AD-A047 206 PURDUE UNIV LAFAYETTE IND SCHOOL OF ELECTRICAL ENGI--ETC F/G 17/2
THE ADAPTIVE ROUTING PROBLEM FOR LARGE STORE-AND-FORWARD COMPUT--ETC(U)
OCT 77 C E HOUSTIS, B J LEON F30602-75-C-0082

UNCLASSIFIED RADC-TR-77-331 NL

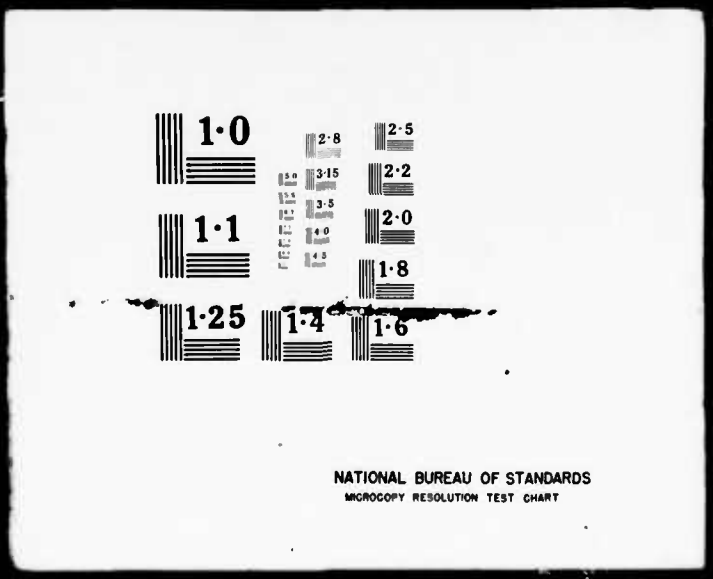
1 OF 2
ADA
047206

UNCLASSIFIED

1 OF 2

ADA

047206



AD A047206

RADC-TR-77-331
Phase Report
October 1977

THE ADAPTIVE ROUTING PROBLEM FOR LARGE STORE-AND-FORWARD COMPUTER-COMMUNICATION NETWORKS

Purdue University

Approved for public release; distribution unlimited.

DDC
RECEIVED
DEC 5 1977
D

AD No.
DDC FILE COPY

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

This report has been reviewed by the RADC Information Office (OI) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public including foreign nationals.

This report has been reviewed and is approved for publication.

APPROVED: Jacob Scherer
JACOB SCHERER
Project Engineer

APPROVED: Joseph J Naresky
JOSEPH J. NARESKY
Chief, Reliability & Compatibility Division

FOR THE COMMANDER: John P. Huss
JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (RBC) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return this copy. Retain or destroy.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: RADC-TR-77-331	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): THE ADAPTIVE ROUTING PROBLEM FOR LARGE STORE-AND-FORWARD COMPUTER-COMMUNICATION NETWORKS.	5. TYPE OF REPORT & PERIOD COVERED: Phase Report.
	6. PERFORMING ORG. REPORT NUMBER: N/A
7. AUTHOR(s): C. E. Houstis, B. J. Leon	8. CONTRACT OR GRANT NUMBER(s): F30602-75-C-0082
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Purdue University, West Lafayette IN 47933	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 95670015 00
11. CONTROLLING OFFICE NAME AND ADDRESS: Rome Air Development Center (RBC), Griffiss AFB NY 13441	12. REPORT DATE: October 1977
	13. NUMBER OF PAGES: 162
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Same	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

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

Same

18. SUPPLEMENTARY NOTES

Project Engineer: Jacob Scherer (RBC)

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

Adaptive
Distributed Routing
Large systems
Network partitioning

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

This research investigates and evaluates routing procedures for large store-and-forward message switched computer communication networks. Existing network topologies are examined. An adaptive routing algorithm is developed dealing with message routing when the full length message is transmitted in its entirety. A thresholding technique is used to work with variations in message length. The algorithms's performance is evaluated by comparison with the performance of the exact mathematical deterministic routing problem.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

292000 1B

In order to apply the routing strategy to large networks, a partitioning approach is used. A partitioning algorithm is developed for communications network topologies. This algorithm uses a generalized labeling technique to find the appropriate partitions of the original network.

The adaptive routing algorithm is modified in order to take into account large network considerations. Then it is applied to the partitioned network in a two-level hierarchical structure. The exact two-level modified adaptive routing algorithm is described.

Some new network topologies found in the literature are examined from the routing problem point of view. All the algorithms are tested by computer simulation. The adaptive routing algorithms have been implemented using two different simulation languages.

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

DDC
RECEIVED
DEC 5 1977
D

PREFACE

This effort was conducted by Purdue University under the sponsorship of the Rome Air Development Center Post-Doctoral Program. This report describes work continued by Purdue University because of its educational and research value. The motivation for the research was a task completed in 1975 for the Defense Communications Agency under the direction of Drs. F.J. Ricci and D. Schutzer.

The RADC Post-Doctoral Program is a cooperative venture between RADC and some Sixty-five universities eligible to participate in the program. Syracuse University (Department of Electrical Engineering), Purdue University (School of Electrical Engineering), Georgia Institute of Technology (School of Electrical Engineering), and State University of New York at Buffalo (Department of Electrical Engineering) act as prime contractor schools with other schools participating via sub-contracts with the prime schools. The U.S. Air Force Academy (Department of Electrical Engineering), Air Force Institute of Technology (Department of Electrical Engineering), and the Naval Post Graduate School (Department of Electrical Engineering) also participate in the program.

The Post-Doctoral Program provides an opportunity for faculty at participating universities to spend up to one year full time on exploratory development and problem-solving efforts with the post-doctorals splitting their time between the customer location and their educational institutions. The program is totally customer-funded with current projects being undertaken

for Rome Air Development Center (RADC), Space and Missile Systems Organization (SAMSO), Aeronautical Systems Division (ASD), Electronics Systems Division (ESD), Air Force Avionics Laboratory (AFAL), Foreign Technology Division (FTD), Air Force Weapons Laboratory (AFWL), Armament Development and Test Center (ADTC), Air Force Communications Service (AFCS), Aerospace Defense Command (ADC), Hq USAF, Defense Communications Agency (DCA), Navy, Army, Aerospace Medical Division (AMD), and Federal Aviation Administration (FAA).

Further information about the RADC Post-Doctoral Program can be obtained from Mr. Jacob Scherer, RADC/RBC, Griffiss AFB, NY, 13441, telephone Autovon 587-2543, commercial (315) 330-2543.

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

In the effort to obtain more potential computer computational capability, more effective ways of utilizing computer systems and resource sharing between geographically dislocated computer systems, significant effort has been spent over the last few years in the analysis, design, and implementation of computer networks.

The first computer networks were established because of the need to distribute computer services to a large number of remote users. In such a network (usually referred to as "centralized") there is one single computer facility connected to a large number of passive terminals by means of a communications network. [ROBE 67] suggested that existing computer systems could be better utilized when interconnected in a network so that their resources could be shared. Among such resources we mention computer power (for load sharing); specialized hardware; specialized software; data banks. This second type of network differs from the first one in that the computer resources are distributed among the nodes, rather than accumulated in the central node. This fact also suggests better survivability of the network. This configuration is hence referred to as a "distributed computer network."

A very important component of the computer network is the communication network which inclues all the hardware and software (modulators, demodulators, channels, concentrators, interface message processors, etc.)

specifically dedicated to the transfer of data from node to node. In some systems, the work of data transmission (message block assembly, storing, routing) is shared between computers and communication networks; in some others, it is completely assigned to the communication network.

Many different communication schemes can be implemented: dial-up lines, dedicated lines, circuit switching, store-and-forward (also called message switching), radio broadcasting, satellite communications, and loop systems. Of course, there are also hybrid systems that include more than one scheme at a time. The selection of the best scheme for a given application is a very difficult problem and depends on many factors (degree of decentralization of the network, traffic characteristics, delay requirements, hardware costs, etc.). A systematic approach to the general problem would be very much technology and application dependent. The selection of the appropriate type of topological structure should also depend on the application.

The communications network considered here is of the store-and-forward type. In such a network a message is stored at each node before being transmitted to the next node in the path; if a link is busy, then a message can be held in storage until the link is free.

Within the network, traffic flow is in terms of messages in their entirety as far as the message length is concerned. The determination of the path that a message takes through the network is controlled by a routing strategy; i.e., a set of rules that guide each message from its source to its destination. The path is not known in advance. Instead, each node determines individually onto which of its output

links to transmit a message, addressed to a specific destination. This determination is based upon network connectivity, network congestion, and link failures. Once the output link is selected, a message is placed on its queue and then transmitted over the link when it is free.

In an operational network, the routing strategy is usually adaptive. It uses current information regarding the network topology and the user demands and is able to readjust the routing logic so as to keep the routing decisions near optimum at every instant of time, despite the fact that the network status is time varying. Adaptive routing procedures protect against network failures and load fluctuations and are essentially for traffic management.

Deterministic routing is used when a network is designed. It implicitly assumes time invariant input rates and a perfectly reliable network configuration. In general, an efficient routing strategy should be able to fully utilize network capability for any load pattern by sending messages on minimum delay or maximum throughput paths and eventually distributing heavy traffic on multiple paths. However, the routing strategy cannot prevent network congestion and a flow control mechanism is needed to control input rates before congestion occurs.

The single most significant measure of performance of these networks is the average time for a message to make its way from its source node through the network to its destination node. We refer to this as the average message delay. This report is essentially devoted to the optimization (minimization) of this performance measure for existing large store-and-forward computer communications networks.

In Chapter 1, which is a reorganization of existing background material, an approximate statistical model of a store-and-forward network is given. This leads to a performance measure for such a network.

In Chapter 2, which presents one of the main contributions of this thesis, an adaptive routing algorithm is presented for handling message traffic, when messages are transmitted in their entirety. Here it is shown that the average delay per message is minimized by routing long messages through direct routes and short messages through indirect routes as much as possible. This is accomplished in the algorithm by the use of interacting bias factors and by updating of node delay information.

In Chapter 3, the deterministic routing problem is summarized and an evaluation of the adaptive routing algorithm of Chapter 2 is presented. In the solution of the deterministic problem, the performance function, Kth mean average delay per message, was selected in such a way as to approximate the function of the adaptive routing algorithm.

In Chapter 4, a network partitioning approach is suggested for large networks in order to be able to apply the adaptive routing algorithm. The main contribution here is a partitioning algorithm which partitions the network topology in subnetworks of a specified size.

In Chapter 5, the adaptive algorithm of Chapter 2 is modified in order to "smooth out" updating of information problems arising in large networks. Then it is rewritten in detail according to a partitioned network.

In Chapter 6, some new network topologies found in the literature are examined as they apply to the adaptive routing problem.

CHAPTER 1

THE MODEL OF A STORE-AND-FORWARD COMPUTER COMMUNICATION NETWORK

Introduction

The message traffic in a store-and-forward network constitutes a stochastic flow problem. In such a problem, both the time between successive arrivals to the system and the demand placed on the link by each of these arrivals are random quantities. As a consequence of that, a queue may form during stochastic flow even if the average flow rate does not exceed the capacity of the link. A great deal of investigation about queueing processes of this type has been done as early as 1907. An outstanding contribution was made by Feller [FELL 66] through his presentation of the birth and death process. A message arrival in the system can be regarded as a birth, and a message departure (delivery) from the system as a death. Queueing network models as they apply to computer systems analysis have been analyzed by many. A summary of such work can be found in Busen's thesis work [BUSE 71].

The particular case of the store-and-forward communications network queueing model was first treated by Kleinrock [KLEI 64]. His model has been generally accepted. Until today, it is used in operational networks for the determination of a performance measure. Several have tested it by simulation [FULT 72], [GERL 73A], [KAMO 76] and all agree that it is quite representative.

In this chapter Kleinrock's model is formulated according to his analysis. In section 1.1 the exact mathematical model is presented and the problems associated with it are discussed. Appropriate assumptions are introduced in order to derive an approximate statistical model. In section 1.2 a performance criterion is defined and analyzed, and in section 1.3 a new performance criterion is introduced.

1.1 The Model

The mathematical model of a store-and-forward communication network can be presented in terms of state variables [KLEI 64]. Such a set of variables must satisfy two conditions:

(a) It must include (explicitly or implicitly) those quantities which are of interest; example, the message delay.

(b) The set must be complete, *or closed*, such that knowledge of the state variables at time t and knowledge of all message arrivals from sources external to the net in the closed interval (t,t') are sufficient to specify uniquely the state variables at time $t' \geq t$. This second condition describes the markovian property.

We consider a communication network with N nodes and M one-way channels. The assumptions made in order to arrive at the model are

1. External Poisson arrival statistics
2. Exponentially distributed message lengths
3. Infinite storage capacity at each node
4. Single destination for each message
5. Full reception of message before retransmission
6. Noiseless links and perfectly reliable nodes and links
7. Stationarity and independence of the stochastic processes.

It is clear that the state of the net at any instant of time must include detailed information as to the number of messages on each queue, the length of each message, and the time required to complete the transmission in progress on each link. We assume that each message is labeled with an origin, a destination, and a priority. Accordingly, we define

C_i = capacity of the ith link, bits/sec

n_i = number of messages waiting for (or being transmitted) on the ith link

γ_i = average arrival rate of messages from external sources to the ith link, mes/sec

$v_{in}(x_{in}, y_{in}, z_{in})$ = message length, in bits, of the nth message waiting for (or being transmitted on) the ith link, whose origin, destination, and priority are x_{in}, y_{in}, and z_{in} respectively; for simplicity of notation, let symbol v_{in} denote this length.

V_{n_i} = set of numbers $(v_{i1}, v_{i2}, \ldots, v_{in_i})$

r_i = time remaining to complete transmission in progress on the ith link.

R = set of numbers $(r_1, r_2, \ldots, r_M)$

$R + dt$ = set of numbers $(r_1+dt, r_2+dt, \ldots, r_M+dt)$

where $i = 1,2,\ldots,M$ and $n = 1,2,\ldots,n_i$

The state of the network at any time t may be completely described by the set of variables,

$$S = (n_1, n_2, \ldots, n_M, V_{n_1}, V_{n_2}, \ldots, V_{n_M}, r_1, r_2, \ldots, r_M)$$

All these quantities vary with time. This state description is of unbounded dimensionality, since the variables n_i, are unbounded. Furthermore, these variables are necessary in order that the state description be complete.

For each state S and each time t a probability density function $p_t(S)$ is associated with the probability that the net will be found in state S at time t. In general, it would be desirable to find an explicit solution for the function $p_t(S)$. To date this problem remains unsolved. A portion of the analysis will be carried out in order to indicate the source of the difficulty. Let us set up the equations under the conditions

$$n_i > 1$$

$$0 < r_i < \frac{v_{i1}}{C_i}$$

The first condition, $n_i > 1$, is included for convenience. The end points zero and $\frac{v_{i1}}{C_i}$ are excluded from the allowed range of r_i in order to eliminate temporarily from discussion any consideration of internal message arrivals.[1]

For this case we write

$$p_{t+dt}(n_1,n_2,\ldots,n_M,V_{n_1},V_{n_2},\ldots,V_{n_M},R)$$

$$= p_t(n_1,n_2,\ldots,n_M,V_{n_1},V_{n_2},\ldots,V_{n_M},R+dt)\left(1-\sum_{i=1}^{M}\gamma_i dt\right)$$

$$+\sum_{i=1}^{M}\gamma_i' dt\,\mu e^{-\mu v_{in_i}}\,p_t(n_1,n_2,\ldots,n_i-1,\ldots,n_M,$$

$$V_{n_1},V_{n_2},V_{n_i-1},\ldots,V_{n_M},R+dt)$$

[1] An internal message arrival occurs when a message completes its transmission between two nodes internal to the net (as opposed to an external message arrival which occurs when a message arrives at its origin from a source to the net).

where γ_i' represents that portion of γ_i which has the x_{in_i}, y_{in_i}, and z_{in_i} which correspond to v_{in_i}, $1/\mu$ is the mean value of the message length exponential distribution. According to the Chapman Kolmogorov equation, this leads to the following partial differential difference equation:

$$\frac{\partial p_t}{\partial t}(n_1,n_2,\ldots,n_M,V_{n_1},V_{n_2},\ldots,V_{n_M},R)$$

$$- \sum_{i=1}^{M} \frac{\partial p_t}{\partial r_i}(n_1,n_2,\ldots,n_M,V_{n_1},V_{n_2},\ldots,V_{n_M},R)$$

$$= \sum_{i=1}^{M} \gamma_i'\mu e^{-\mu v_{in_i}} p_t(n_1,n_2,\ldots,n_i-1,\ldots,n_M,V_{n_1},V_{n_2},\ldots,V_{n_i}-i,\ldots,V_{n_M},R)$$

$$- p_t(n_1,n_2,\ldots,n_M,V_{n_1},V_{n_2},\ldots,V_{n_M},R) \sum_{i=1}^{M} \gamma_i$$

where $n_i > 1$ and $0 < r_i < \frac{v_{ii}}{C_i}$. The equations involving r_i at its end points force one to consider internal message arrivals and thus become considerably more complex. This is because the rules of the routing procedure must be included in order to determine which transitions occur. The task of solving this set of partial-differential difference equations is enormous and no solution has yet been found.

The complexity of the state description is due in part to the constraint that each message, when it enters the net, has a permanent length assigned to it. The message maintains this same length as it travels through the net. This makes it necessary for the variables V_{n_i} to be included in the state description. The assignment then of a permanent message length not only increases the dimensionality of the

state description, but also complicates the stochastic behavior of the net by introducing a dependence among some of the random variables which describe the behavior of the net.

In particular, if we consider two successive messages arriving at node i from some other node internal to the net, then the interarrival time between these messages is not independent of the message length of the second of the two messages.[2]

More specifically a simple two node tandem net is considered in Fig. 1.1 and the joint probability density function $p(\nu_n, a_{2n})$ is derived where

ν_n = message length for nth message, bits

a_{in} = time between arrivals of (n-1)st and nth messages at node i

Let, for convenience, C_i = 1 bit/sec. By assumption the message length is a random variable from an exponential distribution, the arrivals probability distribution is Poisson, then the interarrival time distribution is exponential.

Then let

$$p(a_{in}) = \gamma e^{-\gamma a_{in}}$$

$$p(\nu_n) = \mu e^{-\mu \nu_n}$$

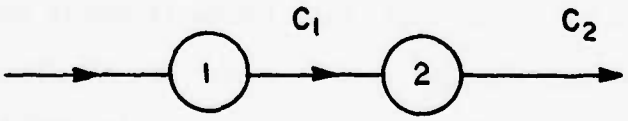


Figure 1.1 A two node tandem network.

[2]This independence exists by assumption for messages which arrive from an external source.

Theorem 1.1 (due to Burke)

The steady-state output of a queue with N channels in parallel with Poisson arrival statistics, and with lengths chosen independently from an exponential distribution is itself Poisson distributed.

For the case considered above, Burke's theorem holds and since the interdeparture times for messages leaving node 1 are, by definition, identical to the interarrivals times for messages entering node 2, we see that

$$p(a_{2n}) = \gamma e^{-a_{2n}}$$

It can be shown [KLEI 64] that

$$p(\nu_n, a_{2n}) \neq p(\nu_n)p(a_{2n})$$

which illustrates the lack of independence between ν_n and a_{2n}.

At this point the famous independence assumption was introduced by [KLEI 64] which says that

> Each time a message is received at a node within the network, a new length ν is chosen for this message from the following probability density function: $p(\nu) = \mu e^{-\mu\nu}$.

It is clear that the assumption does not correspond to the actual situation in any practical communication net. But simulation results have shown that the independence assumption has resulted in rather insignificant changes in the average message delay as long as the network is highly connected, i.e., there are multiple paths emanating from every node. Then the behavior of the message delay for a single node carrying internal traffic is very much the same as that for a node supplied exclusively with external traffic.

Now if we consider only fixed routing,[3] we recognize that the internal traffic flowing into each channel is statistically equivalent to the external traffic entering the net; i.e., the interarrival times and message lengths are independent and are chosen from exponential distributions. Thus, each link satisfies the conditions of the single exponential channel. This fact coupled with Burke's [BURK 56] theorem, allows us to consider each link (or node) separately in the mathematical analysis. Then the original mathematical models have been reduced to a network of queues of the Jackson type [JACK 57]. Baskett and Muntz [BASK 72] showed that each queue behaves as an M|M|1 queue.

1.2 The Average Delay T

The computer communication store-and-forward network is now decomposed into a network of independent queues. The link queues are of the M|M|1 type. Traffic entering the network then from external sources forms a Poisson process with a mean γ_{jk} (mes/sec) for those messages originating at node j and destined for node k. The total external traffic entering the network is defined as

$$\gamma = \sum_{j=1}^{N} \sum_{k=1}^{N} \gamma_{ij} \qquad (1.1)$$

where N is the number of nodes in the network. Note that the above total external traffic definition represents a decomposition of the network on the basis of origin-destination pairs.

---

[3] By fixed routing procedure we mean that given a message's origin and destination, there exists a unique path through the net which this message must follow. If more than one path is allowed, then we speak of an alternate routing.

Since each link in the network is considered as a separate queue, we define as λ_i the average number of messages per second which travel over the ith link. Then the total traffic within the network is redefined as

$$\lambda = \sum_{i=1}^{M} \lambda_i \qquad (1.2)$$

where M is the number of links in the network. This definition represents a decomposition down to the single link level.

A very important performance measure for store-and-forward communications nets is the average message delay [KLEI 76]

$$T = E[\text{message delay}].$$

Define first the quantity

$$Z_{jk} = E[\text{message delay for a message whose origin is j and destination k}].$$

Then the last two quantities are related by

$$T = \sum_{j=1}^{N} \sum_{k=1}^{N} \frac{\gamma_{jk}}{\gamma} Z_{jk} \qquad (1.3)$$

since the fraction γ_{jk}/γ of the total entering message traffic will suffer the delay Z_{jk} on the average.

Let us now denote the path taken by messages that originate at node j and are destined for node k (the "j-k traffic") by π_{jk}. We say that the ith link (of capacity C_i) is included in the path π_{jk} if that link is traversed by messages following this path and symbolize it by $i{:}C_i \in \pi_{jk}$. It is clear, therefore, that the average message flow rate on link i, λ_i, must be equal to the sum of the average message flow rates of all paths that include this link, that is

$$\lambda_i = \sum_j \sum_k \gamma_{jk} \qquad (1.4)$$

$$j,k:C_i \in \pi_{jk}$$

Let us now define the delay of the individual link i as

T_l = E[time spent by messages waiting for and using the lth link]

Thus T_l is just the average time "in system" where the system is defined as the ith link plus the queues of messages in front of that link. We may also write

$$Z_{jk} = \sum_{i:C_l \in \pi_{jk}} T_i \qquad (1.5)$$

From equations (1.3), (1.5) we therefore have

$$T = \sum_{j=1}^{N} \sum_{k=1}^{N} \frac{\gamma_{jk}}{\gamma} \sum_{i:C_i \in \pi_{jk}} T_i$$

After interchanging the summations order we obtain

$$T = \sum_{i=1}^{M} \frac{T_i}{\gamma} \sum_j \sum_k \gamma_{jk} \qquad (1.6)$$

$$j,k:C_l \in \pi_{jk}$$

Using equation (1.4) we have

$$T = \sum_{l=1}^{M} \frac{\lambda_l}{\gamma} T_l \qquad (1.7)$$

We have thus decomposed the average message delay into its single link delays T_i.

We now have to solve for the message average delay on a link. The ith link is now representable as an M|M|1 system with Poisson arrivals at a rate λ_i and exponential service times (message transmission times)

of mean $1/\mu C_i$ sec. The solution for T_1 is given then by

$$T_1 = \frac{1}{\mu C_i - \lambda_i} \tag{1.8}$$

and so equation (1.7) becomes

$$T = \sum_{i=1}^{M} \frac{\lambda_i}{\gamma} \left[\frac{1}{\mu C_i - \lambda_i} \right] \tag{1.9}$$

Letting $\lambda_i/\mu \triangleq f_i$ average bit rate on link i (bits/sec), we obtain

$$T = \frac{1}{\gamma} \sum_{i=1}^{M} \frac{f_i}{C_i - f_i} \tag{1.10}$$

This is the main queuing analysis result for the store-and-forward communications net.

In high-speed networks, it may be important to include the propagation delay P_i which is the time required for the energy representing a single bit to travel down the length of the ith link. It may also be of importance to include the factor k defined as nodal processing time. In the work to follow, we assume $P_i = 0$ and $k = 0$.

1.3 Mean-Kth-Power Delay $T^{(K)}$

It was observed by [MEIS 72] that in minimizing the average delay T as from equation (1.7) a wide variation was allowed among the link delays T_i and, therefore, among message delays between different source-destination pairs. In order to take into account such variation, an alternate performance measure was proposed:

$$T^{(K)} = \left[\sum_{i=1}^{M} \frac{\lambda_i}{\gamma} (T_i)^K \right]^{1/K} \tag{1.11}$$

where $K \geq 1$ is a finite number. If K is sufficiently large, the variations among message delays are considerably reduced.

CHAPTER 2

ADAPTIVE STRATEGY TECHNIQUE

Introduction

In an adaptive strategy the selection of the output channel at node K depends on the estimate of the delays that a message would suffer when transmitted over different output channels. Adaptive algorithms are mostly suited to operational networks because of their ability to adjust to different traffic loads and network failures. In this chapter an adaptive routing algorithm will be developed applicable to store-and-forward computer-communications networks.

Extensive work on adaptive routing techniques has been done by Fultz [FULT 72]. He considered store-and-forward packetized networks, like the ARPA net. The algorithm to follow is based on his work. A quadratic routing scheme is presented by [AGNE 75] where a quadratic bias factor encourages a lot of alternate routing. Recent work was published by Kamoon [KAMO 76] where he treats the network design problem according to adaptive routing consideration.

In section 2.1 a definition of the adaptive routing problem is stated and a classification of the adaptive routing techniques is presented.

In section 2.2 an adaptive routing algorithm is developed. The algorithm is applied to an eight node network.

In section 2.3 the simulation language used for the computer simulation is discussed. The simulation system characteristics along with the drawn conclusions are presented.

2.1 The Adaptive Routing Problem

A general definition of the adaptive routing problem was first stated by Kleinwork [KLEIN 64] as follows:

minimize T (the average message delay)

over the variables { link capacity assignments; message priority discipline; routing doctrine; topology }

subject to a suitable performance criterion and

external traffic requirement.

It is desirable to classify network adaptive routing techniques in order to gain insight into their structure complexity and performance. Such classification has been done by Prosser [PROS 62], Kleinrock [KLEI 64], Fultz [FULT 72], and Gerla [GERL 73], and others. A summary of all these classifications is presented by McQuillan [MCQU 74].

The two major categories are found under the title of adaptive deterministic policies and adaptive stochastic policies. Fultz has presented them in a concise and accurate manner as follows:

Adaptive routing algorithm classification

1. Deterministic techniques

* Flooding < all / selective
* Fixed
* Network routing control center (NRCC)
* ideal observer < present / future scheduling problem.

2. Stochastic techniques

* Distributed < asynchronous update / periodic update
* Isolated < shortest queue + bias / local delay estimate
* Random

2.1.1 Deterministic Techniques

The four basic deterministic techniques are:

Flooding. Each node receiving or originating a message transmits a copy of it over "all" outgoing links, or over a set of "selective" outgoing links; this transmission occurs only after the node has checked to see that it has not previously transmitted the message, or that it is not the destination of the message. This technique has been discussed by Boehm and Madiey [BOEH 66]. Their conclusion is that the inefficiency of this technique is tolerable if one has only a few messages to deliver. However, a large volume of communications traffic necessitates more efficient techniques. Another drawback to this technique

is that each node requires a mechanism to recognize previously transmitted messages.

Fixed routing. Fixed routing algorithms specify a unique path $(N_S,\ldots,N_D)$ followed by a message which depends only upon the source-destination node pair (N_S,N_D). To accomplish this, each node has a routing table. Depending on the message destination, each entry in the routing table contains the next unique node in the message's path. Kleinrock [KLEI 64], [KLEI 76] and Prosser [PROS 62] have examined several of these techniques. Fixed routing techniques require completely reliable nodes and links. However, they do allow for high efficient high volume traffic flow and are very stable. (i.e., they are relatively unaffected by small changes in the network topology.)

Network routing control center (NRCC). With this technique, one of the network nodes is designated as the NRCC. This center collects performance information about the network operation and computes routing tables for each node in the network. Computation of the routes by NCRR is done on a global basis and this insures loop-free paths between all source-destination node pairs. Thus, a fixed routing procedure is maintained between NRCC updates.

There are a number of drawbacks to this technique. By the time the nodes begin using the new routing tables, the performance information that was used in the computation of the routing tables may be out of date in relation to the current state of the network. In addition, transmission costs and vulnerability become significant considerations.

Ideal Observer routing. This technique is essentially a scheduling problem. Each time a new message enters a node its route is computed to minimize its travel time to its destination node, based upon the complete present information about the messages already in the network and their known routes. If the ideal observer has information about the occurrence of future events, then this information could also be utilized in the computation of the route. This technique is obviously impractical for an operational network but theoretically provides the minimum average message delay.

2.1.2 Stochastic Techniques

The three basic stochastic techniques are:

Random routing. Random routing procedures are those decision rules in which the choice as to the next node to visit is made according to some probability distribution over the set of neighbor nodes. The set of neighbor nodes utilized in the decision rule can be "all" of the connected nodes or can be based "selectively" over that set of nodes which are in the general direction of the message destination. Kleinrock)[KLEI 64] and Prosser [PROS 62] have investigated numerous random routing techniques and have shown that they are highly inefficient in terms of message delay, but are extremely stable.

Isolated and distributed techniques. All of the isolated and distributed routing algorithms operate in basically the same manner. A delay table is formed at every node whose entries are the estimated delays to go from the node under consideration to some destination node.

The manner in which these estimates are formed and updated depends upon the specific structure at the routing algorithm.

In the shortest queue + zero bias a message route is selected by placing it in the shortest output channel queue. This is essentially Baran's Hot Potato routing concept [BARA 64]. Since the route selected is independent of the message destination, the delay table requires information only on the output link queues of the sending node.

In the local delay estimate algorithm, a message's route is selected via the delay table estimates. These estimates are formed from information such as the time the message has spent in the network traveling from its source node to the current node and backwards learning delay information from the reverse outgoing line corresponding to the forward line of the full-dublex pair which the message is about to enter. This backwards learning has been investigated by Baran [BARA 64], Boehm [BOEH 64].

In the distributed routing techniques classification, all routing algorithms utilize the same basic techniques to compute and update the delay table estimates, but the instants at which these tables are updated and the route selection procedures depend upon the particular structure of the algorithm.

Of all the stochastic techniques, the distributed routing is the most efficient for handling line and node failures. Once a failure is determined, the proper delay table entries in every node can be forced to remain excessively large as long as the failure persists.

We felt that the distributed stochastic routing techniques have the best potential performance to offer to operational store-and-forward computer-communication networks.

The adaptive algorithm that follows here is of the distributed stochastic type based on Fultz's work for the Arpanet.

2.2 Adaptive Routing Algorithm

There are two ways to implement a store-and-forward network. In the one, each message is broken up into submessages ("packets") of equal length which are routed as individual messages and then reassembled at the destination. In the other, each message is routed in its entirety with consequent variation in message length. The algorithm is going to deal with the message in its entirety.

In the algorithm , each node in the network makes its decisions on the basis of information sent by nodes to which the deciding node is directly connected. Since it in turn sends information to all such nodes, the information can "percolate" through the system. Each piece of control data is sent only from one node to the next, but in order to ensure rapid flow of information, the data must be sent fairly frequently, and thus the control data constitutes the bulk of the non-message data flow.

The optimization problem is essentially a trade-off of control data flow against improvement in message delay. This is because the message delay itself is a function of the data flow since the data are treated as messages and therefore increase the message load.

2.2.1 The Routing Algorithm

Each node maintains a delay table as shown in Fig. 2.1. The table has as many rows as there are destination nodes and as many columns as there are output links from the node. An entry $T_j(D,L_m)$ is the estimated delay to go from node j (the "deciding" node) to a destination node D using link L_m as the next step on the path to D. The routing table is formed from the delay table by choosing for each row, the link which has the minimum delay for that row (destination). If the routing table entry for the jth node is $R_j(D,L_m)$ then $R_j(D,L'_m) = i$ where L'_m corresponds to the link with $\min_{L_m} T_j(D,L_m)$. The heart of the algorithm is the technique by which the delay table entries are changed. A given node receives delay estimates from each node to which it is directly connected. The given node then forms delay estimates of its own, utilizing this information, and transmits these estimates in its turn. In this way, the estimates "percolate" through the system. Suppose a given node J has made a decision to inform its "neighbor nodes" (i.e., those nodes with a direct link to node J) of its estimated minimum current delay to all other nodes in the network. Switch J forms a Delay Vector V_J which has one entry T(i) for each other node in the network, where

$$T(i) = \min_{L_m} T_j(i,L_m) \qquad (2.1)$$

minimized over the set L_m of output links from node J. This Delay Vector is then transmitted to the neighbor nodes. When a switch K receives a delay vector from node J, node K adds the queue delay (in bits) on the node J outbound link L_g to each entry in the Delay Vector (plus an

SAMPLE CONTROL TABLES FOR NODE 4

Delay table

D	L_1	L_2	L_3	L_4
1	10	12	15	27
2	14	16	7	33
3	8	19	13	29
5	26	23	18	24
6	21	25	17	15
7	15	18	26	23
8	32	29	24	13

Minimized delay vector

D	Delay	(L_m)
1	10	(L_1)
2	7	(L_3)
3	8	(L_1)
5	18	(L_3)
6	17	(L_3)
7	15	(L_1)
8	13	(L_4)

Routing table

D	L_1	L_2	L_3	L_4
1	1	0	0	0
2	0	0	1	0
3	1	0	0	0
5	0	0	1	0
6	0	0	1	0
7	1	0	0	0
8	0	0	0	1

The notation refers to Fig. 2.3 (p. 33). D is the destination node.

Figure 2.1 Delay table and delay vector stored in a node.

adjustment factor DELTA explained below) and replaces the L_g column in its delay table with the new values. The delay table equation then is

$$T_k(D,L_g) = \text{queue } (K,L_g) + T(D) + \text{DELTA} \qquad (2.2)$$

where queue (K,L_g) is the outgoing queue of node K on link L_g. $T(D)$ is the entry of the delay vector for destination D. The adjustment factor depends on the type of network. In a packetized system, DELTA is usually a constant related to the average transmission time of a packet. If the tables are updated rapidly, then the delay table estimate in node K will have a term approximately equal to $(N^*(K,D,L_g))x(\text{DELTA})$ where $N^*(K,D,L_g)$ is the number of links encountered on the path between node K and node D through link L_g; i.e., if DELTA is close to the average message length and if we have frequent updating and if the queues are short, then the term $N^*(K,D,L_g)x\text{DELTA}$ approximates the delay caused by the transmission of the message on the intervening links. For variable length message systems this is even more significant. Since the message is accumulated at each node, there should be a "penalty term" for traversing a circuitous path. Such a path would cause delay which is not represented by the existing queue lengths. In the asynchronous update for very long messages explained below, this term is calculated directly using the actual message length. However, this is not the only consideration in choosing DELTA. The routing scheme is suboptimal and constantly being updated. Thus, there is a real possibility of a message traversing a "loop" and passing through the node which initiated it, and this obviously will hurt the average delay. There is no easy way to prevent this completely and as long as the

average delay is reduced, occasional looping is tolerable. The chances of looping are obviously increased if the message follows a circuitous path, and the value of DELTA can be adjusted to introduce an additional penalty on circuitous routes in order to reduce looping. The exact value of DELTA which gives the minimum average delay must be determined by simulation.

For a non-packetized system, the algorithm does not account for the wide variation in message lengths. In the normal operation, the constant DELTA should be at least as large as the average message length. This will force very short messages to take "more direct" but actually slower routes, but this should not be a serious problem. For extremely long messages, the system should have a much stronger bias towards the "most direct" route, and this is built into the algorithm in the form of a special routing technique for long messages. When an incoming message length exceeds a threshhold, the routing is not done via routing tables, but is done dynamically via the following formula:

$$L = \min_{L_m} \{\text{queue}\ (i,L_m) + N^*(I,D,L_m)x\ell - N^*(I,D,L_m)x\ell_{avg} + T(D)\} \tag{2.3}$$

where ℓ = actual message length; ℓ_{avg} = average message length. In this formula, the instantaneous queue is used, but the bias term multiplies the number of links in the shortest path times the length of the message and subtracts the estimated delay caused by the fixed bias term in the average message routine. This must be done explicitly since the delay estimates which are received from other nodes do not take account of

this very long message. The inclusion of this term gives a very strong bias towards the shortest route; but if the queues on this route are long enough, another route can be taken if it appears to have less delay.

A simple algorithm can be added to increase survivability by performing automatic failure detection. By keeping track of the number of links traversed by an incoming message, the receiving node could determine network failures on the basis of traffic data, as described in [TORB74]. When a failed link or node is detected, either by the node or by central control, the delay entry for that path can be made so large that the node algorithm will not attempt to route messages over the failed equipment.

The update of the delay tables information is done as follows. Periodically, new estimates received from a neighbor node replace the older estimates. There are two approaches to update frequency. One sends the update data at regular intervals (synchronous update); the other sends the data only "as required" (asynchronous update). In the case of a network (such as a military one), a prime consideration is reliability and the speedy detection of faults which are enhanced by synchronous update. Thus, the algorithm includes a synchronous update.

The assignment of a very long message to a queue will result in a large enough delay to justify extra information transfer. Thus, an asynchronous update is included in the algorithm and in this update is triggered by the use of the long-message routing algorithm in the node.

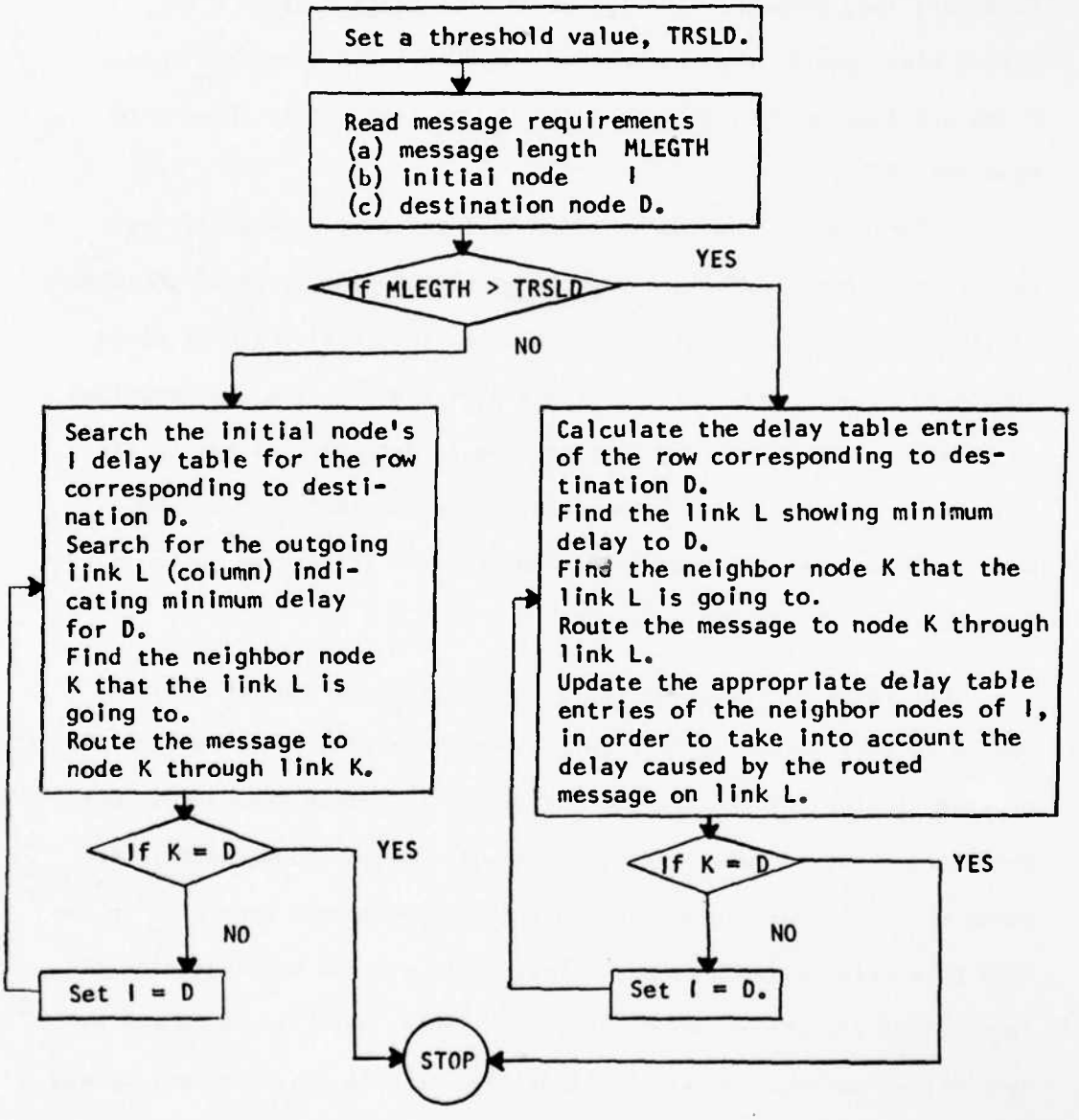


Figure 2.2. Flowchart of a message routing the network.

One can also include a "spill-over" asynchronous update when an asynchronous delay vector is received at a node and the delay vector causes an over-threshhold change in a delay table entry. The effect on average delay of various message-length threshholds and message-length traffic mixes must be evaluated by simulation. A flowchart of a message routing through the net is shown in Figure 2.2.

2.3 The Simulation Language

Of the several languages available, the language ASPOL was chosen for the simulation. Originally developed to simulate multi-processing systems in computers, [ASPOL 72] has several features which make it convenient for the simulation of communication networks. ASPOL is a general-purpose, process-oriented language based on process communication and control structures used in computer operating systems. The language is based on the language SOL developed by D. E. Knuth et. al. [KNUT 68], and it is ALGOL-like. Among the major features of ASPOL are: Language facilities to concisely describe systems of interacting processes; the ability to define and manipulate sets of entities, such as links; and macro facilities, which can extend the language to specialized problems such as communication systems. ASPOL is defined in terms of "processes," "facilities," and "events." A process is a particular example or instance of a set of activities. In the simulation, each message is a process, and the set is all messages being transmitted at a given time.

