AN ABSTRACT OF THE THESIS OF

Clifford John Warner for the degree of Doctor of Philosophy

in Electrical and Computer Engineering presented on July 27, 1979

Title: ANALYSIS OF NEW PROTOCOLS FOR COMPUTER COMMUNICATION NETWORKS

Abstract approved: Redacted for privacy

John D. Spragins

An advanced forward area tactical radar network, now under conceptual development by the Air Force Electronic Systems Division, can be viewed as a novel form of computer network but with other unique problems resulting from the specialized nature of the application. The proposed network will link together a number of short-range radars, each with associated data processing equipment, so that each radar site has a complete file of tracks for all targets seen by any radar in the network. In addition, the communication network is expected to be used for transmission of a variety of other types of command and control messages.

A new class of adaptive routing protocols for computer communication networks have been developed in this thesis, using the radar network as a basis, but applicable to any computer communication network. These new routing protocols utilize new techniques for searching out and using both reciprocal paths (paths over which messages can travel in either direction) and non-reciprocal paths (paths over which messages can travel in only one direction) for directed message routing. The computations required by the new routing protocols are carried out in a distributed manner at each network node. The only information on network structure which a node needs to store in order to carry out any of the routing computations is the identity of its neighbors.

A GPSS simulation model of a 13 node radar network was used to determine the steady state characteristics of the new routing protocols in an undamaged network, from which a performance model was developed, and to determine the transient characteristics of the new routing protocols while adapting to various cases of network damage. The transient tests indicate that each of the new routing algorithms possess varying degrees of adaptability to network damage. Some of the new routing algorithms were shown to possess the capability to adapt to extreme cases of network damage. Further transient tests indicated that when some of the new routing algorithms are combined with acknowledgement techniques, complete protocols which reliably deliver all messages to their destinations, even following severe network damage, are formed.

The new protocols developed in this thesis are suitable for use in conventional computer communication networks. Overhead comparisons with an ARPA type routing protocol indicate that the new routing protocols developed in this thesis generally require less overhead for large networks with low connectivity.

Analysis of New Protocols for Computer Communication Networks

by

Clifford John Warner

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Completed July 1979

Commencement June 1980

APPROVED:

Redacted for privacy

Associate Professor of Electrical and Computer Engineering

in charge of major

Redacted for privacy

Head of Department of Electrical and Computer Engineering

Redacted for privacy

Dean of Graduate School

Date thesis is presented July 27, 1979

Typed by Clover Redfern for Clifford John Warner

To my wife, Marlys

ACKNOWLEDGEMENTS

I wish to thank my major professor, Dr. John Spragins, for his guidance throughout this thesis preparation. I also wish to thank Mr. Otto Wech for his many suggestions which helped improve the presentation of this thesis.

Most of all, I thank my wife, Marlys, without whose constant support, encouragement, and assistance this work would never have been completed.

This research was supported by the Air Force Office of Scientific Research under grant AFOSR-78-3619.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

Figure Page

LIST OF TABLES

Table | Page

LIST OF SYMBOLS

a_i	Curve fitting parameter.
a_{ij}	The A routing table delay estimate to node j via port i.
b_i	Curve fitting parameter.
b_{ij}	The B routing table entry.
C	Communication channel capacity.
c_{ij}	The C routing table delay estimate to node j via port i.
D_B	Delay estimate to node B.
D_{DM}	Steady state directed message delay.
D_{MSG}	Delay experienced by a message in travelling from its source node.
D_{NDM}	Steady state non-directed message delay.
d_{1j}	Delay Vector Table delay estimate to node j via the non-reciprocal link connecting the intermediate and creation nodes.
d_{2j}	Delay Vector Table intermediate node entry.
d_{3j}	Delay Vector Table creation node entry.
d_{SD}	Path length from node S to node D.
e_{ij}	Delay estimate to node j via node i, determined by a_{ij}, b_{ij}, and c_{ij}.
HN	Handover number, an indicator of the path length from a message's source node.
k_1	Learning constant.
k_2	Forgetting constant.
k_3	Limiting constant.

ℓ_A	Packet size used by acknowledgements.
ℓ_M	Packet size used by regular directed and non-directed messages.
m	Average number of ports at each node in a network.
N(S)	Collection of node S's neighbor nodes.
$N_S(P)$	Collection of all nodes whose shortest path from node S is out of port P.
n	Number of nodes in a network.
OH_A	Total overhead required per update by an ARPA type routing algorithm.
OH_{SBL}	Maximum total overhead required per update by the Simple Backwards Learning algorithm.
$\bar{p}$	Average path length traversed by all directed messages in travelling from their source node to their destination node.
R	Parameter used to equate $\bar{W}$ and $\bar{W}_K$.
T(S)	The distance along the shortest path from node S to the message's destination node.
T_{DV}	Elapsed time between delay vector transmissions at a node.
$T_{ND_{max}}$	Maximum elapsed time allowed between non-directed message transmissions at any node.
T_{NR}	Update time limit for considering the Delay Vector Table entries alive.
T_{PA}	Time out period allowed for acknowledgement reception before a directed message retransmission.
T_R	Update time limit for considering the A, B, or C routing table entries alive.

$T_X(S, P)$	Estimated delay to node S from node X via port P.
$t(S, Y)$	Transit time from node S to its neighbor Y.
$\bar{t}_D$	Inter-generation time for directed messages at each node.
$\bar{t}_{ND}$	Inter-generation time for non-directed messages at each node.
$\bar{W}$	Queue waiting time averaged over all queues, computed directly from simulation data.
W_i	Waiting time in queue i.
$\bar{W}_K$	Queue waiting time averaged over all queues, computed from equations based on Kleinrock's independence assumption.
$\bar{W}_{SD}$	Average queue waiting time in the path from node S to node D.
γ	Per node offered traffic rate for directed and non-directed messages.
ρ_i	Utilization of communication channel i.

GLOSSARY OF TERMS

ACK	A positive acknowledgement.
ARPANET	The Defense Department's Advanced Research Projects Agency experimental computer network. This term is abbreviated to ARPA when its meaning is made clear by context.
GPSS	The General Purpose System Simulation language.
learning process	Updating the routing table delay estimates to bring them closer to the current message's value of D_{MSG}.
forgetting process	Causing the routing table delay estimates to increase.
NAK	A negative acknowledgement.
RPS	The Reciprocal Path Search routing algorithm.
RPSDV	The Reciprocal Path Search Algorithm with Delay Vectors.
RRPS	The Rapid Reciprocal Path Search routing algorithm.
RRPSDV	The Rapid Reciprocal Path Search Algorithm with Delay Vectors.
SBL	The Simple Backwards Learning routing algorithm.

ANALYSIS OF NEW PROTOCOLS FOR COMPUTER COMMUNICATION NETWORKS

I. INTRODUCTION

This thesis is primarily concerned with the development and analysis of a new class of protocols for computer communication networks. The protocols are developed to meet the requirements of a new type of computer communication network, a radar network, which is now under conceptual development by the Air Force Electronic Systems Division. Although the radar network has served as the basis for the protocols' development, the usefulness of the protocols for other networks is also investigated.

This chapter introduces the precise form currently visualized for the radar network motivating this study. Chapter II of this thesis surveys the major existing routing protocols for computer communication networks, and discusses their suitability for use in a radar network. Chapter III discusses the techniques available for analyzing routing protocols, and describes a GPSS simulation model of the radar network which was created to analyze the new routing protocols developed in this thesis. In Chapter IV, the new routing protocols will be presented, along with their performance characteristics, which are derived from the GPSS simulation data. Chapter V discusses acknowledgement protocols which are suitable for use in the radar network. The performance characteristics which result when the new routing protocols are combined with acknowledgement protocols are presented. Finally, Chapter VI discusses the suitability of the new protocols developed in this thesis for use in conventional computer communication networks.

Characteristics of Computer Communication Networks

A common definition of a computer communication network is that it is a collection of nodes, at which reside computing resources, and a set of communication channels which the nodes use to communicate with one another (1, 2, 3). Sharing of resources is accomplished by the transmission and reception of messages. The nodes consist of host computer and terminal resources. These form the "user-resource" subnetwork. More than one host computer may reside at each node. Terminals may be either local to a host computer, or they may be remote, or a terminal need not be permanently associated with any host computer at all. The computing resources which reside at each node are normally connected into the network through special purpose communication processors (2, 3). The communication processors along with the communication links form the "communication" subnetwork. The communication subnetwork may also contain some nodes whose only function is message handling. The user-resource subnetwork together with the communication subnetwork comprise the entire computer communication network.

The computer communication network is generally defined by the following features: its host computers and terminals, communication processors, topological layout, communication equipment and transmission media, switching technique, and protocol design (1, 4). These features are chosen to accomplish the particular function of the computer communication network subject to specified performance requirements. The performance measures most commonly quoted include message delay, message throughput, reliability, cost, and security (3, 4, 5).

Message delay can be defined as the elapsed time between transmission of the first bit of the message at its source node, and the successful reception of its final bit at its destination node (thus the delay includes the message transmission time). Throughput can be defined as the average number of information bits successfully delivered at the destination node per unit time. A computer communication network can be considered reliable if it possesses certain qualities, such as the capability to adapt to changes in network structure and the ability to deliver messages over noisy communication channels. Providing a multiplicity of paths between nodes is often used to maintain the network's viability in the event of link or node failures. Quantitative reliability measures exist which depend strongly upon the topological layout of the communication links in addition to the reliability and availability of the individual computer systems and communication facilities (6). The cost of a network is determined by the requirements for network resources such as processing speed and communication channel bandwidth, and by the manner in which these requirements are satisfied (technologies used, etc). Security relates to the availability of the network resources to unauthorized users and to the susceptibility of the network to disruption by hostile forces. Techniques for maintaining network security include using confidential user identification codes, encrypting the network messages, and, depending upon the type of communication links used, the use of various schemes for preventing interference with message transmissions such as spread spectrum or code-division multiple access (CDMA) techniques or narrow beam antennas (7).

Characteristics of the Radar Network

Several initial studies have already been completed on the radar network (8, 9, 10). This thesis will use the results of these earlier studies to define the radar network's function, host computers, communication processors, topology, communication equipment and transmission media, and switching technique. These results are presented in this introductory chapter. The presentation of the protocols is reserved for later chapters.

Function

The radar network consists of a grid of short-range radars which communicate with each other to share information about target tracks. A possible configuration of the grid is pictured in Figure 1.1. A primary purpose of the network is to allow the radar sites (network nodes) to share track reports so that each radar site will be able to maintain a complete file of all target tracks generated any where within the network.

In addition to the radar sites, the network will contain command and control and other specialized nodes. These network nodes will transmit and receive specialized messages, using the radar network as their communication medium, even though their messages may not be directly related to the primary network function of allowing the radar sites to maintain complete files of target tracks.

The function of the communications portion of the radar network, then, is to facilitate the transmission and reception of track reports, as well as other specialized directed messages carrying command and control and other information. This must be accomplished reliably, both when the network undergoes planned changes in structure, which will probably be infrequent, as well as when damage of the network nodes and communication facilities results from hostile actions. Jamming, which often results in the prevention of

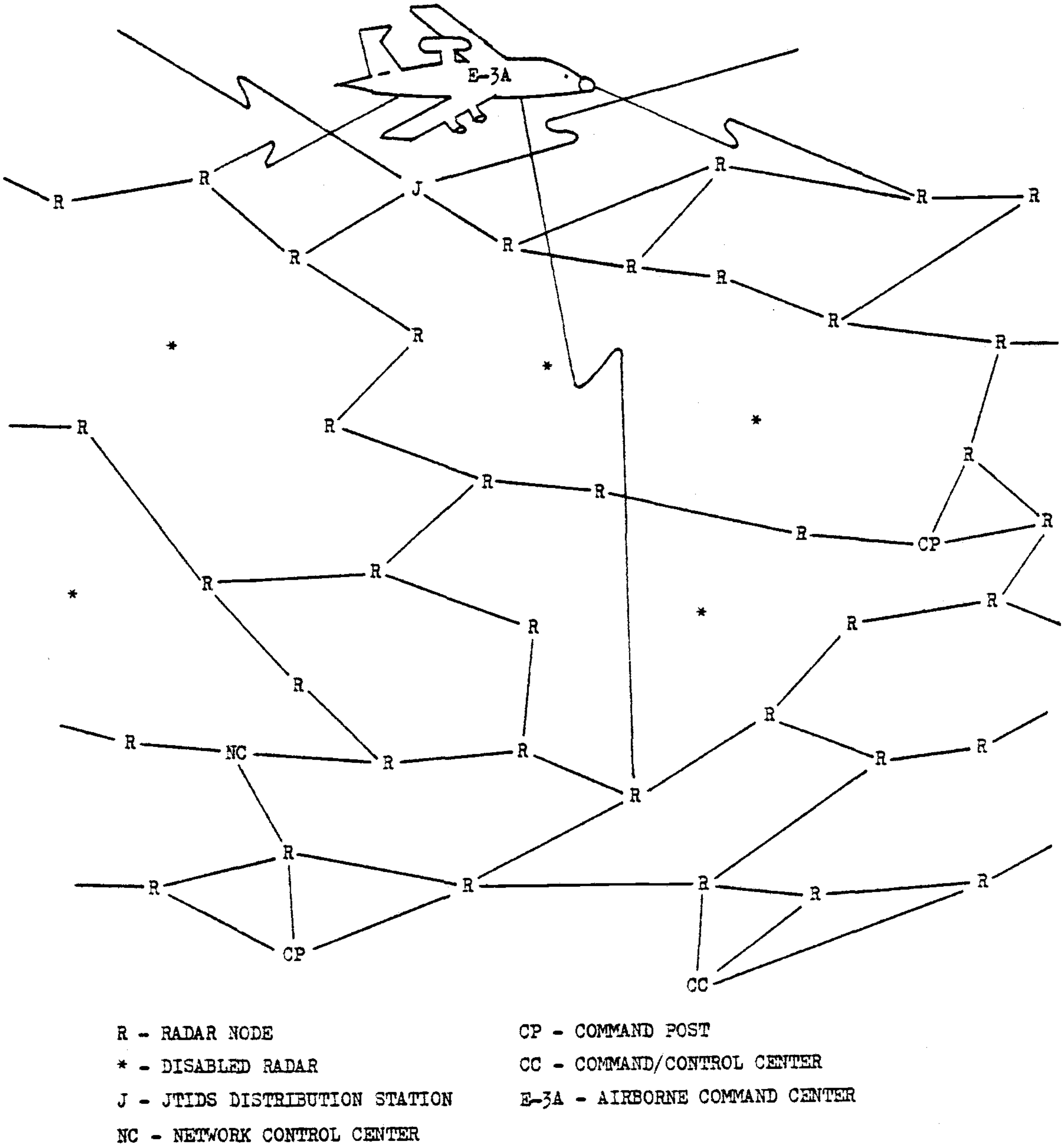

E-3A
J
R
NC
CP
CC
*
R - RADAR NODE
* - DISABLED RADAR
J - JTIDS DISTRIBUTION STATION
NC - NETWORK CONTROL CENTER
CP - COMMAND POST
CC - COMMAND/CONTROL CENTER
E-3A - AIRBORNE COMMAND CENTER

Figure 1.1. Possible form of network with additional types of nodes.

transmission of information in a single direction over a communication link, should be expected. Despite such network changes, messages should be delivered to their destinations with a minimum of delay. This is especially true of the track report messages, since these reports will likely be used for real time tracking.

Host Computers

Each radar site will contain host computers which will, upon sighting a target, employ sophisiticated data processing techniques to generate a target track. Each radar site will be capable of utilizing the tracks it generates from its own sightings, together with the target track reports it receives through the communication network, to update its own track file and to generate new track reports.

Tracking algorithms which utilize information from all radars in an optimum manner probably will not be feasible in a radar network of this type. Rather than sending complete information on all target detections to a central location which will combine the reports to generate target tracks, tracking will be done at each radar location using the information available there. The information used will be track information received from other radar sites, plus any information generated at the particular location itself. This distributed form of tracking is necessary to ensure reliability of the network. It is expected that the radar network will at times by subjected to intentional damage, and the destruction of a centralized tracking site would bring the tracking function to an abrupt halt. This distributed form of tracking, however, would allow the tracking function to continue even with widespread destruction of radar sites.

In order to ensure that each tracking radar is able to utilize information from other radars to generate more accurate tracks than would be possible without shared information, a rather formal data sharing procedure has been proposed. The suggested procedure gives a compromise between the tremendous communications and data processing requirements which would be necessary with complete sharing of all relevant information from every single radar seeing a target, and the poorer quality tracks which would be obtained without sharing. The current concept involves a subset of the radars seeing any given target to form a local group which is in charge of tracking that particular target. Members of the local group share complete information on their detections of their particular target. When any member of the local group determines that it has information indicating a new track report is needed, it generates one and disseminates it throughout the network by means of some of the communications algorithms discussed in this thesis.

Communication Processors

A communication processor will be required at each node to interface between the host tracking processors and the communication subnetwork. The communication processor will perform tasks such as attaching addressing and control information to each message packet, attaching check sum bits for error detection, storing unacknowledged messages, and applying routing and other message handling algorithms.

Topological Layout

Reliability and survivability considerations dictate that the communication network assume the form of a distributed grid whereby each node has multiple paths to the other network nodes. The nominal grid could be basically rectangular, diamond, or hexagonal. It is likely, however, that a largely random configuration similar to that in Figure 1. 1 will actually exist in a tactical situation.

Communication Equipment and Transmission Media

The types of communication links which will be used between nodes could be any of the following: microwave, troposcatter, fiber optics, or satellite communication links. A likely choice will be microwave radio links using narrow-beam antennas since this type of link seems to be feasible and to have definite electronic countermeasure (ECM) advantages. Each communication link would have separate transmitting and receiving antennas. This would provide for bi-directional simplex transmission of messages between nodes (a detailed explanation of simplex transmission is contained in reference 11).

This approach implies that the two directions of transmission will be treated independently, since it is entirely possible only one direction will be operational.

Switching Technique

The basic switching technique used will be a form of packet switching. Target tracks will be transmitted over the bi-directional simplex communication channels in fixed size data blocks, probably approximately 300 bits in length (59). Other miscellaneous types of specialized transmissions may use varying sizes of data blocks, although there is some possibility of fitting most, if not all, of these into the same size packets. The track reports transmitted by the local group members, which are sent to all other members of the network, are by far the most common members of a class of transmissions which will be called non-directed transmissions, since they are not really directed toward any particular node, and are expected to disseminate the same information to all nodes. Some of the track reports sent out by the members of each local group, as well as the majority of the other messages besides track reports, will probably be directed messages sent from one particular node to another. It may be extremely important to ensure the correct reception of some of these directed messages with a minimum of delay.

When messages are sent in the form of packets, there is a fundamental tradeoff between low delay and high throughput (5). Low delay calls for a small packet size to cut transmission time, and short queues to keep queueing delay to a minimum. In contrast, high throughput calls for large packets to decrease overhead, and long queues to provide sufficient buffering and full circuit utilization. Low delay for some types of messages is extremely important in the radar network, so queues should be kept small and the packets reasonably short. However, the protocols must ensure that the messages are delivered. Lost messages will decrease throughput, and in this respect throughput is very important.

II. ROUTING PROTOCOLS FOR COMPUTER COMMUNICATION NETWORKS

In a packet switching computer communication network, there must be some protocol which determines at each node on which link or links an incoming packet or message should be forwarded, if it has a destination other than the node which has just received the message. The choice of the routing protocol which should be used depends on the network geography, and the nature of the communications processes for which it is designed. How well the routing protocol operates will affect the message delay, throughput, and other feactors such as network congestion (12).

Routing protocols can be divided into two categories, deterministic and adaptive (13). A deterministic protocol is time invariant, and does not adjust to varying network conditions. The routes in a deterministic protocol are fixed. An adaptive protocol, on the other hand, is time varying and can adjust to variations in network characteristics such as uneven congestion of the network communication links or changes in the network connectivity.

Besides being either deterministic or adaptive, routing protocols can be classified according to whether the routing protocol requires global or local knowledge of the network structure; whether the routing computation is performed at a single node (centralized), or at each individual node (distributed); and whether the objective of the routing protocol is to minimize the total average delay (system optimizing), or to minimize the individual source-destination delays (user optimizing) (13).

The needs of the radar network motivating this research require that the routing protocol be adaptive since it is expected that the network will be subject to both voluntary and involuntary changes in network structure. The voluntary changes will arise as a result of mobile network nodes wishing to enter, leave, or reposition themselves within the network grid. The involuntary network changes can be a result of hostile actions of a foe, resulting in the jamming of communication links, and the destruction of nodes. Involuntary loss of communication links can also result from equipment failure.

The routing protocol needs to be distributed also because of the potential loss of nodes by enemy actions. If the routing computations were carried out at a single centralized node, the destruction of this node could bring the routing of messages to a halt, or at least the routes would thereafter remain fixed. However, if the routing computations are carried out at each individual node, adaptive routing would continue even with widespread destruction.

Distributed and Adaptive Routing Protocols

Flooding

Flooding can be used to route either non-directed or directed messages. With flooding, a packet is sent by each node to all of its neighbors, except possibly the one from which it was received. It is easy to verify that with this algorithm, a packet will be delivered to all network nodes. Flooding ensures that a message will arrive at each node over the quickest route which is available under the message loading conditions imposed with this algorithm. This speed, however, is achieved at the cost of generating extra traffic.

In a comparative study on adaptive routing algorithms for directed messages (14), Boehm and Mobley suggest that flooding generates so many extra copies of a message that it is extremely wasteful of network resources and therefore a poor choice for directed message routing. They do point out, however, that it has excellent adaptability. By the nature of the algorithm, flooding adapts instantly to any changes in network structure, which is important for a radar network.

Barber and Davies suggest that flooding be used as a path finding method (12). They suggest that when a route is to be established from node S to node D, that a short "path finder" packet be sent from S by flooding, and each packet generated in the "flood" record the route it takes. Then the first packet to reach destination D has a record of the optimum route. They suggest that D return this route information to S, and S can then use this information to route later messages to D.

Random Routing

Random routing is primarily useful for directed message routing. Each node that receives the message sends it out of a randomly selected port. The message makes a "drunkards walk" around the network until it eventually arrives at its destination. While simple to implement, this algorithm allows the message to take longer than necessary to reach its destination. Allowing unnecessary delays also puts an unnecessary burden on the network resources by increasing congestion. Since low message delay is very important in the radar network, random routing does not seem to be a viable candidate for use in it.

Shortest Path Routing

This protocol is useful for routing both directed and non-directed messages. With this protocol, each node stores a complete "map" of the entire network which it uses to determine the shortest path to each other node in the network. Two of the drawbacks of this type of protocol are that for large networks, such as a radar network, much memory may be needed to store the map, and, probably more important, the processing time required to compute the shortest path to each node can be very long. This is an important drawback in an environment where changes in network structure will be frequent, as in a radar network, since the long computation times which may be required to compute all the new shortest paths will degrade the algorithm's adaptability.

Boehm and Mobley include the shortest path routing protocol in their study. They proposed a dynamic programming algorithm be used to compute the shortest paths. The dynamic programming principle which they used is based on the fact that the optimal path from node S to node D has the property that, for any node Y along the path, the remaining path is optimal from Y to D. They define N(S) as the set of neighbors of S, and

$$T(S) = \min_{Y \in N(S)} (t(S,Y)+T(Y)) \tag{2.1}$$

where

T(S) = the distance along the shortest path from node S to the destination node.

t(S, Y) = the transit time from node S to its neighbor Y.

Equation (2.1) presents an iterative scheme for computing T(S) for all S, from the starting condition T(D) = 0.

Whenever any change in network structure occurs, this protocol assumes that these network changes can be sensed by neighboring nodes. The neighboring nodes can communicate the changes to all the other nodes. Each node can then update its network map, and compute new routes for messages. Boehm and Mobley suggest that this technique would work well for small networks, but they admit that the algorithm would probably not work well in a large network. This is a serious drawback in a radar network, which will probably contain on the order of 50 nodes or more. The problem arises from the long computation time which would be required to compute all the new paths. To alleviate this computation time problem, they suggest that a technique devised by Butrimenko (15) would be helpful. The Butrimenko technique is a pertubative technique which relies on the condition that small numbers of changes in the network will not greatly change the ideal routing tables. This technique is rather complex, and is described in detail in reference 14. Boehm and Mobley claim that this works well, although they present no supporting data. They do point out, however, that for the link addition process, the Butrimenko technique requires a good deal of transmission of modifications and produces some temporarily misleading information. They point out that these difficulties are not serious if link addition is a relatively rare event. This is a poor assumption, however, in a radar network since it likely will contain mobile nodes.

Sussman, et al. (7), propose a technique by which the shortest path algorithm can be used to route non-directed messages. They suggest that a non-directed message be routed as a concatenation of directed messages. Suppose that node S originates a non-directed message. Node S first computes the shortest path to all other nodes individually, and determines which output port corresponds to each of these routes. It then groups the destination nodes according to which port should be used to reach them. Let $N_S(P)$ be the collection of all nodes whose shortest path from S is out of port P. Then for each port, P, node S sends a single message out of P appended with the addresses of each member of $N_S(P)$. When node S's neighbor, Y, receives the concatenated message, it performs a similar procedure to further route the message. The neighbor, however, considers only the appended addresses for membership in each $N_Y(P)$. This algorithm should have essentially the same advantages and disadvantages as the shortest path algorithm.

There is nothing about the Sussman technique which restricts it to use with a shortest path algorithm. Any directed message routing protocol which determines a single port for routing directed messages could be used.

Minimum Spanning Tree

Sussman, et al., suggest that a minimum spanning tree algorithm could be used to route non-directed messages. Whereas the shortest path algorithm computed the shortest path to each network node, the minimum spanning tree minimizes the sum of the paths required to reach all other nodes. This algorithm has the desirable feature that it keeps the number of retransmissions of each non-directed message to a minimum. The algorithm which Sussman, et al., propose for computing the minimum spanning tree is as follows:

1. Connect any isolated node to its nearest neighbor.
2. Repeat the procedure connecting the partial network to the nearest isolated node.
3. Continue the procedure until all nodes of the network have been connected.

The final minimum spanning tree is independent of the sequence of connecting nodes (7). This requirement is essential if the algorithm is to be computed in a distributed manner at individual nodes.

When routing a non-directed message with the minimum spanning tree technique, each node sends the message out of all links which belong to the minimum spanning tree, except the one on which it was received.

This algorithm should have some of the same disadvantages as those which have been listed for the shortest path algorithm. For large networks, much memory is needed to store a complete map of the entire network. Also, mechanisms for sensing changes in network structure and for disseminating this information throughout the network so that each node can update its map need to be established and tested. This could be a very difficult problem in a dynamic environment where changes in network structure occur rapidly. This algorithm has not been tested, and therefore no performance data is available.

Backwards Learning

Backwards learning was first proposed by Baran in 1964 as a technique for routing directed messages (16). Boehm and Mobley tested it extensively in their study, which was previously mentioned in the section on flooding, and Pickholtz and McCoy studied it in a paper published in 1976 (17).

Backwards learning is one member of a class of routing protocols in which routing is accomplished by storing at each node a table of the current estimates of the times required to reach each possible destination when leaving via each output port. The table is consulted for each message arriving at the node to determine which output port is the best to route the message to its destination. The message is then sent out of this port. Baran termed this "hot potato" routing. This routing protocol is only as good as the information contained in the routing table, and therefore how the routing table is updated at each node is critically important. The usual technique is to use information contained in incoming directed messages to update the routing tables. Any routing protocol which updates the routing tables with information contained in incoming messages is called a stochastic routing algorithm (14, 17). Backwards learning is the earliest of stochastic routing protocols.

With the version of backwards learning tested in the study by Boehm and Mobley and the study by Pickholtz and McCoy, each incoming message, in addition to the message content, contains the destination node address, D, the source node address, S, and a handover number (HN). The handover number is a tag which is set to zero upon initial transmission of the directed message at the source node. Every time the directed message is passed on at a node, the handover number is incremented. The handover number, then, is an indicator of the path length from the source node.

The value of HN in a message received in port P estimates the time currently required to travel from the source S to the current node, X, via port P. The backwards learning concept assumes that HN is, therefore, a good estimate of the delay required to reach node S from node X, when the message is sent out of port P (hence the name backwards learning). Therefore, with the version of backwards learning tested in the aforementioned studies by Bohem and Mobley and by Pickholtz and McCoy, whenever a message arrives from node S, the routing table entry, $T_X(S, P)$, is updated using a weighted average:

$$T_X(S, P)' = T_X(S, P) + k_1(HN - T_X(S, P)) \tag{2.2}$$

where

$T_X(S, P)$ = the estimated delay to node S from node X by way of port P.

k_1 = the learning constant.

Thus, the new delay estimate, $T_X(S, P)'$, is a certain fraction k_1 $(0 < k_1 \leq 1)$ of the way between the old estimate and the current message's HN. (Values of k_1 outside of the range 0 to 1 result in $T_X(S, P)'$ falling outside of the range between the old estimate and the current message's HN.)

The studies mentioned found that backwards learning, as well as other similar forms of stochastic routing, adapted poorly to changes in the network structure. This is a direct result of using the routings of directed messages to update the routing tables. These routings tend to be fixed unless some extra algorithms to force at least a fraction of the messages to explore alternate routes are introduced.

Another problem with the backwards learning algorithm is that it assumes reciprocity of the communication paths. That is, it assumes that if travel is possible in one direction over a communication path, it is possible in the opposite direction as well (in fact, the algorithm even assumes that the congestion in the two directions is similar). The radar network will, however, exist in an environment where jamming, disabling a communication link in only one direction, will be common. Any routing algorithm under consideration for use in the radar network needs to be thoroughly tested to determine its adaptability to this kind of network damage.

ARPA Routing Protocol

This protocol is used to route directed messages. It was included in the comparative study by Pickholtz and McCoy.

The ARPA network routing algorithm also uses a routing table which is stored at each node. However, instead of updating the routing table with information contained in the incoming messages, the ARPA routing algorithm uses an exchange of "delay vectors" among the nodes to provide information for updating the tables. At set intervals, each node sends to all of its neighbors a list which contains its estimates of the shortest delays from it to all the other nodes. This list, a delay vector, thus contains a number of entries which is one less than the number of nodes in the network. Upon receiving the delay vector from one of its neighbors, a node adds to each delay estimate a measure of the current delay in traveling from itself to the neighbor from which the delay vector was received, thus forming a new list. This new list then provides that node with an estimate of the minimum delay required to reach all destinations if the message is sent out of the port connecting to the neighbor. The routing table for the node is then constructed by combining the delay estimates derived from the delay vectors received from all of its neighbors. The best port for sending a message to node D is the port whose routing table entry for node D is minimum.

