4-5
8	49	1 - R ₁ -2
9	82	1 - 6

FIGURE 2: LOGICAL TOPOLOGY AND LOGICAL LINK COMPOSITION

A trunk group between two switch sites consists of all the logical links that connect the site pair. If no logical link connects a site pair, then there exists no trunk group between the site pair. The trunk group topology of a network consists of a set of switch sites and trunk groups connecting site pairs as shown in Figure 3. A route between a switch pair is specified as a sequence of trunk groups.

Connectivity requirements on a design stipulate that for each switch pair, there must be at least a prescribed number of physically disjoint paths in the physical topology connecting the switches. Connectivity constraints are topological in nature and are taken into account in selecting the logical topology and routes for different switch pairs.



TRUNK GROUP SIZE (TRUNKS)	1	2	3	4	5	6	7	8	9
	18	19	66	52	20	13	33	49	82

FIGURE 3: TRUNK GROUP TOPOLOGY

Associated with each physical link is its maximum capacity. It is expressed as the number of telephone calls that the link can transmit at the same time. The size (s_i) of logical link L_i is the number of trunks it contains, and cannot exceed the capacity of minimum of component physical links in it. The size of a trunk group is the sum of the sizes of logical links that comprise it. Figures 1 through 3 indicate the sizes of a hypothetical design.

The cost c_i (dollars/month) of logical link L_i consists of two components: 1) a fixed cost (b_i) for establishing the logical link, and 2) a variable cost which is proportional to its size. If the cost per circuit in L_i is a_i , the total cost c_i is equal to $(b_i + a_i s_i)$. The specific choice of a logical topology determines the fixed costs of the network. The total cost of the network is the sum of its fixed costs and variable costs. The design problem, considered in the subsequent sections, is concerned with the determination of a feasible set of logical link sizes such that the variable costs (i.e., trunk costs) are minimized. Although not addressed in this paper, the designer interested in minimizing the total cost of the network must consider the choice of logical topology as part of the overall design problem.

3. DESIGN PROBLEM

The classical network design problem may be stated as follows. Given the trunk group topology, offered ETE traffic and performance (GOS) requirements, and routes and route control discipline, determine a set of trunk group sizes such that traffic and performance

requirements are satisfied. The above problem, (henceforth referred to in this paper as the Trunk Sizing Problem) requires the sizing of trunk groups for a single scenario. A scenario is an operating condition of the network characterized by its unique set of traffic and performance requirements.

The classical Trunk Sizing Problem in networks with non-hierarchical routing has been well studied in the literature [3] and [4]. Segal's [3] algorithm is based on single-moment assumption (i.e., calls offered to each trunk group have a Poisson distribution) and is applicable for networks with either originating office control or successive office control. Covo [4] considered the presence of multiple precedence levels of traffic and developed an approximate method for sizing trunk groups. However, Covo's work addresses the requirements of only a single scenario. For our computational studies, we used Segal's algorithm for feasible trunk group sizing. We do not repeat Segal's algorithm here, but refer the reader to [3] for details.

The problem of designing a survivable network (henceforth referred to as SND Problem) requires a sizing of trunk groups and logical links that satisfies the traffic and performance requirements under multiple operating conditions of the network. This is the most important characteristic that distinguishes the problem from the Trunk Sizing Problem. Another unique characteristic of the SND problem is the dichotomy between trunk group sizing and logical link sizing. The key aspects of this distinction between trunk group sizing and logical link sizing are as follows. In classical studies, a physical link is synonymous with a trunk group and destruction of a physical link or a switch would result in the total destruction of any route that uses it. However, for a survivable network, the effect of destroying a physical element (link or site) has the following ramifications. All logical links that use the physical element are destroyed. The loss of a logical link would result in reducing the size of the trunk group that contains the logical link. Thus, the size(s) of one or more trunk group(s) will be reduced. The effect on a route affected by the physical element destruction is one of the following: 1) a reduction in size of some trunk group(s) belonging to the route, or 2) total destruction of the route. The latter occurs when all the logical links composing any trunk group belonging to the route are destroyed. Unlike the classical Trunk Sizing Problem, the loss of a physical element does not necessarily imply the total destruction of one or more routes.

We illustrate the above facts with an example. Consider the network whose logical topology and logical link composition are shown in Figure 2. The impact of destroying relay site R_3 (see Figure 1) would be to destroy logical links L_{4B} , L_6 and L_{7A} that use R_3 (see logical link composition in Figure 2). These, in turn, imply a reduction in size of trunk group T_4 (containing L_{4B}) from 52 to 20. It can be observed that size of trunk group T_7 (containing L_{7A}) is reduced from 33 to 23, while trunk group T_6 (containing a single logical link L_6) is totally destroyed (size reduced to zero). Thus the only routes totally disrupted would be those containing T_6 , while the rest of the routes are likely to operate at a degraded performance level.

For a given scenario, the ETE GOS performance of a network is a function of the trunk group topology, trunk

group sizes, routes, and ETE offered traffic of that scenario. However, in designing a survivable network, we have to size the logical links such that for each scenario, the trunk group sizes (as determined by the surviving logical links) are adequate to meet the requirements of the scenario.

The specification of the design problem must precisely characterize the scenarios of interest. As an example, our interest is in sizing a network with the following requirements. Under a normal scenario in which no network elements are destroyed, the network must be capable of satisfying a nominal set of offered traffic and performance requirements. The performance requirement is $G_{\max}$, the maximum tolerable end-to-end GOS. The value of $G_{\max}$ could be 0.01. Under an overload scenario, no network elements are destroyed, but the offered traffic between each node pair increases to some c (say, $c = 1.5$) times the nominal offered traffic between the pair. The network sizing must satisfy the overload traffic requirement at a less stringent performance level (say, $G_{\max} = 0.10$). In addition, we require that whenever exactly one physical element is destroyed, the surviving network must be capable of handling the nominal traffic requirements at a degraded performance level (say, $G_{\max} = 0.15$). We note that for the physical network shown in Figure 1, there are 21 physical elements (6 switches, 4 relays, and 11 links) and corresponding to each element a damage scenario is identified.

In the next section, we discuss two basic design philosophies for realizing feasible and near minimum cost survivable network designs.

4. DESIGN APPROACHES

We consider two basic philosophies for sizing survivable networks. We assume the following are given - physical topology, logical topology and logical link composition (under normal scenario), routes, physical link sizes, a precise characterization of the scenarios and their requirements. The objective is to compute logical link sizes such that the requirements of each scenario are satisfied. Two basic design approaches we considered are:

1. Parallel Approaches.
2. Serial Approaches.

4.1 Parallel Approaches

Suppose a network requires sizing of logical links and trunk groups to satisfy traffic and performance requirements for s different scenarios. The basic philosophy of a parallel approach is that a feasible survivable design may be obtained by independent solutions of many instances of the trunk sizing problem (one instance for each scenario) and then an appropriate combination of the solutions to determine the logical link sizes. Thus, the two major steps in the parallel approach are:

- Step 1: Determining Trunk Group Size Requirements.
- Step 2: Assign Capacity to Logical Links to Satisfy Trunk Group Size Requirements.

Step 1 is accomplished as follows. For each scenario j

($1 \leq j \leq s$), identify the trunk group topology, surviving routes, ETE offered traffic, and performance requirements. Apply Segal's algorithm [3] and compute a feasible set of trunk group sizes satisfying the ETE requirements of scenario j . Let a_{ij} be the desired size for trunk group i ($1 \leq i \leq t$, where t is the total number of trunk groups under the normal scenario) in scenario j .

Step 2 requires the calculation of logical link sizes based on the trunk group sizes (computed in Step 1) so that the total monthly dollar costs are minimized.

We devote the rest of this Section (4.1) to address Step 2. Consider a specific trunk group, say i . Let the trunk group contain m logical links and let n_{ik} represent the number of circuits in logical link k of trunk group i . Further, let X_{jik} be a binary variable defined as follows:

$$X_{jik} = \begin{cases} 0 & \text{if logical link } k \text{ of trunk group } i \text{ is} \\ & \text{destroyed in scenario } j \\ 1 & (1 \leq j \leq s, 1 \leq k \leq m) \\ & \text{otherwise} \end{cases}$$

Let b_{ik} be the cost per circuit in logical link k in the trunk group i . The minimum cost logical link assignment (MCLLA) problem is stated as follows:

$$\text{Minimize: } \sum_{i=1}^t \sum_{k=1}^m b_{ik} n_{ik} \quad (1)$$

$$\text{Subject to: } \sum_{k=1}^m X_{jik} n_{ik} \geq a_{ij}, 1 \leq j \leq s, 1 \leq i \leq t \quad (2)$$

$$l_{ik} \leq n_{ik} \leq u_{ik}, 1 \leq k \leq m \quad (3)$$

The objective function (1) represents the total cost of circuits in all logical links of the given logical topology. There are s linear inequalities in (2) for each trunk group i , and their interpretation is as follows. For scenario j and trunk group i , the total number of circuits that survive (given by the left side of the inequality) must be at least equal to a_{ij} , the desired size for trunk group i under scenario j . The constraints on the design problem may mandate a minimum size requirement (l_{ik}), and maximum size limit (u_{ik}) for each logical link k . These constraints are given by (3).

In addition to constraints (2) and (3) that are specific to trunk group i , there are constraints on the number of circuits that can be used in a physical link. If a number of logical links (possibly belonging to many different trunk groups) use a physical link, the combined circuit requirement of the logical links using the physical link cannot exceed the maximum specified limit for the physical link. These constraints must be handled by the logical link assignment algorithm.

The MCLLA problem formulated above could be approached via an integer linear programming (ILP) algorithm. However, rather than applying a computationally expensive ILP algorithm, we have developed the following heuristic algorithm to achieve a (near) minimum cost design.

Our approach is based on decomposition of objective function (1) into t separable components (one for each trunk group) and minimizing each component individually. The t trunk groups are assigned to their respective logical links by considering them sequentially. Suppose $e_1, e_2, \dots, e_t$ is a sequence in which the trunk groups are considered for logical link assignment. When the logical links of trunk group e_i ($1 \leq i \leq t$) are assigned, the sizing decisions made earlier for logical links of trunk groups $e_1, e_2, \dots, e_{i-1}$ will be treated as constraints. These constraints arise due to the fact that (i) physical links may be shared by logical links belonging to different trunk groups and (ii) upper bounds on size of physical links.

An outline of the steps of the MCLLA assignment algorithm is given below.

Step a: Choose a sequence in which trunk groups are to be considered for logical link assignment. Without loss of generality, let this sequence be $1, 2, \dots, t$.

Step b: $i \leftarrow 1$.

Step c: Assign circuits to logical links of trunk group i .

Step d: $i \leftarrow i + 1$.

Step e: If $i > t$ then stop; else repeat Step c.

MCLLA Algorithm

Step a of the MCLLA algorithm is accomplished as follows. There is a large number ($t!$) of possible sequence, in which the trunk groups could be considered for logical link assignment. Since the final solution (total cost of logical link circuits) depends on the sequence in which trunk groups are considered, we consider trunk groups in a sequence determined by decreasing order of their size requirement. For each trunk group i ($1 \leq i \leq t$), compute the variable h_i (highest requirement of trunk group i over all scenarios).

$$h_i = \max_{1 \leq j \leq s} \{a_{ij}\}$$

Sort the h_i 's in non-increasing order. Consider the trunk groups for logical link assignment in the sequence determined by the above sort step.

We now consider Step c in the MCLLA algorithm and describe how the algorithm works in meeting the size requirements of a specific trunk group i . The algorithm first considers scenarios in which only one logical link of trunk group i survives, then it considers scenarios in which two logical links survive, and so forth. In assigning circuits to fulfill the requirements of scenario j , suppose there is a choice to assign circuits among two or more logical links. The assignment algorithm assigns circuits first to the least expensive (per circuit cost) logical link, L_1 in that scenario. The assignment is made until either of the following two conditions becomes true:

1. All the circuit requirements of scenario j have been fulfilled.

2. The maximum limit on the capacity of some physical link p in logical link L_1 is reached or the upper bound on the size of L_1 is reached.

If condition (1) becomes true, then we are done with the assignment meeting the requirements of scenario j . If condition (2) becomes true, then the remaining circuit requirements of scenario j are fulfilled by considering the second least expensive logical link L_2 that survives in scenario j . If necessary, the algorithm also considers the third least expensive logical link, and so forth.

For the purpose of illustration of Step c, consider the following example. Let a trunk group have $m (=4)$ logical links which must be sized to meet the trunk group's requirements in $s (=5)$ scenarios. Let the required sizes of the trunk group be: $a_1 = 59, a_2 = 57, a_3 = 61, a_4 = 40, a_5 = 85$. Let the cost per circuit in the logical links be: $b_1 = 10, b_2 = 20, b_3 = 8, b_4 = 12$. Let the upper bound for each logical link be 50 and the lower bound for each be 0. Let X , shown in Table 1, be a 5×4 matrix such that the $(j,k)^{th}$ entry is 1 (or 0) depending on whether logical link k ($1 \leq k \leq 4$) survives (or fails) in scenario j ($1 \leq j \leq 5$). The logical link sizes n_k ($1 \leq k \leq 4$) are computed as follows:

TABLE 1: MATRIX X

SCENARIO J	LOGICAL LINK K			
	1	2	3	4
1	1	0	1	0
2	0	1	1	0
3	1	0	0	1
4	0	0	0	1
5	1	1	1	1

First, scenario 4, with only one surviving logical link (L_4) is considered. Its size, n_4 , is set to 40, the required trunk group size in scenario 4. Of the remaining scenarios, scenarios 1, 2, and 3 have two surviving logical links each and scenario 5 has all four logical links in fact. The constraint on scenario 1 is that logical links L_1 and L_3 be sized such that their combined size ($n_1 + n_3$) is ≥ 59 . Since L_3 is cheaper than L_1 , circuits are assigned to L_3 without violating the upper bound constraint on its size. Hence, n_3 is set at the maximum permissible value of 50. Since this falls short of the required size by 9 circuits, the value of n_1 is set to 9. For scenario 2, ($n_2 + n_3$) must be ≥ 57 . The algorithm first considers L_3 which is cheaper than L_2 . Since L_3 is at its maximum permissible size, circuits are assigned to the second least expensive logical link, L_2 . n_2 is set to 7. The algorithm proceeds in a similar manner and ends up with the following logical link sizes:

$$n_1 = 21, n_2 = 7, n_3 = 50, n_4 = 40.$$

4.2 Serial Approaches

The basic philosophy of a serial approach is that a survivable network design may be obtained by considering the requirements of each scenario in sequence. The initial goal is to identify the scenario whose requirements are the most dominant (according to a criterion, well-defined by the designer) over all the scenarios. The trunk groups are sized and the required trunk group circuits are apportioned to the logical links that survive in the most dominant scenario. When the requirements of a non-dominant scenario j are considered, a baseline design is first derived from a knowledge of sizes of logical links dimensioned prior to scenario j . The requirements of scenario j are sought to be fulfilled by incremental modifications with respect to the baseline design.

Despite the intuitive appeal of the serial approach in designing networks with nearly "similar" scenarios, there are a number of disadvantages to it. Our assessment is that the accuracy of the serial approach is quite sensitive to the sequence in which scenarios are considered. Furthermore, the accuracy depends significantly on the use of effective algorithms for incremental sizing (from one scenario to the next). Effective algorithms for marginal sizing are known in the context of hierarchical networks [5], though not as much is known for the non-hierarchical routing networks of interest to us. A third and very significant disadvantage of the serial approach is its extremely high computational time requirement when compared to the parallel approach. When a baseline design is incrementally modified to meet the requirements of a new scenario, the ETE grade-of-service must be computed to verify if indeed the requirements have been satisfied. Thus the serial approach requires many computationally expensive performance analysis runs as part of its incremental sizing steps. From our computational experience, we have found the parallel approach to yield good feasible designs within reasonable computer time.

REFERENCES

1. Eisenberg, M., "Engineering Traffic Networks for More Than One Busy Hour," B.S.T.J., 56, No. 1, January 1977, pp. 1-20.
2. Elsner, B., "A Descent Algorithm for the Multihour Sizing of Traffic Networks," B.S.T.J., 56, No. 8, October 1977, pp. 1405-1430.
3. Segal, M., "Traffic Engineering of Communication Networks With a General Class of Routing Schemes," Fourth International Teletraffic Congress, 1964.
4. Covo, A.A., "Sizing of Military Circuit-Switched Communication Networks by Computer-Aided Analytic Methods," Proceedings of International Conference on Communications, Vol. 2, 1973, pp. 45.21-45.23.
5. Truitt, C.J., "Traffic Engineering Techniques for Determining Trunk Requirements in Alternate Routing Trunk Networks," B.S.T.J., Vol. 33, No. 2, March 1954, pp. 277-302.

E. Switching Techniques

E.1 Delay and Overhead in the Encoding of Data Sources

Hayes and Boorstyn

IEEE Transactions on Communications, November 1981

Delay and Overhead in the Encoding of Data Sources

J. F. HAYES, SENIOR MEMBER, IEEE, AND R. R. BOORSTYN, MEMBER, IEEE

Abstract—A basic property of data sources in interactive applications is burstiness, i.e., short periods of activity followed by long idle periods. In these same applications message delay is the primary performance criterion. The combination of bursty flow and a delay criterion leads to a source encoding problem in which delay plays a central role. A salient feature of this problem is that there is a tradeoff between delay and the number of protocol bits required to represent the state of the source.

A simple model of a bursty source is studied with the objective of understanding the relationship between coding efficiency and delay. Two encoding schemes, a block encoding technique and a technique employing flags, are examined in some detail. For both the block encoding and the flag schemes, a significant result is that as the source becomes less bursty, delay grows without bound. This result is obtained in spite of the fact that both schemes are reasonable and in the limit the encoding problem disappears. It also appears that the flag encoding technique has much smaller delay than block encoding.

INTRODUCTION

THIS paper is concerned with the interplay of delay and overhead in the transmission of data traffic over a synchronous channel. The relationship of these quantities is due to certain unique properties of data in an interactive environment. Data traffic is bursty, i.e., there are short busy periods interspersed with relatively long idle periods which contain no information. (This stands in contrast to voice and video, where pauses or blanks may be significant.) The second relevant property of data is the effect of delay. In many applications data traffic is highly interactive, placing a premium on rapid response times. The relationship of these quantities allows one to formulate a source encoding problem which considers delay as a basic component.¹

Clearly, the long-term average rate of data generation at the source must be less than the capacity of the synchronous channel. However, long-term averaging will lead to a violation of the delay requirement and a reduced encoding interval is indicated. Short-term encoding leads to increased overhead since there will be times when there is nothing to transmit: a source state which must be conveyed to the receiver unambiguously. In the sequel the problem of striking a balance between overhead and delay is studied for a basic source model.

Paper approved by the Editor for Computer Communication of the IEEE Communications Society for publication after presentation at the International Communications Conference, Seattle, WA, June 1980. Manuscript received July 20, 1980; revised March 16, 1981. This work was supported in part by the National Science and Engineering Research Council of Canada under Grant A0901 and by the U.S. Army CORADCOM under Task B-9-3513 of the Postdoctoral Program, RADCOM. Portions of this work were performed while the authors were at Bell Laboratories, Holmdel, NJ.

J. F. Hayes is the Department of Electrical Engineering, McGill University, Montreal, P.Q., Canada H3A 2A7.

R. R. Boorstyn is with the Department of Electrical Engineering, Polytechnic Institute of New York, Brooklyn, NY 11201.

¹ The connection of these two quantities has been discussed by Lam [1] in another context.

To a large extent the role of delay in this context has not been considered by information theorists. The earliest related work was done by Jelinek [2] and was continued in collaboration with Schneider [3], [4]. The model we consider is depicted in Fig. 1. A source produces symbols at a fixed rate. If the symbols produced by the source are not equally probable, then by using variable length codes, such as Huffman codes, the average number of bits that need to be transmitted over the channel in a time interval can be minimized. In order to implement this technique buffering is required since the channel operates at a fixed rate and the bits are produced by the encoder at a variable rate. Jelinek and Schneider studied the relationship between the probability of buffer overflow and efficient source encoding.

We shall consider a simple model of a source which embodies these basic quantities. Several techniques for encoding this source are studied with the focus upon delay rather than storage. A particular technique using flags is examined in detail. Related work on the topic has been done by Gallager [5], who applied rate distortion theory to the coding problem with delay as the distortion measure. Recently Humblet [6] has considered the problem of encoding randomly arriving messages so as to minimize delay.

SOURCE MODEL

Consider the following model: a source produces either a blank or a binary digit once every second with the following probabilities.

$$\begin{aligned} P[\text{blank}] &= 1 - P \\ P[0 \text{ output}] &= P/2 \\ P[1 \text{ output}] &= P/2. \end{aligned} \quad (1)$$

Successive symbols are independent. The sequence of 0's and 1's produced by this source is to be carried over a synchronous error-free binary channel with a capacity of one bit per second. It is assumed that blanks are of no interest to the information sink at the receiver. This source produces data at an average rate of P bits/s. Thus for $P < 1$ there will be times when the source has nothing to transmit, and this condition of the source must be conveyed over the channel by means of overhead bits transmitted along with information bits. As we shall see, the number of these overhead bits that are required is a function of the allowable delay.

An example of the relationship between delay and the overhead information is given by the following encoding procedure. All of the source outputs in an l second interval are encoded in a block. In each l second interval the information that is transmitted is the number of data bits generated by the source and their values. If more than l bits are required the

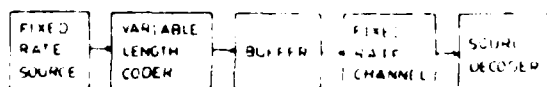


Fig. 1. Source encoding model.

excess is buffered. If this requires less than l bits, zeros can be transmitted in the slack interval remaining after the buffered bits with no ambiguity. Since the position of blanks in an l second interval is of no interest, the number of actual data bits produced by a source in an l second interval follows a binomial distribution. The entropy of the source over an l second interval is

$$H(l, P) = Pl - \sum_{i=0}^l b(i; l, P) \log_2 b(i; l, P)$$

where

$$b(i; l, P) = \binom{l}{i} P^i (1-P)^{l-i}. \quad (2)$$

In Fig. 2 we show plots of $H(l, P)/l$ as a function of l for various values of P . In fact, it can be shown that

$$\lim_{l \rightarrow \infty} \frac{H(l, P)}{l} = P$$

indicating that the portion of capacity devoted to overhead becomes zero as the block size is increased. There is a price to be paid in performance since bit delay increases as the block length increases. Thus, by increasing l , the block length, and by holding P fixed, the source can be matched to the channel in the sense of $H(l, P)/l \leq 1$ (see Fig. 2). On the other hand, we can show that for l fixed, $H(l, P)/l > 1$ as P approaches 1. Define $\epsilon \triangleq 1 - P$. For ϵ small, $H(l, 1 - \epsilon) \approx (1 - \epsilon)l - (1 - \epsilon) \log_2(1 - \epsilon) - \epsilon \log_2 \epsilon$. Setting $H(l, 1 - \epsilon) > 1$ we find that

$$-(1 - \epsilon) \log_2(1 - \epsilon) > \epsilon(1 + \log_2 \epsilon). \quad (3)$$

For $0 < \epsilon < \frac{1}{2}$, the RHS of (3) is negative. Since the LHS of (3) is positive for $0 < \epsilon < 1$, the assertion is proved.

The foregoing illustrates the crux of the problem of encoding of bursty sources; the amount of overhead required is a function of the block length which translates directly into delay. The same phenomena can be seen operating for more complex source models. As an example consider the case of multiplexing several distinct sources on the same channel. If the allowable delay is large enough, the overhead required to distinguish the source can be made negligible. For example, data from each of the sources could be collected in separate buffers. The contents of these buffers could then be time division multiplexed onto the line. In this way no overhead is required to indicate the sources of data. This procedure breaks down when one or more of the source buffers is empty. However, by making delay large enough, the occurrence of this event will have little effect.²

² For a treatment of encoding of multiple sources see [6].

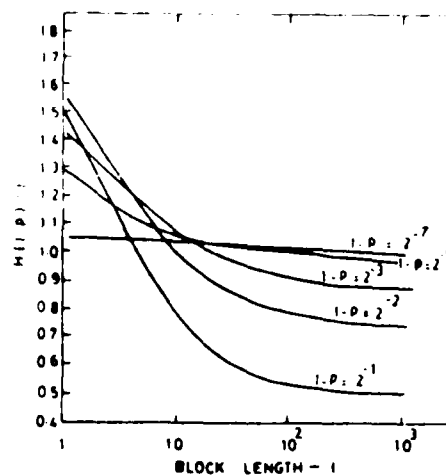


Fig. 2. Source entropy.

ADAPTIVE ENCODING

The block encoding technique discussed above can be improved upon by adding a measure of adaptivity to the process. Assume that the encoding begins in the same way as in the previous section by using a Huffman code to transmit the number of data bits that the source produces in an l_1 second interval. The probabilities of the number of data bits are a function of l_1 as well as P [see (2)]. Thus the Huffman code is in turn a function of l_1 . It is presumed that the information sink knows l_1 and P and therefore knows the Huffman code that is to be used. Let us assume that the transmission of the code word together with the actual data bits consumes l_2 seconds. The source output over the l_2 second interval is encoded in the same fashion as previously. Since in general l_1 is not equal to l_2 , the Huffman code used to encode the number of data bits is different. However, the source and sink can cooperate in the construction of a code since l_2 is known to both. The process repeats for each successive interval. This adaptive scheme is at least as good as the fixed block scheme since encoding begins immediately after the last data bit of a block has been transmitted. The only overhead is the Huffman code word for the number of bits. In the fixed block scheme messages can be delayed while zeros are filling out a block.

The successive coding intervals $l_1, l_2, \dots$ in this scheme form a Markov chain. Since there are a large number of states, finding the state probabilities is a formidable numerical problem. For our purposes a Monte Carlo simulation of a somewhat simplified model of the scheme is adequate. The simplification comes about by upper and lower bounding the Huffman code word by quantities that are much easier to compute. The uncertainty of a symbol with probability P_i is $-\log_2 P_i$. Let $[-\log_2 P_i]^-$ denote the integer part of $-\log_2 P_i$. If a code with lengths $[-\log_2 P_i]^-; i = 0, 1, \dots, l$ existed, its average length would lower bound the entropy, which in turn lower bounds the average length of a Huffman code. A code with lengths $[-\log_2 P_i]^- + 1$ satisfies the Kraft inequality; consequently, a code with these lengths can be constructed. However, such a code must have average length greater than the appropriate Huffman code. The results of the simulation are shown in Fig. 3 where the averages of the upper and lower bounds of the block lengths, $l_1, l_2, \dots$, are plotted as a function of $\log_2(1/l_1)$.

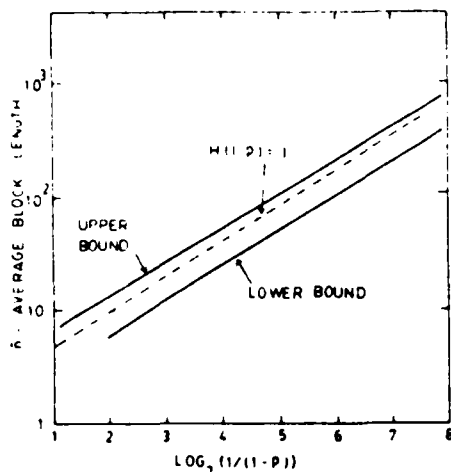


Fig. 3. Performance of adaptive block encoding scheme.

P). The results indicate that as P approaches one, the channel is matched by encoding over longer blocks which implies larger delay. This result is consonant with the results shown in Fig. 2.

The results of Figs. 2 and 3 also suggest that for a given value of P , the average length of a block is given by block $\bar{l} = H(\bar{l}, P)$. It can be shown that there exists an l^* such that $E(l_2/l_1) < l_1$ if $l_1 > l^*$ and $E(l_2/l_1) > l_1$ if $l_1 < l^*$. In Fig. 3 the locus of such points, i.e., $H(l^*, P) = l^*$, is shown. We notice in Fig. 3 that as P increases, this equilibrium point occurs for larger and larger values of l , indicating larger and larger coding delay. In fact, it can be shown that the solution to the equation is approximated by

$$l^* \approx 2.78/(1-P) \quad (4)$$

for P close to one. Consider the entropy of a binomial distribution with parameters P and l . The maximum probability is approximately equal to $[2\pi P(1-P)]^{-1/2}$ implying that the entropy is greater than or equal to $\frac{1}{2} \log_2(2\pi P(1-P)l)$. Now, a theorem attributed to J. M. Massey states that the entropy of a discrete random variable with variance σ^2 is less than or equal to $\frac{1}{2} \log_2(2\pi e(\sigma^2 + 1/12))$. Since the variance of the binomial distribution is $P(1-P)l$, we may approximate the entropy by $\frac{1}{2} \log_2(2\pi e P(1-P)l)$. The solution to the equation $\frac{1}{2} \log_2 2\pi e P(1-P)l^* + Pl^* = l^*$ is given by (4) for P close to one.

FLAGS

An alternative to block encoding is a technique employing flags. Whenever the transmitter has nothing to send, this state is indicated by the transmission of a unique F -bit sequence, the flag. During the time interval required to transmit this flag, binary digits produced by the source are buffered and read out after the entire flag has been transmitted. The output of the source is transmitted without encoding except when the random source replicates the first $F-1$ bits of flag. In this event the transmitter stuffs a bit which is the complement of the last bit of the flag. Thus, the only time a flag is transmitted is when the buffer is empty and a blank appears at the output of the source.

As we shall see presently, for each value of P there is a flag length which minimizes bit delay. This minimum strikes a balance between flag bits and stuffed bits. As P decreases the buffer is more likely to be empty; therefore, the flag length should be decreased. However, decreasing the flag length increases the occurrence of stuffed bits.

The frequency of occurrence of stuffed bits in a random data sequence can be analyzed by means of the theory of recurrent events. The buffer occupancy can be analyzed by means of a Markov chain model. Included in both of these analyses are the parameters P , the probability the source produces a binary digit, and F , the length of the flag. It can be shown that the results also depend upon the particular bit pattern in the flag.

The structure of flags was first studied by Nielsen [7], who defined the concept of bifix in connection with flags. A flag is bifix free if there is no sequence which is a prefix and a suffix to the first $F-1$ bits of the flag. The sequences 10000 and 00001 are examples, respectively, of bifix free and non-bifix free flags. It is a simple exercise to demonstrate that non-bifix free flags can cause delays in the decoding of data bits. While this is no problem in terms of operation, the analysis is complicated. We shall, therefore, concentrate on the bifix free flag 100...0 in the sequel.

The theory of success runs as treated by Feller [8] allows one to find the generating function of the time between occurrence of stuffed bits in a stream of random zeros and ones. For the moment we consider successive source outputs, ignoring intervening blanks. Consider a flag consisting of a one followed by $F-1$ zeros. If the source produces a one followed by $F-2$ consecutive zeros, immediately a one is stuffed. This is a renewal point since the system erases all memory and begins afresh. Let U_j be the probability of a stuffed bit after the j th data bit. The probability of a random data sequence producing a one followed by $F-2$ zeros is 2^{-F+1} . However, a stuff bit can only occur after $F-2$ zeros. We have

$$U_j = 2^{-F+1}. \quad (5)$$

The generating function of the recurrence interval is then

$$F(S) = \frac{S(S/2)^{F-1}}{1 - S + S(S/2)^{F-1}}. \quad (6)$$

The mean recurrence interval is

$$\mu = 2^{F-1}. \quad (7)$$

In (7) the recurrence interval is in terms of successive zero and one outputs of the source. However, between these outputs are a geometrically distributed number of blanks. If these blanks are taken into account, the average time interval in seconds is given by μ/P .

In connection with the frequency of occurrence of stuff bits, the question of stability arises. For the bursty model the mean time between blanks is $1/(1-P)$. In order that the

transmit buffer not overflow, it is necessary that the mean time between blanks be less than the mean time between stuffs. For (8) we have the criterion

$$F > \frac{\ln(P/(1-P))}{\ln 2} + 1. \quad (8)$$

The overhead incurred by the use of flags other than that treated above also follows directly from an analysis of success runs. Furthermore, the same analysis of the frequency bit stuffing can be carried out for the simultaneous use of several different flags.

Camrass and Gallager [9] have studied source encoding by means of flags. There is a similarity to our work in that we both use the theory of success runs to study the frequency of occurrence of stuff bits. However, the difference lies in the use to which the flags are put. Camrass and Gallager append a flag to a message which happens to have a geometric distribution. In our model the burst of consecutive binary digits is geometrically distributed, but we do not delineate different bursts. A flag is transmitted only if there are no source output bits available.

BIT ENCODING DELAY

The encoding delay of a bit in the flag scheme can be studied by means of a two-dimensional Markov chain model. The chain is in state (i, j) if there are i bits in the buffer and if j of the immediate preceding successive bits in the data sequence coincide with the first j bits of the flag. The last component is the memory of the system and has dimension $F-2$. The mechanism for state transmission may be described by considering specific cases. If the buffer is not empty, its content decreases when the data source produces a blank. If the buffer is empty when a blank is produced, then the first bit of the flag is transmitted and the remaining $F-1$ bits are inserted into the transmit buffer. The level of the buffer can also increase due to a stuff bit. We point out that the stuffing of a bit or the transmission of a flag brings the system to the zero memory state.

Let us consider the flag 100...0, a one followed by $F-1$ zeros. The state transition diagram for this case is shown on Fig. 4. Let q_{ij} ; $i = 0, 1, 2, \dots$; $j = 0, 1, \dots, F-2$ denote the steady state probability that the system is in state (i, j) . We can write a set of equilibrium equations by considering particular sets of states. Consider the states such that $i \geq 0$, $2 \leq j \leq F-2$; we have

$$q_{ij} = (P/2)q_{i,j-1} + (1-P)q_{i+1,j}. \quad (9)$$

The state (i, j) is entered by the arrival of a bit replicating a flag bit with probability $P/2$ or the arrival of a blank with probability $1-P$. For $i = 0$ and $j = 0$ we have

$$q_{00} = (1-P)q_{10} + (P/2)q_{00}. \quad (10)$$

Notice that the zero memory state implies that a flag or a stuff bit has been transmitted. The system remains in the zero memory state if a zero is the output of the data source. We

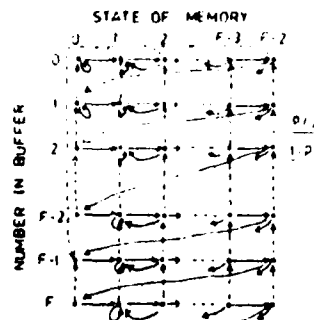


Fig. 4. Markov chain model of flag scheme.

turn now to the case $i \neq 0, F-1, j = 0$. We have

$$q_{i0} = (1-P)q_{i+1,0} + (P/2)q_{i0} + (P/2)q_{i-1,F-2}. \quad (11)$$

For $i \geq 0$ and $j = 1$ we have

$$q_{i1} = (P/2)q_{i1} + (1-P)q_{i+1,1} \quad (12)$$

where $q_i \triangleq \sum_{j=0}^{F-2} q_{ij}$; $i \geq 0$ is the probability that the buffer size is i . The final case to be considered is $i = F-1, j = 0$.

$$q_{F-1,0} = (1-P)q_{F,0} + (P/2)q_{F-1,0} + (P/2)q_{F-2,F-2} + (1-P)q_{00}. \quad (13)$$

Define the z -transform of the buffer occupancy probabilities at a particular memory state as

$$Q_j(z) \triangleq \sum_{i=0}^{\infty} z^i q_{ij}; \quad j = 0, 1, \dots, F-2 \quad (14a)$$

and the buffer occupancy moment generating function as

$$Q(z) = \sum_{j=0}^{F-2} Q_j(z). \quad (14b)$$

Employing (9)-(13) together with the definitions in (14) it can be shown, after some manipulation, that

$$Q(z) = \left[- \sum_{l=1}^{F-2} \left[(z-1) \left(\frac{Pz}{2} \right)^{F-l-1} (z-1+P)^{l-1} \cdot (z-1+P)^{F-2} (z^F-1) \right] (1-P)q_{0l} + (1-P)(z-1+P)^{F-2} (z^F-1)q_{00} \right] / (z-1) \cdot [(1-P)(z-1+P)^{F-2} - (Pz/2)^{F-1}]. \quad (15)$$

In (15) there are $F-1$ unknowns for q_{0l} ; $l = 0, 1, \dots, F-2$. In order to solve for these unknowns we utilize a technique employing Rouché's theorem [10], [11]. $Q(z)$ is analytic within the unit disk; consequently, when the denomina-

tor has a zero within the unit disk, the numerator must have a zero at the same point. From the stability condition of (8) it can be shown that the polynomial in the denominator of (15) has $F - 2$ distinct roots on the unit disk. Let us denote these roots as θ_i , $i = 1, 2, \dots, F - 2$. Since the denominator of $Q(z)$ must be equal to zero at these points, we have then from (15) the following set of equations.

$$\begin{aligned} & - \sum_{i=1}^{F-2} \left[(\theta_i - 1) \left(\frac{P\theta_i}{2} \right)^{F-i-1} (\theta_i - 1 + P)^{i-1} \right. \\ & \quad \left. - (\theta_i - 1 + P)^{F-2} (\theta_i^F - 1) \right] (1 - P)q_{0i} \\ & \quad + (1 - P)(\theta_i - 1 + P)^{F-2} (\theta_i^F - 1)q_{00} \\ & = 0, \quad i = 1, 2, \dots, F - 2. \end{aligned} \quad (16a)$$

We also have the normalizing condition

$$Q(1) = 1. \quad (16b)$$

Equations (16a) and (16b) give $F - 1$ equations in $F - 1$ unknowns.