2.3.1 Description of the Simulation

An eight node store-and-forward communication system was simulated using ASPOL on the CDC 6500 computer. The basic idea of the simulation is:

A global program simulates the routing decisions of the individual nodes, but makes decisions only on the basis of information local to each node. The global program is called SIM and has specialized subroutines to carry out the actual computations. The data for the entire network is stored in global arrays and SIM makes local decisions on the basis of information selected from the arrays. The data transfer which updates the arrays is simulated by a special process called UPDATE. The system information which is the object of the simulation is collected from the operation of UPDATE. This process is initiated at fixed intervals whose length can be varied, giving the synchronous update capability and also at variable intervals determined by SIM which simulates the asynchronous update. The messages are simulated as particular examples of a process MESSAGE, where each example has a length, arrival time, and origin and destination nodes drawn from appropriate probability distributions. Each message is routed by routing tables local to a given node (recalculated for long messages) and assigned to the appropriate outgoing link, where it is queued as needed by the simulation translator; the chosen link is reserved for a period corresponding to the transmission time, then the intermediate node is assigned as the new initial or origin switch and the program loops back. When long-message recalculation takes place, the new minimum Delay Vector is "transmitted" to the entries for the neighboring nodes, but the asynchronous update stops there. It can be extended to a "percolation" exchange if desired.

The SIM Program

The SIM program is the master program which initiates the

messages, determines the length of the simulation and the statistics of the random parameters, and collects data on the operation of the system upon completion of the simulation run. The heart of the program is the while loop; this determines the length of the run in simulation time, and during the run it generates messages with pseudo-random parameters (arrival time, length, and nodes) and initiates the periodic updating of the arrays. The only tricky part is performing the periodic update. The ASPOL program provides a TIME parameter which represents simulated time, but this is not really a continuous parameter. TIME is advanced by process "hold" instructions, which in the SIM program represent the arrival intervals and in the MESSAGE process represent the transmission delays. ASPOL takes account of the initiation times and all "hold" periods generated by particular messages, and advances the internal TIME paramter as these "holds" are executed in "chronogical" order; thus TIME takes on only those values which occur at the end of "hold" periods. Although in a "loaded" network there would be enough message arrivals and retransmissions so that the TIME parameter would assume a fairly dense set of values, nevertheless the density would vary, and thus the synchronous update is initiated by a repetitive "hold" period rather than by "clock-watching." The program has to interleave the pseudo-random arrival "holds" and the synchronous update "holds," and this is accompiished by the nested conditional statements.

The UPDATE Process

The UPDATE Process is not really a simulated process, since there is theoretically only one example in execution at any time, but this

format is convenient. In the actual system, the update is executed independently in each node. The update calculations performed by the process have been exactly explained in the description of the algorithm.

The Message Process

The MESSAGE Process does most of the simulation work. Each simulated message through the network is associated with an example of the MESSAGE process, and at any point in simulated time, many such examples exist. Each example has the following parameters associated with it: the message length, the initial node, the destination node, and the time of arrival at the initial node. These parameters are selected from pseudo-random function routines. In the initial simulation, the length of each message is selected from an exponential distribution with mean equal to the observed mean of present AUTODIN traffic. The initial node and destination node are selected from flat distributions (all nodes equally likely) and the time of arrival at the selected initial node is selected from a Poisson distribution whose parameters can be varied to change the system traffic load. These assumptions seem reasonable for the first steps. Later work would vary the selection of origin and destination nodes to reflect the actual traffic inputs to the network at specific nodes and would permit varying the arrival times at each node independently.

2.3.2 Simulated System Characteristics

The system simulated is AUTODIN in Europe [Appendix I]. With the CONUS gateways, the system has 8 nodes and 32 links as shown in

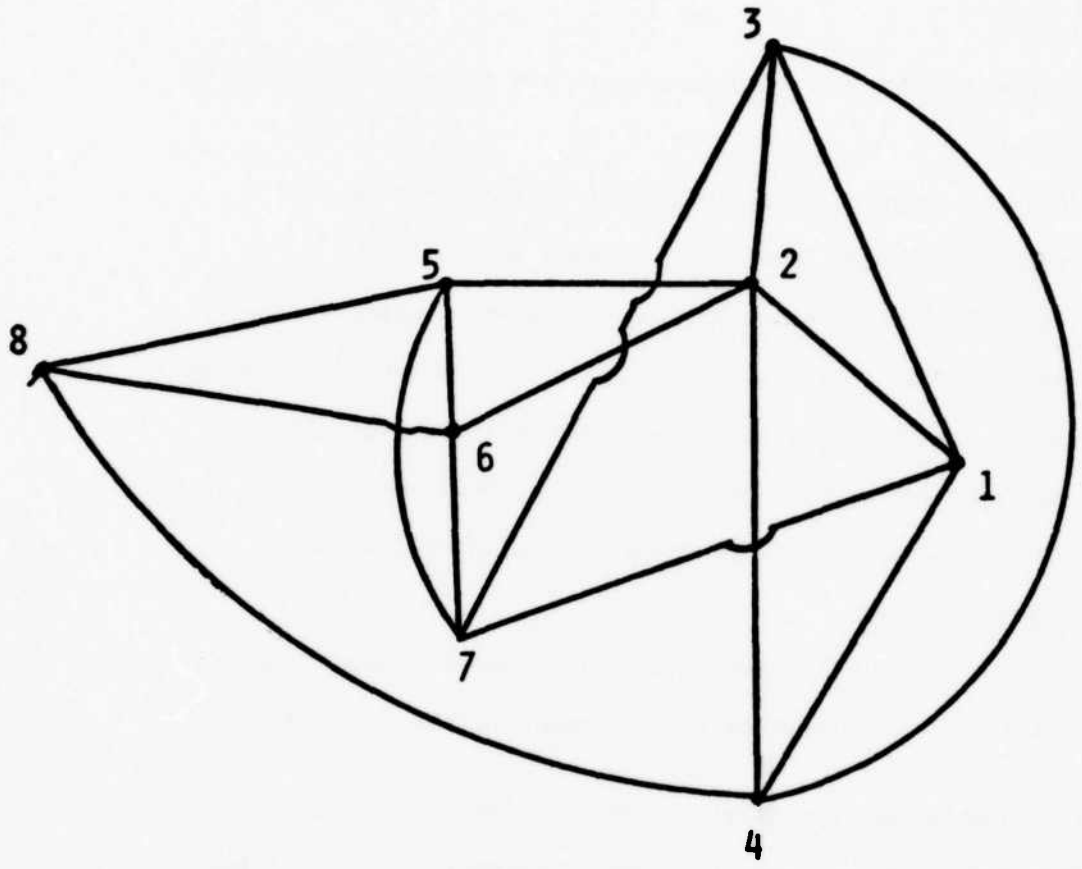


Figure 2.3 The AUTODIN network.

Fig. 2.3 All links are two way. The node connectivity varies from 3 to 5, and each node represents a switch. A detailed description of the simulation is contained in a Purdue University Technical Report [HOUS 75].

The values of the following parameters are drawn from probability distributions:

1. Message length: Exponential distribution with mean of 9 seconds, derived from actual message statistics.
2. Arrival time: Assumed Poisson stochastic process so that interarrival delay is exponentially distributed with average = 1/.88 = 1.14. This is derived from an average of 200,000 messages per day, 38 percent of which are assumed to be inter-switch traffic, giving a rate of .88 messages/sec. (arrival rate).
3. Initial and destination nodes: Independently drawn from a distribution uniform over the integer labels of the nodes.

The simulation was run several times for varying values of: synchronous update period, message length threshhold for initiation of asynchronous update (special routing) and the bias factor DELTA. The mnemonics UPDATE, TRSLD, and DELTA, respectively, are used for these quantities in the presentation of the results.

The variable UPDATE 20, 40, etc. means that the synchronous update process is being initiated every time 20, 40 messages etc. have arrived in the network.

The update information messages that the nodes send to each other after update takes place are accounted for by holding every link in the network for a short period of time (.007 sec. in this case) after the update calculations are over.

For each run, the delay for each message was measured and the average delay computed using the sample mean formula.

2.3.3 Conclusions

Simulation results are presented here using the Autodin network as an example. Data for two or three traffic loads were taken, when the inter-arrival time variable takes the values of $1/\gamma = 1.14$, which represents 38% of the overall Autodin traffic (76000 mes/day), $1/\gamma = .97$ which represents 44.6% of the traffic (89000 mes/day), and $1/\gamma = .9$ which represents 48.1% of the traffic (96200 mes/day). Data are presented for 1/2 hour simulated time.

In Fig.2.4, the RATIO of the ACTUAL TOTAL AVERAGE DELAY PER MESSAGE over the IDEAL TOTAL DELAY PER MESSAGE (ideal delay per message is taken to be the transmission time of the message following the direct route to its destination if there were no queues in the system) is plotted versus the bias factor DELTA for various update periods and with a fixed special-routing (asynchronous update) threshhold of 12. Three traffic loads are being investigated for $1/\gamma = 1.14$, $1/\gamma = .97$, $1/\gamma = .9$ in Fig. 2.4a, Fig. 2.4b, and Fig. 2.4c, respectively.

With a low threshhold of 12, slightly over the average of 9, a large portion of the messages are special-routed. The effect of DELTA is to bias the network towards choosing the shortest path.

The best results for all three traffic loads were obtained for the fastest update of 20 and low values of DELTA. Low values of DELTA would result in low encouragement of the direct route for messages whose length is below the threshold value.

When the update is fast, short length messages are delivered faster through alternate routes since the direct ones are heavily used by longer length messages using special routing.

This is clearly evident in Fig. 2.4c where the traffic is high. The best performance occurs at the lowest value of DELTA and update of 20.

Update of 40 confirms the above observation for every traffic load (see Fig. 2.4a,b,c). In Fig.2.4c, DELTA = 70 and UP = 20 show worse performance than that of UP = 40 for the same value of DELTA. At this point, due to the high traffic, heavy update data traffic is further degrading the network's performance.

As the update gets slower, the queue lengths change faster than the update can sense. Then, best performance moves along the intermediate to high values of DELTA. This occurs because the delay table information gets out of date, the wrong decision is taken, and thus the queues are longer. This is more evident with high traffic loads, (e.g., Fig.2.4c, with update of 100).

When the update gets very slow, high values of DELTA give better performance. It is interesting though to note that for DELTA = 70 and UP = 200 in Fig.2.4a and 2.4c, there is an increase of the ratio value.

It can be explained two ways:

1. There is some randomness introduced in the decision making

process due to the loss of effectiveness of the slow update. Then this randomness results in the lowest ratio value for DELTAs between 40 and 50 instead of higher ones near 70.

2. It might be thought of as the traffic's peculiarity (a sudden pile up of messages) at the time the sample was taken which results in better performance when 40< DELTA < 50.

If we increase the threshhold value to 35, then the immediate effect is a higher ratio value (see Fig.2.5a). This shows up more dramatically as the traffic gets higher (see Fig.2.5b). In Fig. 2.6a and in Fig.2.6b, the threshhold value is increased even more and the same observation is true.

Even with reduced special routing when the traffic is low and the update fast, the system's performance is comparable to that of a low threshhold (see Fig.2.5a, Fig.2.6a). As the traffic gets higher and the update slower, then the ratio value gets high, as expected (see Fig. 2.5b, Fig. 2.6b).

In Fig.2.7a and Fig.2.7b, the relation of the ratio value and DELTA is examined as the amount of special routing varies at a fixed update value of 40 for two traffic loads respectively.

Again, the benefit of the special routing is evident in both figures.

For low traffic, moderate bias towards the direct route shows best results. At a higher traffic, long messages (TRSLD = 70) are doing better when strongly biased towards the direct route. The same thing holds even for TRSLD = 35. At a TRSLD = 50, there is no significant change in performance with a changing DELTA. When special

routing is higher, (TRSLD = 12, 20) low bias values perform better.

For the purpose of a better illustration of the special routing effect, Fig.2.8 was plotted. Two extreme threshhold values were used at a fixed update of 20 and three traffic loads. The high traffic curves show better the need for special routing.

When the update is decreased (see Fig. 2.9a, 2.9b), interesting facts showed up in the high traffic plot. Best performance for high DELTA values almost everywhere. When the amount of special routing is changed, either a lot of special routing (TRSLD = 12, 20) will compensate for the slow update or very little (TRSLD = 70).

Bias toward the direct route for moderate length messages (TRSLD = 35, 50) causes long delays in the direct routes which results in a higher ratio value. This seems to suggest that in high traffic and slow update, it is better to directly route only very long length messages. Those, together with the ones directly routed by mistake because of out-of-date information, result in a better system performance than if, in addition, moderate length messages block the direct routes. That is, the use of alternate routing under the above stated conditions results in better system performance.

The above assertions are being supported from traffic data. The number of messages delivered which followed other than the direct path and took 1 or 2 or 3 or ... ≥, 10 "excess hops" to arrive at their destination was measured for DELTA = 70 and two threshhold values of 50 and 70. From Fig.2.12, although fewer messages followed the direct route in the case of TRSLD = 70 than in the case of TRSLD = 50,

the resulting actual delay per message for TRSLD = 70 is less than that at TRSLD = 50. Therefore, in TRSLD = 70 case, alternate routes are beneficial. In TRSLD = 50 case, overuse of direct routes degrades system performance. This is because when most messages take a direct route, subsequent long messages have to take an indirect one. The result is a high average delay. When small and moderate length messages are following alternate routes, then direct routes are used only for long messages. This results in a better overall average message delay.

Finally, the performance of the bias factor delta is investigated as a function of the update period when very little special routing takes place (TRSLD = 50) for two traffic loads in Fig. 2.10a and in Fig. 2.10b. For low traffic and fast update, low or moderate bias towards the direct routes results in the same ratio value, while strong bias does worse. For slow update, strong encouragement towards the direct route performs better (see Fig. 2.10a,) For higher traffic, Fig. 2.10b strong bias towards a direct path always performs better.

The effectiveness of DELTA to prevent looping has been investigated when the update is slow (UP = 100) and the special routing is moderate (TRSLD = 20) for two traffic loads. In Fig. 2.11 two tables are presented which show the number of messages delivered which followed other than the direct route and took 1 or 2 or 3 ... or ≥, 10 "excess hops" to arrive at their destination. Two values of DELTA 10 and 70 are recorded. They show that under the above stated conditions a high bias factor value prevents looping. This shows up best for high traffic conditions.

Summary

Low Traffic

Best performance when the update period is smaller than the average message length, at low values of the asynchronous update and a value of the bias factor DELTA approximately twice the average message length. The best asynchronous update threshhold value is 12.

When the update period gets greater than the average message length, a low threshhold value of 12 still maintains best performance at values of DELTA approximately twice the average message length. If the threshhold value is high so that only very long messages are specially routed, then the fastest update performs better.

In the worst case of slow update and high threshhold value, much higher than the average message length values of the bias factor DELTA give better results (DELTA = 70).

High Traffic

Best performance at update period smaller than the average message length, threshhold value of 12 and DELTA value of approximately equal to the average message length.

For a slow update, a low threshhold value of 12 maintains best performance at the highest DELTA value used (DELTA = 70).

In the worst case of slow update and high threshhold value, a DELTA value of 50 results in better performance.

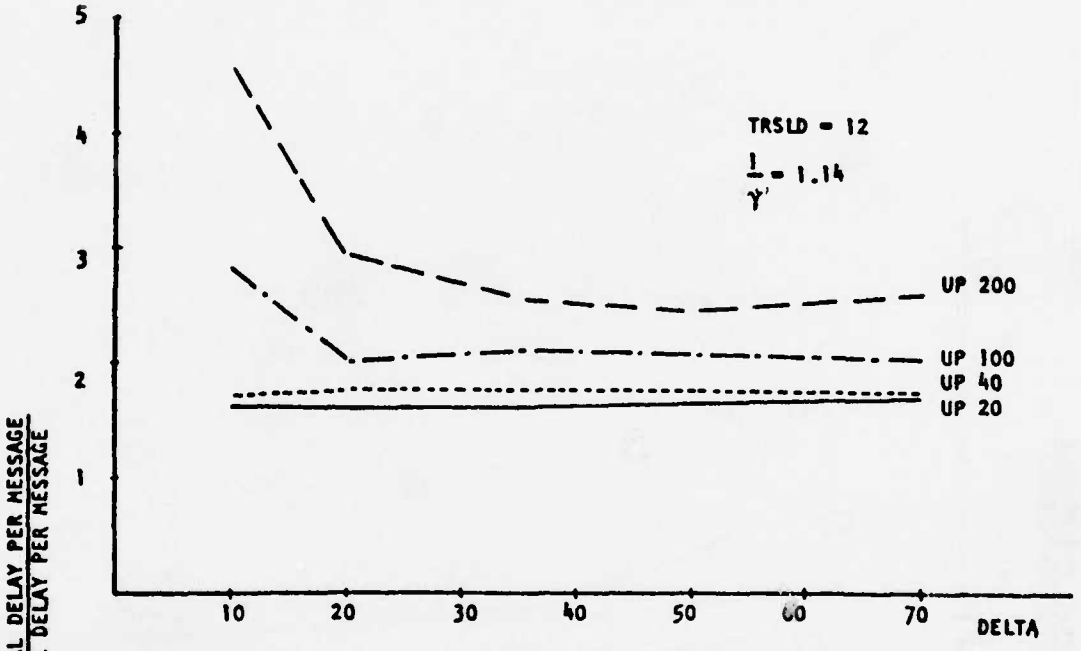


Figure 2.4(a) Ratio vs. Delta for TRSLD = 12, $1/\gamma = 1.14$.

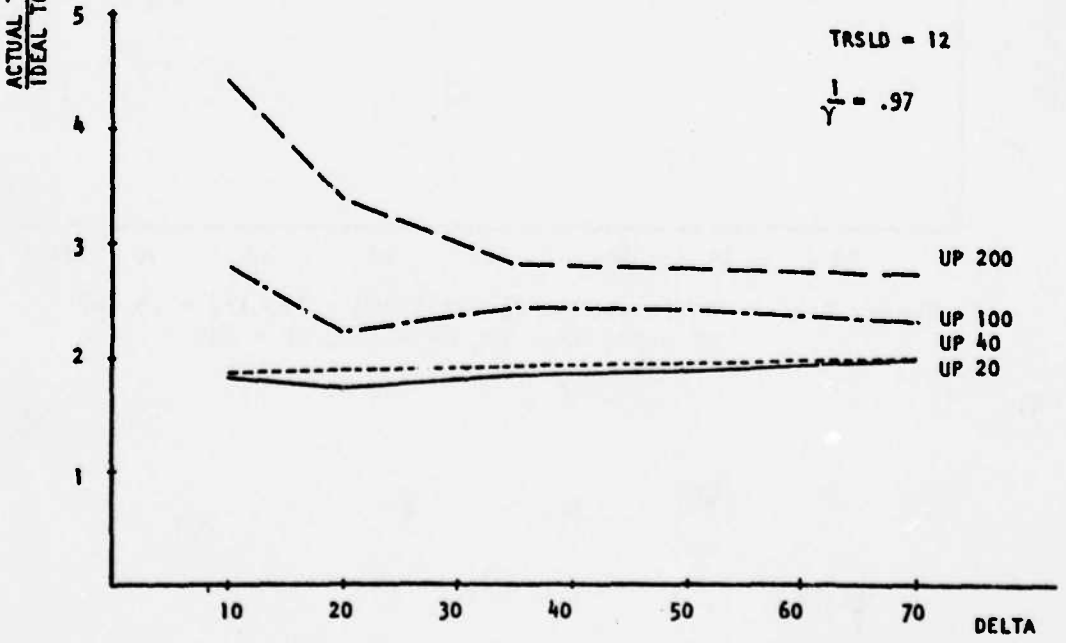


Figure 2.4(b) Ratio vs. Delta for TRSLD = 12, $1/\gamma = .97$ and UP = 20, UP = 40, UP = 100, UP = 200.

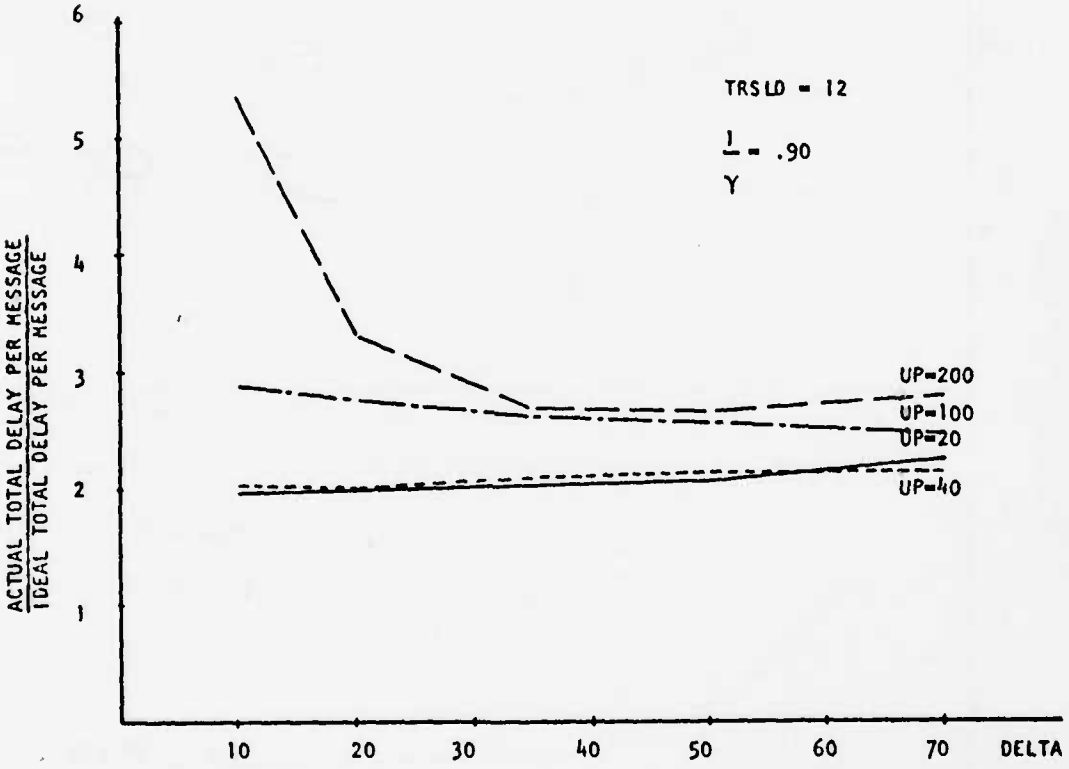


Figure 2.4(c) Ratio vs. Delta for TRSLD = 12, $1/\gamma$ = .9 and UP = 20, UP = 40, UP = 100, UP = 200.

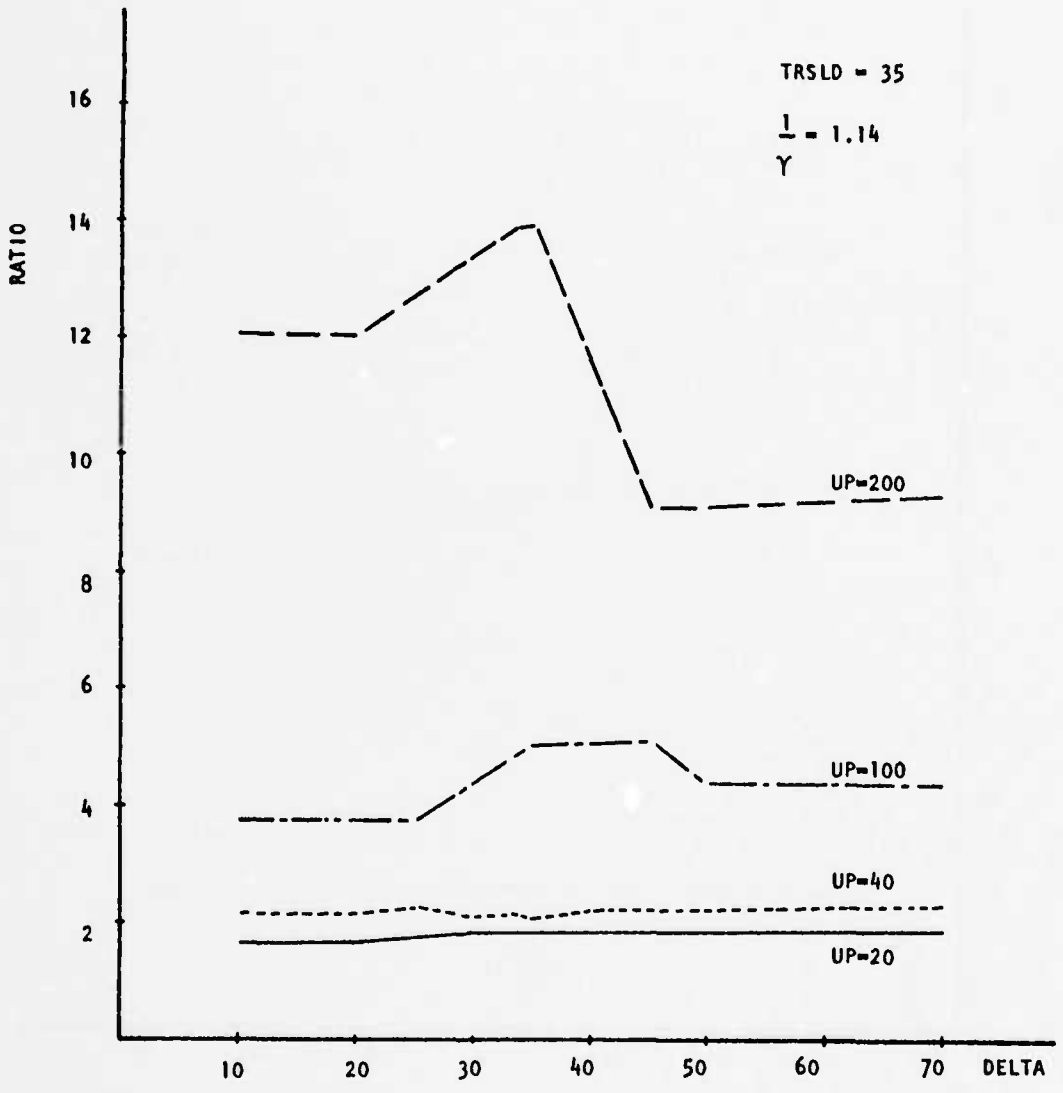


Figure 2.5(a) Ratio vs. Delta for TRSLD = 35, $1/\gamma = 1.14$ and UP = 20, UP = 40, UP = 100, UP = 200.

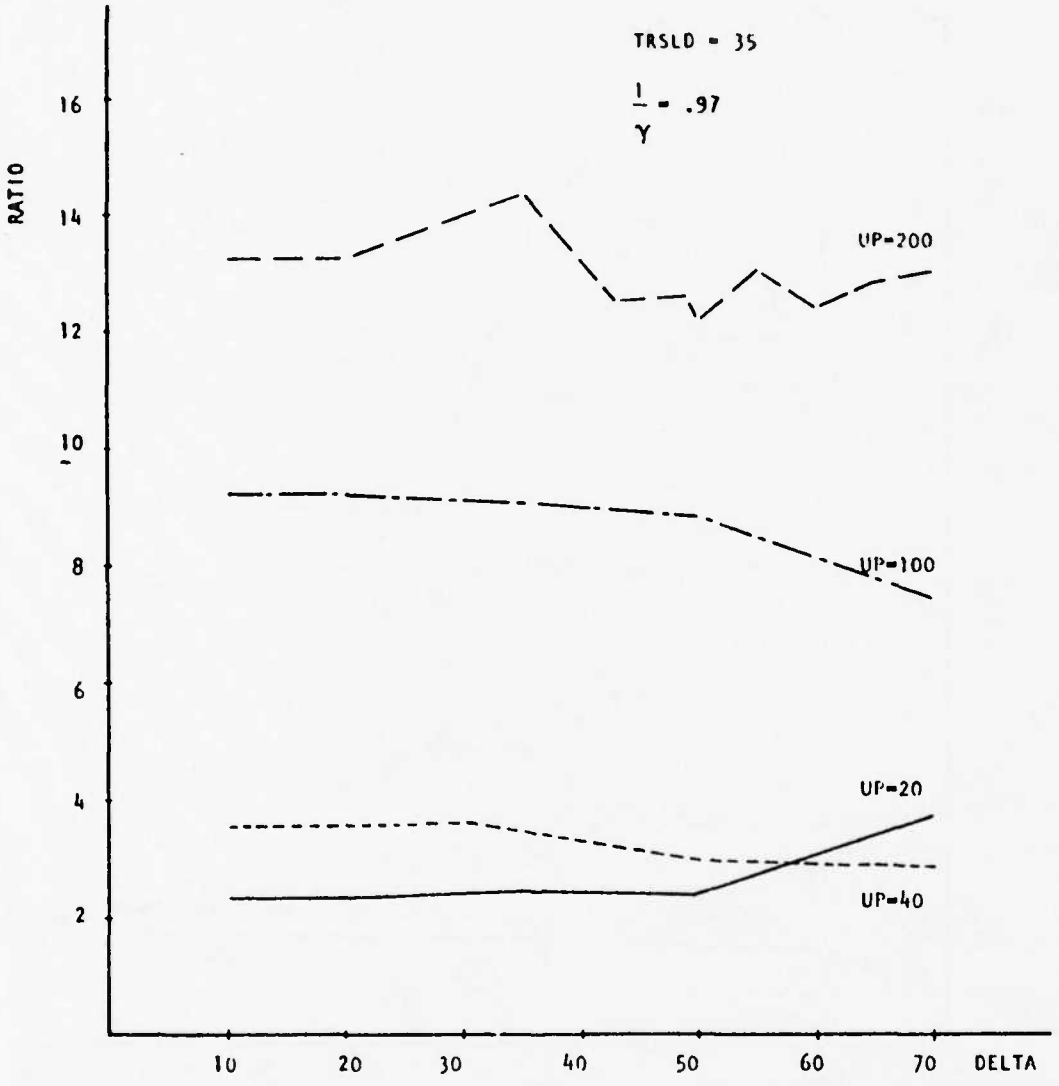


Figure 2.5(b) Ratio vs. Delta for TRSLD = 35, $1/\gamma$ = .97 and UP = 20, UP = 40, UP = 100, UP = 200.

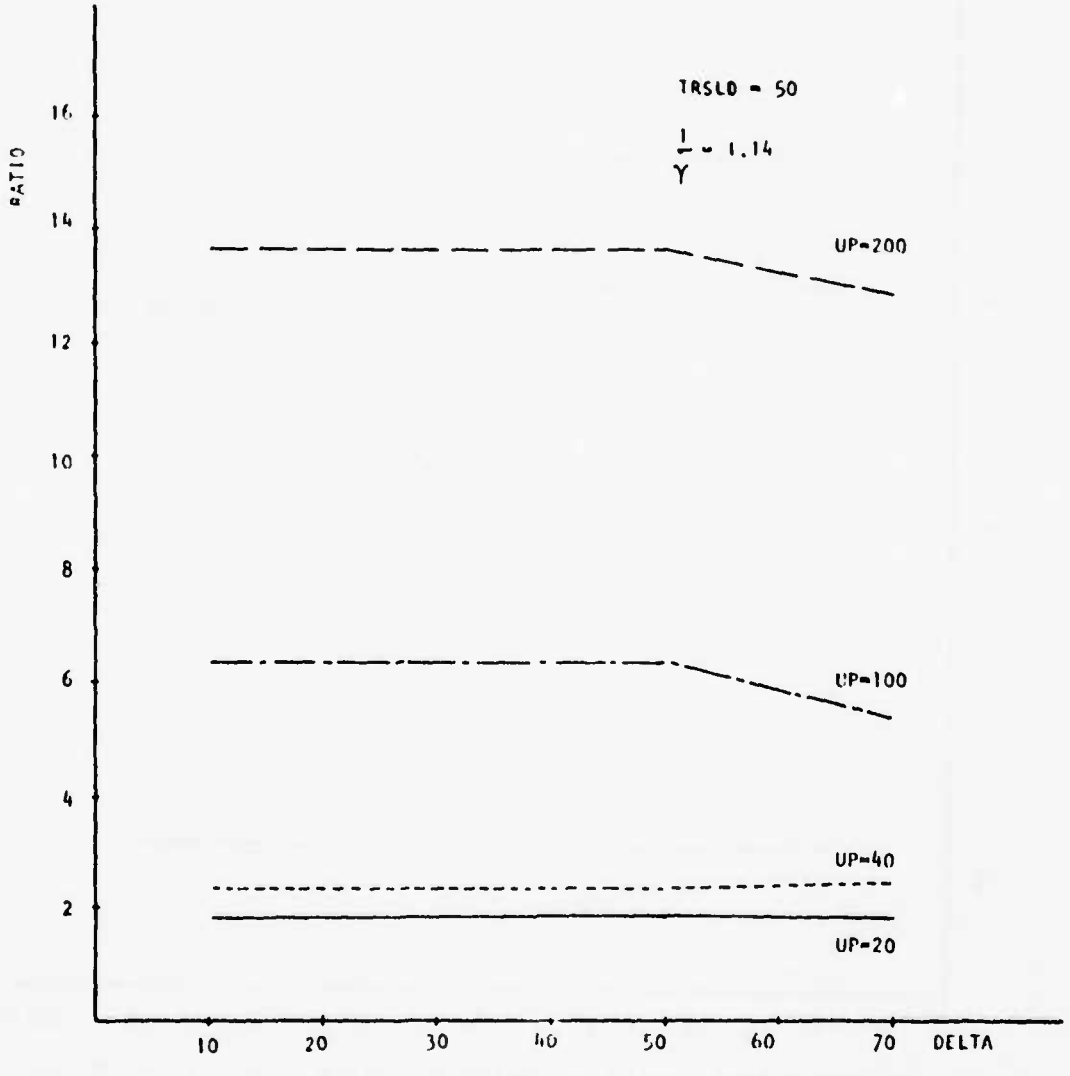


Figure 2.6(a) Ratio vs. Delta for TRSLD = 50, $1/\gamma$ = 1.14 and UP = 20, UP = 40, UP = 100, UP = 200.

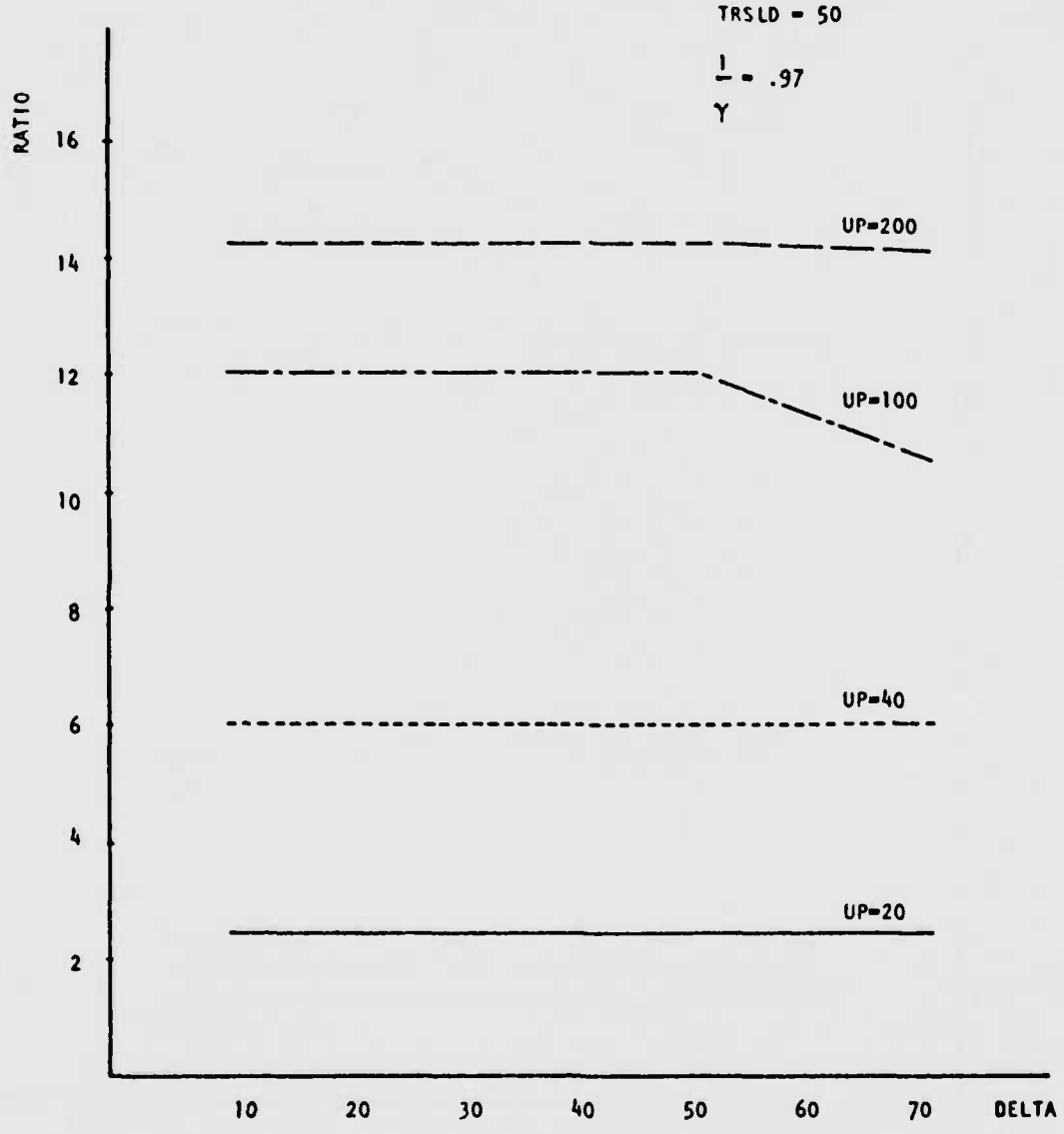


Figure 2.6(b) Ratio vs. Delta for TRSLD = 50, $1/\gamma$ = .97 and UP = 20, UP = 40, UP = 100, UP = 200.

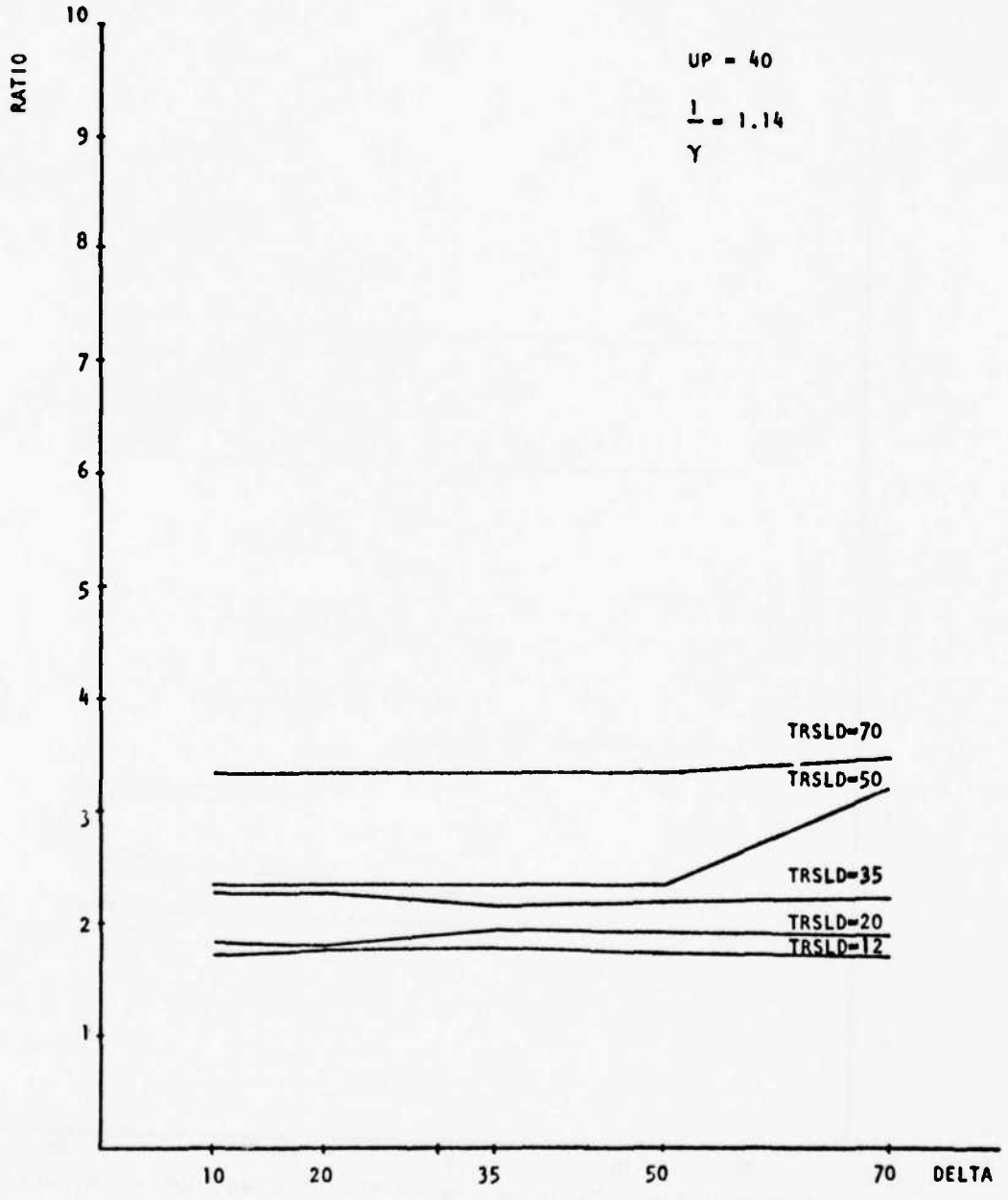


Figure 2.7(a) Ratio vs. Delta for UP = 40, $1/\gamma = 1.14$ and TRSLD=12, TRSLD=20, TRSLD=35, TRSLD=50, TRSLD=70.