The original ARPA routing protocol was designed in 1969. Since then it has undergone various improvements. There is an excellent discussion of the development of this routing protocol in reference 18.

The study by Pickholtz and McCoy indicated that the ARPA routing algorithm does a good job of adapting to the loss of communication links when such loss prevents transmission of messages over the link in either direction. The case of allowing transmission in one direction, but not the other, was not tested. How well it adapts to the latter case of link damage needs to be answered if the ARPA routing protocol is seriously considered for use in the radar network.

Summary

The major routing protocols which seem to be likely candidates for use in the radar network have been discussed. These are flooding, random routing, shortest path routing, minimum spanning tree routing, backwards learning, and the ARPA routing protocol.

Flooding can be used to route either non-directed or directed messages. It has excellent adaptability. However, flooding generates so many extra copies of a message that it would be extremely wasteful of network resources if used for directed message routing.

Random routing is primarily useful for directed message routing. While simple to implement, this algorithm allows the message to take longer than necessary to reach its destination. Since low message delay is very important in the radar network, random routing does not seem to be a viable candidate for use in it.

Shortest path routing can be used for routing both directed and non-directed messages. With this protocol, each node stores a complete map of the entire network which it uses to determine the shortest path to each other node in the network. Two of the drawbacks of this type of protocol are that for large networks, such as a radar network (which will probably contain 50 or more nodes), much memory may be needed to store the map, and the processing time required to compute the shortest path to each node can be very long. The long computation time which may be required to compute all the new shortest paths degrades the algorithm's adaptability. Thus, the shortest path algorithm may work well for a small network, but the algorithm would probably not work well in a large network such as a radar network.

The minimum spanning tree algorithm could be used to route non-directed messages. Whereas the shortest path algorithm computes the shortest path to each network node, the minimum spanning tree minimizes the sum of the paths required to reach all other nodes. With this algorithm, each node also needs to store a complete map of the entire network, which it uses to determine the minimum spanning tree. This algorithm has the desirable feature that it keeps the number of retransmissions of each non-directed message to a minimum. However, for large networks, much memory is needed to store the complete map of the network and mechanisms for sensing changes in network structure and for disseminating this information throughout a network which exists in a dynamic environment need to be established and tested.

The backwards learning algorithm is a technique for routing directed messages. Backwards learning is one member of a class of routing protocols in which routing is accomplished by storing at each node a table of the current estimates of the times required to reach each possible destination when leaving via each output port. The table is consulted for each message arriving at the node to determine which output port is the best to route the message to its destination. A study by Boehm and Mobley and a study by Pickholtz and McCoy (14, 17) tested a version of backwards learning which used information attached to the directed messages to update the routing tables. These studies found that backwards learning adapted poorly to changes in network structure. This is a direct result of using the routings of directed messages to update the routing tables. These routings tend to be fixed unless some extra algorithms to force at least a fraction of the messages to explore alternate routes are introduced.

The ARPA routing protocol is used to route directed messages. This routing algorithm also uses a routing table which is stored at each node. However, instead of updating the routing table with information attached to the incoming messages, the ARPA routing algorithm uses an exchange of "delay vectors" among the nodes to provide information for updating the tables. The study by Pickholtz and McCoy indicated that this algorithm does a good job of adapting to the loss of communication links when such loss prevents transmission of messages over the link in either direction. The case of allowing transmission in one direction, but not the other, was not tested.

In Chapter IV of this thesis, new routing protocols which have been developed with the radar network as a basis will be presented. Performance data on these new protocols will be presented, with emphasis on the adaptability of the protocols to cases of link damage.

Unlike previous studies on routing protocol adaptability, the case of allowing transmission of packets in one direction over a link, but not in the other, will be studied in depth.

Prior to presenting the new routing protocols, however, Chapter III will present the major tools which are available for analyzing the performance of routing protocols. Chapter III will explain why simulation was chosen to analyze the new routing protocols, and describe the GPSS model of the radar network which was created for this purpose.

III. ANALYSIS TECHNIQUES FOR ROUTING PROTOCOLS

There are well known analytical techniques for analyzing and modeling packet switched computer communication networks. These techniques minimize the packet delay, within some cost constraint, by manipulating the channel capacities, the network topology, and the message routing procedure (3, 19, 20, 21). However, all of these references assume a fixed routing procedure, which makes them unsuitable for analyzing adaptive routing algorithms. At the present state of the art, the only reasonably accurate technique for evaluating the performance of adaptive routing algorithms is computer simulation (3, 13, 22).

A GPSS Simulation Model of the Radar Network

A GPSS simulation model of the radar network has been written in order to evaluate the adaptive routing and acknowledgement protocols developed in this thesis. The program models a 13 node network, as shown in Figure 3.1. This particular configuration was chosen because it is hexagonal in nature (the nominal configuration currently favored by Air Force personnel working on initial network studies), it contains several alternate paths between nodes (thus providing sufficient capacity for allowing the changes in routes necessary to adapt to

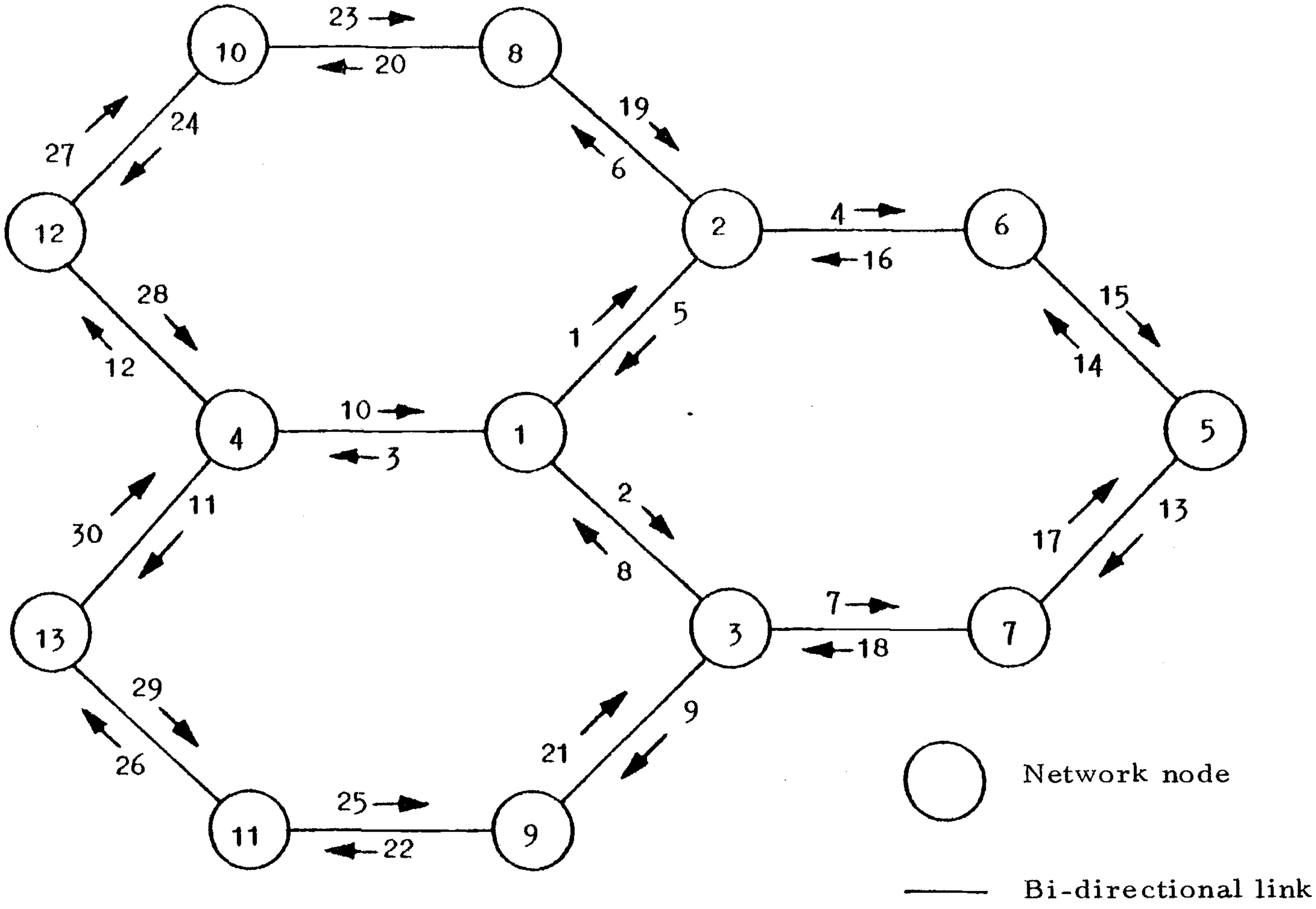


Figure 3. 1. Thirteen node GPSS network.

varying network conditions), and it is small enough to permit reasonable computing costs.

Each node in Figure 3.1 has been assigned an identifying number. Also, each direction of each bi-directional link has been assigned a unique identifying number. These node and link identifiers will be referred to in some of the discussions in Chapter IV.

Each node in the network generates both directed and non-directed messages. The inter-generation times for these two types of messages are exponentially distributed, with per node means of $\bar{t}_D$ and $\bar{t}_{ND}$ respectively. Both the directed and non-directed messages are generated with their source nodes uniformly distributed throughout the network. The destinations for the directed messages are also uniformly distributed throughout the network (i.e., a uniform traffic matrix has been assumed).

Each node receives from each of its neighbors over the communication lines messages which are processed along with the messages which the node generates itself. These messages are handled in the order in which they arrive, have the appropriate routing algorithm applied to them, and are queued for transmission over the appropriate communication channel. The queued messages are served by the communication channels in an order dictated by message priority, with a first come first served ordering within each priority class. All messages except for acknowledgements were assumed to be in the same priority class in the simulations discussed in this thesis. The acknowledgements were given a higher priority.

The communication channels are all assumed to have the same capacity, C. The channel capacity is the number of bits which can be transmitted over the channel per unit of time. The time required to transmit a single message over a single channel is thus equal to its length divided by the channel capacity. It is assumed that all the track report messages and the specialized command and control messages can fit into the same size packet. The length of this packet, ℓ_M, is approximately 300 bits. Acknowledgements are assumed to fit into a smaller size packet. The length of this packet, ℓ_A, is assumed to be one-tenth of ℓ_M, or approximately 30 bits. Therefore, the time required to transmit the track reports and specialized command and control messages over a communication channel is ℓ_M/C, and the time required to transmit acknowledgements across the communication channels is ℓ_A/C, which is one-tenth ℓ_M/C.

Initially, the simulation program is used to evaluate the routing algorithms alone. These results are presented, along with descriptions of the routing algorithms considered, in Chapter IV. The performance characteristics which result when acknowledgements are used in conjunction with some of the better routing algorithms are presented in Chapter V.

In order to evaluate the routing and acknowledgement protocols, the simulation program gathers various statistics. The primary statistics of value are:

1. Communication channel utilization.
2. Average queue length.
3. Average directed message delay.
4. Average non-directed message delay.
5. The number of successful directed message receptions at a destination node.
6. The number of directed messages lost enroute to their destinations.
7. The number of directed messages which have looped.
8. The number of retransmissions of a directed message at a source node as a result of not receiving an acknowledgement.

The communication channel utilization and queue length are measures of the network congestion. High utilization of the communication channels can produce long queues, which in turn cause long delays.

Steady state equations relating channel utilization and queue length for packet switched computer communication networks have been developed by Kleinrock (3, 19). Data on the steady state values of these statistics can be used to determine if the techniques developed by Kleinrock apply to the radar network simulation results. These steady state statistics can also help create a performance model of the network.

The average directed message delay is computed by averaging the elapsed time between when the directed message is originally transmitted at its source node and when it is successfully received at its destination node. This is an average over all directed message transmissions. Steady state values of this statistic can be used to assist in creating the performance model for the network, and transient values of this statistic can be used to evaluate the adaptability of the routing and acknowledgement protocols to various cases of network damage.

A weakness of the average directed message delay statistic computed in this manner is the fact that only the messages which reach their destinations are averaged in. Lost messages are not included in the average. The simulation results have shown that following network damage this average is weighted in favor of those source-destination pairs which are close together, resulting in a misleadingly small value for the average delay. In order to compensate for this, a penalty for each lost message can be included in the average. The size of the penalty (in cases where it has been included) was chosen to be $24(\ell_M/C)$, which is twice what the delay would be if message, after departing the source node, visited the other 11 nodes before it finally reached its destination. This new quantity, the adjusted directed message delay, is often a better indicator of the delay performance of the routing algorithms. Lost messages do not cause the same type of problem, however, when acknowledgements are used with retransmissions of messages when no acknowledgement is received. After one or several transmissions, each directed message will eventually reach its destination (assuming routing is possible and the routing algorithm does adapt, as will be the case in the situations where acknowledgements are used in this thesis), and be included in the average.

The average non-directed message delay is computed by averaging the elapsed times between the instant when the original transmission of the non-directed message at its source node occurs and the instants when it is first received at each of the other nodes. Thus, for the 13 node network, each non-directed message is received at each of 12 nodes, and hence provides 12 entries into the average. Lost messages are not a problem with this statistic since, as will be shown in Chapter IV, the non-directed message routing algorithm used in this thesis adapts perfectly to changes in network structure and never allows messages to be lost (if the network remains connected so that communication between any two nodes is possible).

Transient values of the statistic on directed messages delivered are an important tool for evaluating how fast and how completely the routing algorithms adapt to changes in network structure.

Transient statistics on the number of directed messages lost and looping are also valuable for this purpose.

When the acknowledgement protocols are evaluated, statistics on source node retransmissions of directed messages will replace statistics on lost messages. When acknowledgements are included in the model, messages are never really lost out of the network, since copies of all messages are stored at their source nodes. Assuming that sufficient time is allowed for acknowledgements to be returned to the source node, the retransmission of a directed message is an indicator of the routing protocol's inability to deliver messages.

The statistic on directed message retransmissions can be used to evaluate how much time should be allowed for the return of the acknowledgement. If this time period is too short, retransmission will occur even when all messages are being delivered. Use of an overly short time out period can thus severely congest the network. An overly long time out period will cause unnecessary directed message delays.

The statistics on directed messages delivered, lost, looping, and retransmitted have been normalized for presentation in Chapters IV and V by dividing them by the total number of directed message transmissions. Thus, an optimal value of the normalized statistic on messages delivered is one and an optimal value of the normalized statistic on lost messages is zero.

Summary

A GPSS simulation model of the radar network has been written in order to evaluate the adaptive routing and acknowledgement protocols developed in this thesis. The program models a hexagonal grid of 13 nodes connected by bi-directional simplex communication links.

Each node in the network generates both directed and non-directed messages. Both the directed and non-directed messages are generated with their source nodes uniformly distributed throughout the network. The destinations for the directed messages are also uniformly distributed throughout the network.

The communication channels are all assumed to have the same capacity, C. It is assumed that all the track report messages and the specialized command and control messages can fit into a packet of length ℓ_M, which is about 300 bits. The acknowledgements are assumed to fit into a packet of length ℓ_A, which is assumed to be one-tenth of ℓ_M, or about 30 bits.

In order to evaluate the routing and acknowledgement protocols, the simulation program gathers statistics on the communication channel utilizations, the average queue lengths, the average directed and non-directed message delays, and the number of directed messages delivered, lost, looping, and retransmitted. Steady state values of some of these statistics will be used to develop a performance model of the network. Transient values of some of these statistics will be used to evaluate the adaptability of the routing and acknowledgement protocols developed in this thesis.

IV. A NEW CLASS OF ROUTING PROTOCOLS

Two types of routing algorithms are presented and analyzed in this chapter. The first is for routing non-directed messages, and the second for routing directed messages. Directed message routing algorithms are the primary focus of this thesis. This chapter will evaluate how well these routing algorithms perform for the radar network. Extensions of the directed message routing algorithms for use in conventional computer communication networks will be given in Chapter VI.

Non-Directed Message Routing Algorithm

The favored routing algorithm for non-directed messages is a form of flooding whereby each node which received a non-directed message relays the transmission on to all neighboring nodes except the one from which the message came. It is easy to verify that this algorithm will quickly disseminate such messages to all nodes to which communication paths exist.

By the nature of the algorithm, flooding adapts instantly to network damage, communication link jamming, node additions and deletions, and other changes in network structure. Flooding requires no knowledge of the network structure and requires essentially no processing.

The only drawback of flooding is that it generates more traffic than the other candidates, which are primarily the shortest path algorithm and the minimum spanning tree algorithm. For a network with n nodes, the minimum spanning tree algorithm requires $n-1$ transmissions of each non-directed message in order to reach all the nodes. The number of transmissions which the flooding algorithm generates not only depends upon the number of nodes but also upon the number of neighbors to which each node is connected. Suppose that each of the n nodes is, on the average, connected to m neighbors. Then with flooding, each node will send the message out of at most $m-1$ ports, except for the source node, which will sent it out of all m ports. (If a node receives the same message essentially simultaneously in two or more ports, it can send it out of fewer than $m-1$ ports.) The total number of transmissions which the flooding algorithm generates is thus at most $n(m-1) + 1$. For very large networks, the nominally hexagonal grid which is used in the radar network has an average value of m which approaches three. Therefore, for the radar network as it is currently envisioned, flooding requires at most $2n+1$ transmissions of each non-directed message, or about twice as much as with the minimum spanning tree algorithm. Some initial studies by Air Force personnel (23) indicate that the cases with simultaneous reception of messages on two or more ports occur often enough to significantly reduce this extra traffic penalty.

The other performance considerations, however, favor flooding. The minimum spanning tree and shortest path algorithm do not adapt nearly as well to network damage and require extensive knowledge of the network structure. The shortest path algorithm requires considerable processing at each node. Flooding, on the other hand, adapts instantly to network damage, requires essentially no processing, and requires absolutely no knowledge of the network structure beyond knowledge of a node's nearest neighbors. Therefore, for non-directed messages at least, the increase in traffic which flooding produces is acceptable (for this type of network) when weighted against the tremendous benefits it produces in adaptability and ease of implementation. In addition, flooding non-directed messages can provide knowledge of the network connectivity which can be used to reliably route the directed messages.

New Directed Message Routing Algorithms

A new class of directed message routing algorithms have been developed to meet the requirements of the radar network. These routing algorithms are members of the class of stochastic directory routing algorithms, similar to the backwards leaning algorithm proposed by Baran (see Chapter II).

These new algorithms route messages by storing at each node a table of the current estimate of the time required to reach each possible destination when leaving via each output port. The table is consulted for each message arriving at the node to determine which output port is the best to route the message to its destination. The message is then sent out of this port. With these new routing algorithms, the routing tables are updated with information contained in the incoming non-directed messages. Since the non-directed messages are routed using flooding, which has excellent adaptability, it is expected that this adaptability can be reflected in the routing tables, making a very reliable algorithm. This is accomplished with very little extra overhead since most nodes in the radar network flood out messages independently of the need for them to construct the routing tables. Some specialized command and control nodes may not ordinarily send out non-directed messages, and this routing algorithm would thus require them to occasionally send out short "Here I Am" (HIA) messages. These HIA messages would constitute extra overhead.

For conventional network applications, nodes would also not normally need to flood out non-directed messages. They would also need to flood out the short HIA messages. Even though many transmissions of these HIA messages are produced with flooding, they can be very short, thus keeping the overhead within reasonable limits. An analysis of the overhead requirement for conventional networks is given in Chapter VI.

Several versions of the new routing algorithms have been developed, each with its own performance characteristics. These will be presented individually in the following sections. The first, "Simple Backwards Learning" (SBL), will be evaluated to determine its steady state characteristics in an undamaged network. Then, Simple Backwards Learning will, as will all of the new routing algorithms, be evaluated to determine its transient performance characteristics while adapting (or attempting to adapt) to each of three cases of network damage.

Simple Backwards Learning Algorithm

With Simple Backwards Learning, each node stores a routing table, as shown in Figure 4.1. Each routing table entry, a_{ij}, is roughly proportional to the estimated delay when sending a directed message to node j by way of port i. Thus, each column in the routing table corresponds to one of the network nodes and each row corresponds to one of the outgoing ports. For the simulation model of the network, each node thus stores a routing table with 13 columns and two or three rows. (See Figure 3.1 for the network simulated.)

The routing table entries are updated using information contained in the flooded messages. For this purpose, each flooded message contains, in addition to the message content, the source node address, S, and an estimate, D_{MSG}, of the time expired in reaching the current node from node S.

$$\begin{array}{cc} & \text{Each column corresponds to one node.} \\ \begin{array}{l}\text{Each row} \\ \text{corresponds} \\ \text{to one of the} \\ \text{node's ports.}\end{array} & \begin{bmatrix} a_{ij} = \begin{array}{l}\text{a rough estimate of the delay required} \\ \text{to reach node } j \text{ when the message is} \\ \text{sent out of port } i.\end{array} \end{bmatrix} \end{array}$$

Figure 4. 1. SBL routing table.

Non-Directed Message Handling

The details of how the flooded messages are handled and how the information they contain is used to update the routing tables are contained in Figures 4. 2 and 4. 3. Figure 4. 2 is a conceptual block diagram which provides a general outline of how the non-directed messages are handled. Figure 4. 3 provides a very detailed explanation of how the non-directed messages are handled. This latter diagram follows very closely the programming steps which were used in the simulation program. Used together, these two block diagrams provide a clear yet complete explanation of the non-directed message handling procedure. This convention of providing both a conceptual and a detailed block diagram will be followed for all of the algorithms presented in this thesis.

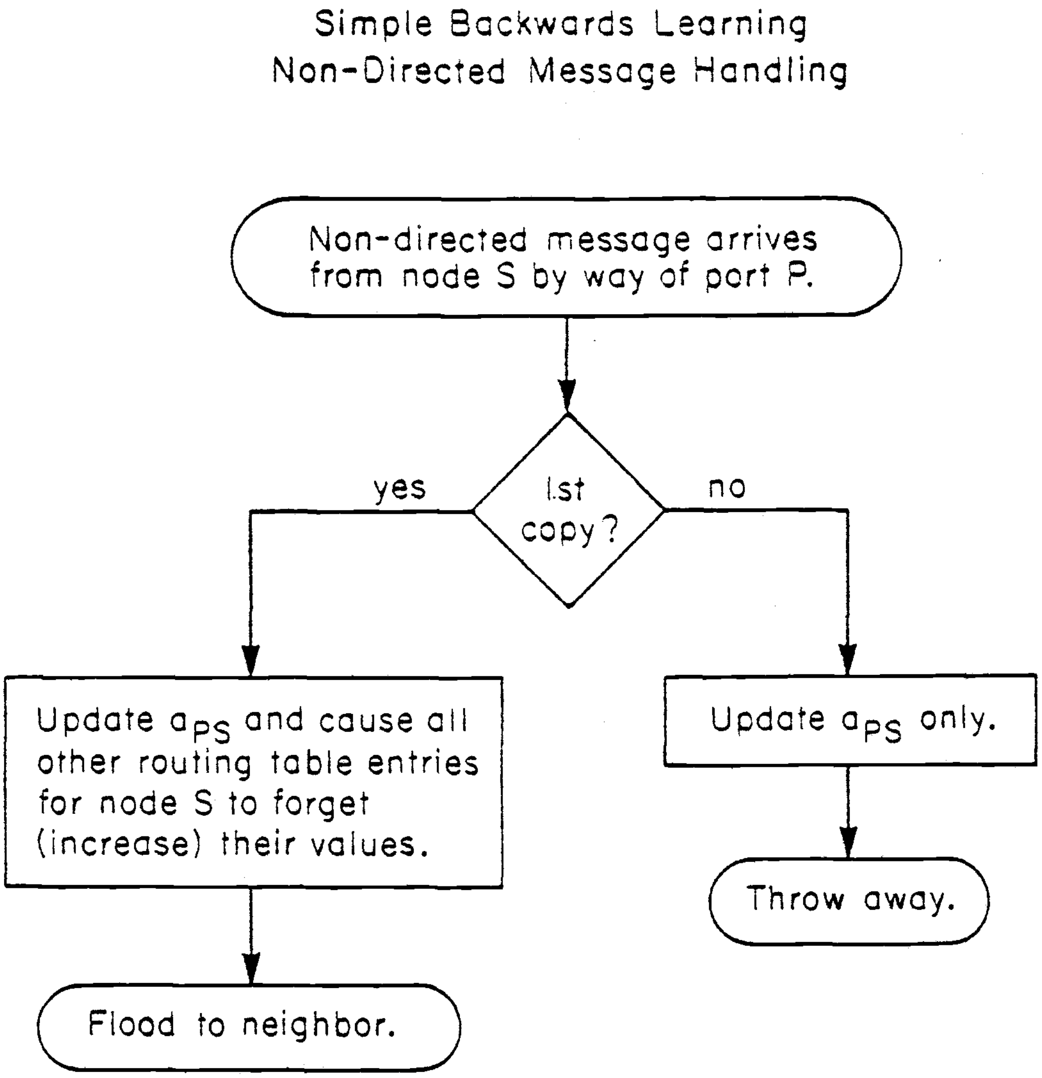
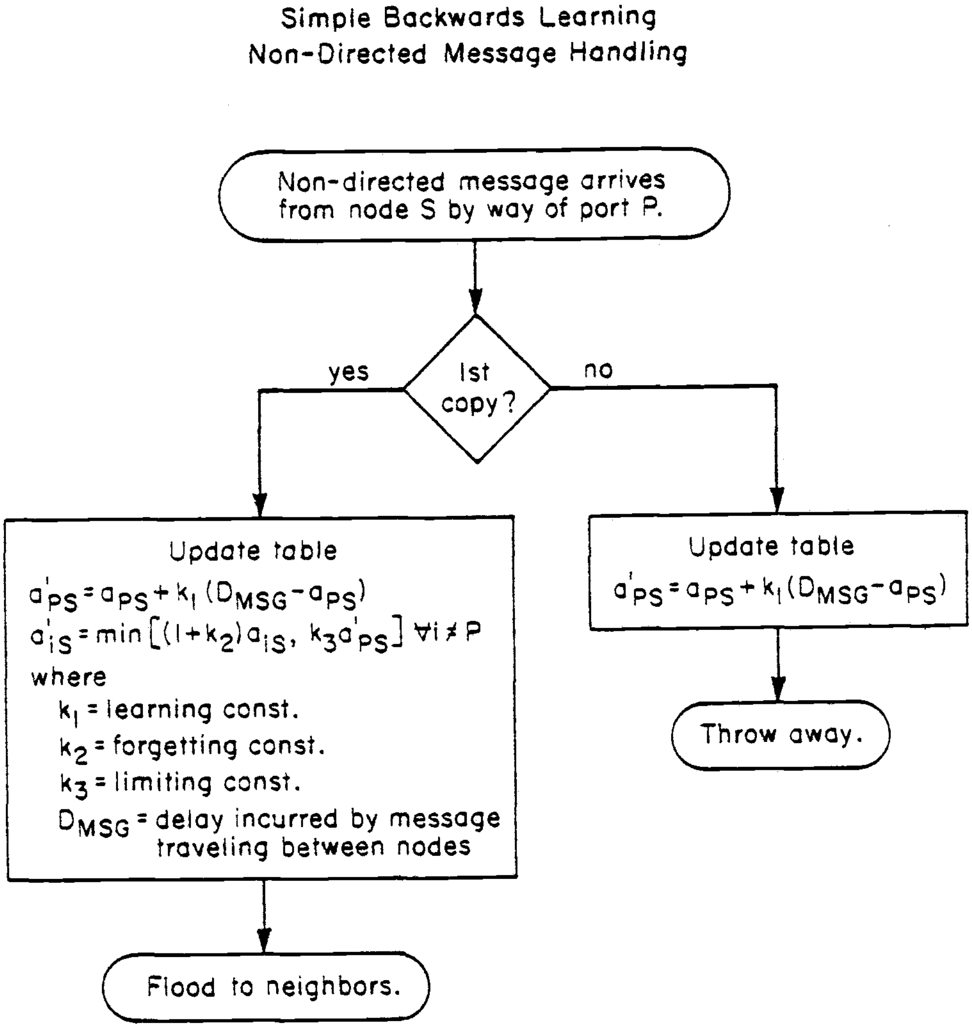

Simple Backwards Learning
Non-Directed Message Handling
Non-directed message arrives from node S by way of port P.
1st copy?
yes
no
Update a_{PS} and cause all other routing table entries for node S to forget (increase) their values.
Update a_{PS} only.
Flood to neighbor.
Throw away.

Figure 4.2. Conceptual block diagram.



Simple Backwards Learning
Non-Directed Message Handling

Non-directed message arrives from node S by way of port P.
1st copy?
yes
no
Update table
$a'_{PS} = a_{PS} + k_1 (D_{MSG} - a_{PS})$
$a'_{iS} = \min[(1+k_2)a_{iS},\ k_3 a'_{PS}]\ \forall i \neq P$
where
k_1 = learning const.
k_2 = forgetting const.
k_3 = limiting const.
D_{MSG} = delay incurred by message traveling between nodes
Flood to neighbors.
Update table
$a'_{PS} = a_{PS} + k_1 (D_{MSG} - a_{PS})$
Throw away.

Figure 4. 3. Detailed block diagram.

Suppose that a flooded message with source node S arrives at node X by way of port P. Node X first checks to see if the received copy is the first copy of the message. If it is, the node updates its routing table entry for port P as follows:

$$a'_{PS} = a_{PS} + k_1(D_{MSG} - a_{PS}) \tag{4.1}$$

where

a'_{PS} = the new estimated delay to node S by way of port P.

a_{PS} = the old estimated delay to node S by way of port P.

k_1 = the learning constant ($0 < k_1 \leq 1$).

D_{MSG} = the delay experienced by the message in travelling from node S to node X.

Thus, the new delay estimate, a'_{PS}, is a fraction k_1 of the way between the old estimate, a_{PS}, and the current estimate, D_{MSG}. Immediately after updating the entry for port P, the entries for the other ports are increased as follows:

$$a'_{iS} = \min(1+k_2)a_{iS}, k_3 a'_{PS}) \quad \forall \quad i \neq P \tag{4.2}$$

where

a'_{iS} = the new estimated delay to node S by way of port i.

a_{iS} = the old estimated delay to node S by way of port i.

k_2 = the forgetting constant ($0 < k_2$).

k_3 = the limiting constant $(1 < k_3)$.

a'_{PS} = the just computed new estimated delay to node S by way of port P.