Finding the roots of the polynomial $P(z)$ and solving (16) for the quantities q_{0i} , $i = 0, 1, \dots, F - 2$ are relatively straightforward numerical problems. We can then calculate $Q(z)$, the generating function of the transmit buffer occupancy. Returning to the equilibrium equations (9)–(13), we see that the probabilities q_{ij} , $i = 0, 1, 2, \dots; j = 0, 1, \dots, F - 2$ can also be found. These will be used to calculate decoding delay in the sequel.

We use as a measure of performance the average delay suffered by a bit as determined by the average number of bits in the buffer. The average number of bits in the buffer can be determined from $Q(z)$. The results are shown in Fig. 5 where $\bar{q}$, the average delay, is shown as a function of flag length with $1 - P$, the probability of a blank, as a parameter.

As shown in Fig. 5, for large values of F , the average delay may be approximated by $(F - 1)/2$ for all values of P . For large values of F , the buffer occupancy jumps to $F - 1$ when a flag is sent. It more or less linearly (on the average) returns to zero as blanks predominate over stuffed bits. When it reaches zero a flag is sent again. As the flag length is decreased, the stuff bits come into play. (Recall that from (7) the minimum value of F is determined by considering stability in connection with stuffed bits.) Although it is not evident from the curves shown in Fig. 5 for large P , there is a minimum value of average delay at a value of F larger than the minimum. This slight dip in the curves is shown by the figures in Table I.

As P approaches one, the behavior of the flag encoding technique is similar to the block encoding technique described in the above. As is illustrated on Fig. 5, the delay for any value of P may be lower bounded by $(F - 1)/2$. However, from (10) we have that $F > \log_2(P/(1 - P)) + 1$, and as P approaches one, average delay becomes infinite.

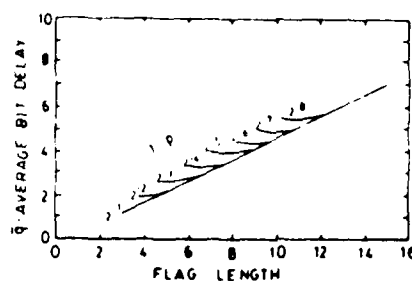


Fig. 5. Performance of flag scheme.

TABLE I

$\frac{F}{1-P}$	7	8	9	10	11	12
2-5	3.82	3.78	4.12	5.03		5.51
2-6		4.40	4.30	4.63		5.53
2-7			4.94	4.82	5.13	5.56
2-8				5.47	5.32	5.64

COMPARISON OF BLOCK AND FLAG ENCODING

A comparison of Figs. 3 and 5 shows a large difference in encoding delay in favor of the flag scheme. For example, for $P = 1 - 2^{-4}$ the average block length is approximately 40 bits. We take this to be an approximation to the number of bit intervals a newly arriving bit must wait, i.e., 40 s. In contrast, the encoding delay for a bit in the flag technique can be less than 4 s for the appropriate flag length. These results apply generally to other probabilities. Further, as $P \rightarrow 1$, delay in the block scheme is proportional to $1/(1 - P)$, whereas in the flag scheme it is proportional to $\log 1/(1 - P)$.

For the flag encoding scheme, encoding delay is not the only delay encountered before a bit is delivered to an information sink at the receiver. When a bit is received there may be an ambiguity that must be resolved. Moreover, this decoding delay is random and depends upon the state of the system. begin the analysis by assuming the state (i, j) for the flag 100...0. Recall that i is the number of data bits in the transmitting buffer and j is the system memory with respect to the flag. The ambiguity is resolved immediately if the received bit does not replicate the flag, i.e., a zero for state $j = 0$ and a one otherwise. The ambiguity persists if the bits generated by the source continue to replicate the flag, and if the source produces i or fewer blanks. To justify this second requirement we note that more than i blanks would deplete the buffer and a flag that would be transmitted resolves the ambiguity. The ambiguity can persist for at most $F - j - 1$ seconds, since the first $F - 1$ bits of the flag would be replicated and a stuffed bit would be transmitted, also resolving the ambiguity. We summarize this in the following equations where the random variable D denotes decoding delay.

$$\begin{aligned} P_i[D > d/(i, j)] \\ = (1/2)(P/2)^d \sum_{l=0}^i (1 - P)^l \binom{d+l-1}{l} \end{aligned} \quad (17)$$

Equation (17) is the probability of a sequence replicating the

flag bits interspersed with i or fewer blanks. Based on (17) we have for the probability of delay

$$P_r[D = 0/i, j] = 1/2 \quad \text{for all } i, j$$

$$P_r[D = 1/i, j] = \begin{cases} 1/2; & j = F - 2 \\ 1/2 - P_r[D > 1/(i, j)]; & j = 0, 1, \dots, F - 3 \end{cases}$$

$$P_r[D = d/i, j]$$

$$= \begin{cases} 0; & d > F - j \\ P_r[D > d - 1/(i, j)] & d = F - j - 1 \\ P_r[D > d - 1/(i, j)] - P_r[D > d/(i, j)] & 0 \leq d \leq F - j - 2. \end{cases}$$

The average delay can be found by averaging over this conditional distribution, then averaging over the state (i, j) whose distribution was obtained in the previous analysis. The results are shown in Fig. 6 where delay is plotted as a function of flag length with $1 - P$ as a parameter. We see that the decoding delay is not large and that it is not very sensitive to F and $1 - P$. We conclude then that decoding delay has little effect upon the flag encoding scheme as related to the other encoding techniques we have examined.

CONCLUSION AND FUTURE WORK

A simple source encoding problem which illustrates the tradeoff between delay and efficient encoding has been posed. Two encoding techniques, one using Huffman codes and one using flags, have been examined in detail. For the second of these, stability conditions were found and a Markov chain model was analyzed. In both cases it was found that as $P \rightarrow 1$ the delay approaches infinity. The paradox in this result is that for $P = 1$ the coding problem disappears. A secondary result is the superiority of the flag encoding scheme over Huffman coding when delay is used to measure performance. This suggests a kind of compatibility between delay and flags.

We have also found other bounds. All lower bounds found tend to a finite value as $P \rightarrow 1$, while for all upper bounds found the delay becomes unbounded. Recently Roskind [12] has found a strategy for which the delay is proportional to $\log [\log 1/(1 - P)]$ as $P \rightarrow 1$.

The work can be continued in several directions. The most compelling of these is the study of a lower bound for delay over all possible codings especially when P is close to one. Also, the source model can be extended by considering more general arrival processes, and by considering several sources which must remain distinct, multiplexing can be addressed. Again it appears that a Markov chain model can be used to describe a flag encoding scheme.

ACKNOWLEDGMENT

The authors would like to express their appreciation to C.-K. Jen for his programming work. The authors would also like to acknowledge the derivation of (4) by an anonymous reviewer.

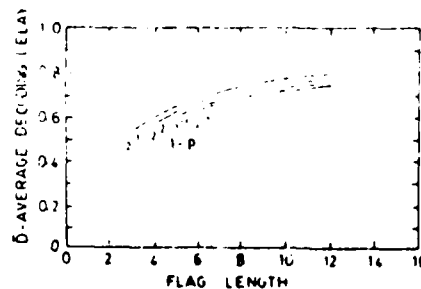
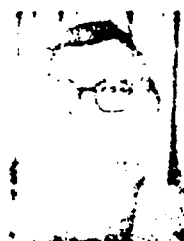


Fig. 6. Average decoding delay versus flag length.

REFERENCES

- [1] S. S. Lam, "A new measure for characterizing data traffic," *IEEE Trans. Commun.*, vol. COM-26, Jan. 1978.
- [2] F. Jelinek, "Buffer overflow in variable length encoding of fixed rate sources," *IEEE Trans. Inform. Theory*, vol. IT-14, May 1963.
- [3] F. Jelinek and K. S. Schneider, "On variable-length to block coding," *IEEE Trans. Inform. Theory*, vol. IT-18, Nov. 1972.
- [4] —, "Variable-length encoding of fixed rate Markov sources for fixed-rate channels," *IEEE Trans. Inform. Theory*, vol. IT-20, Nov. 1974.
- [5] R. G. Gallager, "Basic limits on protocol information in data communication networks," *IEEE Trans. Inform. Theory*, vol. IT-22, July 1976.
- [6] P. A. Humblet, "Source coding for communication concentrators," Massachusetts Inst. of Technol., Cambridge, Rep. ESL-R-798, Jan. 1978.
- [7] P. J. Nielsen, "On the expected duration of a search for a fixed pattern in random data," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 702-704, Sept. 1973.
- [8] W. Feller, *An Introduction to Probability Theory and Its Applications*, vol. I. New York: Wiley, 1950, pp. 299 ff.
- [9] R. J. Camrass and R. G. Gallager, "Encoding message lengths for data transmission," *IEEE Trans. Inform. Theory*, vol. IT-24, July 1978.
- [10] N. T. J. Bailey, "On queuing processes with bulk services," *J. Roy. Stat. Soc. Ser. B*, vol. 16, pp. 80-87, 1954.
- [11] F. Downton, "Waiting time in bulk service queues," *J. Roy. Stat. Soc. Ser. B*, vol. 17, pp. 256-261, 1955.
- [12] R. J. Roskind, "Protocols for encoding idle characters in data streams," Master's thesis, Massachusetts Inst. of Technol., Cambridge, June 1980.



J. F. Hayes (S'65-M'66-SM'79) received the B.E.E. degree from Manhattan College, New York, NY, in 1956, the M.S. degree in mathematics from New York University, New York, in 1961 and the Ph.D. degree in electrical engineering from the University of California, Berkeley, in 1966.

From 1956 to 1960 he was a member of the Technical Staff at Bell Laboratories, Murray Hill, NJ. He worked at the Columbia University Electronics Research Laboratories, New York, from 1960 to 1962. In the interval 1966-1969 he was a member of the faculty at Purdue University, Lafayette, IN. During the summer of 1967 he was employed by the Jet Propulsion Laboratory, Pasadena, CA. From 1969 to 1978 he was a member of the Technical Staff at Bell Laboratories, Holmdel, NJ. Since September 1978 he has been a member of the Department of Electrical Engineering, McGill University, Montreal, P.Q., Canada. His research interest is primarily in the area of computer communications.

Prof. Hayes is currently the Editor for Computer Communication of this TRANSACTIONS.



R. R. Boorstyn (S'58-M'59) for a photograph and biography, see p. 480 of the April 1981 issue of this TRANSACTIONS.

E.2 Delay and Overhead in the Encoding of Bursty Sources

Boorstyn and Hayes

IEEE International Conference on Communications,

June 1980, Seattle

DELAY AND OVERHEAD IN THE ENCODING OF BURSTY SOURCES

R.R. Boorstyn
Polytechnic Institute of New York

J.F. Hayes
McGill University

ABSTRACT

A model of a bursty source whose output is transmitted over a synchronous channel is studied with the objective of understanding the relationship between coding efficiency and delay. Two encoding schemes, a block encoding technique and a technique employing flags, are examined in some detail. The flag technique is analyzed by means of the theory of recurrent events and by two dimensional Markov chains. For both the block encoding and the flag schemes it is shown that as the source becomes less bursty delay approaches infinity. This result obtains in spite of the fact that both schemes are reasonable and in the limit the encoding problem disappears.

INTRODUCTION

This paper is concerned with the interplay of delay and overhead in the multiplexing of data on common facilities. The relationship of these quantities is due to certain unique properties of data in an interactive environment. A basic property of data traffic is burstiness, i.e., short busy periods interspersed with relatively long idle periods which contain no information. The second relevant property of data has to do with delay. In many applications data traffic is highly interactive placing a premium on rapid response times. These properties of data signals allow one to formulate a source encoding problem which considers delay as a basic component.*

SOURCE MODEL

In order to study the problem of delay in coding we consider the following model. A source produces either a blank or binary digit once every second with the following probabilities.

* Related work on the encoding of bursty sources has been carried out by Gallager¹ and Humblet².

This work was supported in part by NSERC Grant A0901 and by the postdoctoral program SADC B9-3513 funded by US Army CORADCOM.

$$\begin{aligned} P[\text{blank}] &= 1 - P \\ P[0 \text{ output}] &= P/2 \\ P[1 \text{ output}] &= P/2 \end{aligned} \quad (1)$$

The sequence of 0's and 1's produced by this source is to be carried over a synchronous, error-free binary channel with a capacity of one bit per second. This source produces information at an average rate of P bits/second. Thus for $P < 1$ there will be times when the source has nothing to transmit. But the channel, being synchronous, transmits only a zero or a one every second and this condition must be suitably encoded. In general the number of overhead bits that are required to encode the state of the source is a function of the allowable delay.

The relationship between delay and channel capacity can be illustrated by means of the following encoding procedure. All of the source outputs in an l second interval are encoded in a block. In each l bit interval the information that must be transmitted is the number of binary digits generated by the source and the values of these digits. If this requires less than l bits intervals, zeros can be transmitted in the slack interval with no ambiguity. Since the number of data bits produced in an l bit interval follows a binomial distribution, the entropy of this encoding is

$$H(l, P) = Pl - \sum_{i=0}^l b(i; l, P) \log_2 b(i; l, P) \quad (2)$$

$$\text{where } b(i; l, P) = \binom{l}{i} P^i (1-P)^{l-i}$$

In Figure 1 we show plots of $H(l, P)/l$ as a function of l for various values of P . We see that $H(l, P)/l$ asymptotically approaches P for large l . Thus, the portion of capacity devoted to overhead becomes zero as the block size is increased. However, the delay increases as the block length increases. Increasing l , the block length, and by holding P fixed the source can be matched to the channel in the sense of $H(l, P)/l \leq 1$ (see Fig. 1). On the other hand it can be shown that for fixed $H(l, P)/l > 1$ as P approaches 1. Define $\epsilon = 1 - P$. For ϵ small $H(l, 1-\epsilon) = (1-\epsilon)l - (1-\epsilon)\log_2(1-\epsilon) - \epsilon\log_2\epsilon$. But for $H(l, 1-\epsilon) > l$ we have

$$-(1-\epsilon)\log_2(1-\epsilon) > \epsilon(1+\log_2\epsilon) \quad (3)$$

For $0 < \epsilon < 1$ the RHS of Eq. 3 is negative. Since the LHS of Eq. 3 is positive for $0 < \epsilon < 1$, the assertion is proved.

The foregoing illustrates the crux of the problem of encoding of bursty sources; the amount of overhead required is a function of the block length which translates directly into delay. The same phenomena can be seen operating for more complex source models involving multiple sources².

FLAGS

Flags can be used to encode the basic bursty source discussed above. The encoding process is initiated by transmitting the digital output of the source. When a blank appears a flag composed of a unique F -bit sequence is transmitted. During the time interval required to transmit this flag binary digits produced by the source are buffered and read out after the entire flag has been transmitted. The output of the source is transmitted without encoding except when the random source replicates the first $F - 1$ bits of flag. In this event the transmitter stuffs a bit which is the complement of the last bit of the flag. The only time a flag is transmitted is when the buffer is empty and a blank appears at the output of the source. For each value of P there is a flag length which strikes a balance between flag bits and stuffed bits. As P decreases the buffer is more likely to be empty therefore the flag length should be decreased. However decreasing the flag length increases the occurrence of stuffed bits.

The frequency of occurrence of stuffed bits in a random data sequence can be analysed by means of the theory of recurrent events. The buffer occupancy can be analyzed by means of a Markov chain model. Included in both of these analyses are the parameters P , the probability the source produces a binary digit and F , the length of the flag. To some extent, the results also depend upon the particular bit pattern in the flag.

The theory of success runs as treated by Feller³ allows one to find the generating function of the time between occurrence of stuffed bits in a stream of random zeros and ones (For an alternative approach see Ref. 4). Consider a flag consisting of F zeros. If the source produces $F - 1$ consecutive zeros immediately a one is stuffed. This is a renewal point since the system erases all memory and begins afresh the counting of strings of zeros. Let U_j be the probability of a stuffed bit after the j th data bit. The probability of the source producing a sequence of $F - 1$ consecutive zeros on bit numbers $j, j - 1, \dots, j - F + 2$ is 2^{-F+1} . There is one and only one stuff bit after one of these zeros. The probability that the stuff is after bit number $j - k$ and the next k bits are zeros is $U_{j-k} 2^{-k}$ ($k=0, 1, \dots, F - 2$). Since $F - 1$ possibilities are mutually exclusive we have the following recurrence relationship

$$U_j = U_{j-1} 2^{-1} + \dots + U_{j-F+2} 2^{-F+2} = 2^{-F+1}; j \geq F - 1 \quad (4a)$$

$$U_1 = U_2 = \dots = U_{F-2} = 0 \quad (4b)$$

$$U_0 = 1 \quad (4c)$$

From Eq. 4 one can find the generating function of the recurrence times.

$$F_1(z) = \frac{(z/2)^{F-1} (1-z/2)}{1-z+(z/2)^F} \quad (5)$$

We can calculate the mean recurrence time as

$$\mu_1 = \frac{1 - (1/2)^{F-1}}{(1/2)^F} = 2^F - 2 \quad (6)$$

The sensitivity of the stuffing process to the particular flag pattern is illustrated by considering the flag 100...0, a one followed by $F - 1$ zeros. The probability of a random data sequence producing a one followed by $F - 2$ zeros is $(1/2)^{F-1}$; however a stuff bit can only occur after $F - 2$ zeros. We have

$$U_j = 2^{-F+1}$$

The generating function of the recurrence times is then

$$F_2(z) = \frac{(z/2)^{F-1}}{1 - z + (z/2)^{F-1}} \quad (7)$$

The mean recurrence time is

$$\mu_2 = 2^{F-1} \quad (8)$$

It is of interest to note that from Eqs. 6 and 8

$$\lim_{F \rightarrow \infty} \frac{\mu_1}{\mu_2} = 2$$

Thus in the limit the sequence 1000...0 evokes twice as many stuff bits as the sequence 000...0.

In connection with the frequency of occurrence of stuff bits the question of stability arises. For the bursty model the mean time between blanks is $1/(1-P)$. In order that the transmit buffer not overflow it is necessary that the mean time between blanks be less than the mean time between stuffed bits. From Eq. 6 we can derive the following stability criterion for the all zero sequence.

$$F > \frac{\ln(2/(1-P))}{\ln 2} \quad (9)$$

From Eq. 8 we have the criterion for the sequence 100...0

$$F > \frac{\ln(1/(1-P))}{\ln 2} + 1 \quad (10)$$

Bit delay for the flag scheme can be studied by means of a Markov chain model. For ease of analysis we only consider the flag 100...0. The same general conclusions apply to the all zero flag. The state of the chain, (i, j) , is two dimensional consisting of the number of bits in the buffer, i , and of the number of bits in the data sequence that replicate the flag, j . The last

component is the memory of the system and has dimension $F - 1$. The mechanism for state transmission may be described by considering specific cases. If the buffer is not empty its contents decrease when the data source produces a blank. If the buffer is empty when a blank is produced then the first bit of the flag is transmitted and the remaining $F - 1$ bits are inserted into the transmit buffer. The level of the buffer can also increase due to a stuff bit. We point out that the stuffing of a bit or the transmission of a flag brings the system to the zero memory state.

The state transition diagram for the flag 100...0 is shown in Fig. 2. Let $q_{ij}; i = 0, 1, 2, \dots; j = 0, 1, \dots, F - 2$ denote the steady state probability that the system is in state (i, j) . We can write a set of equilibrium equations by considering particular sets of states. Consider the states such that $i \geq 0, 2 \leq j \leq F - 2$ we have

$$q_{ij} = (P/2)q_{i,j-1} + (1-P)q_{i+1,j} \quad (11)$$

The state (i, j) is entered by the arrival of a bit replicating a flag bit with probability $P/2$ or the arrival of a blank with probability $1 - P$. Now consider the case where a string of zeros is interrupted by a one. We have

$$q_{00} = (1 - P)q_{10} + (P/2)q_{00} \quad (12)$$

Notice that the zero memory state implies that a flag or a stuff bit has been transmitted. The system remains in the zero memory state if a zero is the output of the data source. We turn now to the case $i \neq 0, F - 1, j = 0$. We have

$$q_{i0} = (1-P)q_{i+1,0} + (P/2)q_{i0} + (P/2)q_{i-1,F-2} \quad (13)$$

For $i \geq 0$ and $j \geq 1$ we have

$$q_{i1} = (P/2)q_{i1} + (1-P)q_{i+1,1} \quad (14)$$

The final cases to be considered are that of $i = F - 1, j = 0$.

$$q_{F-1,0} = (1-P)q_{F,0} + (P/2)q_{F-1,0} + (P/2)q_{F-2,F-2} + (1-P)q_{00} \quad (15)$$

where

$$q_0 = \sum_{j=0}^{F-2} q_{0,j}$$

Define the z-transform of the buffer occupancy probabilities as

$$Q_j(z) \triangleq \sum_{i=0}^{\infty} z^i q_{i,j}; j = 0, 1, \dots, F - 2 \quad (16a)$$

and

$$Q(z) = \sum_{j=0}^{F-2} Q_j(z) \quad (16b)$$

Employing Eqs. 11-14 together with the definitions in Eq. 15 it can be shown, after some manipulation, that

$$Q(z) = \frac{\left[- \sum_{l=1}^{F-2} [(z-1) \left(\frac{Pz}{2}\right)^{F-l-1} (z-1+P)^{l-1} - (z-1+P)^{F-2} (z^F - 1)] (1-P)q_{0l} + (1-P)(z-1+P)^{F-2} (z^F - 1)q_{00} \right]}{(z-1) [(1-P)(z-1+P)^{F-2} - (Pz/2)^{F-1}]} \quad (17)$$

In Eq. 16 there are $F - 1$ unknowns for $q_{0l}; l = 0, 1, \dots, F - 2$. In order to do this we utilize a technique employing Rouché's theorem. $Q(z)$ is analytic within the unit disk; consequently, when the denominator has a zero within the unit disk the numerator must have a zero at the same point. From the stability condition of Eq. 10 it can be shown that the polynomial in the denominator of Eq. 17 has $F - 2$ distinct roots on the unit disk. Let us denote these roots as $\theta_i; i = 1, 2, \dots, F - 2$.

Since the denominator of $Q(z)$ must be equal to zero at these points, we have then from Eq. 17 the following set of equations.

$$\begin{aligned} & - \sum_{l=1}^{F-2} [(\theta_i - 1) \left(\frac{P\theta_i}{2}\right)^{F-l-1} (\theta_i - 1 + P)^{l-1} - (\theta_i - 1 + P)^{F-2} - (\theta_i^F - 1)] (1-P)q_{0l} \\ & + (1-P)(\theta_i - 1 + P)^{F-2} (\theta_i^F - 1)q_{00} = 0, \\ & i = 1, 2, \dots, F - 2 \end{aligned} \quad (17a)$$

We also have the normalizing condition

$$Q(1) = 1 \quad (17b)$$

Eqs. 17a and 17b give $F - 1$ equations in $F - 1$ unknowns.

Finding the roots of the polynomial $P(z)$ and solving Eq. 17 for the quantities $q_{0l}, l = 0, 1, \dots, F - 2$ are relatively straightforward numerical problems. We can then calculate $Q(z)$ the generating function of the transmit buffer occupancy. Returning to the equilibrium Eqs. 11-15, we see that the probabilities $q_{i,j}, i = 0, 1, 2, \dots; j = 0, 1, \dots, F - 2$ can also be found.

We use as a measure of performance the average delay suffered by a bit as determined by the average number of bits in the buffer. The average number of bits in the buffer can be determined from Eq. 16 by differentiation or by calculating

$$\bar{q} = \sum_{i=0}^{\infty} i \sum_{j=0}^{F-2} q_{i,j}. \quad \text{The results are shown in Fig. 3}$$

where $\bar{q}$ average delay is shown as a function of flag length with $1 - P$, the probability of a blank, as a parameter.

As shown in Fig. 3, for large values of F , the average delay may be approximated by $(F-1)/2$ for all values of P . This is as expected since for large values of F , stuff bits have little effect. As the flag length is decreased the stuff bits come into play. From Eq. 10 the minimum value of F is determined by considering stability in connection with stuffed bits. Although it is not evident from the curves shown on Fig. 3, for large P there is a minimum value of average delay at a value of F larger than the minimum. As P approaches one the behavior of the flag encoding technique is similar to the block encoding technique described in the foregoing. As is illustrated in Fig. 3 the delay for any value of P may be lower bounded by $(F-1)/2$ by ignoring the effect of stuff bits. However from Eq. 10 we have that $\lim_{P \rightarrow 1} F = \infty$. Thus

as P approaches one, average delay becomes infinite. Eq. 9 indicates that we may expect the same sort of behavior from the all zero sequence.

CONCLUSION AND FUTURE WORK

A simple source encoding problem which illustrates the tradeoff between delay and efficient encoding has been posed. Two encoding techniques have been examined in detail. For the second of these, utilizing a flag, stability conditions were found and a Markov chain model was analyzed. In both cases it was known that as $P \rightarrow 1$ the delay approaches infinity. The paradox in this result is that for $P = 1$ the coding problem disappears.

The work can be continued in several directions. First of all the minimum delay over all coding schemes for P close to one is of interest. Also, the source model can be extended by considering more general arrival processes, (e.g. Poisson) and by considering several sources which must remain distinct. Again it appears that a Markov chain model can be used to describe a flag encoding scheme. Finally the subject of decoding delay for the flag scheme merits attention. When a bit forms part of a flag sequence there is an ambiguity at the receiver. The time required to resolve this ambiguity is a function of the state of the system.

ACKNOWLEDGEMENT

The authors are indebted to P. Humblet for pointing out Ref. 4.

REFERENCES

1. R.G. Gallager, "Basic limits on protocol information in data communication networks," IEEE Trans. Information Theory, vol. IT-22, July 1976.
2. P.A. Humblet, "Source coding for communication concentrators," MIT Report ESL-R-798, Jan. 1978.
3. W. Feller, "An Introduction to Probability Theory and Its Applications," vol. I, Wiley, 1950, pp. 299 ff.
4. P.J. Nielsen, "On the expected duration of a search for a fixed pattern in random data," IEEE Trans. Information Theory, vol. IT-19, No. 5, pp. 702-704, Sept. 1973.

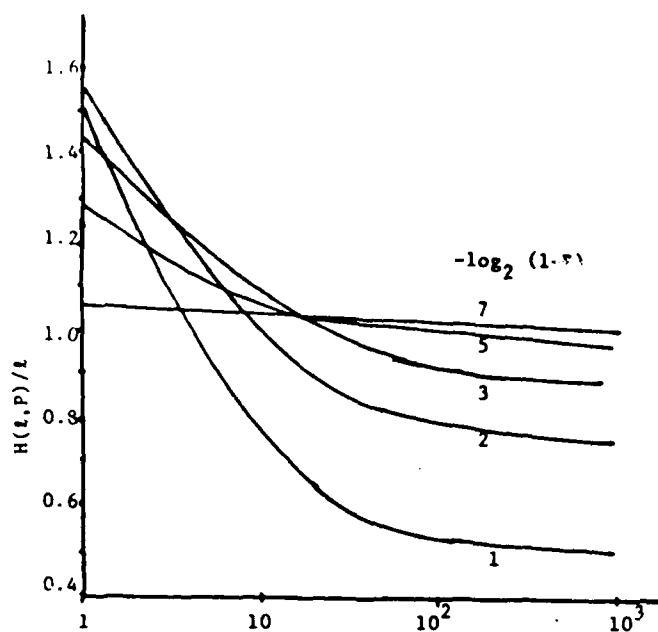


Figure 1 $H(L,P)/L$ vs Block Length

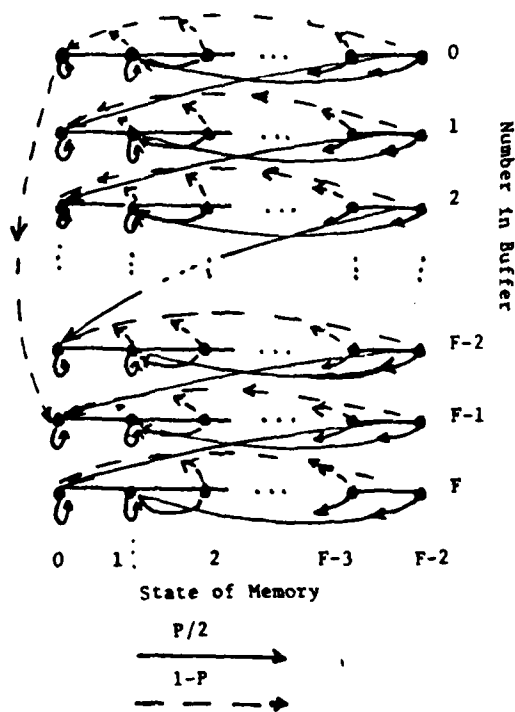


Figure 2 Markov Chain Model of Flag

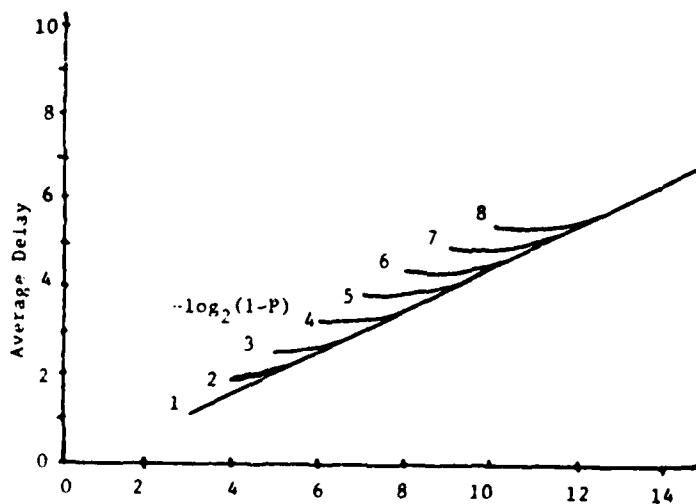


Figure 3 Average Bit Delay vs Flag Length

E.3 An Optimal Strategy for Packetization

Tsao, Kershenbaum, and Boorstyn

An Optimal Strategy for Packetization

By

Chan D. Tsao*
Robert R. Boorstyn**
Aaron Kershenbaum**

Abstract

We consider the problem of obtaining an optimal strategy for packetizing a data stream which enters a network one character at a time. Our objective is to maximize the number of users subject to a delay constraint which includes both queuing and packetization delay. We show that the optimal strategy is to packetize with probability $(1-q)$ after the arrival of N characters and to otherwise packetize at the arrival of the $N+1$ 'st character where N and q are functions of the delay constraint. A proof of optimality and numerical comparisons with other packetization strategies are given.

This work was supported in part by Grant 80-80-K-0579 from the U.S. Army CORADCOM. This work is a result of Chan D. Tsao's Ph.D. Thesis.

* Bell Telephone Laboratories

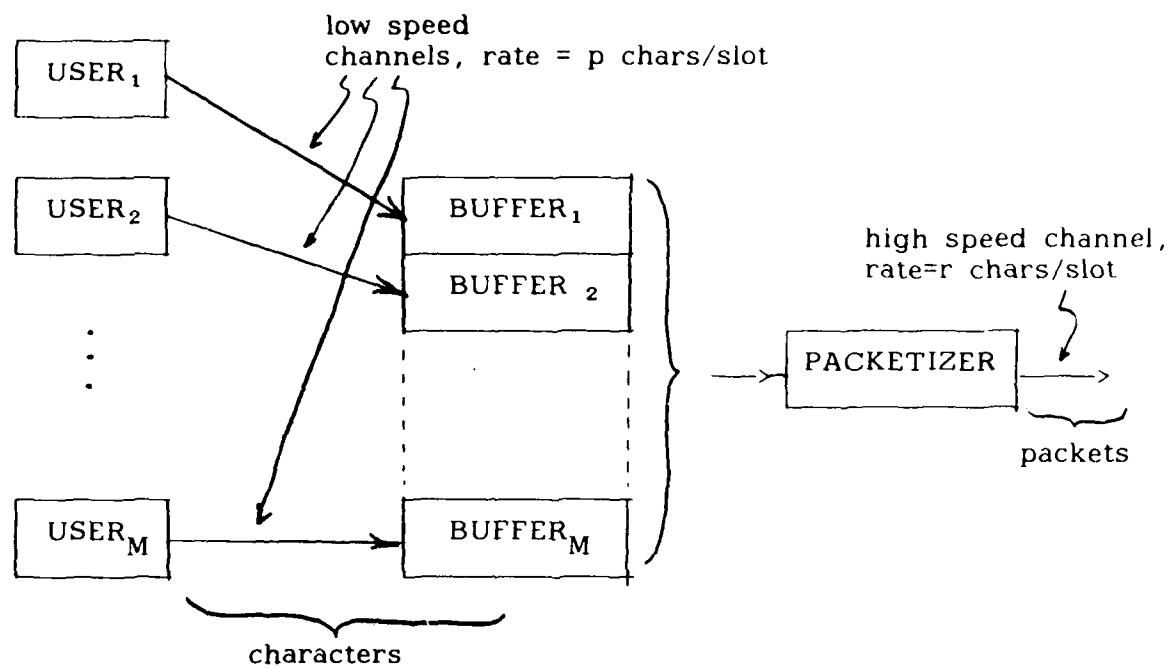
** Polytechnic Institute of New York

1. Introduction

Most packet switched networks are characterized by the "bursty" nature of the traffic that they carry, i.e., by the fact that the traffic sources deliver data to the network intermittently with a high peak-to-average rate. It is this phenomenon that makes packet switching, which gives the communication channel to a user only when he actually wants to use it, so attractive. Ironically, however, in the most extreme case of burstiness, when the sources generate data one character at a time, packet switching may become unacceptable because of the overhead due to the header which must be placed on each packet. Thus, if an H character header is used with only one data character per packet, an overhead of $100H\%$ is incurred. Typical values of H range between 6 and 32 in networks in operation today. If a random access mechanism, where users transmit at random times and contend for the channel is used, a significant number of additional synchronization bits must be added to the header and H may become as large as 60 in some cases. As random access becomes more prevalent, as it is in local area networks today, this problem becomes more severe.

An obvious solution to this problem is to buffer input characters until a reasonable number of them can be placed in a packet together. If N data characters are placed in a packet with an H character header, the overhead is reduced by a factor of N when compared with the case where only a single data character is placed in each packet. Now, however, one must also consider the effect of packetization delay. If we wait for N characters to arrive before packetizing, we incur a packetization delay roughly linear with N. If the input channel or the source is low speed and there is a tight constraint on delay, this

FIGURE 1 NETWORK MODEL



packetization delay is significant and must be considered when selecting the packetization strategy.

Such a situation arises in practice when many interactive users enter a packet switched network over low speed input lines and then share a high speed line within the network, as is shown in Figure 1.

There are M users, each of which produces an average of p characters per slot. The slot is defined to be the length of time it takes the low speed input line to transmit a character. Thus p is always between 0 and 1. We assume that the users are independent of each other and also that characters from a given user arrive independently. In each slot, each user produces a character with probability p . The high-speed network link shared by those users has a speed of r characters/slot.

In some applications, the initial network node can echo characters typed by the user, and the problem of trading network efficiency against delay perceived by the user can be avoided. Thus, the node will echo individual characters back to the terminal, thereby satisfying the user's constraint on delay, and also buffer a reasonable number of characters for inclusion in each packet, thereby keeping the overhead due to the packet header acceptably low.

In many applications, however, this approach is not feasible. For reliability, some users prefer an echo from the remote host to ensure that the data was not altered in transit through the network. In some cases, the host echos not just what is typed but also fills in default values for the user. An example of this is a system which will fill in the remainder of a keyword or file name once the user has entered an

unambiguous prefix (e.g., the user types "COP" and the system echos "COPY FROM" because the only command which starts with COP is the COPY command). Another reason for a remote echo is that the network node may be unable to echo characters on a selective basis and unable to handle the load of echoing all characters. Thus, the problem of handling sources which produce characters at a low rate and which require packetization within an amount of time comparable to the inter-arrival time of individual characters, is significant.

We thus consider a problem where we are given a constraint on delay, D (in units of slots), which includes both packetization and network delay. For simplicity, we consider the case where the network consists of a single link as shown in Figure 1. Other, more complex situations can be modeled by adjusting D to reflect the allowable delay after subtracting off other delays, e.g. host delay, and by adjusting the expression for network delay to reflect the presence of many network elements. This is beyond the scope of this paper and will not be considered any further.