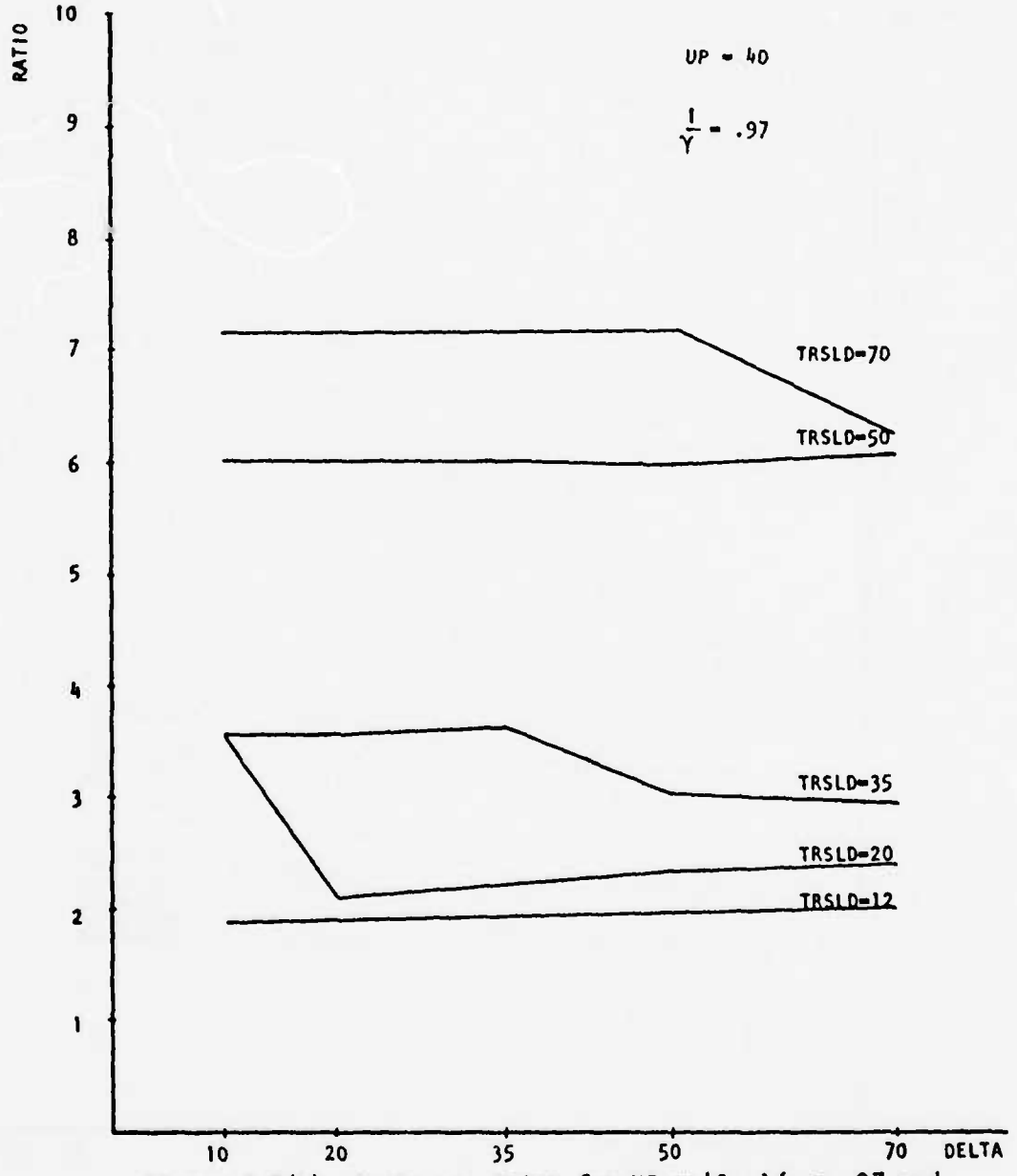


Figure 2.7(b) Ratio vs. Delta for UP = 40, $1/\gamma$ = .97 and TRSLD=12, TRSLD=20, TRSLD=35, TRSLD=50, TRSLD=70.

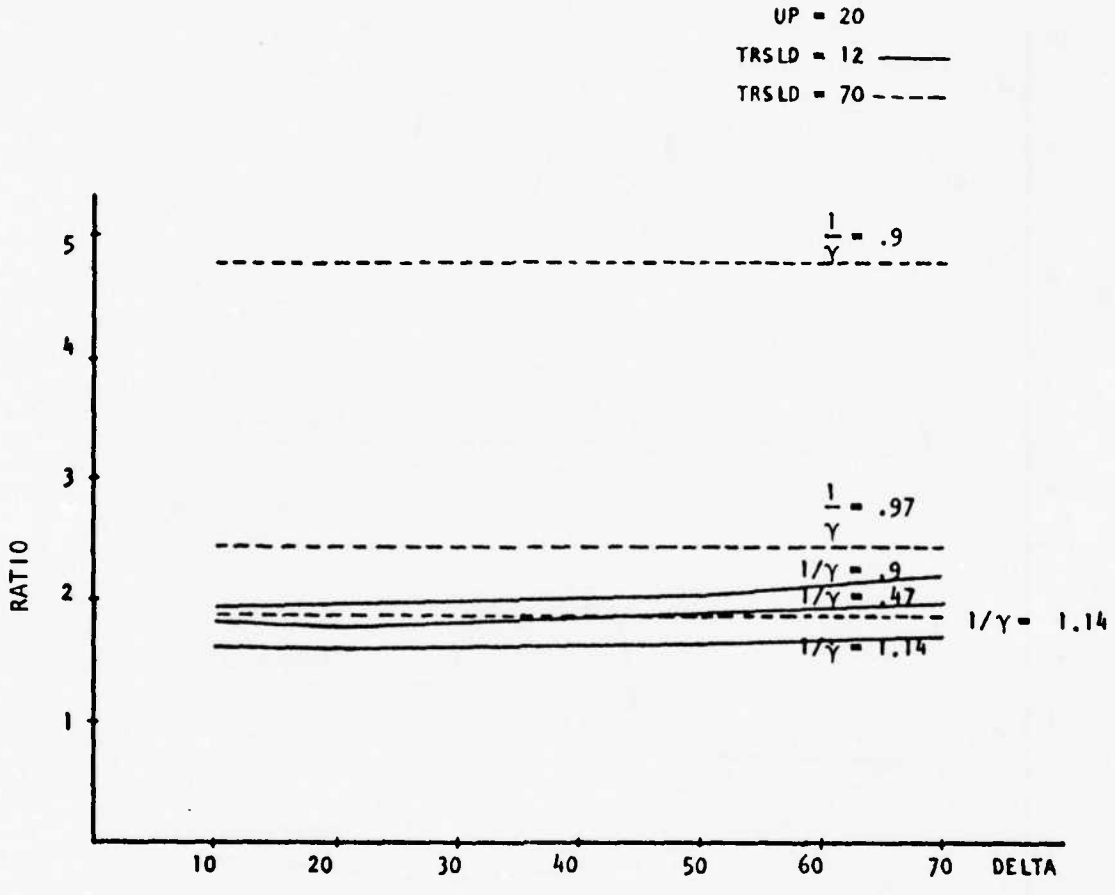


Figure 2.8 Ratio vs. Delta for UP = 20, TRSLD = 12, TRSLD = 70 and $1/\gamma = 1.14$, $1/\gamma = .97$, $1/\gamma = .9$.

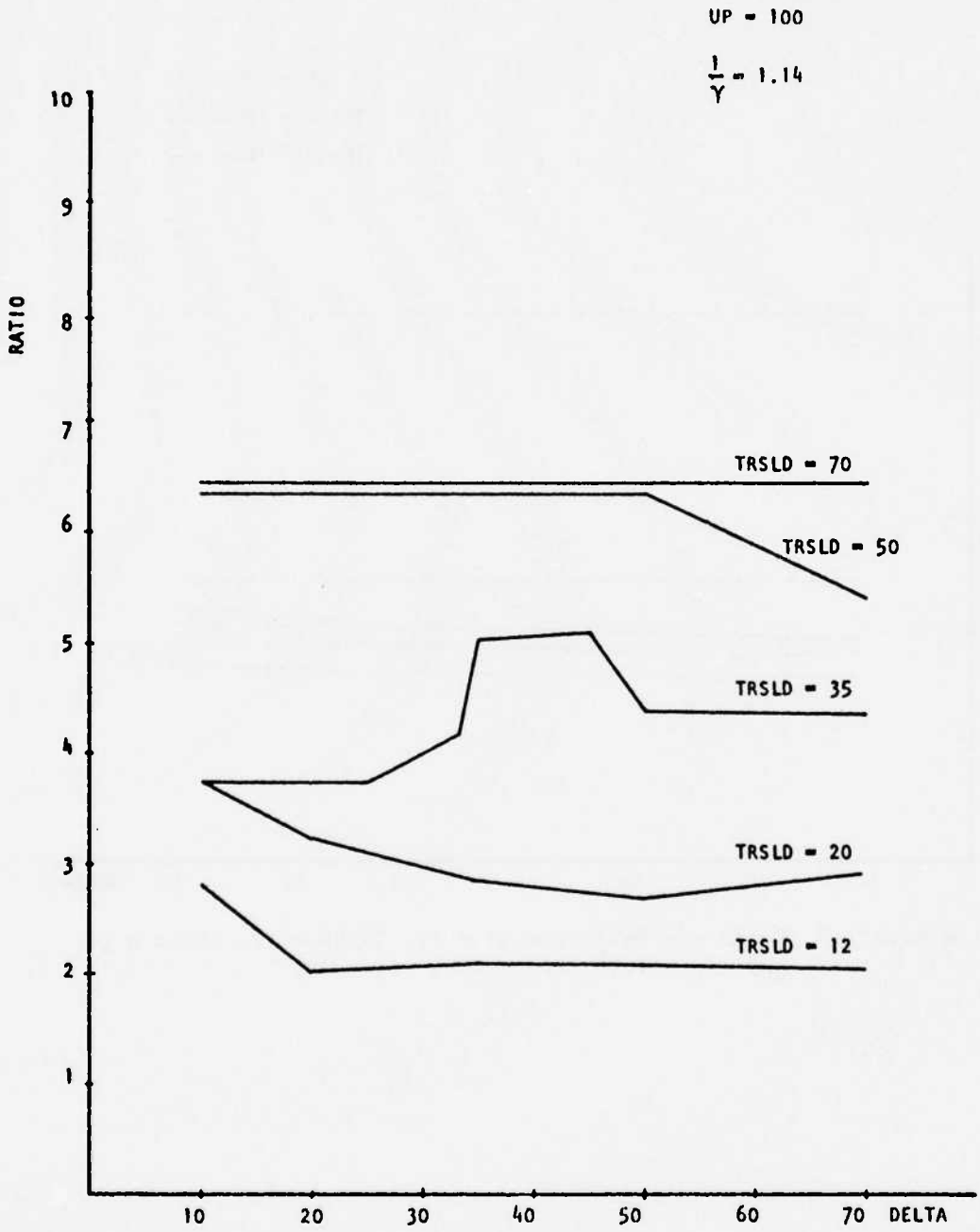


Figure 2.9(a) Ratio vs. Delta for UP = 100, $1/\gamma = 1.14$, and TRSLD=12, TRSLD=20, TRSLD=35, TRSLD=50, TRSLD=70.

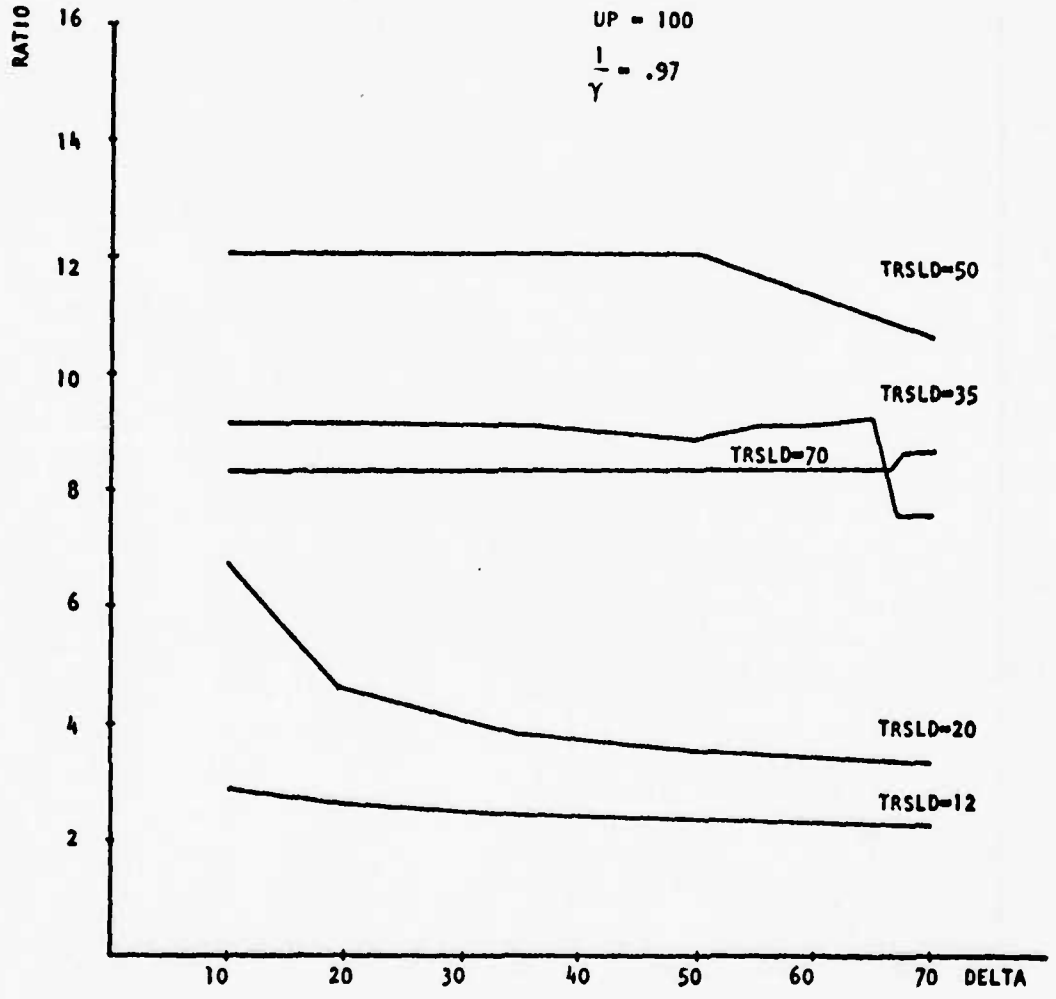


Figure 2.9(b) Ratio vs. Delta for UP = 100, $1/\gamma = 1.14$ and TRSLD=12, TRSLD=20, TRSLD=35, TRSLD=50, TRSLD=70.

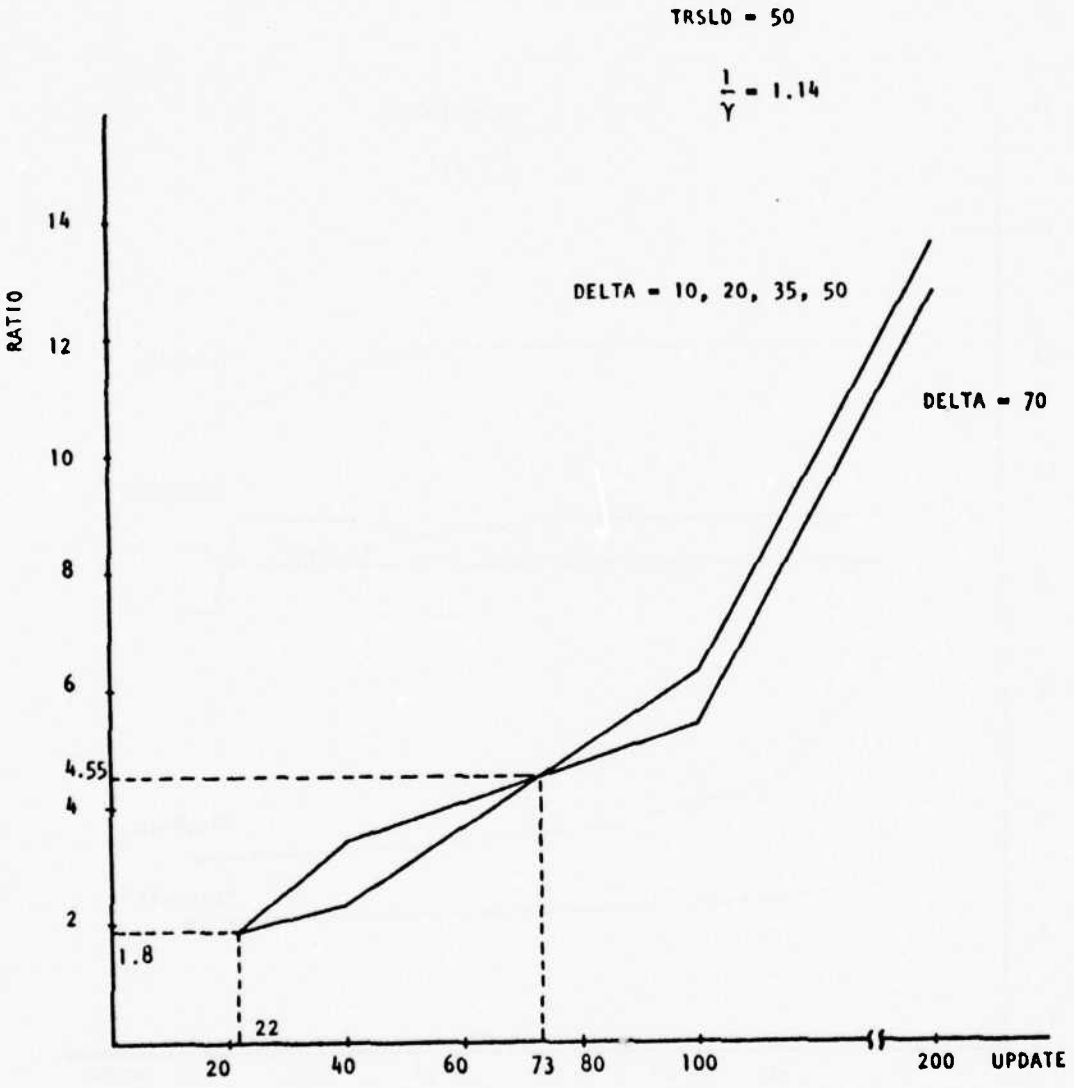


Figure 2.10(a) Ratio vs. update for TRSLD = 50, $1/\gamma$ = 1.14 and DELTA = 10, 20, 35, 50, DELTA = 70.

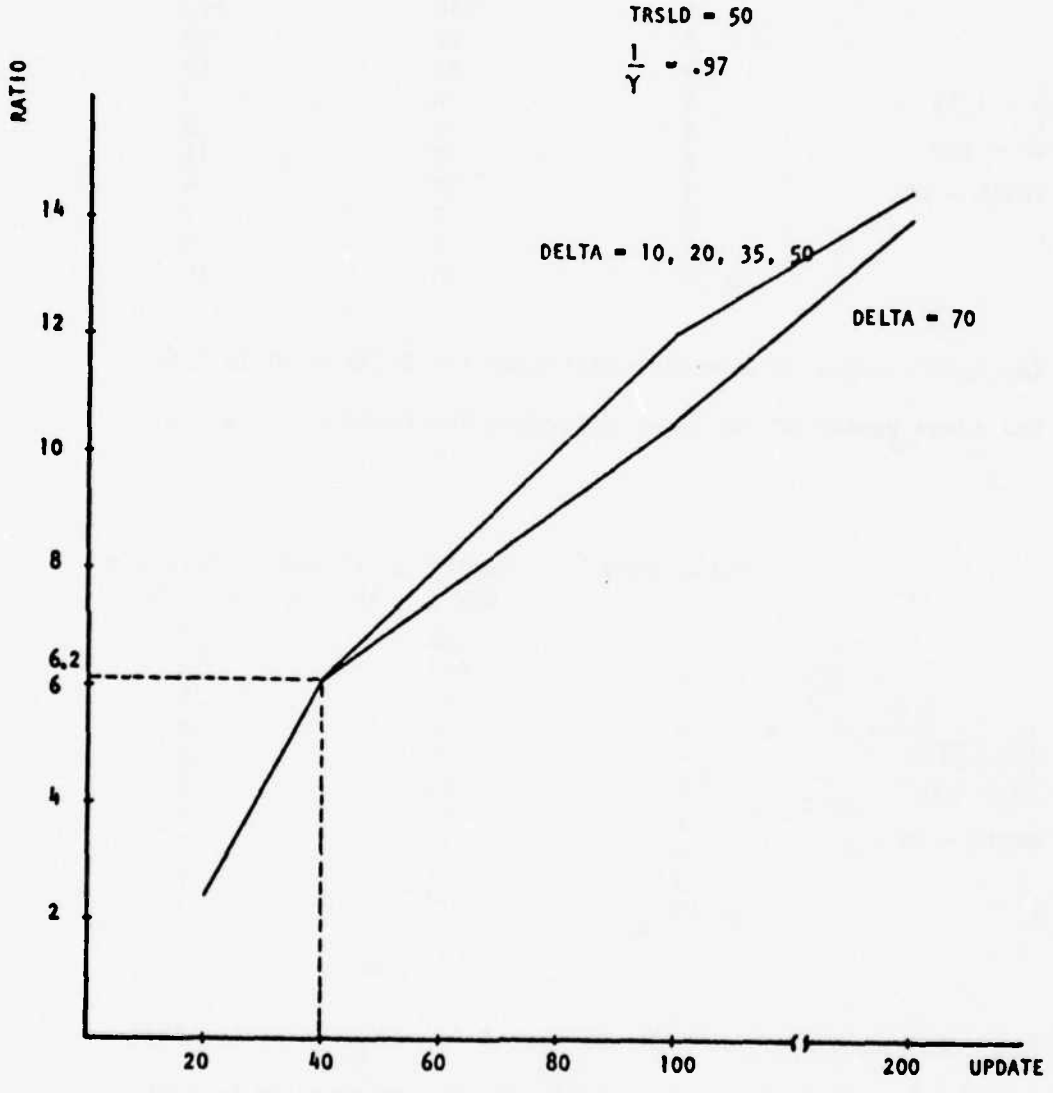


Figure 2.10(b) Ratio vs. update for TRSLD = 50, $1/\gamma$ = .97 and DELTA = 10, 20, 35, 50, DELTA = 70.

	EXCESS HOPS	NUMBER OF MESSAGES DELIVERED DELTA = 10	DELTA = 70
	1	240	193
	2	87	58
	3	55	19
$\frac{1}{\gamma}$ = 1.14	4	0	0
UP = 100	5	9	9
	6	25	15
TRSLD = 20	7	9	4
	8	7	3
	9	2	5
	≥ 10	11	14

The total number of messages delivered for DELTA = 10 is 1365.

The total number of messages delivered for DELTA = 70 is 1367.

	EXCESS HOPS	NUMBER OF MESSAGES DELIVERED DELTA = 10	DELTA = 70
	1	258	225
	2	177	122
	3	78	29
	4	0	0
$\frac{1}{\gamma}$ = .97	5	29	8
UP = 100	6	29	8
	7	15	4
TRSLD = 20	8	12	6
	9	6	3
	≥ 10	15	7

The total number of messages delivered for DELTA = 10 is 1543.

The total number of messages delivered for DELTA = 70 is 1584.

Figure 2.11 Excess hops vs. number of messages delivered.

	EXCESS HOPS	NUMBER OF MESSAGES DELIVERED TRSLD = 10	TRSLD = 70
	1	260	282
	2	201	218
	3	89	73
$\frac{1}{\gamma} = .97$	4	0	0
UP = 100	5	35	22
DELTA = 70	6	24	27
	7	12	7
	8	5	6
	9	3	3
	≥ 10	8	6

Total number of messages delivered for TRSLD = 50 is 1438.

Total number of messages delivered for TRSLD = 70 is 1507.

Figure 2.12 Excess hops vs. number of messages delivered for TRSLD = 10, 70.

CHAPTER 3

DETERMINISTIC ROUTING STRATEGY

Introduction

In a deterministic routing strategy, a predetermined function of all incoming messages is sent to each output queue.

The optimal routing problem consists of finding the deterministic routing strategy (or, equivalently the set of routes) according to which messages have to be routed so as to optimize a well defined performance function.

Under appropriate assumptions, the optimal routing problem can be formulated as a nonlinear multicommodity flow problem [FRAN 71A].

Several techniques for solving the multicommodity problem can be found in the mathematical programming literature [DANT 63],[GERL 73]: however, the straight-forward application of these techniques, which are appropriate for the solution of very general multicommodity problems, to the routing problem in store-and-forward nets are computationally cumbersome. Therefore, an extremely fast routing technique has to be used. For that reason, considerable effort has been spent in developing heuristic and suboptimal methods [FRAN 71B, FULT 72]. The exact mathematical problem has been attacked by [GERL 73A]. He formulated and solved it as an unconstrained nonlinear programming problem. In fact, he presents two methods: the flow deviation (FD) method and the external flows methods.

To make the presentation of this report more complete, the FD method will be adapted here for the solution of the deterministic problem. It is chosen among others because of its simplicity and good performance. The deterministic case for the same eight node network, as in Chapter 2, is simulated with the computer in order to evaluate the adaptive routing algorithm.

In section 3.1, the definition of the routing problem is formulated.

In section 3.2, the multicommodity flow problem is presented in terms of the total flow in every link, along with the constraints that it satisfies.

In section 3.3, the performance function chosen is described.

In section 3.4, a brief description of the solution of the problem is presented according to the FD method and an outline of the FD algorithm is formulated.

In section 3.5, the application of the FD method to the routing problem is investigated.

In section 3.6, the algorithm is applied to the routing of an eight node network.

In section 3.7, an evaluation of the adaptive routing algorithm is presented by comparing its performance with the performance of the deterministic routing.

3.1 The Deterministic Routing Problem

Consider a network consisting of nodes N_i ($i = 1, \ldots, N$) and links L_j ($j = 1, \ldots, M$). In such a network, we are required to route

a quantity r_{ij} of type (i,j) messages from M_i (source node) to N_j (destination node), for all pairs (i,j). The routing problem is defined as the problem of finding the routing of all such messages which optimizes a well-defined performance function such that a set of constraints are satisfied.

Definition: We define $f_i^{(k,\ell)}$ to be the average flow [bits/sec] produced in link i by messages traveling from source node k to destination node ℓ. Let f_i be the total average flow in link i, given by

$$f_i \triangleq \sum_{k=1}^{N} \sum_{\ell=1}^{N} f_i^{(k,\ell)}$$

where N is the number of nodes in the net.

Definition: We denote by "commodity (i,j)" the messages which are generated in node i and have destination node j. Commodity (i,j) produces the single commodity flow $\underset{\sim}{f}^{(i,j)}$. All commodities together produce the multicommodity flow $\underset{\sim}{f}$.

Definition: Let γ_{ij} [messages/sec] be the required average rate of transmission of messages from source i to destination j, and let $1/\mu$ [bits/message] be the average message length. We define the requirement r_{ij} [bits/sec] as follows:

$$r_{ij} \triangleq \gamma_{ij}/\mu$$

and define the requirement matrix

$$R \triangleq \{r_{ij}\}.$$

The routing problem then can be in general formulated in the following way:

given: a network of N nodes and M links and a N x N matrix $R = [r_{ij}]$, the requirement matrix, whose entries are non-negative.

minimize: $P(\underset{\sim}{f})$
over $\underset{\sim}{f}$

where $\underset{\sim}{f} \triangleq (f_1, f_2, \ldots, f_M)$ and $P(\cdot)$ is a well defined performance function.

constraints: (a) $\underset{\sim}{f}$ is a multicommodity flow satisfying the requirement matrix R.

(b) $\underset{\sim}{f}$ must satisfy some additional constraints (e.g. capacity constraints on each link or cost constraints).

In the case of a store-and-forward communication network, the routing problem has the following formulation:

given: a network of N nodes and M links and a N x N matrix $R = [r_{ij}]$, whose entries are non-negative.

$$\text{minimize: } T^{(K)} = \left[\sum_{i=1}^{M} \frac{\lambda_i}{\gamma} (T_i)^K\right]^{1/K} = \left[\sum_{i=1}^{M} \frac{(\frac{1}{\mu})^{K-1}}{\gamma} \frac{f_i}{(C_i - f_i)^K}\right]^{1/K}$$
over $\underset{\sim}{f}$

constraints: (a) $\underset{\sim}{f}$ is a multicommodity flow satisfying the requirement matrix R.

(b) $\underset{\sim}{f} \leq \underset{\sim}{C}$.

where $\underset{\sim}{C} = (C_1, C_2, \ldots, C_M)$

3.2 Representation of a multicommodity flow

A practical representation of a multicomodity flow is through the vector $\underset{\sim}{f}$ of the flow of the links.

$\underset{\sim}{f} = [f_1, f_2, \ldots, f_M]$

where f_i is the total flow on link (or channel) i, sum of all commodities flowing through i.

$\underset{\sim}{f}$ does not completely characterize a multicommodity flow: for instance, two different sets of routes might result in the same $\underset{\sim}{f}$. In many multicommodity problems, however, the performance function depends only on the total flows f_i. In such cases, the $\underset{\sim}{f}$ - representation is sufficient. The problem under investigation belongs to such a category. It is of interest, therefore, to investigate the properties of the set $\underset{\sim}{F}$ defined as the set of all feasible $\underset{\sim}{f}$.

$$\underset{\sim}{F} = \underset{\sim}{F}_a \cap \underset{\sim}{F}_b$$

where $\underset{\sim}{F}_a \triangleq \{\underset{\sim}{f} | \underset{\sim}{f}$ satisfies multicommodity constraint (a)$\}$

$\underset{\sim}{F}_b \triangleq \{\underset{\sim}{f} | \underset{\sim}{f}$ satisfies capacity constraint (b)$\}$

3.2.1. The multicommodity constraint (a)

Let $f_{ij}^{(m,n)}$ be the flow on link (i,j) due to commodity (m,n). For the conservation of flow at each node, for each commodity (l,j), and non-negativity of the flow in each link, the following constraint is true.

$$\sum_{j=1}^{N} f_{ji}^{(m,n)} - \sum_{j=1}^{N} f_{ij}^{(m,n)} = \begin{cases} -r_{mn} & \text{if } l = m \\ 0 & \text{if } l \neq m,n \\ r_{mn} & \text{if } i = n \end{cases} \qquad (3.1)$$

$$l = 1,2,\ldots,N$$

For non-negativity of the flows we have:

$$f_{ij}^{(m,n)} \geq 0 \quad \forall\ i,j \qquad (3.2)$$

Let $\underset{\sim}{f}^{(m,n)} \triangleq (f_1^{(m,n)}, f_2^{(m,n)}, \ldots f_M^{(m,n)})$, where $f_i^{(m,n)}$ is the flow due to commodity (m,n) on the link labeled i, and let $\underset{\sim}{F}^{(m,n)} = \{\underset{\sim}{f}^{(m,n)} | \underset{\sim}{f}^{(m,n)}$ satisfy equation (3.1) and (3.2)$\}$. Then $\underset{\sim}{F}^{(m,n)}$, defined by linear equations (3.1) and inequalities (3.2) is a convex polyhedron. By definition, the total flow $\underset{\sim}{f}$ is given by

$$\underset{\sim}{f} = \sum_{m=1}^{N} \sum_{n=1}^{N} \underset{\sim}{f}^{(m,n)} \tag{3.3}$$

It can be shown [HU 69] that, when, for each commodity (i,j), $\underset{\sim}{f}^{(i,j)}$ spans the respective feasible set $\underset{\sim}{F}^{(i,j)}$, then the total flow $\underset{\sim}{f}$ according to (3.3) also spans a convex polyhedron.

Therefore, $\underset{\sim a}{F}$ is a convex polyhedron (Fig. 3.1). The flows corresponding to the flows of the "corners" (extreme points) of the polyhedron $\underset{\sim a}{F}$ have an interesting property: they are the shortest route flows[1] [FORD 62]. It is also shown that all shortest route flows correspond to extreme points of $\underset{\sim a}{F}$.

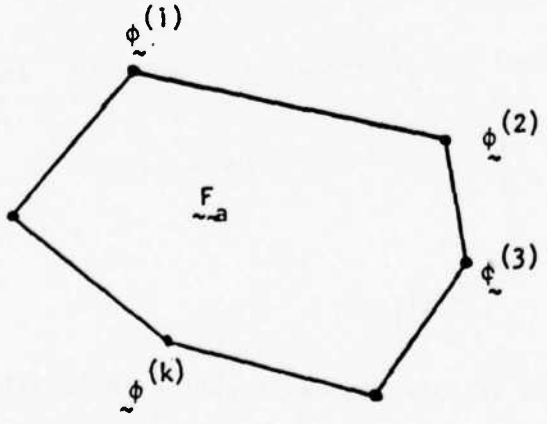


Figure 3.1 Feasible set $\underset{\sim a}{F}$ and extremal flows.

[1] A shortest route flow is a multicommodity flow whose routes can be described by a shortest route matrix, computed for an arbitrary assignment of lengths to the links.

3.2.2 The capacity constraint (b)

The set $\underset{\sim}{F}_b$, defined by:

$$\underset{\sim}{F}_b \triangleq \{\underset{\sim}{f} | \underset{\sim}{f} \leq \underset{\sim}{C}\}$$

is a convex set.

Hence the feasible set $\underset{\sim}{F} = \underset{\sim}{F}_a \cap \underset{\sim}{F}_b$ is also a convex set. See Fig. 3.2.

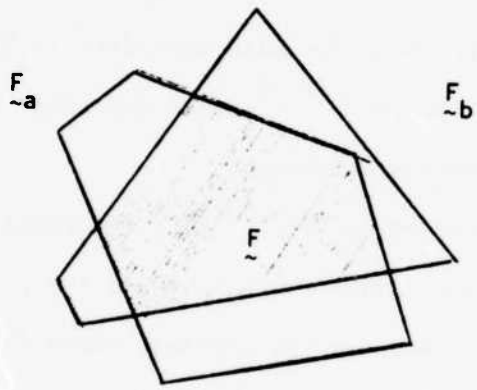


Figure 3.2 Feasible set $\underset{\sim}{F} = \underset{\sim}{F}_a \cap \underset{\sim}{F}_b$.

3.3 The Performance Function

As a performance function, the Mean-kth-Power Delay $T^{(K)}$ is used.

$$T^{(K)} = \left[\sum_{i=1}^{M} \frac{\lambda_i}{\gamma} (T_i)^K \right]^{1/K} \qquad 1 \leq K < \infty \tag{3.4}$$

where

$$T_i = \frac{1}{\mu C_i - \lambda_i} \quad \text{and} \quad \lambda_i = \mu f_i .$$

The function (3.4) is a strictly convex function [MEIS 71].

For K = 1, we obtain the average delay as in eq. (1.7). For K = 2, the weighted variance of message delay is minimized [MEIS 71]. If K

is allowed to become large, optimization under this performance function leads to equallzing the messaye flow in the links.

The Mean-kth-Power Delay does not have a physical significance. It is more sensitive to the needs of the individual node.

3.4 Description of the Flow Deviation Method

Before we go into the description of the FD method, an important observation about the behavior of the performance function will facilitate the solution of the problem.

A simple inspection of the problem presented in section 4.1 shows that

$$\lim_{f_i \to C_i} T(\underset{\sim}{f}) = +\infty$$

This fact will prevent $\underset{\sim}{f}$ from approaching the boundary of $\underset{\sim}{F}_b$ when T is minimized.

Such behavior is quite general for store-and-forward networks. When the capacity constraints are satisfied with equality, usually some saturation occurs, the queues at nodes grow large, and the delay T increases rapidly. We can then say that the performance function incorporates the capacity constraints as a penalty function.

This property is very important because it guarantees automatically the (b)-feasibility during the application of the usual nonlinear optimization techniques, once a feasible flow $\underset{\sim}{f}^0$ is formed.

Therefore, if we have a starting feasible flow $\underset{\sim}{f}^0$, we can disregard the capacity constraints, and we can consider the routing problem as an unconstraint multicommodity flow problem. Then the

above problem is classified as an unconstrained multicommodity flow problem with nonlinear performance function.

The problem is separable* convex and differentiable. The approach for solving it consists of approximating the performance function with the tangent hyperplane, which is expressed in terms of the partial derivitives $\{\partial P/\partial f_i\}$. Then the min cost problem of the linearized problem is the shortest route flow**, where the length of link i is defined as $\partial P/\partial f_i$. Such shortest route flow represents the direction of the steepest descent flow deviation.

The above idea is the essence of the FD method, which consists of repeated evaluations of steepest descent directions and of one variable minimization along such directions; the method is conceptually very similar to the gradient method applied to nonlinear minimization problems.

The idea of using shortest routes (computed with partial derivitives) is not new. Dafermos [DAFE 69] solved various traffic problems, formulated as unconstrained, convex m.c. flow problems. Yaged [YAGE 71] solved the min cost capacity assignment problem for a communication network, which was formulated as an unconstrained m.c. flow problem.

---

*A separable m.c. flow problem has the form

$$P(f) = \sum_{i=1}^{M} P_i(f_i)$$

**A shortest route flow is a multicommodity flow whose routes can be described by a shortest route matrix, computed for an arbitrary assignment of lengths to the links.

3.4.1 The FD Method

Gerla [GERL 73B] proved that if $\underset{\sim}{f}$ is a stationary flow*, then the shortest route flow evaluated under metric $\ell_K = \partial P/\partial f_K$ represents the flow deviation of steepest decrease for P. This fact suggests a method which is called Flow Deviation Method, for the determination of stationary solutions of the unconstrained nonlinear convex differentiable flow problems.

Let's define FD as an operator

$$FD(\underset{\sim}{v},\lambda) \odot \underset{\sim}{f} \triangleq (1-\lambda)\underset{\sim}{f} + \lambda\underset{\sim}{v} = \underset{\sim}{f}'$$

where

$\underset{\sim}{v}$ is a properly chosen m.c. flow $\in \underset{\sim}{Fa}$

λ is the step size $(0 \leq \lambda \leq 1)$

Now for each $\underset{\sim}{f} \in \underset{\sim}{Fa}$, we want to determine a pair $(\underset{\sim}{v},\lambda)$ in such a way that the repeated application of $FD(\underset{\sim}{v},\lambda)$ produces a sequence $\{\underset{\sim}{f}^n\}$ which converges to a stationary flow.

For a function $P(\underset{\sim}{f})$ which is continuous nondegenerate** and lower bounded, the following conditions are sufficient for the convergence of the FD to a stationary flow:

---

*$\underset{\sim}{f}$ is defined as stationary if for any infinitesimal perturbation $\underset{\sim}{\delta f}$ (such that $\underset{\sim}{f} + \underset{\sim}{\delta f}$ is also a m.c. flow)

$$P(\underset{\sim}{f} + \underset{\sim}{\delta f}) \geq P(\underset{\sim}{f})$$

A local minimum is always stationary. The opposite is not true: for instance, a local maximum is also stationary.

**P(f) is defined to be nondegenerate, for any two distinct stationary flows, say $\underset{\sim}{f}^1$ and $\underset{\sim}{f}^2$, we have

$$P(\underset{\sim}{f}^1) \neq P(\underset{\sim}{f}^2)$$

(i) $\Delta P(\underset{\sim}{f}) \geq 0 \quad \underset{\sim}{f} \in \underset{\sim}{F}$

(ii) $\Delta P(\underset{\sim}{f}) = 0 \rightarrow \underset{\sim}{f}$ stationary

where

$$\Delta P(\underset{\sim}{f}) = P(\underset{\sim}{f}) - P(FD \circledcirc \underset{\sim}{f})$$

Conditions (i) and (ii) require that the FD method be a true steepest descent method.

Under the assumptions tht $P(\underset{\sim}{f})$ is continuous, lower bounded, first partial derivitives continuous and nonnegative; second partial derivitives $< \infty$, $P(\underset{\sim}{f})$ nondegenerate, the following definition of $FD(\underset{\sim}{v}, \lambda)$ satisfies conditions (i) and (ii):

$\underset{\sim}{v} \triangleq$ shortest route flow under metric ℓ_K

$\lambda \triangleq$ minimizer of $P[(1-\lambda)\underset{\sim}{f} + \lambda \underset{\sim}{v}]$, $0 \leq \lambda \leq 1$

where $\ell_K = \dfrac{\partial P(f)}{\partial f_K}$ is the assigned weight (length) to the links of the network. The shortest route π_{ij} is, therefore, the route for which $\sum_{K \in \pi_{ij}} \ell_K$ is minimum.

An algorithm then based on the definition of the FD operator is outlined as follows:

1. Find a feasible starting flow $\underset{\sim}{f}^0$
2. Let $n = 0$
3. Find the shortest route flow $\underset{\sim}{v}$ under metric ℓ_K
4. Find λ such that $\min_{0 \leq \lambda \leq 1} P[(1-\lambda)\underset{\sim}{f}^n + \lambda \underset{\sim}{v}^n]$
5. $\underset{\sim}{f}^{n+1} = FD(\underset{\sim}{v}^n, \lambda^n) \circledcirc \underset{\sim}{f}^n$

6. If $\{P(\underset{\sim}{f}^{n}) - P(\underset{\sim}{f}^{n+1})\} < \varepsilon$ stop where ε is an acceptable positive tolerance. Otherwise, let $n = n+1$ and go to 3.

The algorithm converges to stationary points; however, the stationary points of stable equilibrium are the local minima; therefore, we can assume that the algorithm converges to local minima. In the case of $P(\underset{\sim}{f})$ strictly convex, the algorithm clearly converges to the global minimum.

In order to find a feasible starting flow, i.e., $\underset{\sim}{f}^{0} \in \underset{\sim}{F}$, several methods are available [CANT 72], [GERL 73B]. The one that is used here consists of picking any $\underset{\sim}{f} \in \underset{\sim}{F}_a$, and then reducing the flows in all links by a scaling factor ρ until a feasible flow $\underset{\sim}{f}^{0} = \rho \underset{\sim}{f} \in \underset{\sim}{F}$ is obtained. $\underset{\sim}{f}^{0}$ satisfies a reduced requirement matrix $R_0 = \rho R$. The FD method is applied using $\underset{\sim}{f}^{0}$ as the starting flow and R_0 as the starting requirement. After each FD iteration, the value of ρ is increased up to a level very close to saturation. The search for a feasible flow is terminated when one of the two following cases occurs: either $\rho \geq 1$ and a feasible flow is found; or the network is saturated, $T(\underset{\sim}{f})$ is minimized and $\rho < 1$. In the latter case, the problem is infeasible (with respect to a given tolerance).

The FD algorithm then consists of two phases, Phase 1 and Phase 2 (see Appendix II). In Phase 1 a feasible flow $\underset{\sim}{f}^{0}$ is found (if it exists), or the problem is declared infeasible. In Phase 2 the optimal routing is obtained.

3.5 The Routing Problem

In the routing problem formulated in section 3.1, the performance function $T^{(K)}(\underset{\sim}{f})$, equation (3.4), is continuous, lower bounded by zero,

strictly convex (separable sum of strictiy convex functions) and differentiable. The feasible set $\underset{\sim}{F} \triangleq \underset{\sim}{F}_a \cap \underset{\sim}{F}_b$ is a convex poiyhedron. Therefore, if the problem is feasible, there is a unique stationary point, which is the giobal minimum. The capacity constraints are inciuded in $T^{(K)}(\underset{\sim}{f})$ as penalties. Then if we can find a feasibie starting flow $\underset{\sim}{f}^0 \varepsilon \underset{\sim}{F}$, the probiem of section 3.1 can be regarded as an unconstrained m.c. flow problem and be solved with the FD method.

Let us check if $T^{(K)}(\underset{\sim}{f})$ satisfies the conditions for convergence. It is equivaient to check $[T^{(K)}(\underset{\sim}{f})]^K$ instead.

$$[T^{(K)}(\underset{\sim}{f})]^K = \sum_{i=1}^{M} \frac{\lambda_i}{\gamma} (T_i)^K = \sum_{i=1}^{M} \frac{\mu^{1-K}}{\gamma} \frac{f_i}{(C_i - f_i)^K}$$

The first partial derivitive is

$$\frac{\partial}{\partial f_i} (T^{(K)})^K = \frac{\mu^{1-K}}{\gamma} \left[\frac{C_i + (K-i) f_i}{(C_i - f_i)^{K+1}} \right]$$

for this function to be positive we require that K+i is an even number. Then K is a finite odd number.* The second partial derivite is

$$\frac{\partial (T^{(K)})^K}{\partial f_i \partial f_j} = \begin{cases} 0 & i \neq j \\ \frac{\mu^{1-K}}{\gamma} \left[\frac{2KC_i + f_i(K^2-K)}{(C_i - f_i)^{K+2}} \right] < \infty & i = j \end{cases}$$

*When the first partial derivitive is positive, then the shortest route aigorithm is guaranteed to work since the weights $\left(\frac{\partial (T^{(K)})^K}{\partial f_i} \right)$ of the iinks are positive.