Equation (4. 2) represents a forgetting process. This is an essential step to ensure that old routing table entries which are no longer valid because of changes in network structure or conditions are forgotten by causing the entries to increase. This forgetting process is caused by the first copy of the non-directed message, which may be the only copy received. The routing table entries are allowed to increase only up to a limit which is determined by k_3. This prevents the routing table entries from becoming excessively large, thus allowing them to be decreased through the learning process in a reasonable amount of time if a change in network structure necessitates it.

After the node has updated its routing tables, it sends this first copy of the non-directed message on to each of its neighbors, except the one from which it was received, completing the flooding process for that message at node X.

If a second copy of the non-directed message arrives, the node updates the entry for the port the copy arrived in, using Equation (4. 1). This second copy does not, however, cause the other routing table entries to forget their values since the forgetting process was already accomplished by the first copy of the message. After updating the routing table, the message is then discarded, since the neighbors have already been sent a copy of it.

Directed Message Handling

The details of how directed messages are handled and routed are contained in the conceptual block diagram of Figure 4. 4 and the detailed block diagram of Figure 4. 5.

Suppose that a directed message arrives at node X, and is directed for node D. Node X first checks to see if a copy of the directed message has ever visited the node previously. If it has, the message is discarded and a global counter of looping messages is incremented. If it is the first copy of the message, the routing tables are consulted to determine the best port(s) to send the message out of to reach node D. A message is sent out of port M if

$$\begin{aligned}&(((a_{MD} \leq a_{iD}) \text{ or (link out of port i is down)) } \forall \text{ i)} \\ &\qquad\qquad \text{and (link out of port M is up)} \qquad\qquad (4.3)\end{aligned}$$

Steady State Tests and Performance Model

The Simple Backwards Learning algorithm was first tested to determine its steady state characteristics in the undamaged 13 node network in Figure 3. 1. This network has 13 nodes and 30 communication channels.

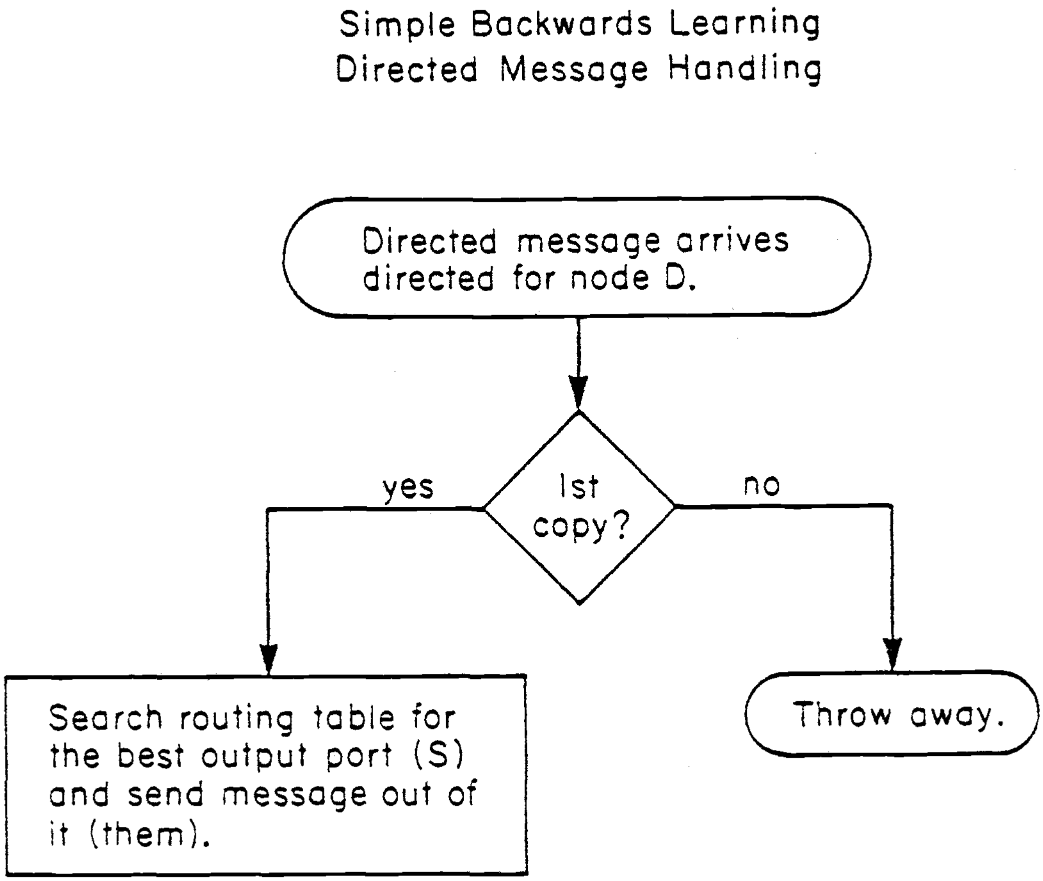

Simple Backwards Learning
Directed Message Handling
Directed message arrives directed for node D.
Ist copy?
yes
no
Search routing table for the best output port (S) and send message out of it (them).
Throw away.

Figure 4. 4. Conceptual block diagram.

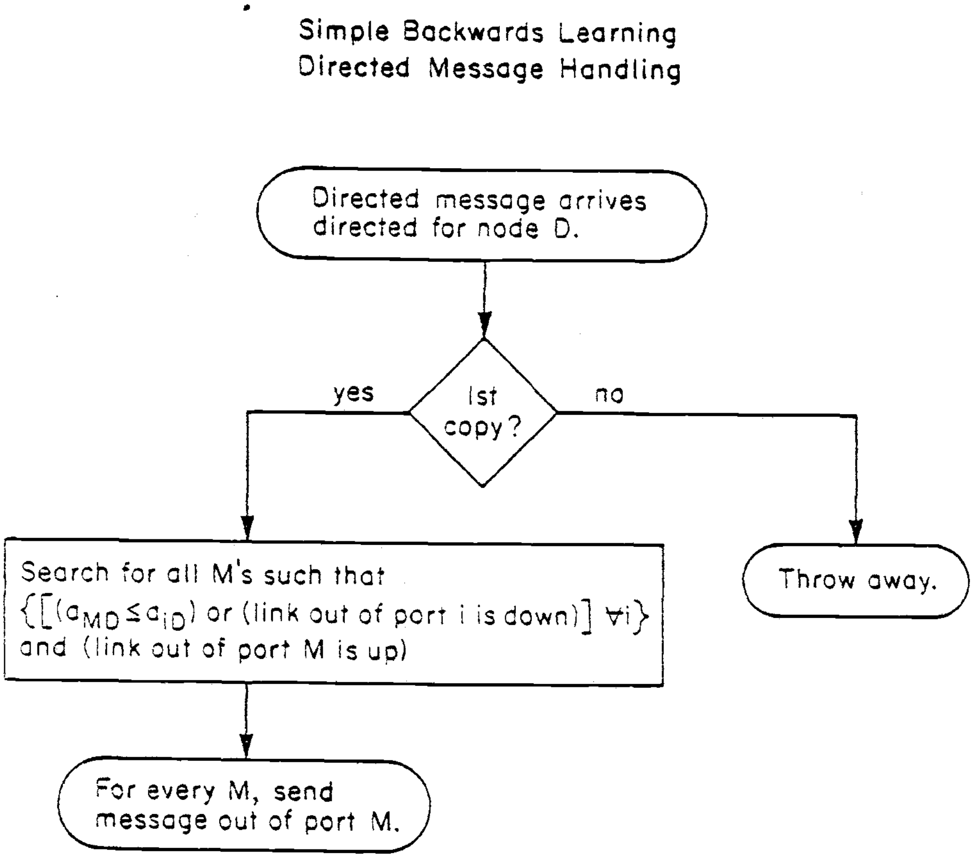

Simple Backwards Learning
Directed Message Handling
Directed message arrives directed for node D.
1st copy?
yes
no
Search for all M's such that $\{[(a_{MD} \leq a_{iD})$ or (link out of port i is down)] $\forall i\}$ and (link out of port M is up)
Throw away.
For every M, send message out of port M.

Figure 4.5. Detailed block diagram.

The values of the learning, forgetting, and limiting constants which were used in the steady state tests, as well as the value of ℓ_M/C, the time required to send a single message over a communication link, are given in Table 4.1. These parameters will remain unchanged for all the steady state tests.

Table 4.1. Values of parameters used in the steady state tests of the Simple Backwards Learning algorithm.

Parameter	Value
k_1	0.25
k_2	0.25
k_3	10.0
ℓ_M/C	10.0

The steady state tests were conducted to determine the directed and non-directed message delays, communication channel utilization, and queue sizes, for various values of $\bar{t}_D$ and $\bar{t}_{ND}$, the per node values of the mean inter-generation times. Each network node is assumed to generate equal amounts of directed and non-directed traffic. Thus, the per node offered traffic rate for directed messages (the rate at which directed messages enter the network at each node) is equal to the per node offered traffic rate for non-directed messages (the rate at which non-directed messages enter the network at each node). Thus, γ, the per node offered traffic rate for directed messages and non-directed messages, is equal to both $1/\bar{t}_D$ and $1/\bar{t}_{ND}$.

The results of simulation tests relating the steady state directed message delay, D_{DM}, and the steady state non-directed message delay, D_{NDM}, to variations in γ are given in Figure 4.6. (The plots and tables in this section will use the per node values of γ described above. The total directed and non-directed traffic entering the 13 node GPSS model of the radar network is 26γ.)

For the purpose of creating a performance model, the following statistics were gathered: the average path length, $\bar{p}$, experienced by all directed messages in travelling from their source nodes to their destination nodes, the average waiting time for messages in each queue, W_i, and the utilizations for each communication channel, ρ_i. It is the object of the performance model to use these statistics to determine a sound theoretical relationship between D_{DM} and γ.

The delay which a directed message experiences in travelling from its source node S to its destination node D is

$$d_{SD} = \frac{\ell_M p_{SD}}{C} + p_{SD}\bar{W}_{SD} \qquad (4.4)$$

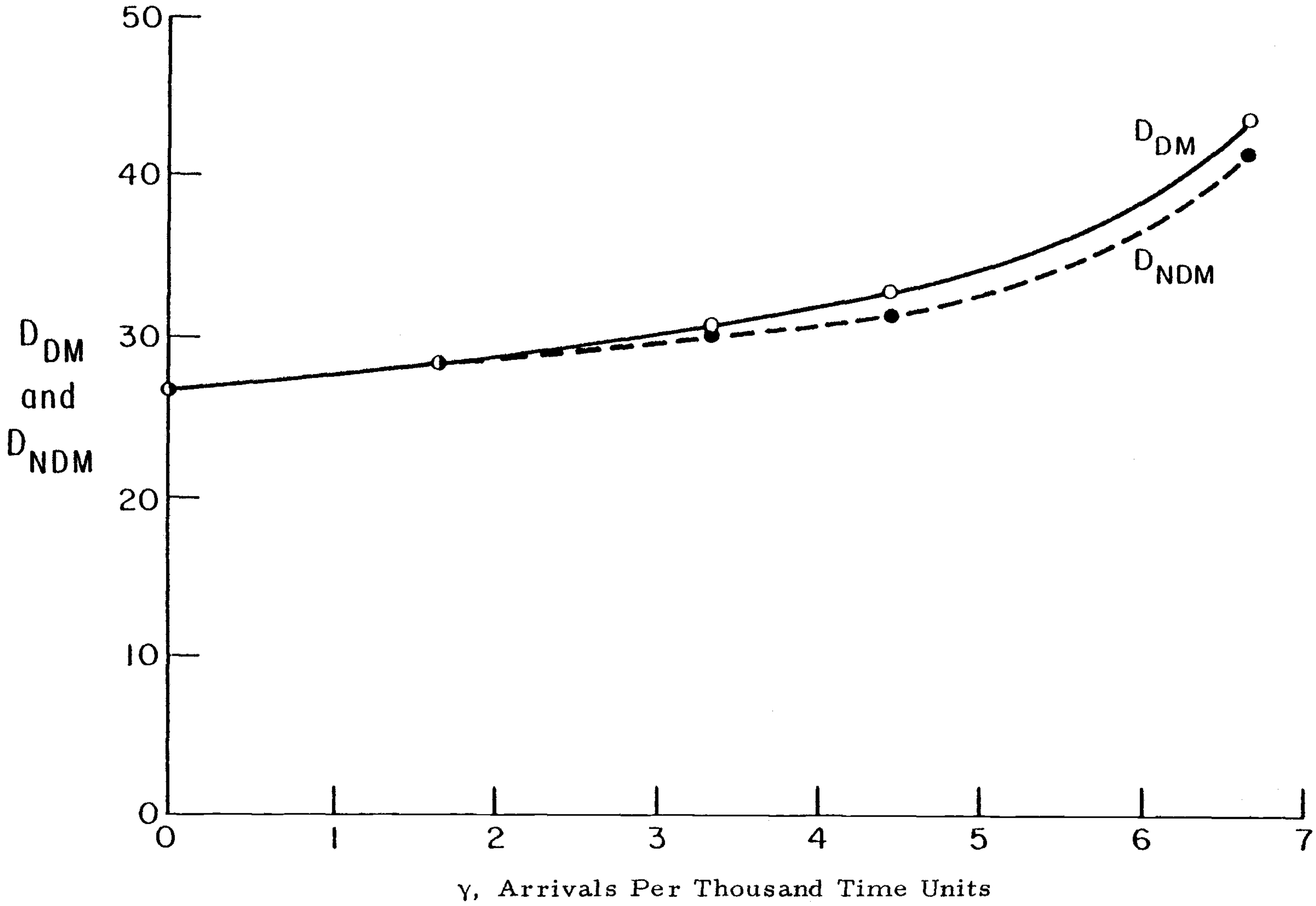


Figure 4.6. Steady state delays.

where

p_{SD} = the path length from node S to node D.

$\overline{W}_{SD}$ = the average queue waiting time in the path from node S to node D.

Equation (4. 4) suggests that a reasonable equation for D_{DM} is

$$D_{DM} = \frac{\ell_M \overline{p}}{C} + \overline{p}\ \overline{W} \tag{4. 5}$$

where

$$\overline{W} = \sum_{i=1}^{m} \frac{\rho_i}{\sum_{j=1}^{m} \rho_j} W_i \tag{4. 6}$$

and

m = the total number of communication links in the network.

These equations relate the directed message delay to the sum of two quantities, the path delay and the queueing delay. The path delay is the average path length travelled by each directed message multiplied by the time it takes one message to traverse a single communication link. The queueing delay is the path length (i. e., the average number of queues each message travels through) multiplied by the average of the queue waiting times, with each queue waiting time in the average weighted by the fraction of the total traffic using that queue.

The values of average directed message delay resulting from Equations (4.5) and (4.6) are compared with the actual values of D_{DM} in Table 4.2.

Table 4.2. Comparison of the actual values of steady state directed message delay and the values computed from Equations (4.5) and (4.6).

	D_{DM}	
γ	From Equations (4.5) and (4.6)	Actual Value
1.67×10^{-3}	28.0	28.4
3.33×10^{-3}	30.5	30.7
4.44×10^{-3}	32.1	32.9
6.67×10^{-3}	44.8	43.5

The next step in the performance model is to determine a relationship between the channel utilization, ρ_i, and the queue waiting time, W_i.

However, there is a problem with arriving at a relationship between ρ_i and W_i. The problem arises as a result of the existence of a dependence among the interarrival and service times for messages arriving at a queue imbedded within a communication network. The interarrival time between two successive messages on a communication channel in a network can be no less than the service time for the first of these two messages. Since the service time for a message on its next channel is directly related to its service time on the preceding channel (they are equal in our model of the radar network since all messages are assumed to fit in the same size packet and all the communication channels are assumed to have the same capacity), the arrival process of messages to a node due to the internal traffic in the network (i.e., messages arriving from neighboring nodes) is not independent of the service time these messages received at that node. This dependence between interarrival and service times is especially important when messages arriving on a given channel all depart over the same channel. In this instance, the messages will all arrive spaced far enough apart that they will never have to wait before being served by the outgoing channel, regardless of its utilization. The dependence between interarrival and service time is not as important, however, if messages arriving over the same channel all depart over different channels, or if messages departing on a given channel all arrived at the node over different channels. This is called mixing. If there is sufficient mixing at a node, then Kleinrock has shown (3),

$$W_i = \frac{\ell_M \rho_i}{C(1-\rho_i)} \tag{4.7}$$

Equation (4.7) is only an approximation. It has been shown to be reasonably accurate when each communication channel at a node receives messages from as few as two or three incoming channels (21).

To test the validity of Equation (4.7) for the radar network, Table 4.3 compares the following two quantities

$$\bar{W} = \sum_{i=1}^{m} \frac{\rho_i}{\sum_{j=1}^{m} \rho_j} W_i \tag{4.6}$$

and

$$\bar{W}_K = \sum_{j=1}^{m} \frac{\rho_i}{\sum_{j=1}^{m} \rho_j} \left(\frac{\ell_M \rho_i}{C(1-\rho_i)}\right) \tag{4.8}$$

Clearly, the comparison between $\bar{W}_K$ and $\bar{W}$ is not good. There are several possible reasons for this. First, Kleinrock's equation assumes that the message lengths are exponentially distributed, whereas the simulation results for the radar network assume fixed message lengths. This means that Kleinrock assumes that the time required to transmit a message across a communication channel is exponentially distributed, whereas in our model of the radar network this time is fixed. The Pollaczek-Khinchin mean value formula (3) predicts that this difference in transmission time distributions will cause a reduction of one-half in the waiting times for our simulation results versus the waiting times given by Equation (4. 8). (Since the effect of greater regularity in interarrival times is similar to that of greater regularity in service times, more regular than Poisson arrivals at a node could also reduce delays. This factor could not be studied from the simulation data, however, since only mean interarrival times were available.) Second, as previously stated, Kleinrock's equation has been shown to be reasonably accurate when each communication channel at a node received messages from as few as two or three incoming channels. This is true for the nodes in the interior of our GPSS model of the radar network (see Figure 3. 1), however, the nodes at the exterior of the network contain channels which receive messages from only one incoming channel (in addition to the directed and non-directed messages which the node generates). Third, Kleinrock's equation has been verified primarily by simulations of networks which contain predominantly directed messages. It is possible that the large amount of flooded non-directed messages used in simulations on the radar network could also degrade the accuracy of Equation (4. 8) by contributing to a regularity in the arrivals of messages at each outgoing channel, thus decreasing the queue lengths.

Table 4.3. Comparison of $\overline{W}_K$ and $\overline{W}$.

γ	$\overline{W}_K$	$\overline{W}$
1.67×10^{-3}	1.95	0.463
3.33×10^{-3}	5.03	1.37
4.44×10^{-3}	7.95	1.94
6.67×10^{-3}	23.0	6.65

To remedy the discrepancy between $\overline{W}_K$ and $\overline{W}$, the relationship

$$\overline{W} = (1/R)\overline{W}_K \tag{4.9}$$

was tested. Minimum square error analysis indicated that a suitable value of R is 3.52. The relationship between $\overline{W}$ and $(1/3.52)\overline{W}_K$ is shown in Table 4.4. The relationship appears reasonably good.

Table 4.4. Comparison of $\overline{W}$ and $(1/3.52)\overline{W}_K$.

γ	$(1/3.52)\overline{W}_K$	$\overline{W}$
1.67×10^{-3}	0.554	0.463
3.33×10^{-3}	1.43	1.37
4.44×10^{-3}	2.25	1.94
6.67×10^{-3}	6.53	6.65

The above results indicate that

$$\overline{W} = \frac{1}{3.52} \sum_{i=1}^{m} \frac{\rho_i}{\sum_{j=1}^{m} \rho_j} \left(\frac{\ell_M \rho_i}{C(1-\rho_i)}\right) \tag{4.10}$$

Substituting Equation (4.10) into Equation (4.5) yields

$$D_{DM} = \frac{\ell_M \overline{p}}{C} + \frac{\overline{p}}{3.52} \sum_{i=1}^{m} \frac{\rho_i}{\sum_{j=1}^{m} \rho_j} \left(\frac{\ell_M \rho_i}{C(1-\rho_i)}\right) \tag{4.11}$$

Equation (4.11) is compared to the actual directed message delay in Table 4.5. The results are very good.

Table 4.5. Comparison of the actual values of steady state directed message delay and the values computed from Equation (4.11).

	D_{DM}	
γ	From Equation (4.11)	Actual Value
1.67×10^{-3}	28.3	28.4
3.33×10^{-3}	30.6	30.7
4.44×10^{-3}	33.0	32.9
6.67×10^{-3}	44.6	43.5

The final step in relating D_{DM} to γ is to show a relationship between γ and the ρ_i's. Figure 4.7 shows a plot of three representative ρ_i's (see Figure 3.1 for the location of the channels). It appears reasonable to fit the curves in Figure 4.7 using the relationship

$$\rho_i = a_i \gamma^2 + b_i \gamma \tag{4.12}$$

The values of a_i and b_i for the curves in Figure 4.7 are given in Table 4.6.

Table 4.6. Values of the parameters used to fit the curves in Figure 4.7.

ρ_i	a_i	b_i
ρ_1	7.25×10^3	69.1
ρ_{10}	4.41×10^3	60.0
ρ_{30}	5.86×10^3	33.2

Communication channels 1, 10, and 30 span nearly the entire range of communication channel utilizations. Communication channel 1 is representative of the channels with high utilization $(\rho_1 \approx \rho_2 \approx \rho_3)$, communication channel 30 is representative of the channels with the lowest utilizations $(\rho_{30} \approx \rho_{28} \approx \rho_{21} \approx \rho_{19} \approx \rho_{18} \approx \rho_{16})$, and communication channel 10 is representative of the channels with medium

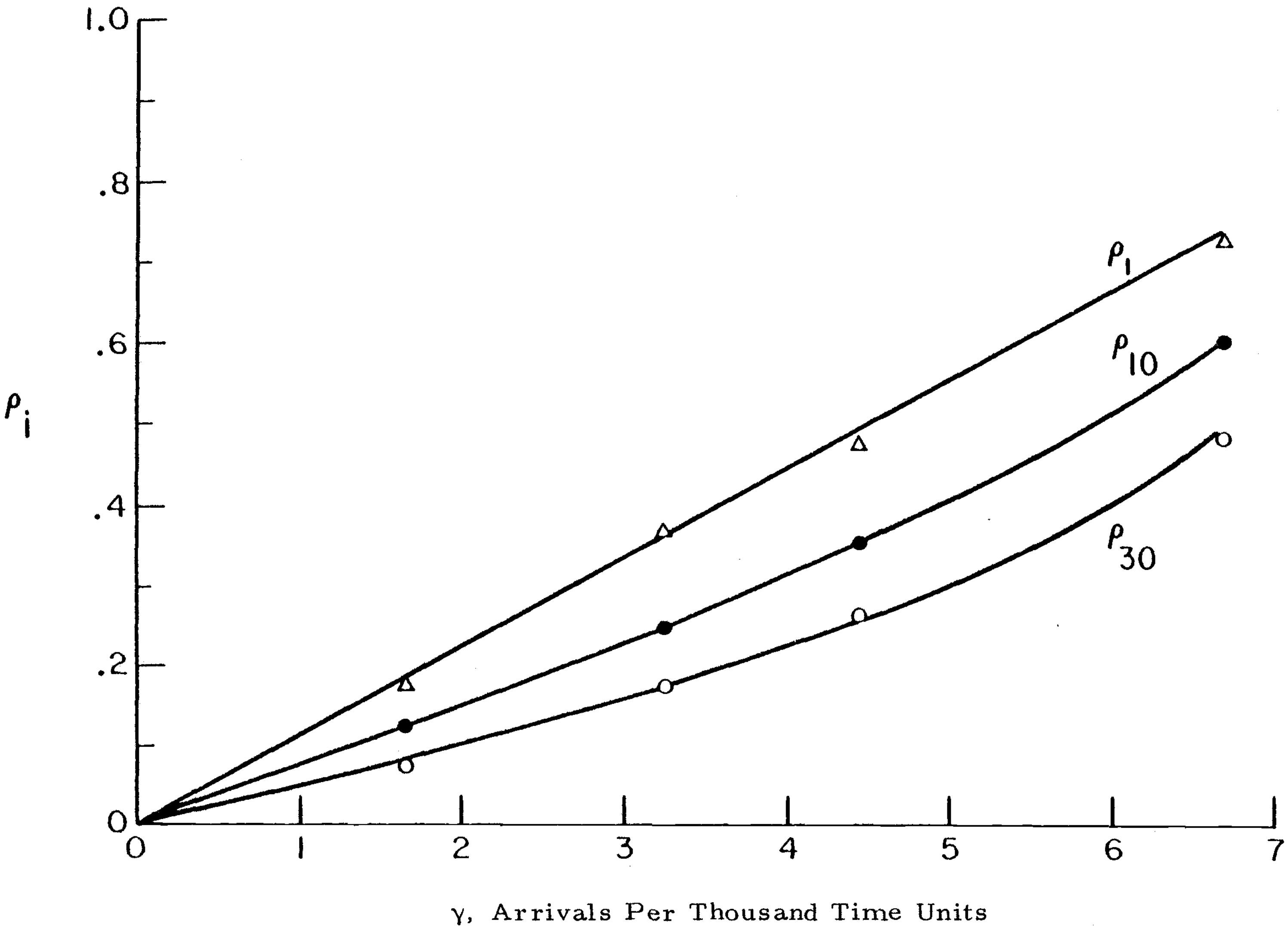


Figure 4. 7. Steady state channel utilizations.

utilization $(\rho_{10} \approx \rho_5 \approx \rho_8)$. The values of a_i and b_i for channels 1, 10, and 30 are reasonable for many other channels with high, medium, and low utilizations, respectively.

Equations (4. 11) and (4. 12) together provide a way of computing D_{DM} directly from γ, thus completing the performance model.

Adaptability Tests

For the transient simulation tests of the routing algorithm's adaptability, three cases of network damage are considered. The first case has three of the communication links destroyed so that transmission is inhibited in both directions over them. This is a form of reciprocal damage, since the damage is the same in both directions over the links. This case is shown in Figure 4.8. The destroyed links are omitted from the diagram. Figure 4. 9 shows the second case of network damage. Here the same three links are jammed instead of destroyed. Jamming is a form of non-reciprocal damage whereby transmission is permitted in one direction over a communication link but not the other. The direction in which transmission is permitted corresponds to the direction of the arrowheads in the diagram. The final case, six links jammed, is shown in Figure 4. 10. This is a very severe case of jamming, and many of the node pairs have no reciprocal paths between them.

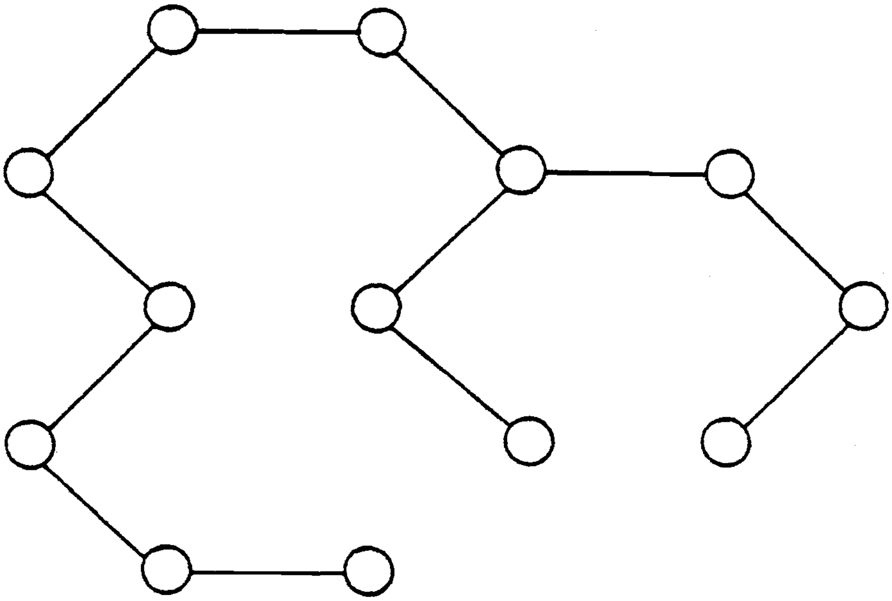


Figure 4.8. Three links destroyed.

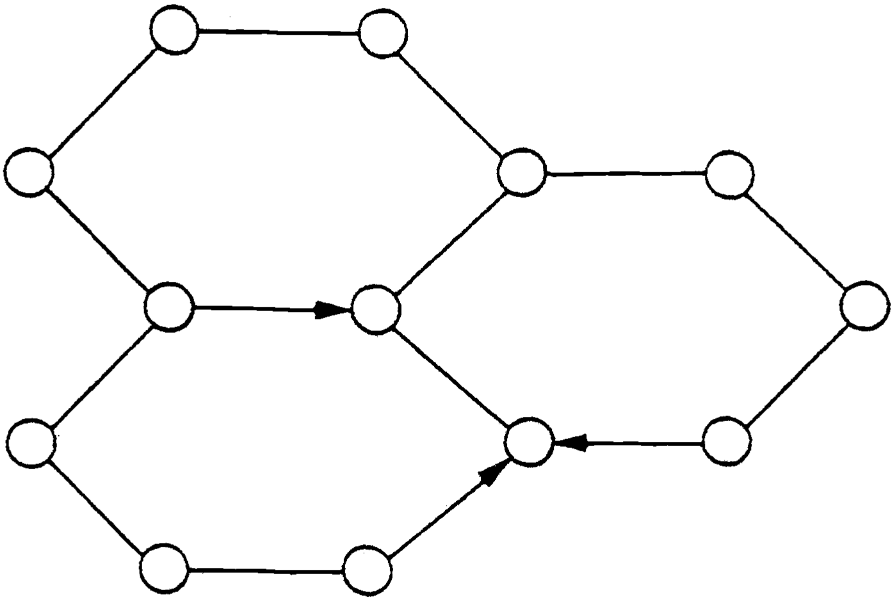


Figure 4.9. Three links jammed.