Our objective is to maximize M , the number of users active at any time, subject to a constraint on D , the overall delay, by obtaining the optimal packetization strategy which keeps both the packetization and network delay small. Specifically, we seek to maximize M subject to a constraint on $\bar{D}$, the average per character delay, where $\bar{D}$ is given by:

$$\bar{D} = \bar{D}_{pk} + \bar{D}_s + \bar{D}_w$$

where $\bar{D}_{pk}$, $\bar{D}_s$, and $\bar{D}_w$ are the average per character delays due to packetization, service (transmission) and waiting (for the shared network link), respectively.

We develop a packetization strategy suitable for use in tightly constrained situations as described above; i.e., in situations when both channel efficiency and time delay are important. We prove the optimality of this strategy and offer computational evidence to show that our new strategy increases channel efficiency significantly, in some cases over 50%, when compared with simply using fixed length packets. Our strategy is also easily implemented and also offers the advantage of small variations in delay from packet to packet.

II. Delay and Packetization Models

We model a packetization scheme by a state transition diagram as shown in Figure 2. We consider only packetization schemes which form packets entirely from characters from a single user and furthermore restrict consideration to memoryless strategies, i.e., strategies which consider only what has happened since the last packet was formed in making the decision when to form the next packet.

Figure 2 can be thought of as the state of the buffer corresponding to a single user. The initial state (at the top of the diagram) corresponds to the buffer being empty immediately after a packet is formed. Transitions in the diagram correspond to state changes over a single slot of width Δ . With probability p , a character arrives and a transition to the left to a shaded node occurs. With probability $1-p$, no character arrives and a transition to the right occurs. If a packet is formed, as at node j in Figure 2, the transition is upward to a state immediately following the formation of a packet.

Each state, i , has associated with it n_i , the number of characters in the buffer; d_i the aggregate packetization delay suffered by all these characters; and r_i , the probability of reaching state i . The initial state, 0 , has $n_0=0$, $d_0=0$, and $r_0=1$. Figure 3 shows the general situation for a downward transition (i.e., from a state in which a packet is not formed. Note that the relations between the quantities at nodes j , k and l are not functions of whether or not a character arrived at node j .

The packetization rule can be thought of as choosing which nodes in this tree to packetize at. Indeed, any memoryless rule which forms single user packets can be so represented.

FIGURE 2
STATE TRANSITION DIAGRAM
CORRESPONDING TO A PACKETIZATION SCHEME

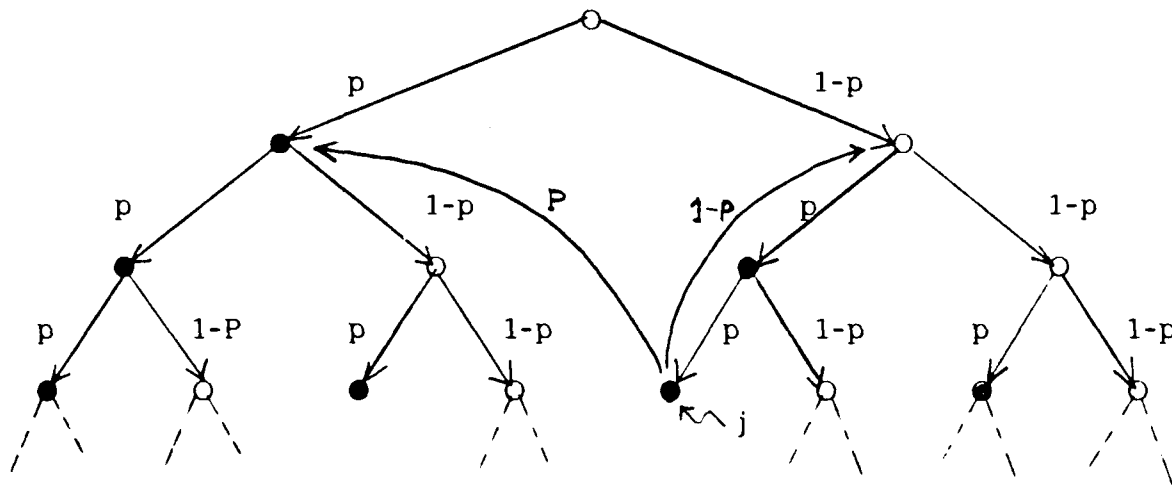
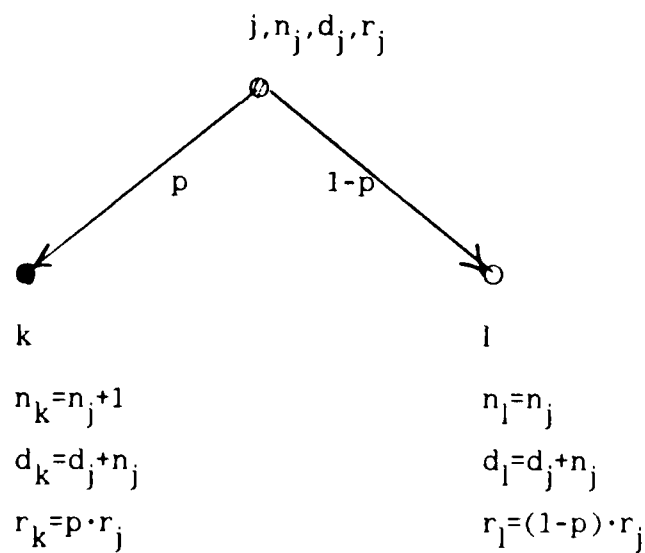


FIGURE 3
INCREMENTAL EFFECT OF NOT PACKETIZING
AT NODE j



A. Packetization Delay

If the packetization scheme results in a packet length distribution of

$$f_i = \text{Prob \{Packet length is } i \text{ characters\}}$$

then $\bar{D}_{pk}$, the average per character packetization delay, is given by

$$\bar{D}_{pk} = \sum_i \frac{f_i \bar{d}_i}{\bar{y}}$$

where $\bar{d}_i$ is the total average packetization delay suffered by all characters in a packet of length i and $\bar{y}$ is the average packet length which is given by

$$\bar{y} = \sum_i i f_i$$

Since characters arrive independently at a rate of p characters per slot, the average interarrival time between characters is $\frac{1}{p}$ slots and $\bar{d}_i$ obeys the recurrence relation

$$\bar{d}_i = \bar{d}_{i-1} + \frac{i-1}{p}$$

This, together with the initial condition $\bar{d}_0=0$ yields

$$\bar{d}_i = \frac{i(i-1)}{2p}$$

Thus, for example, if the scheme resulted in a 2-point distribution with $f_n=1-q$ and $f_{n+1}=q$ then

$$\bar{y} = n+q$$

$$\begin{aligned}\bar{D}_{pk} &= \frac{(1-q)n(n-1) + q(n+1)n}{(n+q)2p} \\ &= \frac{\bar{y}-1}{2p} + \frac{q(1-q)}{2p\bar{y}}\end{aligned}$$

This function is drawn in Figure 4. As can be seen, the function is continuous and piecewise-differentiable. It behaves roughly like the linear function $\frac{\bar{y}-1}{2p}$ and indeed matches that function for $q=0$ and $q=1$. The difference between the two functions is greatest near $q=n[(1+\frac{1}{n})^{\frac{1}{2}}-1]$. This difference decreases as $\bar{y}$ increases.

B. Service Time

The average service time per character, $\bar{D}_s$, is given by

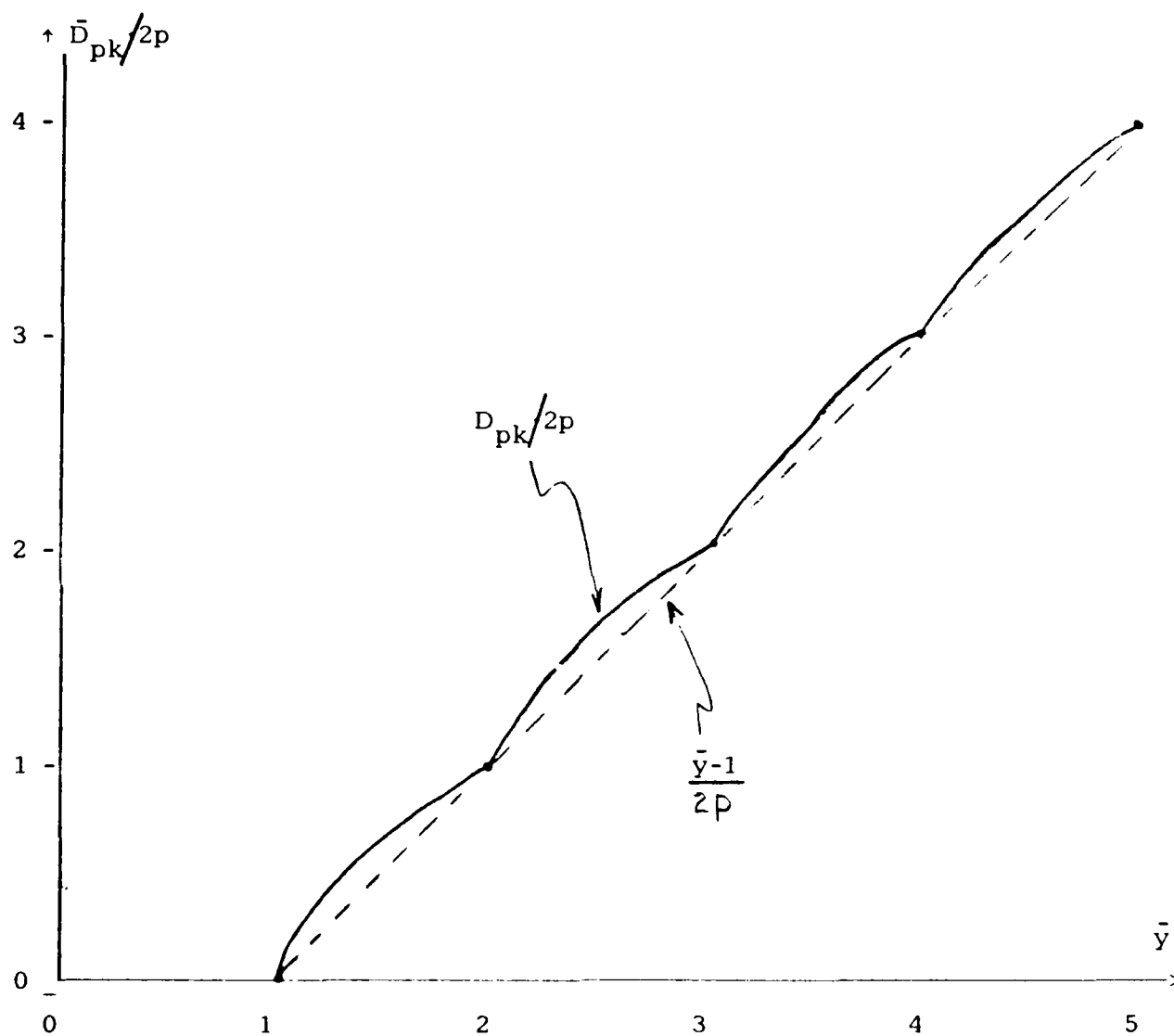
$$\bar{D}_s = \frac{1}{r} \left(\sum_i \frac{i^2 f_i}{\bar{y}} \right) + H$$

where r is the speed of the high-speed lines in characters/slot and H is the number of characters in the packet header. For the 2-point distribution above,

$$\bar{D}_s = \frac{1}{r} \left[\bar{y} + H + \frac{q(1-q)}{\bar{y}} \right]$$

Thus we see that $\bar{D}_s$ behaves in a manner very similar to $\bar{D}_{pk}$. It is continuous, piecewise-differentiable and follows a linear function of $\bar{y}$ very closely, the difference getting smaller as $\bar{y}$ increases.

FIGURE 4
AVERAGE PACKETIZATION DELAY VERSUS
AVERAGE PACKET LENGTH



C. Waiting Time

The exact relationship between the character arrival rate, packetization strategy and $\bar{D}_w$, the average per character waiting time for the high speed channel, is complex. We believe a reasonable approximation to this delay is given by approximating $\bar{D}_w$ by the waiting time in an M/G/1 queuing system. Since the character arrival process is Poisson, this seems reasonable. Thus, $\bar{D}_w$ is approximated by

$$\bar{D}_w = \frac{m\lambda\bar{x}^2}{2(1-\rho)}$$

where M is the number of users

λ is the average arrival rate per user in packets per slot

$\bar{x}^2$ is the second moment of the service time for a packet

and ρ is the average channel utilization.

For the system at hand,

$$\lambda = \frac{p}{\bar{y}}$$

$$\rho = \frac{pM}{r} \left(\frac{\bar{y}+H}{\bar{y}} \right)$$

$$\bar{x}^2 = \sum_i \frac{f_i(i+H)^2}{r^2}$$

For the 2-point distribution above,

$$\bar{x}^2 = \frac{1}{r^2} [(\bar{y}+H)^2 + q(1-q)]$$

and

$$\bar{D}_w = \frac{Mp \left[\frac{(\bar{y}+H)^2}{\bar{y}} + \frac{q(1-q)}{\bar{y}} \right]}{2r^2 \left[1 - \frac{pM}{r} \frac{\bar{y}+H}{\bar{y}} \right]}$$

D. Total Delay

From the above, we have a general expression for $\bar{D}$, the average per character delay in slots,

$$\bar{D} = \sum_i f_i \left[\frac{i(i-1)}{2p\bar{y}} + \frac{i^2/\bar{y}+H}{r} + \frac{M\lambda(i+H)^2}{2r^2(1-\rho)} \right] \quad (*)$$

$$\bar{D}_w = \frac{pM \left[\frac{(\bar{y}+H)^2}{\bar{y}} + \frac{q(1-q)}{\bar{y}} \right]}{2r^2 \left[\bar{y} - \frac{pM(\bar{y}+H)}{r} \right]}$$

III. Derivation of the Optimum Packetization Strategy

We now consider the problem of maximizing M , the number of users, subject to a constraint, D^* , on the average per character total delay. We thus seek the optimal value of the packet length, $\bar{y}$, and the optimal packetization scheme which will yield this optimal length.

Given M and $\bar{y}$, we see from equation (*) that the packetization scheme that minimizes $\bar{D}$ is one which minimizes the second moment of the packet length, $\sum_i i^2 f_i$. We are constrained, of course, to packet length distributions where the packet length only takes integer values. It is easy to see that the distribution which minimizes the second moment subject to these constraints is the 2-point distribution which we have been discussing; i.e.,

$$f_i = \begin{cases} 1-q & i = n \\ q & i = n+1 \\ 0 & \text{otherwise} \end{cases}$$

where $\bar{y} = n+q$. To see this, we use the technique of Lagrange multipliers [9] to find the minimum of

$$\sum_i i^2 f_i + \alpha \sum_i i f_i + \beta \sum_i f_i \text{ where}$$

α and β are the Lagrange multipliers. The last two terms are used to satisfy the constraints $\sum_i i f_i = \bar{y}$ and $\sum_i f_i = 1$. In addition, all $f_i \geq 0$. Differentiating with respect to the f_i we get

$$\begin{aligned} i^2 + \alpha i + \beta &= 0 \text{ for all } i \text{ such that } f_i > 0 \\ &\geq 0 \text{ for all } i \text{ such that } f_i = 0 \end{aligned}$$

Thus there are at most two non-zero values of f_i that can satisfy the

equality condition above. To satisfy the inequality condition these two values of i must differ by one.

Thus we see that for a given value of $\bar{y}$ and M , the total delay is minimized by any packetization scheme which gives rise to a 2-point packet length distribution.

We see from this that we need only consider 2-point packet length distributions in order to obtain an optimal packetization strategy since minimizing delay for a given number of users and maximizing the number of users for a given delay are complementary problems. The question of how to find the average packet length, $\bar{y}$, still remains and will be discussed below.

A similar result, proving that a 2-point packet length distribution is optimal, can also be obtained by approaching the problem in a different way by considering packetization delay and service time together. Any packetization scheme can be thought of as a decision rule which, given the state of the buffers, decides whether to packetize or not. Figure 2 shows the state space for this decision process.

Tracing a path from the root to any node, j , one can determine the probability, r_j , of reaching that state by multiplying all of the link probabilities. One can also find the number of characters, n_j , in the buffer by counting the number of shaded nodes in the path from the root to node j . Finally, one can determine d_j , the total delay due to packetization and service time suffered by all characters in the buffer, if a packet is formed at node j . This is done by adding together the delays of all the characters in the packet.

Let x_j be a decision variable taking the value 1 if we packetize at node j and zero if we do not. Any packetization scheme can be defined

solely in terms of the x_j . Furthermore,

$$\bar{y} = \sum_j x_j r_j n_j$$

and

$$\bar{D}_1 = \sum_j x_j r_j d_j$$

where $\bar{D}_1$ is the average packetization and service delay for all characters. We seek to maximize $\bar{y}$ subject to the constraint $\bar{D}_1/\bar{y} \leq D^*$. There is also an additional constraint that the packetization scheme be consistent. In particular, $x_j = 1$ implies that $x_k = 0$ for all successors k of node j since once we packetize at node j we will never reach node k .

It is somewhat easier to see what the optimal policy is if we consider the decision process in a slightly different way. We define a new set of decision variables, w_j , associated with the nodes j . The w_j are related to the x_j above by the rule that if $x_j = 1$ then $w_i = 1$ for all predecessors of j and $w_i = 0$ otherwise. For all consistent values of x_j , the w_i are well defined. Setting $w_i = 1$ can be thought of as deciding not to packetize at node i .

We now define g_j as the increment in packet length due to the decision not to packetize at node j . Similarly we define h_j as the increment in the average total delay, $\bar{D}_1$.

The expressions for $\bar{y}$ and $\bar{D}_1$ can then be rewritten in terms of the new decision variables, w_j , as follows:

$$\bar{y} = \sum_j w_j g_j$$

$$\bar{D}_1 = \sum_j w_j h_j$$

The incremental effect of not packetizing at node j is shown in Figure 3. As can be seen if $w_j = 1$, two nodes, k and l , will be counted instead of node i in the sums for $\bar{y}$ and $\bar{D}_1$. Thus:

$$g_j = (1-q)r_j$$

$$h_j = (n_j + \frac{2n_j + H + 1}{c})r_j$$

In order to maximize $\bar{y}$ for a given D we would prefer to choose nodes j with maximal g_j/h_j , i.e. nodes which maximize the incremental gain in $\bar{y}$ per unit $\bar{D}_1$. Note that in reality it is $\bar{D}_1/\bar{y}$ not $\bar{D}_1$ which is the constraint but since we seek to maximize $\bar{y}$ we are simultaneously loosening the constraint as much as possible.

In order to maximize g_j/h_j we choose nodes j with the smallest possible n_j and set $w_j = 1$ for those nodes. We continue to pick nodes until we find that setting the next $w_j = 1$ would cause $\bar{D}_1/\bar{y}$ to exceed D^* . Until this point, we have assumed for the sake of simplicity that w_j would only take the values 0 or 1. In fact we are free to assign any value in the range 0 to 1 to each w_j . At the point where setting $w_j = 1$ would violate the constraint on $\bar{D}_1$, there exists a value, a , of w_j which would cause the constraint to be satisfied exactly, since $D^*\bar{y}$ is a monotone function of w_j . Thus, we set $w_j = a$ for this last j and satisfy the constraint with equality.

It is clear that this procedure will indeed find an optimal set of values for the w_j in the sense that $\bar{y}$ is maximized. To see this we need only note that in order to increase the value of any w_j above the value set by this procedure we would have to correspondingly decrease

the value of some other w_k with a resulting decrease (or at best no change) in $\bar{y}$. This is because we are selecting the nodes, j , in order of g_j/h_j , largest first.

Note that at any stage there is an infinite number of nodes j with the same value of n_j to choose from. Any selection will yield exactly the same value of $\bar{y}$. Indeed, one need only ask if w_j can be set to 1 for all nodes with a particular n_j . If so, we do this before asking the question for the next larger value of n_j . Thus we avoid any conceptual difficulty with the finiteness of the decision process.

We are left then only with the question the consistency of the packetization policy. This too, however, is no problem. Note that if we set $w_j = 1$ if $n_j < n^*$ and also for a collection of nodes with $n_j = n^*$ and with aggregate probability $1-q$ then we have a consistent packetization policy which adheres to the optimization criteria. This policy can be simply stated, and equivalently implemented, as waiting until the n^{*th} character arrives, flipping a coin which comes up heads with probability q , packetizing at that point if the coin comes up tails, and packetizing after the next character arrives otherwise.

We have thus shown, again, that a 2-point distribution minimizes delay due to packetization and service time. Given M , the number of users, this also minimizes queuing delay, as shown above. We now turn to the final question, how to determine $\bar{y}$, the optimal average packet length, in practice.

IV. Computation of the Optimal Packet Length

Examination of the expressions for the components of delay given in Section II shows that grossly, packetization delay and service time grow linearly with packet length while queuing packetization delay decreases as the packet length increases (since the overhead due to the header, which contributes to the utilization, decreases). There is, however, a term proportional to $q(1-q)$ (where q is the fractional part of the packet length) present in each delay component. This term causes the derivative of delay with respect to packet length to be discontinuous and prevents us from simply doing a gradient search to find the optimal value of $\bar{y}$. As can be seen by examining Figure 2, however, one might expect a simple search to yield a reasonable approximation to $\bar{y}$ since the $q(1-q)$ term is small and becomes even less significant as $\bar{y}$ increases. We are thus led to the following approach.

Ignoring the $q(1-q)$ terms in $\bar{D}$, we can find $\hat{y}$, an approximate value of $\bar{y}$, using a gradient search or other simple search procedure. Thus we let

$$\hat{D} = \frac{\bar{y}-1}{2p} + \frac{1}{r} (y+H) + \frac{\frac{Mp}{\bar{y}} (y+H)^2}{2r^2 [1 - \frac{Mp}{r} (\frac{y+H}{\bar{y}})]}$$

and

$$\frac{d\hat{D}}{dy} = \frac{1}{2p} + \frac{1}{r} + \frac{Mp(\frac{y+H}{\bar{y}}) \{ (1 - \frac{Mp}{r} (\frac{y+H}{\bar{y}})) (\frac{y+H}{\bar{y}}) - \frac{2H}{\bar{y}} \}}{2r^2 [1 - \frac{Mp}{r} (\frac{y+H}{\bar{y}})]^2}$$

Given M , we can then minimize $\hat{D}$ by finding $\hat{y}$, the value of y , where the derivative is 0. Given $\hat{D}$, the value of M can then be adjusted since $\hat{D}$ is monotone with M ; i.e., if $\hat{D}$ is less (greater) than the required

delay, then M can be increased (decreased). A binary search on M can thus be performed. A simple initial upper limit on M for this purpose is $M=r/p$. Given these approximate values of y and M , it is then possible to proceed in practice using the real values of D including the $q(1-q)$ terms. The derivative is now only continuous over individual integral values of $\bar{y}$ and so, strictly speaking, a separate search must be performed over each such interval. In practice, at most 2 such intervals would have to be searched once the above approximate values were determined since the $q(1-q)$ terms do not alter the result much. This is verified empirically by the computational experience presented below.

V. Computational Experience

We now examine the effect in practice of using this optimal packetization scheme rather than simply forming fixed length packets. This issue, along with a comparison with other strategies, is examined in more detail in [1]. A simple optimization procedure was implemented for this purpose, and is described below.

The search procedure used is shown in Figure 5. The variables y and M are the current values of the packet length and number of users being examined, respectively. The variables y^* and M^* are the current estimates of the optimal values of y and M . D_{req} is the delay requirement in slots. The variables Δ and T control the length and precision of the search and were set to .01 and 2, respectively. We are interested primarily in cases where y is small and so this simple linear search is quite reasonable. For large y the procedure described in the preceding section could be implemented or, more simply, larger values of Δ and T could be used with this procedure since small differences in y would not alter M .

We examined problems with p , the number of characters per user per slot, of $\frac{1}{4}$ and r , the transmission speed of the high-speed channel in characters per slot, of 300. This corresponds to users producing characters at an average rate of 1 per second on a low-speed channel with peak rate 4 characters per second and sharing a high-speed channel of 1200 characters per second. Values of H , the packet header size, of 1, 6, and 32 were examined.

FIGURE 5 SIMPLIFIED SEARCH PROCEDURE

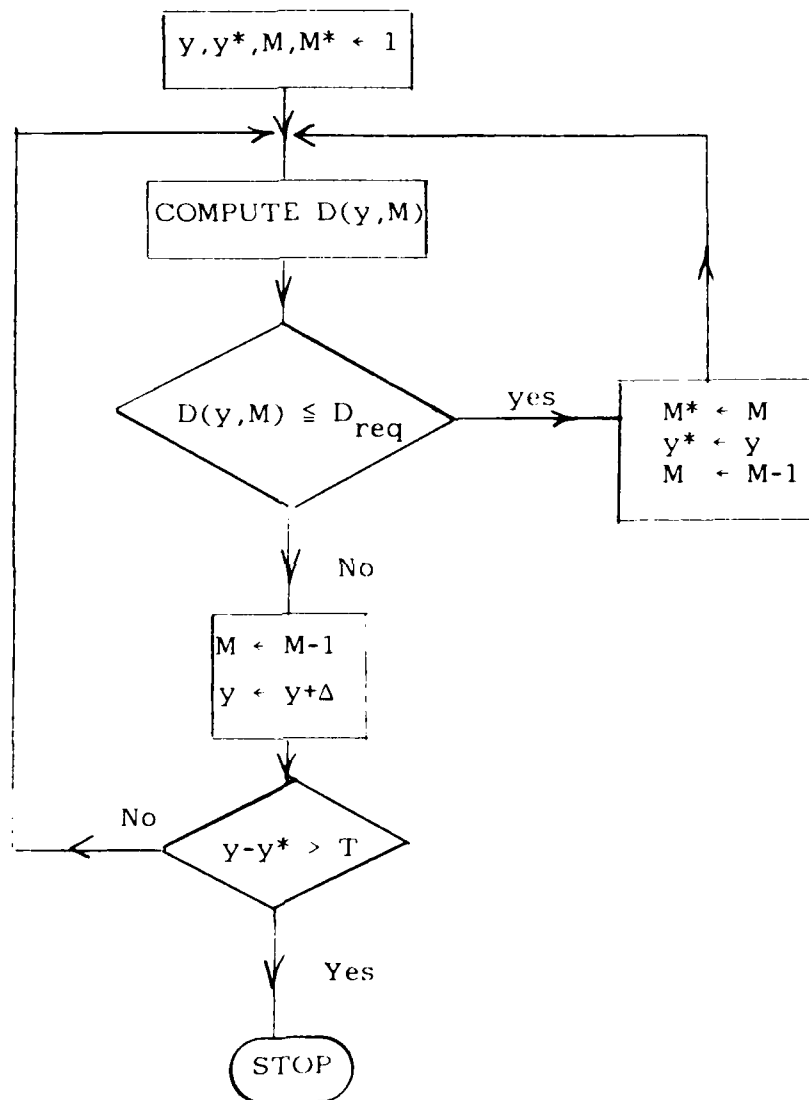


TABLE 1
COMPARISON BETWEEN OPTIMAL AND FIXED SCHEMES

$D_{\text{req(slots)}}$	γ^*	M^*	γ_{fixed}	M_{fixed}	% difference	M_{UE}
.1	1.00	579	1	579	0.0	614
.2	1.01	590	1	589	0.2	628
.5	1.09	612	1	595	2.9	666
1.0	1.26	653	1	597	9.4	720
1.5	1.50	701	1	598	17.2	764
2.0	1.83	755	1	598	26.2	800
2.25	2.00	783	2	783	0.0	816
3	2.26	816	2	795	2.6	857
4	2.77	864	2	797	8.4	900
4.25	3.00	875	3	875	0.0	909
5	3.21	901	3	893	.9	933
6	3.70	929	3	896	3.7	960
6.5	4.00	943	4	943	0.0	971
7	4.13	953	4	952	0.1	981
10.	5.53	1003	5	994	0.9	1028

Table 1a. (H=1)

<u>D_{reg}</u>	<u>y*</u>	<u>M*</u>	<u>y_{fixed}</u>	<u>M_{fixed}</u>	<u>%difference</u>	<u>M_{UB}</u>
0.1	1.00	148	1	148	0.0	179
0.5	1.07	172	1	167	2.9	206
1.0	1.24	194	1	169	14.8	240
2.0	1.81	259	1	170	52.3	300
2.5	2.06	293	2	291	0.7	327
4.0	2.78	361	2	297	21.5	400
4.5	3.02	387	3	387	0.0	423
10.0	5.64	563	5	540	4.3	600

Table 1b. (H=6)

<u>D_{reg}</u>	<u>y*</u>	<u>M*</u>	<u>y_{fixed}</u>	<u>M_{fixed}</u>	<u>%difference</u>	<u>M_{UB}</u>
0.2	1.00	22	1	22	0.0	39
1.0	1.16	36	1	34	5.8	53
2.0	1.12	49	1	35	40.0	74
3.0	2.13	67	2	66	1.5	86
10.0	5.49	163	5	157	3.8	189

Table 1c. (H=32)

FIGURE 6 NUMBER OF USERS vs. DELAY



Table 1 shows the results of a comparison between the optimal scheme and fixed length packets. The column labeled "%-difference" shows the percentage improvement in M using optimal packet lengths (y^*) rather than the best fixed length (y_{fixed}). Figure 6 shows this comparison for $H=6$. As can be seen, the improvement is quite significant in some cases. In particular, this difference is most significant for small delay and large H .

Note that the difference in performance is not monotone. In particular, as the allowable delay increases, y^* increases. As y^* takes a larger fractional part, M^* increases. The fixed length scheme, however, cannot do this and so M_{fixed} increases only very slightly. The rise in M^* with increasing y^* is due to the decrease in header overhead as the packet length increases. The small increase in M_{fixed} is due to the slight increase in utilization permitted by increasing the allowable delay.

The column labeled M_{UB} in Table 1 and shown in Figure 1 is an upper bound on M computed from the simple observations that the packetization delay must be smaller than the total allowable delay, D , and the packetization delay is at least $\frac{y-1}{2p}$. This puts an upper bound on y :

$$y \leq 2p D + 1 \quad (2)$$

An upper bound on M is then obtained by observing that the utilization of the high speed channel is bounded by 1:

$$M \leq \frac{r}{p} \frac{y}{y+H} \quad (3)$$

Thus

$$M \leq \frac{r}{p} \frac{2pD+1}{2pD+1+H}$$

As we can see, this bound is tightest for large D and small H . It is sometimes useful in quickly estimating M^* to compare it with M_{fixed} which is easily estimated by equations 2 and 3 and restricting y to integers.

VI. Conclusions

We have presented a packetization strategy along with a proof of its optimality. The technique is easily implemented in practice. A comparison with fixed length packets is presented and the optimal technique exhibits significantly higher throughputs, in some cases over 50% higher. We conclude therefore that the packetization strategy should be used, in practice, especially when the delay constraint is tight and the header size is large.

References

1. C.D. Tsao, "A Comparison of Switching Techniques," Ph.D. Thesis, Polytechnic Institute of New York, 1982.
2. R. Boorstyn and J. Hayes, "Delay and Overhead in the Encoding of Bursty Sources," International Conference on Communications, IEEE, Seattle, June 1980.
3. D. Doll, "Multiplexing and Concentration," Proceeding of the IEEE, Vol. 60, November 1972.
4. W. Feller, An Introduction to Probability Theory and Its Applications, Vol. 1, John Wiley & Sons, Inc., 1968.
5. L. Kleinrock, Communication Nets Stochastic Message Flow and Delay, McGraw Hill Company, pp. 49-53, 1964.
6. L. Kleinrock, Queueing Systems Volume I: Theory, John Wiley and Sons, Inc., 1975.
7. A.G. Konheim and B. Meister, "Waiting Lines and Times in System With Polling," JACM, 21, No. 2, pp. 470-490, July 1974.
8. L. Takacks, "Two Queues Attended by a Single Server," J. Oper. Res. Soc. America, 16, 3, p. 639, 1968.
9. S. Shapiro, Mathematical Programming, Prentice Hall, 1979.

E.4 Optimal Fixed Frame Multiplexing in Integrated Line-
and Packet-Switched Communication Networks

Maglaris and Schwartz

IEEE Transactions on Information Theory, March 1982

Optimal Fixed Frame Multiplexing in Integrated Line- and Packet-Switched Communication Networks

BASIL S. MAGLARIS, MEMBER, IEEE, AND MISCHA SCHWARTZ, FELLOW, IEEE

Abstract—Recently, emphasis has been placed on integrated communication facilities capable of handling both line-switching and packet-switching digital traffic. The problem of dynamically allocating the bandwidth of a trunk to both types of traffic is formulated as a Markovian decision process. Line switching is modeled as a time division multiplexing loss scheme over a varying portion of a fixed time frame. Packet-switching traffic is served through the remaining portion of the frame and requires queueing at the multiplexer-concentrator. Two different cost criteria are examined involving probability of blocking for line switching and average queueing delay for packets. The corresponding optimization problems are presented under reasonable simplifying assumptions. The movable boundary scheme suggested for commercial implementation of integrated multiplexers is shown to offer optimal or near-optimal performance.

I. INTRODUCTION

IN ORDER to multiplex and concentrate different data-sending facilities over a large bandwidth channel, a choice of a switching technique must be made. Line switching is used for transmission of long messages and requires a permanent connection between the communicating ends for the duration of the session. This connection can be a physical line, a synchronous slot allocation in a time division multiplexing scheme, or a frequency band allocation in frequency division multiplexing. Initial setup times must be short compared to the message length for efficient channel utilization. Most of the actual line-switched systems are loss systems, i.e., a request for a new line is blocked when no bandwidth is available, and a new trial must be made later. The probability of blocking a new request is a measure of the performance of the system.

For short interactive types of traffic, messages arriving at the concentrator wait in a queue (either as whole messages or divided into smaller portions called packets) and are served according to some discipline. A header in the message or packet is required containing identification and destination information. Routing and reassembly of messages at the destination node are additional problems in

such systems. A measure of the performance of the system is the average queueing delay experienced by an average message.

Closs in [1] presented a comparative evaluation of line- and packet-switching lossless schemes and showed the superiority of the former for lengthy message traffic and of the latter for bursty short packet traffic. This basic result has been further demonstrated in [2]–[4] for various classes of models.

In a more general-purpose digital data communication network, it is desirable to integrate both line- and packet-switching facilities. The user or the network manager may decide on the most suitable discipline. This integrated facility can accommodate simultaneously line-switching traffic such as voice traffic or computer batch traffic, and packet-switching data traffic, such as bursty terminal-to-host traffic.

Various schemes have been studied and implemented. Fixed frame fixed boundary schemes allocate a predetermined portion of a time frame (consisting of a given number of constant duration slots) to each type of traffic. Fixed frame movable boundary schemes improve the packet traffic throughput by introducing some intelligence to the previous scheme. Packet traffic is allowed to occupy any idle slot of the line switching portion of the frame. The movable boundary scheme was first analyzed by Kummerle [5] and an implementation research project has been undertaken by IBM Zurich [6]–[8]. The proposed architecture is summarized in [9].

A parallel effort was motivated by the U.S. Department of Defense aimed at replacing the AUTOVON line-switched voice network and AUTODIN packet-switched data network by an integrated structure. Various alternatives have been considered by the Defense Communications Agency (DCA) and its contractors. These include either movable boundary protocols [10]–[12] or the use of packetized voice techniques to eliminate line switching [13]–[15]. Commercial vendors (such as TRAN Corporation [16] and Codex Corporation [17]) have announced integrated approaches. The Codex approach uses variable frame multiplexing with the frame size adjusted to the traffic variations. Analyses of variable frame integrated multiplexers can be found in [18] and [19].

The growing interest in integrated line- and packet-switched networks, as demonstrated in the foregoing survey,

Manuscript received February 5, 1980; revised February 6, 1981. This work was supported by the National Science Foundation under Grants ENG-74-17152 and ENG-78-08260. Part of this paper was presented at ICC 79, Boston, MA, June 1979.

B. S. Maglaris was with the Network Analysis Corporation, Great Neck, NY. He is now with the Department of Electrical Engineering and Computer Science, Polytechnic Institute of New York, 333 Jay Street, Brooklyn, NY 11201.

M. Schwartz is with the Department of Electrical Engineering, Columbia University, New York, NY 10027.

and specifically the efforts to improve the efficiency of these systems in terms of the trunk capacity utilization, led to the study presented in this paper. Its aim is to investigate the limit of the capacity utilization improvement under a basic constraint: fixed frame size and synchronous service provision to line-switching sessions once they get access to the system. A loss model is assumed for the line-switching traffic and a first-in-first-out queueing model for the fixed size packet traffic. The problem of optimal allocation of frame slots is formulated as a Markovian decision process. Two optimality criteria are examined. Under the first, the average time delay for the packet traffic is minimized with the probability of blocking a new line-switching arrival constrained to be below a given level. A linear programming method is used to determine the optimal policy under a birth-death process model for the line-switching traffic. Under the second criterion, a linear combination of the average time delay and the probability of blocking is minimized. A policy improvement algorithm is used to obtain the optimal policies both under the same birth-death assumption and for the general case of multiple arrivals or completions in a frame. It is shown that under the birth-death assumption, the movable boundary scheme is in some cases optimum or close to optimum in the examples studied. Similar results are obtained for the general case.