Therefore the sufficient conditions or the first two derivitives on the performance function are satisfied. The FD algorithm converges to a global minimum.

3.6 Applications: Deterministic Routing of the AUTODIN Network

The application of the FD problem was investigated for the 8-node European AUTODIN network, Fig. 2.3. The communication link capacity is equal for every link and equal to 2400 bits/sec. The traffic requirement is assumed Poisson starting from a minimum of 65 bits/sec and increased until saturation. The saturation occured at about 676 bit/sec. Several values of K were investigated. In Fig. 3.4 the average delay per message is plotted versus the input requirement. The values of K were taken K = 1,3,5,11. After K = 5 the results were practically the same.

It is of interest that when K is increased, the flow in the links tends to be spread uniformly to all links which results in lower average message delay. It is also true that the more equalized the flow in the links the fastest the network on the average goes into saturation. See Fig. 3.4.

3.7 Evaluation of the Adaptive Routing Algorithm

The performance of the adaptive routing algorithm was evaluated by comparing it with the performance of the deterministic routing algorithm.

Adaptive routing is applied to operational networks while deterministic routing is used in the design of networks according to their input requirements. From simulation data, it has been

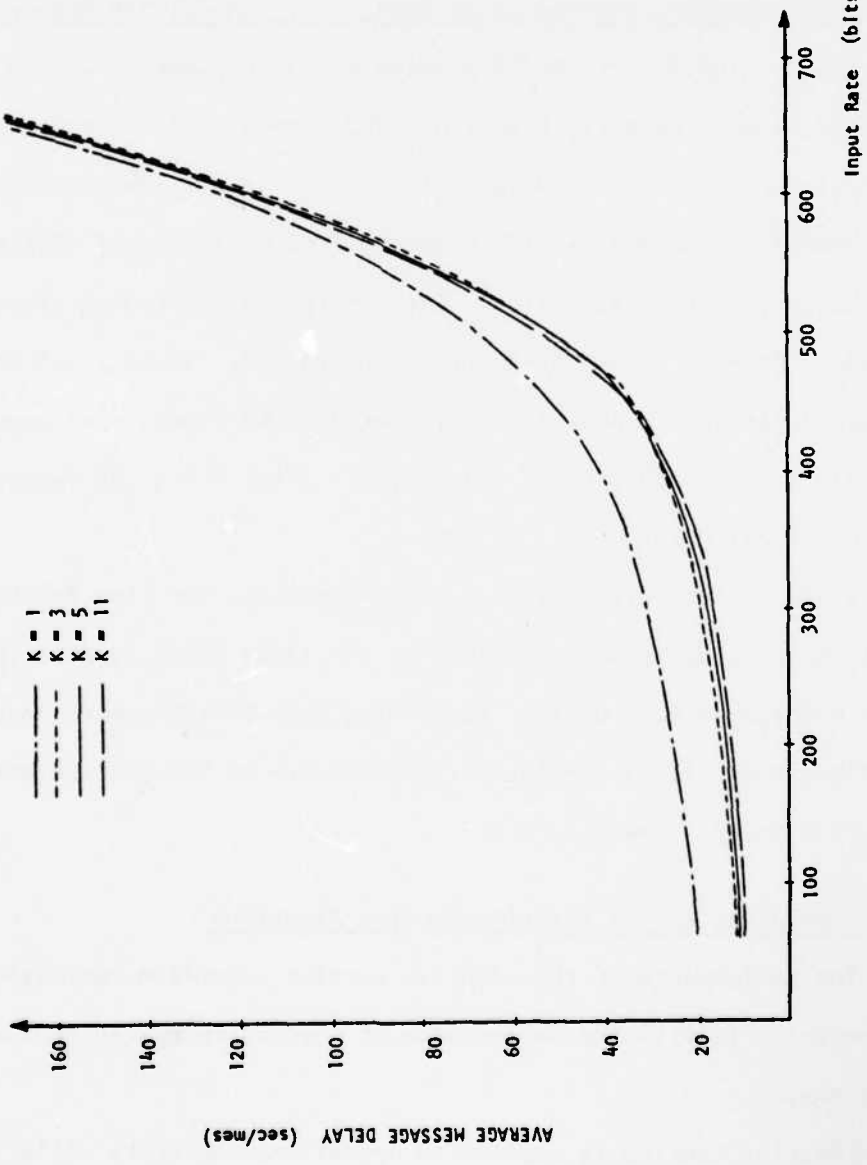


Figure 3.3 Average message delay vs. Input rate for K = 1, 3, 5, 11.

observed that in an operational network it is important that the adaptive routing algorithm distributes the traffic to the link queues in such a way as to keep them short and well balanced. This results in a reduced average message delay. In the deterministic routing case, the above mentioned effect is observed as we increase the value of K in the performance function $T^{(K)}$. For this reason, the comparison of the two algorithms is done for K = 11 (deterministic case) as this represents the case closest to the function of the adaptive algorithm. When K > 11 the resulting average message delay obtained was practically the same as for K = 11. Therefore, for the network under consideration and its input requirements, K = 11 is considered the highest value that K takes.

In Fig. 3.5 the average message delay was plotted versus the input requirements for both the deterministic and adaptive routing algorithms. In every case, the input requirements were assumed to follow a Poisson distribution with parameter equal to the input requirement. In the adaptive routing for each different input requirement the factors of DELTA and TRSLD were chosen to be those which resulted in the least average message delay. These values were DELTA = 10 and TRSLD = 12. The adaptive algorithm simulationwas run for one half hour simulation time in every case.

From the plotted results we observe three regions of interest: Light, medium, and heavy input traffic requirements.

It is generally expected that the deterministic routing should result in a lower average message delay than the adaptive routing since it is an optimal method. The adaptive routing is suboptimal and

in addition the update information data, which do not exist in the deterministic routing, further increase the average message delay.

In light input traffic conditions, the delay in the adaptive case is higher, as expected, because even if a few routing decisions were not optimal, they affected significantly the total average delay per message since there are relatively a few messages in the system to begin with.

In medium input traffic conditions, that most networks usually operate upon, it is encouraging to observe that adaptive routing resulted in a lower average message delay. The adaptive routing decisions are near optimum in this case. In the deterministic routing case, we essentially look at the traffic situation of the network at a single point in time and find the mathematically precise optimal routing. In the adaptive routing case we observe the average message delay for a period of time. Then on the average the adaptive routing shows a better performance over a long period of time, for the same input traffic conditions.

For heavy input requirements, the system was driven into saturation faster with the adaptive routing than in the deterministic. The average message delay increases exponentially. In this case, the nonoptimality of the adaptive decisions is pronounced. Generally heavy traffic conditions drive operational networks into saturation very fast.

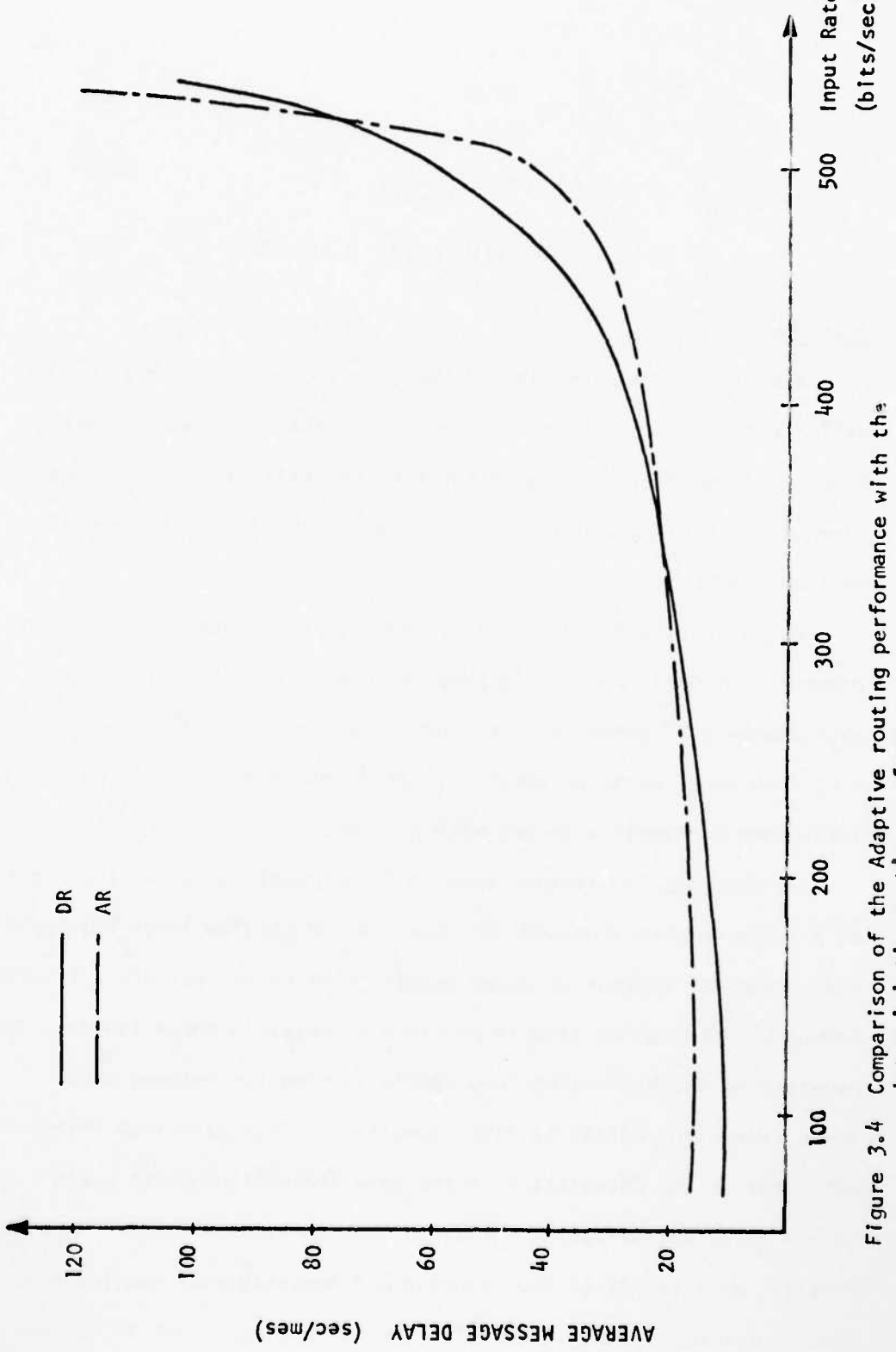


Figure 3.4 Comparison of the Adaptive routing performance with the deterministic routing performance.

CHAPTER 4

PARTITIONING OF A NETWORK

Introduction

Given a general, distributed topology store-and-forward communication network, it is generally recognized that it should be partitioned into smaller components for a better efficiency of the solution of the routing problem. This is also true for design considerations [GERL 73B].

As far as the deterministic routing is concerned, the algorithm presented by Geria requires a computational time proportional to approximately N^3 (where N is the number of nodes in the network) and a storage requirement of about N^2. It seems, therefore, to be inadequate to handle problems with $N > 100$.

In the adaptive routing case, a direct application of the algorithm to a large network (hundreds of nodes) would require large storage for every node for storage of delay tables, routing tables, etc. A large amount of computation time is required to maintain those tables. The requirement for exchanging long update information between nodes would reduce the effective link capacities. Excessive time delays can occur in the percolation of the data information which would result in an out-of-date information when it reaches its destination. Finally, as a result of high storage and computational requirements placed upon every node, nodes will become more expensive to maintain.

On solution to such a problem could be to partition the original network into subnetworks for which the above mentioned algorithms are still applicable and for which an economy of storage and computation may be achieved.

The idea of partitioning was suggested by Gerla [GERL 73B]. A lot of clustering techniques are found in the pattern recognition literature. Those techniques can apply to network problems when the objects to be partitioned can be represented by a network graph [FU 75]. This idea is used in the proposed algorithm. In the large store-and-forward network design problem [KAMO 76] the network is partitioned in tree structured subnetworks.

In this chapter a partitioning algorithm is developed mostly suited to existing communications network topology.

In section 4.1 a definition of the partition used is stated, along with general observations about the traffic requirements of a partitioned network. Considerations for a partitioning approach are discussed and two partitioning solutions are proposed.

In section 4.2 the hierarchical method approach is presented.

In section 4.3 a clustering method is developed. An algorithm is proposed for the partitioning of communications networks of the distributed type. The main idea in the algorithm is the use of a labeling technique to discover the appropriate regions for partitioning. Two examples are worked out at the end of the section which illustrate the application of the algorithm.

4.1 A partitioning definition

The following definition of partitioning is proposed here.

The set of nodes N is partitioned into clusters $N^{(k)}$ (k=1,...p) such that:

$$N = \bigcup_{k=1,\ldots,p} N^{(k)}$$

$$N^{(k)} \cap N^{(\ell)} = \phi \qquad \text{for } k \neq \ell$$

$$N^{(k)} \neq \phi, \qquad k = 1,2,\ldots,p$$

If node i and node j belong to the same cluster, say $N^{(k)}$, all the traffic, which arrives (or is generated) at node i and is directed to node j, is routed through intermediate nodes $N^{(k)}$. In other words, the traffic between two nodes belonging to the same cluster never leaves the cluster.

According to such a definition, we can associate with each cluster its own subnetwork, and study it independently from the rest of the network. The definition also implies that each subnetwork must be connected. Every subnetwork carries internal traffic (generated, and with destination, inside the cluster) as well as external traffic (traffic from internal to external nodes and vice versa).

When partitioning a large network, consideration to the following important questions must be given [GERL 73B].

1. How can nodes be grouped so that the network can be partioned in such a way that the number of nodes within each partition can be specified.

2. The message traffic of the different partitions should be fairly balanced.
3. The geographical diameter of each partition should be considerably smaller than the average distance between different partitions.
4. For design considerations the geographical diameter of each partition should be selected according to the line cost characteristics.

Two approaches are being proposed for the solution of the partitioning problem.

1. Hierarchical approach
2. Clustering approach

4.2 Hierarchical Method Approach

A hierarchical network is one in which message paths are selected according to a routing policy which is influenced by the specific network topology [FULT 72]. Special topological features associated with general networks can be exploited in order to simplify and increase the operational efficiency of the routing procedures.

The hierarchical structure can be broken into a number of hierarchical levels. The application of this principle permits a proper separation so that each level can be implemented essentially independently and, therefore, can be modified and extended without affecting the other levels. In many cases, this form of partitioning is natural due to the geographical locations of the nodes.

The hierarchical structures proposed assume that the network can be partitioned usually on a geographical basis, into local, regional,

and national subnetworks. In order to make the discussion easier, only networks with regional and national partitions are examined.

The hierarchical structure utilizing this concept of partitioning is described as follows: The network is divided into subnetworks according to the partitioning definition. Nodes in each partition are assigned either central or local nodes. Within any subnetwork any connection is possible. Connections between subnetworks are only allowed between central nodes of each region. Any connection between subnets is possible. Fig. 4.1 shows an example of such a network topology resulting from this imposed hierarchical structure where each subnet contains only one central node.

Each node can be labeled by the two-tuple (R,J) where R is the subnet number and J is the node number within the subnet. At each central node, a list must be maintained of the other central nodes reachable from it. This list of reachable subnets from a given subnet can be transmitted to all nodes in the given subnet.

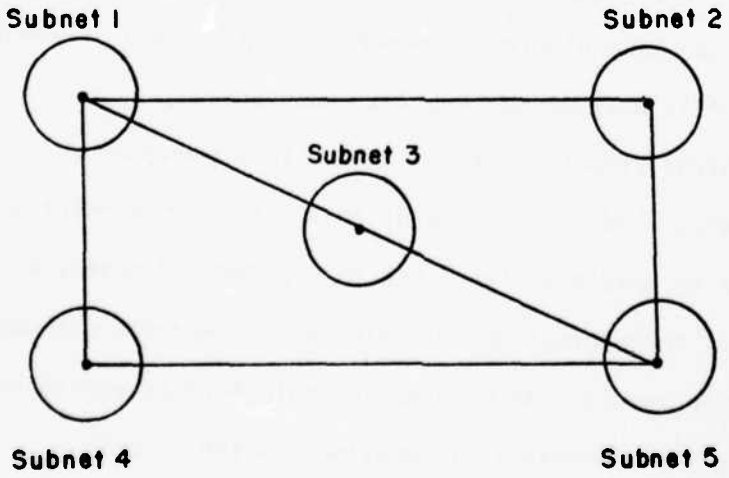


Figure 4.1 A simple hierarchical network.

Unfortunately, in topological structures with only one central node per subnet, the failure of a central node can disconnect a local node in the subnet from the remainder of the subnets in the network. Then possibly two nodes (or more) must be declared as central nodes. At this point, nice properties of the structure in Fig. 4.1 start to disappear. Fig. 4.2 shows a possible network topology.

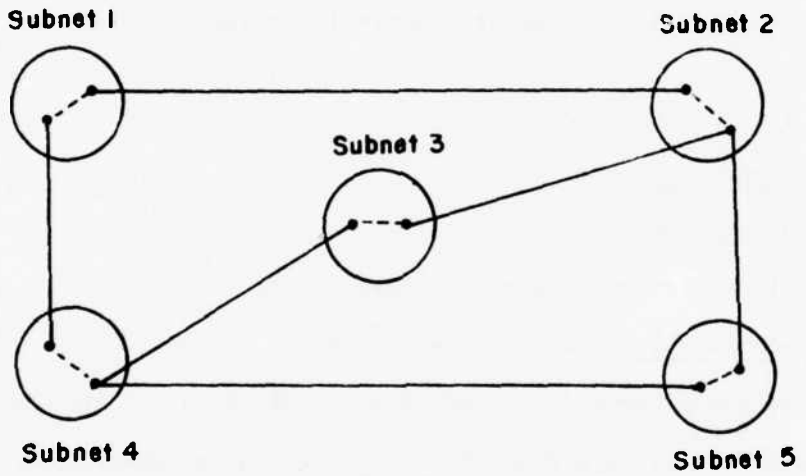


Figure 4.2 A two central node per subnet hierarchical network.

If the central node within each subnet is forced to be directly connected (dashed lines) then routing in the national and regional subnetworks can be achieved independently. However, if a message must be routed from a given subnet to another subnet, there is a choice of routing the message to either of the central nodes.

Since the regional networks would not have information as to which central node is the best choice, larger message delays can occur. For example, if a message must go from subnet 4 to subnet 1 and the link between the two central nodes has failed, then much larger message delay results if it enters the national network from

the right hand node than if it enters by the left hand side control node.

In addition, a failure of the direct link between central nodes would force messages destined from one subnet to another to go through local nodes which would result in additional delay and more complicated routing strategy.

In order to assess the difficulties at the hierarchical networks described, their topologies must be studied in the network design problem.

4.3 Clustering Method Approach

A graph theory approach

The network is represented as a connected graph.

Definition: A node cut-set of a connected graph $G = (V,\Gamma)$ is a nonempty set of nodes whose removal (including all links incident to nodes in the set) from G results in a disconnected or trivial subgraph. An S-T node cut-set is a nonempty set of nodes (not containing nodes in S and T) whose removal from G interrupts all paths between two subsets of nodes S and T.

A link cut-set of a connected graph can be defined similarly.

A node cut-set is also called an articulation set. "Node cut-set" and "articulation set" will be used interchangeably.

From observation of distributed network graphs, it is evident that as a consequence of poor connectedness, there exist articulation sets that would help partition the network.

Then the problem is to detect small articulation sets which would reveal reasonably large subnets in a parent net. No restriction is

made to finding only one single minimal articulation set of a large net, since all reasonably large subnets are desirable.

Efficient methods for finding minimal node cut-sets or link cut-sets are available in [FORD 56], [FULK 62] but they don't solve the general problem.

An algorithm containing a generalized version of Frisch's algorithm [FRIS 67], [Fu 75], has been developed. It sequentially results in subnets from the original net. The conditions for desirable size subnets can be specified by a splitting criterion.

Some definitions should be given at this point relevant to the connectivity of a network. Some important theorems are also stated.

Definition: The connectivity of a node in a graph G is the number of links incident to the node. The minimum connectivity $\delta(G)$ of G is the smallest of the connectivities of nodes in G.

Definition: A node is star connected when it is connected to at least one node of connectivity one.

Definition: The connection number $\omega(G)$ (or the node connectivity) of a graph G is the number of nodes in a minimal node cut-set. The cohesion number $\chi(G)$ of a graph G is the number of links in a minimal edge cut-set.

In a graph, the connection number $\omega(G)$, the cohesion number $\chi(G)$ and the minimum connectivity $\delta(G)$ are related by an inequality due to Whitney [WHIT 32].

Theorem 4.3.1: (Whitney). For any graph G, the following inequality holds:

$$\omega(G) \leq \chi(G) \leq \delta(G)$$

Definition: if h is an integer ≥ 1, G is said to be h-connected if $\omega(G) \geq h$, and G is said to be h-coherent if $\chi(G) \geq h$.

Theorem 4.3.2: If a graph G is h-connected or h-coherent, where h is some positive integer, then the connectivity of every node in the graph G must be greater than or equal to h.

For the proof of theorems 4.3.1 and 4.3.2 refer to [WHIT 32].

Theorem 4.3.3: The necessary and sufficient condition for a graph $G = (V,\Gamma)$ to be h-connected is that two arbitrary distinct nodes v_i and v_j, $v_i, v_j \in V$, can be joined by h paths which have no nodes (links) in common except the nodes v_i and v_j. The proof of the theorem is based on Ford and Fulkerson's labeling algorithm. For a detailed proof, refer to [BERG 62].

The last theorem does not give a direct tool to verify a graph being h-connected. To show that a graph consisting of N nodes is h-connected, where h is an integer, $1 \leq h < N$, one has to check all $\binom{N}{2}$ combinations of pairs of nodes. When N becomes large, this approach is obviously impractical.

A different approach is developed using the h-connectedness concept. In order to gain an understanding as to how this concept is used, three examples will be presented demonstrating three different situations. The size of the networks in the examples are kept small for demonstration purposes.

In Fig. 4.3(a) a network (represented as a graph) contains ten nodes, each of which is connected to exactly three other nodes. The network is 1-connected since the minimal node cut-set contains one node. The articulation set here is the minimal one. Removal of node a or b will separate the net into two subnets.

If a network has a high value of h-connectedness, it is tightly connected. However, if it has a low value of h-connectedness, it is not always loosely connected. For example, in Fig. 4.3(b), the network is 1-connected even though it is rather strongly connected, because the connectivity of the network is spoiled by a few nodes of connectivity 1.

In Fig. 4.3(c) a large network is at most 4-connected, by Theorem 4.3.2 because the connectivity of the network is 4. There exists a node cut-set containing 5 nodes {u, v, w, x, y}. The size of the node cut-set exceeds the connectivity of the network, but it is very small compared with the size of the network.

The removal of this node cut-set breaks the parent network into two subnetworks of almost equal size. Then it is necessary to find an S-T node cut-set between two certain subsets of nodes S and T whose sizes are larger than the maximum degree of the network. In this case, the minimal node cut-set is not meaningful because it fails to reveal the interesting subnetworks.

4.3.1 Partitioning of a Network

The subnetworks hidden in a large parent network are required to satisfy the following splitting criterion:

(1) The sizes of the articulation sets should be as small as possible.

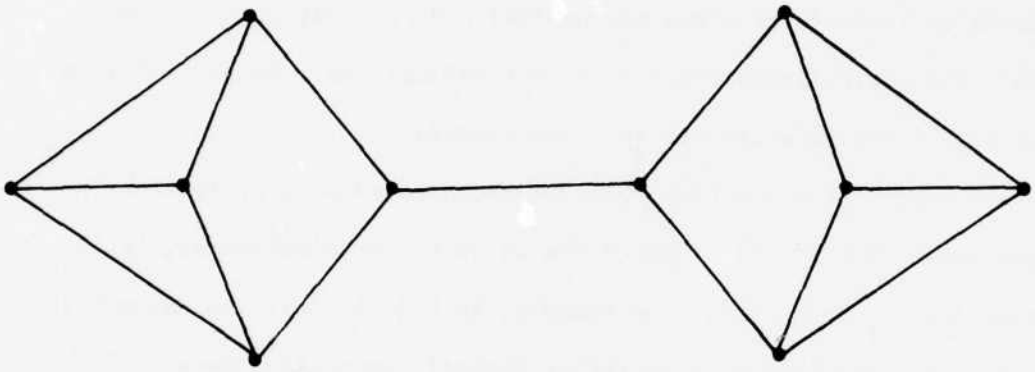


Figure 4.3(a) A loosely 1-connected network.

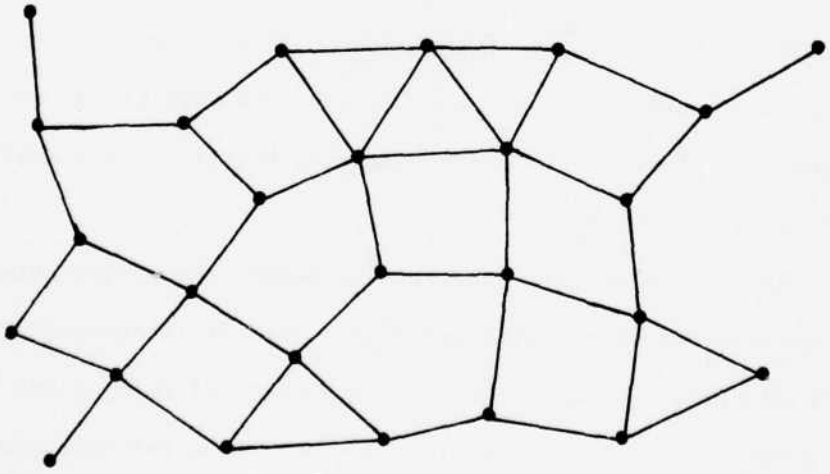
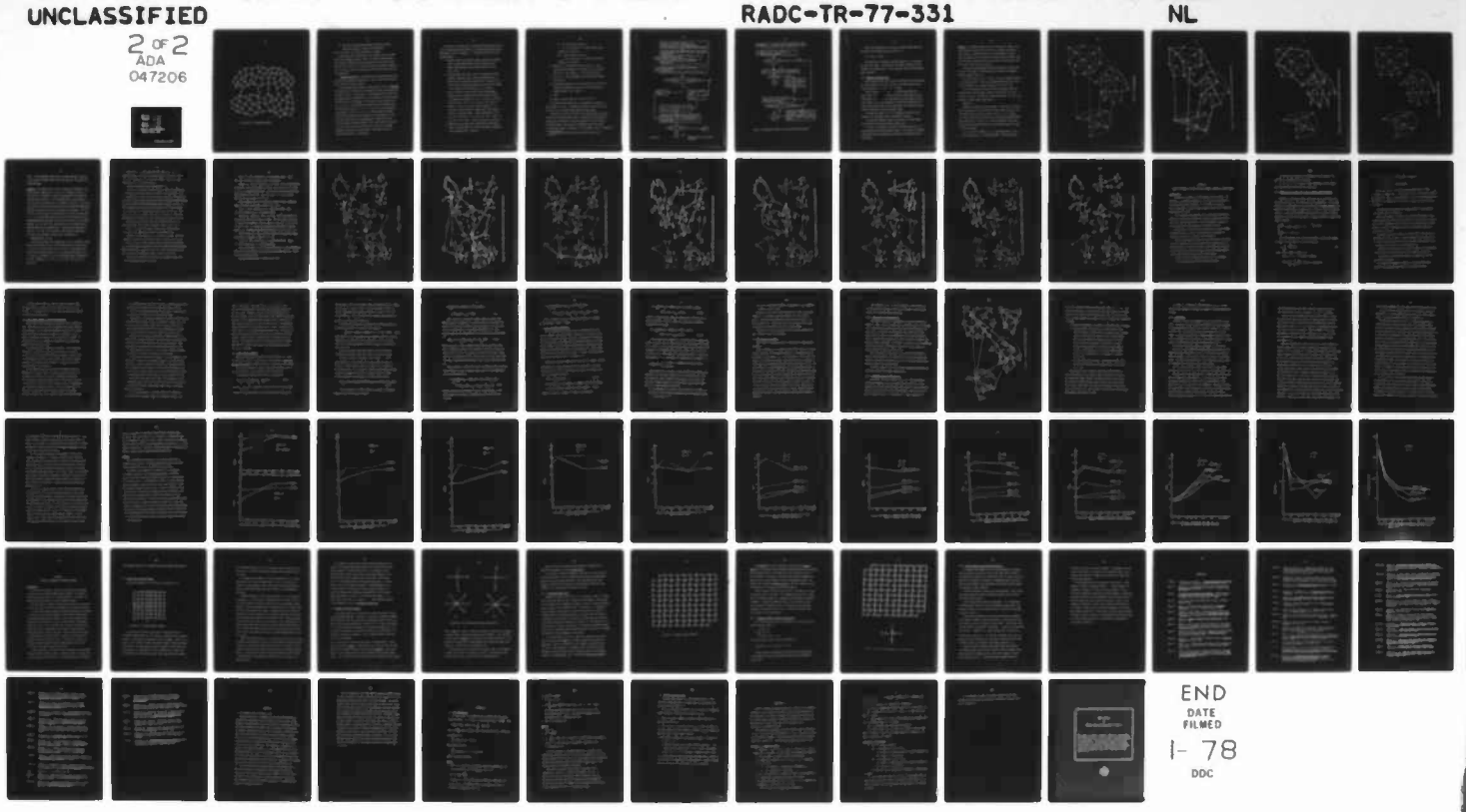


Figure 4.3(b) A tightly 1-connected network.

AD-A047 206 PURDUE UNIV LAFAYETTE IND SCHOOL OF ELECTRICAL ENGI--ETC F/G 17/2
THE ADAPTIVE ROUTING PROBLEM FOR LARGE STORE-AND-FORWARD COMPUT--ETC(U)
OCT 77 C E HOUSTIS, B J LEON F30602-75-C-0082

UNCLASSIFIED RADC-TR-77-331 NL

2 OF 2
ADA
047206



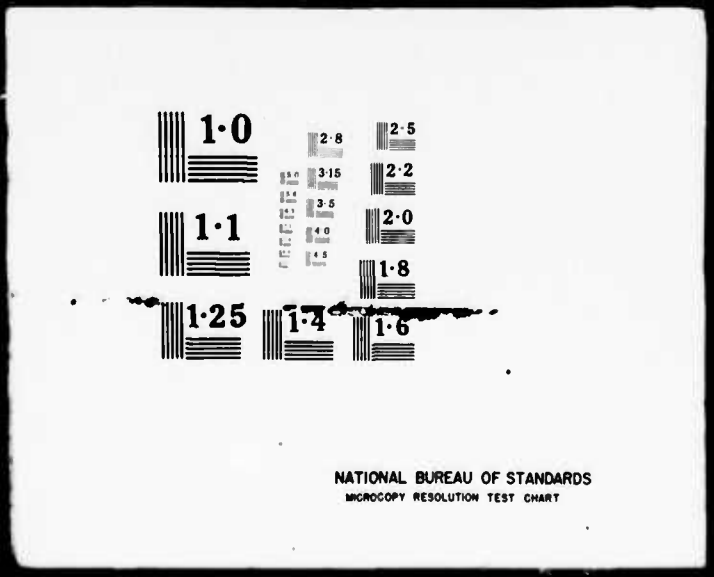
END
DATE
FILMED
1-78
DDC

SSIFIED

2 OF 2

ADA

047206



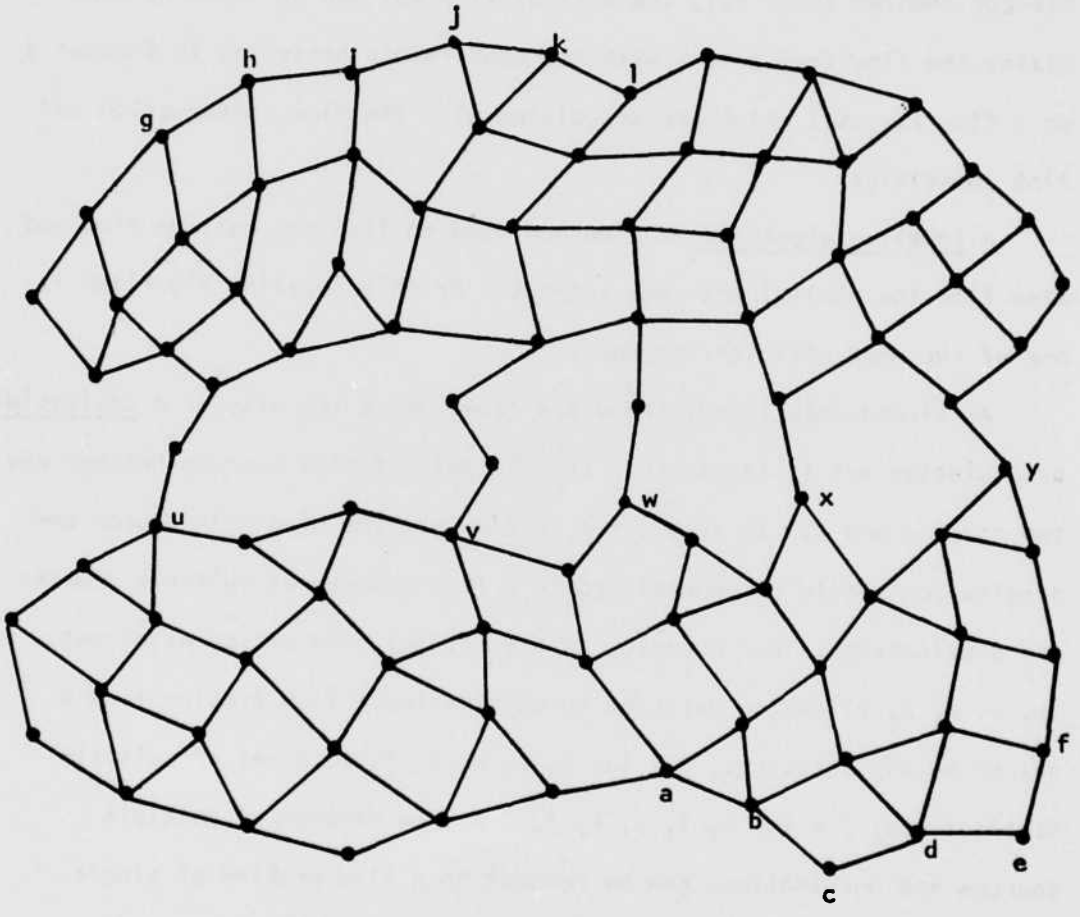


Figure 4.3(c) A 4-connected network.

(2) The sizes of the resulting subnets should be reasonably large and connected in order to be subnetworks.

Any trivial graph will not be considered as a subnet.

In the case where an articulation set is equal to an s-t minimal node cut-set between two nodes s and t, by the well-known max-flow, min-cut theorem [FORD 62], the articulation set can be found by maximizing the flow from s to t with the constraints presented in Chapter 3 on a flow $f(v_i, v_j)$ which are associated with the flow conservation and link capacities.

A labeling algorithm is a method used to find the maximum flow and also find the minimal s-t node cut-set. Frish's labeling algorithm is one of the most efficient methods.

As already mentioned, there are cases where the size of a desirable articulation set is larger than any minimal s-t node cut-set between any two nodes s and t. In such cases, a flow problem of single source and destination should be generalized to a flow problem of multiple sources and destinations. For example, in Fig. 4.3(c), the articulation set {u, v, w, x, y} can be detected by considering a flow problem from a set of multiple sources, S = {a, b, c, d, e, f} to a set of multiple destinations; T = {g, h, i, j, k, l}. A flow problem of multiple sources and destinations can be reduced to a flow problem of single source and destination in the following way: Replace the set of multiple sources (including removal of all links incident to the set) by an artificial source, and connect the new source to all nodes adjacent to the set of multiple sources. Then treat the set of mutliple destinations in the same way.

An algorithm is proposed here, containing a generalized version of Frisch's labeling algorithm [CHEN 73]. The generalized labeling algorithm (GFLA) is given in Appendix III. The algorithm consists of the following steps.

1. Specify a splitting criterion; namely, specify the reference size m of a desirable subnet, and the reference size k. Where $k \leq \frac{m}{2}$ represents the allowable deviation from m. Go to step 2.
2. From the graph consisting of the set of nodes under investigation, choose two small connected subgraphs which are about as far as possible from each other. Consider the two sets of nodes in these two subgraphs as the multiple sources, S, and the multiple destinations, T. Go to step 3.
3. Apply the GFLA to the set of nodes under investigation. Obtain the set of labelled and unlabelled nodes, and the articulation set, denoted as L, U, A. Let $|L|$, $|U|$ and $|A|$ be their sizes. List L, U, A, $|L|$, $|U|$. Go to step 4.
4. Examine if there are any star connected nodes $\in A$. If there are, let this set be M. When $m_i \in M$ is also $m_i = s_i \in S$ create $N = \{n_i : n_i \in M, n_i \notin S\}$. Store N and define $L = L-N$, $|L| = |L| - |N|$ and $U = U + N$, $|U| = |U| + |N|$. List L, U, $|L|$, $|U|$. Go to step 5. If there are not, go to step 5.
5. Break the set of nodes under investigation into two sets L and U by removing the set of links connecting a labelled and an unlabelled node. Examine

(a) If $|L| \leq m+k$ go to step 7.

If $|L| > m+k$ store L. L needs further investigation. Go to step 2.

(b) If $|U| \leq m+k$ go to step 7.

If $|U| > m+k$ store U. U needs further investigation. Go to step 6.

6. Examine N. If $N \neq \emptyset$ then let $N \subseteq S$ or $N \subseteq T$. Find a small connected subgraph T or S about as far as possible from S or T. Go to step 3. If $N = \emptyset$ go to step 2.
7. List L or U. These subgraphs are the legitimate subnets. Go to step 8.
8. Combine each subgraph of size less than m-k with one of its neighboring subgraphs in such a way as to produce a new possible subnet. Go to step 9.
9. Stop.

A flowchart of the algorithm is given in Fig. 4.4.

The algorithm given above will yield sequentially subnets in a finite number of steps. It preserves all nodes in the original network and all links between nodes within the same subnet.

The GFLA will find all distinct[++] flow paths between the source nodes and the terminal nodes, and will terminate when the terminal nodes cannot be labelled. At the termination of the GFLA the set of labelled nodes L and the set of unlabelled nodes U are found. L is not empty since S is labelled. U is not empty since T is unlabelled.

[++] Two paths are distinct when they have no node or link in common.

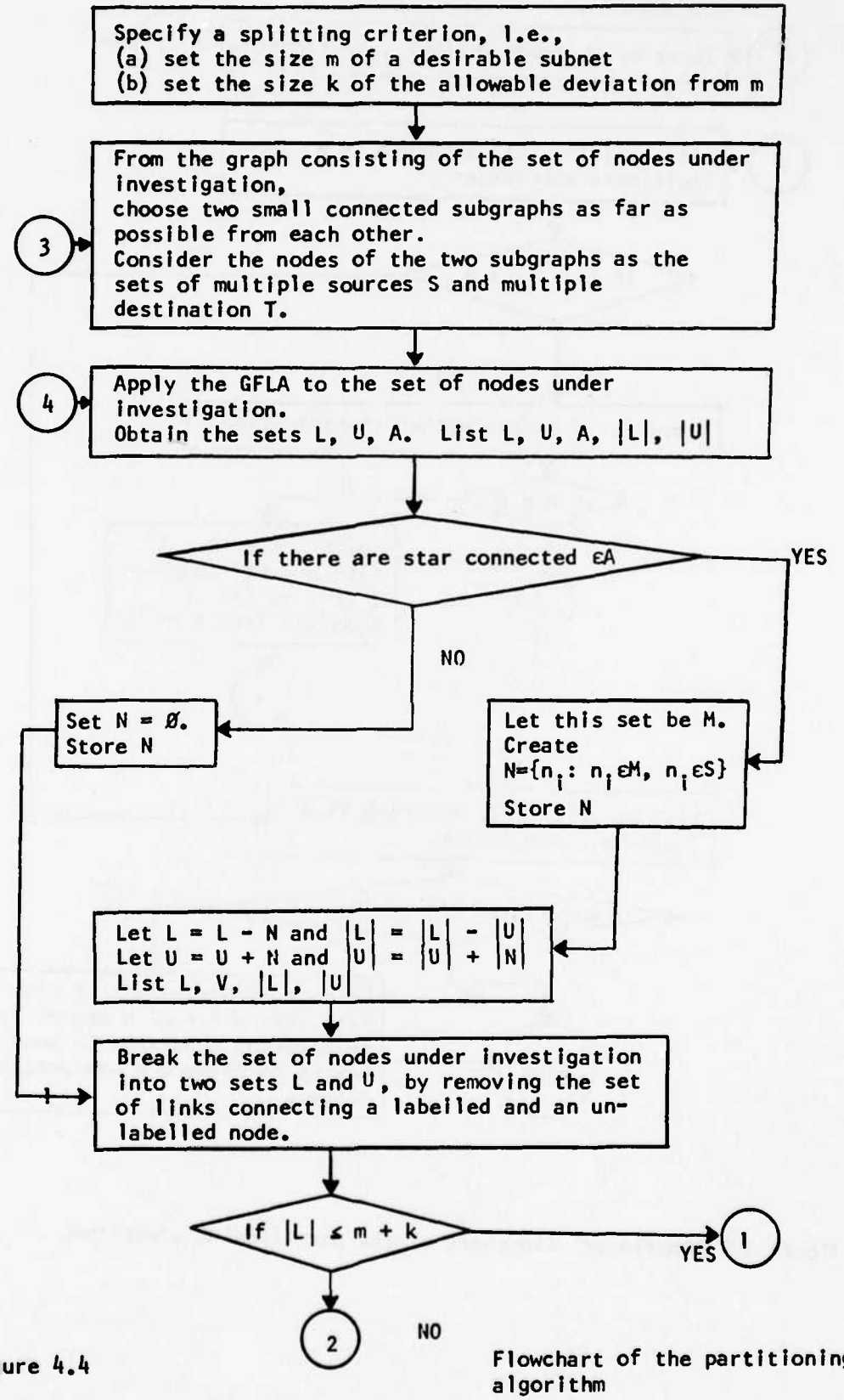


Figure 4.4 Flowchart of the partitioning algorithm

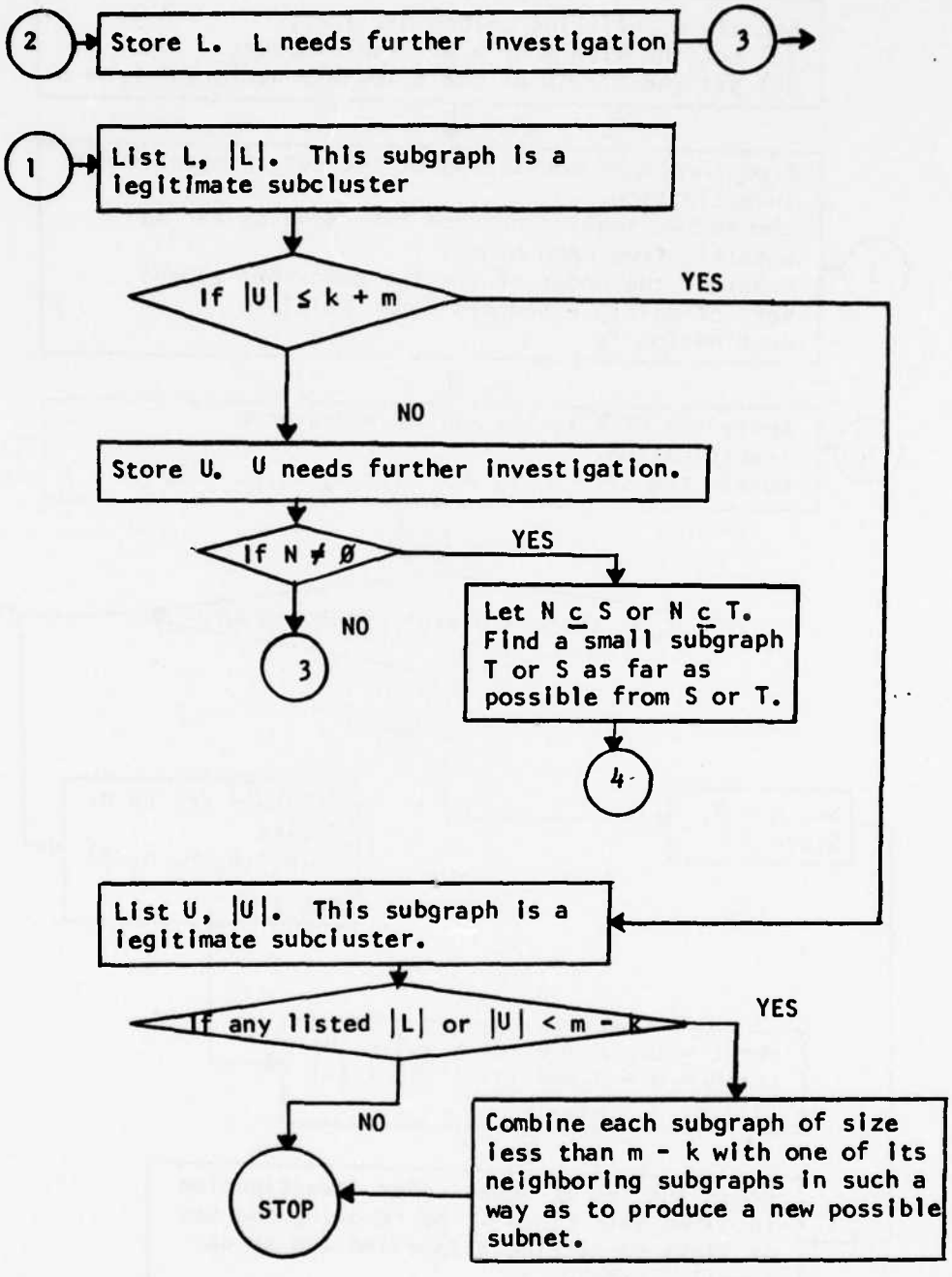


Figure 4.4 (Continued) Flowchart of the partitioning algorithm.