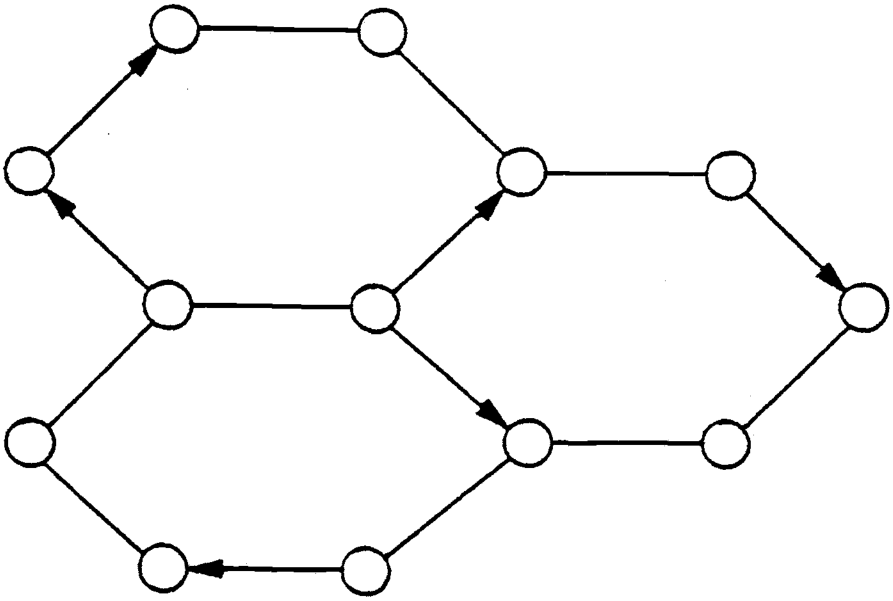


Figure 4. 10. Six links jammed.

In order to test the routing protocol's adaptability to network damage, the simulator brings the undamaged network up to steady state, disables the appropriate communication links, and compiles the transient statistics thereafter. The statistics used here are:

1. The number of directed messages delivered and lost per directed message sent.
2. The number of looping messages per directed message sent.
3. The adjusted directed and non-directed message delays.

These statistics are averaged over a "window" (interval of time) of width $4\bar{t}_{ND}$ for statistics 1 and 2 above, and of width $8\bar{t}_{ND}$ for 3. Thus, any point on the curves of these statistics provides an indication of the near term performance of the routing protocols.

The values of the various simulation and routing algorithm parameters which were used in the transient simulation tests are given in Table 4.7. These parameter values will remain fixed, unless otherwise specifically stated, throughout the transient simulation tests of all the routing algorithms.

Table 4.7. Values of parameters used in the transient simulation tests of the new routing protocols.

Parameter	Value
k_1	0.25
k_2	0.25
k_3	10.0
$\bar{t}_D$	300.0
$\bar{t}_{ND}$	300.0
ℓ_M/C	10.0

Three Links Destroyed

Figure 4.11 is a plot of the directed messages delivered and lost per directed message sent, for the Simple Backwards Learning algorithm. This plot shows that immediately following the instant of network damage, a temporary condition exists when messages are lost. This condition lasts for a period of approximately $8\bar{t}_{ND}$. During this period, the routing tables have not yet learned the new

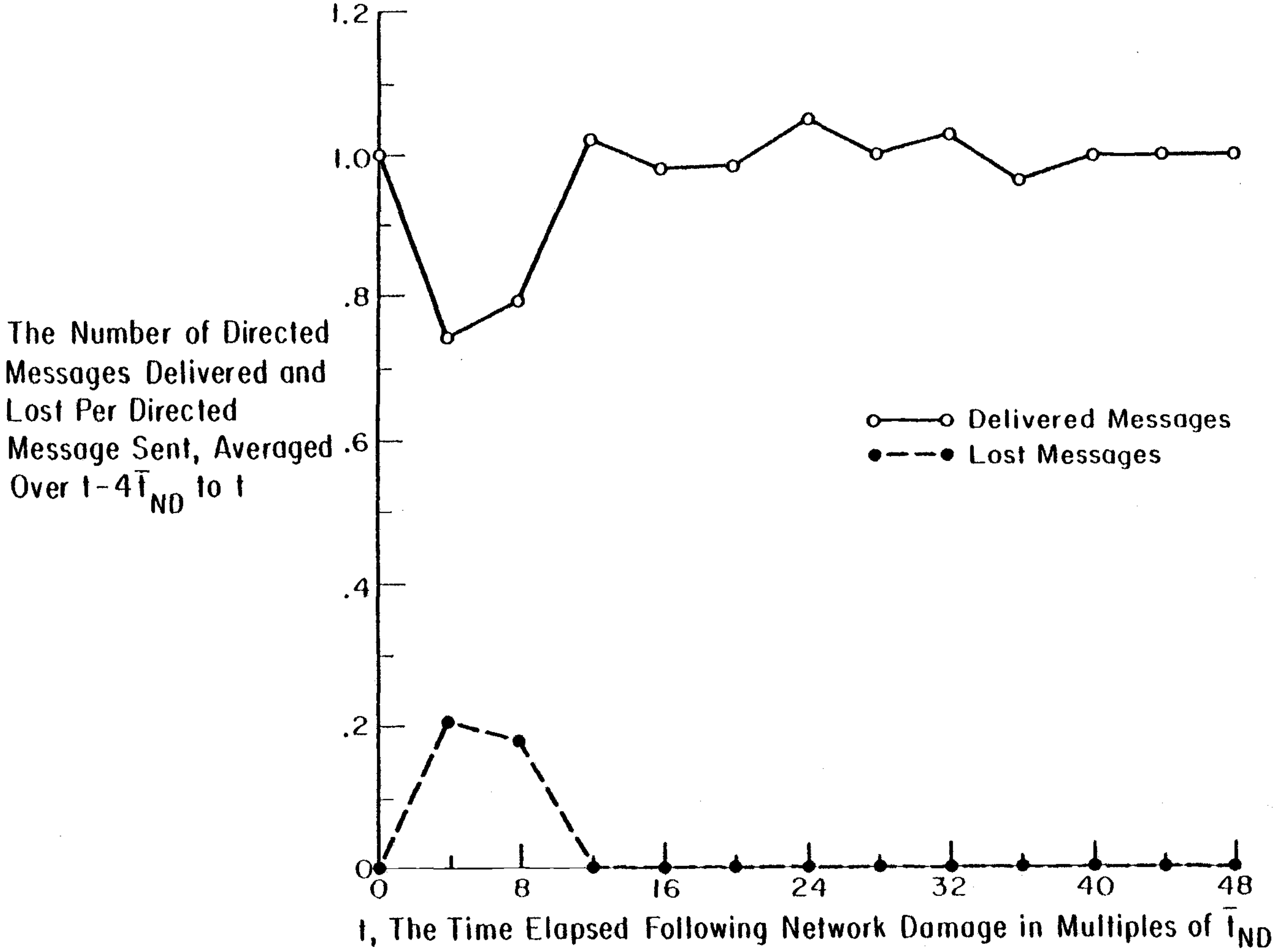


Figure 4.11. SBL algorithm, three linkds destroyed.

routes by which the flooded messages are now arriving at the nodes. How fast these new routes are completely learned is determined by the learning constant, k_1, the forgetting constant, k_2, and the limiting constant, k_3. Following the learning period, the simulation results show that the algorithm has completely adapted to the new network configuration, and no more messages are lost.

Figure 4. 12 is a plot of looping messages following the instant of network damage. A message is considered to have looped if it or a copy of it ever returns to a node. Looping messages are another indicator of bad routing table entries. As can be seen, the looping is temporary, lasting only as long as messages are being lost.

It should be explained why, after no more messages are being lost, the delivered message curve fluctuates about the 1.0 mark. This is caused by fluctuations in the number of messages in progress to their destinations. During each averaging period of $4\bar{t}_{ND}$, messages may be delivered which were sent during a previous averaging period, and messages sent during the present averaging period may remain in progress to their destinations. If the number of messages sent during the previous period which arrive during the present averaging period exceeds the number of messages sent during the present period which remain in progress, then the delivered message curve rises above 1.0. The opposite situation causes the curve to dip below 1.0.

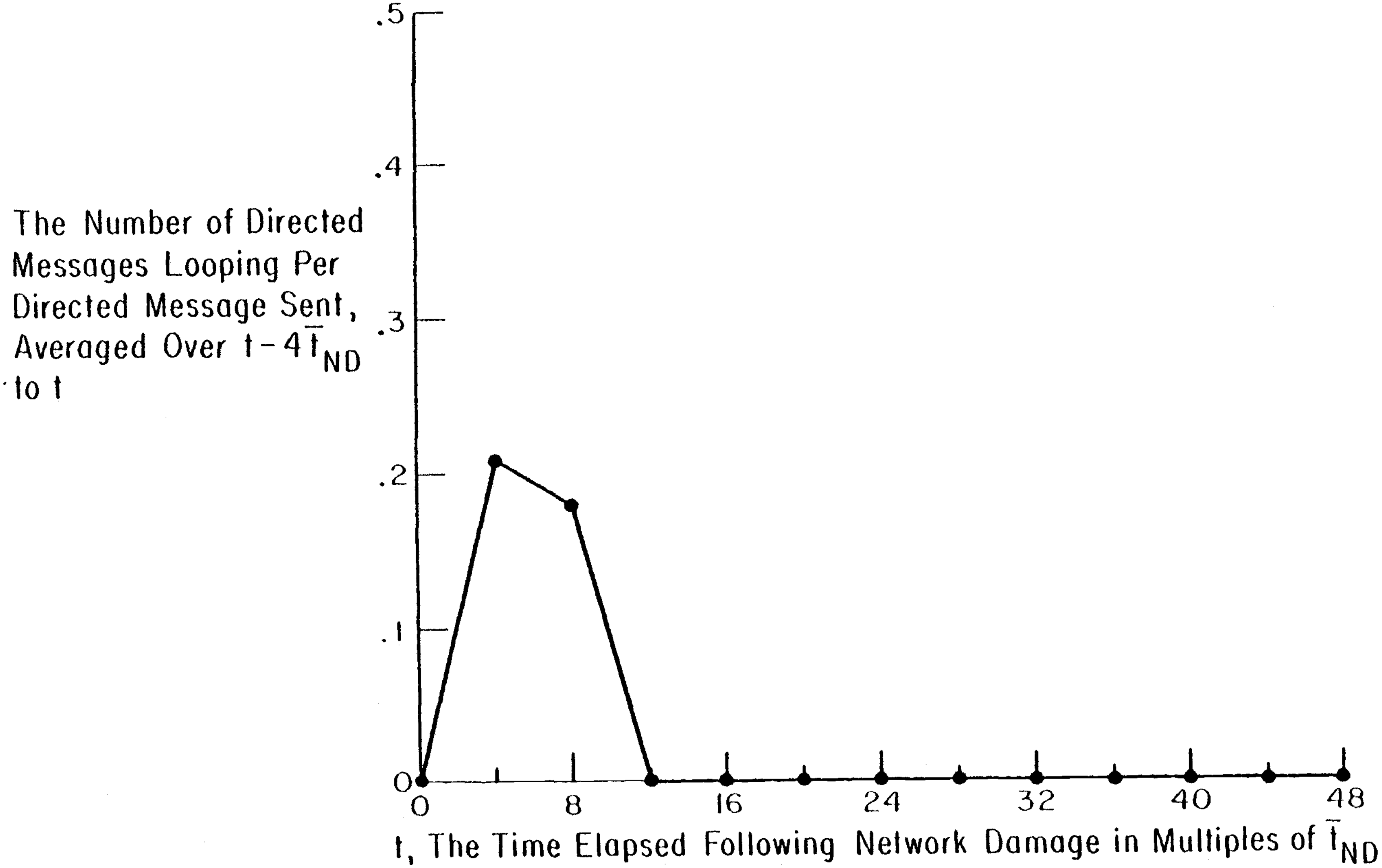


Figure 4. 12. SBL algorithm, three links destroyed.

Figure 4. 13 is a plot of the adjusted directed message delay and non-directed message delay, following the instant of network damage. Also included is a line showing the optimum delay. The optimum delay is computed by determining the average minimum path length between all node pairs, and multiplying this average by the time required for a message to traverse a single link. The result is a steady state value of the optimum delay. A consistent increase in the directed and non-directed message delays above this optimum delay is due to network congestion and to the inability of the routing algorithms to route the messages by the shortest paths.

The non-directed messages always take the shortest path between nodes. Any difference between the non-directed message delay and the optimum delay is due to network congestion causing queueing delays. Initially, the non-directed message delay takes a jump from its pre-damage value and then settles out. The non-directed message delay remains larger than the optimum delay, indicating the existence of a certain amount of network congestion.

Any significant difference between the adjusted directed message delay and the non-directed message delay is due to either the penalty of $24(\ell_M/C)$ (which is 240 for the values given in Table 4. 6) imposed on the adjusted directed message delay for each lost message, or due to messages reaching their destinations by longer than necessary routes. As can be seen, the adjusted directed message delay initially

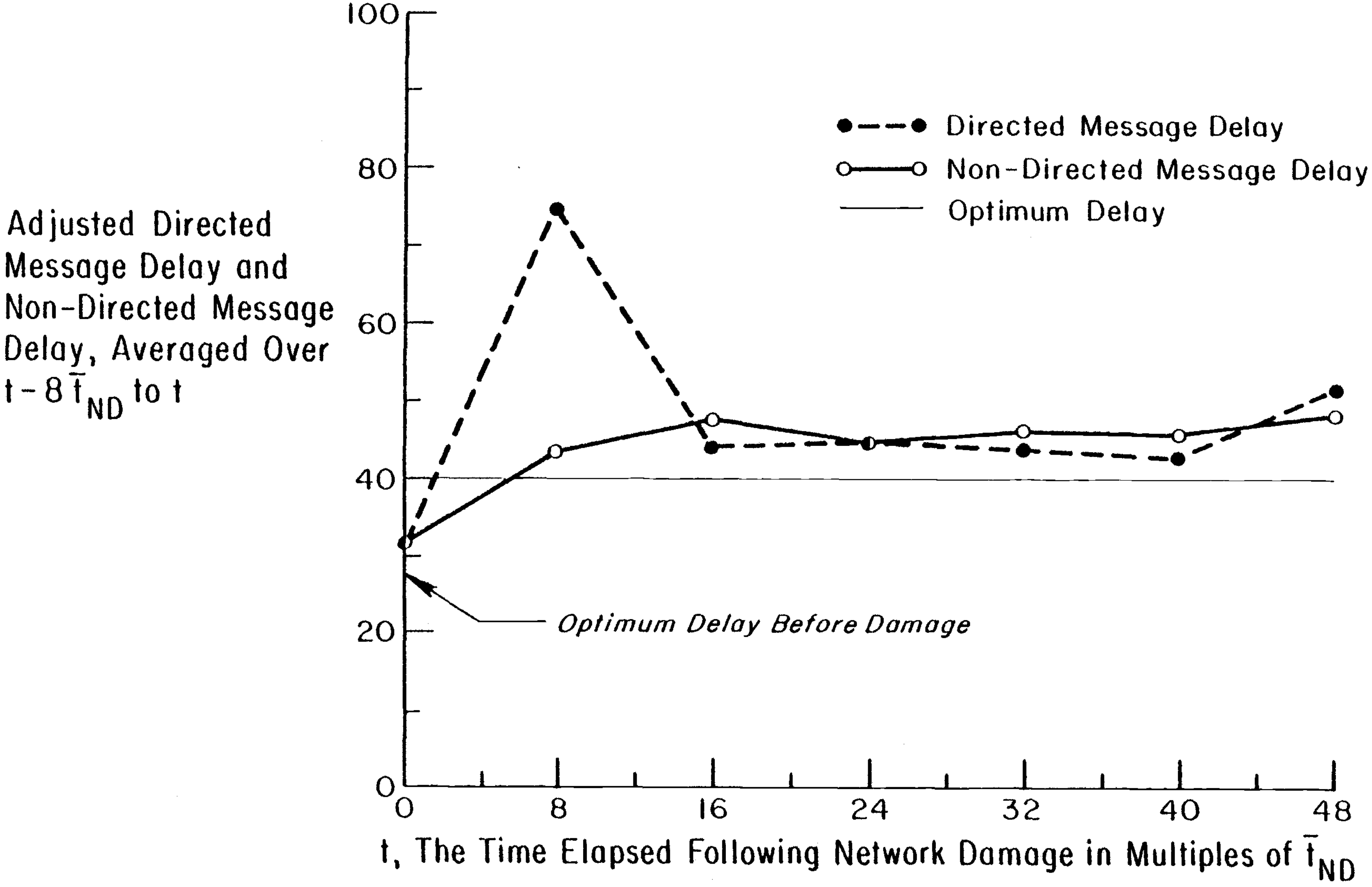


Figure 4. 13. SBL algorithm, three links destroyed.

is much larger than the non-directed message delay. This, of course, is due primarily to the lost messages. However, once the routing algorithm has completely adapted and messages are no longer being lost, the directed message delay and non-directed message delay appear to be essentially equal, indicating that the routing tables are routing the directed messages by shortest paths.

The simulation results have shown that for the case of three links destroyed, the Simple Backwards Learning routing algorithm adapts very well. This is primarily due to the excellent adaptability of the flooding algorithm being reflected in the routing tables as the flooded messages bring in information on new paths.

Three Links Jammed

Figures 4.14 and 4.15 show that for the three links jammed case, messages are both lost and looping continuously following the network damage. This is a direct result of the inherent assumption by the Simple Backwards Learning algorithm of reciprocity of the communication links. That is, the algorithm assumes that if transmission is possible in one direction over a link, that it is possible in the opposite direction also. Therefore, whenever a shortest path between two nodes contains a jammed link, messages will be directed toward that link, resulting in ping-ponging (a loop of length two), and lost messages. The Simple Backwards Learning algorithm does not

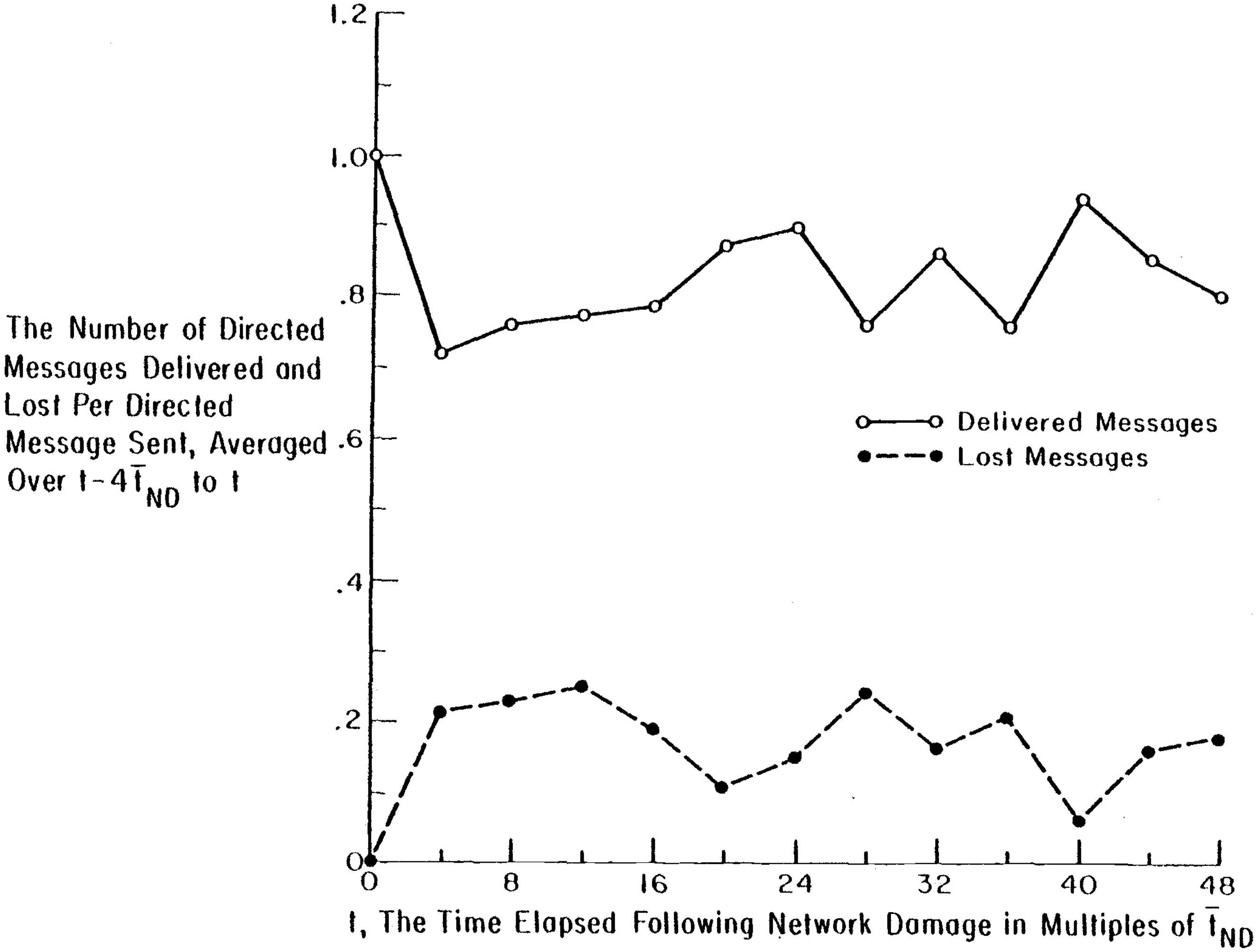


Figure 4.14. SBL algorithm, three links jammed.

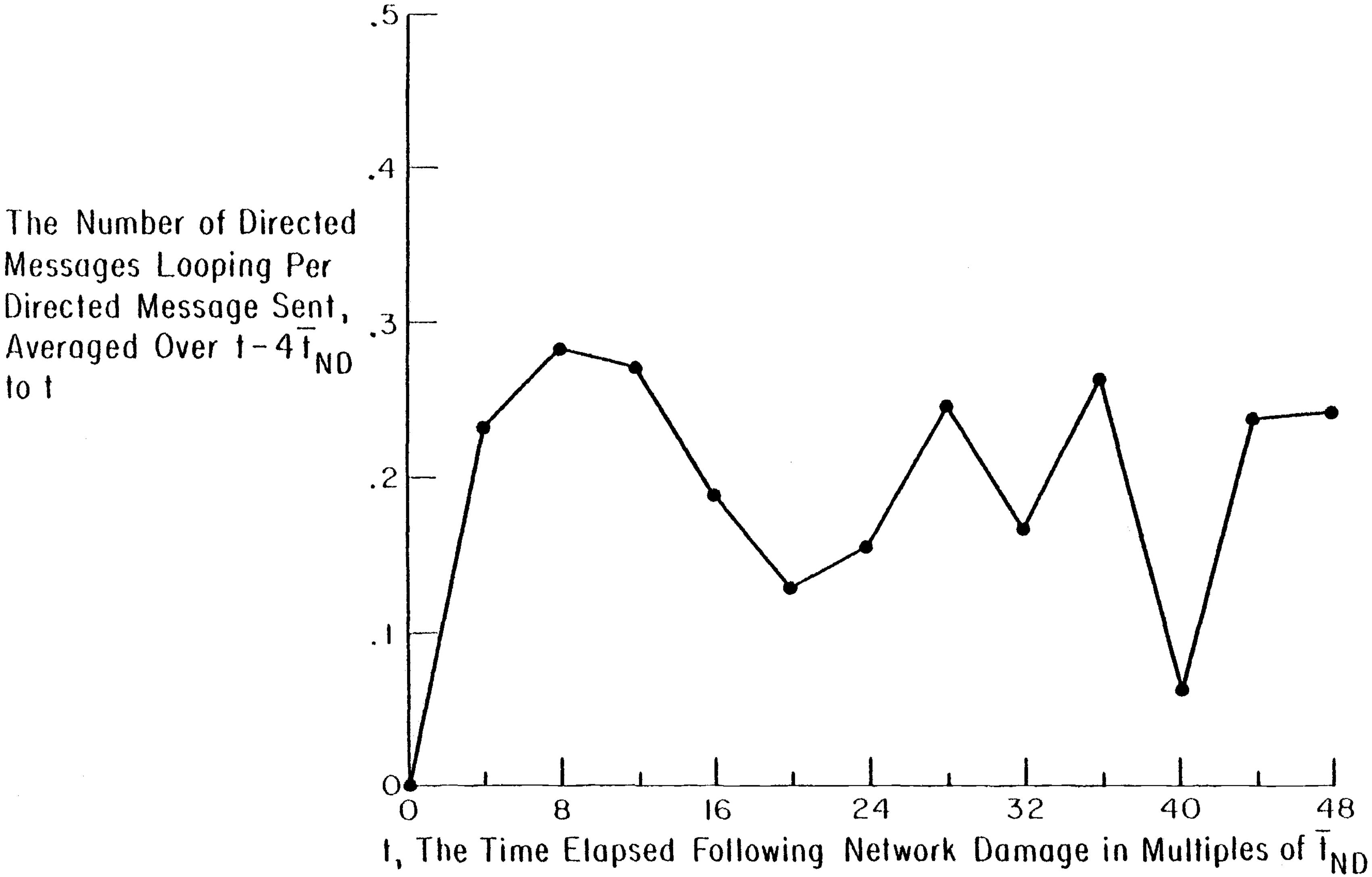


Figure 4.15. SBL algorithm, three links jammed.

have the ability to adapt to non-reciprocal forms of damage, which is a serious drawback.

Figure 4.16 shows the adjusted directed message delay and non-directed message delay. As can be seen, the adjusted directed message delay is very large, which is a result of the large number of messages being lost.

Six Links Jammed

Figures 4.17, 4.18, and 4.19 show the simulation results for the case of six links jammed. Again, large numbers of messages are lost and looping as a result of the non-reciprocal nature of the damage. Many more messages are lost and looping than with the three links jammed case, which is expected since the damage to the network is much more severe. The directed message delay remains very large due to the large numbers of messages which are being lost throughout the test.

Reciprocal Path Search Algorithm

The results for the Simple Backwards Learning algorithm show a need to create an algorithm which adapts to non-reciprocal forms of network damage, such as link jamming, in addition to adapting to the reciprocal forms such as total link destruction. The Reciprocal Path Search (RPS) algorithm is a first attempt at this.

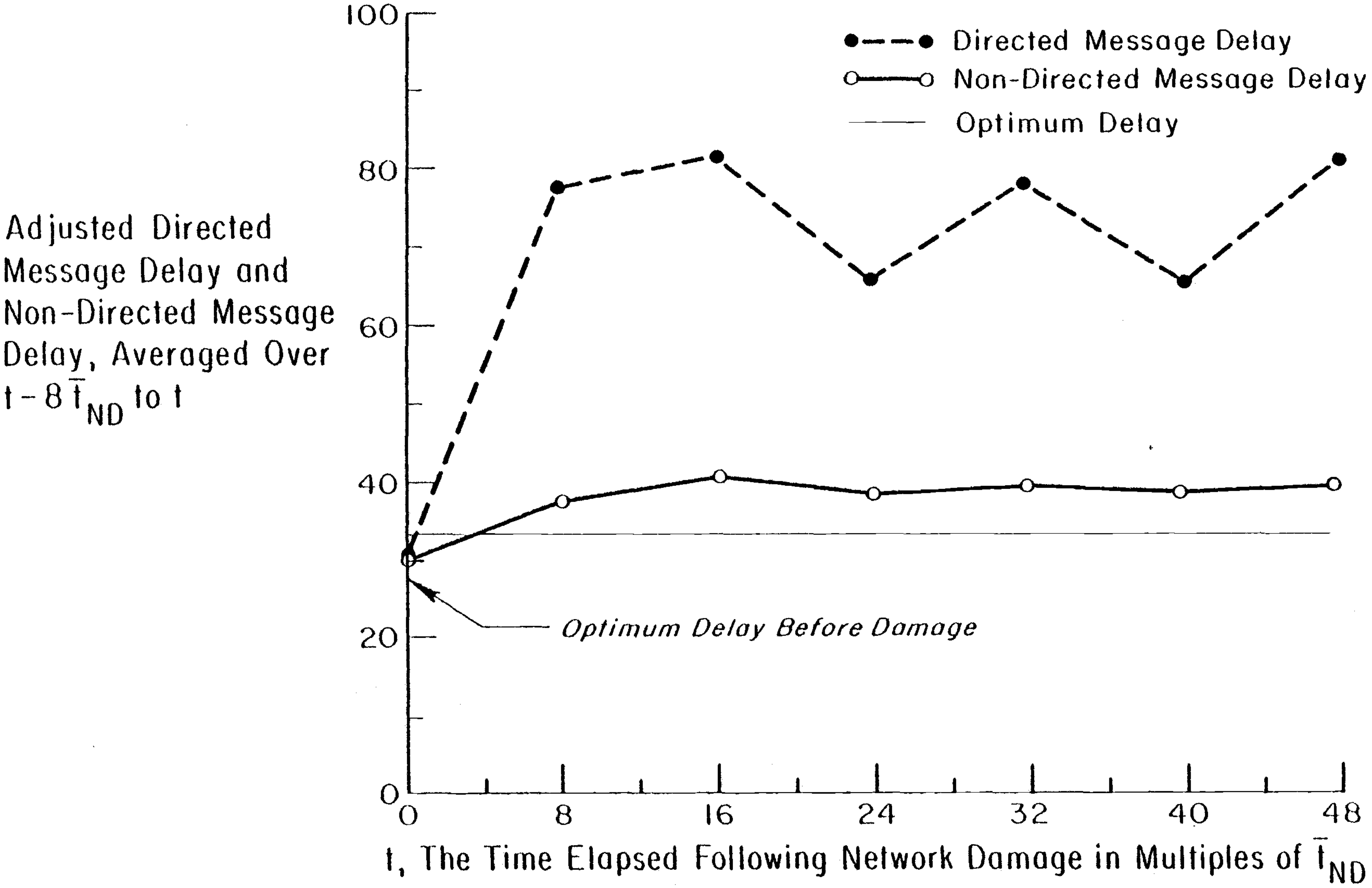


Figure 4. 16. SBL algorithm, three links jammed.