II. THE MODEL

A. Input Traffic

Line-switching traffic is assumed to consist of messages arriving with a Poisson rate of λ_1 messages/s. Message length is exponentially distributed with mean $1/\mu$ s. A new arrival may either be accepted or blocked. Packet-switching traffic consists of fixed length packets arriving at a Poisson rate of λ_2 packets/s and stored in a finite queue of capacity Q packets. Q is assumed to be large enough to result in negligible packet overflow.

B. Output Trunk

The incoming traffic of both types is multiplexed onto a main trunk of capacity C packets/s. A frame of b s duration, divided into M slots as in Fig. 1, accepts both the line-switching and packet-switching traffic. The slot size is chosen to accommodate exactly one packet. Hence $C = M/b$. S_j slots in the j th frame (Fig. 1) are allocated to the line-switching messages, one slot per message, in a time division multiplexing mode (every frame processes a fraction of a message equal to the packet or slot size). The multiplexer must guarantee no interruption of synchronous service to the message already in the system. A new setup call is stored and considered for acceptance to the trunk at the beginning of a new frame. The remaining $M - S_j = N_j$ slots are allocated to the packets present in the packet queue at the beginning of the j th frame, in a first-in-first-out mode. The object is to determine the optimum allocation

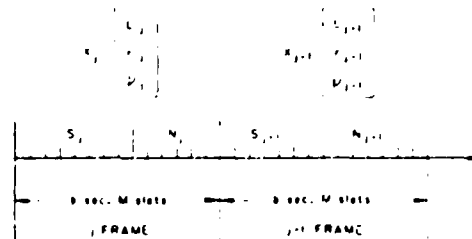


Fig. 1 Integrated frame structure.

tion of S_j and N_j to circuit and packet traffic, respectively. As noted earlier, two optimality criteria are investigated for this purpose.

C. Decision Mechanism—State of the System

We define the state of the system X_j at the time instant just prior to the opening of the j th frame as follows:

$$X_j \triangleq \begin{bmatrix} L_j \\ r_j \\ v_j \end{bmatrix}$$

where

- L_j number of packets waiting in the packet queue,
- r_j number of messages already in the system and requiring continuation of service, and
- v_j number of new setup calls during the $j - 1$ frame.

The multiplexer decides on S_j , N_j based on X_j , the previous history of the system, and some optimizing rule. Continuation of service for the line-switching traffic must be guaranteed and any portion of the frame not used by long messages must be allocated to packet-switching traffic. Hence $r_j \leq S_j \leq \min(r_j + v_j, M)$.

The transition probabilities from a present state X_j to a new state X_{j+1} depend only on X_j and the number of slots allocated to the line-switching traffic of the j th frame S_j . In order to derive them, we need the probability distributions of the components of state X_{j+1} . These are given by (1)–(3) below and are obtained by noting the following.

- v_{j+1} Number of new setup calls during the j th frame. The number follows a Poisson distribution with mean $\lambda_1 b$. (Equation (1) follows directly.)
- r_{j+1} Number of messages requiring continuation of service among the S_j messages currently in the system. Since we assumed that the message lengths were exponential, r_{j+1} depends only on S_j . (This is due to the memoryless property of the exponential distribution.) The binomial distribution of (2) follows directly. (The probability of a completion in a frame b s long is $1 - e^{-\mu b}$.)
- L_{j+1} Depends on L_j and S_j according to the transition equations of a finite queue with Poisson arrivals $\lambda_2 b$, constant service b , and $N_j = M - S_j$ servers.

This leads to (3).

$$P(\nu_{j+1}) = \frac{(\lambda_1 b)^{\nu_{j+1}}}{\nu_{j+1}!} e^{-\lambda_1 b}. \quad (1)$$

$$P_{S_j}(r_{j+1}) = P_{S_j}(r_{j+1} | X_j) \\ = \left(\frac{S_j}{r_{j+1}} \right) (1 - e^{-\mu b})^{S_j - r_{j+1}} (e^{-\mu b})^{r_{j+1}}. \quad (2)$$

$$P_{S_j}(L_{j+1} | L_j) = \begin{cases} \frac{(\lambda_2 b)^{L_{j+1}}}{L_{j+1}!} e^{-\lambda_2 b}, & \text{if } L_j \leq N_j = M - S_j, \\ \frac{(\lambda_2 b)^{L_{j+1} - L_j + N_j}}{(L_{j+1} - L_j + N_j)!} e^{-\lambda_2 b}, & \text{if } N_j \leq L_j \leq L_{j+1} + N_j, \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

for all $L_{j+1} < Q$, $L_j \leq Q$, and $P_{S_j}(Q | L_j) = 1 - \sum_{k=0}^{Q-1} P_{S_j}(L_{j+1} = k | L_j)$. (Recall that $L_{j+1} \leq Q$.)

Since each of the ν_{j+1} , r_{j+1} , L_{j+1} random variables depends only on the previous combination, we conclude that the state transition probabilities are given by

$$P_{S_j}(X_{j+1} | X_j) = P(X_{j+1} | X_j, S_j) \\ = P(\nu_{j+1}) P_{S_j}(r_{j+1}) P_{S_j}(L_{j+1} | L_j). \quad (4)$$

Depending on the length of the frame b and the arrival rate λ_1 , we may define, within any acceptable level of accuracy, the maximum number of new calls per frame CMAX such that $P(\nu_j > \text{CMAX}) \approx 0$. Thus ν_j , r_j , L_j may assume a finite number of values each, and the process has a finite state space.

The transitions (4) define a finite state, discrete time Markov process with a decision mechanism controlling the evolution of the process. Note that the vector state representation does not lead to the erroneous implicit independence of its components as in [20], reported in [21]. The theory of Markovian decision processes (see [22]) provides us with the analytical and computational tools for the evaluation of an optimal policy π , i.e., an assignment rule, possibly probabilistic, of an action S_j (the number of slots to be assigned to line-switched traffic) to a state X_j based on the history of the process.

III. FIRST OPTIMALITY CRITERION

The first optimality criterion we consider is that of minimizing the average packet queueing delay Td with the probability PI of blocking newly arriving line-switching calls constrained to be no more than a specified acceptable level PLOSS.

By Little's formula [23] the packet delay is proportional to the average packet queue size $E(L)$. The policy of assignment rule π that minimizes $E(L)$ is identical with one that minimizes Td , and thus we focus on $E(L)$ only. PI , henceforth called loss probability, may also be written

in terms of expectations of the various parameters previously determined in Section II. The number of line-switched calls turned away in a frame is $r_j + \nu_j - S_j$, the excess of calls over the slots made available. PI is the ratio of the average of this quantity to the average number of calls in a frame, $\lambda_1 b$. Using subscript π to denote policy π , PI is given by

$$PI_\pi = \frac{E_\pi(r_j + \nu_j - S_j)}{\lambda_1 b}. \quad (5)$$

The optimality criterion is thus formulated as

$$\min_\pi \{E_\pi(L_j)\},$$

given the transitions (4) with

$$\frac{E_\pi(r_j + \nu_j - S_j)}{\lambda_1 b} \leq \text{PLOSS}. \quad (6)$$

Here PLOSS is the maximum permitted fraction of lost calls. The transition probabilities (4) and the criterion (6) define a finite state decision process with minimization of the average expected cost $E_\pi(L_j)$ and a constraint on the average linear action-state combination $E_\pi(r_j + \nu_j - S_j)$. It turns out (see [24]) that if π is restricted to the irreducibility class $\{\pi\}_M$, as defined in the Appendix, then there exists a stationary policy, assigning actions to states independently of the past history, which satisfies the optimality criterion of (6). It is shown in [24] and [25] that the inequality constraint in (6) leads, in general, to a possibly probabilistic (i.e., nondeterministic) assignment rule for the optimum policy.

A considerable simplification of the problem arises if we assume the frame duration b is very short compared to the average message length $1/\mu$. (This corresponds to the case of a high-speed trunk.) This assumption is identical to the one used in the Erlang-B approximation for synchronous line-switching traffic, and it is shown to be valid for a wide variety of applications in [26]. Under this assumption, the number of line-switching messages in the system may not change by more than one from frame to frame. This corresponds to a birth-death process model for the line-switching traffic. There are thus only two possible actions for the optimal policy—to block a new call or to allocate a slot to it. The three-dimensional vector X_j is then reduced to a two-dimensional vector $X_j = (L_j, r_j)$, and the possible actions may be written as

$$A(L_j, r_j) = 0, \quad \text{if a new call is blocked,}$$

$$A(L_j, r_j) = 1, \quad \text{if a new call is given service.}$$

Obviously, a decision must be made only if a new arrival occurs (with probability $\lambda_1 b$). The boundary S may, therefore, either accommodate the r_j sessions already in the system or increase by one in order to allocate a newly arrived session

AD-A131 915

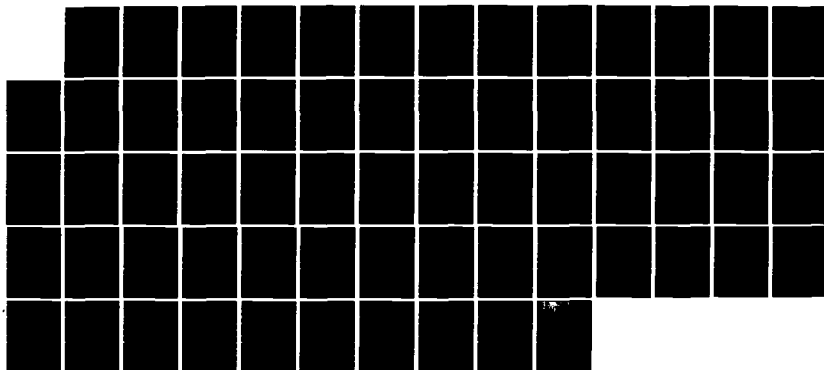
RESEARCH IN NETWORK MANAGEMENT TECHNIQUES FOR TACTICAL
DATA COMMUNICATION. (U) POLYTECHNIC INST OF NEW YORK
BROOKLYN R R BOORSTYN ET AL. 01 SEP 82 CECOM-80-0579-F
DAAK80-80-K-0579

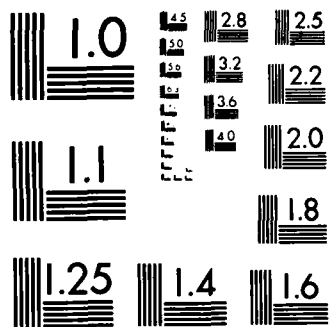
5/5

UNCLASSIFIED

F/G 17/2.1

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

Note that the probability of loss PI_* will in this case be

$$PI_* = \frac{E_s(r_i + r_i - S_i)}{\lambda_i h} = \frac{P_s(A(L_i, r_i) = 0)P(r_i = 1)}{\lambda_i h} \\ = P_s(A(L_i, r_i) = 0). \quad (5a)$$

Under the birth-death approximation, therefore, PI_* is the actual probability of loss (or probability of blocking) for the line-switched traffic. This justifies the use of the term probability of loss for the average fraction of (5).

The problem of minimizing the average packet queue length given the inequality constraint on the probability of blocking new setup calls can now be formulated [22, p. 152] as the following linear program.

For every class of policies $\{\pi\}_M$, $\lambda_1/\mu < M' \leq M$ (see the Appendix) determine the nonnegative variables $P(L, r, A)$ (the joint probabilities of state (L, r) and action $A \in \{0, 1\}$ defining the probabilistic state action assignment) for $0 \leq L \leq Q$, $0 \leq r \leq M'$, in order to minimize

$$E(L) = \sum_{i=0}^1 \sum_{r=0}^{M'} \sum_{L=0}^Q LP(L, r, i)$$

under the probability of blocking constraint

$$\sum_{L, r} P(L, r, A = 0) \leq \text{PLOSS}. \quad (7)$$

the linear transition constraints, and the probability measure constraint (all probabilities are nonnegative and sum to one).

The linear programming algorithm may be used to compute different time delay probability of blocking optimal pairs. Apart from the computational complexity, the main disadvantage of the method is the nondeterministic structure of the resulting policies due to the probability of blocking constraint. It is thus desirable to introduce another optimality criterion leading to deterministic policies. This is done in the next section. Computational results for both criteria are presented in Section V.

IV. SECOND OPTIMALITY CRITERION

The second optimality criterion, which leads to deterministic policies, simply minimizes the weighted sum of the average packet queue size and the probability of loss for line-switched calls. Specifically, we choose to minimize

$$g = E_s(L) + \alpha PI_*. \quad (8)$$

By letting the weight factor α vary ($0 \leq \alpha \leq \infty$), different optimal pairs $(E_s(L), PI_*)$ may be obtained that minimize g . It can be shown [27] that these pairs correspond to points on the optimal average queue length-loss probability curve (the first criterion) as well. This second criterion thus also serves as a way of computing some points on the curve for the first. By incorporating the previous constraint on loss probability in the average cost function, the problem turns out to be identical to a finite state Markov decision process with average cost minimization. It follows [22] that for every irreducible class of policies $\{\pi\}_M$ an

optimal policy exists, and it is deterministic, i.e., it has the form of a fixed assignment of an action to every state. We first consider the birth-death approximate model of Section III, then take the exact model with no such approximation made.

A. The Birth-Death Approximation

Let us define the one-period cost $C(X, S)$ as follows:

$$C(X, S) \doteq C(L, r, A) \doteq L + \alpha(1 - A),$$

$$A = \begin{cases} 0, & \text{if } (L, r) \text{ is a blocking state,} \\ 1, & \text{if } (L, r) \text{ is an allowing state.} \end{cases}$$

Then (see [22, p. 141]), the average cost g will be independent of the initial state $(L_0, r_0) = X_0$ and given by

$$g = \lim_{n \rightarrow \infty} E_s \sum_{i=0}^n \frac{C(X_i, S_i) | X_0}{n+1} = E_s(L) + \alpha PI_*. \quad (8a)$$

For every class of policies $\{\pi\}_M$, the transitions (4) and the average expected cost (8) define a finite state Markov decision process with average cost minimization and valid irreducibility condition. Therefore, from [22, theorem 6.17] we conclude that an optimal policy π exists within every class $\{\pi\}_M$, and it is *stationary deterministic*. Equivalently, there exists a set of bounded functions $h(X) = h(L, r)$, $0 \leq r \leq M'$, $0 \leq L \leq Q$ and a fixed number g such that

$$g = h(L, r) \\ = \min_{A \in \{0, 1\}} \left\{ C(L, r, A) \right. \\ \left. + \sum_{L', r'} P(L', r' | L, r, A) h(L', r') \right\}, \quad (9)$$

where

$$C(L, r, A) = L + \alpha(1 - A)$$

and

$$P(L', r' | L, r, A) = \begin{cases} P_s(L' | L)P_s(r'), & \text{if no arrivals occur,} \\ P_{s-1}(L' | L)P_{s-1}(r'), & \text{if one arrival occurs.} \end{cases}$$

Again, $P_s(L' | L)$ is given by (3), while under the birth-death assumption

$$\text{probability of no arrivals} = 1 - \lambda_1 h,$$

$$\text{probability of one arrival} = \lambda_1 h,$$

$$P_s(r') = \begin{cases} \mu h s, & \text{if } r' = s - 1 \text{ (one departure),} \\ 1 - h s, & \text{if } r' = s \text{ (no departure),} \\ 0, & \text{otherwise.} \end{cases}$$

The minimization of the right side of (9) is carried out under two distinct options: a) the state (L, r) blocks a new arrival, or b) it allows a new arrival in the system. Thus (9) defines the deterministic optimal policy within the class $\{\pi\}_M$. The global optimal policy must be selected among the optimal policies of all classes, with $\lambda_1/\mu < M' \leq M$.

Equation (9) can be used to carry out a search algorithm for the determination of the optimal policy. This algorithm, referred to as the policy improvement algorithm (see [22, p. 150]) starts with an arbitrary initial policy, solves the resulting optimality equation for $h(L, r)$ and g , and searches for state-action assignment alternatives leading to a smaller cost under the same $h(L, r)$. The improved policy is set for the new iteration until no improvement can be found. The advantage of this algorithm compared to the linear programming approach is that it searches only for a deterministic optimal policy. Equation (9) may be used to derive some properties of the optimal policy, in order to reduce the amount of searching and the computational complexity of the policy improvement algorithm.

Optimality and the Movable Boundary: The optimal policy discussed above can be represented as a function $l(r)$, where $l(r)$ is the minimum L for which (L, r) is a blocking state. In Fig. 2 the transition diagram for the birth-death line-switching traffic case is given. The dwell time Tr (the number of transitions the process stays in state r) will have a mean value

$$E(Tr) = \frac{1}{\mu br + \lambda_1 b P(L < l(r))} \geq \frac{1}{\mu M b + \lambda_1 b} \gg 1$$

(see [28]). It is, therefore, reasonable to assume that the packet queue size is a much faster changing process and hence reaches equilibrium within Tr . The factor by which equilibrium is reached independently of the initial state depends on the eigenvalues of the one step transition matrix for the queue state equation. It is shown in [22] that the process "shrinks" to its equilibrium by a factor equal to the product of the absolute values of these eigenvalues, the so-called shrinkage factor. Typical values of the shrinkage factor for this particular queue model range from 10^{-10} to 10^{-60} .

Under the above reasoning, the expected queue length will depend only on the stationary probabilities $P_\pi(r)$:

$$E_\pi(L) = \sum_{r=0}^{M'} E(L|r) P_\pi(r),$$

with $E(L|r)$ independent of π .

The Markov decision process that minimizes the average cost (8) under these conditions is thus defined by the state r_j , decisions $A = 0, 1$, and transitions as in Fig. 2. The optimal policy π will again be deterministic; i.e., the set of states is partitioned into two subsets through a boundary M' : if $r < M'$, a new setup call is served; if $r \geq M'$, a new call is blocked. This is identical to the movable boundary scheme described in the introduction. M' will depend on the value of α , the weight factor. Note that not all movable boundary schemes may correspond to an α -optimal scheme.

B. The Exact Model

We now proceed to the exact model analysis under the second optimality criterion. Since the number of arrivals or departures is not restricted to one, the results of Section IV-A under the birth-death assumption are not applicable.

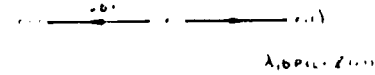


Fig. 2. Birth-death transitions.

Using the state notation of Section II-C and (5) for PL in (8), we formulate the finite state Markov decision process with transitions (1)-(4) and average expected cost minimization as

$$\min_{\pi} \left\{ E_\pi(L) + \alpha E_\pi \left(\frac{r_j + \nu_j - S_j}{\lambda_1 b} \right) \right\} = g. \quad (10)$$

From (2) (derived from the memoryless property of the exponential distribution), we may easily show that

$$E(r_{j+1}) = \sum_{S_j} P(r_{j+1}|S_j) P_\pi(S_j) = e^{-\alpha b} E_\pi(S_j).$$

Assuming irreducibility with π belonging to an irreducibility class $\{\pi\}_U$, the process achieves equilibrium and $E_\pi(r) = e^{-\alpha b} E_\pi(S)$ or

$$E_\pi(S) = e^{\alpha b} E_\pi(r). \quad (11)$$

Redefining the weight $\alpha = (\alpha/\lambda_1 b)(e^{\alpha b} - 1)$ and $g = g - \alpha$ we have, from (10) and (11), as the equivalent cost function to be minimized:

$$g = \min_{\pi \in \{\pi\}_U} \{E_\pi(L) - \alpha E_\pi(r)\}, \quad \alpha > 0. \quad (12)$$

Equations (1)-(4) and the cost function (12) guarantee the existence and deterministic nature of the optimal policies within the irreducibility classes $\{\pi\}_U$. These policies assign the number of line-switched slots in a frame S such that $r \leq S \leq \min(r + \nu, M')$, based on the state $X = (L, r, \nu)$: $X \xrightarrow{\pi} S$.

The average expected cost in (12) suggests that the one-period cost $C(X, S)$ is

$$C(X, S) = C(L, r) = L - \alpha r. \quad (13)$$

The optimality equations can be simplified by reducing the three-dimensional state space to a two-dimensional one. After some algebra, it is concluded that the optimality equations are defined over the set of bounded functions $h'(L, r)$ satisfying

$$g + h'(L, r) = L - \alpha r + \sum_{\nu=0}^{C_{MAX}} P(\nu) \left\{ \min_{r' \in \{r, r+1\}} \sum_{L'=0}^Q \sum_{S'=0}^S P_\pi(r') P_\pi(L'|L) h'(L', r') \right\} \quad (14)$$

with $0 \leq L \leq Q$, $0 \leq r \leq S$, and $0 \leq \nu \leq C_{MAX}$ (Recall that C_{MAX} is the maximum number of calls per frames in the truncated Poisson arrival process.) Probabilities $P(r)$, $P_\pi(r')$, $P_\pi(L'|L)$ are given by (1), (2), and (3), respectively. The policy improvement algorithm defined by (14) follows.

Let $H(S, L)$ denote the function to be minimized in (14):

$$H(S, L) = \sum_{r=0}^Q \sum_{L'=0}^S P_r(r') P_{L'}(L'|L) h'(L', r). \quad (15)$$

Within the irreducibility class $\{\pi\}_U$, we denote a deterministic policy by the state-action mapping $A_U(L, r, \nu) = S$.

The algorithm is then formulated as follows: for every class of policies $\{\pi\}_U$, $\lambda_1/\mu < M' \leq M$.

Step 1: Initialize to the movable boundary policy: $A_M(L, r, \nu) = \min(r + \nu, M')$.

Step 2: Solve the linear system for g and $h'(L, r)$:

$$g + h'(L, r) = L - \alpha r + \sum_{\nu=0}^{C_{MAX}} P(\nu) \sum_{L'=0}^Q \sum_{r'=0}^S P_r(r') P_{L'}(L'|L) h'(L', r'), \quad \forall (L, r),$$

with $S = A_M(L, r, \nu)$, $h'(0, 0) = 0$.

Step 3: Compute $H(S, L)$ of (15) for $0 \leq S \leq M'$, $0 \leq L \leq Q$.

Step 4: For every (L, r, ν) , $\nu > 1$, $r < M'$, find the integer S in the interval $r \leq S \leq \min\{r + \nu, M'\}$ which minimizes $H(S, L)$. Denote this S by $A'_U(L, r, \nu)$.

Step 5: If $A'_U(L, r, \nu) \neq A_U(L, r, \nu)$ for some states (L, r, ν) , set $A_U(L, r, \nu) = A'_U(L, r, \nu)$ and go to 2. If not, stop. Select the policy $A_U(L, r, \nu)$ resulting in the minimum g for all M' , $\lambda_1/\mu < M' \leq M$.

The optimal policy, in this general case, does not necessarily have a movable boundary format as it did for the birth-death approximation. It has been found in all the examples we have worked out that $H(S, L)$ defined by (15) is a convex function of S for all L , $0 \leq L \leq Q$, achieving a minimum $S_{min}(L)$ within $0 \leq S \leq M'$ (see Fig. 3). Then the optimal policy will assign to every state (L, r, ν) the action S with S the closest integer to $S_{min}(L)$ satisfying the condition $r \leq S \leq \min\{r + \nu, M'\}$, or

$$(L, r, \nu) \xrightarrow{\pi} S = \begin{cases} S_{min}(L), & \text{if } r \leq S_{min}(L) \leq r + \nu, \\ r, & \text{if } r > S_{min}(L), \\ r + \nu, & \text{if } r + \nu < S_{min}(L). \end{cases}$$

The seemingly complex policy thus reduces for convex $H(S, L)$ to a simple one that depends on the packet queue size L . The other components of the state vector $X = (L, r, \nu)$ are taken into account in simple fixed point comparisons and additions. The multiplexer needs to maintain in its local storage only the table $S_{min}(L)$ of size $Q + 1$.

We have not been able to prove formally the convexity of $H(S, L)$. However, the intuitive understanding of the optimal policy format and our computational experience suggest that, for all practical purposes, the statement above is valid.

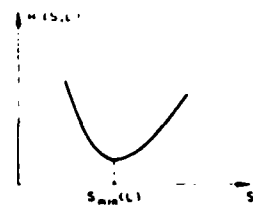


Fig. 3 Structure of $H(S, L)$.

V. NUMERICAL RESULTS

A. Birth-Death Approximation

Two numerical examples were studied under this approximation. The linear programming and policy improvement algorithms were implemented on a computer. The policy improvement algorithm led to movable boundary schemes as expected, within at most three iterations. Linear programming was applied only to the first example (consisting of 126 variables and 66 constraints) since the size of the second example (386 variables and 147 constraints) did not permit convergence of the modified simplex method used. The two examples follow.

Case 1: The following parameters were used in this case:

$$\begin{aligned} \lambda_1 &= 2.0 \text{ messages/s.} & 1/\mu &= 1 \text{ s.} & \rho_1 &= \lambda_1/\mu = 2, \\ \lambda_2 &= 2000 \text{ packets/s.} & & & \rho_2 &= \lambda_2 b = 2, \\ b &= 0.001 \text{ s.} & M &= 6 \text{ slots/frame.} & Q &= 8 \text{ packets.} \end{aligned}$$

In Fig. 4 we have plotted the resultant loss probability versus the average queue length. Points resulting from $\{E(L) + \alpha PI\}$ minimizations (small circles) belong to the optimal curve $\{\min E(L), PI \leq K\}$ extrapolated from the linear programming results (black circles). The former points are achieved through movable boundary policies ($M' = 3, 4, 6$), whereas the latter correspond to non-deterministic schemes. We include the movable boundary policy with $M' = 5$, which does not belong to the optimal curve. It is, however, very close to that curve, offering excellent (suboptimal) performance. In order to check the validity of the reasoning in Section IV-A concerning the fast evolution of the packet process compared to the line-switching slot number variations, the shrinkage factor for transition matrix $P_r(L|L')$ was evaluated. It was found to range from 0.26×10^{-45} for $S = 1$ up to 0.22×10^{-12} for $S = 6$. The values of this factor justify the fast evolution assumption. As a further check, the stationary conditional expectations $E(L|S)$ were used to compute $E(L)$ and PI for the movable boundary schemes with $M' = 3, 4, 5, 6$. The steady-state probabilities $P_r(S)$ were assumed to be given by the Erlang-B probabilities. A comparison with the policy improvement exact results showed that the two methods agreed within two decimal digits.

Case 2: We repeated the calculations above (except for the linear programming approach) for a larger state-space

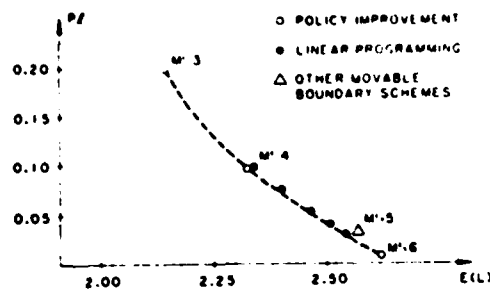


Fig. 4. Cost functions for case 1

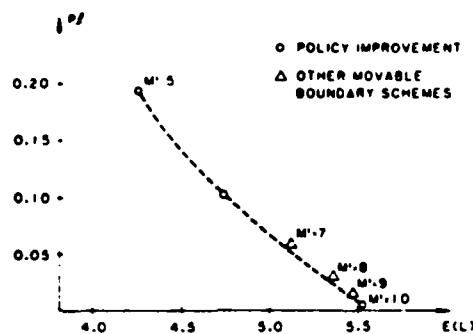


Fig. 5. Cost functions of case 2.

example:

$$\begin{aligned}\lambda_1 &= 4.0 \text{ messages/s.} & 1/\mu &= 1 \text{ s.} & \rho_1 &= \lambda_1/\mu = 4, \\ \lambda_2 &= 4000 \text{ packets/s.} & & & \rho_2 &= \lambda_2 b = 4, \\ b &= 0.001 \text{ s.} & M &= 10, & Q &= 12 \text{ packets.}\end{aligned}$$

Results are plotted in Fig. 5. Again, all movable boundary policies either belong to the optimal curve or are quite close to it. The validity of the fast evolution assumption was checked up to two decimal digits.

As a conclusion, under the birth-death assumption the movable boundary schemes, which turn out to be quite dense in the region of interest ($Pl \leq 0.05$), offer optimal or near-optimal performance.

B. The Exact Model

The policy improvement algorithm of Section IV-B was implemented for the general case (the number of arrivals and departures are permitted to have any value). The algorithm is used to determine the limitations of the birth-death assumption and to derive the properties of the optimal policy.

1) *The Validity of the Birth-Death Assumption:* We applied the algorithm to the example of case 1 in Section V-A with the parameter $\alpha = 3$, leading to an optimal boundary scheme with $M' = 4$. The exact optimization was carried out by varying the frame length b from 0.001 to 0.1. It turns out that the general policy improvement algorithm leads to the same optimal schemes (movable boundary with

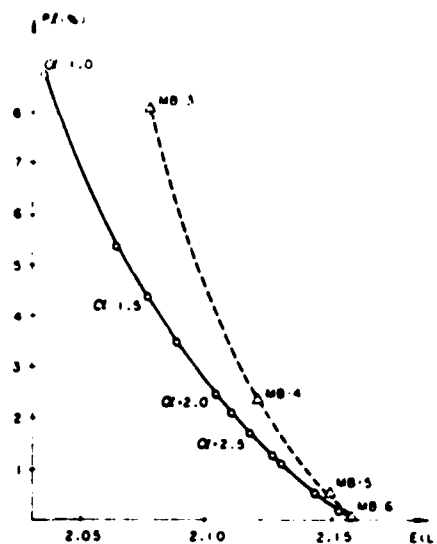


Fig. 6. Policy improvement, case 3

$M' = 4$) and performance measures as the simplified one under the birth-death assumption, with values of b less than 0.01. For larger frame durations (lower speed trunks) the optimal deterministic schemes are different from the movable boundary ones, having the table look-up form described in Section IV-B.

2) *Policy Improvement, Case 3.* The following parameters were selected for the application of the general policy improvement algorithm:

$$\begin{aligned}\lambda_1 &= 0.6 \text{ messages/s.} & 1/\mu &= 3.33 \text{ s.} & \rho_1 &= 2, \\ \lambda_2 &= 2.0 \text{ packets/s.} & & & \rho_2 &= 2, \\ b &= 1 \text{ s.} & M &= 6 \text{ slots/frame.} & Q &= 8 \text{ packets.}\end{aligned}$$

This case has the average message length three times the frame size, hence the birth-death assumption is not valid. Optimal deterministic policies were computed for different values of α . The corresponding average queue length $E(L)$ and probability of loss Pl have been plotted in Fig. 6. For comparison, movable boundary schemes were evaluated as well (small triangles in the figure). As expected, $E(L)$ is increasing, and Pl is decreasing with the weight α . Unlike the birth-death case, the optimal points found by varying α form a rather dense or closely spaced set. Hence there is no need for the complicated linear program to determine the optimal curve. For a given level of loss probability, a deterministic scheme or, equivalently, a weight α can be found minimizing the average queue length. This flexibility in adjusting the multiplexer performance to an optimal combination of line-switching probability of loss and packet-switching time delay is the main advantage of the method.

The optimal policies were found to have the form described in Section IV-B, i.e., for every L , the number of packets in queue, a number $S_{\min}(L)$ exists such that the optimal policy assigns actions $0 \leq S \leq M'$ to state (L, r, v)

TABLE I
 $S_{\min}(L)$

$\alpha \backslash L$	0	1	2	3	4	5	6	7	8
1.00	6	5	4	3	2	1	1	0	0
1.25	6	5	4	3	2	2	2	1	0
1.50	6	5	4	3	3	2	2	1	1
1.75	6	5	4	3	3	2	2	2	1
2.00	6	5	5	4	4	3	2	2	1
2.25	6	5	5	4	4	3	3	2	2
2.50	6	6	5	5	4	3	3	2	2
2.75	6	6	5	5	4	4	3	3	2
3.00	6	6	5	5	4	4	4	3	3
3.25	6	6	6	5	5	4	4	4	3
3.50	6	6	6	6	5	5	6	6	6
3.75	6	6	6	6	6	6	6	6	6

as follows:

$$S = \begin{cases} S_{\min}(L), & \text{if } r \leq S_{\min}(L) \leq r + v, \\ r, & \text{if } S_{\min}(L) < r, \\ r + v, & \text{if } r + v < S_{\min}(L). \end{cases}$$

As previously, r is the number of line-switched sessions requiring continuation of service, v represents the new setup line-switched calls per frame, and S is the number of slots allocated to the line-switched traffic. In Table I, we show $S_{\min}(L)$ as found for the values of α used above. As expected, $S_{\min}(L)$ is, in general, increasing with α (i.e., more emphasis is given to the probability of loss) and decreasing with L . Note that in the case of $\alpha \geq 3.25$ and $L > 5$, this pattern is violated (see lower right part of Table I). This is mainly due to the finite queue length assumption, which imposes boundary conditions on the optimization problem. Note that the packet queue overflow probabilities did not exceed 5 percent for the selected queue size $Q = 8$.

Comparing with the movable boundary policy, we note that the improvement in the expected queue length variance provided by the optimal policy is not significant over the movable boundary scheme for comparable values of loss probability. (Note that the expected queue length coordinates in Fig. 6 are greatly expanded.)

In order to further illustrate the comparative performance of the optimal and the movable boundary schemes, we evaluated their performance under variations of the input line-switched traffic rate λ_1 . We constrained the line-switching loss probability P_l to a maximum acceptable level of 2.8 percent and computed the corresponding performance of the optimal and movable boundary schemes. Every optimal policy search involved several runs for the determination of the weight providing a loss probability closest to the 2.8-percent figure. For comparison, the performance of the fixed boundary scheme (see [20]) was evaluated as well. This scheme allocates two predetermined portions of the frame to the two traffic categories without permitting the packet traffic to occupy any idle slots in the line-switched portion.

We further assumed that signaling packets, necessary for the control of the line-switched sessions (path setup, mapping of line-switched slots, termination of calls, etc.) are

transmitted as ordinary packets; thus they introduce overhead in the packet traffic. We approximated this overhead using the method described in [1]. Specifically, we assumed that every line-switched session introduces a packet overhead equal to $K(\lambda_1/\mu)$ with K a parameter depending on the signaling protocol. The packet input rate is then given by $\lambda_2 = \lambda_1' + K(\lambda_1/\mu)$, with λ_1' the input rate of the data packets. We did not explicitly consider this overhead in the example of case 3, since the values of λ_1 and λ_2 were assumed fixed. Here, the dependence of λ_2 on variations of λ_1 must be measured since we let λ_1 vary. We chose $K = 0.3$ for the particular calculations, a value used in [1].

In Fig. 7 we demonstrate the performance of these schemes (optimal, movable, and fixed boundary) with the line-switched traffic rate λ_1 varying. It can be seen that the movable boundary schemes result in almost optimal packet queue length Fig. 7(a) with probability of loss adjusted below 2.8 percent. In Fig. 7(b) the flexibility of the optimal scheme in reaching a given level of loss probability is demonstrated. The packet input rate λ_2 variation with λ_1 due to the signaling overhead is shown in Fig. 7(c). This variation introduces the early queue overflow of the fixed boundary scheme in Fig. 7(a).

Similar results are found with the variation of the packet input rate λ_2 , while keeping the line-switched traffic rate λ_1 fixed. $\lambda_1 = 0.6$ messages/s was chosen as the fixed parameter in the calculations here. These results appear in Fig. 8.

Finally, a sensitivity analysis for small variations of λ_1 around the reference point $\lambda_1 = 0.6$ messages/s and $\lambda_2 = 2$ packets/s of case 3 showed no significant difference between the optimal policy with $\alpha = 2$ (see Fig. 6) and the movable boundary scheme with $M' = 4$. These results are plotted in Fig. 9. Note that the two sets of curves are both approximately linear about the initial operating point and track one another rather closely over the range of variation of λ_1 shown.

VI. CONCLUSION

The problem of dynamically allocating the bandwidth of a trunk to both line-switched and packet-switched digital traffic has been formulated in this paper as a Markov decision process. Line switching was modeled as a time division multiplexing loss scheme using a varying portion of a fixed time frame. Packet-switching traffic is served by the remaining portion of the frame and requires queueing at the multiplexer-concentrator.

Two different cost criteria were examined involving probability of loss for line-switching and average queueing delay for packets. The corresponding optimization problems were presented under reasonable simplifying assumptions.