Then the articulation set is the set of labelled nodes which are connected to an unlabelled node by a link, i.e.

$$A = \{x:(x,y) \in (L,U)\}.$$

The labelling in the GFLA is a one end labelling. It uses both plus and minus labels. The scanning of the nodes is done on a first-labelled first-scanned basis. It also applies directly to the original graph (network).

4.3.2 Remarks on the Algorithm

The articulation set is not unique. The subnets that the algorithm will define depend on the choice of the set of source and the set of terminal nodes every time the GFLA is applied.

Usually nodes with high connectivity make a good choice of source nodes. After source nodes are chosen, a good choice for terminal nodes is still high connected nodes as far as possible from the source nodes. Star connected nodes are usually high connected also. Then star connected nodes fulfilling the above requirements are satisfactory selection for source at terminal nodes.

The number of nodes in S or T will depend on the size of the network to be partitioned and its topological structure.

For large size networks usually more source or terminal nodes are required for fast partitioning to begin with.

For small site networks the minimal criterion works well, i.e. only one source and only one terminal node. In this case, the minimal articulation set will represent the strength of the weakest part of the network.

Example 1: The network shown in Fig. 4.5 consists of 31 nodes. Let the splitting criterion be m = 10 and k = 3. Choose as the set of source nodes S = {3,8} and the set of terminal nodes T = {30,31}. After the application of the GFLA four distinct paths are found. They are indicated by an arrow (Fig. 4.6), pointing in the direction of the flow, next to every link in the path.

The set of labelled nodes are shown carrying their label(s) next to their node number. The flow paths, the labelled and unlabelled nodes are shown in Fig. 4.6.

The articulation set is A = {3,4,7}. The set of labelled nodes is L = {1,2,3,4,5,6,7,8}. The set of unlabelled nodes consists of the rest of the nodes in the original network, $|L| = 8$, $|U| = 23$.

The links connecting the nodes ϵA with an unlabelled node are broken and the resulting graph is shown in Fig. 4.7, $|L| = 8 < m+k$. Therefore, the set L constitutes a legitimate subnet. $|U| = 23 > m+k$ therefore U needs further investigation.

Examine U.

Choose as the new set of source nodes S = {9,11} and T = {19,31}. After the application of the GFLA two distinct paths are found. The flow paths the set L and U are shown in Fig. 4.7. $A_1 = \{15,18\}$, $L_1 = \{9,10,11,12,13,14,15,16,17,18\}$, $U_1 = \{19,20,21,22,23,24,25,26,27,28,29,30,31\}$, $|L_1| = 10$, $|U_1| = 13$. After the removal of the links converting the articulation nodes with an unlabelled node, the graph in Fig. 4.8 results.

$|L_1| = 10 < m+k$ therefore, L_1 is a legitimate subnet. $|U_1| = 13 = m+k$ therefore U_1 is a legitimate subnet. The algorithm has

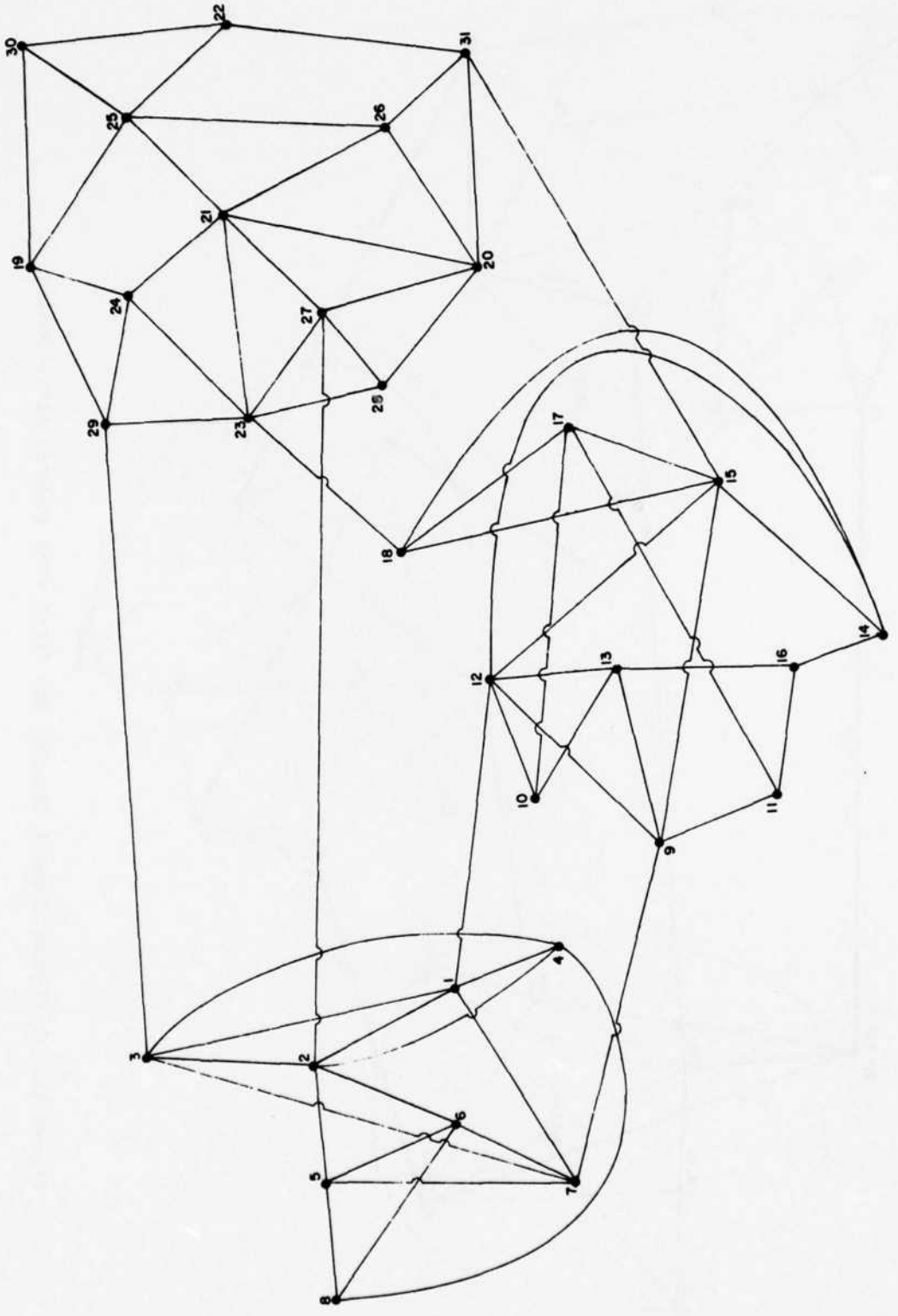


Figure 4.5 A 30-node network.

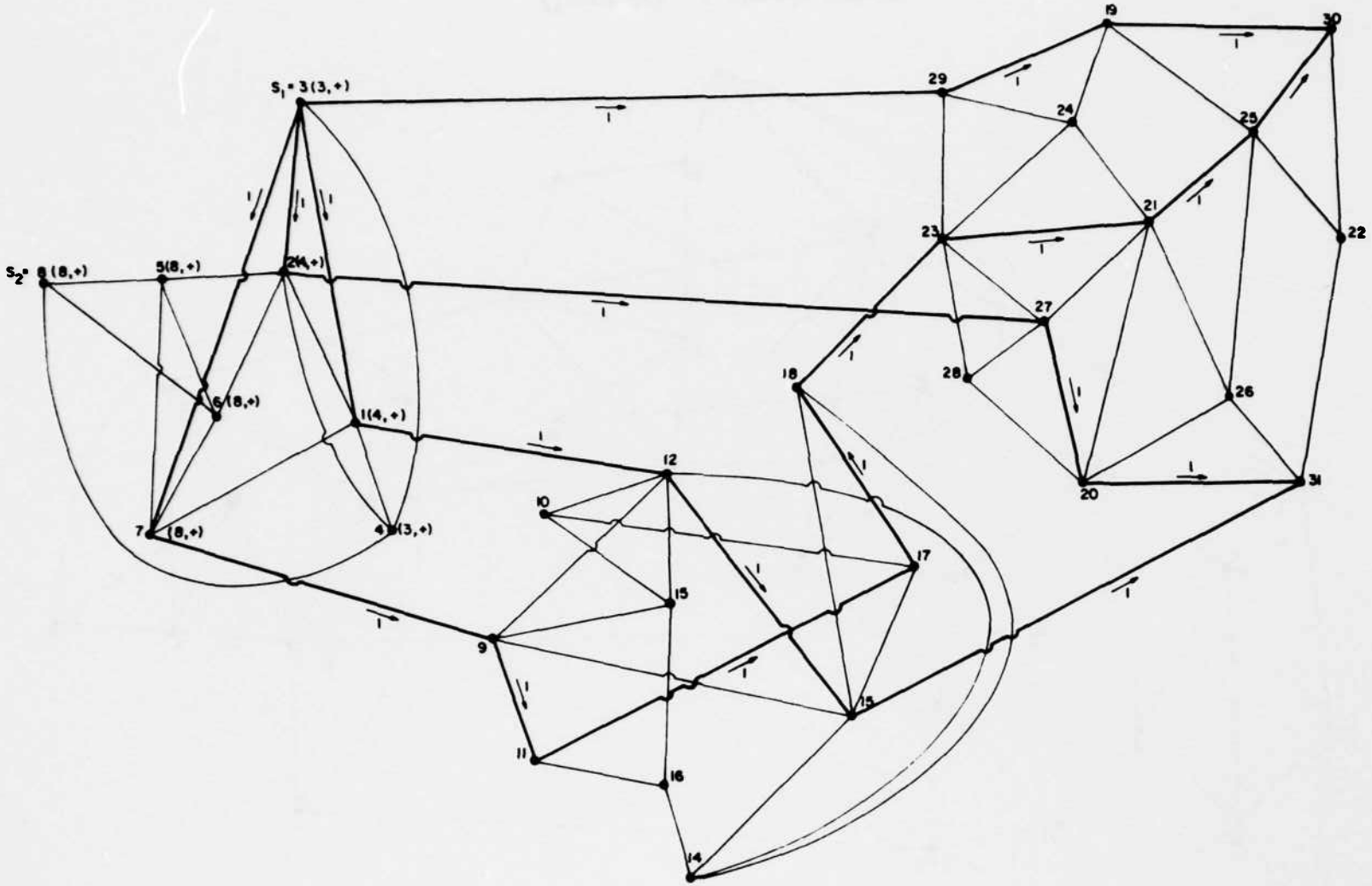


Figure 4.6 A 30-node network showing the first step towards partitioning.

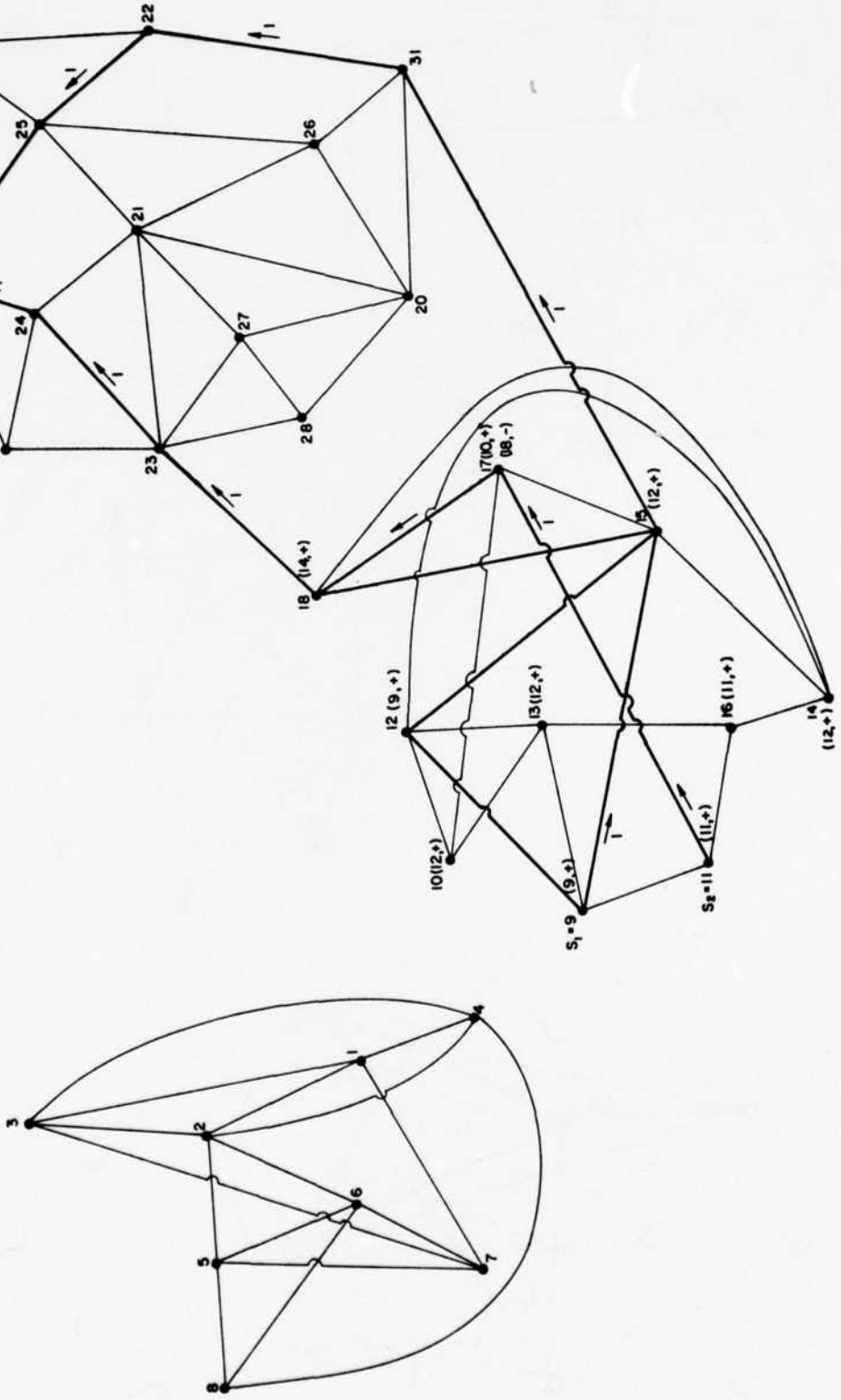


Figure 4.7 A 30-node network showing the second step towards partitioning.

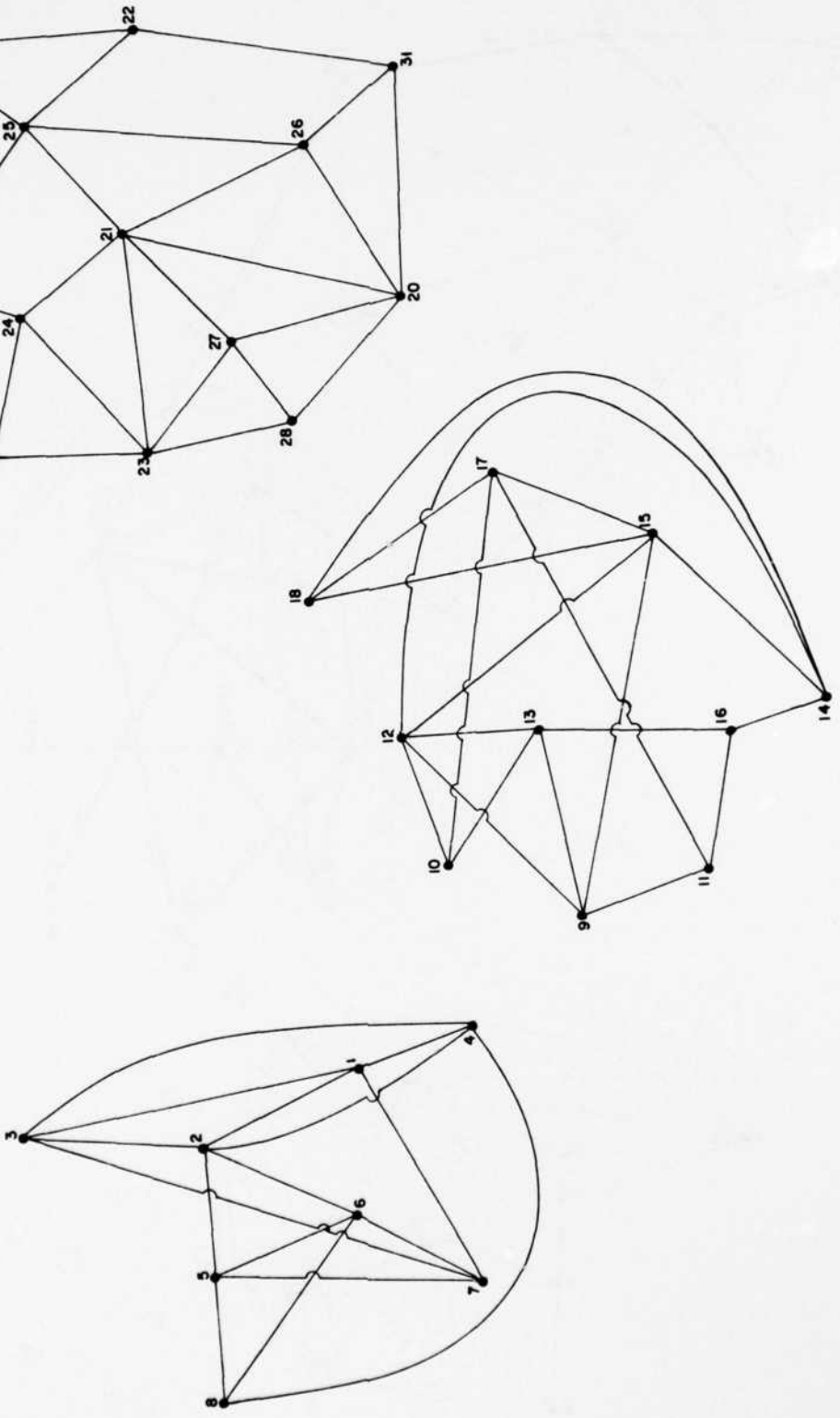


Figure 4.8 A 30-node partitioned network.

yield three connected subnets from the original network. The size of all subnets is greater than m-k and less than m+k. Therefore, the algorithm stops.

Example 2: The network shown in Fig. 4.9 consists of 95 nodes. Let the splitting criterion be m = 12 and k = 6. Choose as the set S = {4,7} and as T = {89,77,94}. After the application of the GFLA four distinct paths are found. The flow paths, the sets L,U, are shown in Fig. 4.10. The articulation set A = {17,30,70}. The set L = {1,2,3,4,5, 6,7,8,9,10,11,12,13,14,15,17,18,19,20,21,22,23,24,25,26,27,28,30,32,70}. The set U consists of the rest of the nodes in the network. |L| = 30, |U| = 65. All the nodes in the articulation set A are star nodes then L = L - A = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,18,19,20,21,22,23,24, 25,26,27,28,32} and U consists of the rest of the nodes in the original network. |L| = 27, |U| = 68. Break the set of links connecting the labelled nodes with an unlabelled one. The resulting graph shows in Fig. 4.11. |L| = 27 > m+k therefore L needs further investigation. |U| = 68 > m+k therefore U needs further investigation.

Examine the set L first.

Let the source nodes for the subgraph L be S_1 = {15} and T_1 = {26}. After the application of the GFLA one distinct path is found, shown in Fig. 4.10. The articulation set A_1 = {4}, L_1 = {4,6,7,8,9,10,11, 12,13,14,15,18,19,20,21,22,23}, U_1 = {1,2,3,5,24,25,26,27,28}, $|L_1|$ = 17, $|U_1|$ = 9. The node in A_1 is a star connected node therefore M = N = {4}. Redefine $L_1 = L_1 - \{4\}$, $U_1 = U_1 + \{4\}$. Then $|L_1| = 16$, $|U_1| = 10$.

Break the set of links connecting the nodes in L_1 with an unlabelled node. The resulting subgraph shows in Fig. 4.12. $|L_1| = 17 < m+k$. Therefore, L_1 is a legitimate subnet. $|U_1| = 10 < m+k$. Therefore, U_1 is a legitimate subnet.

Examine the subgraph whose nodes belong in U.

Let $S_2 = A = \{17,30,70\}$ and $T_2 = \{62,94\}$. After the application of the GFLA, three distinct paths are found. $A_2 = \{30,55,78\}$. Then $M = \{30,78\}$ and $N = \{78\}$. The set of labelled nodes $L_2 = \{16,17,29,30, 31,33,34, 35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55, 70,71,78,79,80,81,82,83,84,85,86,87,88,89,90, 91,92,93\}$, the set $U_2 = \{56,57,58,59,60,61,62,63,64,65,66,67,68,69,72,73,74,75,76,77,94,95\}$. $|L_2| = 46$, $|U_2| = 22$. Since $N \neq \emptyset$, redefine $L_2 = L_2 - N = L_2 - \{78\}$ and $U_2 = U_2 + N = U_2 + \{78\}$. Then $|L_2| = 45$, $|U_2| = 23$. The flow paths found, the labelled and unlabelled nodes are shown in Fig. 4.12.

Break the links connecting the set of labelled nodes with an unlabelled node. The resulting subgraph is shown in Fig. 4.13.

$|L_2| = 45 > m+k$. Therefore L_2 needs further investigation. $|U_2| = 23 > m+k$. Therefore U_2 needs further investigation.

Examine L_2 first. Choose $S_3 = \{17,30\}$ and $T_3 = \{89\}$. After the application of the GFLA two distinct paths are found. $A_3 = \{30,55\}$, $M = \{30\}$, $N = \emptyset$, $L_3 = \{16,17,29,30,31,33,34,35,36,45,46,47,48,49,55\}$. $U_3 = \{37,38,39,40,41,42,43,44,52,53,54,70,71,79,80,81,82,83,84,85,86,87, 88,89,90,91,92,93\}$. $|L_3| = 15$, $|U_3| = 28$. The flow paths found, the set of labelled and unlabelled nodes are shown in Fig. 4.13.

Break the links connecting labelled and unlabelled nodes. The resulting subgraph shown in Fig. 4.14.

$|L_3| = 15 < m+k$, therefore L_3 is a legitimate subnet. $|U_3| = 28 > m+k$, therefore U_3 needs further investigation.

Examine U_2. Let $S_4 = \{58\}$ and $T_4 = N = \{78\}$. After the application of the GFLA, two distinct paths are found. $A_4 = \{61, 62\}$ $M = N = \{62\}$, $L_4 = L_4 - N = \{56,57,58,60,61,95\}$. $U_4 = U_4 + N = \{62,63,64,65,66,67,68, 69,72,73,74,75,76,77,78,94\}$. $|L_4| = 7$, $|U_4| = 16$. The flow paths and the sets L_4, U_4 show in Fig. 4.14.

After breaking the links connecting a labelled and an unlabelled node, the resulting subgraph is shown in Fig. 4.15.

$|L_4| = 7 < m+k$, therefore L_4 is a legitimate subnet. $|U_4| = 16 < m+k$, therefore U_4 is a legitimate subnet.

Examine U_3. Choose $S_5 = \{70\}$, $T_5 = \{89\}$. After the application of the GFLA, two distinct paths are found. The flow paths and the set of labelled and unlabelled nodes are shown in Fig. 4.15. $A_5 = \{70,71\}$, $M = \{70\}$, $N = \emptyset$. $L_5 = \{37,38,31,40,41,42,43,44,52,53,54,70,71,79,80, 82,83\}$, $V_5 = \{81,84,85,86,87,88,89,90,91,92,93\}$, $|L_5| = 17$, $|V_5| = 11$.

After breaking the links connecting labelled and unlabelled nodes, the resulting subgraph is shown in Fig. 4.16.

$|L_5| = 17 < m+k$, therefore L_5 is a legitimate subnet. $|U_5| = 11 < m+k$, therefore U_5 is a legitimate subnet.

The original network is partitioned sequentially in seven subnetworks. The size of these subnets is less than m+k and greater than m-k. Therefore, the algorithm stops.

The partitioned network is shown in Fig. 4.16.

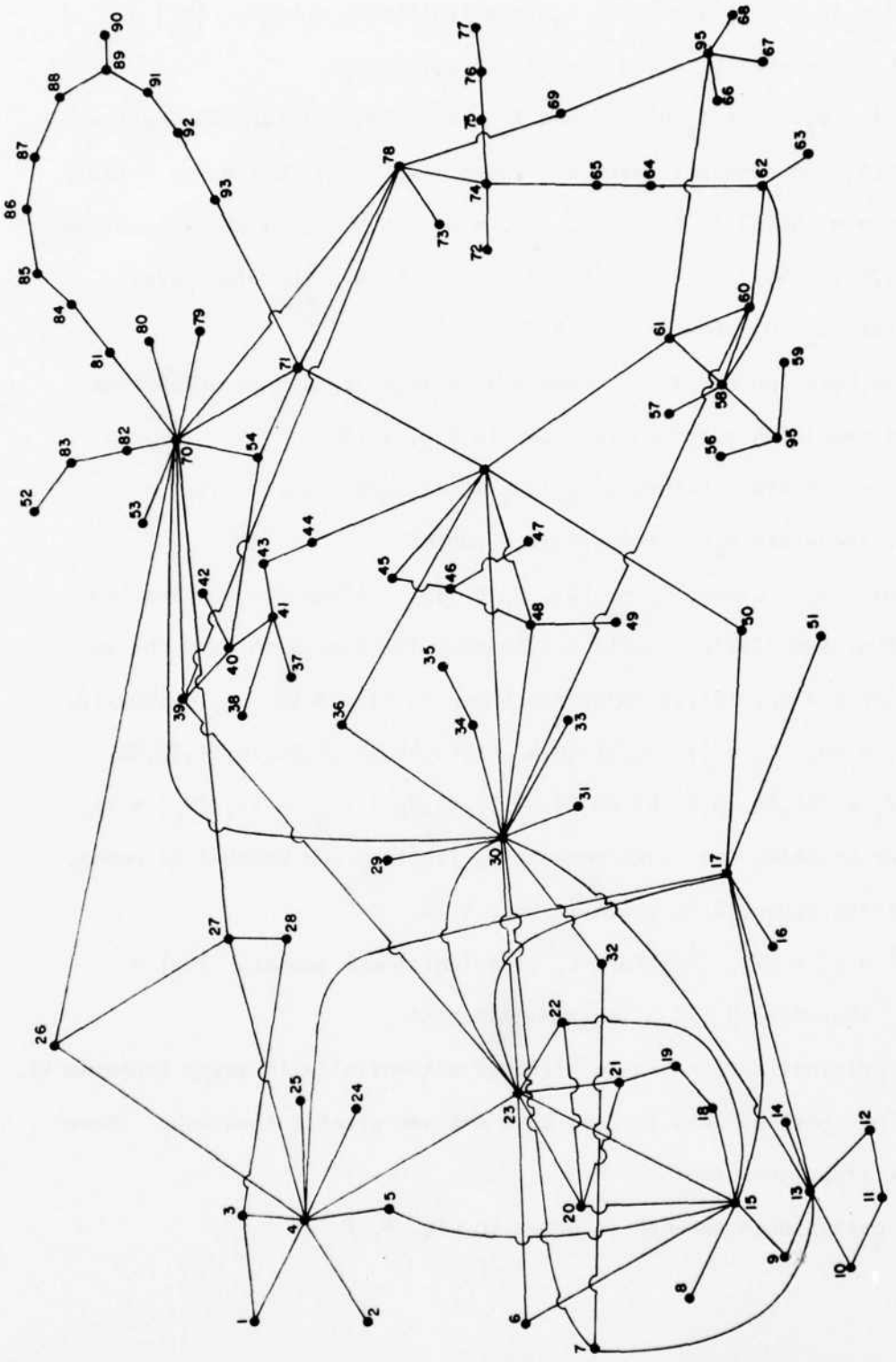


Figure 4.9 A 95-node network.

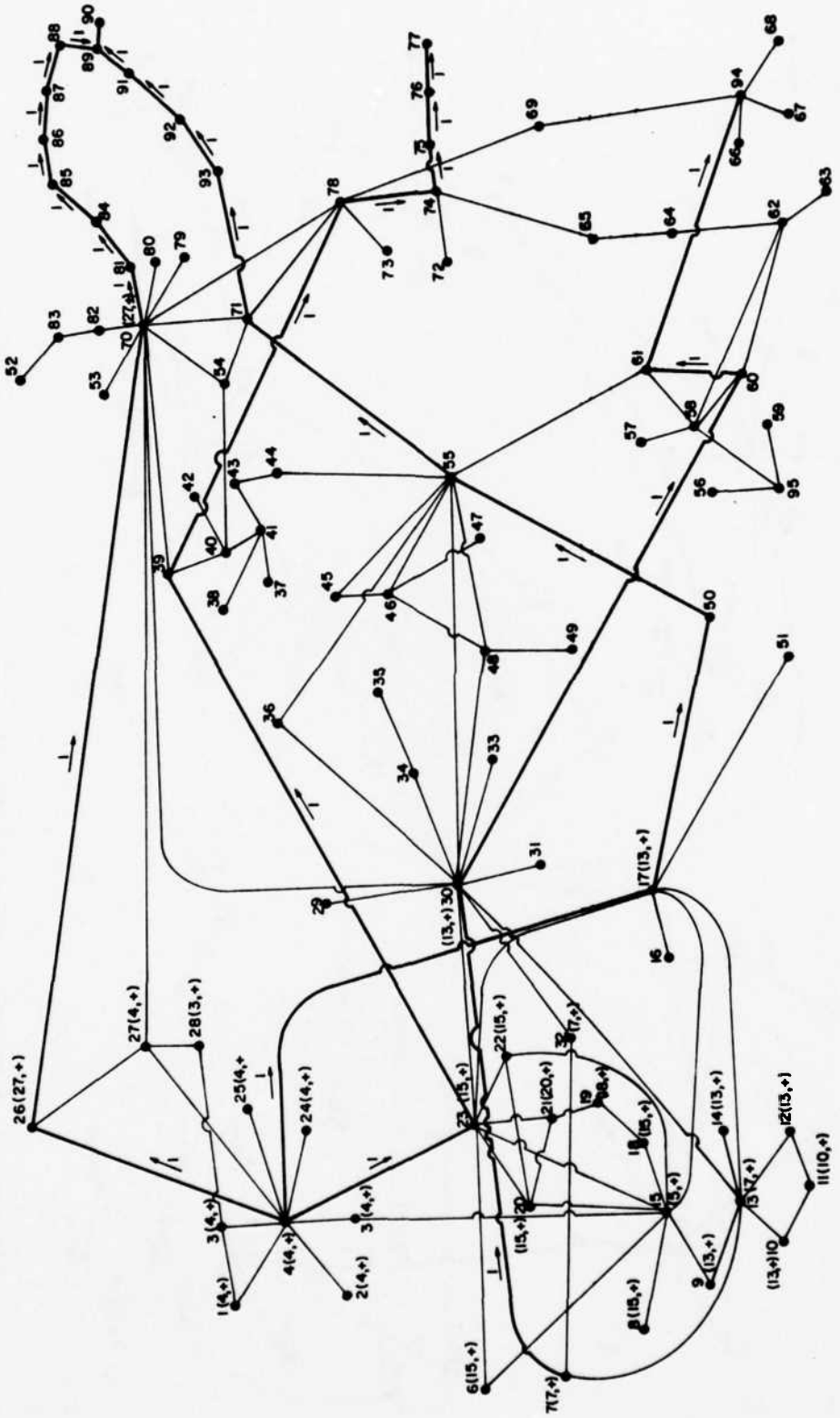


Figure 4.10 A 95-node network showing the first step towards partitioning.

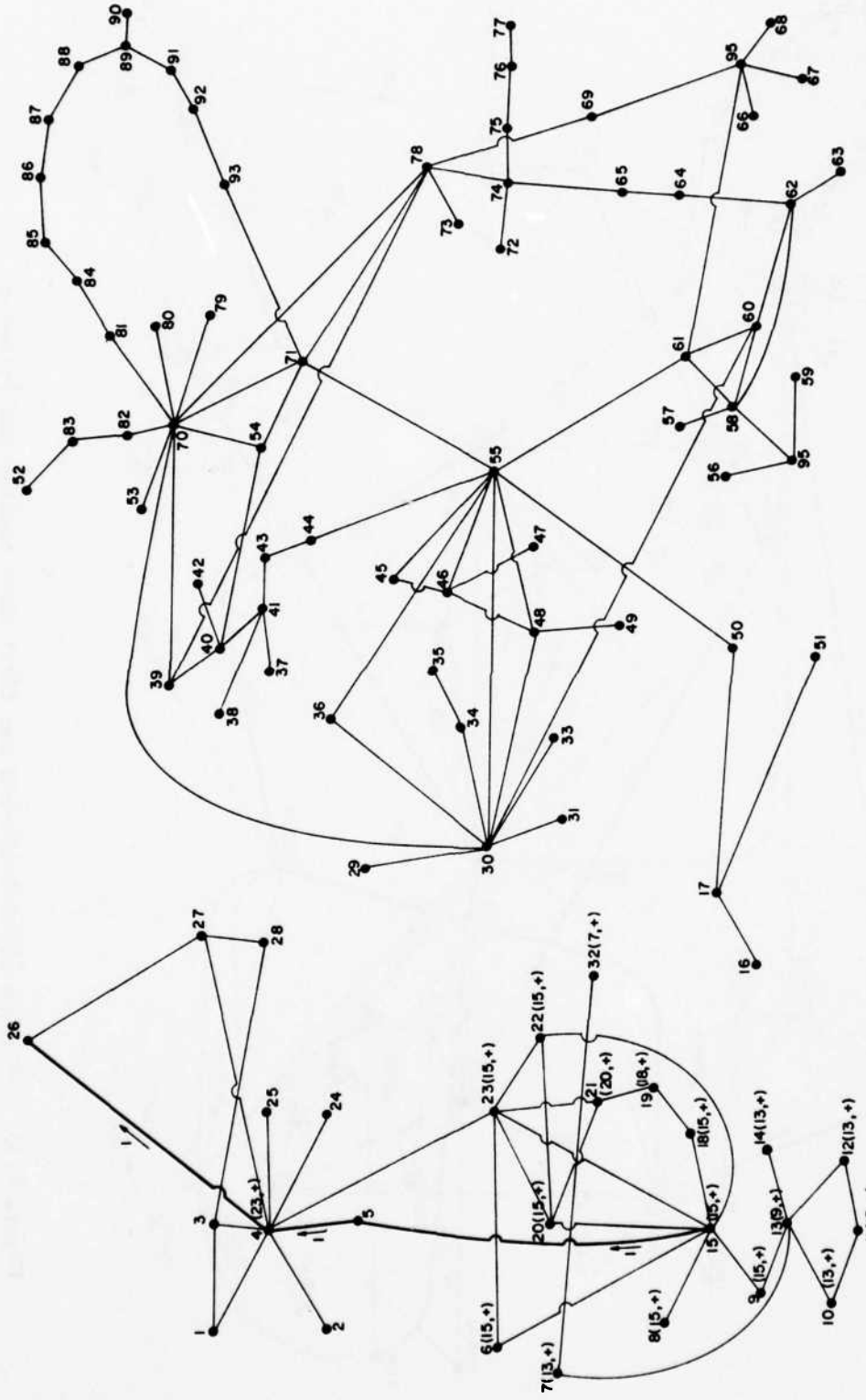


Figure 4.11 A 95-node network showing the second step towards partitioning.

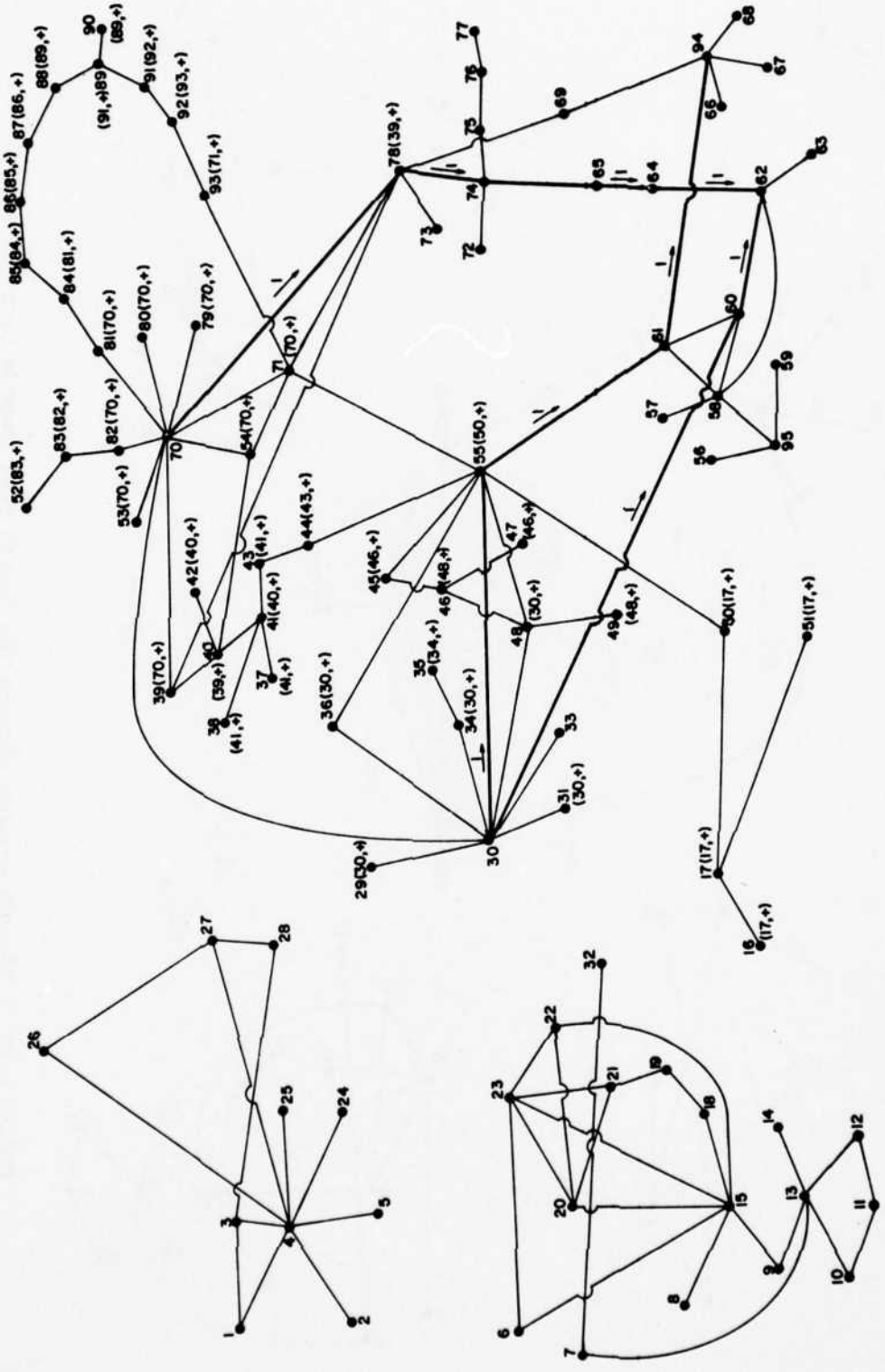


Figure 4.12 A 95-node network showing the third step towards partitioning.