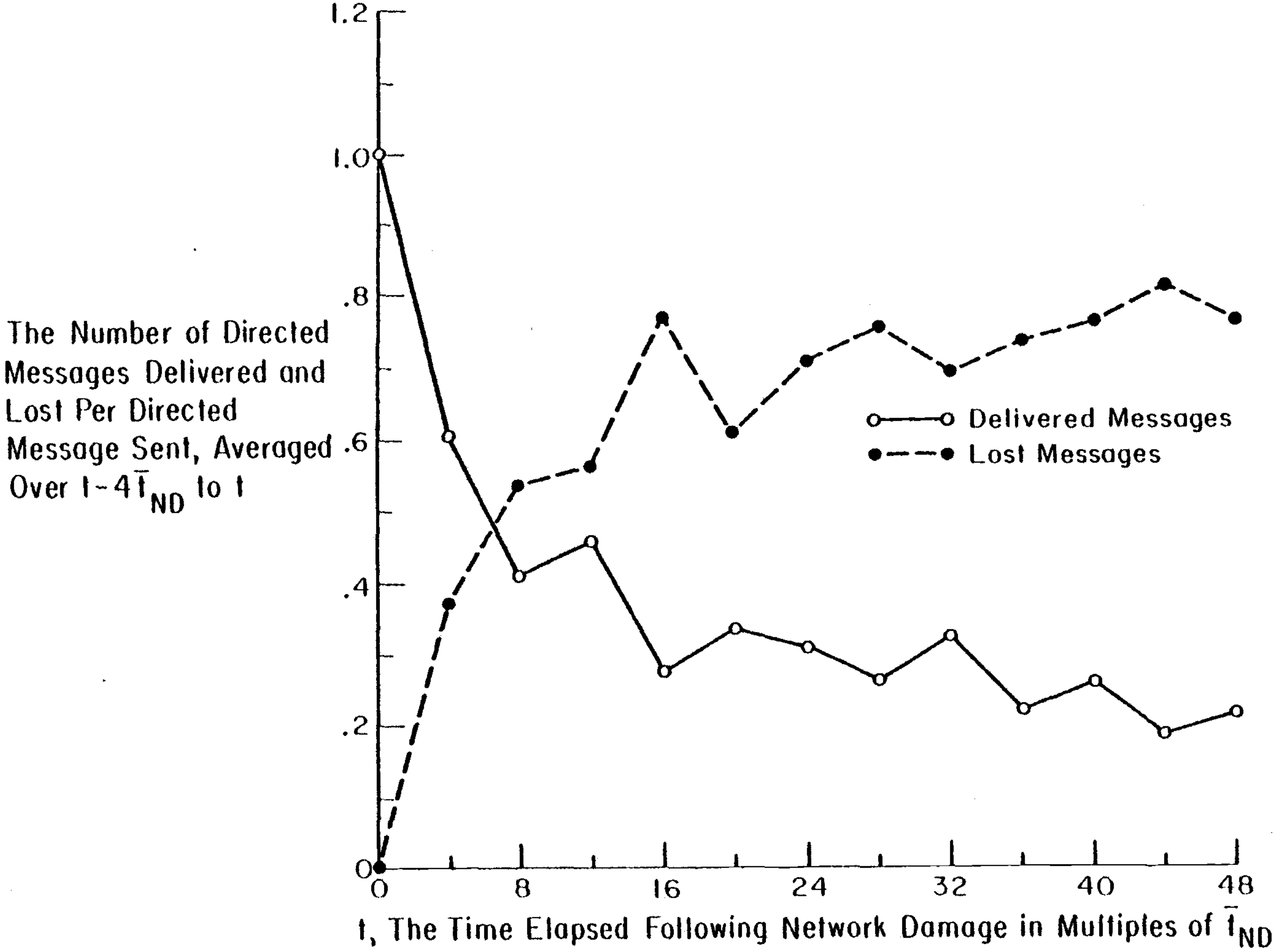


Figure 4. 17. SBL algorithm, six links jammed.

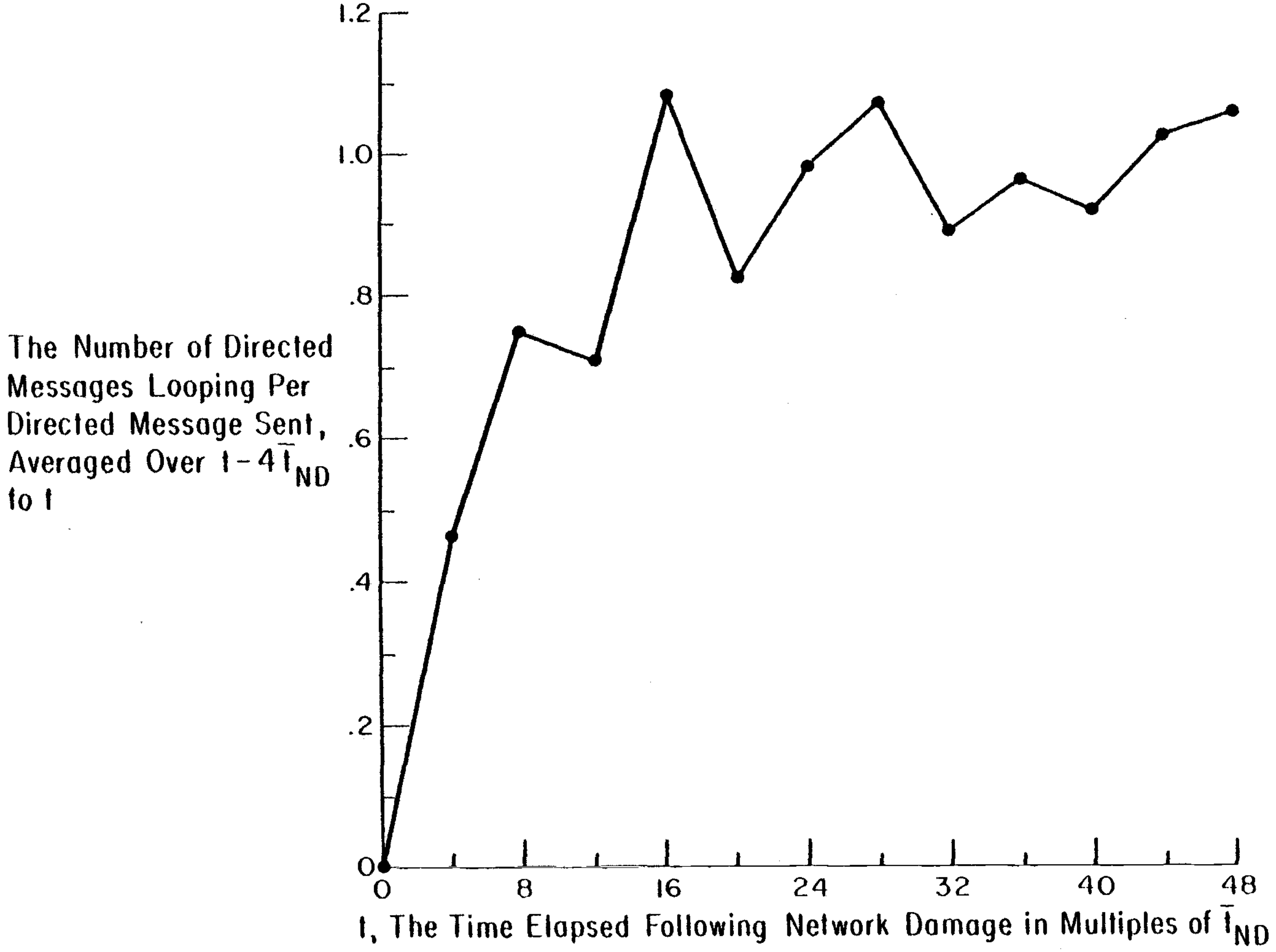


Figure 4. 18. SBL algorithm, six links jammed.

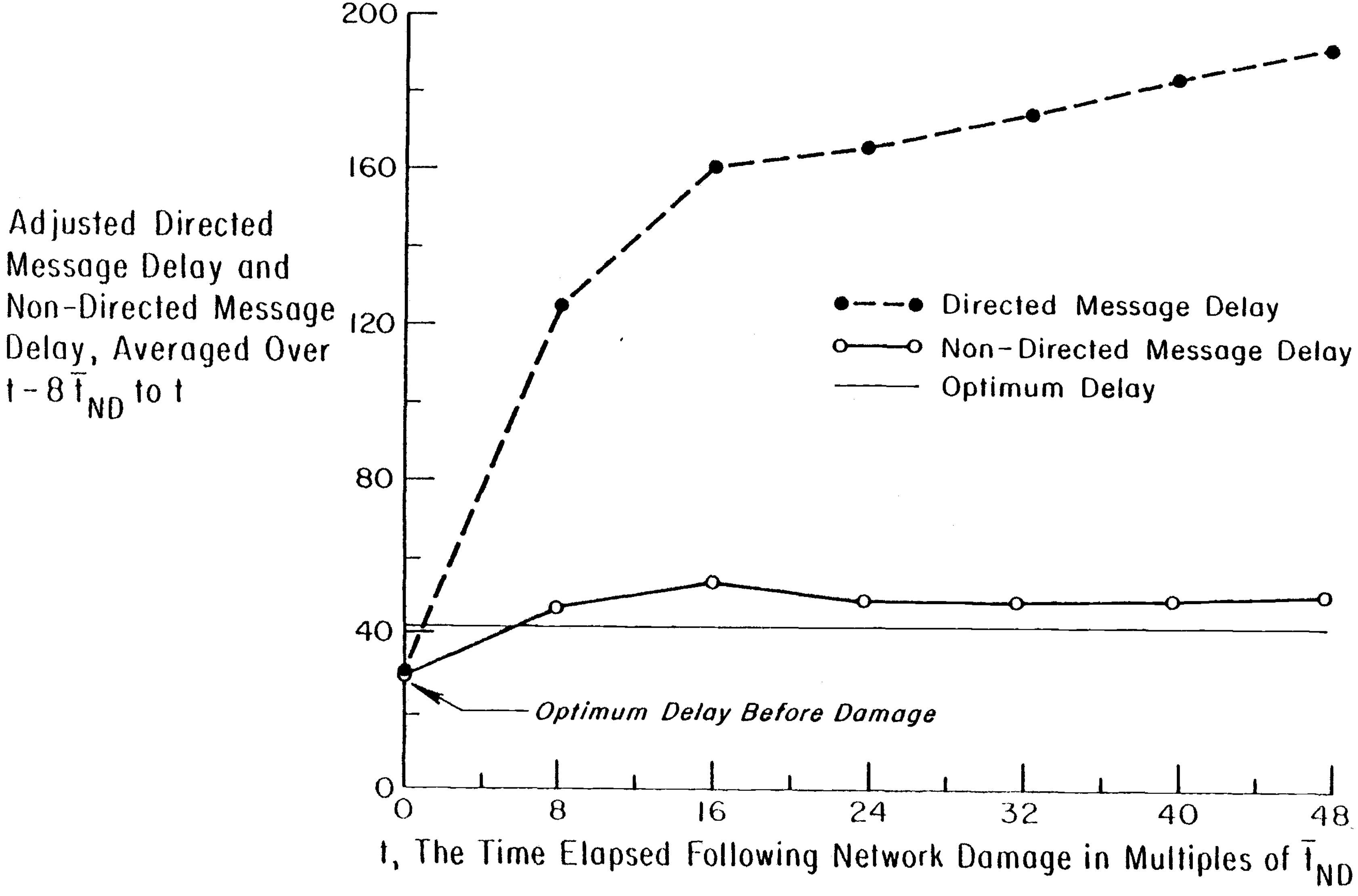


Figure 4. 19. SBL algorithm, six links jammed.

Since the basic problem with the Simple Backwards Learning algorithm is that it assumes reciprocity of the message paths, information concerning non-reciprocal paths must be sent to all nodes in the network, and the nodes must use this information to construct better routing tables. The problem, then, is twofold. First, pertinent jamming information must be communicated to the network nodes. Second, adaptation techniques using this information must be developed.

The first problem is solved by attaching to each non-directed message a "Non-Reciprocal Path Indicator" (NRPI) bit. This bit is initially set to zero when the message originates at its source node. Then, whenever the non-directed message traverses a link which is non-reciprocal, i. e., being jammed, the NRPI is set to one. (A node should be easily able to determine when one of its incoming links is being jammed. Further, if jamming is only one directional, it can readily inform the other end of the link of this fact.) The nodes use the information contained in the NRPI to flood out extra non-directed messages to search out the reciprocal paths, which can then be learned by the routing tables. The details of how this is accomplished are explained below.

Non-Directed Message Handling

How the non-directed messages are routed and used to update the routing tables is indicated by Figures 4.20 and 4.21.

When a non-directed message arrives at a node, it is checked to see if it is the first copy of that message received at that node. If it is, it is next checked to see if it has arrived over a reciprocal path, by checking the NRPI. If the NRPI equals zero, indicating that it did arrive over a reciprocal path, the routing tables are updated and the message is flooded to the node's neighbors. If the NRPI equals one, indicating that the message arrived over a non-reciprocal path, the tables are not updated, since this would teach the tables to incorrectly route the directed messages into the jammed links. Even though the message came in over a non-reciprocal path, it is still flooded on to the node's neighbors to ensure that they receive the information contained in the message with a minimum of delay.

Now suppose that the received copy is not the first received at the node. The copy is still checked to see if it came in over a reciprocal path. If it has, the node further checks to see if it is the first to have come in over a reciprocal path (all previous copies came in over non-reciprocal paths). If it is the first, the node uses it to update a_{PS}, causes the other entries for node S to forget their values, and then floods the message on to its neighbors so that they

Reciprocal Path Search Algorithm
Non-Directed Message Handling

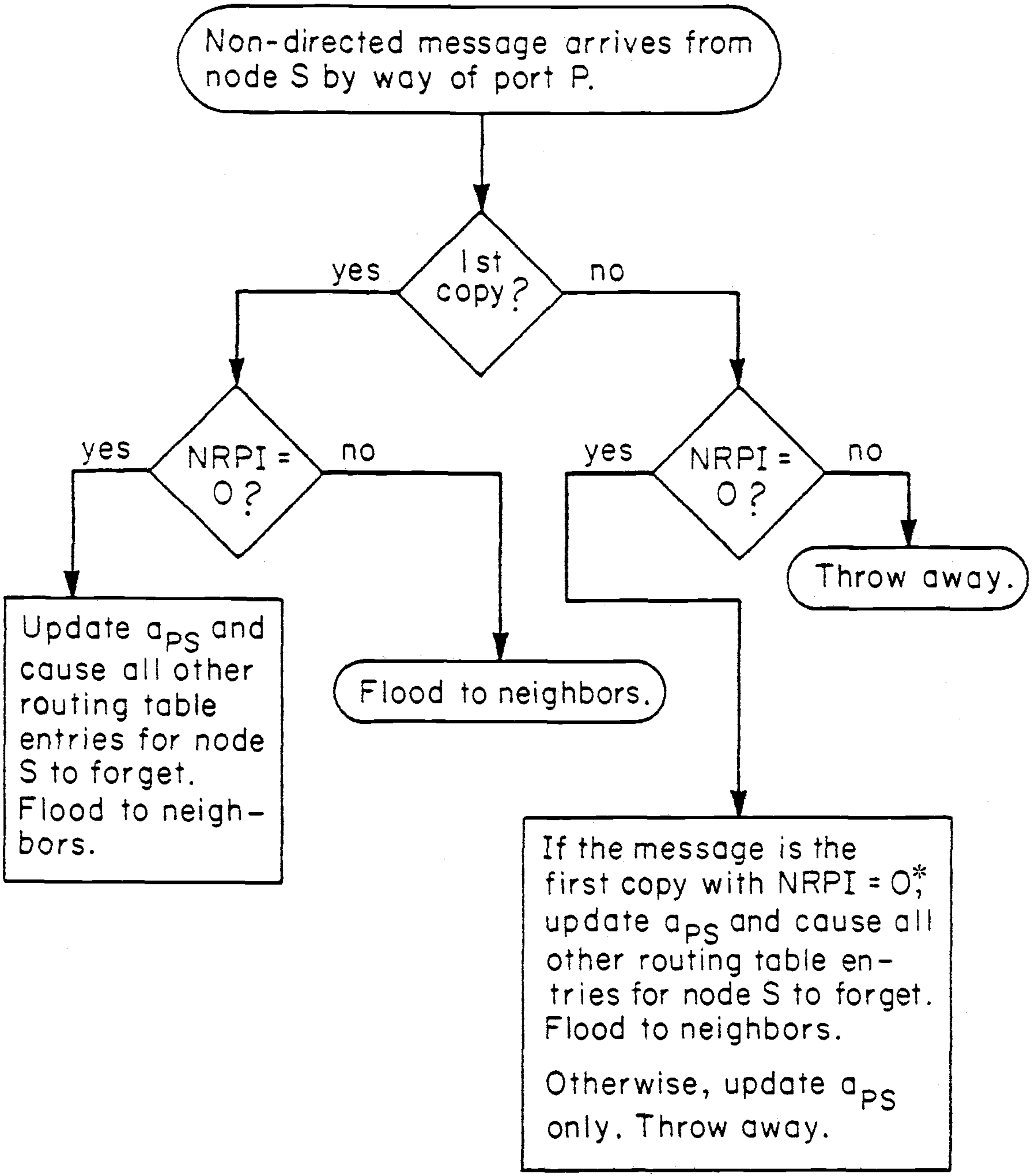

Non-directed message arrives from node S by way of port P.
1st copy?
yes
no
NRPI = 0?
yes
no
NRPI = 0?
yes
no
Update a_{PS} and cause all other routing table entries for node S to forget. Flood to neighbors.
Flood to neighbors.
Throw away.
If the message is the first copy with NRPI = 0;* update a_{PS} and cause all other routing table entries for node S to forget. Flood to neighbors.
Otherwise, update a_{PS} only. Throw away.

Figure 4.20. Conceptual block diagram.

[*] Although not the first copy received at the node, all previous copies had NRPI = 1.

Reciprocal Path Search Algorithm
Non-Directed Message Handling

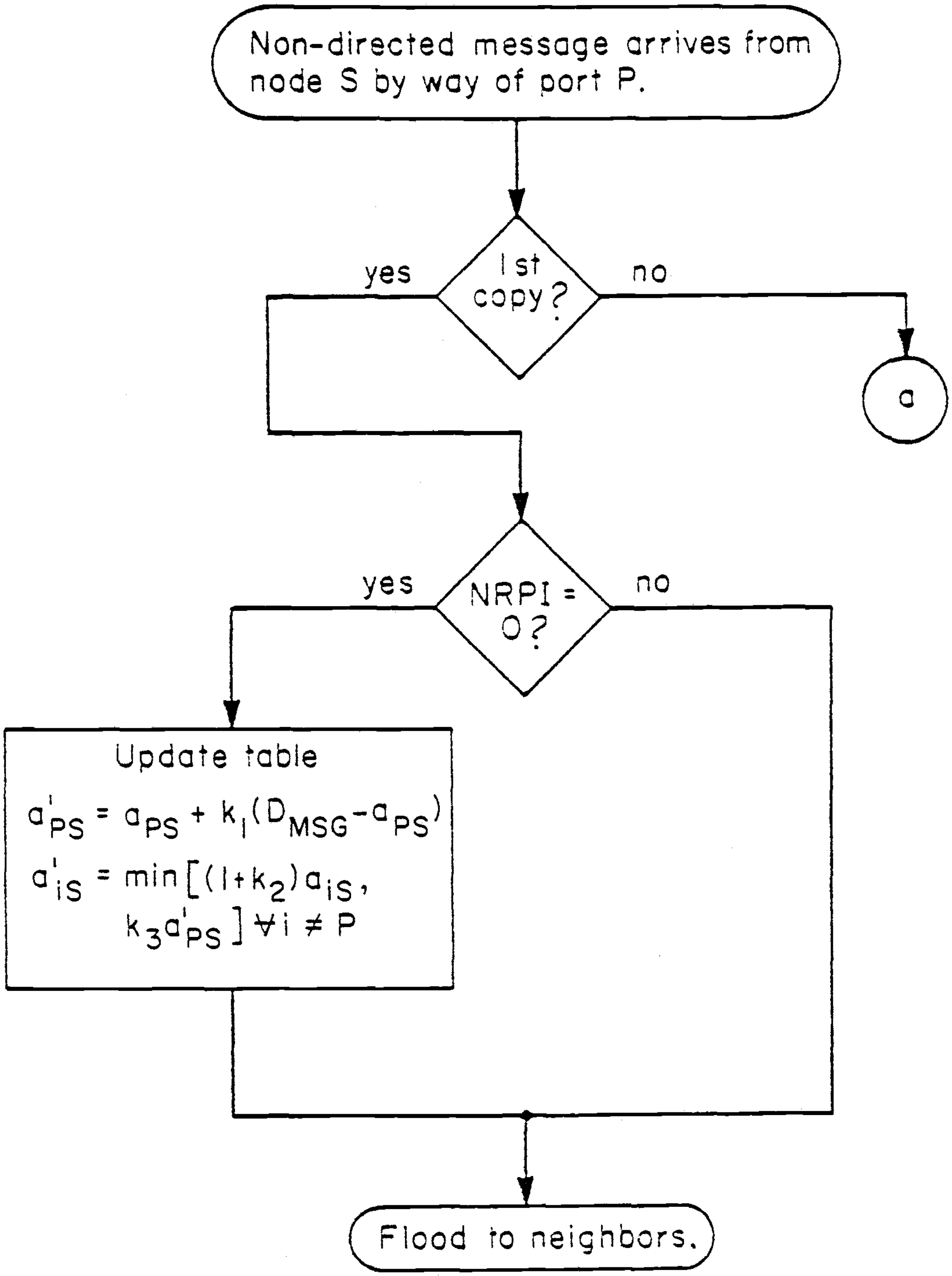

Non-directed message arrives from node S by way of port P.
1st copy?
yes
no
a
NRPI = 0?
yes
no
Update table
$a'_{PS} = a_{PS} + k_1(D_{MSG} - a_{PS})$
$a'_{iS} = \min[(1+k_2)a_{iS}, k_3 a'_{PS}] \;\forall i \neq P$
Flood to neighbors.

Figure 4.21. Detailed block diagram.

Reciprocal Path Search Algorithm
Non-Directed Message Handling (cont.)

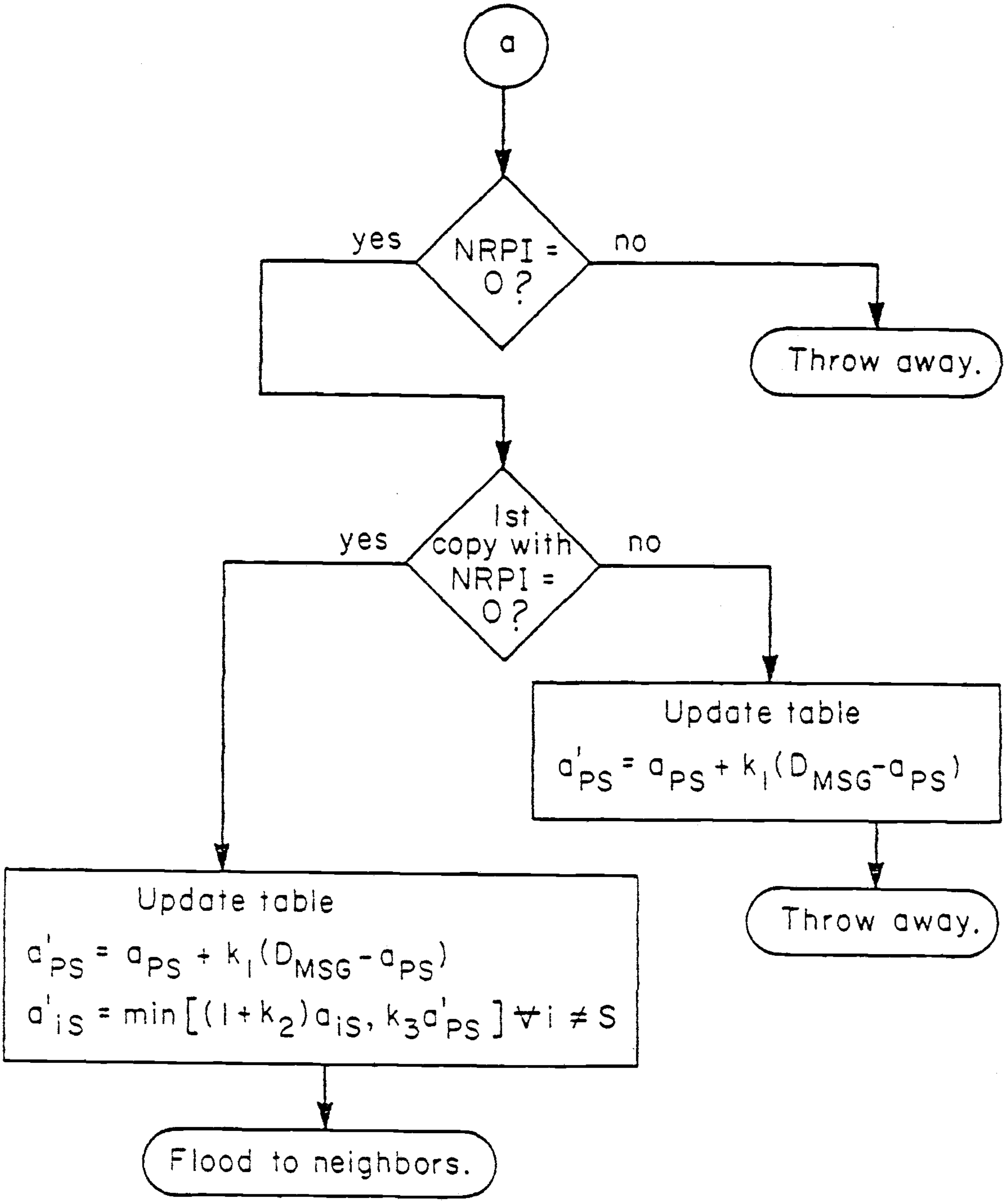

a
NRPI = 0?
yes
no
Throw away.
1st copy with NRPI = 0?
yes
no
Update table
$a'_{PS} = a_{PS} + k_1(D_{MSG} - a_{PS})$
Throw away.
Update table
$a'_{PS} = a_{PS} + k_1(D_{MSG} - a_{PS})$
$a'_{iS} = \min[(1+k_2)a_{iS}, k_3 a'_{PS}] \forall i \neq S$
Flood to neighbors.

Figure 4.21. Continued.

too will have information on this reciprocal path. If, on the other hand, a previous copy of the message had arrived over a reciprocal path, the current copy is allowed only to update a_{PS}, and is then discarded since the neighbors have previously been sent a copy. Finally, if the message is not the first copy received at the node and if it has arrived over a non-reciprocal path, it is merely discarded.

The essential feature of this algorithm is that not only is the first copy of a message flooded to a node's neighbors, but the first copy to come in over a reciprocal path, whether it be the first, second, or third copy received, is also flooded on to the neighbors so that they will receive information about a reciprocal path. This algorithm, as the simulation results verify, is able to search out a reciprocal path between nodes whenever such a path exists. Only the flooded messages which arrive over reciprocal paths are used to update the tables so that the tables will only learn the reciprocal routes.

Directed Message Handling

The details of how the directed messages are handled and routed are contained in Figures 4.22 and 4.23. The procedure is exactly the same as in Simple Backwards Learning. This algorithm relies on the improved routing table updating technique to produce better performance characteristics.

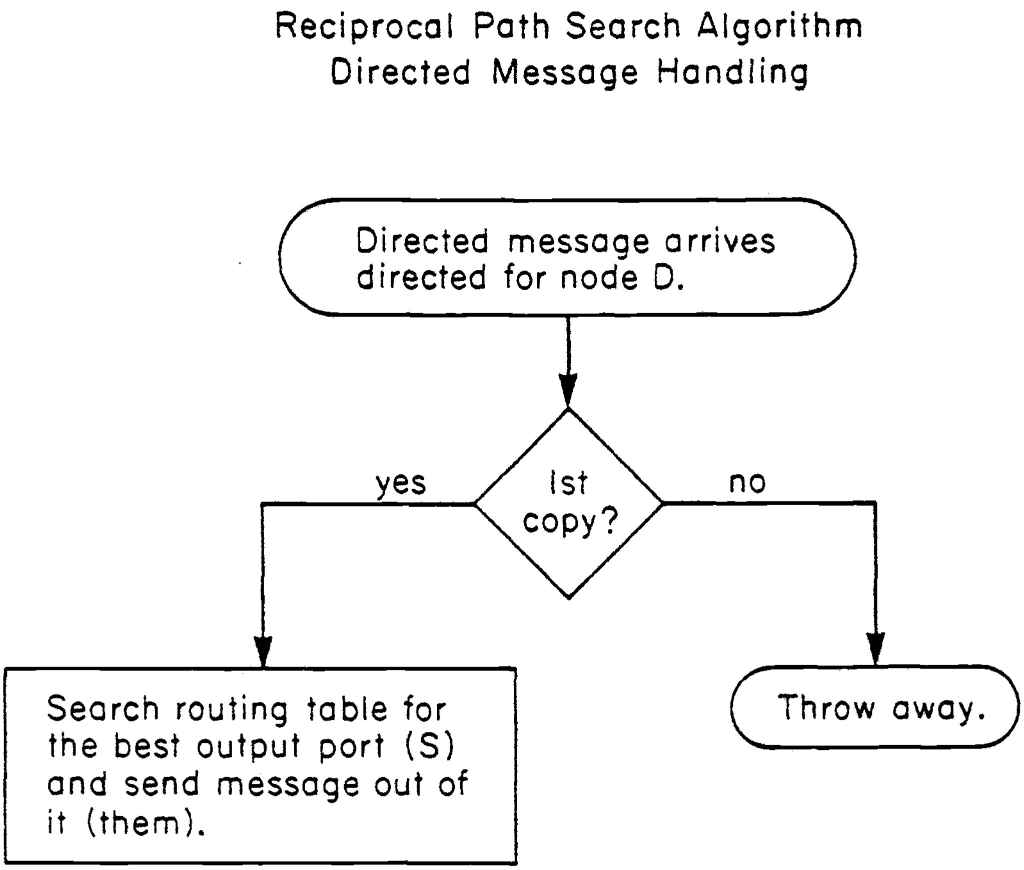

Reciprocal Path Search Algorithm
Directed Message Handling
Directed message arrives directed for node D.
lst copy?
yes
no
Search routing table for the best output port (S) and send message out of it (them).
Throw away.

Figure 4.22. Conceptual block diagram.