For the birth-death approximate model, valid for high-speed trunks serving sufficiently lengthy line-switched sessions (thus requiring many frames for transmission), the movable boundary scheme was found to provide optimal

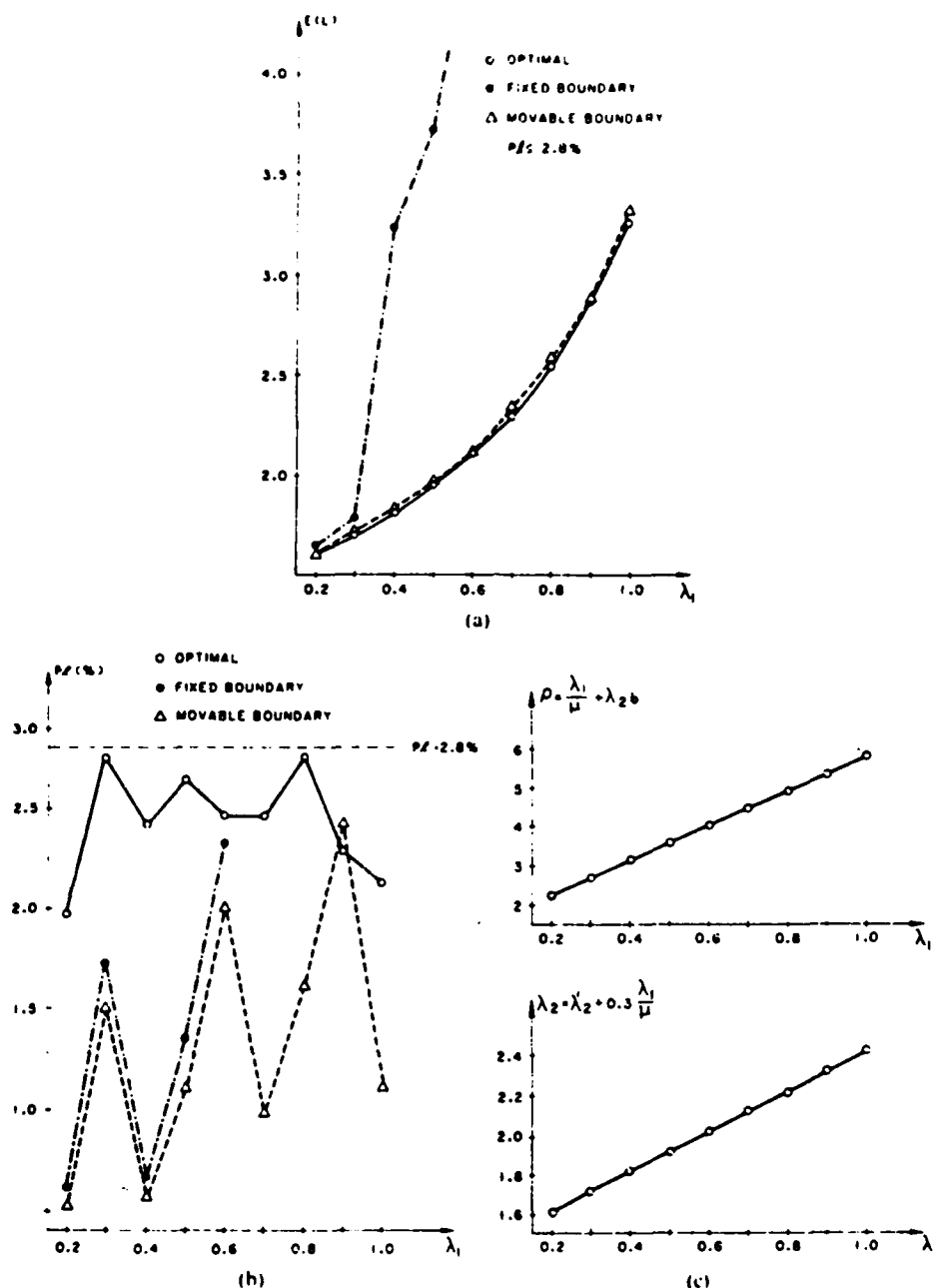


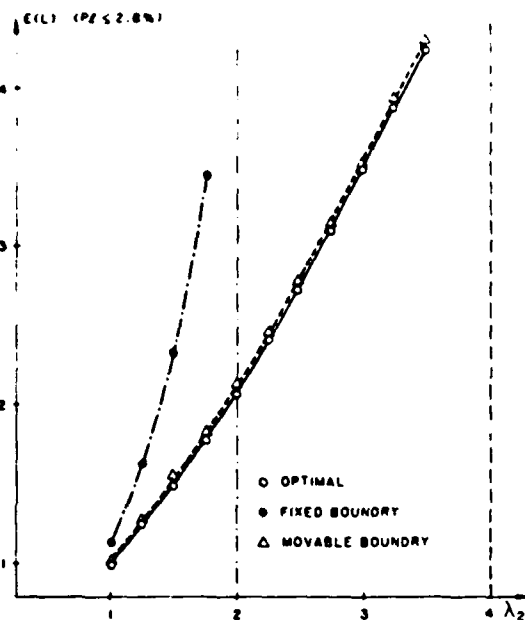
Fig. 7 (a) Average queue length with λ_1 varying (b) Probability of loss with λ_1 varying (c) Signaling overhead

points in the loss probability-average packet delay plane. The optimal points, falling on the curve connecting the movable boundary points, may be achieved by more complicated and even nondeterministic schemes. Those are, however, difficult to define and to implement.

For larger values of the frame duration (or shorter time-switched sessions), the birth-death assumption is not valid and optimal deterministic schemes found are different from the movable boundary ones. An optimal *deterministic* policy can be defined in order to minimize the packet delay with the loss probability close enough to a given value. The structure of such policies is simple and may be easily implemented for an intelligent multiplexer with the

capability of table look-up. The length of these tables equals the packet queue length.

It is, however, possible to achieve acceptable levels of loss probability and near-optimal packet delay with movable boundary schemes. We therefore suggest the use of a movable boundary technique for those integrated systems requiring a fixed frame. It offers excellent performance and simplicity of implementation. We have thus shown that the movable boundary technique, introduced some years ago in an *ad hoc* manner as a simple scheme for improving the capacity utilization of the fixed boundary procedure, does in fact display near-optimum properties in the sense discussed in this paper.

Fig. 8. Average queue length with λ_2 varying

APPENDIX IRREDUCIBILITY CONDITION

In order to proceed in determining the optimal policy, we need to specify whether the process exhibits the *irreducibility condition*. This condition [22] states that under any *deterministic* policy (assigning an action to a state with probability one regardless of the past history of the process), the resulting discrete-time Markov process consists of one class of communicating states. In our case, there are policies which do not allow the number of line-switched (LS) sessions per frame to exceed a given number $M' < M$. (For example, the simple movable boundary policy described in Section I does not allow the LS traffic to occupy the whole frame.) Under these policies, states X_{i-1} with $M' < r_{i-1} < M$ may not be reached from state X_i with $0 \leq r_i \leq M$. In order to preserve the irreducibility condition, we therefore need to define the following classes of policies (referred to hereafter as "irreducibility classes") based on the different barriers imposed on the line-switched slot numbers per frame: the set $\{\pi\}$ of all policies is divided into the mutually exclusive subsets or irreducibility classes $\{\pi\}_M$, $0 < M' < M$, such that any policy belonging to the class $\{\pi\}_M$ may assign actions achieving but never exceeding M' ($S_i \leq M'$).

If we limit the search for an optimal policy within a particular class $\{\pi\}_M$, the state action space is reduced to $0 \leq r_i \leq M'$ and $0 \leq S_i \leq M'$. Obviously, any deterministic policy belonging to $\{\pi\}_M$ results in an irreducible Markov chain: (1) permits r_i to take any value up to CMAX; (2) lets r_i take any value up to S_i with $S_i \leq M'$; and (3) from $M/G/N$ queue analysis defines an irreducible process with $0 < L_i \leq Q$ for any S_i pattern as long as the frame format can accommodate the average traffic requirement. The latter condition leads to lower bounds for M' and M . M' must be greater than the LS traffic intensity and M greater than the total traffic intensity (line switching plus packet switching). Thus $\lambda_1/\mu < M' < M$ and $\lambda_1/\mu + \lambda_2/b < M$. This reasoning suggests that the irreducibility condition holds within an irreducibility class of policies and for the accordingly reduced state space

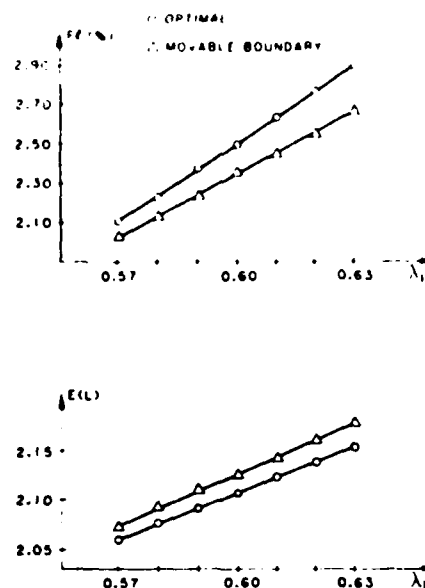


Fig. 9. Sensitivity analysis

REFERENCES

- [1] F. Closs, "Message delays, trunk utilization in line switching and message switching networks," in *AFIPS Conf. Proc.*, 1972, pp. 524-529.
- [2] K. Kummerle and H. Rudin, "Packet and circuit switching: Cost performance boundaries," *Comput. Networks*, vol. 2, pp. 3-17, Feb. 1978.
- [3] H. Miyahara *et al.*, "A comparative evaluation of switching methods in computer communication networks," in *ICC-75 Proc.*, pp. 6-6-6-10.
- [4] D. Maiwald *et al.*, "Integrated communication system performance," in *ICC-73 Proc.*, pp. 24-13-24-21.
- [5] K. Kummerle, "Multiplexer performance for integrated line- and packet-switched traffic," in *ICCC Proc.*, 1974, pp. 508-515.
- [6] P. Zafiropoulos, "Flexible multiplexing for networks supporting line-switched and packet-switched data traffic," in *ICCC Proc.*, 1974, pp. 517-523.
- [7] E. Port *et al.*, "A network architecture for the integration of circuit and packet switching," in *ICCC Proc.*, 1976, pp. 505-514.
- [8] P. Zafiropoulos *et al.*, "Extension of a circuit-switched user-network interface to packet switching," in *ICCC Proc.*, 1976, pp. 515-522.
- [9] H. Rudin, "Studies of the integration of circuit and packet switching," in *ICC-78 Proc.*, pp. 20-2-1-20-2-7.
- [10] G. Coviello and P. Vena, "Integration of circuit-packet switching by a SENET (slotted envelope network) concept," in *ICC-75 Proc.*, pp. 42-12-42-17.
- [11] "SENET-DAX Study," GTE Final Rep., Contract DCA-100-75-C-10071, 1976.
- [12] M. Ross *et al.*, "Design approaches and performance criteria for integrated voice, data switching," *Proc. IEEE*, vol. 65, Sept. 1977.
- [13] G. Coviello *et al.*, "System design implications of packetized voice," in *ICC-77 Proc.*, pp. 38-3-49-38-3-53.
- [14] "Economic analysis of integrated DOD voice and data networks," Network Analysis Corp., Great Neck, NY, Final Rep. Contract DAHC-15-73-C-0135, Sept. 1976.
- [15] I. Gitman and H. Frank, "Economic analysis of integrated voice and data networks: A case study," *Proc. IEEE*, vol. 66, Nov. 1978.
- [16] *M200 Network Switching and Management System*, System Description Manual, Tran Corporation, El Segundo, CA, May 1976.
- [17] J. Vander Mey, "The architecture of a transparent intelligent network," in *ICC-76 Proc.*, pp. 7-2-1-7-2-5.
- [18] B. Maglans and M. Schwartz, "Performance evaluation of a variable frame multiplexer for integrated switched networks," to be published in *IEEE Trans. Commun. Technol.*
- [19] H. Miyahara and T. Hasegawa, "Integrated switching with variable

- frame and packet," in *ICC-78 Proc.*, pp. 2031-2035.
- [20] M. Fisher and T. Harris, "A model for evaluation of the performance of integrated circuit- and packet-switched multiplex structure," *IEEE Trans. Commun. Technol.*, vol. COM-24, pp. 197-202, Feb. 1976.
- [21] C. J. Weinstein *et al.*, "Data traffic performance analysis of an integrated circuit- and packet-switched multiplex structure," *IEEE Trans. Commun. Technol.*, vol. COM-28, pp. 873-878, June 1980.
- [22] S. Ross, *Applied Probability Models with Optimization Applications*. San Francisco, CA: Holden-Day, 1970.
- [23] L. Kleinrock, *Queueing Systems*, vol. 1. New York: Wiley, 1975.
- [24] C. Derman, *Finite State Markov Decision Processes*. New York: Academic, 1970.
- [25] Bertsekis, D., *Dynamic Programming and Stochastic Control*. New York: Academic, 1976.
- [26] E. Goltman *et al.*, "Issues in integrated network design," in *ICC-77 Proc.*, pp. 381-36-381-43.
- [27] I. Rubin, "Group random-access disciplines for multi access broadcast channels," *IEEE Trans. Inform. Theory*, vol. IT-24, Sept. 1978, pp. 578-592.
- [28] J. Keilson, "Markov chain models, rarity and exponentiality," Univ. of California, Berkeley, Tech. Rep., 1974.

E.5 Satellite Access Methods for a General Purpose Packet

Network with a Time Critical Traffic Component

Maglaris and Lissack

To be published in Journal of Telecommunication Networks

**Satellite Access Methods for a General Purpose Packet
Network with a Time Critical Traffic Component**

Basil S. Maglaris and Tsvi Lissack
Network Analysis Corporation
130 Steamboat Road
Great Neck, New York 11024

ABSTRACT

Various satellite access alternatives are studied in terms of accommodating packetized traffic mix including a tight delay constrained component. The time critical traffic delay due to contention and frame latency is assumed to be, at most, 50 msec with 95% confidence. Two major access categories are studied. Under the first, the time critical traffic is served in a dedicated subchannel, either via slotted ALOHA random access, or via TDM slots. As a result, the non-time critical bulk traffic is decoupled from the stringent delay constraint and may use more efficient demand assignment technique. Under the second, a fixed TDMA frame is partitioned into fixed portions. Each transmitting earth station fills its dedicated portion, giving priority to the time critical traffic. The partition should be able to accommodate the anticipated time critical traffic within the given specifications.

Analysis of the above schemes is being carried using slotted ALOHA, M/D/1, and M/D/N queueing models. It is shown that the slotted ALOHA alternative provides a very poor candidate. Among dedicated TDM subchannels and priority TDMA alternatives, the latter is shown to provide a simple and robust scheme, especially suitable in case of relatively fixed traffic proportionality among the accessing earth stations.

I. INTRODUCTION

Satellite channels have been widely recognized as an attractive alternative to long-haul voice and data communications. Specifically, the development of high speed digital transmission hardware and high speed burst modems permits the transmission of digital data at rates above 50 Mbps. Broadcast geostationary satellites have been used for the transfer of packetized data among a population of users with appropriate earth station (ES) equipment. The corresponding protocols for satellite packet network, e.g., satellite access, transponder capacity sharing and error control have been developed, based on the satellite channel salient characteristics: Broadcast nature, 250-270 msec one-hop propagation delay, multiple access (shared use of the satellite transponder capacity), and Gaussian noise statistics. Frequency-division multiple access (FDMA) techniques have been traditionally associated with voice satellite communications. Recently, however, time-division multiple access (TDMA) has been suggested as a more flexible and efficient technique for both digitized voice and packet data. In TDMA, the data from each earth station are segmented into bursts of transmissions which are timed and interleaved within a time frame. Initial acquisition and synchronization are achieved via frame and burst overhead symbols.

Depending on the flexibility of a TDMA scheme to adapt itself to traffic variations, the following three major classes are defined in [1]: Dedicated uplink/dedicated downlink (Fixed) TDMA with fixed data-rate point-to-point links connecting every possible source destination pair, dedicated uplink/shared downlink having a fixed length burst assignment per source, dynamically filled with various destination data, and shared uplink/shared downlink with the source burst assignment dynamically varying according to the accessing earth station traffic requirements. From the above approaches, the first is unnecessarily rigid and wasteful for a packet data network. The second and third provide viable

alternatives depending on the traffic burstiness and delay/throughput requirements. For a bursty interactive packet environment, demand assignment techniques have been developed from the early satellite communication phases. These techniques, which are associated with the shared uplink/shared downlink class, range from random access pure and slotted ALOHA [2], [3] to implicit [4], and explicit reservation [5], [6] techniques. The latter schemes introduce centralized or distributed control mechanisms in order to resolve scheduling conflicts among the accessing units, thus increasing the system throughput at the cost of additional delay and implementation complexity.

A natural extension to the application of packet satellite network for interactive traffic has been the use of the satellite large bandwidth resources and broadcast nature to support multiple purpose integrated digital communications. These networks should be able to provide timely and reliable data communications to a variety of users and applications. It is, therefore, necessary to devise priority access schemes in order to satisfy performance requirements for traffic categories ranging from interactive packet delivery, requiring less than half a second end-to-end delay, to message-switching electronic document distribution, overnight large file transfer, and voice/video communications. The PODA protocol described in [1], is a typical scheme aiming at dynamically accommodating such a diversity of applications.

The main issue associated with a general purpose integrated satellite packet network is the handling of the time critical applications. In order to provide comparable service to terrestrial value-adding networks, the end-to-end delay should be of the order of 300-350 msec including access queueing delay and latency and the 250-270 msec propagation delay. Obviously, no explicit reservation scheme can fulfill the above requirement since it introduces an additional one-hop reservation delay. A direct access scheme should, therefore, be evaluated, such that frame latency, queueing and/or retransmission statistics

do not violate the given specifications. The stringent time critical delay requirement is expected to result in some inefficiency in both the time critical and the non-time critical (bulk) traffic capacity utilization. It is the purpose of this paper to comparatively evaluate various feasible access methods and determine their design parameters as a function of the packet rate of the time critical traffic. Two general access categories are studied, as shown in Figure 1. Under the first, the time critical traffic access a dedicated subchannel either via slotted ALOHA or via time-division multiplexed (TDMA) slots dedicated to every earth station. As a result, the bulk traffic is decoupled from the very stringent delay requirement and may use an appropriate dynamic assignment technique. Under the second category, a TDMA frame is partitioned into segments or slots, preassigned to the accessing earth stations. The slot sizes depend on the relative traffic intensities of the earth stations and are updated periodically with the period spanning several frames. Each transmitting earth station fills its dedicated slot, giving priority to the time critical traffic. An M/D/N queueing model is used to define the minimum slot size such that to guarantee proper delivery of the time critical packets. For a description of this priority TDMA protocol see [7].

The models used for the performance evaluation of the above strategies assume a finite population of earth stations generating fixed-size time critical packets with Poisson statistics. The packet arrival rate λ is identical for all earth stations. Note, that the Poisson assumption is justified by the realistic model of a large terminal population clustered around a particular earth station. The tolerable time critical access delay is assumed throughout this paper to be 50 msec within a 95% confidence interval. This criterion provides a more meaningful performance measure than the average delay, despite the analytical complexity it involves. Whenever analytically inaccessible, the 95th percentile is estimated from first and second moments. A discrete event simulation is used

to check the validity of the above estimation. The 50 msec access delay specification is a reasonable choice resulting approximately in 300-350 msec end-to-end delay including processing and propagation.

II. DEDICATED TIME CRITICAL SUBCHANNEL STRATEGIES

In this section, we evaluate the two methods for handling the time critical traffic in a separate dedicated subchannel. Since these methods decouple the time critical and non-time critical components of the traffic, their adequacy and ability to meet the delay constraints are prerequisites for the use of demand assignment reservation schemes to handle the bulk components.

A. Slotted ALOHA Access Analysis

A slotted ALOHA subchannel can be used to provide direct random access to time critical interactive (IA) packets. The 250-270 msec one-hop propagation time results in a 500-550 msec delay to receive an ACK or detect a collision. In case of collisions and retransmissions, the time critical end-to-end delay is violated by far. Hence, the only viable solution is to keep the collision probability below 5%, allowing the 95% of the IA packets to experience only the frame latency assumed less than 50 msec. (By frame latency, we mean the delay experienced by a packet from its arrival to the time it may be considered for service.)

The slotted ALOHA subchannel consists of slots of length L (in Kbits), T sec apart as shown in Figure 2. The time critical packets of fixed length L , are generated in the N transmitting ES's with Poisson rate λ packets/sec and contend for the allocated slots. Collisions detected after 500 msec are handled by retransmitting the collided packets in future slots selected at random over an infinite interval. Following the analysis of the finite population model in [2], we have that the average number of retransmissions per packet R is given by,

$$E = \frac{G}{S} - 1 \quad (1)$$

where, $S = M \cdot \lambda \cdot T$ and G is given by the nonlinear equation,

$$S = G(1 - \frac{G}{M})^{M-1} \quad (2)$$

The probability of collision for small values of E will be approximately equal to E ,

$$E = 0 \cdot P(0) + 1 \cdot P(1) + \dots \approx P(1)$$

To evaluate the performance of the system, we must compute the maximum repetition interval T for which $E \leq 5\%$ under various arrival rates λ . Obviously, more frequent repetition of the dedicated slots (small T) results in putting aside a greater portion of the channel bandwidth for the time critical traffic, more fragmentation of the frame format and, hence, deterioration of the system performance.

B. TDM Subchannel Analysis

The model assumes that slots of length L , dedicated to every accessing ES, are separated by T sec as in Figure 3. The repetition period should again be less or equal to 50 msec to account for the frame latency. The time critical packets of length L arrive with Poisson rate λ and wait on a FIFO (First-In, First-Out) single server queue at the source ES. Service of outstanding packets occurs every T sec via the dedicated slot. An M/D/1 analysis is performed to compute the first and second moments of the access delay. (For a similar treatment, see [8].) We assume that the total delay D , consists of the sum of two

independent random variables: The uniformly distributed latency ℓ , $0 \leq \ell \leq T$ and the M/D/1 queueing W . The mean $\bar{D}$ and variance σ_D^2 are then given by,

$$\bar{D} = \frac{\lambda \cdot T^2}{2(1 - \lambda T)} + \frac{T}{2} \quad (3)$$

$$\sigma_D^2 = \frac{T^2}{12} + 2 \left[\frac{\lambda \cdot T^2}{2(1 - \lambda T)} \right]^2 + \frac{\lambda \cdot T^3}{3(1 - \lambda T)} \quad (4)$$

Let $D_{.95}$ denote the 95 percentile of the total access delay. We define the spread factor SF as the factor which, when multiplied by the standard deviation, determines the 95% confidence interval; namely,

$$D_{.95} = \bar{D} + SF \cdot \sigma_D \quad (5)$$

Various estimates can be considered for the value of the SF in (5). Using Chebyshev's inequality, it turns out that,

$$\Pr(D > \bar{D} + SF \cdot \sigma_D) = 5\% \leq \frac{1}{(SF)^2} \implies SF \leq 4.5$$

This is a loose upper bound on SF. Other relaxed ad-hoc choices may be $SF = 3$, and $SF = 1.96$ corresponding to the normal distribution. A more reasonable choice can be found using the following argument: If $T = 50$ msec, the probability of meeting the $\{D_{.95} \leq 50 \text{ msec}\}$ constraint is almost equal to the probability of an arrival in an empty queue (the access delay consists of latency only). This probability equals to λT . In order to make this probability equal to 5%, we select $\lambda = 1$ packet/sec with $T = 50$ msec.

$$D_{.95} \left| \begin{array}{l} \lambda = 1 \text{ p/sec} \\ T = 50 \text{ msec} \end{array} \right. = 50 \text{ msec}$$

From (3), (4), (5) with $\lambda = 1$ p/sec and $T = 50$ msec, we conclude that $SF = 1.5$. In Table 1, we present the 95% delay under $\lambda = 1$ packet/sec and $T = 50$ msec for different SF choices. Our choice $SF = 1.5$ leads to the most realistic value.

The M/D/1 95% analysis presented in this section is used to evaluate the maximum IA rate which does not result in more than 50 msec delay within 95% confidence for different values of T . Again, small values of T correspond to larger percentages of the total channel capacity dedicated to the time critical subchannel.

C. Evaluation

The analyses presented in the previous sections have been used to evaluate the required dedicated capacity (in percents of total available capacity) as a function of the packet arrival rate of the time critical traffic. Two wideband satellite channel capacities have been chosen as reference examples: $C = 14$ Mbps and $C = 20$ Mbps. The number of accessing ES's, M , assumes two extreme values 5 and 20. The capacity dedicated to the time critical subchannel C_{TC} , follows directly from the corresponding repetition time T of the dedicated slots of length $L = 1$ Kbit. Thus, for the slotted ALOHA case,

$$\frac{C_{TC}}{C} = \frac{L}{T \cdot C} \quad (6)$$

For the dedicated TDM case,

$$\frac{C_{TC}}{C} = \frac{L \cdot M}{T \cdot C} \quad (7)$$

The numerical results are plotted in Figures 4 and 5. As it can be seen, slotted ALOHA saturates much faster than the dedicated TDM. This is due to the fact that the slotted ALOHA scheme should be designed such that the probability of contention is less than 5% to account for the 500 msec delay induced with any collision. The dedicated TDM can allow contention of arriving packets at the transmitting queue. If, for example, the dedicated slot is repeated every 10 msec, a packet may remain on queue for five periods without violating the 50 msec delay constraint. Therefore, the increase of the dedicated capacity (equivalently the decrease of the period T) has a greater impact on the dedicated TDM than on the slotted ALOHA scheme. In addition, S-ALOHA schemes involve more implementation complexity and require flow control for stability. We conclude, therefore, that dedicated TDM is preferred to S-ALOHA for handling the time critical traffic on a dedicated subchannel.

Another important advantage of using a dedicated subchannel approach is that these schemes decouple the time critical from the bulk traffic, thus permitting a dynamic assignment scheme for the latter. Their main disadvantages are due to the protocol complexity of handling separate logical links and the overhead associated with the frame fragmentation.

III. PRIORITY TDMA

A. Motivation and Description

This section deals with the implications of incorporating the time critical IA traffic within the bulk traffic access. We concluded, in the previous section, that slots of length L , dedicated to every transmitting source and dispersed in fixed time intervals, provide an acceptable mechanism for handling the time critical packets. This motivated the design of the simple and robust priority TDMA scheme presented in this section. Instead of the single server model of Section II, B, we simply collocate all slots dedicated to one source for a frame period less than or equal to 50 msec. By doing so, we reduce the frame fragmentation overhead and allow the non-time critical traffic to occupy any idle capacity not used by the time critical packets (recall that the 95% confidence constraint results in large percentages of unused capacity). We briefly describe its main features (see Figure 6):

- All traffic accesses the channel via a T sec frame, $T \leq 50$ msec. The frame length should be long enough to result in acceptable fragmentation without violating the 50 msec latency constraint. Note, that the high speed satellite channel is the main reason for making this scheme attractive. As an example, a 50 msec frame carries 1 Mbits when using a 20 Mbps channel.
- A variable size slot is allocated to every ES. The minimum allocation per ES should be such that 95% of the arriving time critical packets can be accommodated within 50 msec. The ES's serve the time critical packets with non-preemptive priority over the other traffic classes.

- The slot allocation remains fixed for a period of time. Every ES has full responsibility to allocate its data according to given priority tables and using some scheduling algorithm. Thus, the scheme can be classified as fixed uplink/shared downlink multiple access. It is, however, possible to adapt the capacity assignment to the traffic variations using a centralized control mechanism as in [7]. The central controller ES can update the allocations either by monitoring the ES's traffic statistics or after considering periodic traffic request reports. Note, that the synchronization and allocation update processes may span many frames. In addition, the complexity involved in updating the slot allocation suggests that this should not be performed very frequently (typically every 1,000 or 2,000 frames). The scheme can, therefore, adapt to relatively slow variations of the traffic mix: this is the case of a large population of terminals connected to an ES, resulting in quite smooth traffic with no rapid peaks due to averaging effects.

In the following section, we present the analytic tools needed to evaluate the minimum allocation per ES in order to guarantee proper time critical service.

B. Delay Analysis

The 95% confidence interval delay analysis is based on the M/D/N queue model of the TDMA scheme. Similar multiserver queueing models have been reported in [9], [10], and [11]. The M/D/N₁ queue is an accurate model for the time critical packet-service mechanism of Figure 3. Specifically, the service time of a packet is defined as the frame duration T, since all packets are considered for service at the beginning of a new frame.

The number of servers N_i corresponds to the number of time critical packets fitting in slot i . Whenever the allocation Q_i is greater than the minimum N_i , the delay constraint is further relaxed. We focus, therefore, on the computation of the minimum allocation N_i . For simplicity, we will drop all subscripts i , denoting ES_i . In order to simplify our analysis, we assume that the frame length T is depicted such that the 95% delay $D_{.95} = 50$ msec corresponds to an integer number of frames:

$$T \stackrel{\Delta}{=} T_K = \frac{D_{.95}}{K}, \quad K = 1, 2, \dots \quad (8)$$

The delay D encountered by a packet will depend on the number of packets q found in queue: specifically, we have,

$$\begin{aligned} D &= \text{Latency if } q < N \\ D &= \text{Latency} + T_K \text{ if } N \leq q < 2N \\ D &= \text{Latency} + 2T_K \text{ if } 2N \leq q < 3N \\ &\vdots \\ D &= \text{Latency} + iT_K \text{ if } N \leq q < (i+1)N \end{aligned}$$

Furthermore, since the frame latency equals to one frame with 100% confidence, we have,

$$q < iN \iff D \leq iT_K$$

It immediately follows that with $D_{.95} = K \cdot T_K$,

$$\Pr \{ D \leq D_{.95} \} = \Pr \{ q < KN \} \quad (9)$$

We wish to evaluate λ_K , the maximum rate of the time critical traffic for which the r.h.s of (9) is kept above 95%:

$$\text{With } T_K = \frac{D_{.95}}{K}, \quad K = 1, 2, \dots$$

$$\text{Find, } \lambda_K = \max \{ \lambda \} \quad (10)$$

$$\text{Such that, } \Pr \{ D \leq D_{.95} \} = \Pr \{ q < KN \} \geq 95\%$$

The queue state statistics, q , needed to evaluate (10) are summarized in the Appendix. For $K = 1$, $\Pr \{ q < N \}$ is given explicitly by (A-3) in the Appendix. Thus, the maximum rate λ_1 can be easily computed for various values of N using a simple search algorithm. For $K > 1$, no exact formula for the probability $\Pr \{ q < KN \}$ is available. Instead, we use an approximate method based on the first and second moments of the queue state as evaluated in the Appendix. Our approximation consists of the assumption that the M/D/N queue size distribution with arrival rate λ_K and frame length T_K , as defined in (10), will have for a given N , the same shape if normalized with respect to K . Under the above assumption, for every N we define the spread factor $SF(N)$, such that the 95% queue size KN of (10) is,

$$KN = E_K(q) + SF(N) \sigma_{q,K} \quad (11)$$

Here, $E_K(q)$ and $\sigma_{q,K}$ are the mean and standard deviation of the queue size under T_K and λ_K , given by (A-4) and (A-5) in the Appendix. The assumption that $SF(N)$ is independent of K permits its analytical evaluation from the known case $K = 1$. The $SF(N)$ estimate can then provide confidence intervals for larger values of K . Other choices for the spread factor,

such as $SF = 4.5$ (Chebyshev's inequality), $SF = 3$, $SF = 1.96$ (normal approximations), may prove valid for specific parameter sets, but are inflexible in adjusting to every particular queue distribution. Our method, which provides flexible and tight estimates has been checked via discrete simulation (see the next section).

The corresponding algorithm is summarized below:

Step 1: For a given $D_{.95}$ and N , set $K = 1$, $T_1 = D_{.95}$.

Find $\lambda_1 = \max \{ \lambda \}$, such that $\Pr \{ q < N \} \geq 95\%$ (Eq. A-3)

Step 2: With $\lambda = \lambda_1$ and $T = T_1$ compute $E_1(q)$ (Eq. A-4) and $\sigma_{q,1}$ (Eq. A-5).

$$\text{Compute } SF(N) = \frac{N - E_1(q)}{\sigma_{q,1}}$$

Step 3: For $K = 2, 3, \dots$, and $T_K = \frac{D_{.95}}{K}$

Find $\lambda_K = \max \{ \lambda \}$, such that $KN \leq E_K(q) + SF(N) \cdot \sigma_{q,K}$

where, $E_K(q)$ and $\sigma_{q,K}$ are given from Eqs. (A-4) and (A-5) with $\lambda = \lambda_1$.

$T = T_K$.

C. Numerical Results and Evaluation

The analysis of the previous section has been carried out with $D_{.95} = 50$ msec and $K = 1, 2, 3, 4$ corresponding to frame durations $T_1 = 50$ msec, $T_2 = 25$ msec, $T_3 = 16.66$ msec, and $T_4 = 12.5$ msec. The minimum number N of capacity "quanta" (a quantum can accommodate one time critical packet length ℓ) is varied from 1 to 45. The corresponding tolerable time critical packet rates λ_K , $K = 1, 2, 3, 4$, have been computed. A fairly simple

time driven simulation program has been used to simulate the M/D/N queue recursive equation (see Appendix). Experiments have been conducted to evaluate the probability P_K of violating the 50 msec delay. The simulation was run over 10,000 frames and statistics have been taken after the first 1,000 transient frames.

In Table 2, we tabulate the analytically obtained λ_K for $N = 1, \dots, 45$ and the corresponding simulation result P_K . In the same table, we show the various SF (spread factor) choices. These results are plotted in Figures 7 and 8. Specifically, in Figure 7, we show the maximum tolerable rate, λ_K versus the total number of capacity quanta dedicated to a particular ES during 1 sec (N slots in a $T_2 = 75$ msec scheme correspond to $2N$ slots in the $T_1 = 50$ msec case). The quite striking observation from Figure 7 is the considerable improvement obtained under the $T_2 = 25$ msec frame over the $T_1 = 50$ msec one. Their improvement, ranging from 50% to 70%, can be understood by observing that the one frame queueing, tolerated under the $T_2 = 25$ msec frame scheme, has an averaging effect on the bursty Poisson arriving traffic. It is remarkable to see that further partitioning of the frame (by a factor of 3 and 4) does not lead to any substantial improvement; this is due to the fact that the 95% confidence constraint limits the utilization to low levels and, hence, the probability of large queue sizes, requiring more than two frames queueing, is very small.

In Figure 8, we plot the simulation derived probabilities of violating the 50 msec constraint for the (N, λ_K) pairs computed using the analytic technique described above. These probabilities fall below the 5% specification, thus indicate that our analysis assumptions are valid and, even more, rather conservative. Finally, the histograms of the queue size distribution are plotted in Figure 9 for sample values of N and the extreme cases $K = 1$ and $K = 4$. These histograms, obtained from the 10,000 frame simulations, show that the queue size distributions have similar shapes for a given N but differ for different N values. Hence, our algorithm, which evaluates for every choice of N a corresponding spread factor $SF(N)$, is based on a reasonable assumption.

Note, that the results plotted in Figure 7 are independent of the time critical packet size (equal to the capacity quantum size). Figure 7 can be used as a working tool in order to evaluate the minimum allocation per FS for frame sizes $T = 50$ msec and $T \leq 25$ msec. It can be easily seen from the figure that frame lengths shorter than 25 msec does not provide substantial improvement. We can, therefore, use the 25 msec frame to approximately evaluate the minimum capacity for any value $T < 25$ msec. As an example, an ES with time critical packet arrival rate $\lambda = 150$ packets/sec and frame length $T = 20$ msec, needs under the 25 msec curve of Figure 7, 200 quanta/sec or 4 quanta per 20 msec frame. This technique relaxes the simplifying assumption (8) of integer number of frames to the $D_{.95} = 50$ msec delay specification. In case that no other constraints are imposed (e.g., synchronization, etc.), the frame length should be set equal to $D_{.95}/2 = 25$ msec for near-optimal utilization and reduced frame fragmentation.

IV. CONCLUSIONS

The various satellite access alternatives for the time critical traffic in a general purpose satellite packet network have been analyzed and evaluated. The stringent end-to-end delay specification has imposed a 50 msec access delay with 95% confidence. Two general categories have been considered. The dedicated subchannel, first category, consists of access schemes which decouple the time critical traffic service from the total bulk traffic access mechanism. The dedicated subchannel is used for the direct access of outstanding time critical packets. The inevitably introduced contention should be kept such that the delay constraint is not violated. The analysis has clearly shown that slotted ALOHA random access reduces substantially the system efficiency over the dedicated TDMA subchannel strategy. Furthermore, the TDMA scheme is superior in terms of its stability and robustness and the inherent capability to provide a control and reservation subchannel for the bulk traffic. The bulk low priority traffic can be served via any demand assignment scheme involving a traffic adapting control mechanism, e.g., explicit reservations. The mechanism of decoupling the time critical component of the traffic is, however, associated with considerable protocol complexity and involves various fragmentation overheads.

The second category consists of the Priority TDMA scheme which provides a uniform method for all traffic categories. The accessing ES's are responsible to schedule the pending traffic within their transmitting bursts according to priority rules. A centralized control mechanism partitions the transponder capacity into dedicated variable size time slots, comprising a frame, such that 95% of the outstanding time critical traffic will be given access within 50 msec. An M/D/N queueing analysis has been used to evaluate the minimum capacity assignments which guarantee the validity of the delay constraint over a range of anticipated time critical traffic rates. A frame length of 25 msec has been found to provide

near optimal capacity utilization. Among the obvious advantages of this scheme, we mention its stability and robustness, single logical link protocol simplicity, reduced fragmentation overhead and flexibility to accommodate on priority the ES traffic mix. Its disadvantage lay on its rigidity in real time adapting to fast traffic variations among ES's.