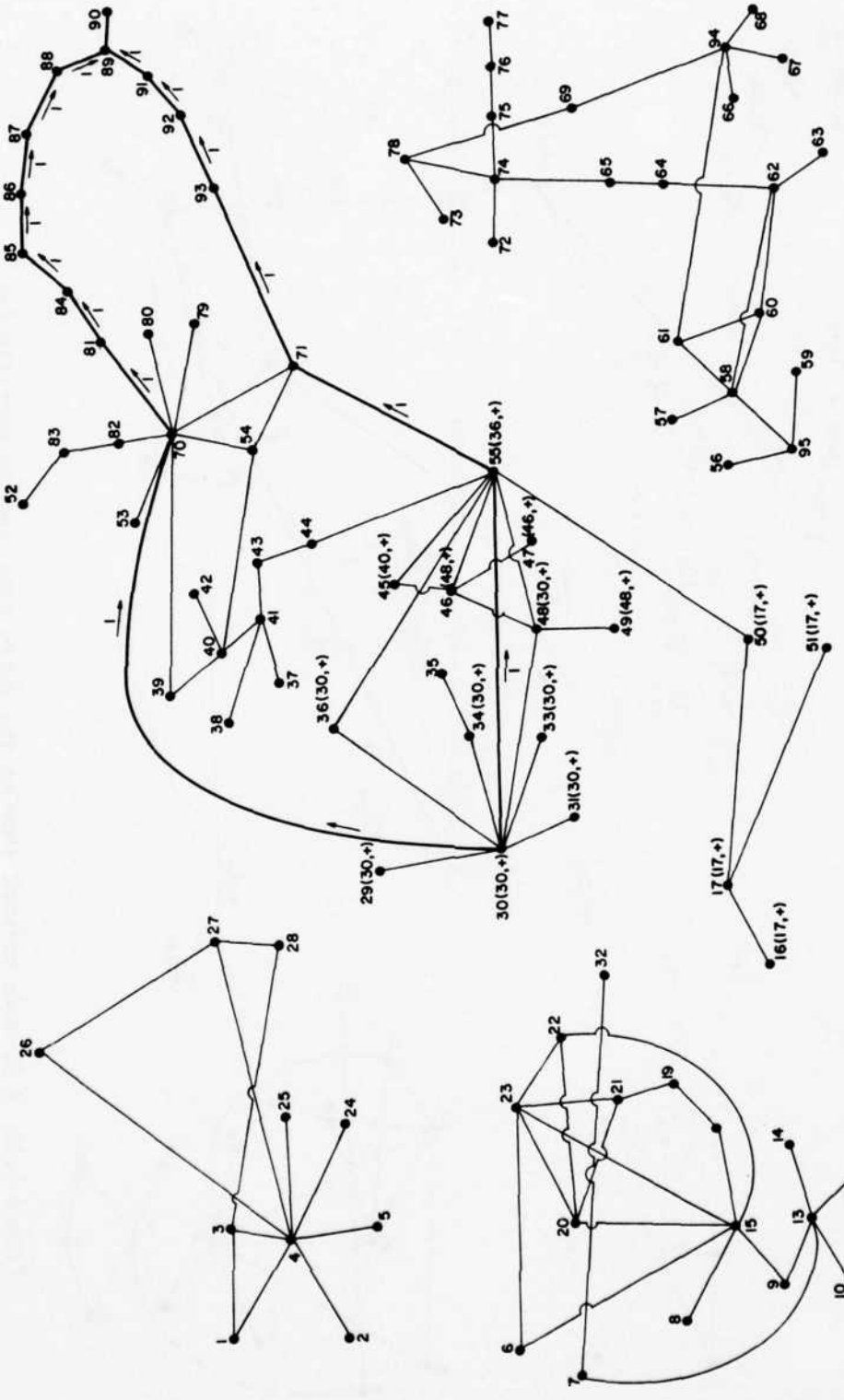


Figure 4.13 A 95-node network showing the fourth step towards partitioning.

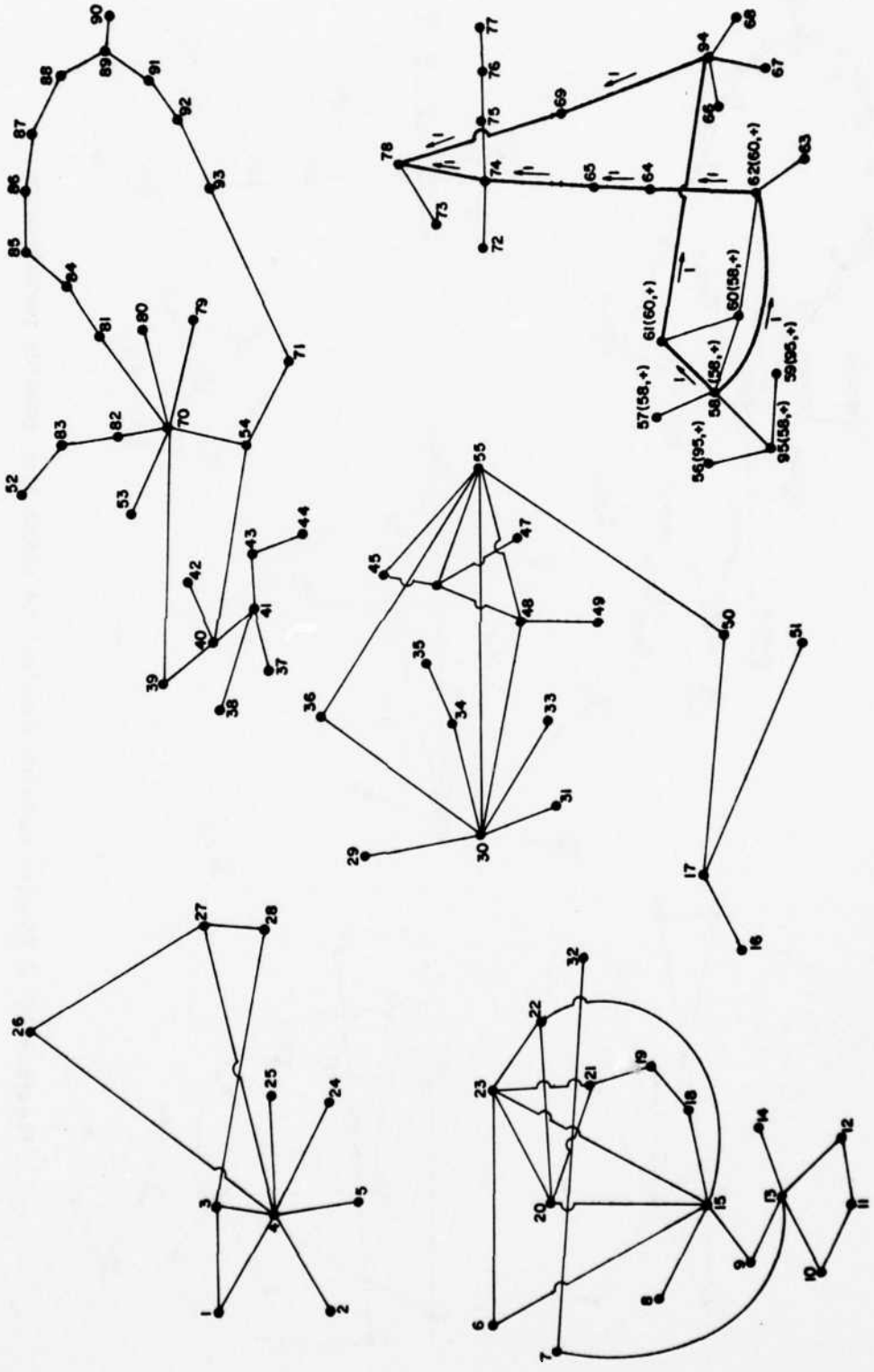


Figure 4.14 A 95-node network showing the fifth step towards partitioning.

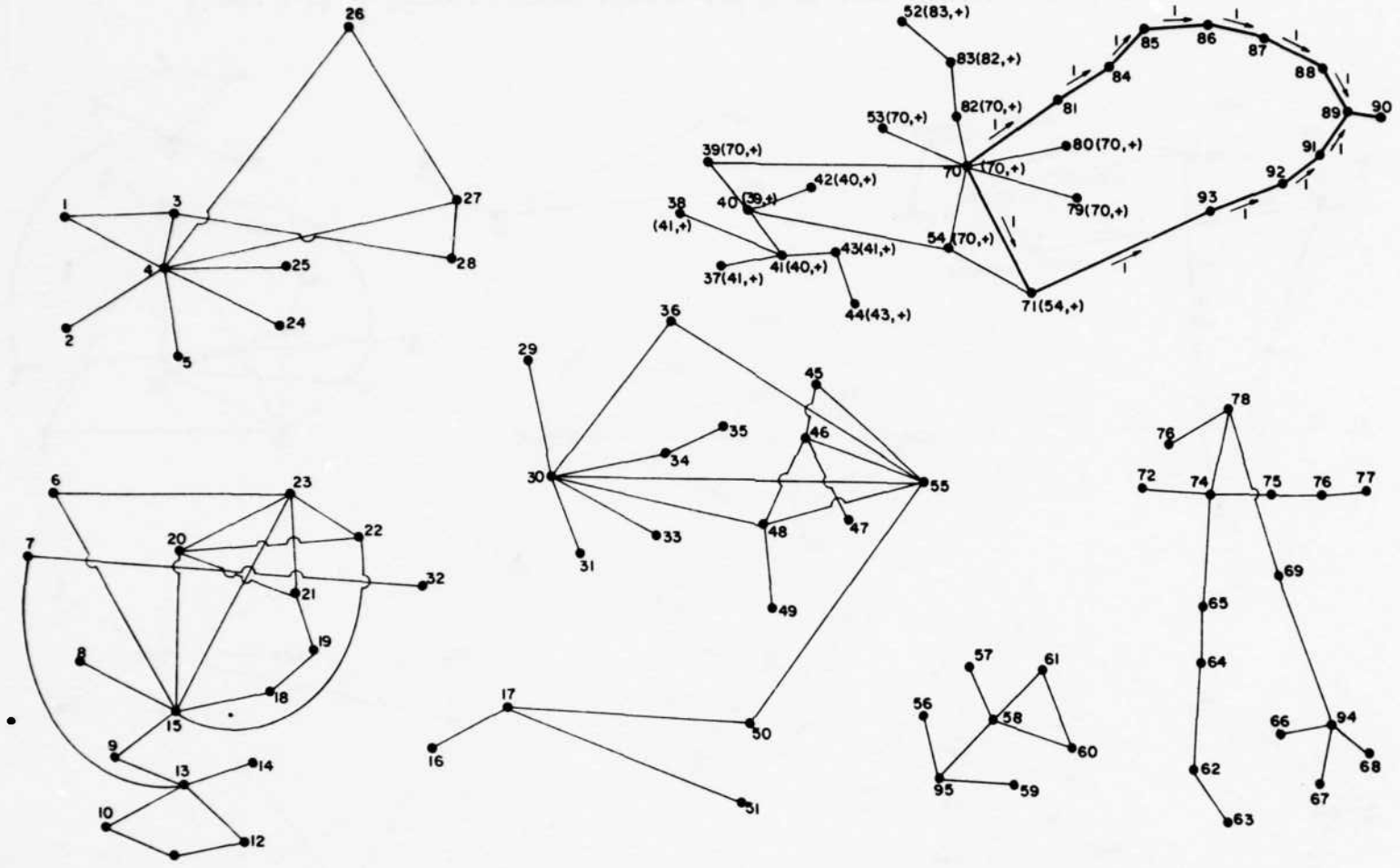


Figure 4.15 A 95-node network showing the sixth step towards partitioning.

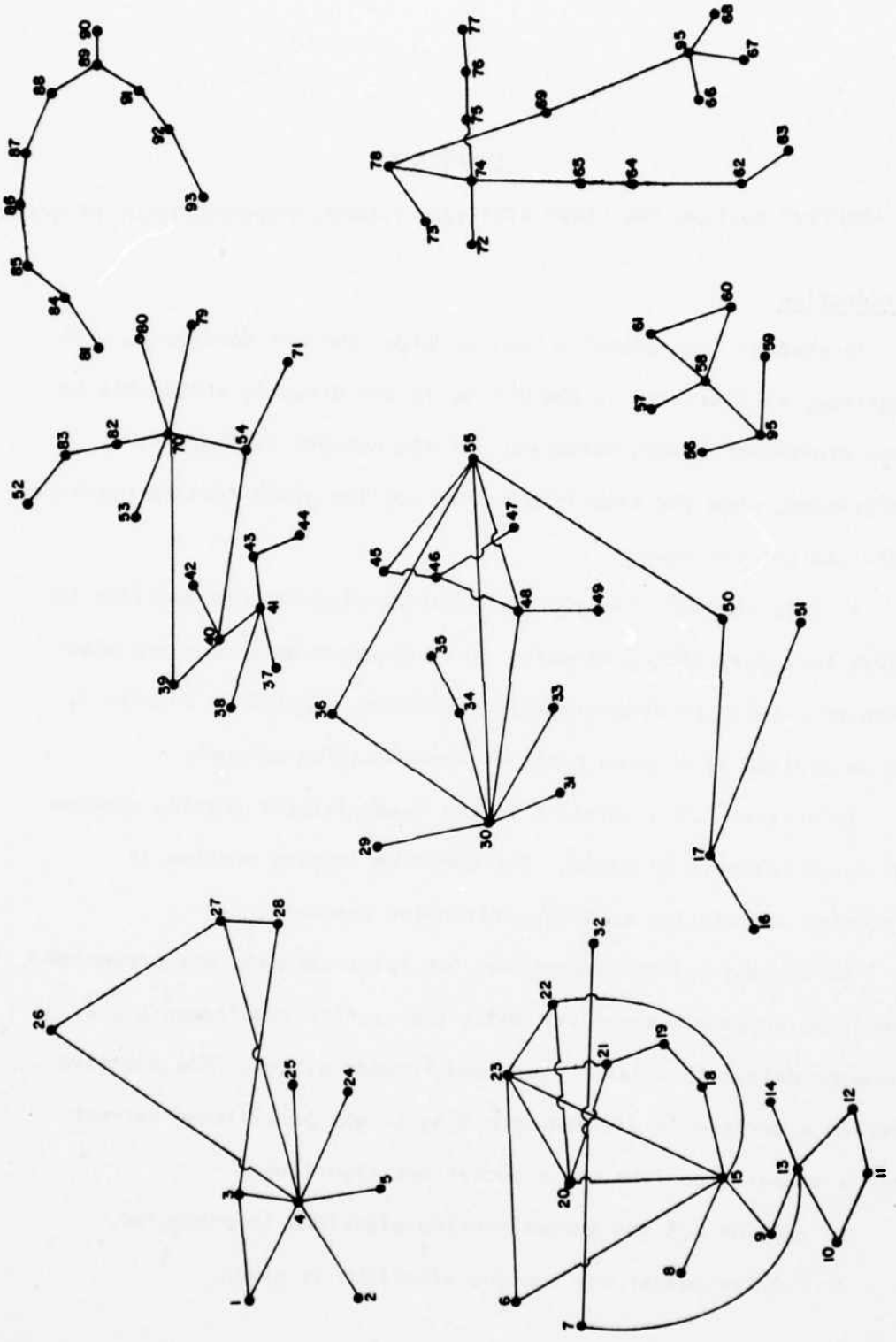


Figure 4.16 A 95-node partitioned network.

CHAPTER 5

ADAPTIVE ROUTING FOR LARGE STORE-AND-FORWARD COMMUNICATIONS NETWORKS

Introduction

In chapter 2 an adaptive routing algorithm was developed. The algorithm, as discussed in chapter 4, is not directly applicable to large store-and-forward networks. If the network is properly partitioned, then the algorithm can be applied after certain necessary modifications are made.

In this chapter, the adaptive routing algorithm is modified to adjust to a partitioned network. The necessary equations are also given such that the deterministic algorithm, adapted in chapter 2, can be applied to a large computer communication network.

In section 5.1 a solution to the deterministic routing problem for large networks is given. The adaptive routing problem is discussed in relation to the partitioning approach.

In section 5.2 considerations for large networks are presented. Intuitive arguments are given about the traffic requirements and queueing delays in a large store-and-forward system. The adaptive routing algorithm is divided according to the partitioned network into a subnet algorithm and a master net algorithm.

In section 5.3 the subnet routing algorithm is presented.

In 5.4 the master net routing algorithm is given.

In section 5.5 an example is simulated with the computer. The simulation language used is discussed.

In section 5.6 the simulated system characteristics are given, and the conclusions drawn from the simulation.

5.1 Deterministic routing for large store-and-forward networks

According to the partitioning algorithm, we can partition a network into subnetworks. Then study each subnet separately. We can then associate with each subnet a supernode and study the master network that connects all supernodes. In such a study, we only consider the external traffic, i.e. ,the traffic between nodes belonging to different subnets. Therefore, if m and n are supernodes, we define the composite requirement $r_{mn}^{(0)}$ as follows:

$$\begin{cases} r_{mn}^{(0)} = \sum_{i \in N(m)} \sum_{j \in N(n)} r_{ij} , & \text{for } m \neq n \\ r_{mn}^{(0)} = 0 , & \text{for } m = n \end{cases}$$

The total average delay T can be expressed in terms of the delays of master net and subnets in the following way:

$$T = \sum_{k=0}^{p} \frac{r^{(k)}}{r} T(k) \qquad (5.1)$$

where: $T(0)$ = delay of master net

$T(k)$, $k = 1, \ldots, p$ = delay of subnets

$r^{(0)} \triangleq \sum_{m=1}^{p} \sum_{n=1}^{p} r_{mn}^{(0)}$ = thruput of master net

$$r^{(k)} \triangleq \sum_{i,j \in N^{(k)}} r_{ij}^{(k)} = \text{thruput of subnet } k$$

$$r \triangleq \sum_{i=1}^{N} \sum_{j=1}^{N} r_{ij} = \text{total thruput.}$$

The simple relation (5.1) was obtained using Little's result as shown in [GERL 73A]. The delay T(k) has an interesting physical interpretation: T(0) is the average delay in the internode communications, and T(k) is the average delay suffered by internal and transit traffic in subnet k.

The deterministic problem for a large network can be formulated in a similar way as in chapter 2.

After partitioning the network into subnets each subnet is considered separately. The existing algorithms for deterministic routing are directly applicable. The average delay per message T(k) for each subnet k is exactly calculated using the flow deviation algorithm presented in chapter 2.

The master net can be viewed as a subnet itself since the size of the number of nodes, is small. Then the average delay of the master net T(0) can be calculated in the same way. The total average delay T of the entire network is given by equation 5.1.

Therefore, by partitioning the original network into subnetworks, the solution of the deterministic routing problem is possible.

The adaptive routing problem can also be solved by using the partitioning approach.

The algorithm presented in chapter 2 can be modified into a subnet routing strategy and a master net routing strategy.

In each subnet a number of nodes are chosen as central nodes. The exact number of control nodes will depend on the size of the subnet. It is assumed that the central nodes are connected directly and thus form the master net.

5.2 Large network adaptive routing considerations

The problems discussed below are present for small networks as well. But they are more imminent and magnified when large networks are considered. In the algorithm presented in chapter 2, the delay table information transmitted at update is a "snapshot" of the traffic situation in the network at the instant of the update. If the variations in queue lengths are too "fast" the situation may change within the update period.

In general, we have a "sampling" situation. The queues have variations: random due to the stochastic nature of the inputs; periodic due to the time-of-day variations, seasonal variation, etc. We are attempting to adapt to these changes. The sampling theorem applies in which case we have to sample "fast enough" to be able to sense ("see") the variations.

The stochastic variations are related to message length and message arrival times. Their interactions have been assumed to be independent (according to Kleinrock's independence assumption). But in actuality, since the message length is kept constant while it is routed from its source to its destination, there is a dependence and interaction of these variations. How they interact exactly is very hard to analyse. Unless we sample those variations fast enough, we have no way to do a sensible adaption to these changes.

Very fast update periods would be, in general, undesirable because of the additional load they put on the data information transfer.

Then the following modifications in the adaptive algorithm will be considered to smooth the above mentioned problems.

"Smoothing": In the case of stochastic variations, there are two distinguishable problems. First, the problem of "aging," i.e., the choice of a path through a distant node is based on data that is several update periods old. Given a large enough network, "aging" will affect optimality no matter how short the update period is.

Second, the data sent at update may be "unrepresentative" in some fashion and the correction of the delay tables information made accordingly may be very nonoptimal over the update period. That could cause a "deciding" node to misjudge even an adjacent node.

The approach taken here, to try to correct the above problems, is to use the statistics of the queues in the network.

In the adaptive algorithm of chapter 2, the queue length at the instant of the update is incorporated directly in the delay tables information. When the update of information is not extremely fast, the instantaneous queue length may change within the update period. In other words, one could expect that the instantaneous queue length is not representative of the situation. Thus, we attempt to bring it closer to the mean value of the queue. The weighted sum of the instantaneous queue length and the mean value of the queue length is considered replacing the instantaneous queue length in the delay table calculations.

From simulation data, it has been observed that there are "reliable" and "unreliable" routes for a message to follow, i.e.,

routes consistlng of llnks where the queue length changes more slowly 'on the average" and others where queue lengths fluctuate all over the place. The goal is to try to follow rellable routes as much as possible. In an unrellable route, large changes of the queue lengths will take place before the message routed to a destination node reach the node and thus take circuituous paths. The mechanism used to penalize the links whose variance is large is a penalty term, which results as a bias against choosing these links. The biasing is done by multiplying the queue length times the variance of the queue length. The above term is incorporated in the delay tables calculations. The adaptive algorithm is modified according to the above arguments and divided into a subnet routing strategy and a master net routing strategy.

5.3 Subnet routing strategy

Each node keeps local routing tables where destinations within the subnet are listed. In addition, the node receives a master net minimum delay vector from each local central node to any other subnet.

Let S_1 be the symbol of subnet S where node 1 belongs. And let j_{S_i} be the index of the central node j local to subnet S_i.

The master net minimum delay vector is then given by

$$MT_{j_{S_s}}(S_D) = \min_{i_{S_D}} (\min_{Lm} MT_{j_{S_s}}(i_{S_D}, Lm)) \qquad (5.2)$$

where $MT_{j_{S_s}}(i_{S_D}, Lm)$ is the delay table entry of the master net that represents the estimated delay to go from the initial node j_{S_s}, le

central node j local to the subnet S where the message source s belongs, to node l_{S_D}, i.e., central node l local to the subnet S where the message destination node D belongs, through the outgoing link Lm of initial node j_{S_s}. $MT_{j_{S_s}}(S_D)$ then is the estimated minimum delay to go from j_{S_s} to destination subnet S_D.

The delay table entries for messages with destination within the subnet are calculated via the delay table equation

$$T_K(D,Lg) = [Q(K,Lg) + MVQ(K,Lg)] \times 1/2 + T(D) + DELTA \qquad (5.3)$$

where Q(K,Lg) is the instantaneous queue length of node K on link Lg, MVQ(K,Lg) is the mean value of the same queue. T(D) is the delay vector entry for destination D as defined in Chapter 2, and DELTA is the bias factor.

When the variance of the queue length on link Lg exceeds a prespecified threshold value, then (5.3) is modified as follows:

$$T_K(D,Lg) = Q(K,Lg) \times VAR(K,Lg) + T(D) + DELTA \qquad (5.4)$$

where VAR(K,Lg) is the variance of the queue length of node K on link Lg. The term Q(K,Lg) x VAR(K,Lg) represents a bias against unreliable routes. As in Chapter 2, when the message length exceeds a certain threshold, the routing is not done through the delay tables information but dynamically via the following formula

$$L=\min_{Lm} [Q(K,Lm)+MVQ(K,Lm)]\times 1/2+N^*(K,D,Lm)\times \ell - N^*(K,D,Lm)\times \ell_{avg} + T(D) \qquad (5.5)$$

If the variance of the queue length on Lm exceeds a threshold equation (5.5) is modified as follows:

$$L=\min_{Lm}(Q(K,Lm) \times VAR(K,Lm) + N^{*}(K,D,Lm)x\ell -$$

$$- N^{*}(K,D,Lm) \times \ell_{avg} + T(D)) \qquad (5.6)$$

When a message is directed to another subnet, the source node determines after inspection of the routing tables and the master net delay vectors, the local central node which minimizes the sum of the local plus master net delay and sends the message to it. The equation used for the determination of the minimum delay central node is

$$T_K(D_{S_D},Lg') = \min_{j_{S_s}}(\min_{Lg} T_K(j_{S_s},Lg) + MT_{j_{S_s}}(S_D)) \qquad (5.7)$$

where $T_K(D_{S_D},Lg')$ is the estimated minimum delay to go from source node K to destination node D which belongs to subnet S_D, through the outgoing link of K,Lg'. The minimum is taken over the central nodes local to the message source node. Notice that the local central nodes are used as intermediate destinations to get to the destination subnet S_D.

When messages are routed outside their origination subnet and their length exceeds a certain threshold, again routing is done dynamically via the following formula:

$$L' = \min_{j_{S_s}}\{\min_{Lm}\{[Q(K,Lm) + MVQ(K,Lm) \times 1/2 + N^{*}(K,j_{S_s},L_m)\ell x -$$

$$- N^{*}(K,j_{S_s},Lm)\ell x_{avg} + T(j_{S_s})\} + MT_{j_{S_s}}(S_D)\} \qquad (5.8)$$

Equation (5.8) is the analogous of (5.7). If the variance of the queue length on Lm exceeds a threshold then (5.8) is modified as follows:

$$L' = \min_{j_{S_s}} \{ \min_{Lm} \{ Q(K,Lm) \times VAR(K,Lm) + N^*(K,j_{S_s},Lm) \times \ell - N^*(K,j_{S_s},Lm) \times \ell_{avg} + T(j_{S_s}) \} + MT_{j_{S_s}}(S_D) \} \quad (5.9)$$

After a message arrives at a local central node, it is handled by the masternet routing strategy.

5.4 Master Net Routing Strategy

All the nodes forming the master net are central nodes of the various subnets. Each central node in addition to the local routing tables of the subnet to which it belongs, is equipped with the master net routing tables, which show routes and minimal delays to all other central nodes. Using the master net delay tables, each central node computes the minimum delay vector, as in equation (5.2), which transmits to the nodes of the subnet to which it belongs. The delay table equation for the master net, as in Chapter 2, is

$$MT_j(i,Lm) = [Q(j,Lm) + MVQ(j,Lm)] \times i/2 + MT(i) + \text{DELTA} \quad (5.10)$$

where $MT_j(i,Lm)$ is the estimated delay to go from node j to node i through link Lm. MT(i) is the delay vector local to the master net and is calculated as follows:

$$MT(i) = \min_{L_m} MT_j(i,Lm) \quad (5.11)$$

If the variance on link Lm exceeds a threshold, then (5.10) becomes

$$MT_j(i,Lm) = Q(j,Lm) \times VAR(j,Lm) + MT(i) + \text{DELTA} \quad (5.11)$$

Special routing as in Chapter 2 is done via the formula:

$$ML = \min_{Lm}\{[MQ(j,Lm) + MMVQ(j,Lm)] \times 1/2 + MN^{*}(j,i,Lm)x\ell - MN^{*}(j,i,Lm) \times \ell_{avg} + MT(i)\} \quad (5.12)$$

where M is added in every symbol to indicate that the master net is being considered. Similarly, when the variance on Lm exceeds a threshold (5.12) becomes

$$ML = \min_{Lm}\{MQ(j,Lm) \times MVAR(j,Lm) + MN^{*}(j,i,Lm)x\ell - MN^{*}(j,i,Lm) \times \ell_{avg} + MT(i)\} \quad (5.13)$$

In the master net in addition to the routing tables, a connection vector is stored. That is, each central node keeps track of the nodes of its subnet, that it can reach through a local route, and stores the information in a N-dimensional connection vector. The lth entry of the vector is 0 if local node i is unreachable (because it is down or disconnected); it is 1 otherwise. The connectinon vector equation is

$$CVECTR_j(\ell) = \begin{cases} 1 \text{ if node } \ell \text{ is reachable from central node } j \\ 0 \text{ otherwise.} \end{cases}$$

Each central node, therefore, stores the connection vectors, one indicating the local nodes reachable from itself and the rest indicating the local nodes reachable from the other central nodes respectively. Fromthe inspection of these vectors, each central node after it receives a message determines:

(1) the message can be directly sent to a node of its subnet, or

(2) must be routed through the appropriate central node to reach its destination, or

(3) cannot be delivered because the destination is not reachable from any central node.

The update of the delay tables information is done as described in Chapter 2. At update, the subnets update their local routing tables. The master net updates the master net routing tables. It also calculates the minimum master net delay vector which is sent to the subnets nodes through their local central nodes.

The adaptive routing algorithm for large computed communication nets, presented above, was simulated.

5.5 The Simulation Language

The simulation language used to fit the case of a large network was GASP4.

GASP4, developed by A. Pritsker at Purdue [PRIT 74], is an event-oriented language. The fundamental approach is the decomposition of simulated time into events, i.e., into the points in time at which the state of the system could change. Each event routine is a snapshot of time. Time does not elapse during the course of an event routine. The event routine is a detailed description of the decisions and steps to be executed at the time of the event. Scheduling an event simply means filing in the event list a record identifying the event type and the event time. Events generally either initiate an activity or indicate the end of an activity. In coding an event, one looks at a point in time and considers all that can happen at that point. When no move can take place, an exit from the routine is made. Simulated time is advanced to the time of the next scheduled event and the procedure is repeated.

This approach is well suited to situations in which there are a large number of entities (in this case messages) flowing through a series of relatively simple processes.

5.5.1 Description of the Simulation

The adaptive algorithm is implemented in terms of two distinct events in time. An arrival event, when a message arrives in the system, and an end-of-service event, when a message leaves a node towards the next node in the path to its destination. The scheduling of the arrival event is done by using exponential distributed interarrival time. An end-of-service event occurs when simulated time equal to the message length has elapsed. The update of information is also associated with an event which is scheduled to occur after the occurance of a certain number of message arrival events.

The activities that take place at the end of an end-of-service event are keeping data of the queue statistics which the message has recently left. When a message arrives at its destination, statistics are kept based on the message itself.

The GASPIV output is statistics based on message observations and statistics based on link queues observations.

5.5.2 Simulated System Characteristics

The network simulated is shown in Fig. 4.4. The partitioning algorithm of Chapter 4 was used to partition the network in three subnets, Fig. 4.7. The two hierarchial level adaptive routing algorithm was then simulated with the aid of the computer. The two level network is shown in Fig. 5.1. The master net is shown separately in

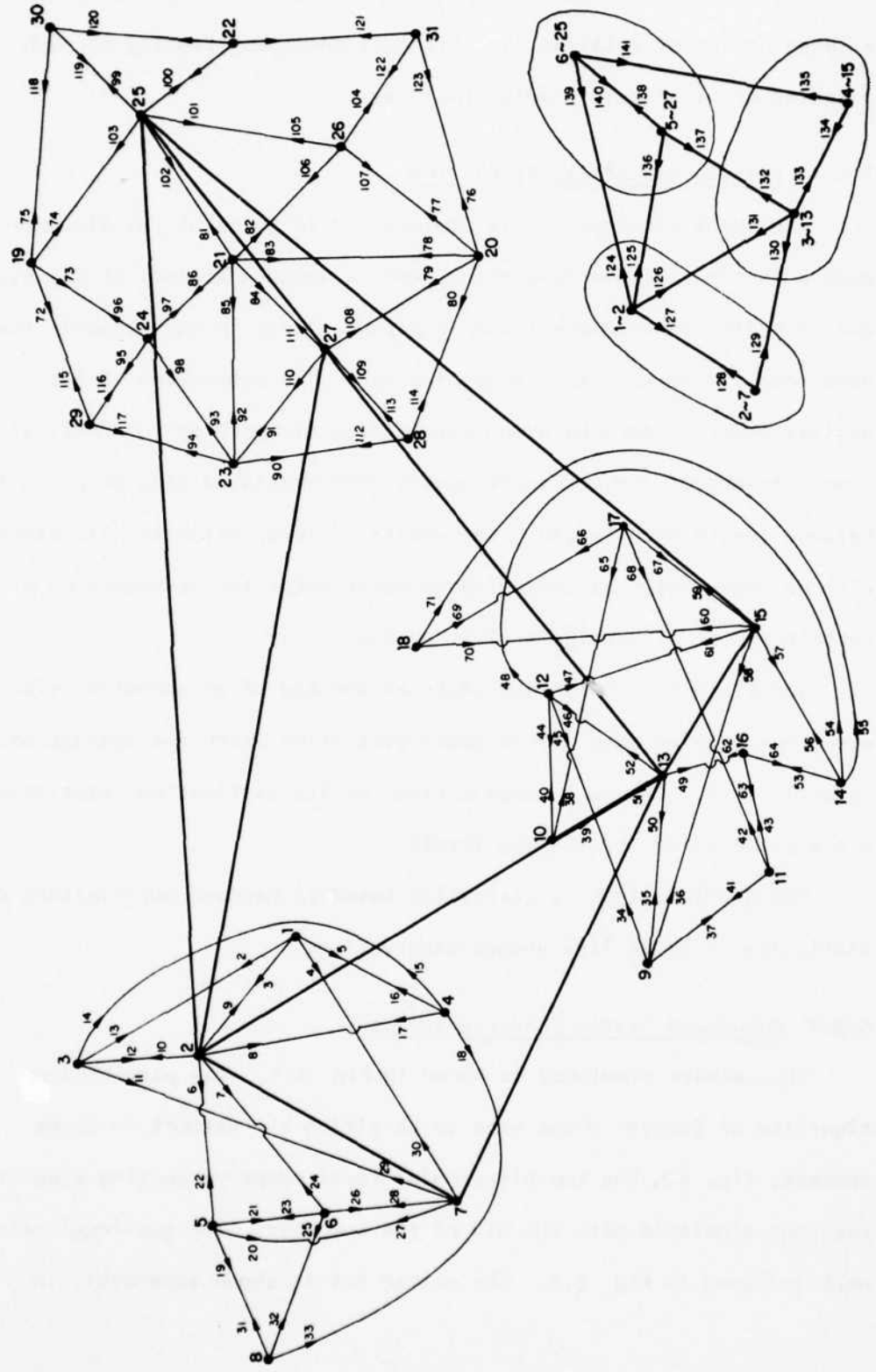


Figure 5.1 A two-level hierarchical network.

Fig. 5.1. The entire network consists of 31 nodes and 141 links. Two nodes are selected out of every subnet to be their local central nodes. The set of all central nodes form the master net. All links are two way. The node connectivity varies from 3 to 6, and each node represents a switch.

The traffic characteristics assigned to the network are as follows:

1. Message length: Exponentially distributed with a mean value of 21,500 bits/mes, or mean transmission time 9 sec/mes since the link capacity is taken to be C = 2400 bits/sec.
2. Arrival time: Poisson distributed such that the inter-arrival delay is exponentially distributed. The mean value of the exponential distribution was taken equal to $1/1.43 \simeq .7$ and to $\frac{1}{2} = .5$ for two traffic loads respectively. This corresponds to 1.43 mes/sec or 2 mes/sec arrival rate.
3. Initial and destination nodes: Independently drawn from a uniform distribution over the integer labels of the nodes (1 to 31).

The simulation was run several times for varying values of: synchronous update period, message length threshold for initiation of asynchronous update (special routing), and the bias factor DELTA. The update information messages, that the nodes send to each other after update takes place, are accounted for again by holding every link in the network for a short period of time (.01 sec, in this case) after update calculations are over.

For each run, the delay for each message was measured and the average delay computed per message was computed using the sample mean formula.

5.5.3 Conclusions

Simulation results are presented here for two different traffic loads. Two update periods, UPDATE 20 and UPDATE 100, were considered. The threshold value was varied from a value of TRSLD = 12 to a value of TRSLD = 70. The intermediate values were TRSLD = 20, 35, 50. TRSLD = 12 would correspond to the highest amount of special routing allowed and TRSLD = 70 to the lowest or practically no special routing, since very few message lengths will exceed a value of 70.

The values of DELTA were taken to be DELTA = 10, 20, 50, 70.

The RATIO of the ACTUAL TOTAL AVERAGE DELAY PER MESSAGE over the IDEAL TOTAL DELAY PER MESSAGE was formed. The actual total average delay per message is the summation of the transmission plus queueing delay. The ideal delay per message is only transmission delay. Data are presented for a simulation time of 600 seconds.

In Fig. 5.2a, the RATIO vs. DELTA was plotted for two update periods, a low traffic of $1/\gamma = .7$ sec/mes and a threshold value of 12.

The best results were obtained for the lowest value of DELTA used and an UPDATE of every 20 messages. It is of interest that when messages are strongly bias toward the direct route DELTA $\geq$ 50,70 UPDATE 100 results in lower delay than UPDATE 20. The delay increases for these values of DELTA because the direct route is overused causing longer queues and therefore longer delays. In addition to that, frequent update data messages create further delay rather than improving

the routing decisions. The same situation is shown even more in Fig. 5.2b where the traffic is higher, i.e., $i/\gamma = .5$ sec/mes.

In Fig. 5.3a the threshold value was raised to TRSLD = 35 and the same data as before were taken. A higher threshold value means that special routing is less and therefore direct routes are less encouraged to begin with. In this case the routing depends more on the update of routing information. Then the UPDATE 20 shows less delay than the UPDATE 100 curve. In higher traffic Fig. 5.3b the same situation is true. UPDATE 20 performs considerably better than UPDATE 100.

When the threshold was raised even more to a value of TRSLD = 70 frequent update (see Fig. 5.4a) performed better everywhere for low traffic ($i/\gamma = .7$ sec/mes). In higher traffic though ($i/\gamma = .5$ mes/sec) frequent update did not always show less delay. The UPDATE 100 (Fig. 5.4b) curve is fluctuating considerably as a function of DELTA. When the routing decisions are left practically to the updating of the routing tables (TRSLD = 70) and the traffic is high, there are instants depending on traffic peculiarities where practically random routing decisions result in reduced average message delay. If we define the output rate as the number of messages delivered during the simulation time period, then the output rate (Fig. 5.4b) at DELTA = 50 is lower for UPDATE 100 (156 messages delivered) than it is for UPDATE 20. Therefore, if we look at the update period from the output rate point of view, UPDATE 20 still performs better than UPDATE 100 for DELTA = 20,50,70. For DELTA = 10 (Fig. 5.4b) higher delay means lower output rate as well UPDATE 20). Therefore, for high traffic, alternate routing (DELTA = 10)

and low update (UPDATE 100) gives lower delay and higher output rate.

In Fig. 5.5a and b the effectiveness of the special routing is examined, by varying the TRSLD value, for the traffic loads respectively. The update is kept constant at a value of 20.

The special routing by the way it is defined always bias the routing decisions towards the direct path. The bias factor DELTA also bias the routing decision towards the direct path, by an amount directly proportional to its value. Then for low threshold value, i.e., a large number of messages are special routed, a low value of DELTA shows best performance for both traffic loads. That is, the two bias factors (TRSLD, DELTA)counteract each other. When the low TRSLD value strongly biases most routing decisions towards the direct path, then the lowest delay is obtained for the lowest DELTA value; that is, the least DELTA bias towards the direct path. When the threshold value is high (TRSLD = 50,70) higher DELTA values result in minimum delay per message (DELTA = 50). In high traffic situations (Fig. 5.5b), the same observations are more evident. This interaction of the two bias factors is expected to result in good performance, since in this way direct and alternate routes are used in a balanced way and therefore a minimum message delay is obtained.

In Fig. 5.6a and b the same data are plotted when the update is slower UPDATE 100. The same general observations are true and in addition the slower update has resulted in higher message delay everywhere. The effectiveness of the special routing is more dominant (spreading of the curves) than that of the DELTA bias for the slower update case. (Compare Fig. 5.5a,b, Fig. 5.6a,b). The data of Fig. 5.6 b

are plotted as a function of the threshold value in Fig. 5.7. A high value of DELTA = 50 shows minimum delay for the highest TRSLD = 70 value. For low TRSLD = 12 the DELTA bias factor performance is about the same for any value of DELTA which indicates the dominant effect, in this case, of the special routing.

As it is defined in Chapter 2, the number of "excess hops" measures the number of messages delivered which followed other than the direct path and took n excess hops to arrive at their destination. For n = 10, high traffic, and slow update, the interaction of the TRSLD and DELTA are examined in Fig. 5.8 in relation to the excess hops measurement. A low value of DELTA = 10 and a high value TRSLD = 50 showed least looping. Therefore, a value close to the average message length would prevent looping of messages, as it was claimed in the algorithm, in conjunction with the appropriate threshold value.

In Fig. 5.9 the same data as in Fig. 5.8 are plotted against the output rate, i.e., number of messages delivered to their destination. The maximum output rate occured for the lowest TRSLD value and lowest DELTA value. The same case also shows least average delay per message (see Fig. 5.2b). It is of interest to note that the same curve, i.e., DELTA = 10, TRSLD = 12 (see Fig. 5.8) gives the maximum number of messages going in a loop. This is because a low threshold value will route even short length messages (TRSLD = 12) to direct routes. When direct routes are overused, their queues are long and even slight variations in their length will tend to make

the short messages hop from queue to queue without actually being delivered. This fact creates a lot of looping for a portion of short length messages. It does not result in higher average delay per message because a portion of short length message delay contributes little to the average message delay. Therefore, looping of messages is inefficient but does not always result in higher average message delay.

Summary

Best performance was obtained for the fastest update used, a threshold value close to the average message value and a bias factor DELTA approximately equal to the average message length.

The effect of the update was as expected, i.e., in general, fast update resulted in less average message delay. The TRSLD variable interacted with the bias DELTA variable, for a balanced use of the direct route. A low DELTA value, biases the routing decisions toward alternating routes essentially, while a low TRSLD bias routes most messages towards the direct route. With this combination of the two bias factors, the best performance was obtained everywhere. The TRSLD bias, as designed, gave a stronger bias towards the direct route than the DELTA bias. A close to the average message length value of DELTA resulted in preventing messages of going into a loop. The looping of messages did not result in significant average message delay under the best performance conditions because mostly only a portion of short length messages went into loops in their path to their destinations. All of the above observations were true for both low and high message traffic conditions.

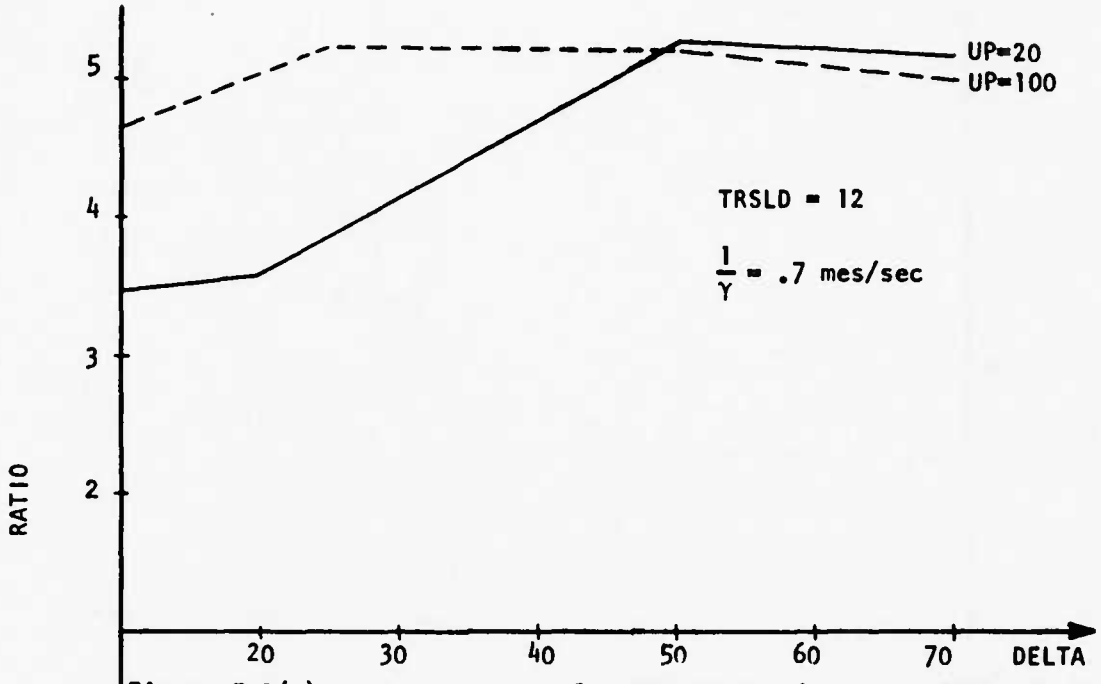


Figure 5.2(a) Ratio vs. Delta for TRSLD=12, 1/γ=.7 and UP=20,UP=100.