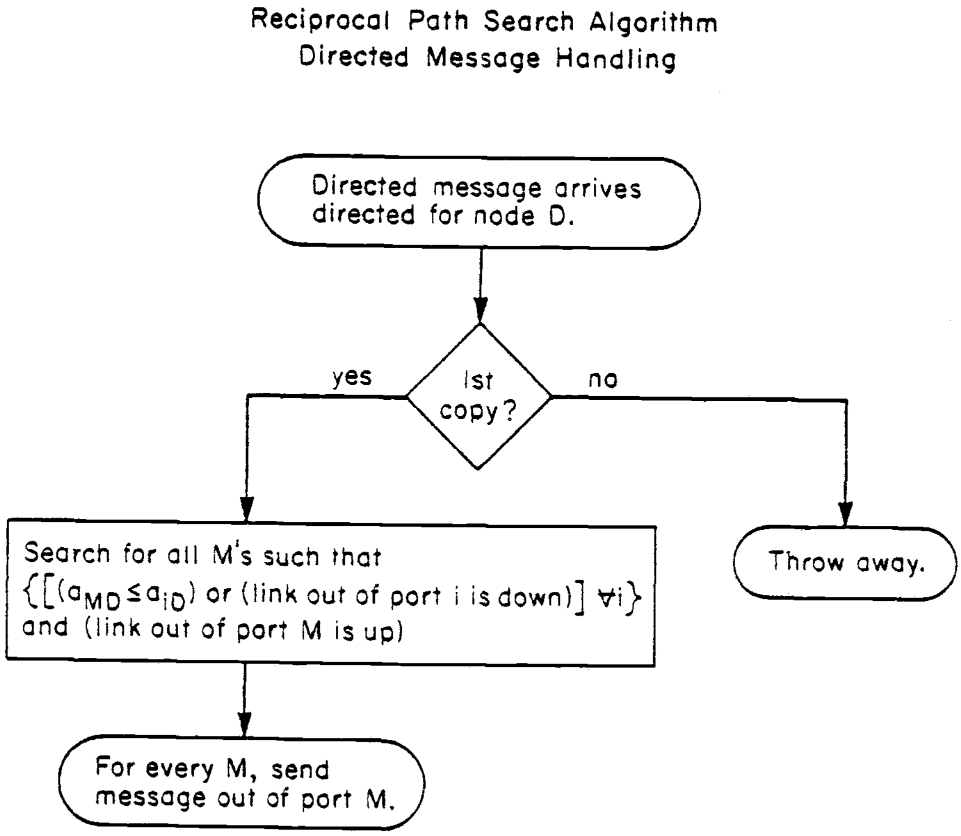

Reciprocal Path Search Algorithm
Directed Message Handling
Directed message arrives directed for node D.
1st copy?
yes
no
Search for all M's such that $\{[(a_{MD} \leq a_{iD})$ or (link out of port i is down)$]\ \forall i\}$ and (link out of port M is up)
Throw away.
For every M, send message out of port M.

Figure 4. 23. Detailed block diagram.

Since this algorithm differs from the Simple Backwards Learning algorithm only in the way it adapts to network damage, the steady state performance characteristics in an undamaged network will not change. Thus, the Reciprocal Path Search algorithm, and all the succeeding algorithms, will only be tested to determine how well they adapt to the three cases of network damage.

Three Links Destroyed

The results for the three links destroyed case are shown in Figures 4.24, 4.25, and 4.26. As should be expected, these results are essentially the same as obtained with the Simple Backwards Learning algorithm. This is because the damage is reciprocal and the Reciprocal Path Search algorithm adapts differently only to non-reciprocal damage. Briefly, these results show a temporary period during which messages are lost and looping. After this period, the algorithm completely adapts, and no further messages are lost. The adjusted directed message delay again has an initial period when it is much larger than the non-directed message delay as a result of the penalty imposed for lost messages. Once the algorithm has adapted, the directed message delay is essentially the same as the non-directed message delay, indicating that the directed messages are being routed over shortest paths.

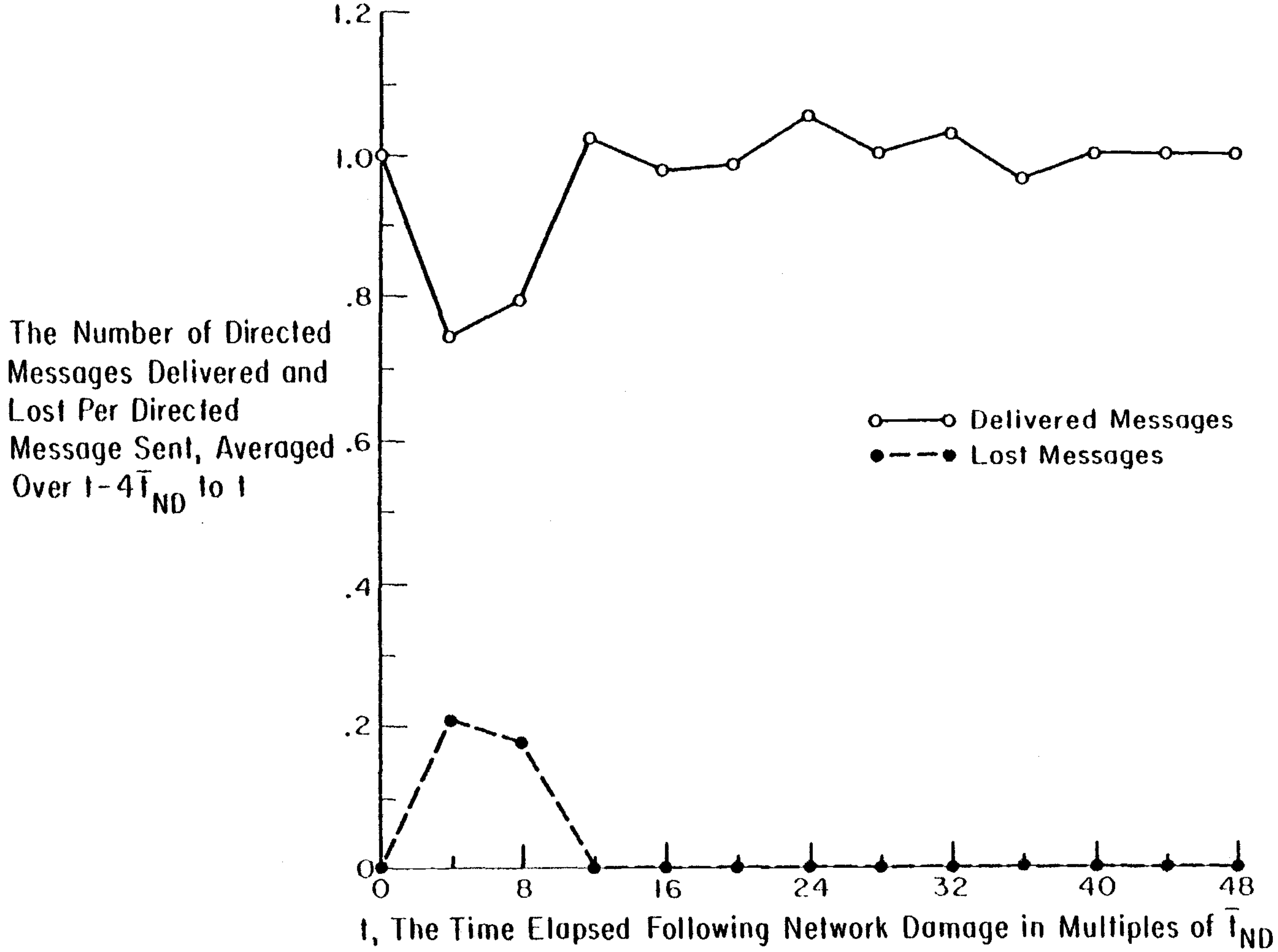


Figure 4.24. RPS algorithm, three links destroyed.

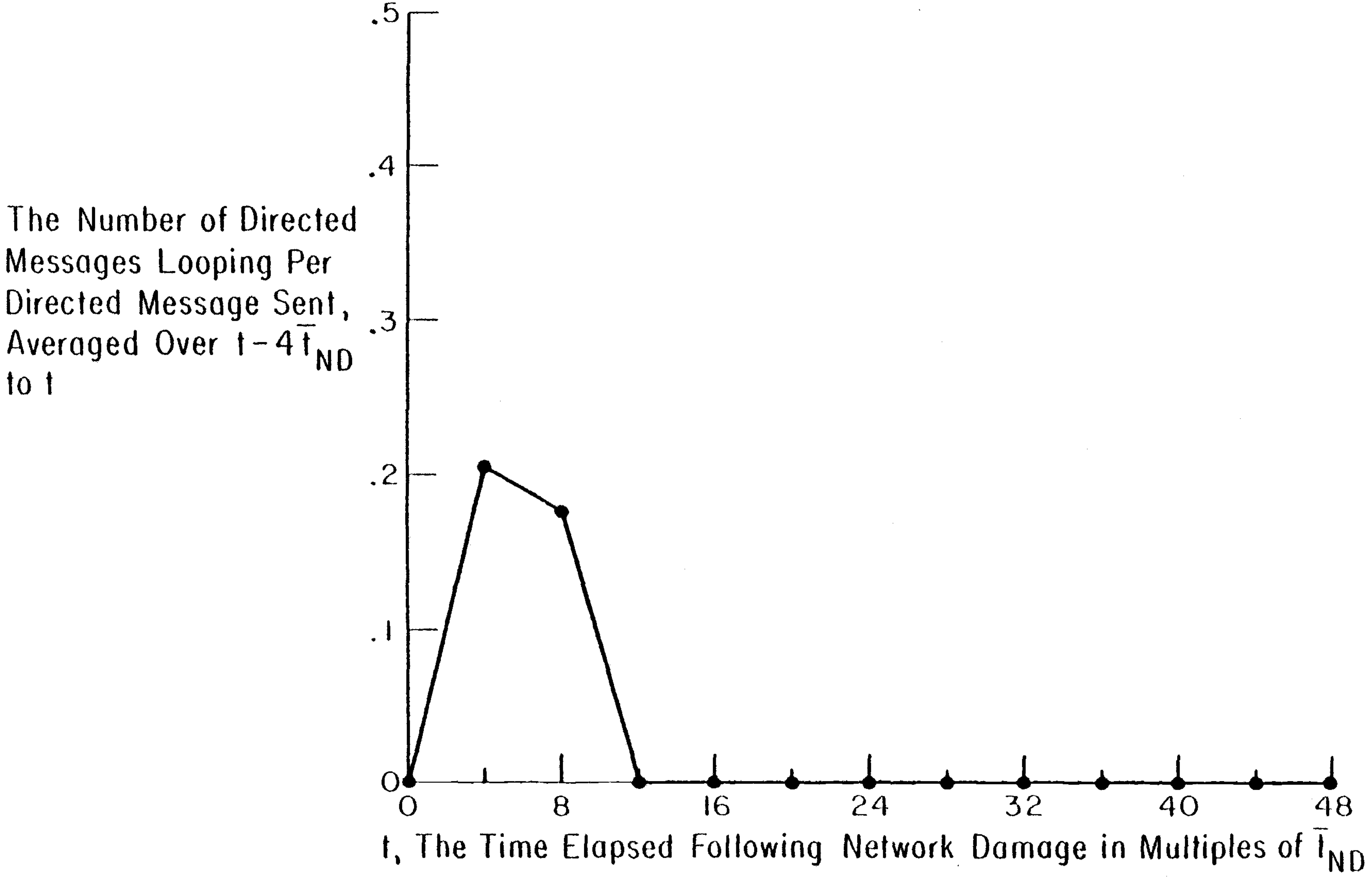


Figure 4.25. RPS algorithm, three links destroyed.

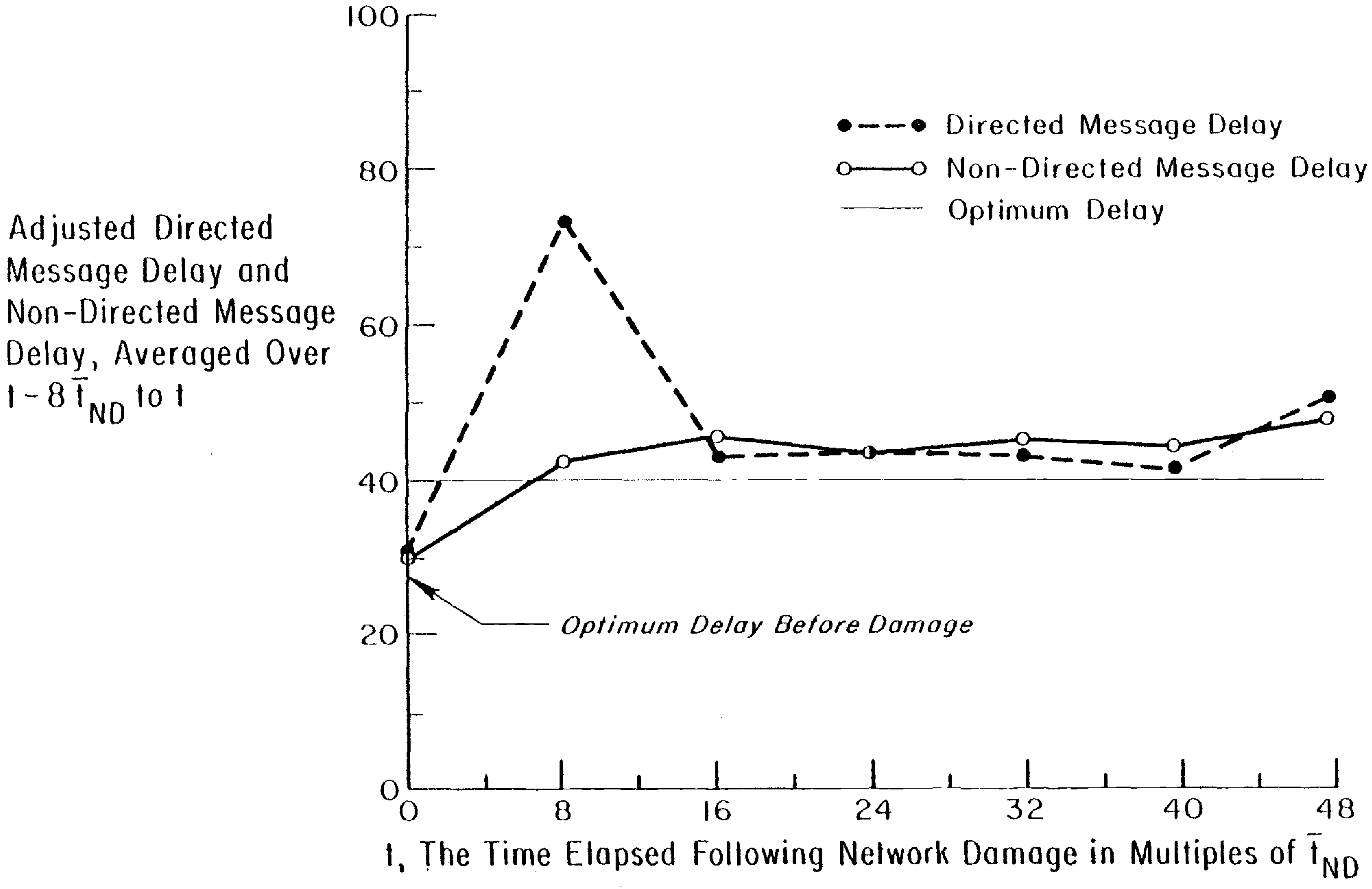


Figure 4.26. RPS algorithm, three links destroyed.

Three Links Jammed

The results for the three links jammed case are shown in Figures 4.27, 4.28, and 4.29. These results differ dramatically from the results obtained with the Simple Backwards Learning algorithm.

Figure 4.27 indicates that following an initial learning period, the Reciprocal Path Search algorithm is able to adapt to the link jamming, and no further messages are lost. This is a result of the algorithm's ability to use the flooded messages to search out the reciprocal paths and provide information to the nodes so that they can update their tables to learn the new reciprocal paths.

The adjusted directed message delay shows an initial period when delay is very large due to the lost messages. However, even after messages are no longer being lost, the directed message delay remains larger than the non-directed message delay, indicating that the algorithm is unable to route all of the directed messages over the shortest paths. This is a result of the Reciprocal Path Search algorithm's built-in function of routing the directed messages completely around jammed links. This prevents lost messages, but it also prevents the directed messages from using a jammed link in its good direction, causing some of them to take longer than necessary paths.

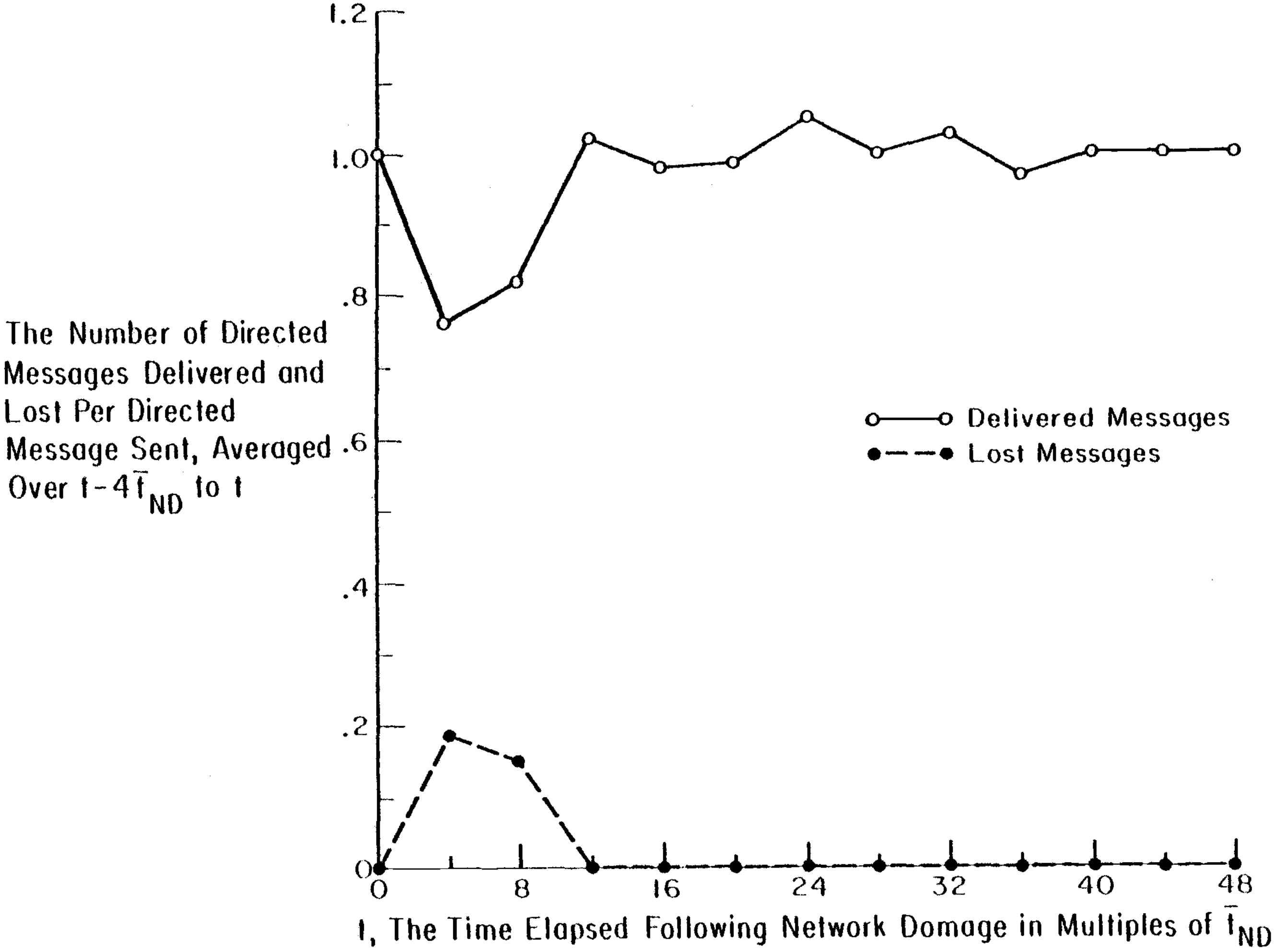


Figure 4. 27. RPS algorithm, three links jammed.

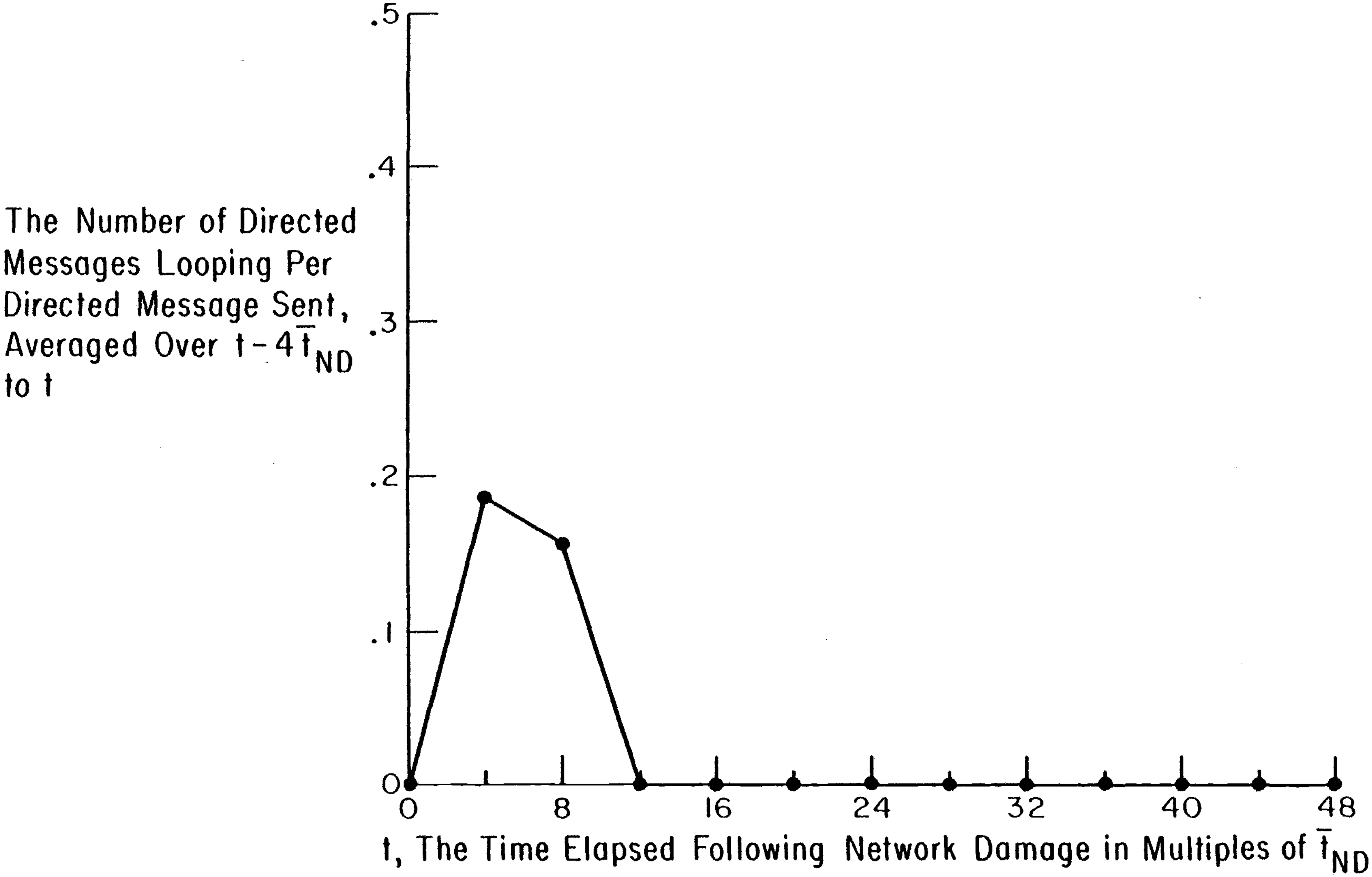


Figure 4.28. RPS algorithm, three links jammed.

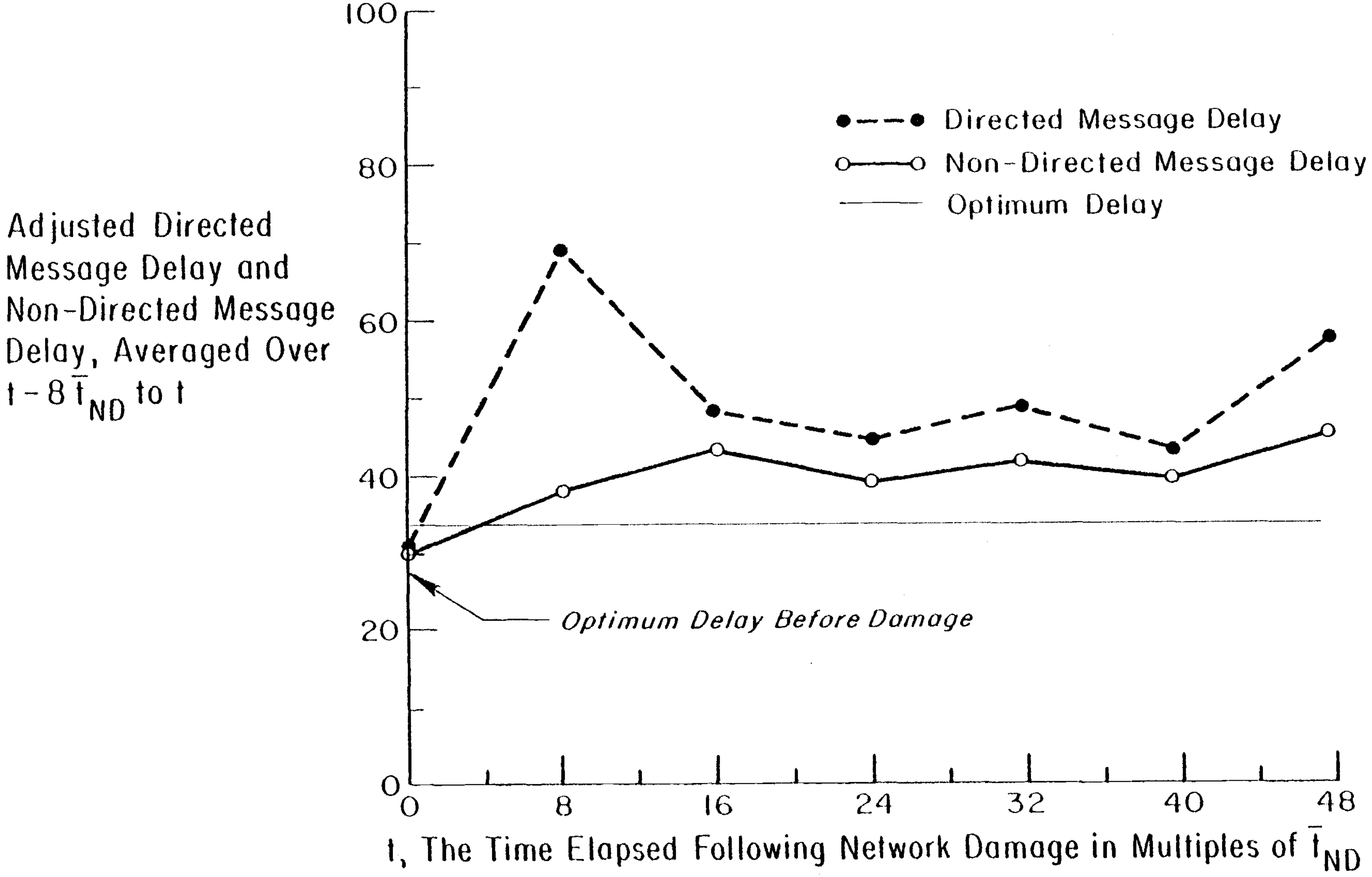


Figure 4.29. RPS algorithm, three links jammed.

Six Links Jammed

The results for the six links jammed case are shown in Figures 4.30, 4.31, and 4.32. Unlike the three links jammed case where all node pairs have at least one reciprocal path between them, the six links jammed case has several node pairs with only non-reciprocal paths between them. Since the Reciprocal Path Search algorithm depends upon the existence of a reciprocal path between nodes, it is unable to adapt to this case of damage. The lack of a reciprocal path means there is nothing new for the routing tables to learn, and also inhibits the forgetting process. This results in the nodes attempting to route the messages by old, no longer good, paths. This problem is evidenced by the results given in Figures 4.30 and 4.31. Lost and looping messages occur continuously after the network damage. There is very little change in the amount of lost and looping messages over the duration of the test, indicating that there is little or no change in the routes over which the messages are being sent.

Figure 4.32 gives the adjusted directed and non-directed message delays. As expected, the adjusted directed message delay remains large due to the large fraction of lost messages.

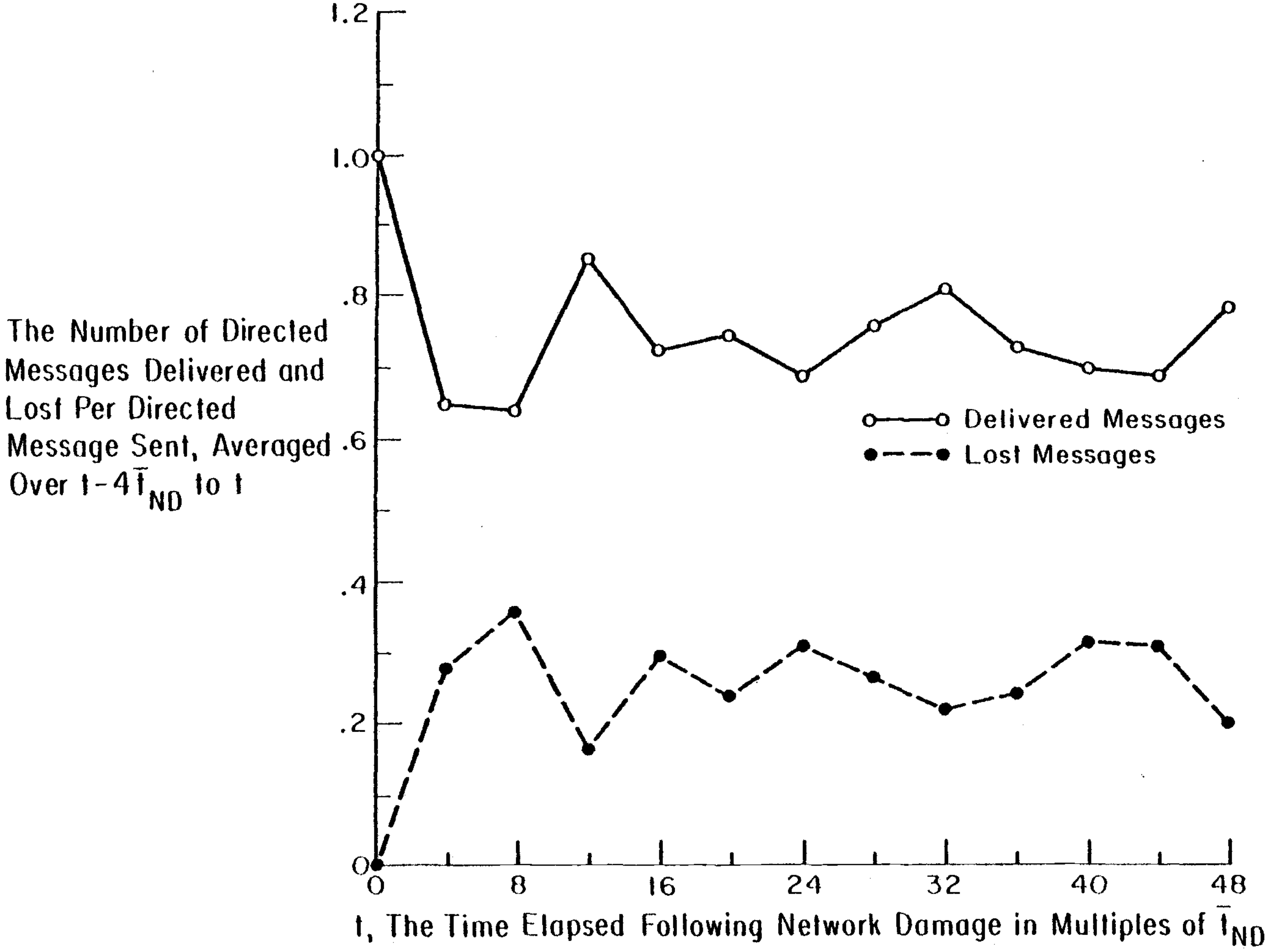


Figure 4. 30. RPS algorithm, six links jammed.