As a general conclusion, we feel that the priority TDMA scheme provides a simple and attractive access candidate, especially in the case of ES's collecting data from a large and relatively fixed terminal population. The bursty character of a terminal traffic is then averaged over the entire population and does not dramatically affect the transponder partition requirements. Non real-time bandwidth adjustments are, of course, possible via a centralized reallocation protocol.

APPENDIX: M/D/N QUEUE STATISTICS

Let T be the frame duration and λ the time critical packet arrival rate in packets/sec. The queue state q_j will be the number of packets in queue, just after the opening of the j^{th} frame. With v_j , new arrivals generated during the j^{th} frame and, at most N packets served per frame, as in Figure A.1, the queue difference equation will be:

$$q_{j+1} = (q_j - N)^+ + v_j$$

where,

$$a^+ = \begin{cases} 0 & \text{if } a < 0 \\ a & \text{if } a \geq 0 \end{cases}$$

and v_j is Poisson distributed with mean $\rho = \lambda T < N$. From the M/D/N queue analysis in [12], the stationary generating function of the queue state probabilities is given by,

$$G(z) = \frac{(N - \rho)(z - 1)}{1 - z^N \text{EXP}[\rho(1 - z)]} \prod_{r=1}^{N-1} \frac{z - z_r}{1 - z_r} \quad (\text{A-1})$$

where, z_r are the $(N-1)$ roots, $|z_r| < 1$ of the equation,

$$z^N \text{EXP}[\rho(1 - z)] = 1 \quad (\text{A-2})$$

The N^{th} root of (A-1) is the obvious $z_N = 1$. A simple numerical algorithm is given in [13] for the calculation of these roots. With,

$$X_r = \text{Re} [z_r], \quad Y_r = \text{Im} [z_r]$$

it easily turns out that X_r, Y_r satisfy the nonlinear system,

$$X_r = \text{EXP} [\rho/N (X_r - 1)] \cos \left[\frac{2r\pi + \rho Y_r}{N} \right]$$

$$Y_r = \text{EXP} [\rho/N (X_r - 1)] \sin \left[\frac{2r\pi + \rho Y_r}{N} \right]$$

From Eq. (A-1) we get, after some algebra, the following queue size statistics,

$$\text{Pr} \{c < N\} = \sum_{i=0}^{N-1} P(q=i) = \frac{N-\rho}{\prod_{r=1}^{N-1} (1-z_r)} \quad (\text{A-3})$$

Note, that this is the probability that an arrival encounters at most, $N-1$ packets in queue and, hence, gets access immediately into the next frame. The mean and variance of c are given by,

$$E(q) = \sum_{i=1}^{N-1} \frac{1}{1-z_i} + \frac{N-N^2+\rho^2}{2(N-\rho)} \quad (\text{A-4})$$

$$\sigma_q^2 = Q''(1) + Q'(1) - [Q'(1)]^2 \quad (\text{A-5})$$

with,

$$Q'(1) = \sum_{i=1}^{N-1} \frac{1}{1-z_i} - \frac{N^2 + \rho^2 - N - 2\rho N}{2(N-\rho)}$$

$$Q''(1) = \frac{A - B}{N - \rho} - \frac{N(N - 1) - 2\rho N + \rho^2}{N - \rho} \cdot Q'(1)$$

where,

$$A = (N - \rho) \sum_{i=1}^{N-1} \frac{1}{1 - z_i}$$

$$B = (N - \rho) \sum_{\substack{i, j = 1 \\ i \neq j}}^{N-1} \frac{1}{(1 - z_i)(1 - z_j)}$$

REFERENCES

- [1] Jacobs, I.M., et. al., "General Purpose Satellite Networks," Proc. of the IEEE. Vol. 66, No. 11, Nov. 1978, pp. 1443-1467.
- [2] Abramson, N., "Packet Switching with Satellites," AFIPS Conf. Proc., Vol. 42, June 1973.
- [3] Kleinrock, L. and S. Lam, "Packet Switching in Slotted Satellite Channel," AFIPS Conf. Proc., Vol. 42, June 1973.
- [4] Crowther, W., et. al., "A System for Broadcast Communication: Reservation ALOHA," 6th Hawaii Int. Systems Science Conference, Proc., Jan. 1973.
- [5] Roberts, L., "Dynamic Allocation of Satellite Capacity Through Packet Reservations," AFIPS Conf. Proc., Vol. 42, June 1973.
- [6] Binder, R., "A Dynamic Packet Switching System for Satellite Broadcast Channels," ICC-75 Proc., San Francisco, California, June 1975.
- [7] Maglaris, B. and T. Lissack, "A Priority TDMA Protocol for Satellite Data Communications," submitted for publication in the ICC-81, Denver, Colorado.

- [8] Lam, S., "Delay Analysis on a TDMA Channel," IEEE Trans. on Communications, Vol. COM-25, No. 12, Dec. 1977, pp. 1489-1977.
- [9] Aein, J.M. and O.S. Kosovych, "Satellite Capacity Allocation," Proc. of the IEEE, Vol. 65, No. 3, March 1977, pp. 332-342.
- [10] Ng, S.F.W. and J.W. Mark, "Multi-Access Model for Packet Switching with a Satellite Having Processing Capability: Delay Analysis," IEEE Trans. on Communications, Vol. COM-26, Feb. 1978, pp. 283-290.
- [11] Rubin, I., "Access Control Disciplines for Multi-Access Communication Channels: Reservation and TDMA Schemes," IEEE Trans. on Information Theory, Vol. IT-25, No. 5, Sept. 1979, pp. 516-536.
- [12] Syski, R., "Congestion Theory in Telephone Systems," Oliver Boyd, London, 1960.
- [13] Fisher, M., and T. Harris, "A Model for Evaluating the Performance of Integrated Circuit and Packet Switched Multiplex Structure," IEEE Trans. on Communications, Vol. COM-24, No. 2, Feb. 1976, pp. 195-202.

TABLE 1: $D_{.95}$ FOR VARIOUS SF CHOICES

$\lambda = 1$ packet/sec $T = 50$ msec

SF = 4.5 (Chebyshev)

$D_{.95} = 98.1$ msec

SF = 3.0

$D_{.95} = 90.0$ msec

SF = 1.96 (Normal)

$D_{.95} = 77.9$ msec

SF = 1.5

$D_{.95} = 50.2$ msec

TABLE 2: MAXIMUM TIME CRITICAL PACKET RATE

Frame	50 msec		25 msec		16.66 msec		12.5 msec		
N	λ_1	$P_1\%$	λ_2	$P_2\%$	λ_3	$P_3\%$	λ_4	$P_4\%$	SF
1	1	4.72	6	1.20	13	1.60	34	1.00	4.19
2	6	3.70	33	1.88	73	1.40	115	1.16	3.07
3	16	4.66	73	3.20	135	1.38	200	2.08	2.42
4	26	3.94	110	2.74	195	2.18	280	2.20	2.34
5	38	4.32	150	2.98	255	2.56	360	3.82	2.22
6	51	4.66	190	2.96	315	2.64	440	1.54	2.13
7	64	4.54	225	2.12	375	3.48	520	5.22	2.09
8	78	4.56	265	2.90	435	4.02	600	5.00	2.04
9	92	4.34	305	3.12	495	4.02	680	3.80	2.02
10	106	4.26	346	3.20	556	2.90	761	4.48	2.01
15	181	4.46	546	3.78	856	4.04	1151	4.68	1.95
20	261	4.54	746	4.23	1151	3.50	---	---	1.91
25	341	4.73	945	3.87	---	---	---	---	1.90
30	426	4.89	---	---	---	---	---	---	1.85
35	511	4.70	---	---	---	---	---	---	1.84
40	596	4.58	---	---	---	---	---	---	1.84
45	686	5.04	---	---	---	---	---	---	1.79

λ_K = Maximum Rate (Packets/Sec)

P_K = $P_r \{ q/K \geq N \}$ Simulation

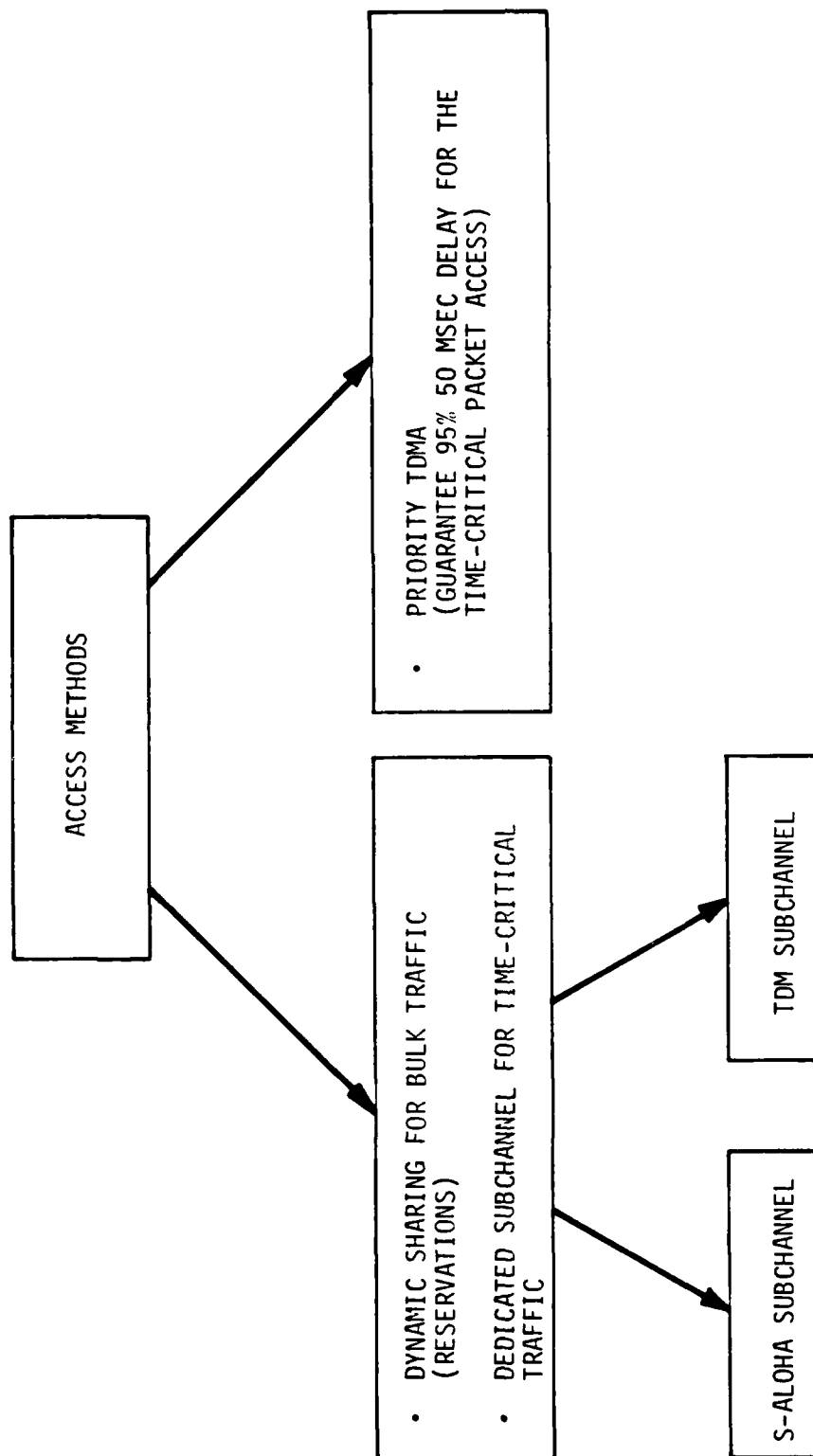
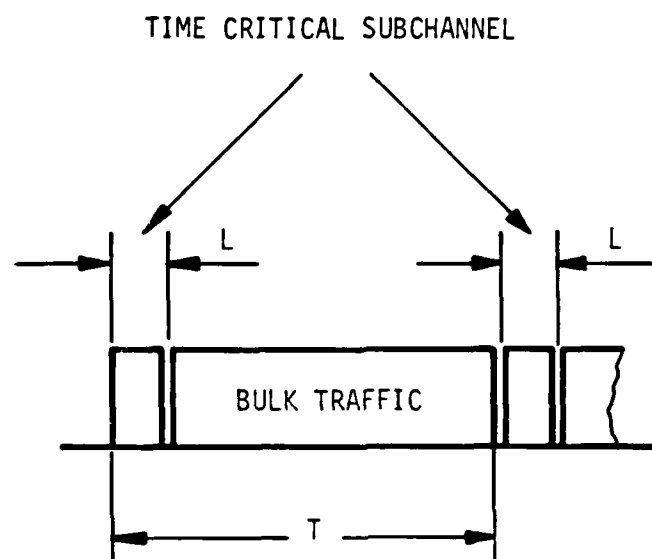


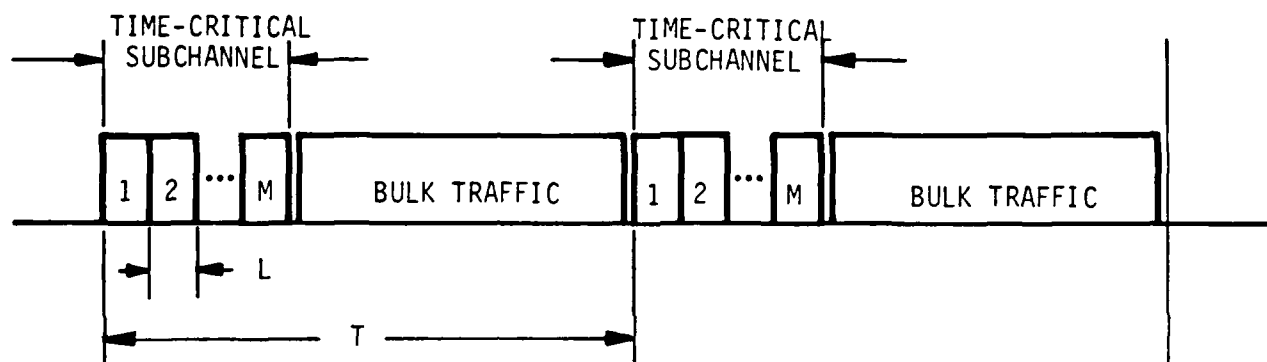
FIGURE 1: ACCESS ALTERNATIVES



T = FRAME DURATION ($T \leq 50$ MSEC)

L = SLOT LENGTH (KBITS)

FIGURE 2: S-ALOHA SUBCHANNEL



M = NUMBER OF ES'S

T = FRAME DURATION ($T \leq 50$ MSEC)

L = SLOT LENGTH (KBITS)

FIGURE 3: TDM SUBCHANNEL

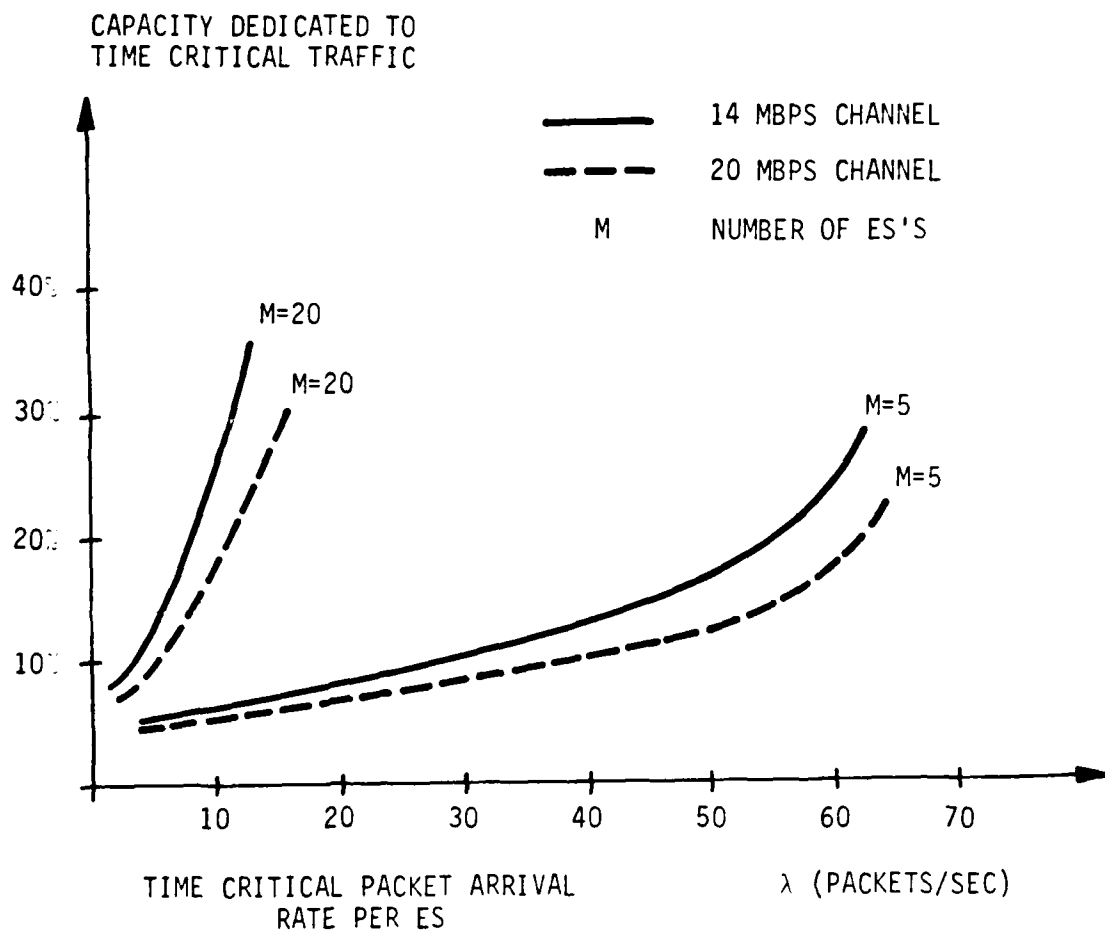


FIGURE 4: DEDICATED SLOTTED ALOHA CAPACITY FOR
50 MSEC 95' DELAY

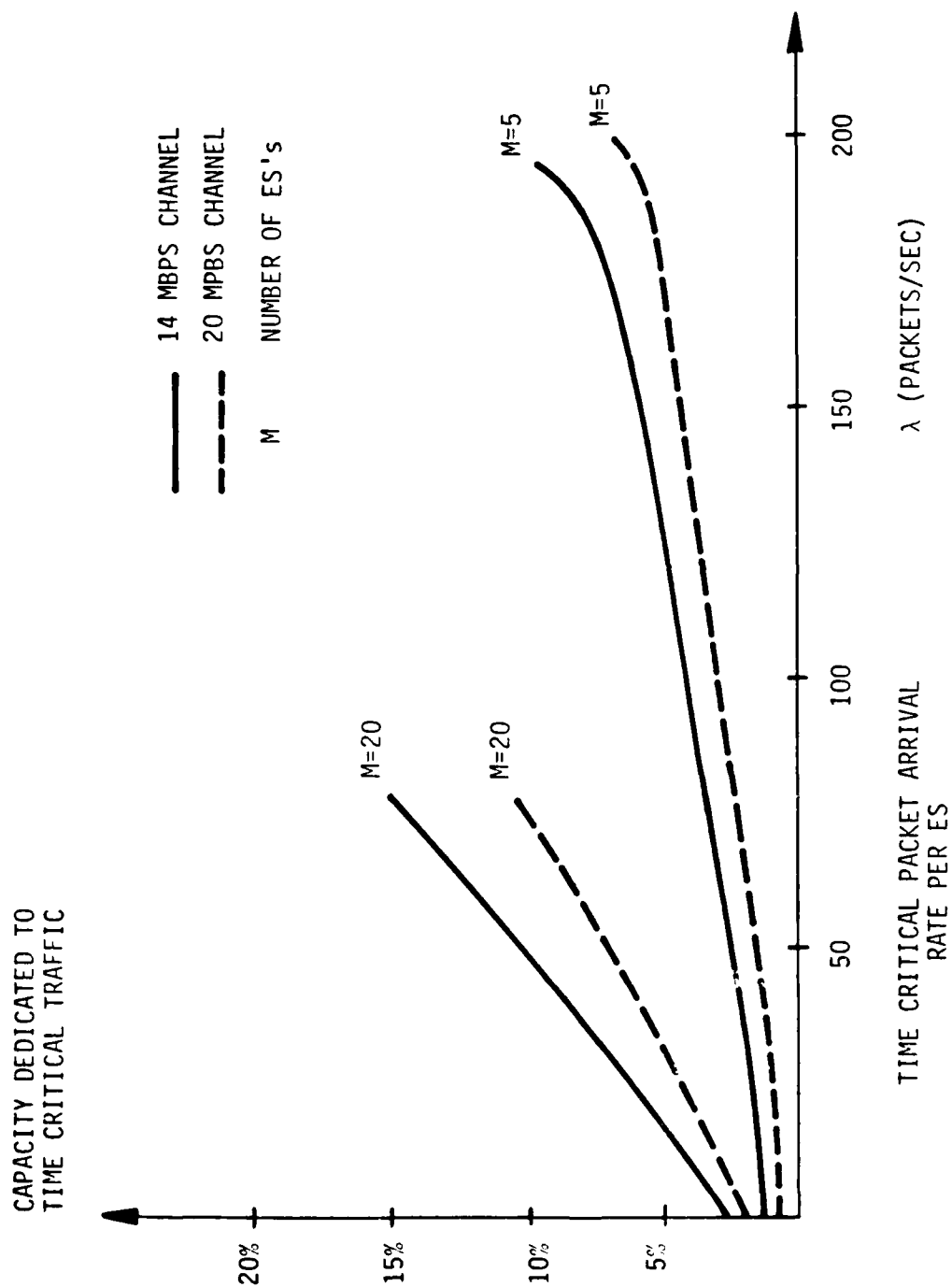
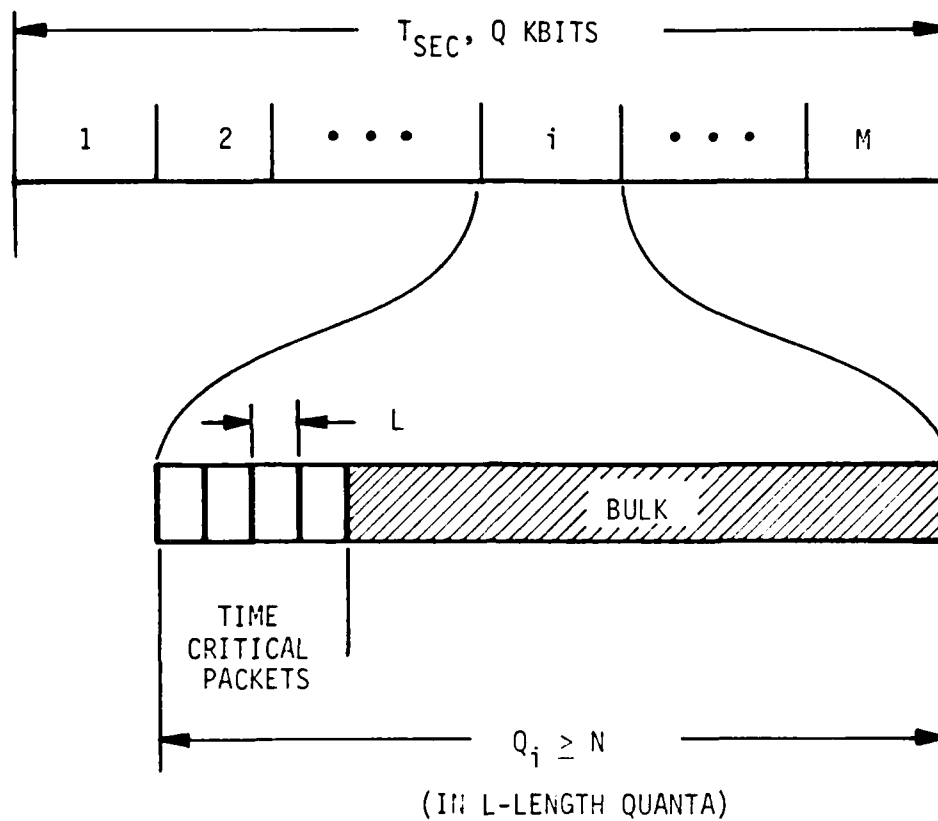


FIGURE 5: DEDICATED TDM CAPACITY FOR
50 MSEC 95% DELAY



- M - NUMBER OF ACCESSING ES'S
- T - FRAME DURATION (SEC)
- Q - FRAME LENGTH (KBITS)
- L - TIME CRITICAL PACKET LENGTH (KBITS)
- Q_i - i^{th} SLOT LENGTH (IN L-LENGTH QUANTA)
- N_i - MINIMUM SLOT LENGTH (IN L-LENGTH QUANTA)