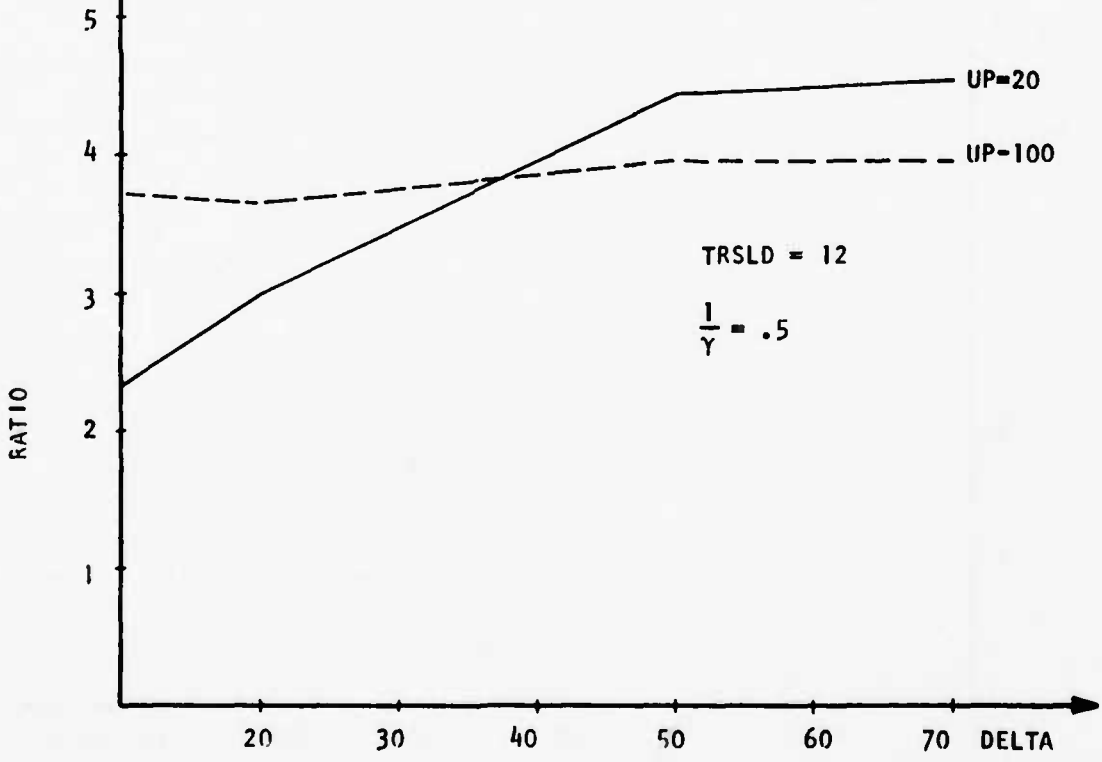


Figure 5.2(b) Ratio vs. Delta for TRSLD=12, 1/γ=.5 and UP=20,UP=100.

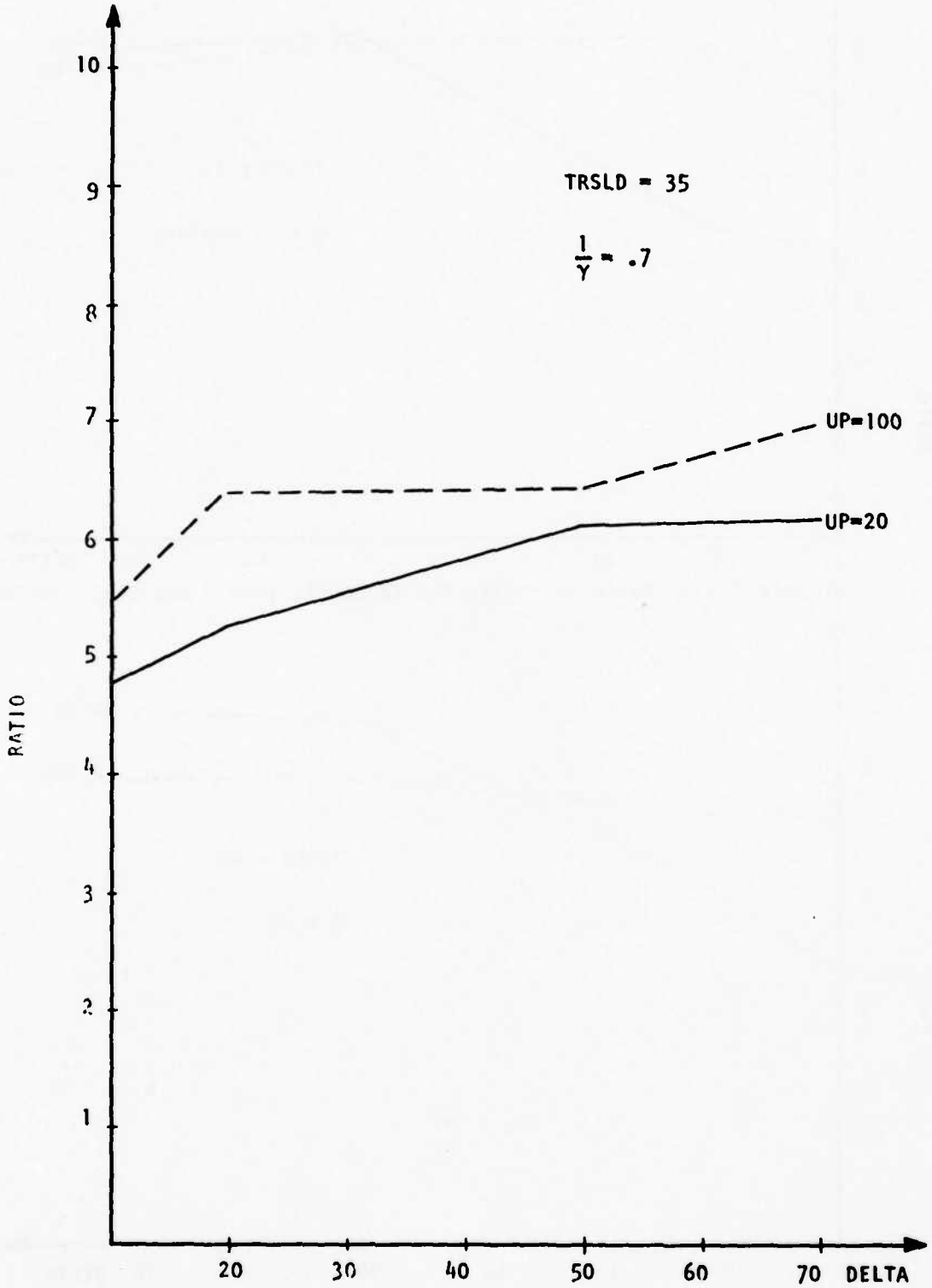


Figure 5.3(a) Ratio vs. Delta for TRSLD = 35, $1/\gamma$ = .7 and UP = 20, UP = 100.

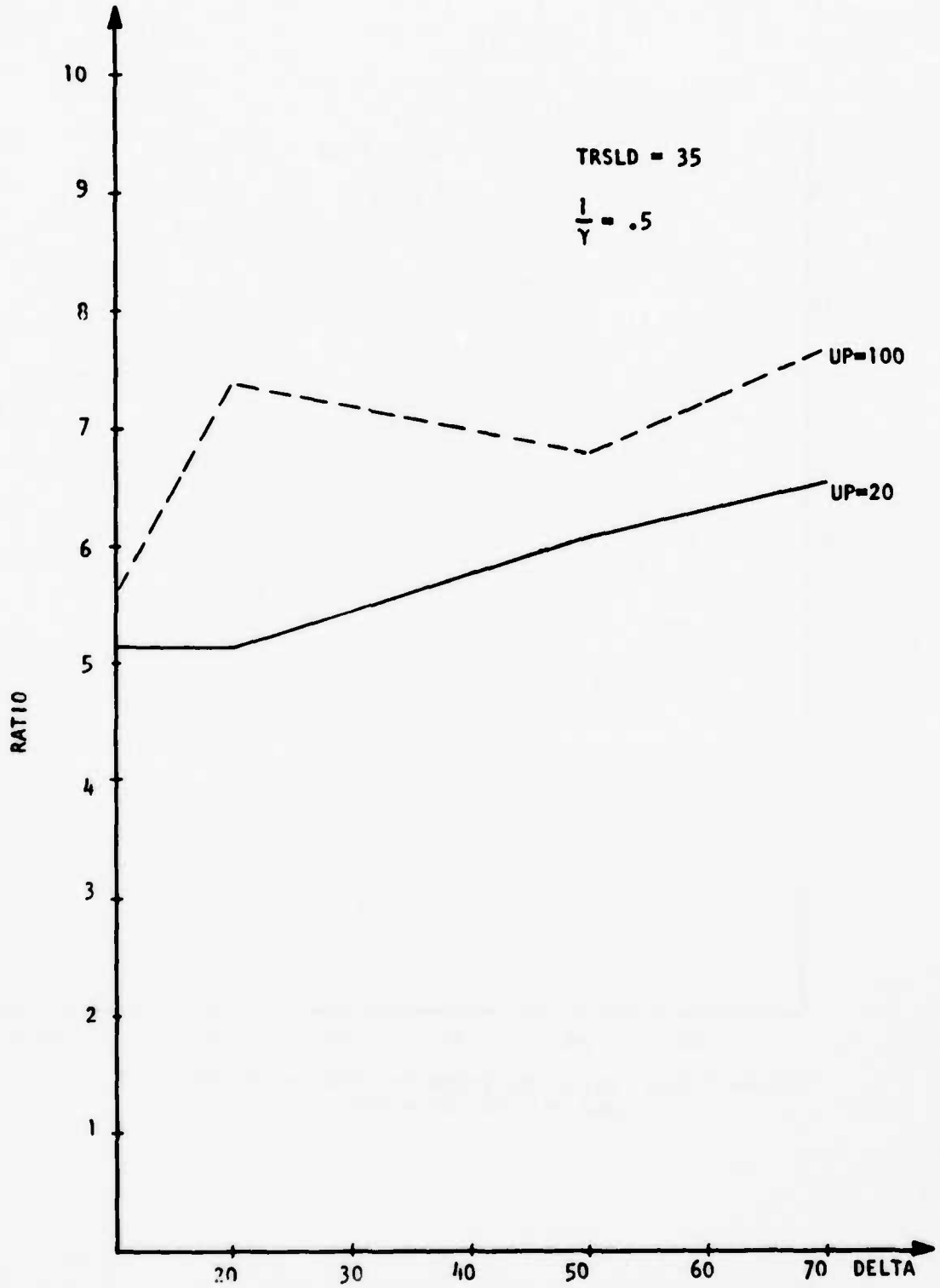


Figure 5.3(b) Ratio vs. Delta for TRSLD = 35, 1/Y = .5 and UP = 20, UP = 100.

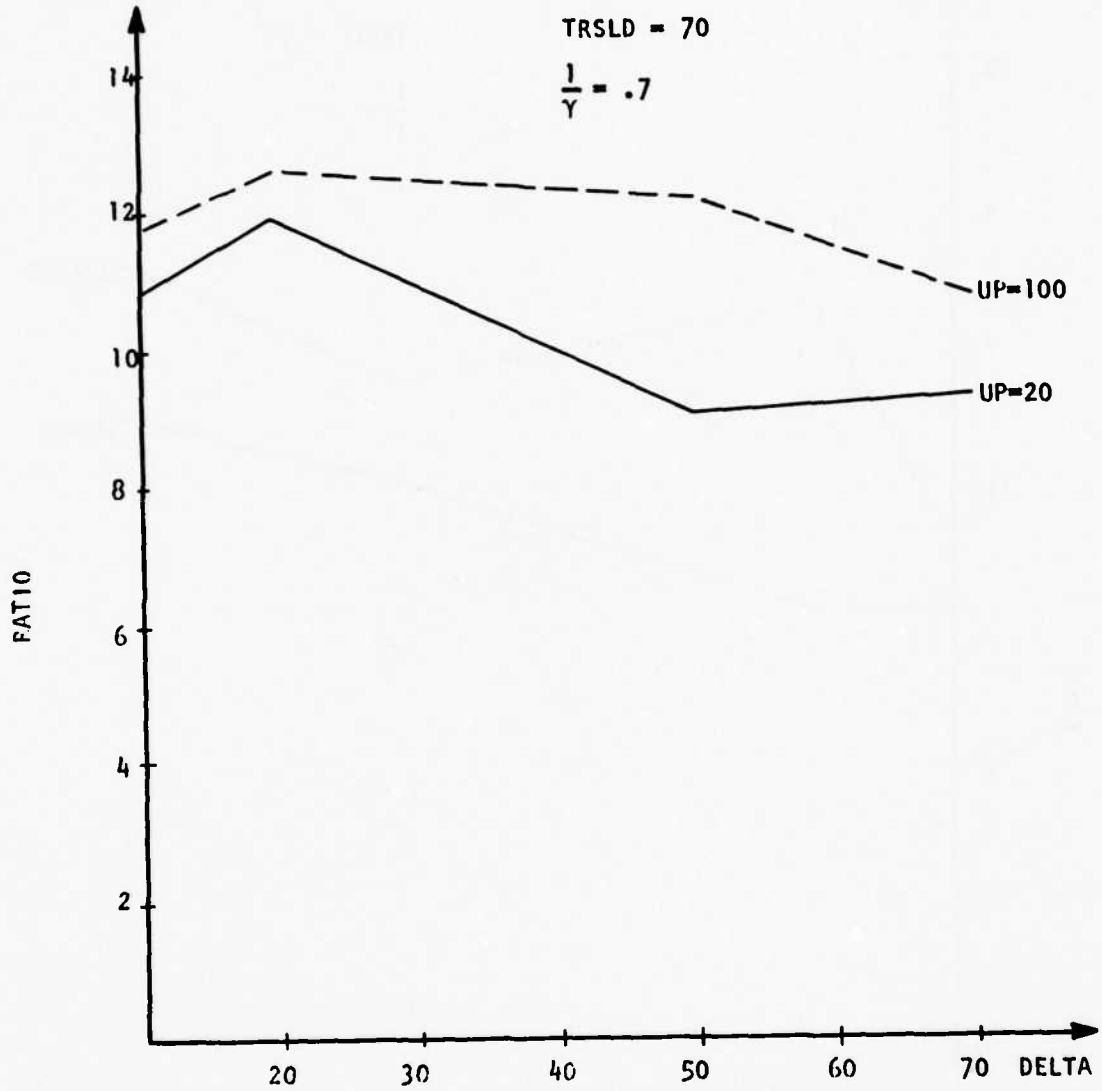


Figure 5.4(a) Ratio vs. Delta for TRSLD = 70, 1/Y = .7 and UP = 20, UP = 100.

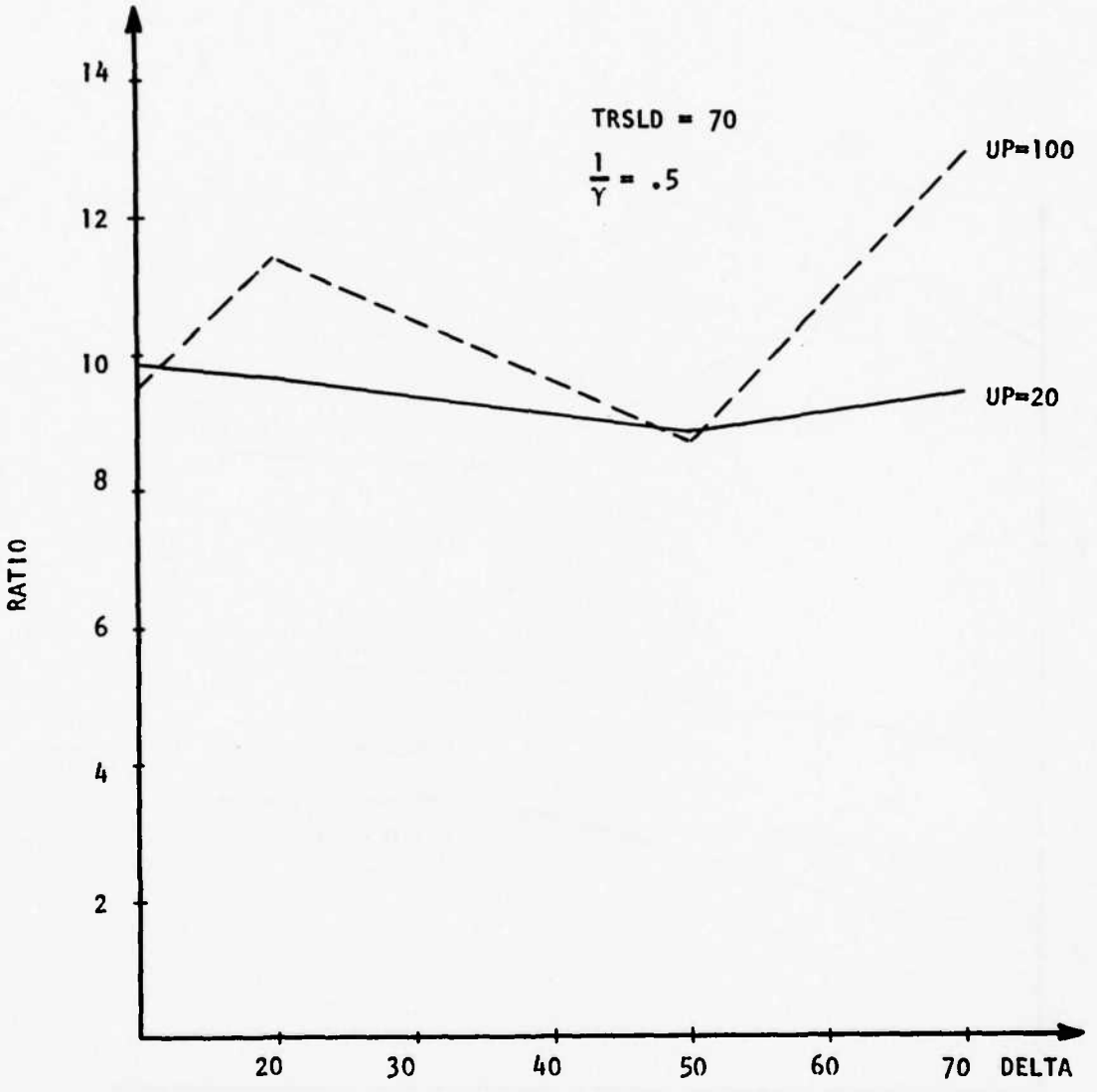


Figure 5.4(b) Ratio vs. Delta for TRSLD = 70, $1/\gamma$ = .5 and UP = 20, UP = 100.

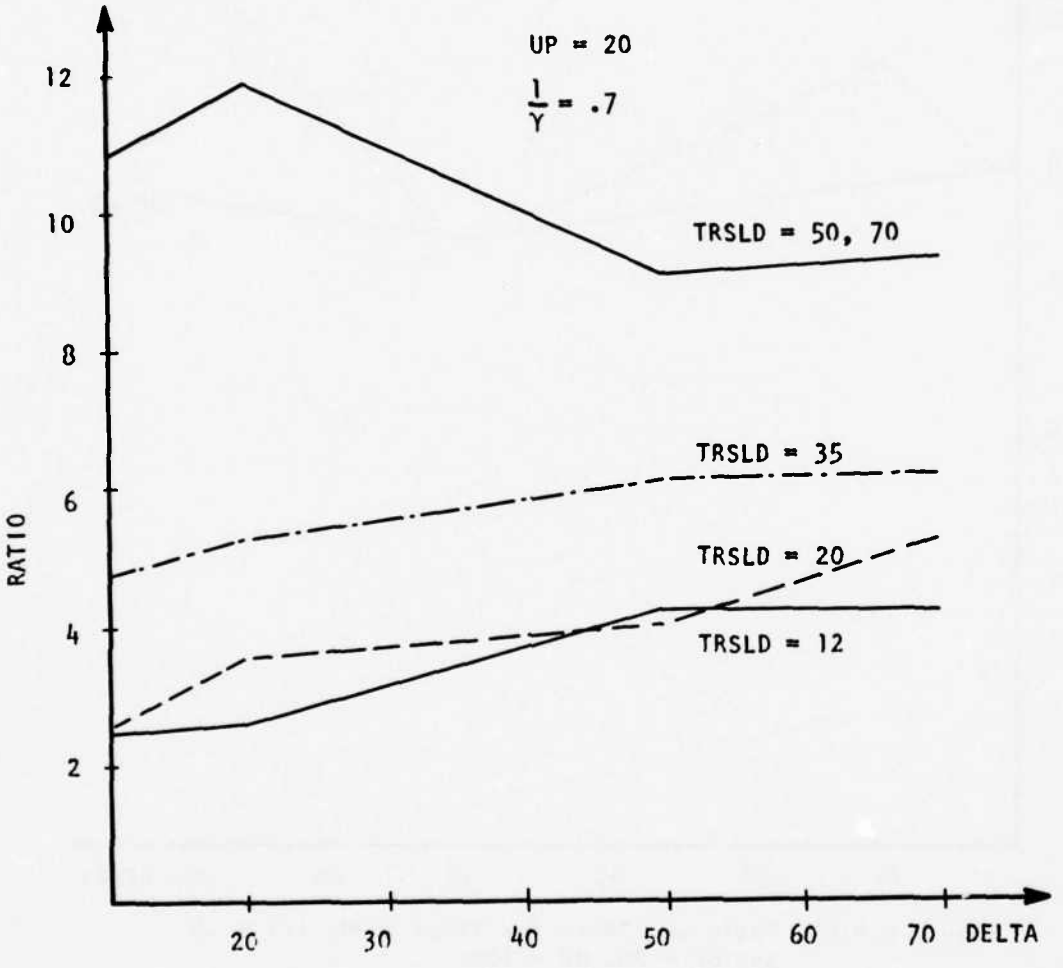


Figure 5.5(a) Ratio vs. Delta for UP = 20, $1/\gamma$ = .7 and TRSLD=12, TRSLD=35, TRSLD=50, 70.

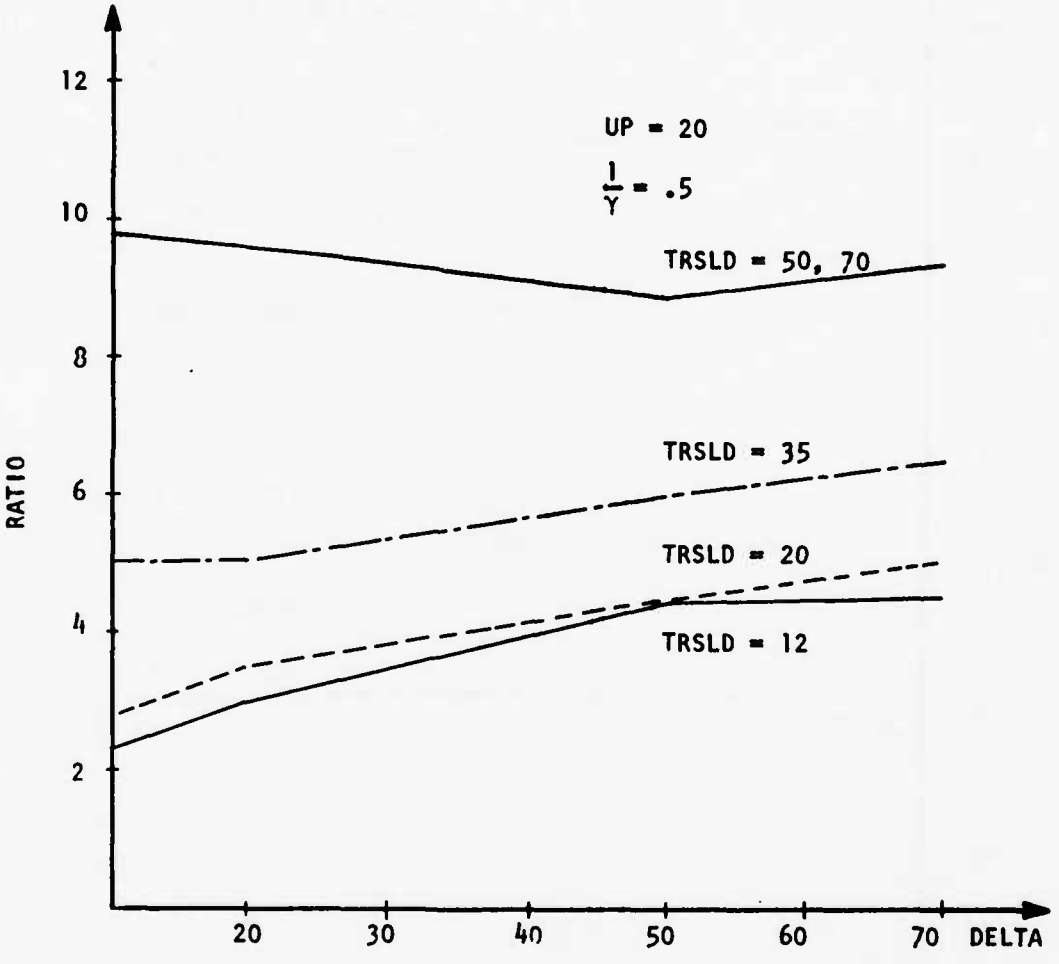


Figure 5.5(b) Ratio vs. Delta for UP = 20, $1/\gamma$ = .5 and TRSLD=12, TRSLD=20, TRSLD=35, TRSLD=50,70.

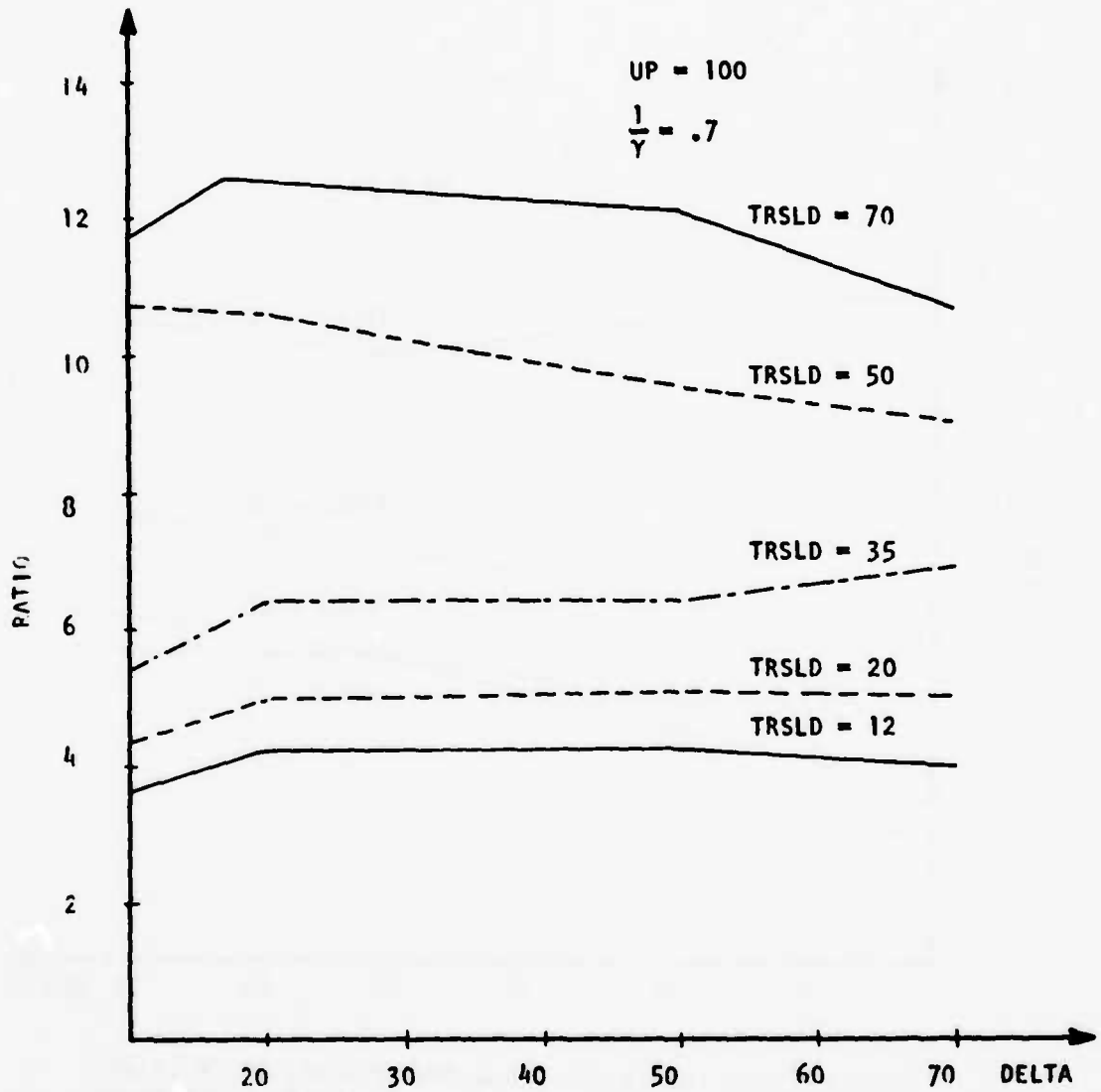


Figure 5.6(a) Ratio vs. Delta for UP = 100, 1/γ = .7 and TRSLD=12, TRSLD=20, TRSLD=35, TRSLD=50, TRSLD=70.

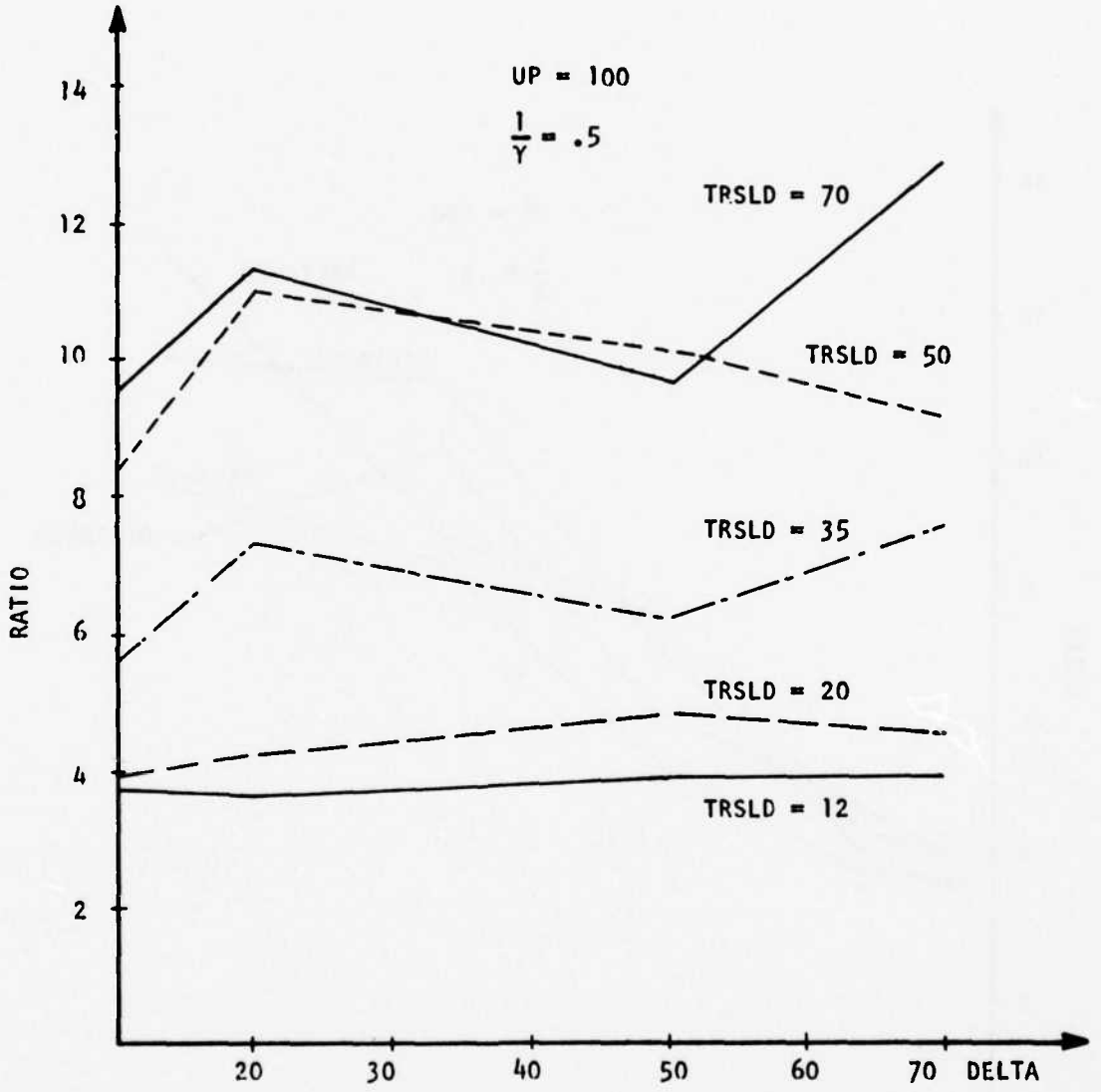


Figure 5.6(b) Ratio vs. Delta for UP = 100, $1/\gamma$ = .5 and TRSLD=12, TRSLD=20, TRSLD=35, TRSLD=50, TRSLD=70.

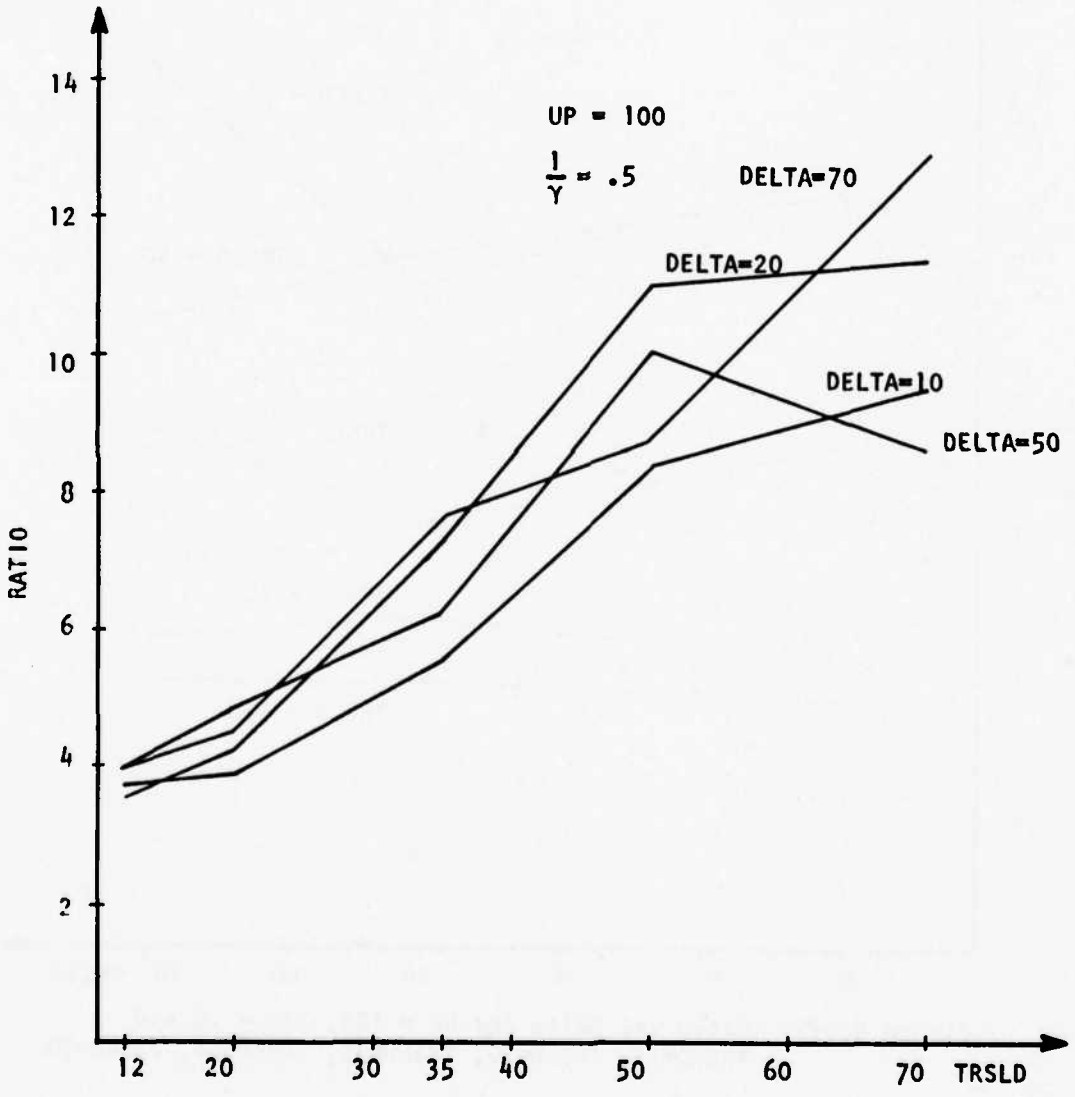


Figure 5.7 Ratio vs. Threshold for UP = 100, $1/\gamma = .5$ for DELTA=10, DELTA=20, DELTA=50, DELTA=70.

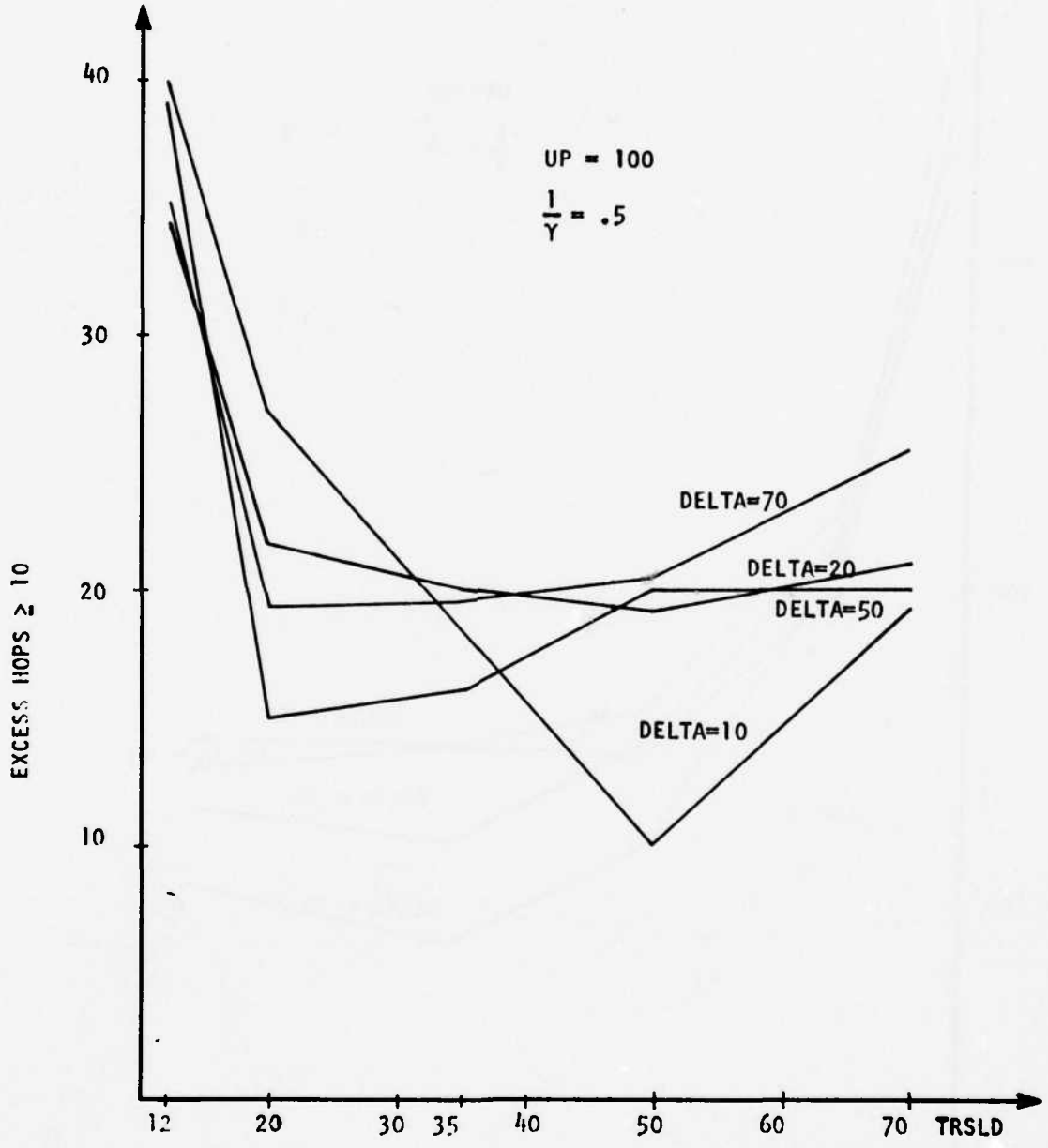


Figure 5.8 Ratio vs. Excess hops ≥ 10 for UP = 100, $1/\gamma$ = .5 and DELTA = 10, DELTA = 20, DELTA = 50, DELTA = 70.