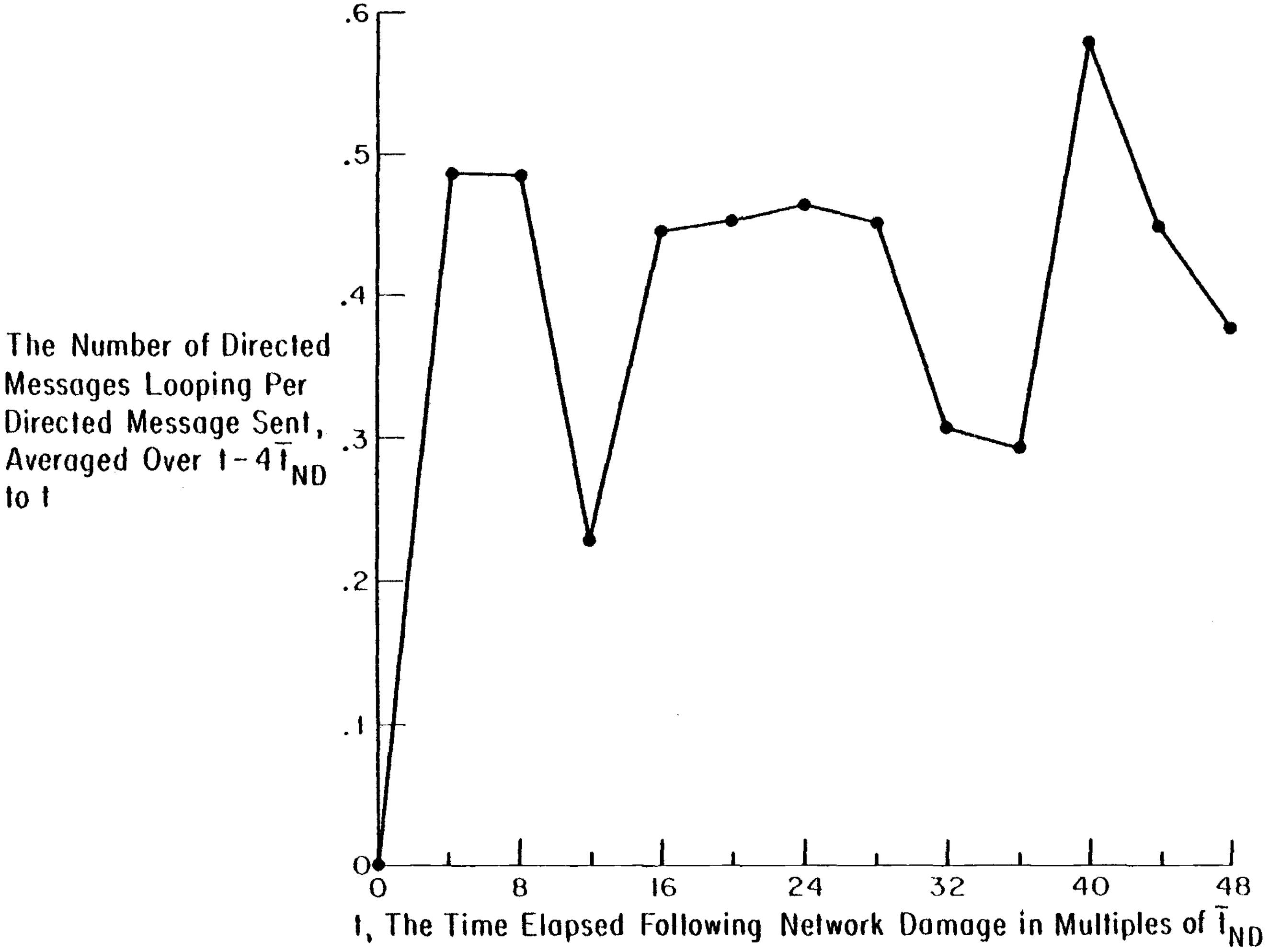


Figure 4.31. RPS algorithm, six links jammed.

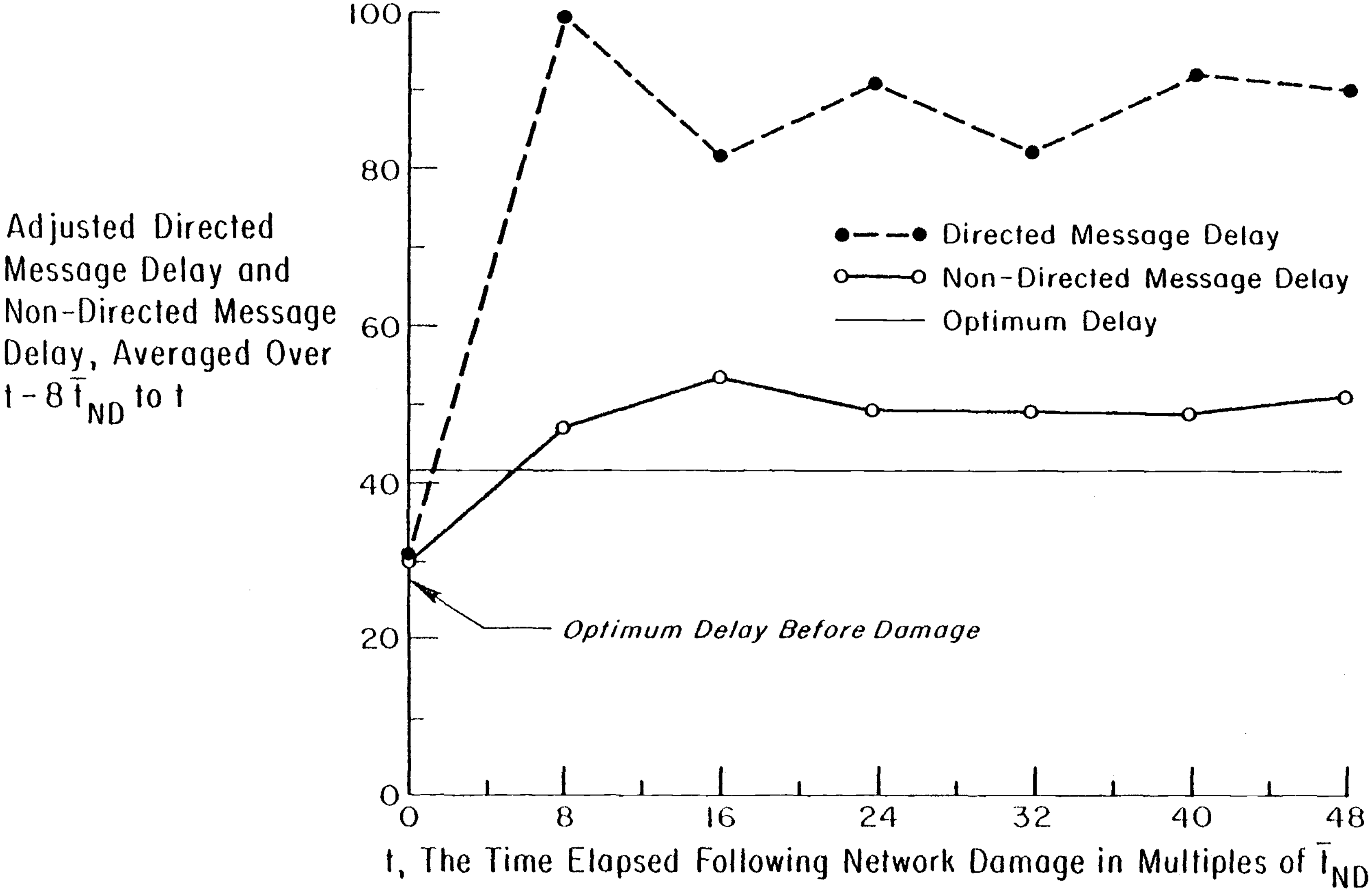


Figure 4. 32. RPS algorithm, six links jammed.

Rapid Reciprocal Path Search Algorithm

The Rapid Reciprocal Path Search (RRPS) algorithm is designed to adapt to cases of jamming more quickly and completely than does the Reciprocal Path Search algorithm, as its name indicates. A reexamination of the results for the Reciprocal Path Search algorithm reveals that it adapted to the case of three links jammed no faster than it did to the case of three links destroyed. The Rapid Reciprocal Path Search algorithm was developed in an attempt to find an algorithm which adapts to jammed links more quickly than it does to destroyed links. (Intuitively, link jamming would appear to be less serious damage than link destruction.) It accomplishes this by using the NRPI bit attached to the non-directed messages to determine when a link has suddenly become non-reciprocal, and allow the algorithm to utilize alternative routing techniques to temporarily circumvent the normal process of learning and forgetting the routing table entries. This results in an algorithm which adapts very quickly.

This algorithm is also designed to sense when no reciprocal path exists between nodes, and thus it is able to take appropriate measures to ensure that the message is delivered to its destination even in extreme cases of network damage, as with the six links jammed case.

The Rapid Reciprocal Path Search algorithm uses three routing tables, as shown in Figure 4.33. The Primary Routing Table (A table), the Selection Table (B table), and the Secondary Routing Table (C table). The Primary Routing Table is used mostly when no network damage exists, or the damage is only reciprocal in nature. It is updated by the first copy of a message, even when that copy comes in over a non-reciprocal path. The Selection Table is updated using the NRPI bits. A Selection Table entry of $b_{ij} = 0$ indicates that the a_{ij} entry was last updated by a message which had an NRPI = 0 (i.e., the message came in over a reciprocal path), and is thus useful for routing directed messages. A value of $b_{ij} = 1$ indicates that the a_{ij} entry was last updated by a message which arrived with an NRPI = 1, and thus signals immediately (circumventing the forgetting process) that the a_{ij} is no longer useful for routing directed messages. When $b_{ij} = 1$, the algorithm uses the Secondary Routing Table entry, c_{ij}, for routing directed messages. The Secondary Routing Table entries are updated only by the extra flooded messages which are generated to search out the reciprocal paths. For any destination node, the algorithm is designed so that the Secondary Routing Table entries are all equal to each other as well as to the Primary Routing Table entries which correspond to previously unused ports, at the time of network damage. This allows the initial updates of the table entries by extra flooded messages which search out

Primary Routing Table (A table)

When in use

a_{ij} = a rough estimate of the delay required to reach node j when port i is used.

Selection Table (B table)

If $b_{ij} = 0$, use a_{ij} for delay estimates.

If $b_{ij} > 0$, use c_{ij} for delay estimates.

Secondary Routing Table (C table)

When in use

c_{ij} = a very rough estimate of the delay required to reach node j when port i is used. All entries updated only by messages using reciprocal paths.

Figure 4.33. RRPS algorithm routing tables.

reciprocal paths to quickly indicate which port to take to use this reciprocal path. If no reciprocal path exists, the fact that all the routing table entries chosen by the Selection Table are equal indicates to the algorithm that no reciprocal path exists, allowing the node to take appropriate action to ensure the message delivery. In this case, the algorithm will flood the directed messages out.

Non-Directed Message Handling

Figures 4.34 and 4.35 explain the non-directed message handling procedure. Suppose a non-directed message with source node S arrives at node X by way of port P. Node X first checks to see if it is the first copy of the message which has been received. If it is, the node then checks the NRPI bit to see if it arrived over a reciprocal path. If the NRPI = 0, indicating the message did arrive over a reciprocal path, a_{PS} is updated and the other a_{iS}'s are made to forget their values, the B table is updated by setting $b_{PS} = 0$, and the C table entries for node S are also all made to forget their values. The message is then flooded to the node's neighbors. Setting $b_{PS} = 0$ tells the directed message routing algorithm to use the a_{PS} entry as a valid estimate of the delay to node S by way of port P. As long as the first copy of the message comes in over a reciprocal path, such as when the network is undamaged or the damage is reciprocal in nature, the above updating procedures causes this algorithm

Rapid Reciprocal Path Search Algorithm
Non-Directed Message Handling

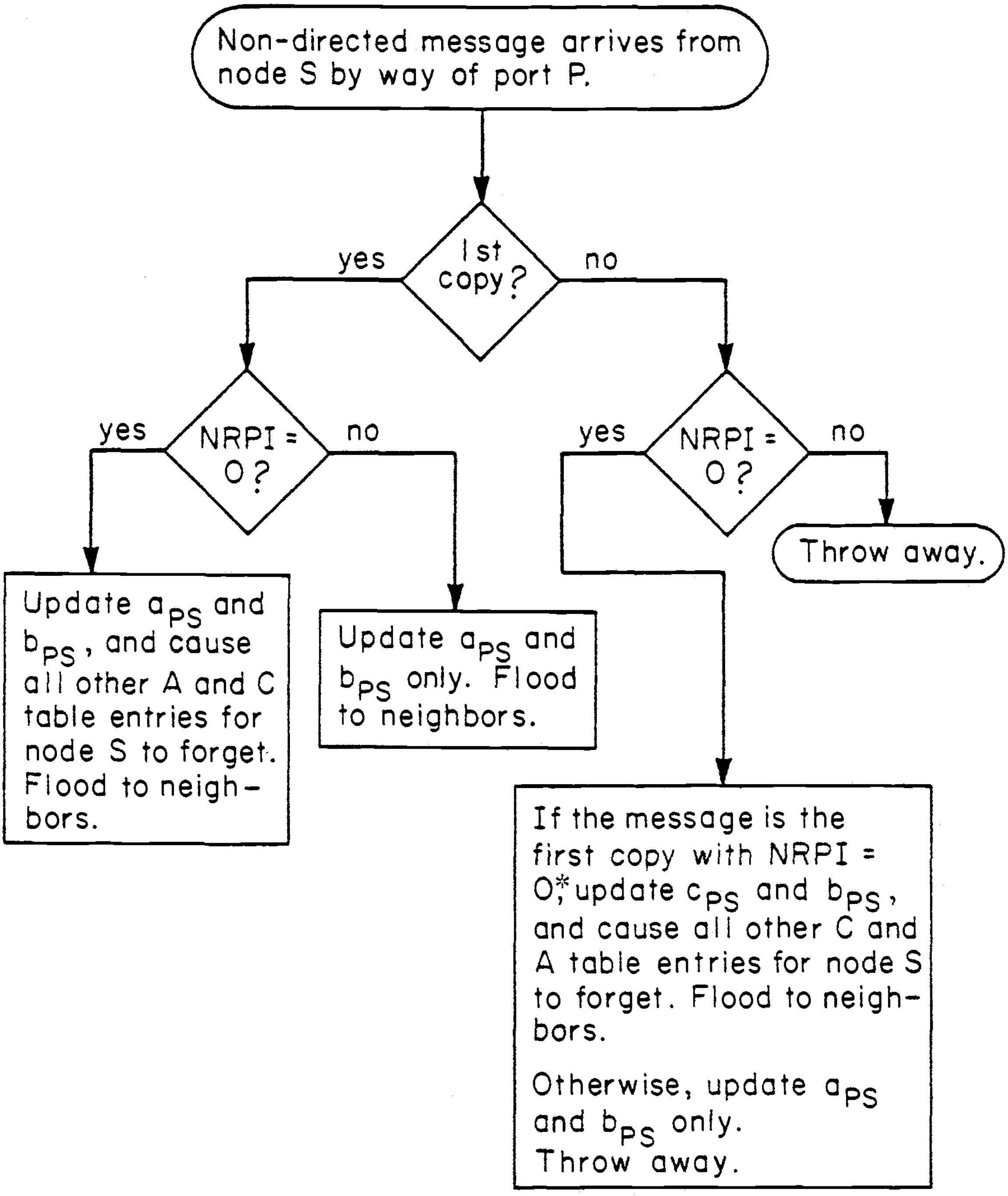

Non-directed message arrives from node S by way of port P.
1st copy?
yes
no
NRPI = 0?
yes
no
NRPI = 0?
yes
no
Throw away.
Update a_{PS} and b_{PS}, and cause all other A and C table entries for node S to forget. Flood to neighbors.
Update a_{PS} and b_{PS} only. Flood to neighbors.
If the message is the first copy with NRPI = 0,* update c_{PS} and b_{PS}, and cause all other C and A table entries for node S to forget. Flood to neighbors.
Otherwise, update a_{PS} and b_{PS} only. Throw away.

Figure 4. 34. Conceptual block diagram.

[*] All previous copies had NRPI = 1.

Rapid Reciprocal Path Search Algorithm
Non-Directed Message Handling

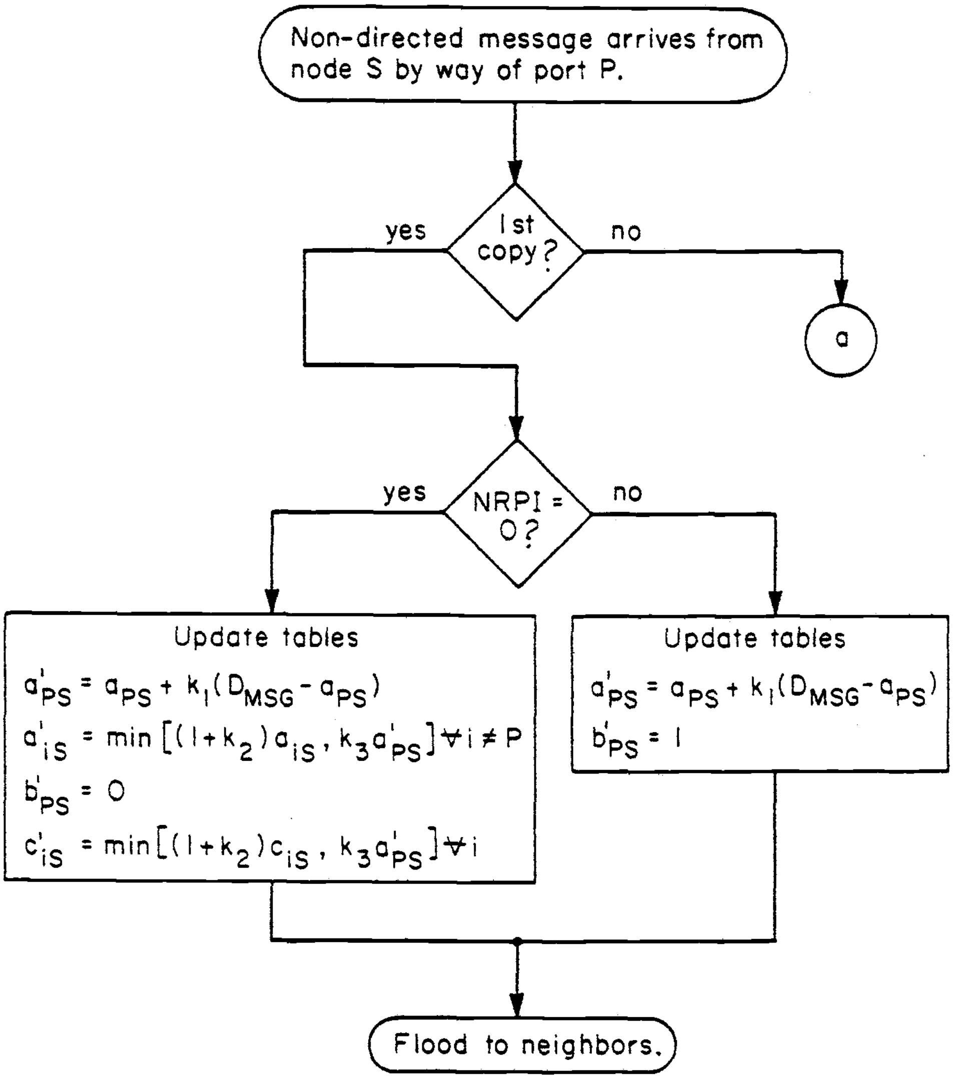

Non-directed message arrives from node S by way of port P.
yes
1st copy?
no
a
yes
NRPI = 0?
no
Update tables
$a'_{PS} = a_{PS} + k_1(D_{MSG} - a_{PS})$
$a'_{iS} = \min[(1+k_2)a_{iS}, k_3 a'_{PS}] \forall i \neq P$
$b'_{PS} = 0$
$c'_{iS} = \min[(1+k_2)c_{iS}, k_3 a'_{PS}] \forall i$
Update tables
$a'_{PS} = a_{PS} + k_1(D_{MSG} - a_{PS})$
$b'_{PS} = 1$
Flood to neighbors.

Figure 4. 35. Detailed block diagram.

Rapid Reciprocal Path Search Algorithm
Non-Directed Message Handling (cont.)

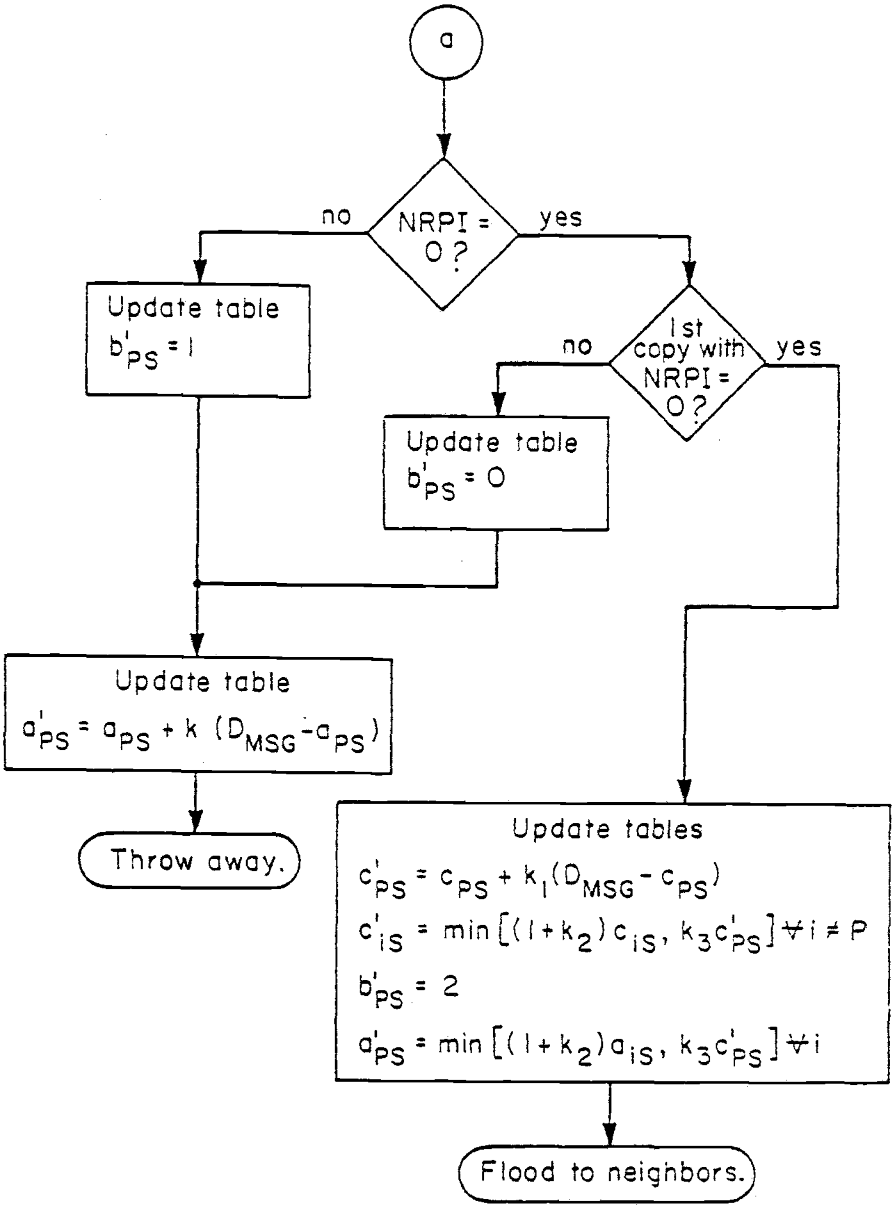

a
NRPI = 0 ?
no
yes
Update table
$b'_{PS} = 1$
1st copy with NRPI = 0 ?
no
yes
Update table
$b'_{PS} = 0$
Update table
$a'_{PS} = a_{PS} + k\ (D_{MSG} - a_{PS})$
Throw away.
Update tables
$c'_{PS} = c_{PS} + k_1(D_{MSG} - c_{PS})$
$c'_{iS} = \min[(1+k_2)c_{iS}, k_3c'_{PS}] \forall i \neq P$
$b'_{PS} = 2$
$a'_{PS} = \min[(1+k_2)a_{iS}, k_3c'_{PS}] \forall i$
Flood to neighbors.

Figure 4. 35. Continued.

to perform essentially the same as did the Simple Backwards Learning and Reciprocal Path Serach algorithms.

If the first copy of the message came in over a non-reciprocal path, i.e., the NRPI = 1, the A table entry a_{PS} is still updated, but no A or C table entries are caused to forget their values, and b_{PS} is set to 1. The message is then flooded to the node's neighbors so that they will have the information contained in the message, and with a minimum of delay. Setting $b_{PS} = 1$ tells the routing algorithm to use the c_{PS} entry (which will be updated when and if a later copy of the message arrives over a reciprocal path) as the delay estimate to node S when a message is sent out of port P. The a_{PS} entry is updated, even though it is not used, so that if the jamming does cease at some future point of time, resulting in the resumption of the use of a_{PS} for routing purposes, it will be kept current.

Now suppose that the message is not the first copy to arrive. If the NRPI = 1, again a_{PS} is updated, b_{PS} is set to one, and the message is discarded. The other A and C table entries are not caused to forget their values.

If, however, this new copy has a NRPI = 0, the node then checks to see if a previous copy also had a NRPI = 0. If that is the case, a_{PS} and b_{PS} are updated, but again the other A and C table entries are not caused to forget their values since the previous copy of the message already completed the forgetting function. The message is discarded since the neighbors have already been sent a copy with NRPI = 0. If, however, no previous copy with a NRPI = 0 has arrived, then the current copy is possibly one of the extra copies generated to search out a reciprocal path. Therefore, the node updates c_{PS}, sets $b_{PS} = 2$ to indicate that c_{PS} should be used for routing purposes (1 could have been used, but 2 was used in the program in order to assist in verifying the correct operation of the algorithm), and the other C and A table entries for node S are caused to forget their values. This copy of the message is then flooded on to the neighbors so that they can receive information on the reciprocal path.

Directed Message Handling

The procedure for handling directed messages is described in Figure 4.36 and 4.37.

The routing algorithm for directed messages is very similar to the one used in the Simple Backwards Learning and the Reciprocal Path Search algorithms. There are two major differences. First, the Rapid Reciprocal Path Search algorithm draws its delay estimates from either of two routing tables, whereas only one routing table was used previously. If a message arrives directed for node D, the delay

Rapid Reciprocal Path Search Algorithm
Directed Message Handling

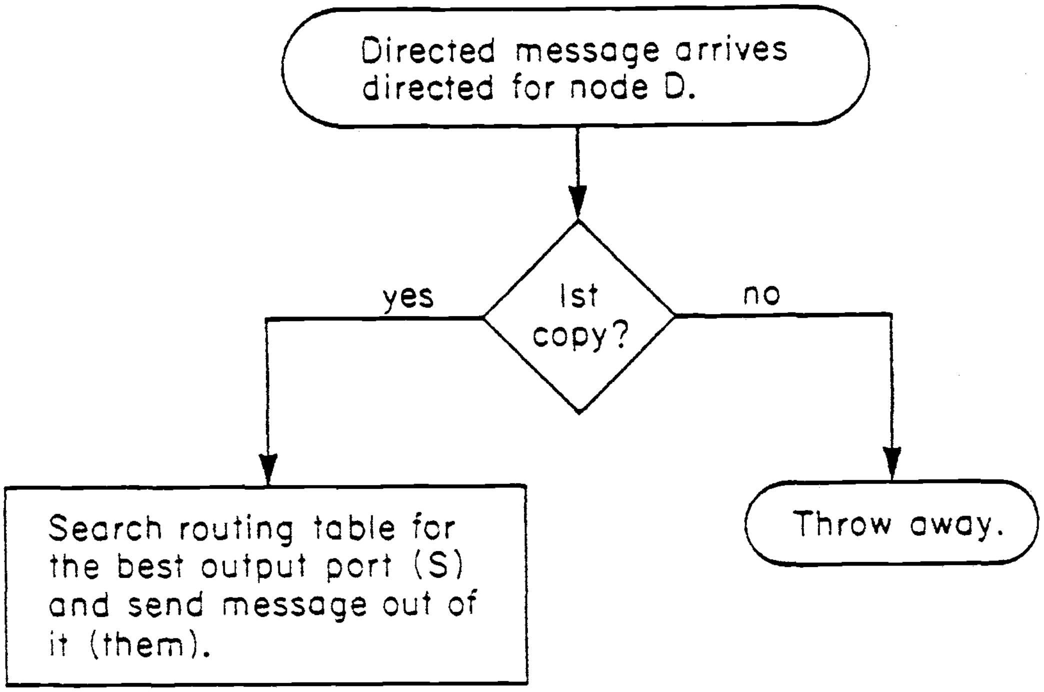

Directed message arrives directed for node D.
yes
Ist copy?
no
Search routing table for the best output port (S) and send message out of it (them).
Throw away.