FIGURE 6: PRIORITY TDMA SCHEME

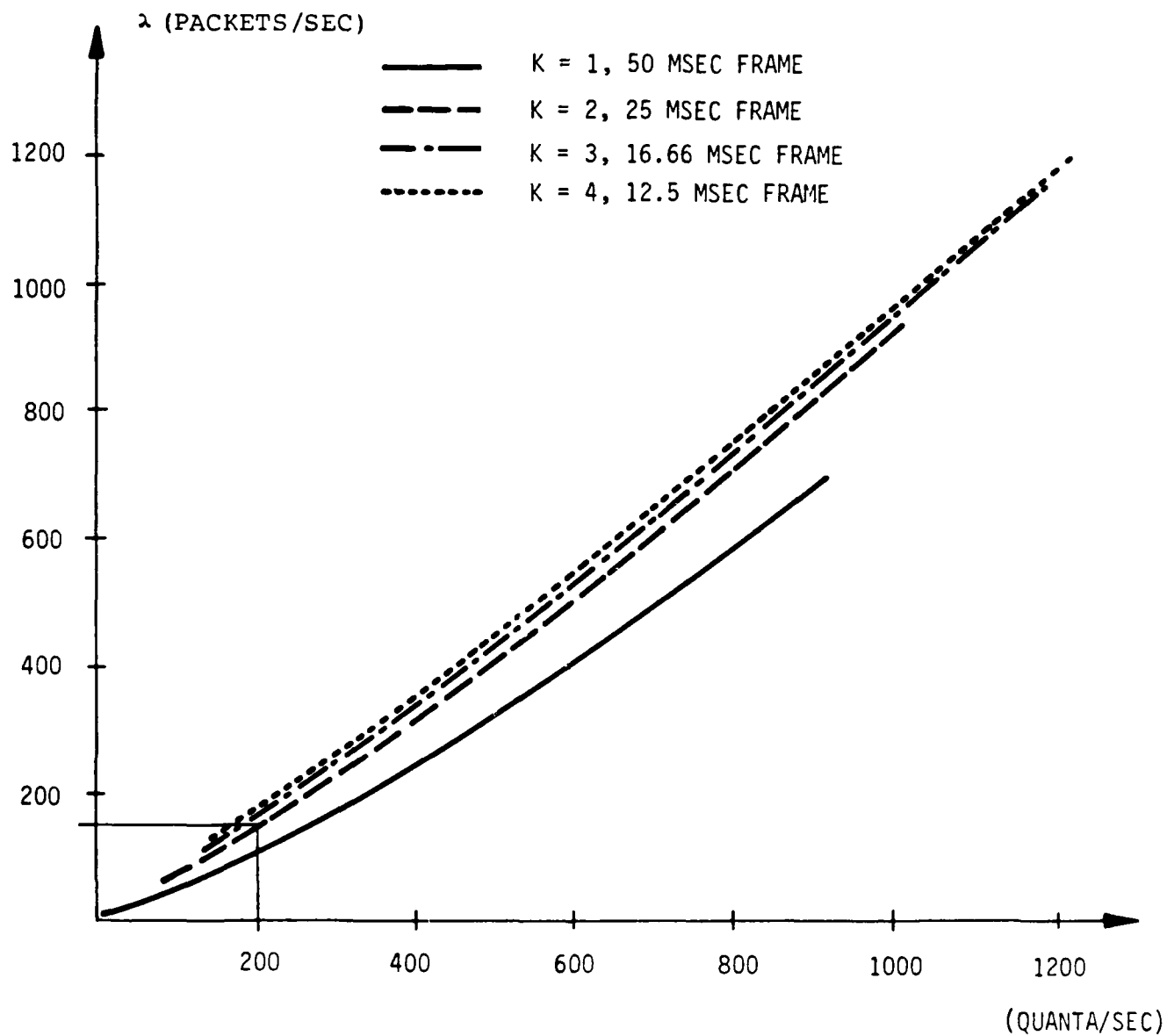


FIGURE 7: MAXIMUM TIME CRITICAL RATE λ VERSUS AVAILABLE NUMBER OF SLOTS/SEC PER ES FOR 50 MSEC 95% DELAY

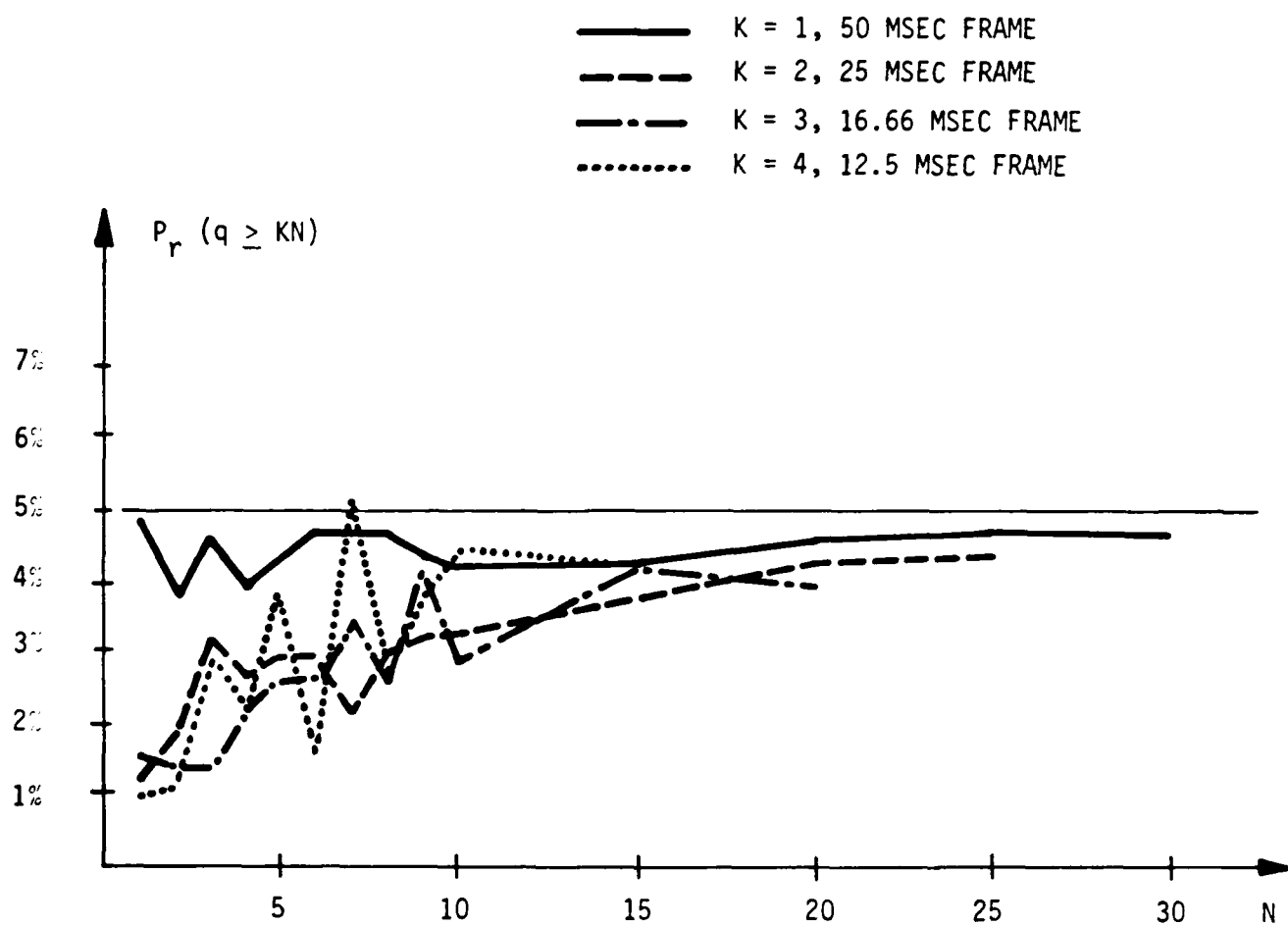


FIGURE 8: SIMULATION RESULTS, 10,000 FRAMES

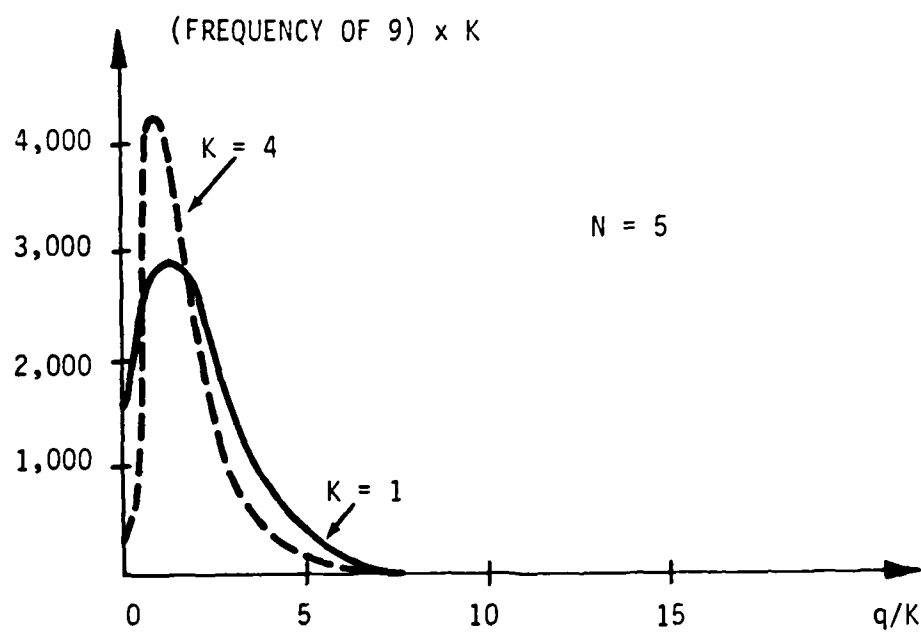
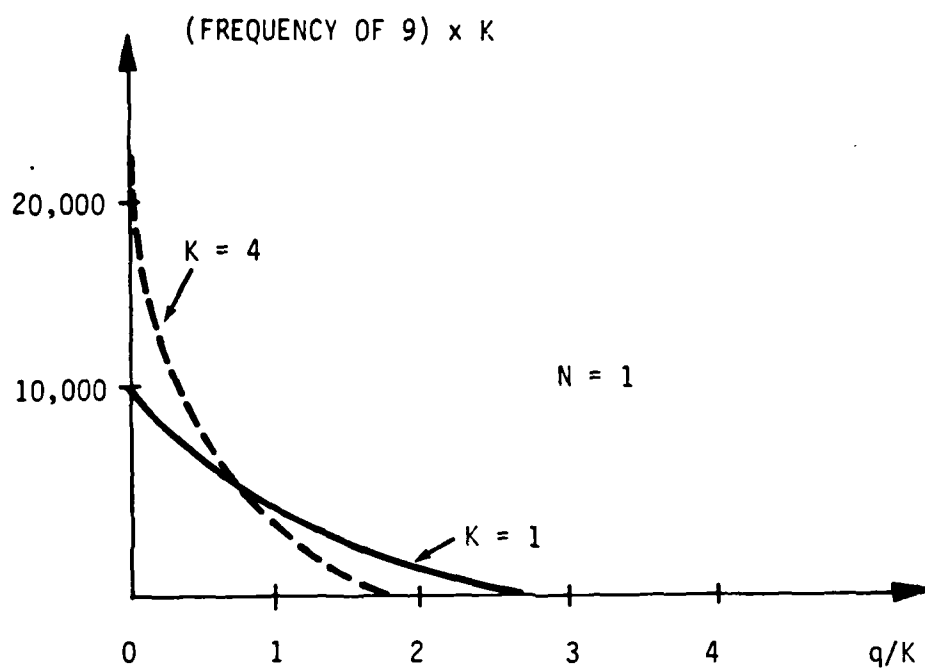
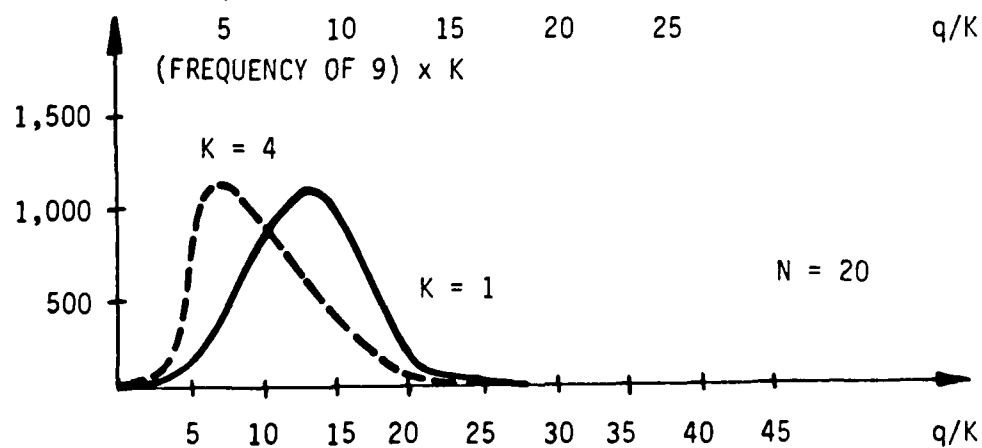
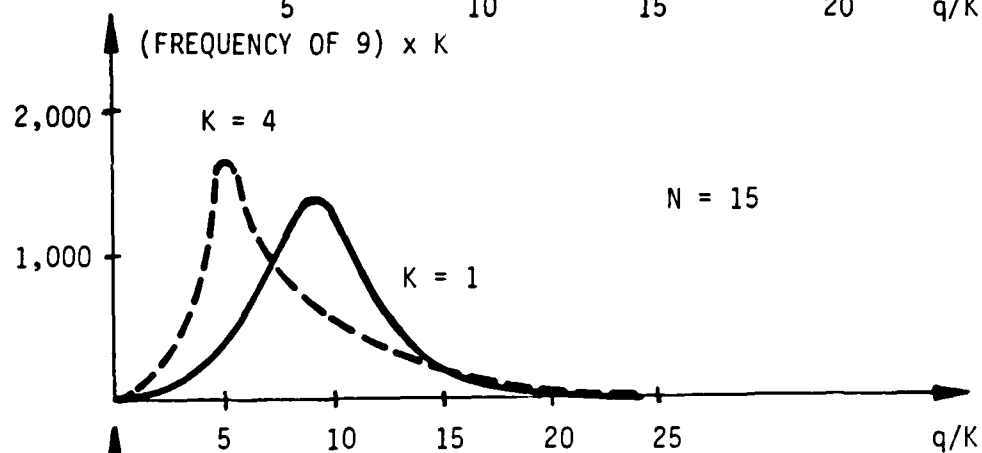
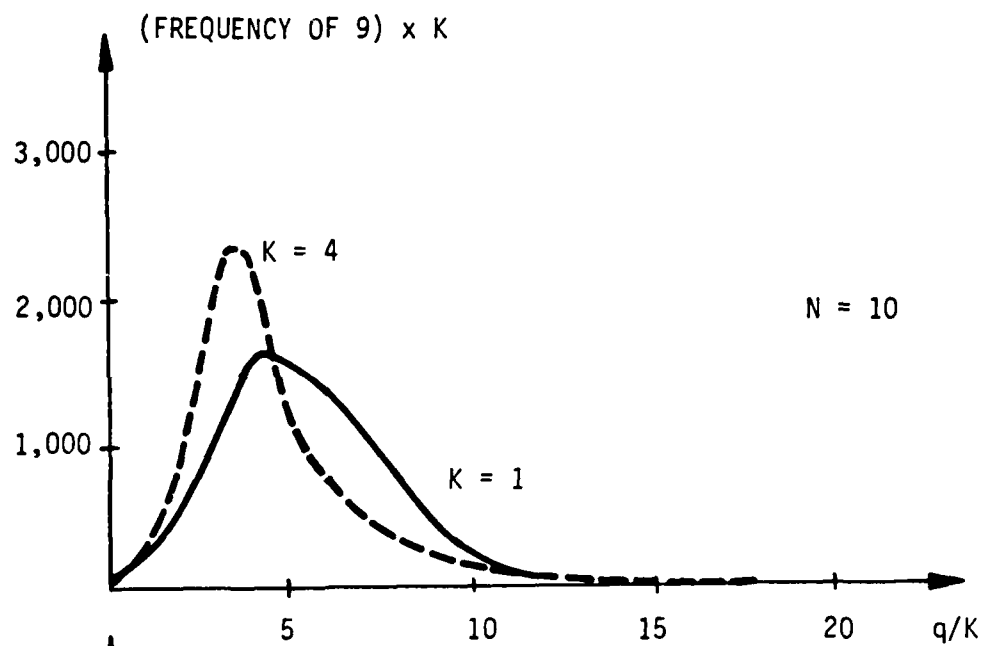
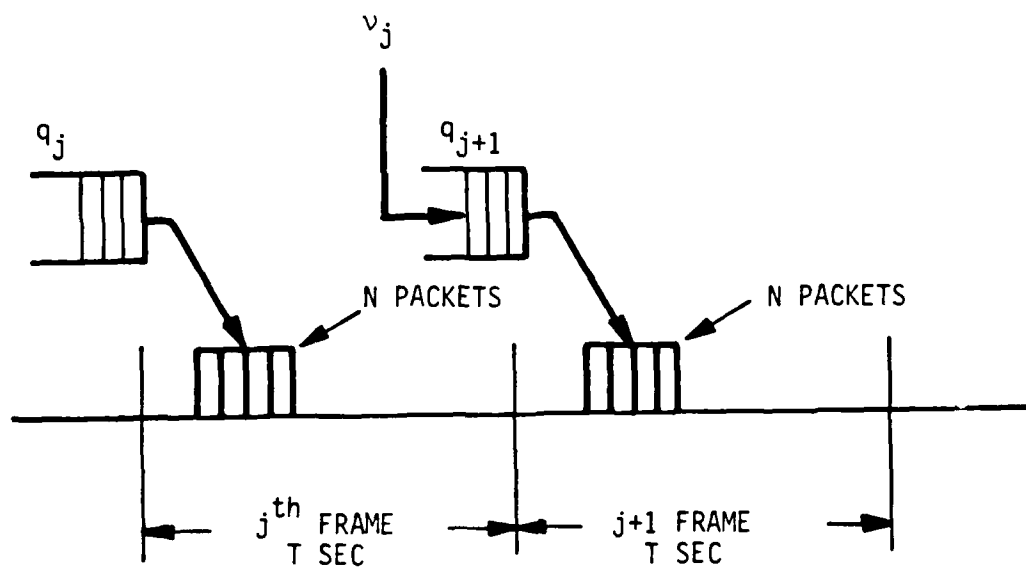


FIGURE 9: SIMULATION RESULTS: QUEUE SIZE DISTRIBUTION OVER 10,000 FRAMES





$$q_{j+1} = (q_j - N)^+ + v_j$$

FIGURE A.1: THE M/D/N MODEL

F. Local Area Networks

F.1 End-to-End Delay Analysis on Local Area Networks:

An Office Building Scenario

Maglaris, Lissack, and Austin

IEEE National Telecommunications Conference, November
1981, New Orleans

END-TO-END DELAY ANALYSIS ON LOCAL AREA NETWORKS: AN OFFICE BUILDING SCENARIO

Basil Maglaris, Tsvi Lissack, and Michael Austin

NETWORK ANALYSIS CORPORATION
130 Steamboat Road
Great Neck, NY 11024

ABSTRACT

Coaxial cable based, broadcast Local Area Networks (LANs) are widely recognized as an effective means to provide both communication capabilities to a variety of office automation equipment as well as interoperability among data processing configurations. This paper presents the results of a performance analysis of a typical environment to which both of these factors contribute. The environment includes a variety of line speeds and protocols and access is made to remote data bases and hosts.

The network carries messages, packetized to the optimal packet length and transmitted via the CSMA/CD protocol (Carrier Sensing Multiple Access/Collision Detection). End-to-end delay and access delay of the CSMA/CD protocol, is studied. It is shown that for a realistic traffic mix, the principal contribution to end-to-end delay arises from the queueing and NIU processing.

Sensitivity of the performance of the LAN to variations in the number of users and the traffic volume is investigated. This analysis yields the result that end-to-end performance is not sensitive to significant increases (up to a factor of 6) in the number of users and the associated traffic volume above the relatively high volumes used in the reference scenario.

1. INTRODUCTION

Recent developments in intra-building or intra-campus communications indicate that local networking is taking a path distinct from traditional long haul communications and computer-to-peripheral data bases. The unique environment of local networks in terms of geographical dispersion (from 100 meters to 10 Km), transmission speed (between 9 Kbps to 50 Mbps), unregulated environment and support of dissimilar devices and protocols is summarized in [1]. Interoperability among various user requirements is the key factor to the development of local networks, often imposing the major cost and performance constraints instead of bandwidth, which was the typical scarce commodity in a more conventional communications environment.

From the early R&D efforts in Local Area Networks (LAN), it became apparent that less controlled random access methods could be very efficient for high-speed broadcast media. This led to the Carrier Sensing Multiple Access protocol with Collision Detection (CSMA/CD) as first introduced in [2]. In parallel, experiments showed that off-the-shelf cable TV technology represents an excellent vehicle for implementing broadcast LAN [3].

More recently, the data communication and office automation users community has shown great interest in local networking. This interest was strongly influenced by extensive vendor activity, new hardware/software announcements, the IEEE standards committee for LAN, and the intensive advertising campaign in the mass media. The technical and scientific literature is flooded by articles with the main controversial issues being the transmission method (baseband versus broadband/RF) [4], [5] and network access scheme [6], [7]. Most of the analytical work is dealing with performance comparisons [8], [9] of the various proposed channel partition methods, in particular the two that have been blessed as IEEE standards by the LAN committee, Carrier Sense Multiple Access with Collision Detection (CSMA/CD) and the Token Ring [10].

The 97% channel utilization capability of ETHERNET reported by Shoch and Hupp [11], completely ignores the processing time at the Network Interface Unit (NIU), the interaction between the CSMA/CD access and higher level protocols, buffer management at NIU, and the nodes speed mismatch. Watson [12] performed simulations of two CSMA/CD type networks with different methods of buffer management and speed mismatch and concludes that network performance is strongly dependent on these factors, sometimes reducing utilization to less than 50%. In [13], we presented a detailed modeling of two microprocessor architectures for local network interface adapters and analyzed their impact on throughput/delay performance on end-to-end (ETE) character transfer. Our studies showed that the main limitation to LAN performance is due to the interface processing and queueing, while its sensitivity to the coaxial cable speed and access protocol is rather limited.

In this paper, we do not attempt to model closely the NIU architecture at the microprocessor level. Having established in [13] that the bottleneck exists on the protocol processing and the Packet Assembly/Disassembly (PAD), we assess at the macroscopic level the ETE delays for a multiple applications scenario, typical of an office automation environment. The NIU is modeled as a single server queue, where messages are input from a Data Terminal Equipment (DTE), (e.g., a TTY, a minicomputer port, a word processor, a digital FAX machine), packetized appropriately and transmitted to another NIU via the CSMA/CD protocol. Both packet and message delays are analyzed among various types of DTE's. Before proceeding in describing the mathematical models and presenting the experiments, we establish in a rigorous manner the delay definitions:

- **Packet Delay:** From the time the first bit of a packet is ready to be sent at the source NIU to the time the last bit is received at the destination NIU.
- **Message Delay:** From the time the first bit of the message is ready to be sent over the source DTE access channel, to the time the last bit of the message is received at the destination DTE.

(A message is typically chopped into several packets which may be interleaved as they travel through the network).

In what follows, we first introduce the models for the packet and the message delay analysis. We then describe the data-base of the office building scenario and subsequently present results and sensitivity analysis.

2. MATHEMATICAL MODELING

A) Packet Delay Analysis

The packet delay will consist of the transmission time the packet spends in the coaxial cable and the waste and back-off delays due to busy condition or collisions. We follow the approximate CSMA/CD analysis presented in [14], adapted to reflect various classes of packet lengths depending on the application type.

Let T_i be the transmission time of packet class i , (including all link level overhead) and y be the propagation delay (in nsec/ft). Then the normalized propagation delay relative to packet size of class i is

$$a = \frac{y \cdot L}{T_i}$$

where L is the maximum cable length. This assumption is a pessimistic one, since it implies that all packet transmissions will encounter the maximum possible propagation delay.

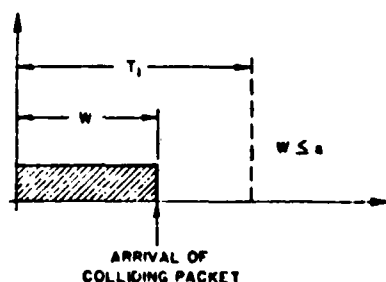


FIGURE 1: ILLUSTRATION OF THE AVERAGE WASTED INTERVAL DUE TO COLLISION W

The aggregate class i packet arrival rate summed over all DTE's transmitting class i packets, is denoted by λ_i . It follows that the total cumulative packet rate S , normalized to T_1 is

$$S = T_1 \sum_i \lambda_i$$

S is related to the total offered packet traffic G , including retransmissions due to collisions and busy conditions, as

$$S = \frac{Ge^{-aG}}{(X + a) Ge^{-aG} + (1 + aG)(1 - e^{-aG})^2 + 1}$$

where X is the packet length, weighted by the class mix and normalized to T_1 :

$$X = \frac{\sum_i \lambda_i T_i}{T_1 \sum_j \lambda_j}$$

The packet delay per class i will be

$$D_i = (A_1 + A_2 + A_3) T_i$$

when A_1 is the waste due to collisions, A_2 the dead time due to the back off algorithm in case of collision or carrier busy and as the propagation and transmission time (all normalized to T_1). It can be shown that

$$A_1 = N_2 (N + a)$$

$$A_2 = \frac{R_2}{T_1} (2 N_2 + 1 - 1) + (N_1 - N_2) \frac{R_1}{T_1}$$

$$A_3 = a + \frac{T_i}{T_1}$$

where

$$W = \frac{1}{G} - \frac{e^{-aG}}{G} - ae^{-aG} \quad (\text{the waste due to collisions, Figure 1})$$

$$N_1 = \frac{G}{S} - 1 \quad (\text{the average number a packet has to back-off})$$

$$N_2 = (1 + aG) e^{-aG} - 1 \quad (\text{average number a packet collides})$$

R_1, R_2 are the mean retransmission intervals in case of busy condition, or collision in which case the binary back-off algorithm is applied as in [2].

B) Message Delay Analysis

This delay will consist of the serialization delay at the DTE port, the protocol execution time at the source and destination NIU, the CSMA/CD packet delays as computed above and queuing delays at the source (a packet waits to be transmitted) and destination (packets wait to be reassembled into a message to be transmitted over the serial DTE port).

Several types of DTE's will be considered, each transmitting or receiving various classes of messages. Let

$D_i(j, k)$	The ETE delay of message class i from a DTE of type j to a DTE of type k
T_N	Protocol execution time
D_i	Packet delay due to CSMA/CD
$TR_i(j)$	Class i packet transmission time and type j DTE
M_i	Average number of packets per message of class i
$Q_i^{IN}(j)$	First packet queueing of class i messages on access queue of DTE type j port
$Q_i^{OUT}(j)$	Packet queueing of class i , at destination queue of DTE type j port

In case the source DTE is a low speed DTE or host connected via multiple parallel ports, then $Q_i^{IN}(j) = 0$, since no queueing is practically taking place at these access ports. For DTE's connected via a high speed port, the NIU may have to buffer incoming messages to avoid overflow. To compute $Q_i^{IN}(j)$, we note that a message will be input to the NIU as a whole. Thus, the M/M/1 formula on a message basis can be used (assuming Poisson message arrivals and exponentiated message length) to compute the message queueing reflected at the first packet.

At the destination DTE packets belonging to a message, will not follow one another at regular intervals and may be interleaved with other message packets. We assume that received packets are Poisson distributed and have exponential length. Thus the M/M/1 formula on a packet basis can be used to provide $Q_i^{OUT}(j)$.

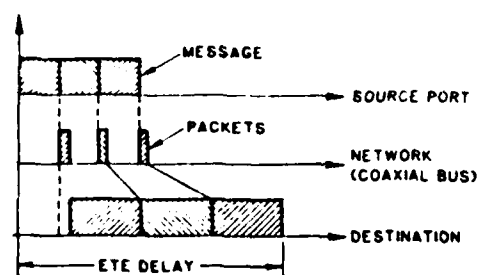
To compute the ETE message delay we distinguish between two cases.

Case 1 - The delay bottleneck occurs at the destination NIU. This will happen if the source channel speed is higher than the destination channel speed. In this case, the message delay will be dominated by the packet queueing and transmission time at the destination DTE port, $M_i [Q_i^{OUT}(k) + TR_i(k)]$. Other components to the ETE delay include message protocol processing $2T_N$, the CSMA/CD delay D_i and the first packet queueing and transmission over the source port, as illustrated in Figure 2a. It follows that

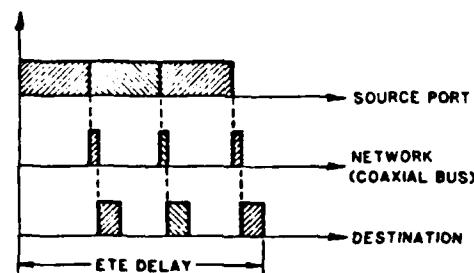
$$D_i(j, k) \approx Q_i^{IN}(j) + TR_i(j) + 2T_N + D_i + M_i [Q_i^{OUT}(k) + TR_i(k)]$$

Case 2 - The delay bottleneck is the source NIU. This occurs if the source channel speed is lower than the destination channel speed. Here the message residence (ETE delay) will be dominated by the transmission time at the source port $M_i TR_i(j)$ as in Figure 2b. Thus

$$D_i(j, k) \approx Q_i^{IN}(j) + M_i TR_i(j) + 2T_N + D_i + [Q_i^{OUT}(k) + TR_i(k)]$$



A. SOURCE PORT SPEED > DESTINATION PORT SPEED



B. SOURCE PORT SPEED < DESTINATION PORT SPEED

FIGURE 2: ILLUSTRATION OF ETE MESSAGE DELAY

3. REQUIREMENT ASSUMPTIONS

The models above have been used to evaluate delay/throughput characteristics for a typical office building LAN. The maximum length of the single coaxial cable was assumed 5,000 feet (enough to span a 20 story average office building). The LAN was assumed to support 1,000 terminals, 10 minicomputers, 25 word processing centers (each including several CRT's and floppy disk drivers), and 20 digital FAX units. Communications with the outside world were supported via a high-speed gateway (1.544 Mbps) to some long-haul T_1 - rate line.

In Tables 1, 2, and 3 we summarize the device characteristics and the traffic volumes they feed to the network, the message types and the routing of messages both internally and to the gateway. Note that traffic numbers are purposely exaggerated. For example a CRT terminal is assumed to generate 120 messages/hour, each message approximately consisting of 40 characters. This corresponds to an operator being able to type 80 characters/min. continuously without waiting for a response! Similarly the number of devices in the building exceeds current office automation needs.

4. NUMERICAL RESULTS

The packet and message delay analysis were implemented in a computer package called PLAN (Performance of Local Area Networks). This program accepts as input a Data Base on requirements in a format similar to Tables 1, 2 and 3 and outputs packet and message ETE delays per device-type pair.

TABLE 1: DEVICE CHARACTERISTICS

Device Type	Access Number (Kbps)	Message Speed (Msg/hour)	Rate
1. Gateway (GW)	1	1,544.0	*
2. Word Processing	25	9.6	120
3. Asynch. Terminals	600	1.2	120
4. BSC Terminals	400	9.6	120
5. Mini Computers	10	(**)	(**)
6. Low Speed FAX	10	1.2	5
7. High Speed FAX	10	9.6	50

(*) The gateway traffic is evaluated from the message routing information (Table 3).

(**) Two configurations are considered:

(1) Multiple Parallel ports per mini

- File transfer ports at 56 Kbps, generating 5 messages (files)/hour.
- Asynch. ports at 1.2 Kbps. The message volume to/from these ports is computed from the routing information.
- BSC ports at 9.6 Kbps. The message volume to/from these ports is computed from the routing information.

(2) Single Multiplexed port at 128 Kbps.

TABLE 2: NETWORK/TRAFFIC PARAMETERS

Packet Header	256 bits
Cable Capacity	1 - 10 Mbps
Cable Length	5000 feet
Retransmission	10 Packets
Protocol Processing	40 sec/bit (*)

Message Class	Length (Information Kbits)
Terminal Input	0.48
Asynch Output	2.00 (**)
BSC Output	8.00 (**)
Word Processing	16.00
File	200.00
FAX (page)	1,000.00

* As assessed in [13].

** The output to the terminal will be much larger than what the operator types in.

TABLE 3: ROUTING ASSUMPTIONS

From	To
WP	GW, WP (50%, 50%)
TERMINAL	GW, MINI, TERMINAL (75%, 12.5%, 12.5%)
FAX	GW (100%)
MINI (file XFER)	GW, MINI (70%, 30%)
MINI (other)	TERMINAL (symmetric to other direction)
GW	FAX (3 times the other direction)
GW	MINI, TERMINAL, WP (symmetric)

Two classes of experiments were performed using PLAN. The first class included studies on effects of network load, cable speed and packet length to the packet access (CSMA/CD) delays. The second class consisted of experiments on ETE message delays and their sensitivity to various factors.

A. Experiments on Packet Delay

In Figure 3 we plotted the average packet delay for the data-base of section 3 (hereafter referred to as reference scenario) under various packet length choices. Very small packet sizes (less than 2 Kbits) result into considerable message fragmentation and consequently large protocol overhead. On the other hand, larger packet sizes result into unfairness (a single packet will take over the cable for a prolonged time), need for large buffer sizes and incompatibilities with the device access method. Thus, the optimal choice for a packet length as indicated in Figure 3 is 4 Kbits above which no significant performance improvement is achieved. Note that if a message is less than 4 Kbits it will form a single packet for transmission.

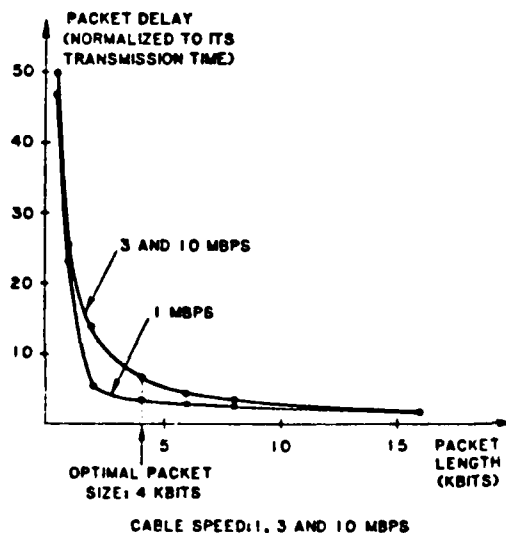


FIGURE 3: EFFECT OF PACKET LENGTH (INFO BITS) ON THE AVERAGE PACKET DELAY

In Figure 4, the packet delay is given as a function of the aggregate load, varying up to 3 Kbps (the cable capacity). Our reference scenario, generates approximately 300 Kbps. Thus the cable is utilized only at 10% and the margin for growth, insensitive to packet delays, is up to 65% utilization (2 Mbps). This demonstrates that the access method is not a significant factor to delays under a wide range of overload scenarios. Note that our reference scenario is by far exaggerating traffic volumes in a typical office building.

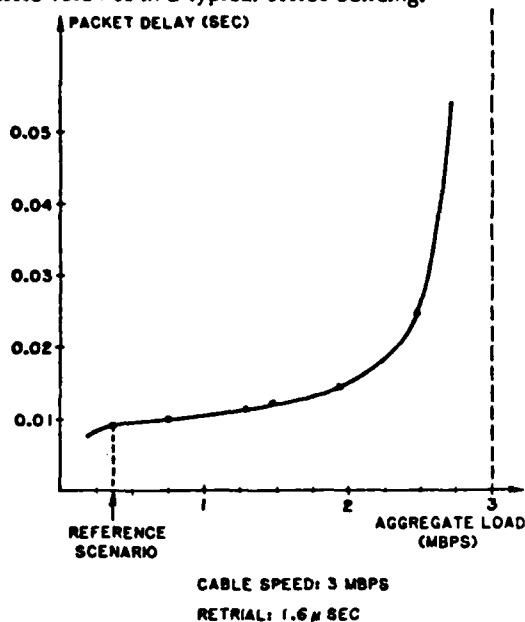


FIGURE 4: EFFECT OF BUS LOAD ON PACKET DELAY

B. Experiments on Message Delays

In Table 4 we depict the ETE message delays under the parallel-port and single-multiplexed minicomputer attachment (as expected, the single high-speed port at 128 Kbps performs better than parallel ports although the basis of comparison is rather weak). Note that delay can be rather large (of the order of minutes) due to the device access speed and queueing. This is further demonstrated in Figure 5, whereby queueing at the access port of Word Processing centers (WP) results into considerable delay degradation as message rate increases. Again for our reference scenario, and its vicinity, delays are rather insensitive. In Figure 6, we show the dramatic effect of the single-multiplexed port speed on minicomputer traffic as it results from the reference scenario.

Subsequent experiments showed that there is no significant sensitivity on the number of devices, assuming that the aggregate load remains constant.

5. CONCLUSIONS

In summary we conclude that the CSMA/CD access delay is of the order of milliseconds, orders of magnitude under the message ETE delays. These are due mainly to limited device access speeds and queueing at the serial port at the DTE-NIU interface. Furthermore, it was established that coaxial cable LAN's can support extensive office automation requirements up to six times a reference scenario, already overloaded without any significant performance degradation.

Note that the LAN technology can offer a multiplicity of the 3 - 10 Mbps capacity either via interconnecting among various coaxial cable structures or by exploring the virtually unlimited capability of RF frequency division multiplexing in the VHF/UHF CATV bands.

TABLE 4: END-TO-END MESSAGE DELAY (SEC)

REFERENCE SCENARIO - MULTIPLE PORTS/MBPS 10 MBPS CHANNEL							
	GP	WP	MBP	ASYN	BSC	HS-FAX	LS-FAX
GP	---	2.2	19.8	1.9	0.9	177	22
WP	1.9	2.8	---	---	---	---	---
MBP	19.8	---	19.8	3.8	1.3	---	---
ASYN	0.6	---	1.2	1.2	0.7	---	---
BSC	0.99	---	0.7	0.7	0.2	---	---
HS-FAX	177	---	---	---	---	---	---
LS-FAX	22	---	---	---	---	---	---

REFERENCE SCENARIO - SINGLE PORT/MBPS 10 MBPS CHANNEL							
	GP	WP	MBP	ASYN	BSC	HS-FAX	LS-FAX
GP	---	2.2	19.8	1.9	0.9	177	22
WP	1.9	2.8	---	---	---	---	---
MBP	0.2	---	18.8	1.9	0.9	---	---
ASYN	0.6	---	0.7	1.2	0.7	---	---
BSC	0.99	---	1.14	0.7	0.2	---	---
HS-FAX	177	---	---	---	---	---	---
LS-FAX	22	---	---	---	---	---	---

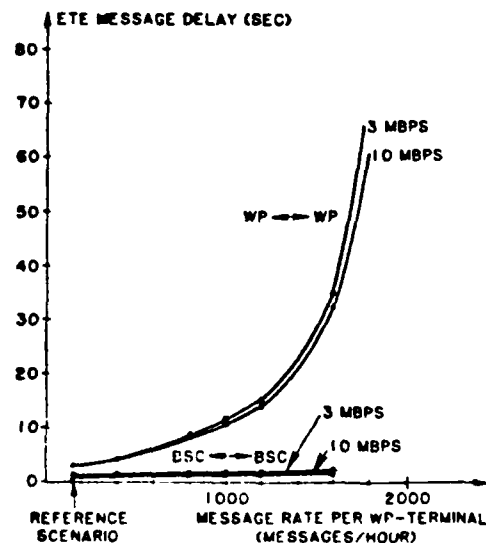


FIGURE 5: EFFECT OF MESSAGE RATE ON END-TO-END MESSAGE DELAY: WORD PROCESSING AND BSC-ASYN TERMINALS

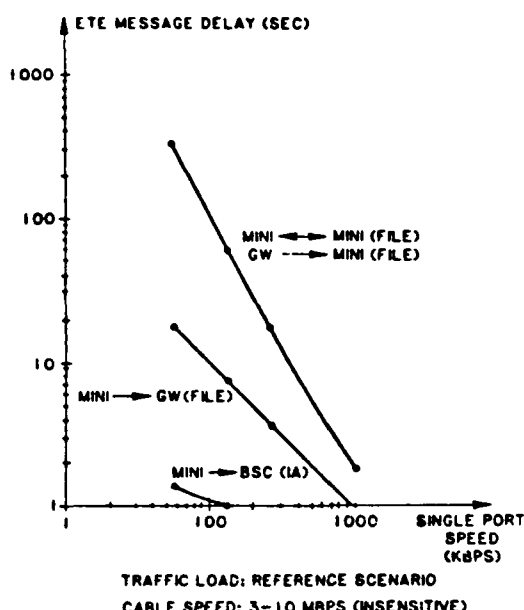


FIGURE 6: EFFECT OF PORT SPEED ON MINI-COMPUTER
MESSAGE ETE DELAY: SINGLE PORT

REFERENCES

1. Clark, D., et al. "An Introduction to Local Area Networks," Proceedings of the IEEE, Vol. 66, No. 11, November 1978, pp. 1497 - 1517.
2. Metcalfe, R. and D. Boggs, "Ethernet - Distributed Packet Switching for Local Computer Networks," Communications of the ACM, Vol. 19, No. 7, July 1976, pp. 395-404.
3. Frisch, I., "Experiments on Random Access Packet Data Transmission on Coaxial Cable Video Transmission Systems," IEEE Transactions on Communications, Vol. COM-25, No. 10, October 1977.
4. Dineson, M.A. and T.T. Picazo, "Broadband Technology Magnifies Local Networking Capability," Data Communication, February 1980, pp. 61-79.
5. Maglaris, B. and T. Lissack, "An Integrated Broadband Local Network Architecture," Proceedings of the 5th Conference on Local Computer Networks, Minneapolis, Minnesota, October 1980, pp. 86-93.
6. Christiansen, G.S. and W.R. Franta, "Design and Analysis of the Access Protocol for Hyperchannel Networks," 3rd USA-Japan Computer Conference, San Francisco, October 1978, pp. 86-93.
7. Tobagi, F.A., "Multiaccess Protocols in Packet Communication Systems," IEEE Transactions on Communications, Vol. COM-28, No. 4, April 1980, pp. 468-488.

8. Tobagi, F.A. and V. Bruce Hunt, "Performance Analysis of Carrier Sense Multiple Access with Collision Detection," Proceedings of the LACN Symposium, May 1979, pp. 217-245.
9. Bux, W., "Local Area Networks: A Performance Comparison," Local Area Network Workshop, August 1980, Zurich, Switzerland.
10. Kirstein, P.T. and S.R. Wilbur, "University College London Activities with the Cambridge Ring," Local Area Network Workshop, August 1980, Zurich, Switzerland.
11. Shoch, J.F. and J.A. Hupp, "Performance of an Ethernet Local Network - A Preliminary Report," Proceedings of 20th Computer Soc. Intl. Conf., San Francisco, California, February 1980.
12. Watson, W.B., "Performance in Contention Bus Local Network Interconnection," Technical Report, Lawrence Livermore National Laboratory, Livermore, California, 1980.
13. Lissack, T., B. Maglaris and H. Chin, "Impact of Microprocessor Architectures on Performance of Local Network Interface Adaptors," Proceedings of the International Conference on Local Networks and Distributed-Office Systems, London, May 1981.
14. Sherman, R.H. et al., "Concepts, Strategies for Local Data Network Architectures," Data Communications, July 1978, pp. 39-49.

F.2 Performance Evaluation of Interface Units for Broad-
cast Local Area Networks

IEEE COMPCON, September 1982, Washington

Maglaris and Lissack

PERFORMANCE EVALUATION OF INTERFACE UNITS FOR BROADCAST LOCAL AREA NETWORKS

Basil S. Maglaris *

Tsvi Lissack

Polytechnic Institute of New York
Brooklyn, New York

ConTel Information Systems
Great Neck, New York

ABSTRACT

This paper deals with analysis and design issues encountered in the development and evaluation of microprocessor-based interface adapters to broadcast carrier sensing multiple access (CSMA) local computer networks. Queueing network models are presented and approximate analytic techniques are used to obtain throughput/delay measures associated with the interface processors.

In the first part of the paper, the protocol partition issue is studied as it is implemented via two stage architectures. It is shown that off-loading link level functions to the terminal end CPUs, improves dramatically the system throughput versus partitions between separate intelligent boards at the terminal and the network ends of the interface unit. In the second part, a closed queueing network model is used to study the interface throughput under window flow control. It is shown that the main limitation is due to protocol processing, while performance is unaffected by parameters referring to the network overall utilization and speed within a wide region of operation.

1. INTRODUCTION

Local area networks (LANs) are rapidly emerging as a major and distinct class of computer communications networks. Their salient characteristics make them particularly attractive as an efficient and economic solution to high speed, unregulated connections within a limited distance geographical area [1]. From the early R&D efforts, it became apparent that random access protocols could be more efficient for high speed broadcast media over conventional contention resolution techniques used in long haul networks. This led to carrier sense multiple access protocols (CSMA), such as the Ethernet access scheme [2], employing collision detection and the hyperchannel access method [3], a hybrid between CSMA and polling. Another class of access protocols use token passing techniques, a variation of hub polling [4].

Various design issues have been raised by LAN vendors and users. Mainly concerning access protocols, network topology, and transmission technology. On access protocols, extensive study have demonstrated relative merits of the two prevailing techniques, CSMA with collision detection and token passing, e.g. [5], [6], and [7]. Among topological designs, bus (tree) networks are preferred for CSMA protocols and rings for token passing. Finally, a major "dichotomy" on transmission technology exists between modulated RF broadband cable transmission (employing CATV components) and baseband tapping on a passive coaxial cable. All the above issues could be classified as referring to the lower protocol functions which refer to the physical layer in the ISO model [8]. As a consequence, the IEEE LAN Standards Committee [9] came up with a draft proposal which includes all the major physical layer options (e.g. bus, CSMA/CD, ring - token polling, bus - token polling) and builds link and network level standards independent of the physical layer. This recognizes the fact that other protocol issues may become more important than the access method and transmission technology, especially in view of the ample bandwidth (high speed) available in LANs.

Immediately related to the protocol function in LANs is the architecture of the network interface unit (NIU), within which most of the the lower layers are implemented. These multiple microprocessor units have to interface a variety of data terminal equipment (DTE), including high speed terminals, multi-vendor hosts, etc. They differ from conventional communication processors, especially on the very high output speed (e.g. 10 Mbps) to be achieved at a cost justifying their existence. For example, on a per port basis an NIU supporting asynchronous terminals (DTEs) should not exceed the cost of a short haul modem.

It was long suspected that throughput analysis ignoring processing times at the NIU and interaction between higher level protocols failed to identify potential bottlenecks, more severe than the limits of the access schemes and the medium speed. As an example, the 97% Ethernet utilization reported in [10] ignores the limitations of the NIU architecture. On the other hand, simulation results on Ethernet type networks with different buffer management and speed mismatch [11], showed that network throughput is very sensitive to those factors, sometimes reducing utilization to less than 50%. In [12] we presented a detailed queueing network model of two microprocessor architectures for interfacing clusters of terminals to a coaxial cable based CSMA/CD network. Using queueing network modeling and similar analytical methods used in performance evaluation of computer systems, we

* This work was supported in part by the USARMY SECOND, CENCOMS under Contract No. DAAK-80-80-K-0579

studied the effect of the NIU on end-to-end character transfer. Our studies showed that the main limitation to LAN performance is due to the interface processing and queueing, while its sensitivity to the cable speed and access protocol is rather limited. In [13] we further demonstrated via approximate analytic tools, that for realistic office automation scenario, the delay bottleneck exists at the DTE/network interface.

These studies, apart from identifying a serious performance limitation, helped us to acquire a valuable analysis/design tool to study NIU architectures. Their variations in the design variables predict weak links and often system parameters which are important to the evaluation of network performance. In this paper, we first summarize the trade-off between the two architectures studied in [12]. It is shown that clearly one class of NIU designs is superior to the other. An improved architecture (within the best class) is then outlined and an analytic tool is presented to model and evaluate the performance of such designs. Finally, sample numerical results are provided, along with sensitivity studies on NIU throughput and delays or functions of critical processing times and protocol parameters.

2. TRADE-OFF STUDIES ON TWO NIU ARCHITECTURES

In most implementations, NIU's consist of two basic units: (1) the terminal interface unit (TIU) and (2) the transceiver unit (TRU). The former serves the user end of the interface, while the latter the network end. In our model we assumed that each NIU includes up to four TIU boards; every TIU can service two to eight FDX ports at speeds up to 9.6 Kbps, as shown in Fig. 1. Each TIU has a dedicated 8 bit NMOS microprocessor with a 1 microsec minimum instruction cycle, like 8085, 6809, or Z80. The TIUs and the TRU are interconnected via a parallel master bus, having a speed equal to the coaxial cable transmission speed (4 or 10 Mbps.)