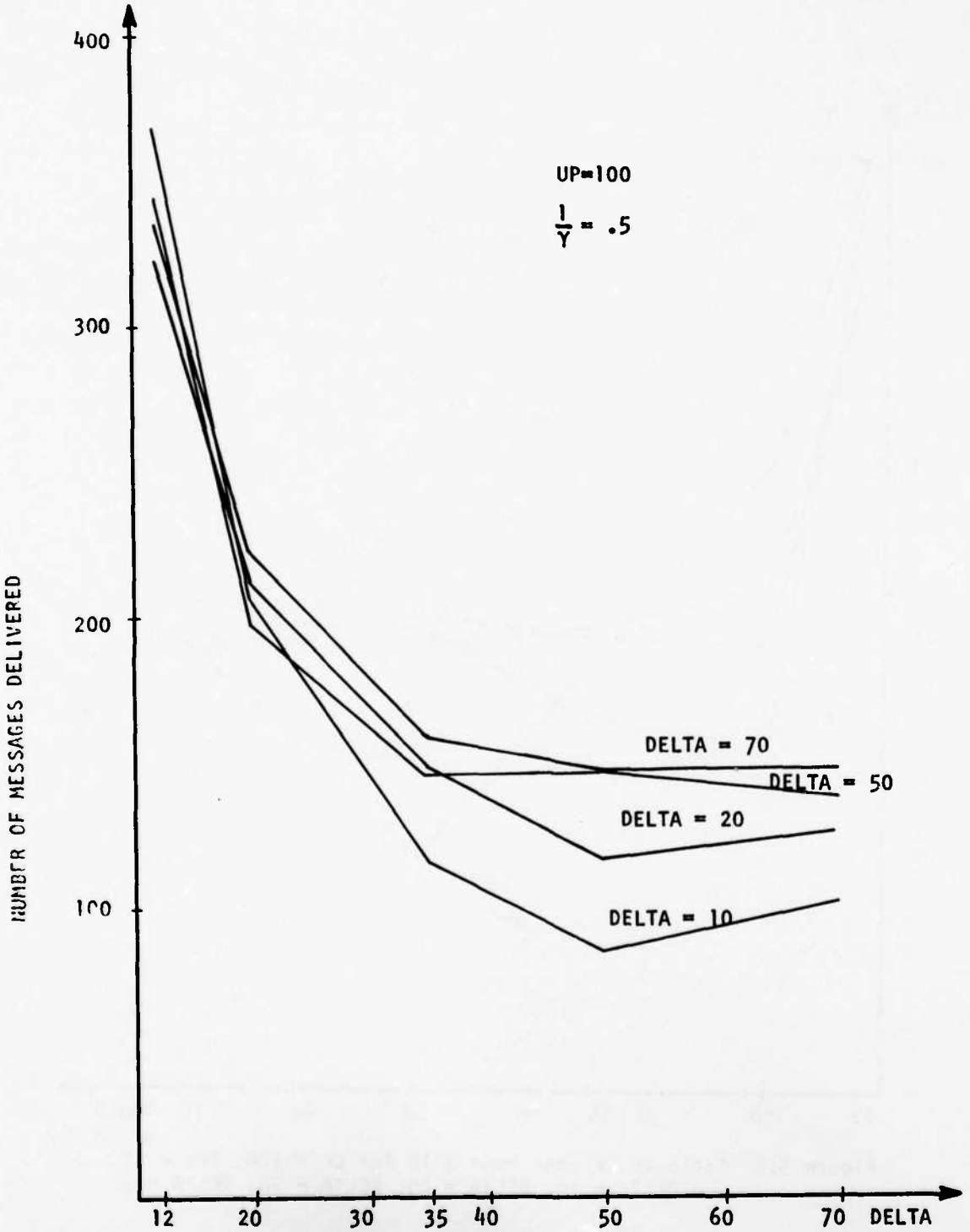


Figure 5.9 Number of messages delivered vs. Delta for UP = 100, $1/\gamma$ = .5 and DELTA = 10, DELTA = 20, DELTA = 50, DELTA = 70.

CHAPTER 6

ROUTING FOR GEOMETRICAL NETWORK TOPOLOGIES

Introduction

In the previous chapters, adaptive routing has been investigated as it applies to distributed communications networks. The network's topology was not exploited in the routing decisions. In general, the algorithm performs very well in a well connected network.

When the network's topology follows an exact geometrical structure, then this fact can be taken into consideration in the routing decisions.

In this chapter several geometrical network topologies are discussed in relation to the routing of messages. Specifically lattice, square, and hexagonal topologies are examined. In the lattice topology case, routing decisions are based on current information about the network queues and a local rule. The local rule is a set of instructions that every node follows, independent of the network's traffic and which takes advantage of the lattice geometry. When a large lattice network is considered, then the application of the local rule takes care of the partitioning of the network also. In the case of a square or an hexagonal network "response routing" depends completely on a local rule which fits specific network applications.

The discussion in this chapter intends to examine how the network's topology can be exploited in order to simplify routing. In Section 6.1 lattice networks are investigated. In Section 6.2, square-x, square-y,

and hexagonal networks are considered along with possible applications.

6.1 Routing with lattice networks

The network topology is in the form of a lattice, Fig. 6.1.

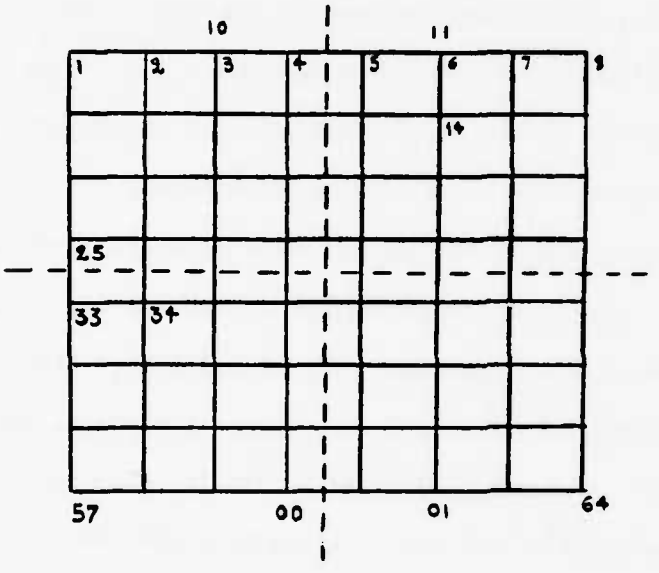


Figure 6.1 A lattice network topology.

A possible partitioning of the network can be based upon the Shannon-Fano method [FULT 72]. The network is divided into four regions as shown. The regions are numbered by a two bit address. The first bit, either 1 or 0, indicates whether the region is above or below the origin, respectively. Then each region is again subdivided into four parts and each subdivision is labeled with a two bit address code. Regions are subdivided by this technique until each small region contains one node.

For the example shown in Fig. 6.1, the network requires three levels of partitioning. Then each node can be assigned a unique number from the partitions.

For example, node number 14 is given the address (11, 10, 01) and node number 33 is given the address (00, 11, 11). For this type of network, the node addresses, which are unique by the partitioning procedure, provide information about their location in the topological structure.

Suppose a message is to be routed from node (11, 10, 01) (node 33) to node (00, 11, 11) (node 14). By comparing the first pair of bits in the source address with the first pair of bits in the destination node address, it can be determined that the destination node is located above and to the right of the current node. A logical choice is to route the message to the right (node 34) or up (node 25) since both of these nodes are in the general direction of the destination node. When the message originating at node 33 arrives at any node (11, xx, xx) contained in the major region of destination, there is a match between the first pair of bits in the address codes of the current node and the destination node. At this point, the second pair of bits in the node address codes is used for routing.

When more tnan one path out of any node exists which allows a message to proceed in the general direction of its destination, then a choice must be made as to which line should be used to exit from the node. Either a selective random routing procedure could be used or the shortest outgoing queue. In this way, the routing information depends only on the check on the addresses of the nodes and a random routine or a check of the shortest queue.

This method is replacing the routing decision computations by a check of the message destination address plus a check of the shortest outgoing queue or a random decision rule. The routing is of the alternate type but under restrictions, since once a message is in a certain address level it cannot return to the level it came from, even if in the meantime there are available routes there. This fact helps to avoid long looping of messages. The disadvantage of this scheme is that it is restricted to lattice topology networks. It would also require fast transmission communication links and short packetized messages to result in an acceptable performance. It is most likely to be used in a network when immediate response is not important, i.e., of the message switched type.

6.2 Routing with Selcuk networks

The network consists of a large array of nodes each having limited logic and memory [SAHI 74]. Each node is connected to immediate neighbors with one-way links in such a way that a directed path exists from any one node to any other node (condition of connectivity). By varying the number of links per node and the arrangement of the one-way channels (e.g., all incoming links adjacent or incoming and outgoing links alternating), it is possible to realize many networks. Fig. 6.2 shows different node types that will be considered.

In Fig. 6.2(a) and (b) the node types result in square networks. Hence, they will be called square. In the same way, Fig. 6.2(c) and (d) the node types are for hexagonal networks, hence the term hexagonal. Notice that nodes in Fig. 6.2(a) and (c) have their incoming links adjacent to each other. In Fig. 6.2(b) and (d) the incoming links alternate with the outgoing

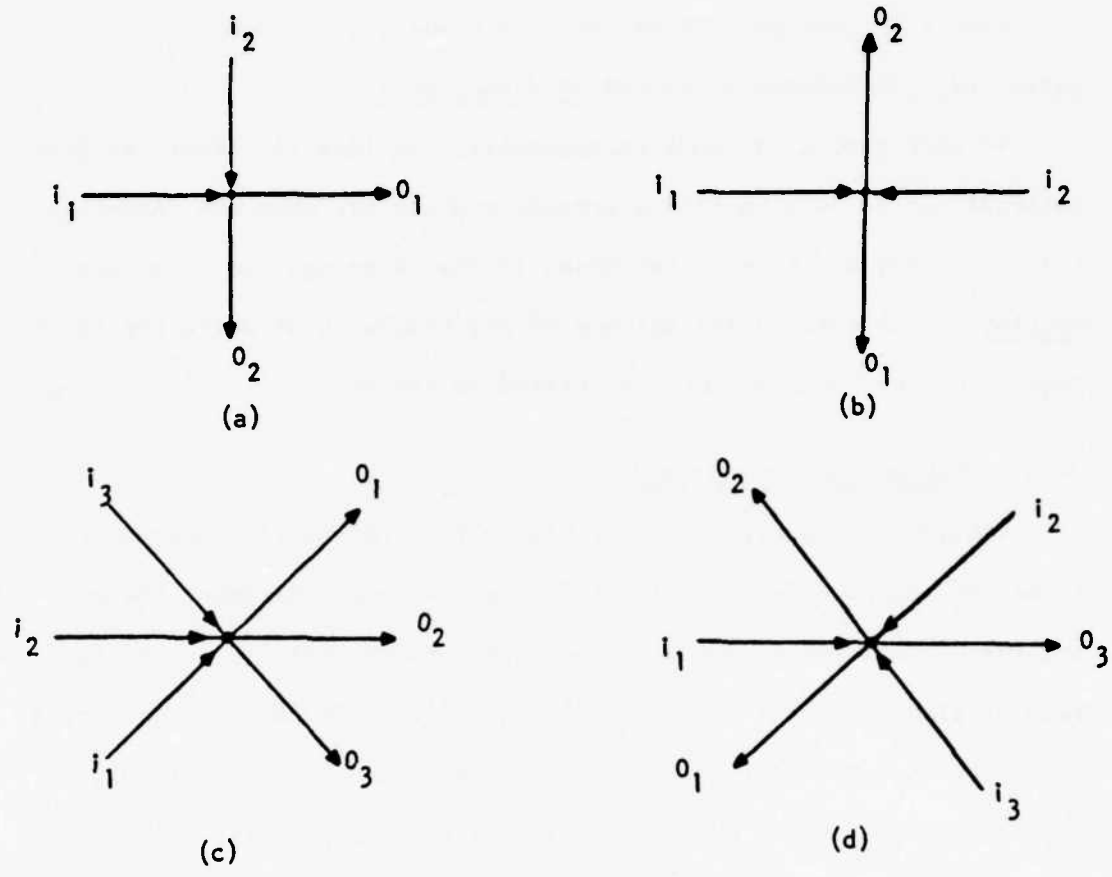


Figure 6.2 Four different node types configurations.

links. Type X will refer to nodes with adjacent incoming links and type Y to those in which incoming and outgoing links alternate. Thus, the node of Fig. 6.2(a) is square-X. In all cases, the labeling of links is in the clockwise direction, with i_1 denoting the first incoming link and i_2 the second, etc., and similarly the outgoing links.

The network in Fig. 6.3 contains only square-X node, the arrangement of which is such that it is possible to travel in straight paths from left to right, right to left, as well as up down and down up.

When all links are one way and the condition of connectivity is satisfied, the network is called unidirectional.

In such kind of network arrangements, the idea is: Given an unidirectional net be able to find a procedure where any node can interrogate the entire net and receive responses to the interrogation such that routing is achieved in the absence of any knowledge of where the interrogator or the respondents are located in the net.

6.2.1 The Selcuk Proposition

Consider the square-X net of Fig. 6.3. Arbitrarily, node 1 is chosen to be the source node that will initiate the interrogation. The quickest and the easiest way to propagate a message to the net is to have each node send it along all of its outgoing links. Since the net is connected, there are many circular paths to a given node and that will result in retransmissions of the message causing it to remain in the net forever. The echoes can be stopped by requiring that a node should not retransmit a particular message the second time it receives it. The stopping can be achieved either through a recognition of the already retransmitted message or by having a refractory period immediately after the arrival of a message, during which further messages are not accepted.

It is required that the time it takes for a message to go from one node to the next be the same at all nodes and define this interval as unit time. Using this convention and the stopping of second arrivals, in Fig. 6.4 the propagation of a message is shown initiated at node 1. The heavy elliptic dots show the input channels upon which the message appears for the first time. This is the message first arrival pattern.

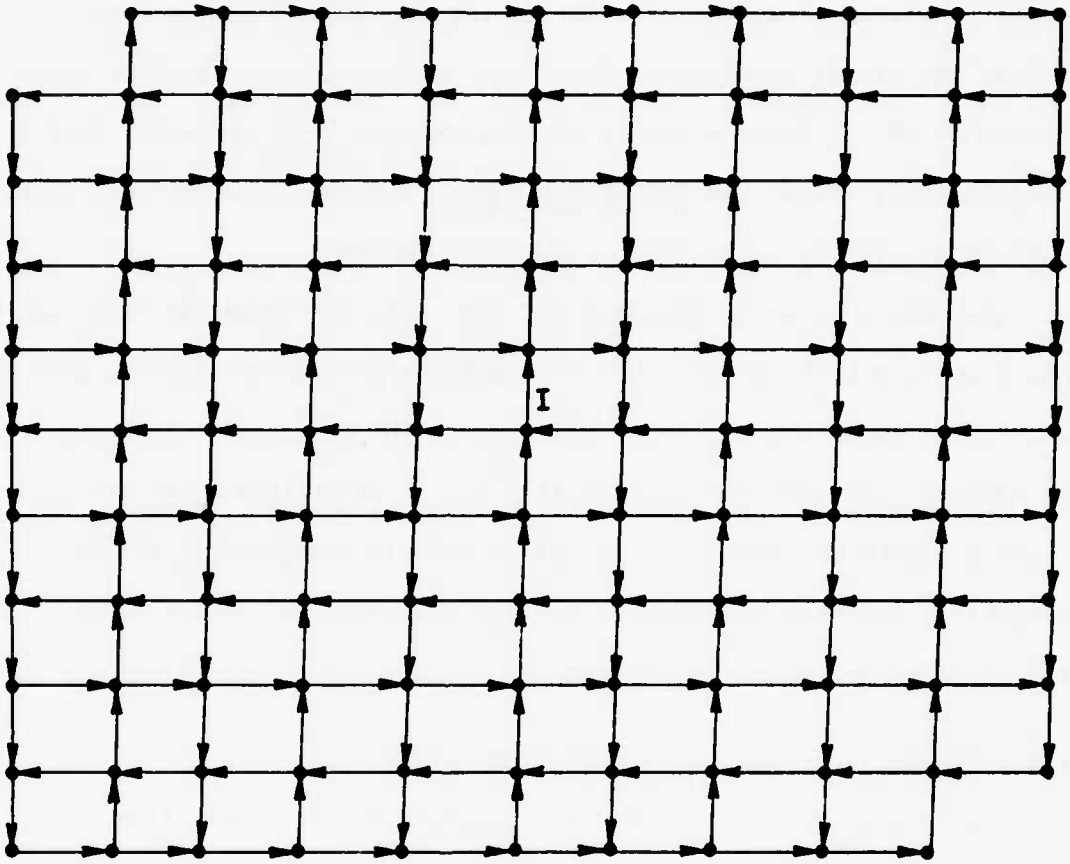


Figure 6.3 A square-X network topology.

The proposition is that in properly constructed nets response routing can be achieved without message routing information but by locally applied routing rules, in the nodes, which only depend on the first arrival pattern of a message. Therefore, each node need remember only upon which of its input channels a given message first arrived. This proposition is called the Selcuk principle. Any net in which this principle holds will be identified as a Selcuk network.

Consider the net in Fig. 6.3 and Fig. 6.4. For nodes in this net, the routing rule is as follows: If a message arrives on link i_1, when and if the receiving node has a response to this message, the response is routed through the outgoing link 0_2; if it arrives on 1_2 the routing should be through 0_1; and if it arrives on 1_1 and 1_2 simultaneously, the routing should be on 0_1. This rule is clearly local and dependent on local information.

6.2.2 Propagation Speed in Square-X Selcuk Nets

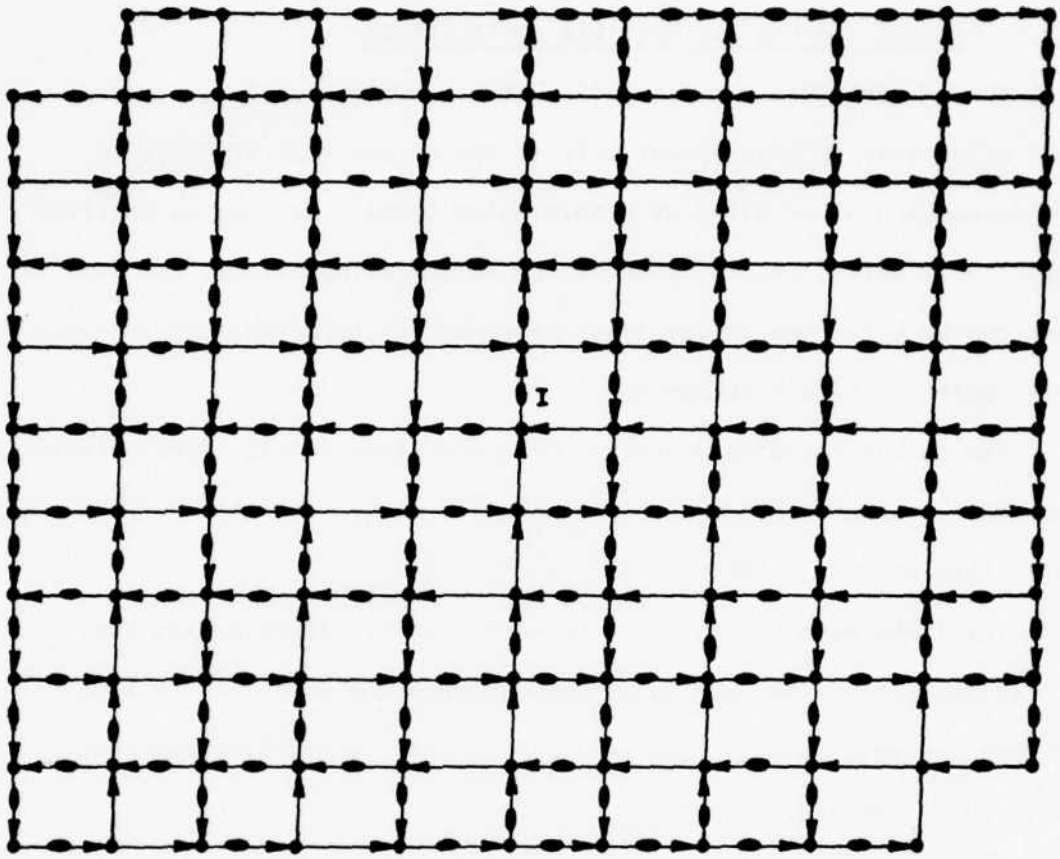
It is shown [SAHI 74] that in a square-X net at a unit time t the number of nodes covered is

$$n = \begin{cases} 2 \text{ at } t = 1 \\ 11 \text{ at } t = 4 \\ 4(t-1) \text{ at } t \geq 2 \quad t \neq 4 \end{cases}$$

The total number of nodes N covered in general is

$$N = \sum_{1}^{t} 4(t-1) + 2-1 = 2t(t-1) + 1 \quad t \geq 4$$

Similar kinds of arrangements (nets) and rules can be achieved with Hexagonal-Y Selcuk nets, Square-Y Selcuk nets, and Hexagonal-X Selcut nets.



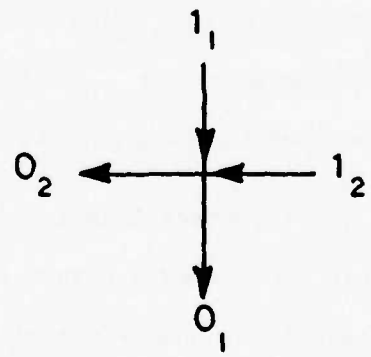


Figure C.4 First arrival pattern in a square-X network.

6.2.3 General Remarks and Possible Applications

The network that appears in Fig. 6.2 has a geometric structure. Such a topology is significant only to the extent that it reflects the assumption of equality of transmission times. As long as the time equality is satisfied, for example for compensating delays, the network can be stretched in any imaginable way provided that the connectivity patterns remain unchanged.

The Selcuk networks discussed were two dimensional. Sahin claims that cubic networks are under investigation along with the routing rules they should satisfy.

The links have been assumed unidirectional. It is proved that if bidirectional links are to be used instead the nodes suffer a 50 percent increase in logic and memory with only negligible improvement in speed of service.

Some specific applications of the Selcuk networks are proposed. As an example, consider a stock market which is distributed such that the investors can fully interact with each other. In such a net, each investor would be interested in propagating asking price and quantity to all other investors, i.e., send a message with or without his identification added to it. The investors interested in the offer or wishing to make a different bid could send a response with their identification such as a telephone or code number attached to it.

Another example of computer communication that would benefit from Selcuk schemes is one used for dynamic resource allocation among many branches of a company, such as a car rental company. Say one branch has excess cars. It could then send a message asking for

branches with certain degrees of shortages. The responses could be required to contain the location of the responding branch. Using mileage indicated by the located code, the excess cars then could optimally be allocated.

An area under extensive investigation is the use of Selcuk networks in pattern recognition [SAHI] In this context, the messages are viewed as initiators of local processes. Also of interest are reported findings about neuronel networks. Collectivity of neurons seems to constitute networks in which links are one way. Such nets appear to support spread of excitation. There is also evidence of information transfer from one part to another part of the network. Efforts to find a central switchboard to facilitate such transfers have failed. While there is no claim that the Selcuk principle is used in the brain, the idea behind this principle could be a useful starting point for a new approach to understanding brain behavior [SAHI 73].

REFERENCES

ABRA 73 Abramson, N. and Kuo, F. Computer-Communication Networks, Prentice-Hall, (Englewood Cliffs, New Jersey), 1973.

AGNE 75 Agnew, E. C. "On Quadratic Adaptive Routing Algorithms," Communications of the ACM, Vol. 19, No. 1, January 1976, pp. 18-26.

ASPO 72 A Simulation Process-Oriented Language. Reference Manual by Control Data Corporation Systems Publications, 1972, 215 Moffett Park Drive, Sunnyvale, CA 94086.

AUTO 75 System Description DCS AUTODIN, DCA Circular, 370-DG5-1, November 1975.

BARA 64 Baran, P. "On Distributed Communications," The Rand Corporation, Series of 11 Memoranda, August 1964.

BASK 72 Baskett, F. and R. Muntz. "Queueing Network Models with Different Classes of Customers," IEEE Computer Conference Proceedings, pp. 205-209, San Francisco, CA, Sept. 1972.

BERG 62 Berge, C. Theory of Graphs and Its Applications, Wiley, New York, 1962.

BOHE 64 Boehm, S. and P. Baran. "Digital Simulation of Hot Potato Routing in a Broadband Distributed Communication Network," The Rand Corporation Memorandum, Rm. 3103-PR, August 1964.

BOHE 66 Boehm, B. W. and R. L. Mobley. "Adaptive Routing Techniques for Distributed Communications," The Rand Corporation, Memorandum, Rm. 4781-PR, 1966.

BURK 56 Burke, P. J. "The Output of a Queueing System," Operations Research, 4:699-704, 1956.

BUSE 71 Buzen, P. J. "Queueing Network Models of Multiprogramming," Ph.D. thesis, Harvard University, Cambridge, MA 02138, August 1971.

CANT 72 Cantor, D. and M. Gerla. "The Optimal Routing of Messages in a Computer Network via Mathematical Programming," IEEE Computer Conference Proceedings, pp. 167-170, San Francisco, CA, September 1972.

CHEN 73 Chen, Z. and K. S. Fu. "Nonparametric methods for supervised and nonsupervised pattern recognition," Tech. Rept. TR-EE 73-24, School of Electr. Engr., Purdue University, West Lafayette, IN 47907, 1973.

DAFE 69 Dafermos, S. C. and F. T. Sparrow. "The Traffic Assignment Problem for a General Network," Journal of Research of the National Bureau of Standards 13, Vol. 73B, No. 2, April 1969.

DANT 63 Dantzig, G. B. Linear Programming and Extensions, Princeton University Press, Princeton, NJ, 1963.

DAVI 73 Davies, D. W., Barber, D. L. Communication Networks for Computers, John Wiley and Sons, London, 1973.

FELL 66 Feller, W. An Introduction to Probability Theory and Its Applications, Vol. II, New York: Wiley, 1966.

FLOY 62 Floyd, R. "Algorithm 97, Shortest Path," Communications of the ACM 5, 1962, p. 345.

FORD 56 Ford, L. R. and Fulkerson, D. R. "Maximal flow through a network," Canad. J. Math. 8, pp. 399-404, 1956.

FRAN 71A Frank, H., I. T. Frisch, W. Chou, and R. Van Slyke. "Optimal Design of Centralized Computer Network," Networks, 1:43-57, 1971.

FRAN 71B Frank, H. and W. Chou. "Routing in Computer Networks," Networks, 1:99-112, 1971.

FRAT 73 Fratta, L., M. Gerla and L. Kleinrock. "The Flow Deviation Method: An Approach to Store-and-Forward Communication Network Design," Networks 3, 1973, pp. 97-133.

FRIS 67 Frisch, I. T. "Analysis of the Vulnerability of Communication Nets," Proceedings of the First Annual Princeton Conference on Systems Science, pp. 188-192, Princeton, NJ, 1967.

FU 75 Fu, K. S. and Z. Chen. "On the Connectivity of Clusters," Information Sciences 8, pp. 283-299, 1975.

FULT 71 Fultz, G. L. and L. Kleinrock. "Adaptive Routing Techniques for Store-and-Forward Computer-Communication Networks," Proceedings of the International Conference on Communications, pp. 39-1 and 38-9, Montreal, Canada, 1971.

FULT 72 Fultz, G. L. Adaptive Routing Techniques for Message Switching Computer-Communication Networks, School of Engineering and Applied Science, University of California, Los Angeles, UCLA-ENG-7252, July 1972.

GERL 73A Gerla, M. The Design of Store-and-Forward Networks for Computer Communications, School of Engineering and Applied Science, University of California, Los Angeles, UCLA-ENG-7319, January 1973.

GERL 73B Gerla, M., W. Chou, and H. Frank. "Computational considerations and routing problems for large computer communication networks," IEEE National Telecommunications Conf. Proc., Vol. 1, p. 28-1, November 26-28, Atlanta, GA, 1973.

HARR 72 Harrer, J. R. "A Graph Optimization Synthesis Formulation for Cable Communications Systems," IEEE Trans. on Circuit Theory, May 1971, pp. 252-255.

HIMM 72 Himmelblau, D. M. "Applied Nonlinear Programming," McGraw Hill, 1972.

HU 69 Hu, T. C. Integer Programming and Network Flows, Addison-Wesley, Reading, MA, 1969.

HOUS 75 Houstis, C. E., W. J. Kelley, and B. J. Leon. "A Distributed Control for Autodin," TR-EE 75-25, School of Electrical Engineering, Purdue University, West Lafayette, IN 47907, July 1975.

JACK 57 Jackson, J. R. "Networks of Waiting Lines," Operations Research, 5:518-521, 1957.

KAMO 76 Kamoon, F. "Design Considerations for Large Computer Communication Networks," Ph.D. thesis, Computer Science Dept., School of Engineering and Applied Science, University of California, Los Angeles, 1976.

KLEI 64 Kleinrock, L. Communication Nets: Stochastic Message Flow and Delay, McGraw-Hill, New York, 1964.

KLEI 75 Kleinrock, L. Queueing Systems: Theory and Applications, Wiley Interscience, New York, 1975.

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

KNUT 68 Knuth, D. E. Series in Computer Science and Information Processing, "Fundamental Algorithms", Vol. 1, Addison-Wesley Publishing Company, 1968.

MART 72 Martin, J. Systems Analysis for Data Transmission, Prentice-Hall, (Englewood Cliffs, New Jersey), 1972.

MCQU 74 McQuillan, J. M. Adaptive Routing Algorithms for Distributed Computer Networks, Ph.D. Thesis, Harvard University, May 1974.

MEIS 71 Meister, B., H. R. Mueller, and H. R. Rudin. "New Optimization Criteria for Message Switching Networks," IEEE Transactions on Communication Technology, Vol. COM-19, No. 3, June 1971.

MEIS 72 Meister, B., Muller, H. R., Rudin, H. R. "On the optimization of Message-Switching Networks," IEEE Transactions on Communications, Vol. COM-20, No. 1, February 1972, pp. 8-14.

GORD 67 Gordon, W. J. and Newell, G.F. "Closed Queuing Networks with Exponential Servers," Operations Research, April 1967, pp. 254-265.

ROBE 67 Roberts, L. G. "Multiple Computer Networks and Inter-Computer Communications," ACM Symposium on Operating Systems Principles, Gatlinburg, TN, October 1967.

ROBE 70 Roberts, L. G., B. D. Wessler. "Computer Network Development to Achieve Resource Sharing," AFIPS Conference Proceedings, Vol. 36:543-599, SJCC, Atlantic City, NJ, 1970.

PRIT 74 Pritsker, B. A. The GASPIV Simulation Language, Wiley-Interscience publication, 1974.

PROS 62 Prosser, R. J. "Routing Procedures in Communication Networks, Part I: Random Procedures," IRE Transactions on Communications Systems, CS-10, pp. 322-329, 1962.

PROS 62 Prosser, R. J. "Routing Procedures in Communication Networks, Part II: Director Procedures," IRE Trans. on Communications Systems, CS-10, pp. 329-335, 1962.

RUBI 74 Rubin, I. "Communication Networks: Message Path Delays," IEEE Transactions on Information Theory, Vol. IT-20, No. 6, November 1974, pp. 738-745.

RUBI 75 Rubin, I., "Message Path Delays in Packet-Switching Communication Networks," IEEE Transactions on Communicatcns, Vol. COM-23, No. 2, February 1975, pp. 186-192.

SAHI Sahin, K.E. U. S. Patent 3 794 983.

SAHI 73 Sahin, K. E. "Response Routing in Selcuk Networks and Lashley's Dilemma," Int. J. Man-Machine Studies, Vol. 5, 1973, pp. 567-575.

SAHI 74 Sahin, K. E. "Message-Based Response Routing with Selcuk Networks," IEEE Transactions on Computers, Vol. C-23, No. 12, pp. 1250-1257, December 1974.

SCHW 72 Schwartz, M., R. R. Bourstyn, and R. L. Pickholtz. "Terminal-Oriented Computer-Communication Networks," IEEE Proceedings, Vol. 60, No. 11, November 1972, pp. 1408-1423.

TORB 74 Torbett, A. E. "Control hierarchy configuration study for a unified worldwide DCS control system," AFOSR Contract F44620-74-C-0026 SCI Project SIIB (task III), 1974.

ULRI 75 Ulrich, Y. L. "Heuristic Algoirthms for Solving a Large Scale Muiticommodity Fiow Probiem on a Network with a Step Function Cost," Studies in Applied Mathematics, Voi. LIV, No. 3, September 1975, pp. 207-227.

WASH 76 Washam, W. B. "Gaspii: GASPIV with Process Interaction Capabilities," M. S., Purdue University, West Lafayette, IN 47907, May 1976.

WHIT 32 Whitney, H. "Congruent graphs and the connectivity of graphs," Am. J. Math. 54, pp. 150-168, 1932.

YAGE 71 Yaged, B., Jr. "Minimum Concave Cost Flows in Certain Networks," Management Science, 14: March 1968, pp. 429-450.

ZIEG 71 Ziegler, J. F. Nodal Blocking in Large Networks, School of Engineering and Applied Science, University of California, Los Angeles, UCLA-ENG-7167, 1971.

Appendix I

1. The AUTODIN System (from [Hous 75])

AUTODIN [AUTO 65] is a message switched store-and-forward data communication system which may be considered as two sections: CONUS AUTODIN in the continental United States, which is managed by a private contractor, and overseas AUTODIN, which is managed by the Department of Defense. The characteristics of the two systems are different, and this study was primarily concerned with overseas AUTODIN. AUTODIN works on the store-and-forward principle, in which the entire message is accumulated at the AUTODIN switch and receipt is verified before initiating further transmission. This represents a fundamental difference from a packetized system such as the ARPANET [ROBE 70] or the projected AUTODIN II, in which the message is broken up into "packets" of uniform size, each packet is treated as an independent message, and the packets are reassembled at the destination. In AUTODIN, the message length can vary from 1 kb to 336 kb with an average of 21.5 kb.

The second feature of overseas AUTODIN, a feature important in any control scheme, is the relatively slow speed of the overseas interswitch links, which transmit data at 2.4Kbits. This means that, in general, the limiting factor in system performance is the interswitch delay, and the system control algorithm will be concerned with control of interswitch routing.

The low transmission rate means that the longest allowable message can take 13 minutes to transmit, and thus the long messages have a disproportionate effect on the delay of the other messages. The present routing scheme involves "lookup" tables stored in every switch. These tables have as many entries as there are switches, each entry labelled with a destination switch, and the entry gives the outgoing link on which messages to that destination switch are sent. At present, these tables are updated by a central control, but usually at long intervals and after computer simulation of the change. The system is unable to respond quickly to sudden variations in loading or to equipment failures. In addition, the system is deliberately underdesigned for the sake of economy, and thus peaks in loads can cause larger delays. The short delay required for important messages is ensured by a priority (precedence) system. These priorities are assigned to messages by subscribers, although not all subscribers can assign the highest-level priorities.

APPENDIX II

III. The FD Algorithm

In section 3.4.1 two conditions were given for the convergence of the FD method. In (i) $\Delta P(\underset{\sim}{f}) \geq 0$ where $\Delta P(\underset{\sim}{f}) \triangleq P(\underset{\sim}{f}) - P(FD\underset{\sim}{f})$. Then

$$\Delta P(\underset{\sim}{f}) \triangleq P(\underset{\sim}{f}) - P(FD\,\underset{\sim}{f}) \cong \Delta\lambda \sum_{K=1}^{M} \ell_K(f_C - v_K)$$

where $\ell_K \triangleq \frac{\partial P}{\partial f_t}$ and $\underset{\sim}{v}$ shortest route flow under matric ℓ_K. Therefore, by condition (i)

$$\sum_{K=1}^{M} \ell_K(f_K - v_K) \geq 0 \qquad \underset{\sim}{v} \in \underset{\sim}{Fa}$$

and by condition (ii)

$$\sum_{K=1}^{M} \ell_K(f_K - v_K) < \varepsilon \rightarrow \underset{\sim}{f} \text{ stationary}$$

where ε positive tolerance.

Phase 1

1. With $\rho_0 = 1$, let $\underset{\sim}{f}^0$ be the shortest route flow computed at $\underset{\sim}{f} = 0$, i.e., with metric $\ell_K = [\partial T/\partial f_K]_{f_K=0}$. Let $n = 0$.

2. Let $\sigma_n = \max_K \left(\frac{f_K^n}{C_K}\right)$.

   If $\sigma_n/\rho_n < 1$, let $\underset{\sim}{f}^0 = \underset{\sim}{f}^n/\rho_n$ and go to Phase 2. Otherwise, let $\rho_{n+1} = \rho_n(1 - \varepsilon\,(1-\sigma_n))/\sigma_n$, where ε is a proper tolerance, $0 < \varepsilon < 1$. Let $\underset{\sim}{g}^{n+1} = \underset{\sim}{f}^n(\rho_{n+1}/\rho_n)$. Go to 3.

3. Let $\underset{\sim}{f}^{n+1} = FD\underset{\sim}{g}^{n+1}$

4. If n = 0, go to 6

5. If $\left|\sum_{K=i}^{M} \ell_K(v_K - g_K^{n+1})\right| < \theta$ and $|\rho_{n+i} - \rho_n| < \delta$, where θ and δ are proper positive tolerances, and $\underset{\sim}{v}$ is the shortest route flow computed at $\underset{\sim}{g}^{n+i}$, stop: the problem is infeasible within tolerances θ and δ. Otherwise, go to 6.

6. Let n = n+1 and go to 2.

Phase 2

1. Let n = 0

2. $\underset{\sim}{f}^{n+1} = FD\underset{\sim}{f}^{n}$

3. If $\left|\sum_{K=i}^{M} \ell_K (f_K^n - v_K)\right| < \theta$, where θ is a proper positive tolerance, stop: $\underset{\sim}{f}^{n}$ is optimal within a tolerance θ. Otherwise, let n = n+1 and go to 2.

In every iteration of the algorithm there is a shortest route computation. Therefore, we have both the flow assignment in the links and the shortest route routing. At the last iteration then we obtain the optimal flow assignment along with the optimal routing which is the shortest route for the optimal flow vector.

Note that in the first step of Phase 1 a shortest route computation is done at $\underset{\sim}{f} = 0$, and $\underset{\sim}{f}^0$ is obtained. With $f_K \to 0$, there are no queues in the network. Then the shortest route π_{ij} minimizes a function of the sum of transmission plus propagation delay. Note also that $\underset{\sim}{f}^0$ computed in this way guarantees that $\underset{\sim}{f} \in \underset{\sim}{Fa}$ since a shortest route flow is an extreme point $\in \underset{\sim}{Fa}$.

112. Shortest Route Algorithm

There are many shortest route algorithms available in the literature [FORD 62], [FRIS 71], [FLOY 62]. The one used here is presented by Frish [FRIS 71].

Let ℓ_K be the weight of link K, and let K be the link connecting node i to node j. Then ℓ_K can be written as $\ell_{(i,j)}$.

1. Assign all nodes i labels of the form $(\cdot,\delta(i))$ where $\delta(s) = 0$ and $\delta(a) = \infty$ for $a \neq S$.

2. Find a link (i,j) such that $\delta(i) + \ell_{(i,j)} < \delta(j)$. If such a link is found, change the label on node j to $(i,\delta(i) + \ell_{(i,j)})$. Repeat this operation until no such branch is found.

3. To identify the nodes in the shortest directed s - a path $a \neq s$:
   a) Let i = a.
   b) Identify node K from the label $(K,\delta(i))$ on vertex i. If K does not exist, then there is no s - a path in the graph.
   c) Let i = K. If i = S, terminate. Otherwise return to (b).

It is proved [FRIS 71] that the shortest path algorithm terminates in a finite number of steps and finds the shortest s - a path, provided that the sum of the link weights along a path is positive.

APPENDIX III

Frisch's connectivity labeling algorithm is used to find a feasible flow path, if any, between a single source and a single destination in each iteration of the labeling process. This algorithm is generalized to find more than one feasible flow path, if muitiple flow paths exist, between a set of sources and a set of destinations in each iteration of the labeling process.

Let a network be denoted as $G = (V,\Gamma)$, and let a set of sources be $S = (s_1,\ldots,s_p)$ and a set of destinations $T = (t_1,\ldots,t_q)$, where p and q are two positive integers. A generaiized version of Frisch's labeling aigorithm is given below. The aigorithm assumes a zero initial flow.

Routine A -- Labeling Process

1. Label $s_1,\ldots,s_p$ by $(s_1,+),\ldots,(s_p, +)$. Now $S = (s_1,\ldots,s_p)$ is + iabeied and + unscanned. Ail other nodes are unlabeled and unscanned.

2. For any node $x \in T$ which is labeled and unscanned:

(a) Label every node y as (x, -) if $(y,x) \in \Gamma$, y is - unlabeled, $f(y,x) = i$, $y \in S$ and

(i) x is + labeled, + unscanned and - unlabeled, or

(ii) x is - labeled and - unscanned.

If in (ii) x is + labeled and + scanned, then add "!" to the label on y. If y is + labeied and + unscanned, then erase the + label.

(b) Label every node y as (x, +) if $(x,y) \in \Gamma$, y is unlabeled, (or y is + labeled and in T), $f(x,y) = 0$, $y \in S$ and

(i) $x \in S$ and x is + labeled and + unscanned, or

(ii) $x \in S$, x is + labeled, + unscanned, and - unlabeled and $f(x,V) = 0$, or

(iii) $x \in S$ and x is - labeled and - unscanned.

If in (iii) x is + labeled and + scanned, then add "!" to the label on y.

(c) In cases a(i), b(i) and (ii), + scan x. In cases a(ii) and b(iii), - scan x.

3. Repeat step 2 until either no more labels can be assigned and no $t \in T$ is labeled or some $t \in T$ is labeled.

(a) In the former case, terminate the procedure.

(b) In the latter case, use routine B to obtain a feasible flow path, if possible. Then, if any further labeling is possible, go to step 2; otherwise, go to step 1.

Routine B -- Flow Change

1. Let z = t and go to step 2.

2. (a) If the label on z is (w, + and

(i) $f(w,z) = 0$, increase $f(w,z)$ by 1 and go to step 3; or

(ii) $f(w,z) = 1$, go to step 5.

(b) If the label on z is (w, -) and

(i) $f(z,w) = 1$, decrease $f(z,w)$ by 1 and go to step 3; or

(ii) $f(z,w) = 0$, go to step 5.

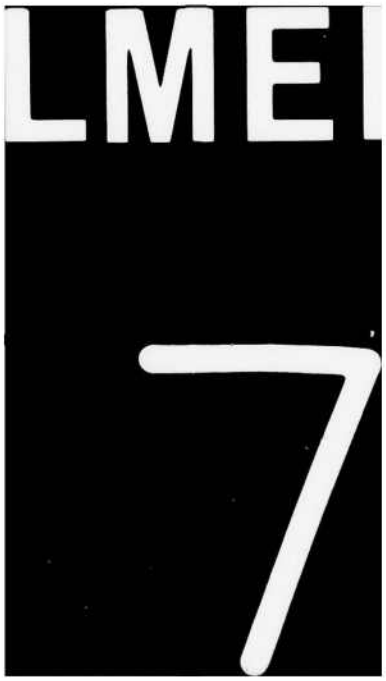
3. If $w \in S$, then return to step 3(ii) of Routine A. Otherwise, go to step 4.

4. If in step 2 the label (w, +) or (w, -) is followed by an "!," then replace the current value of z by w and go to step 2(b). If in step 2 the label is not followed by an "!," then replace z by w and go to step 2.

5. No feasible flow path is possible. Restore the old flow pattern existing before the application of Routine B, and go to step 3(ii) of Routine A.

MISSION
of
Rome Air Development Center

RADC plans and conducts research, exploratory and advanced development programs in command, control, and communications (C^3) activities, and in the C^3 areas of information sciences and intelligence. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.


LME
7