The two architectures referred to as architecture A and architecture B, differ at the TRU block:

(a) Hardware

Type A TRU has a CPU and memory on board, which controls the DMA transfer to/from the TIUs and an HDLC LSI chip controller, serviced by the CPU.

Type B TRU has no CPU or memory on board. The DMA transfer to/from TIUs are controlled by the TIUs and an arbitration logic. The frame functions are performed by MSI hardware (such as CRC and flag generations, carrier sensing, etc.)

(b) Protocol Level Partition

In NIU Type A, all physical, link, and partly network level functions are performed by the TRU with the TIU performing only packet assembly/disassembly (PAD) routines. In NIU type B, the TRU is merely performing physical level functions (except the backoff timer) and no link functions.

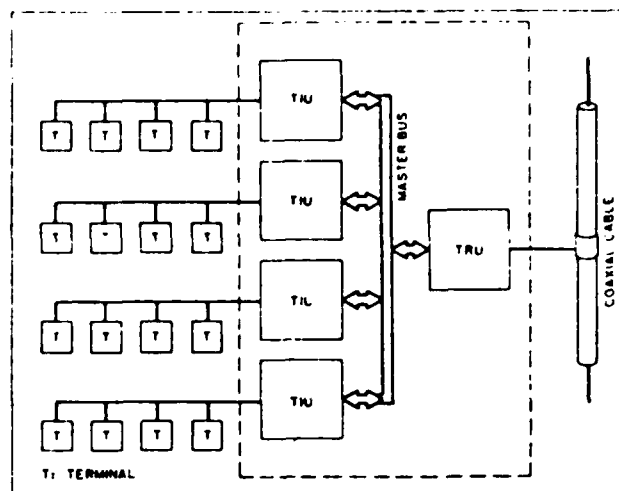


FIGURE 1. GENERAL NIU LAYOUT

2.1 NIU Modeling

In order to analyze the performance of the two architectures, we modeled each CPU board (the TIUs and the TRU for NIU-A) as a central server network with priorities and feedback passes through the central queue. The internal parallel bus of a CPU board, which is exclusively required at every processing pass, is the "commodity" for which requests are queued-up. As an example, consider the TIU queueing model for NIU-A as give in Fig. 2. Packets from the terminals enter the

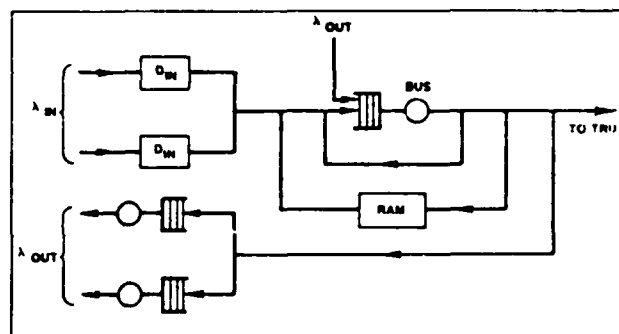


FIGURE 2. THE TIU MODEL, NIU-A

BIU at rate λ_{in} and after experiencing the serialization delay D_{in} hold the parallel bus for the duration of the input interrupt processing. A second pass through the bus models the PAD processing functions and then packets reside in RAM waiting to be fetched by the DMA controller. If the DMA transfer is done on cycle stealing, it will be transparent to the bus functioning apart from slowing down background processing. Otherwise, the DMA transfer is modeled as a third pass with pre-emptive priority over background processing. Similar passes will occur on packets routed from the TRU to one of the TIU serial ports. For a detailed specification of the packet flow in the various stages and the processing times of each pass through the central server the reader is referred to [11].

To integrate two various stages into the NIU model, two approaches are taken depending on the architecture:

For NIU A - the queueing networks representing TIUs and TRU are interconnected via a polling mechanism which simulates the DMA controller, as in Fig. 3-A.

For NIU B - the TRU is modeled as a FCFS queue with service rate equal to the coaxial cable speed. The server will be disabled (switch in off position, Fig. 3-B) if the carrier on the network is sensed busy. The BIUs/TRU coupling via arbitration is also modeled as a FCFS mechanism.

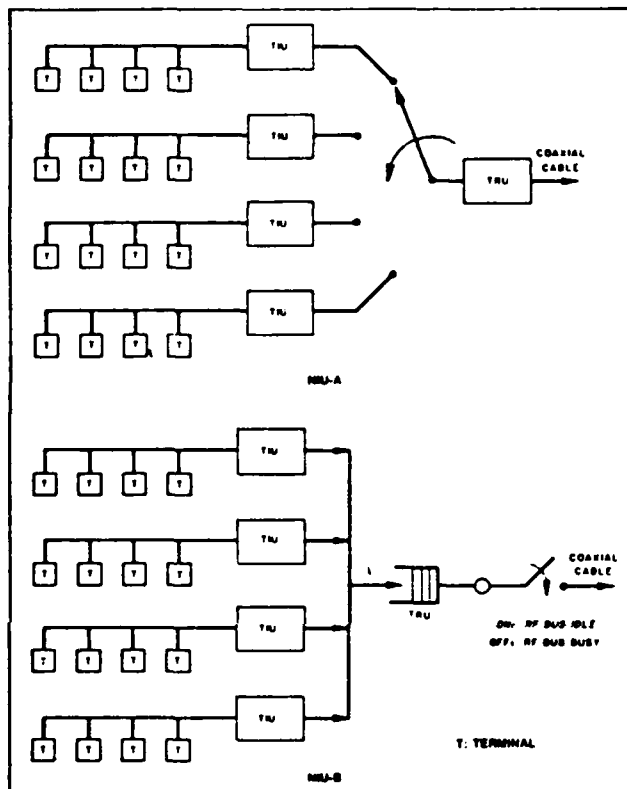


FIGURE 3: MODELS FOR INTEGRATING QUEUEING STAGES

2.2 Results and Conclusions

The models were used to run experiments and perform sensitivity studies. When the two architectures are lightly loaded, they provide similar delay performance. As traffic increases NIU-A saturates at the TRU CPU for link level processing. For NIU-B, saturation occurs due to contention among TIUs to access the TRU pipeline.

For the set of parameters used and for packets including only 1 character of information (10 bits), the analysis showed (Fig. 4) that NIU-B saturated at terminal traffic of 3,000 characters/min whereas NIU-A saturated at 150 characters/min only. Clearly, NIU-A suffers from the separation of link protocol functions between TIU-TRU in series and the single TRU is the bottleneck of the system. NIU-B achieves the highest possible degree of parallel processing among TIUs, with the TRU acting as a mere transceiver. Thus results the 15-fold improvement over NIU-A. The single character per packet requirement was imposed from real echo FDX specifications and rendered throughput calculation quite pessimistic. The results, however, are very useful for comparative purposes.

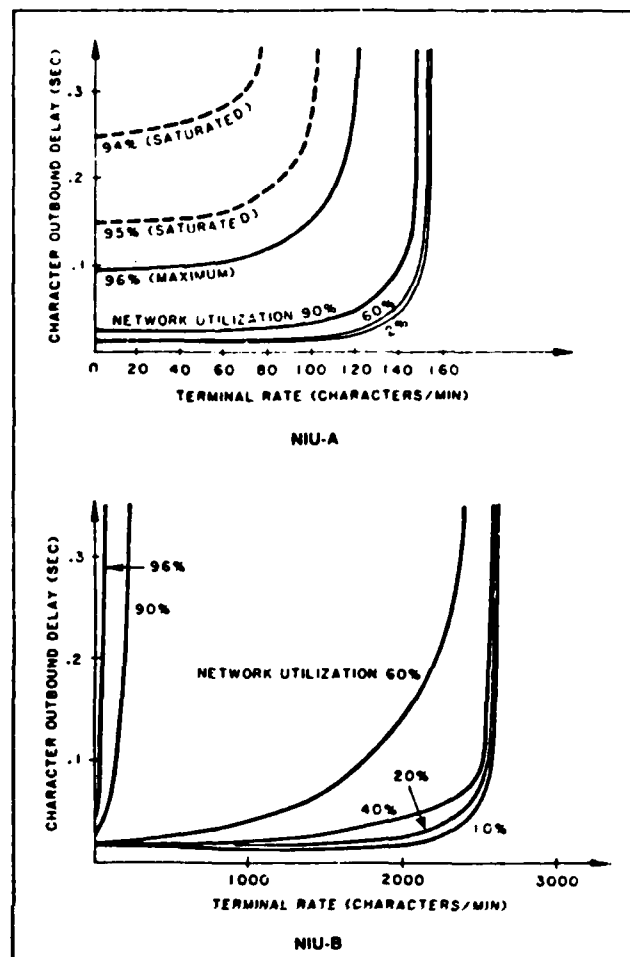


FIGURE 4: COMPARATIVE NIU PERFORMANCE

Another interesting study concerns the sensitivity on NIU throughput and delay from variation of the network (coaxial cable) utilization. As seen in Fig. 4, the NIU performance is almost insensitive to network utilizations from 10 up to 40%. Higher utilizations result into excessive carrier busy and collision conditions and the performance of the system degrades.

In conclusion, it appears that the main limitation on LAN performance is due to interface processing and queueing, while its sensitivity to the cable speed and access protocol is rather limited. Furthermore, performance is greatly affected by the protocol partition among TIU and TRU with bottlenecks arising if the three lower layers (physical, link and network) are allocated in both ends. A pipeline TRU requires customized hardware and lacks the flexibility in adapting to different access protocols through software, but exhibits impressive improvement on the NIU throughput. In the next section, we show how to approach the modeling and analysis of a more sophisticated version of NIU-B, not limited to single character packets and incorporating window flow control to keep memory sizes within reasonable limits.

3. ANALYSIS OF AN IMPROVED NIU ARCHITECTURE

The trade-off studies reported above demonstrated the merit of offloading the TRU from all link level functions, except those performed by hardware at the speed of the network (CRC checking, flag generation and incoming packet address filtering). The models, however, were of limited practical use, since they assumed single information character per packet and did not model flow control techniques at the network level, in order to alleviate buffer overflow and speed mismatching. In what follows, we will highlight an improved NIU architecture which implements variable packet sizes and end-to-end flow control. Then we present a queueing network model and an analytic technique to obtain throughput/delay measures. The analysis is applied to study NIU maximum throughput and its sensitivity to various design parameters.

3.1 Architecture and Simplifying Assumptions

We consider an NIU interfacing data terminals to a broadcast CSMA cable. As before, the NIU consists of a number of parallel TIU CPU boards (up to four) and a TRU with no CPU or memory on board. The DMA transfer between TIUs and TRU is performed via arbitration on a master bus at speed equal to the coaxial cable speed (typically 10 Mbps). Cycle stealing is assumed although burst transfers could be modeled as well.

Each TIU provides a number of serial ports for terminal interfacing. Typically, the number of terminals supported per NIU varies from eight 4.8 Kbps terminals to four 9.6 Kbps and one 56 Kbps high speed DTE. As in Fig. 5, a task queue controls the CPU processing from the I/O interrupt to packet assembly/disassembly, address formation and control

field setting. The completed data packet, with its "TIU header", Fig. 6, will then reside in RAM awaiting for DMA transfer. The TIU processing time will be given approximately by the linear formula.

$$\text{TIU Processing} = p + q \cdot N$$

Where p is a fixed processing time per packet, N is the number of information byte per packet and q the interrupt processing time per bit. Typically, $q = 0.01$ msec/bit and $p = 10 - 70$ msec per input data packet or $p = 5 - 35$ msec for ACKs and output packets. Here, input refers to the DTE-to-network direction and output to the network-to-DTE.

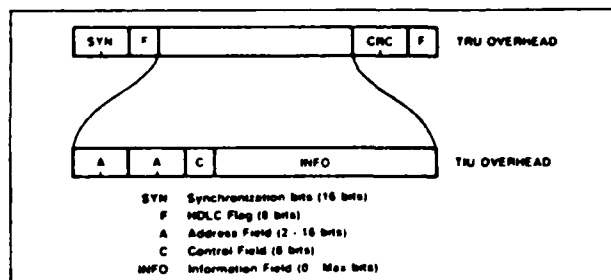


FIGURE 6. PACKET FORMAT (LEVELS 1 and 2)

Terminal data are packetized upon occurrence of either of the following events:

- (1) carriage return of other control character.
- (2) buffer space full (typically 2048 bits)
- (3) no new character arrived from a DTE within a certain window (typically 10 to 50 msec.)

In the case of terminal-to-computer interactive mode, which is the basis of our model, the information field of host response packets is usually ten times longer than a terminal query. In our analysis, we assume that terminal packets are exponentially distributed with mean 10-100 bits (1-10 characters) and response packets are exponential with mean 100-1000 information bits. These numbers are consistent with measurements on 9.6 Kbps terminals performing interacting computing.

The DMA cycle stealing mechanism is assumed to slow down the CPU activity by a factor of 30% during actual data transfer. We ignore DMA controller turn-around and initialization times by assuming that each DMA transfer direction is associated with a separate DMA channel. Hence reinitialization is done during transmission on the opposite direction. It may be desirable to have direct memory access between adjacent TIU's and implement memory shared parallel processing. This option is not incorporated in our present performance analysis.

Date packets (including the TIU 40 bits overhead, Fig. 6), are processed at the TRU, where synchronization bits, flags and CRC bits are generated at the coaxial cable speed (1-10 Mbps). The TRU overhead (48 bits) is illustrated in Fig. 6. The TRU is responsible for sensing the carrier, deferring transmission upon busy condition and retry after a random retransmission period. In our analysis, we assumed that the random interval has mean 5 msec. No collision detection was modeled although it could be simply incorporated in the model.

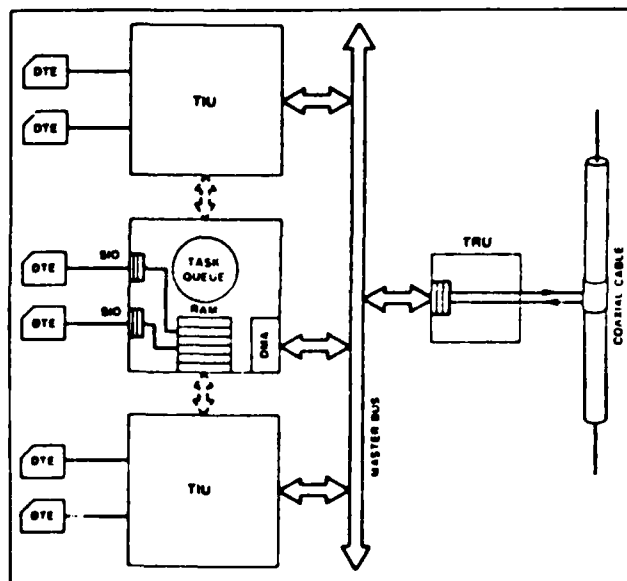


FIGURE 5. HIGH LEVEL DESCRIPTION OF IMPROVED DESIGN

Collided or otherwise destroyed packets would be detected via the acknowledgement mechanism, of the link protocol. In Fig. 7 a logical link is established through the network between interacting DTEs. On that link, reliable exchange is guaranteed by an acknowledgement (ACK) mechanism. Each query packet (or group of packets) is answered by a special ACK packet (or negative ACK in case of error or collision) or an ACK bit piggybacked on information packets flowing in the opposite direction. The window is the maximum number of outstanding packets awaiting ACK's. In this paper, we considered only the case of window size one. In other words, the protocol is a stop-and-wait scheme, whereby no packet is transmitted until the previous one is correctly acknowledged. Negative acknowledgements would result in retransmissions of the same packet. We assumed that the delay to receive a negative ACK, automatically introduces the random deferment needed in random access schemes. Retransmitted packets follow the same route from the TIU RAM to the TRU transfer via DMA. The assumption of a single packet window limits the maximum throughput of the system. It is however a simple technique, especially suitable for interactive applications where the network induced delay is orders of magnitude smaller than user think time and query processing at the host end. An extension to larger window sizes can be easily derived using the same modeling methodology.

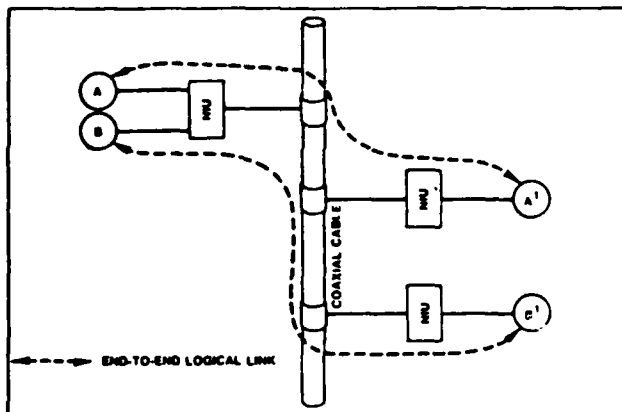


FIGURE 7. END-TO-END DATA FLOW

3.2 Queueing Network Model

The NIU architecture is modeled as a closed queueing network using the methodology outlined in [14]. The finite population of "jobs" travelling within the network, equals the number of terminals connected to the NIU. These jobs will be either packets under processing at the TIU CPU, packets waiting to be transmitted, ACKs travelling back to the NIU, host originated responses to be transmitted to the terminals, or "permits" at the user terminals, whereby a response has been received and the terminal is either in the process of serializing a new packet or simply in a "thinking" state. Each user has a corresponding "job" in one of the above states.

In Fig. 8, the queueing network is illustrated for two TIU's per NIU, each interfacing two terminals. The terminals (small circles to the left), have a pure delay associated with them, representing the time needed to input a new packet, when no other packet from this user waits at the TIU buffer. This delay will be zero in

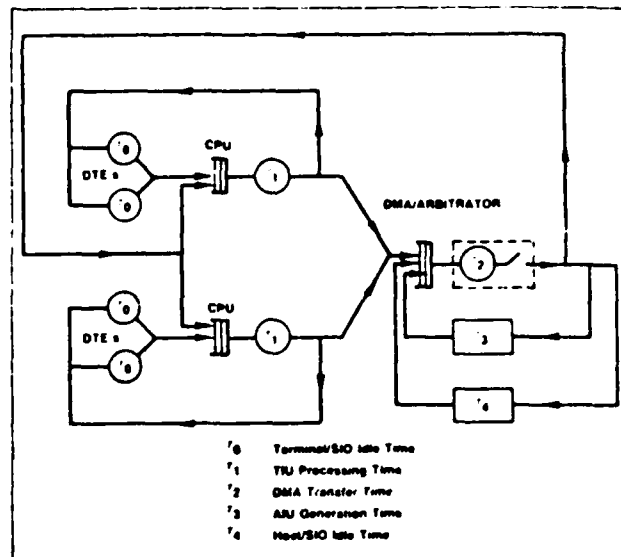


FIGURE 8. NIU QUEUEING MODEL

case of heavy users who do not wait for a response to generate a new query. In some interactive environments, users do not fill the interface buffer until they see the echo from the previous packet. In such case, the "idle" time includes serialization of input and output characters, and user think time. Due to this excessive delay, piggybacking ACK's is impossible and stand-alone ACK's are used. In our analysis we assumed heavy terminal users and mostly piggybacked ACK's in order to study maximum throughput scenario. Obviously, maximum throughput is achieved when there is always at least one packet ready to be transmitted upon receipt of a response and the pure delay at the terminals is zero.

The CPU processes a mix of packets consisting of ACK's, input and output packets. Its service time is slowed down due to the DMA cycle stealing.

The control queue, serves all "ready" packets. Due to the FCFS nature of the DMA arbitration mechanism, we include all TIU RAM's within a single buffer, although this is not physically the case. The same mix of packets are served as in the TIU CPU, with rate equal to the DMA transfer (or the coaxial cable speed). However, due to the CSMA protocol, the server is modeled with an ON/OFF switch, representing the unavailability of service when the carrier is sensed busy. The switch will increase the effective service rate by introducing retrial intervals weighted by the probability of busy condition (see Appendix 2). We assume that the reserving interval is random with a mean of 5 msec.

Upon completion of service at the DMA queue, host responses and ACK's are sent back to the CPU. As explained above, a user data packet will generate either a negative ACK on collision or a positive ACK (stand-alone or piggybacked within host response packets). ACK's will be generated at the destination end after a delay equal approximately to $C \approx 5-35$ msec. Host response packets will be either immediately transmitted (if they already wait in RAM) or need to be generated after some random "idle" time. In Fig. 8, ACK generation and host response times are modeled as pure (non-queueing) delay elements. For maximum throughput studies, we assumed that all positive ACK's are piggybacked on host packet ready to

be transmitted. Thus in heavy traffic scenario, $\tau_0 = \tau_y = 0$, or users and hosts generate packets within the time delay of the flow control window.

An exact analysis of the closed queueing network of Fig. 8, cannot be obtained due to the non-exponential mix of queues 1 and 2. Reasonable results may be obtained via the FCFS approximations in mean value analysis found in [15]. The network is a multichain model, where each TIU board corresponds to one chain with chain population equal to the number of terminals per TIU. In Appendix 1, we provide further details on the analytic technique as implemented for the single chain (one TIU/TRU) simplified version reported in 3.3.

3.3 Numerical Examples

The modeling and analytic technique presented in 3.2 was implemented for a single TIU per NIU configuration as in Fig. 9. Several parallel TIU's per NIU can definitely be studied as well, but since most of the existing LAN interfaces do not support such parallelism for cost considerations, we preferred to limit this study to presently relevant architectures.

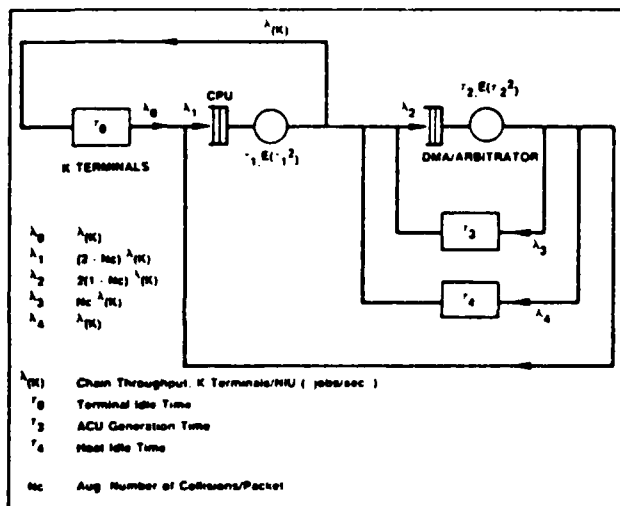


FIGURE 9 NIU QUEUEING MODEL. SINGLE TIU/NIU. K T/NIU

A highlight of the algorithms used in our analysis is given in the two Appendices. Our design parameters were set initially to the values reported in 3.2; more specifically we assumed:

Number of Terminals per NIU: $U = 4$
 Cable Speed: $C = 10$ Mbps
 Input Data Packet: $X_{in} = 100$ bits
 Output Data Packet: $X_{out} = 1000$ bits
 TIU Overhead: 40 bits
 TRU Overhead: 44 bits
 Processing Time per Input Packet: $p = 10$ msec
 Processing Time per ACK/Output Packet = $p/2$
 Interrupt Time per bit = 0.01 msec
 Carrier Resensing Delay: 5 msec
 DMA Cycle Stealing Fraction: 30%

To assess the CSMA parameters, we also need the network-wide cable utilization s and the cable propagation delay a . We assumed that these were not affected by the individual NIU traffic variations

(except changes on the packet size which are assumed to be universal). This is a reasonable assumption for a broadcast LAN with 200 NIU's connected to it, each with different throughput. The s and a parameters were taken as:

$$s = 22\%$$

$$a = 1.35 \text{ microsec.}$$

In Fig. 10, we present the NIU throughput and delay variations, as the terminal and host "idle" times vary. Note that by NIU throughput, it is meant the aggregate amount of information bits/sec flowing in both directions. By NIU delay, we mean the additional time a terminal generated data packet has to wait for transmission due to NIU processing. It is apparent from the figure, that maximum throughput values are obtained with very short idle times (packets always wait in memory to be transmitted as soon as the window opens). The effect of the TIU processing time p , is dominant in heavy traffic conditions. As idle times increase, the throughput is limited by external factors rather than the NIU bandwidth. Similarly delays depend critically upon p .

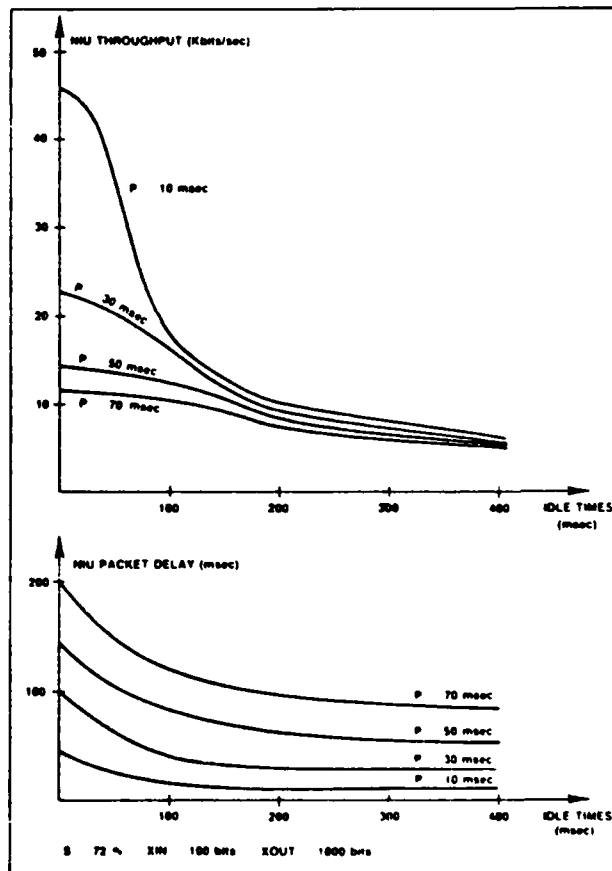


FIGURE 10 NIU THROUGHPUT/DELAY CURVES

Network parameters were shown to have a minimal effect on NIU throughput. Specifically, with cable speeds 1 Mbps and 10 Mbps, maximum throughput values were 46.46 Kbps and 47.06 Kbps respectively, for the same network utilization. In Fig. 11, we plotted the maximum NIU throughput versus network utilization. For utilizations less than 50%, NIU performance was basically insensitive to s . The instability at utilizations higher than 60%, is due to the CSMA saturation.

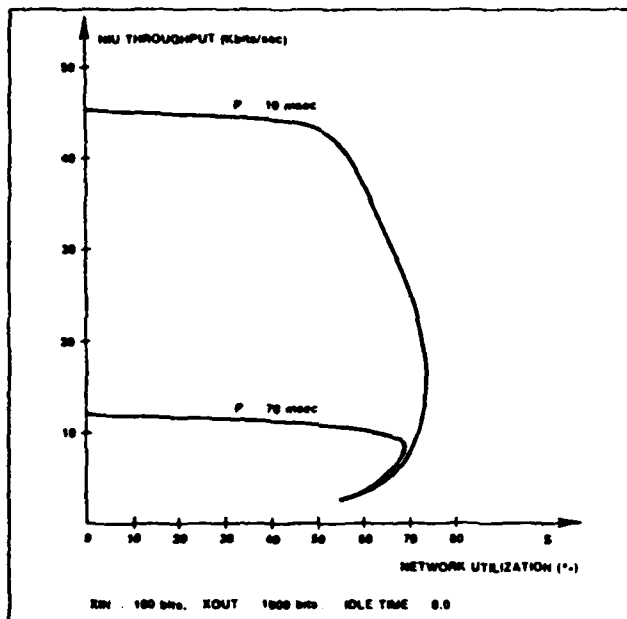


FIGURE 11: NIU MAXIMUM THROUGHPUT WITH s VARYING

Finally, in Fig. 12, we studied the effect of packet sizes on NIU throughput, NIU delay and "round-trip" delay (the time elapsed from closing the window at the terminal NIU, till a response on ACK from the host reopens it). It is important to note that although the throughput increases with the packet size (since more data bits experience the fixed processing time per packet), the curve approaches some limit, whereby the bit interrupt processing and cable transmission delay offset the advantage gained from larger packet sizes. Note that the packet size is mainly limited by other considerations, such as buffer size and packetization latency.

4. SUMMARY OF CONCLUSIONS

A conclusion consistent with experience of LAN users and vendors is that the network parameters do not greatly affect throughput/delay performance, if the cable is utilized below saturation (e.g., 60%). The limiting factor in most cases reside at the interface unit, where bandwidth can be much lower than the network capacity to carry data.

In analyzing the internal NIU behavior, we first identified that protocol partition within the NIU is an important design factor. Specifically, execution of the link protocol at the tranciever unit by a centralized CPU for all terminal interface boards, appear highly inefficient. On the other hand, an architecture whereby the terminal interface units control their

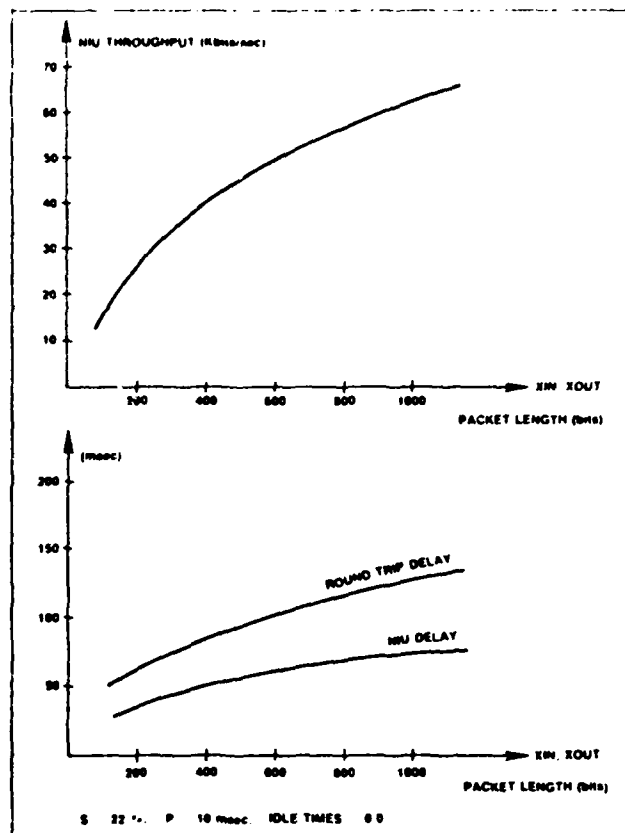


FIGURE 12: NIU MAXIMUM THROUGHPUT WITH IN, XOUT VARYING

associated links, with the tranciever performing basic transmission functions in hardware, achieves a high degree of parallel processing and alleviates the control bottleneck.

The second part of the paper, dealt with modeling and analyzing an improved NIU architecture. The window flow control of the end-to-end link was modeled via closed queueing network methods and approximate analysis were performed. It was found that the NIU throughput was very sensitive to the protocol processing delays, where as cable speed and utilization do not have any obvious effect in normal operating conditions. An important design parameter (apart from the CPU speed), is the packet size with the trade-off being between throughput and buffer size.

The above results were obtained for a simple NIU architecture (one terminal interface unit per NIU) and a stop-and-wait (window 1) protocol. Immediate extensions, include more than one parallel boards per NIU and window sizes greater than 1. Further areas of research, include modeling the sources (terminals) more precisely, assess quantitatively buffer size requirements and improve the approximate analysis on the FCFS non-exponential queues.

ACKNOWLEDGEMENTS

The authors wish to thank Mr. Nicole Di Iorio of the Polytechnic Institute of New York, and Mr. Hubert Chin for their contributions to this study.

APPENDIX 1

Description of the Algorithms

In order to compute the first and second moments of the service times at queue 1 and 2, we define the probabilistic breakdown of the traffic mix and weight means and second moments of various components appropriately. As an example, for queue 1, we have ACKs short (input) and long (output) data packets.

At queue one, the service time τ_1 must also include the DMA cycle stealing effect. Thus, if τ_1 is the service time without DMA overhead and $UTRU$ is the TRU utilization (fraction of time data are transmitted or received).

$$\tau_1 = \frac{\tau_1'}{1 - 0.3UTRU}$$

Here, the cycle stealing is assumed to consume 30% of the CPU power. The utilization of the TRU is a function of the NIU throughput. This, however, is not known a priori and an iterative procedure is used, whereby an initial run is performed assuming $UTRU = 0$ and subsequently new iterations correct τ_1 until convergence is reached. In our numerical examples, the procedure converged within 1-2 iterations.

At queue two the service time τ_2 must include the effect of CSMA carrier busy delays. Let P_s be the probability the carrier is sensed busy as computed in Appendix 2 and D_s the mean interval at which the TRU retries to sense the cable. If this interval is uniformly distributed and τ_2^* is the service time with $P_s = 0$, we have that

$$\tau_2 = \tau_2^* + N_s D_s$$

$$E(\tau_2^2) = E(\tau_2'^2) + \frac{4}{3} D_s^2 N_s^2 (1 + N_s)$$

$$\text{where } N_s = \frac{P_s}{1 - P_s} \quad \text{the average number of retrials}$$

The throughput/delay evaluation for the finite population network of Fig. 9 follows the single chain mean value analysis for closed queueing networks as reported by Reiser in [15]. Due to the non-exponential servers no product form solution exists and the FCFS approximate technique of [15] is used. The input consists of $\tau_0, \tau_1, E(\tau_1^2), \tau_2, E(\tau_2^2), \tau_3, \tau_4$ and the average number of retransmissions of collided packets, N_c .

The algorithm proceeds as follows:

1. set $w = 1, \lambda(0) = \lambda_1 = \lambda_2 = n_1 = n_2 = 0$
2. $t_1 = \tau_1 + \tau_1 (n_1 - \lambda_1 \tau_1) + 0.5 \lambda_1 E(\tau_1^2)$
 $t_2 = \tau_2 + \tau_2 (n_2 - \lambda_2 \tau_2) + 0.5 \lambda_2 E(\tau_2^2)$
3. $t = \tau_0 + N_c \tau_3 + (2 + N_c) t_1 + 2(1 + N_c) t_2$
 $\lambda(w) = w/t$
 $\lambda_1 = (2 + N_c) \lambda(w)$
 $\lambda_2 = 2(1 + N_c) \lambda(w)$
 $n_1 = t_1 \lambda_1$
 $n_2 = t_2 \lambda_2$
4. IF $w = K$ (the number of terminals/NIU), STOP;
 ELSE, $w = w + 1$
 GO TO 2
 END;

APPENDIX 2

CSMA Parameters

We used the pure CSMA throughput formulation as reported in [16]. It assumes that Poisson sources (NIUs) connected to the cable transmit packets of length d sec at a rate S packets/sec. The packet length is computed as the average among input, output and ACK packets. The cable propagation delay was assumed $a = 1.36$ microsec. If the total scheduled rate G (including unsuccessful and collided attempts) is Poisson, the following formula applies:

$$\frac{S}{G} = \frac{e^{-aG}}{G(2a + d) + e^{-aG}} = \text{Pr \{success\}}$$

the probability of collision, P_c is given by:

$$P_c = \frac{1 - e^{-aG}}{G(2a + d) + e^{-aG}}$$

and the probability of busy carrier P_s by

$$P_s = 1 - P_c - \frac{S}{G}$$

References

1. Clark D., et al, "An Introduction to Local Area Networks", Proceedings of the IEEE, Vol. 66, No. 11, November 1978, pp.1497-1517.
2. Metcalfe R., and Boggs D., "Ethernet, Distributed Packet Switching for Local Computer Networks", Communications of the ACM, Vol. 19, No. 7, July 1976, pp.395-404.
3. Christensen G., and Franta W.R., "Design and Analysis of the Access Protocol for Hyperchannel Networks", 3rd U.S.A.-Japan Computer Conference, 1978, pp.86-93.
4. Kirstein P.T., and Wilbur S.R., "University College London Activities with the Cambridge Ring", IFIP WG 6.4 Local Area Network Workshop, Proceedings, Zurich, Switzerland, August 1980, p. 171-196.
5. Tobagi F.A., "Multiaccess Protocols in Packet Communication Systems", IEEE Trans. on Communications, Vol. COM-28, No. 4, April 1980, pp.468-488.
6. Bux W., "Local Area Subnetworks: A Performance Comparison", IFIP WG 6.4 Local Area Network Workshop, Proceedings, Zurich, Switzerland, August 1980, pp. 171-196.
7. Arthurs E., and Stack B.W., "A Theoretical Analysis of Polling and Carrier Sense Collision Detection Systems", Proceedings Seventh Data Communications Symposium, Mexico City, October 1981, pp. 156-163.
8. Open Systems Interconnection, ISO Document, ISO/TC97/SC16 N719
9. IEEE Local Network Standards Committee (Project 802), Draft C, IEEE Computer Society, May 1982.

END

FILMED

9-83

DTIC