Figure 4. 36. Conceptual block diagram.

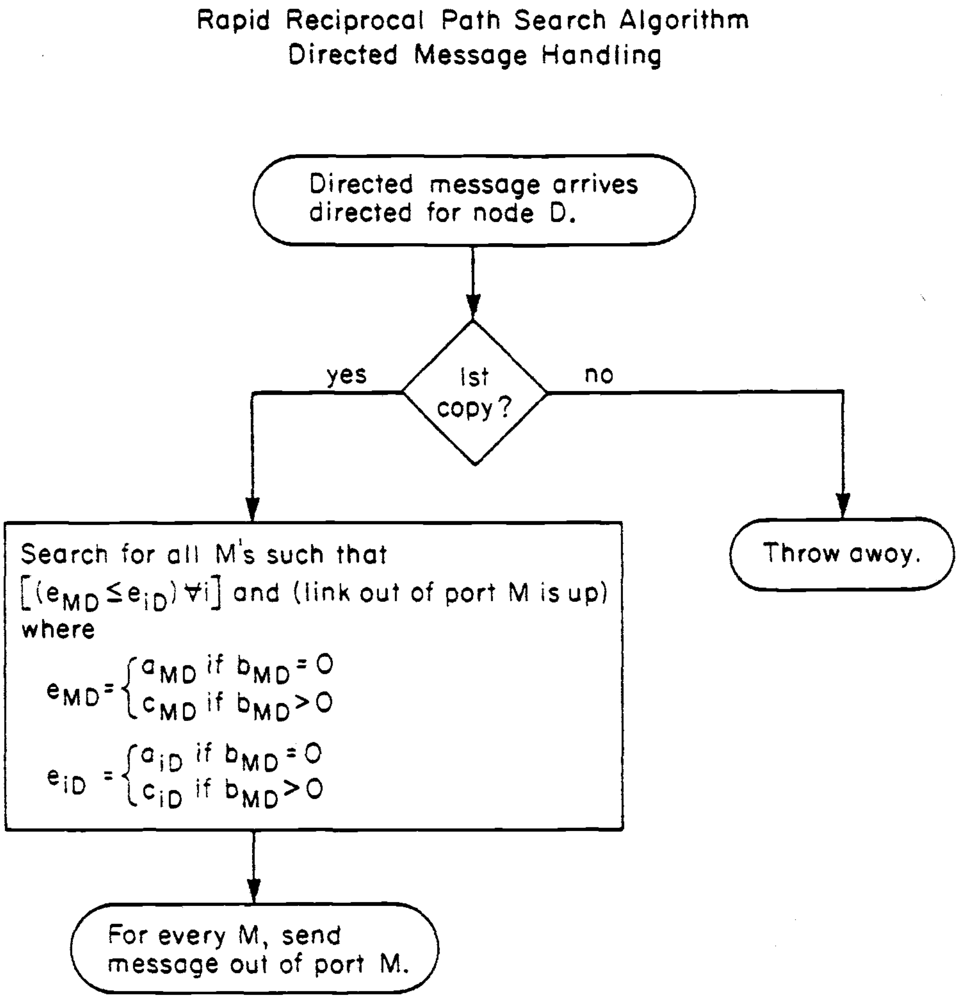

Rapid Reciprocal Path Search Algorithm
Directed Message Handling
Directed message arrives directed for node D.
yes
Ist copy?
no
Search for all M's such that
$[(e_{MD} \le e_{iD}) \forall i]$ and (link out of port M is up)
where
$e_{MD} = \begin{cases} a_{MD} \text{ if } b_{MD} = 0 \\ c_{MD} \text{ if } b_{MD} > 0 \end{cases}$
$e_{iD} = \begin{cases} a_{iD} \text{ if } b_{MD} = 0 \\ c_{iD} \text{ if } b_{MD} > 0 \end{cases}$
Throw awoy.
For every M, send message out of port M.

Figure 4. 37. Detailed block diagram.

estimate used is now e_{iD}, where

$$e_{iD} = \begin{cases} a_{iD} & \text{if } b_{iD} = 0 \\ c_{iD} & \text{if } b_{iD} > 0 \end{cases} \qquad (4.13)$$

The other major difference is that, previously, port M was considered better than port N if either the delay estimate for port M is less than the delay estimate for port N, or the link out of port N is down (i.e., not able to carry messages). This latter step, checking to see if port N is down, is no longer done since the Rapid Reciprocal Path Search algorithm depends upon the b_{ij}'s to reflect this information in the routing algorithm. This works fine for jammed links, but the b_{ij}'s do not contain any information on destroyed links. The looping which occurred before was primarily due to messages sent to a node which was previously, but is no longer, connected to the destination node. The disconnected node could no longer deliver the message to its destination, so it sent it back to the nodes to which it was still connected, resulting in ping-ponging (a form of looping), and occasionally allowing the message to proceed on to its destination. With the Rapid Reciprocal Path Search algorithm, the disconnected node merely recognizes that it is no longer connected to the destination node, and simply does not send out the message. This prohibits looping, decreases network congestion, and also results in a very slight increase in messages lost for the case of destroyed links. However, as will be seen below, this algorithm provides a dramatic improvement in the message delivery for the jammed links cases.

Three Links Destroyed

The results for the three links destroyed case are contained in Figures 4.38 and 4.39. As explained above, looping does not occur, so that plot is omitted. The results are nearly identical to the results obtained by the previous algorithms for this case of network damage. This is expected since the primary difference between this and previous routing algorithms is in how it adapts to non-reciprocal forms of damage, such as jamming.

Three Links Jammed

The results for the three links jammed case are contained in Figures 4.40, 4.41, and 4.42. The Rapid Reciprocal Path Search algorithm is able to dramatically reduce the number of lost and looping messages, when compared to the previous algorithms. Messages are lost for only a period of duration $4\bar{t}_{ND}$. In addition, only about one-third as many messages were lost as with the Reciprocal Path Search algorithm.

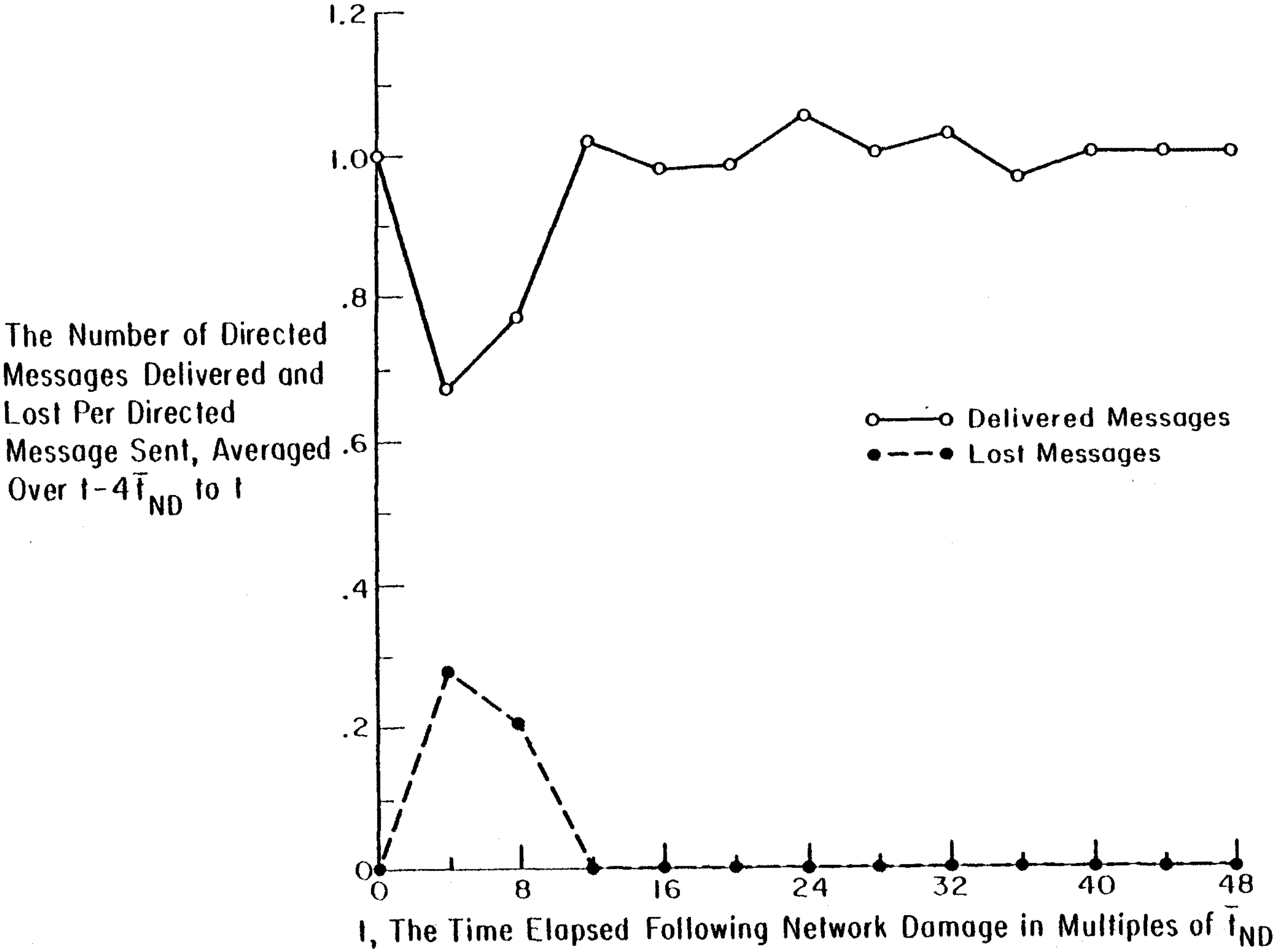


Figure 4. 38. RRPS algorithm, three links destroyed.

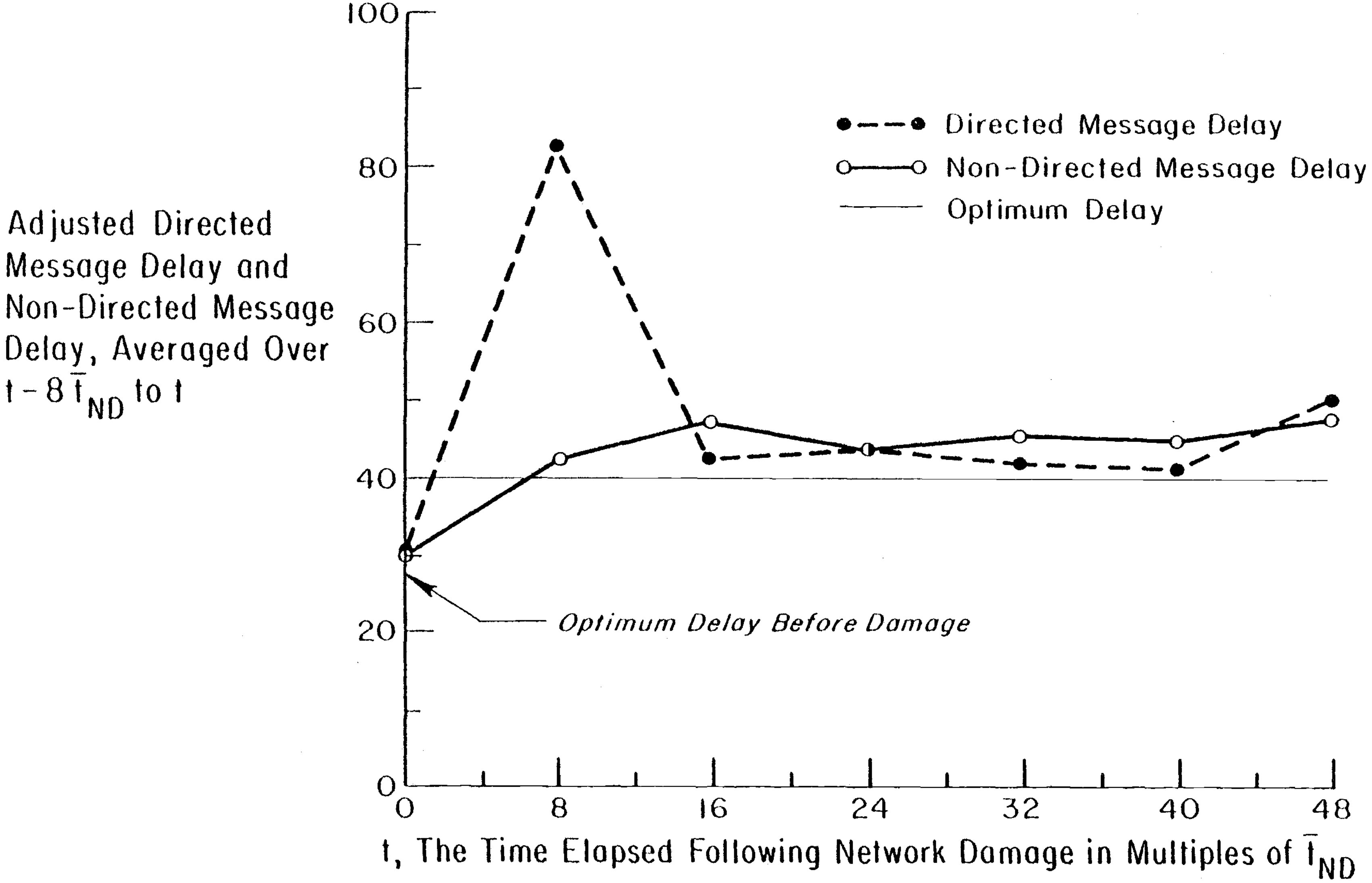


Figure 4.39. RRPS algorithm, three links destroyed.

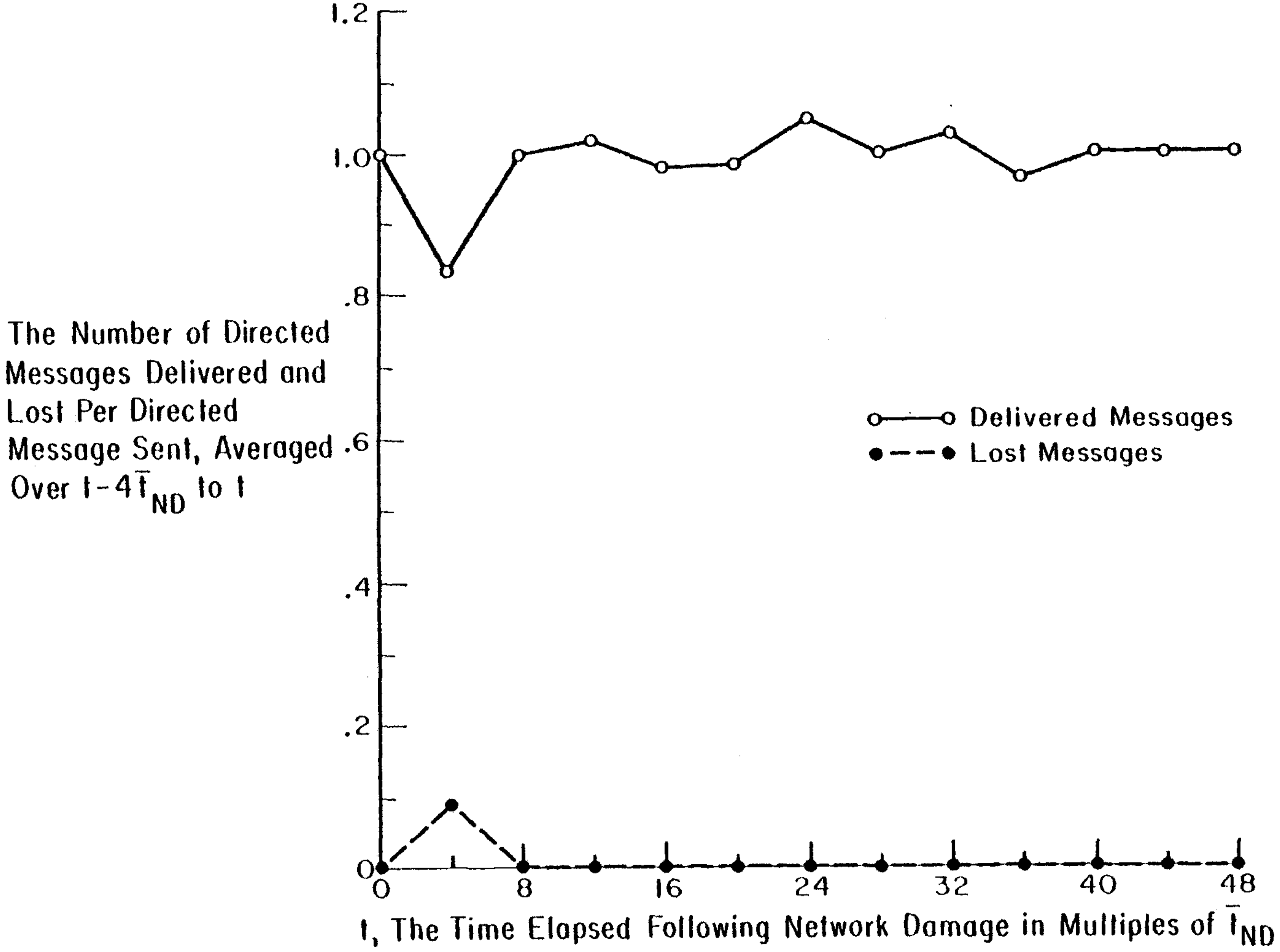


Figure 4. 40. RRPS algorithm, three links jammed.

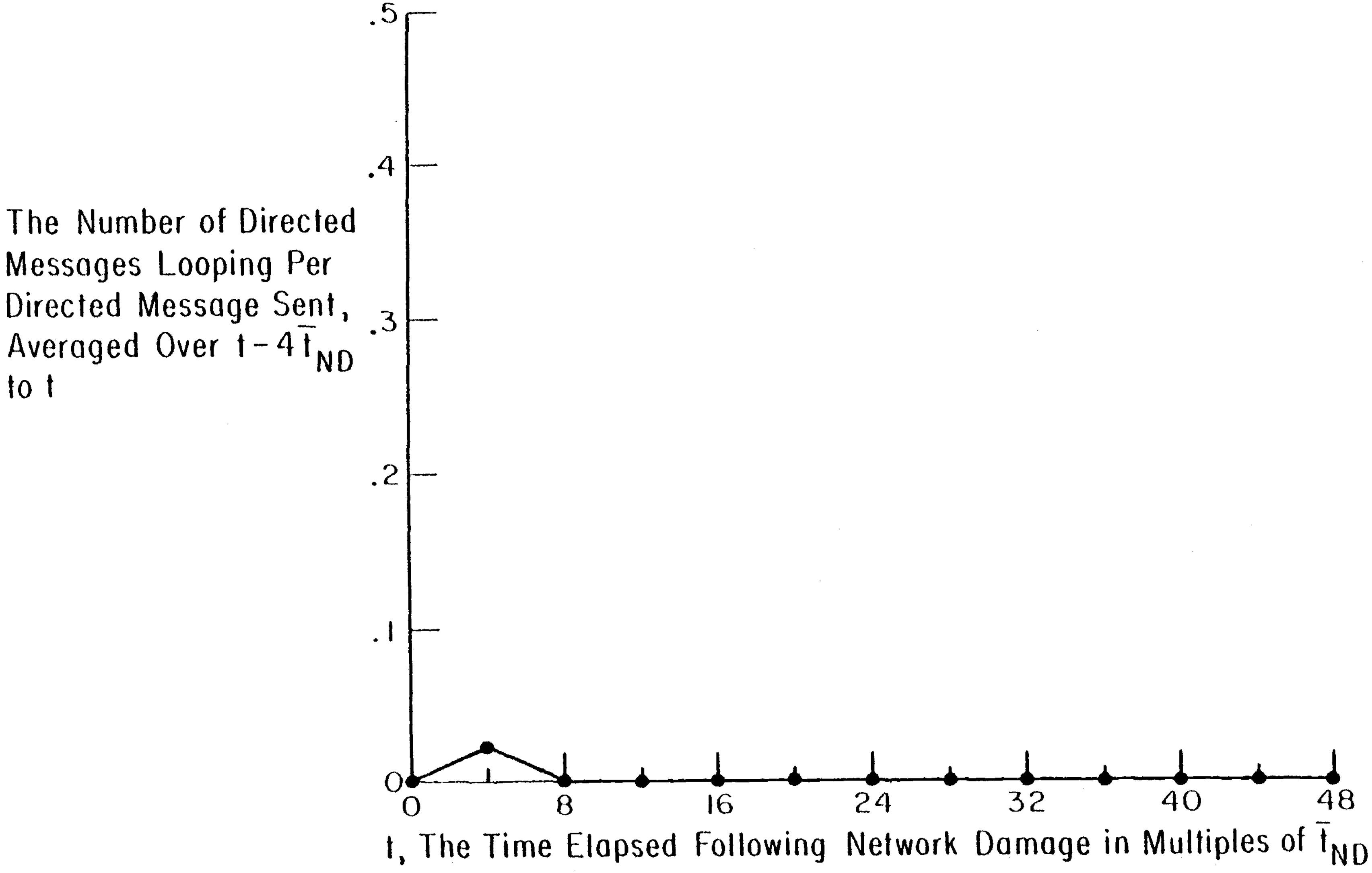


Figure 4.41. RRPS algorithm, three links jammed.

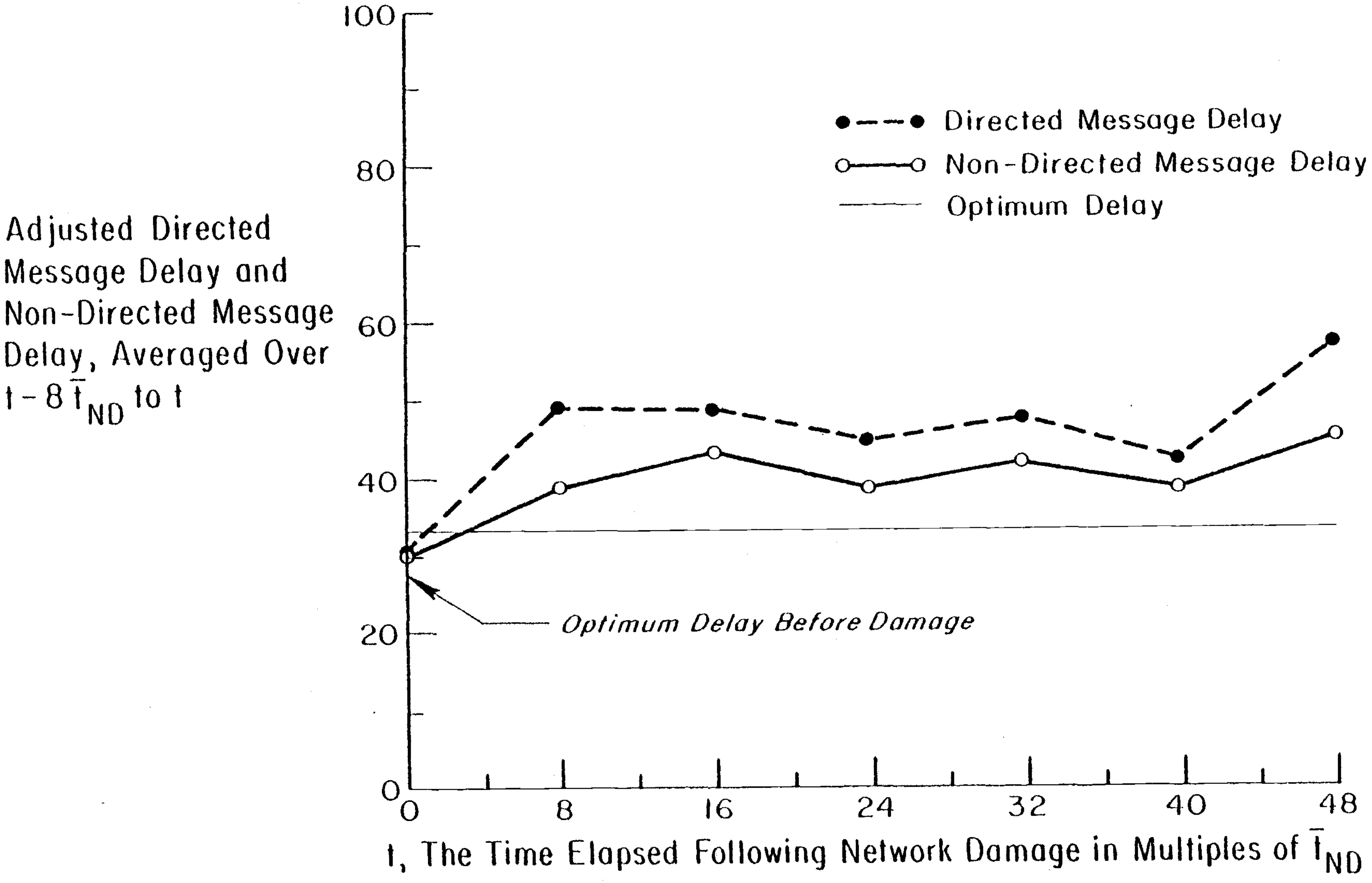


Figure 4. 42. RRPS algorithm, three links jammed.

The adjusted directed message delay shows a gradual rise from its pre-damage value. Unlike the Reciprocal Path Search algorithm, no large initial jump occurs since so few messages are lost. After the routing tables have adapted, and messages are no longer being lost, the directed message delay remains larger than the non-directed message delay. This indicates that some of the directed messages are being routed over suboptimal paths. As was the case with the Reciprocal Path Search algorithm, the Rapid Reciprocal Path Search algorithm completely avoids using the jammed links (except in cases where mesages are flooded), resulting in some messages taking longer than necessary routes.

Six Links Jammed

The results for the six links jammed case are contained in Figures 4.43, 4.44, and 4.45. In contrast to the Simple Backwards Learning and Reciprocal Path Search algorithms, the Rapid Reciprocal Path Search algorithm adapted quickly and completely to this severe case of jamming. Messages were lost for only a period of $8\bar{t}_{ND}$. The algorithm was able to quickly locate the new reciprocal routes, and to determine when no reciprocal path exists between nodes. When no reciprocal path is found, the algorithm simply sends the message out of all the node's ports, i.e., it floods the message out. This flooding is the direct result of the three delay estimates

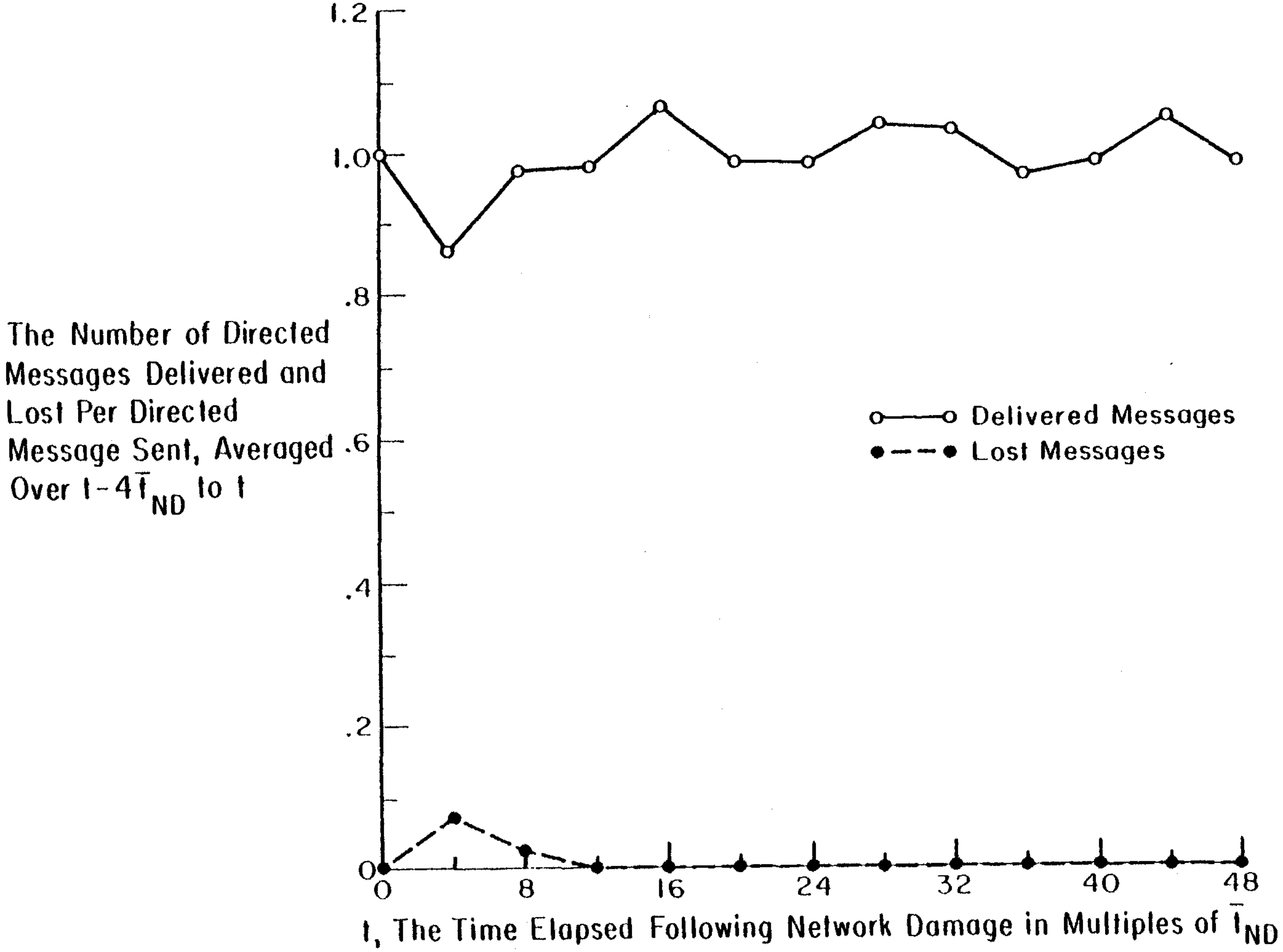


Figure 4. 43. RRPS algorithm, six links jammed.

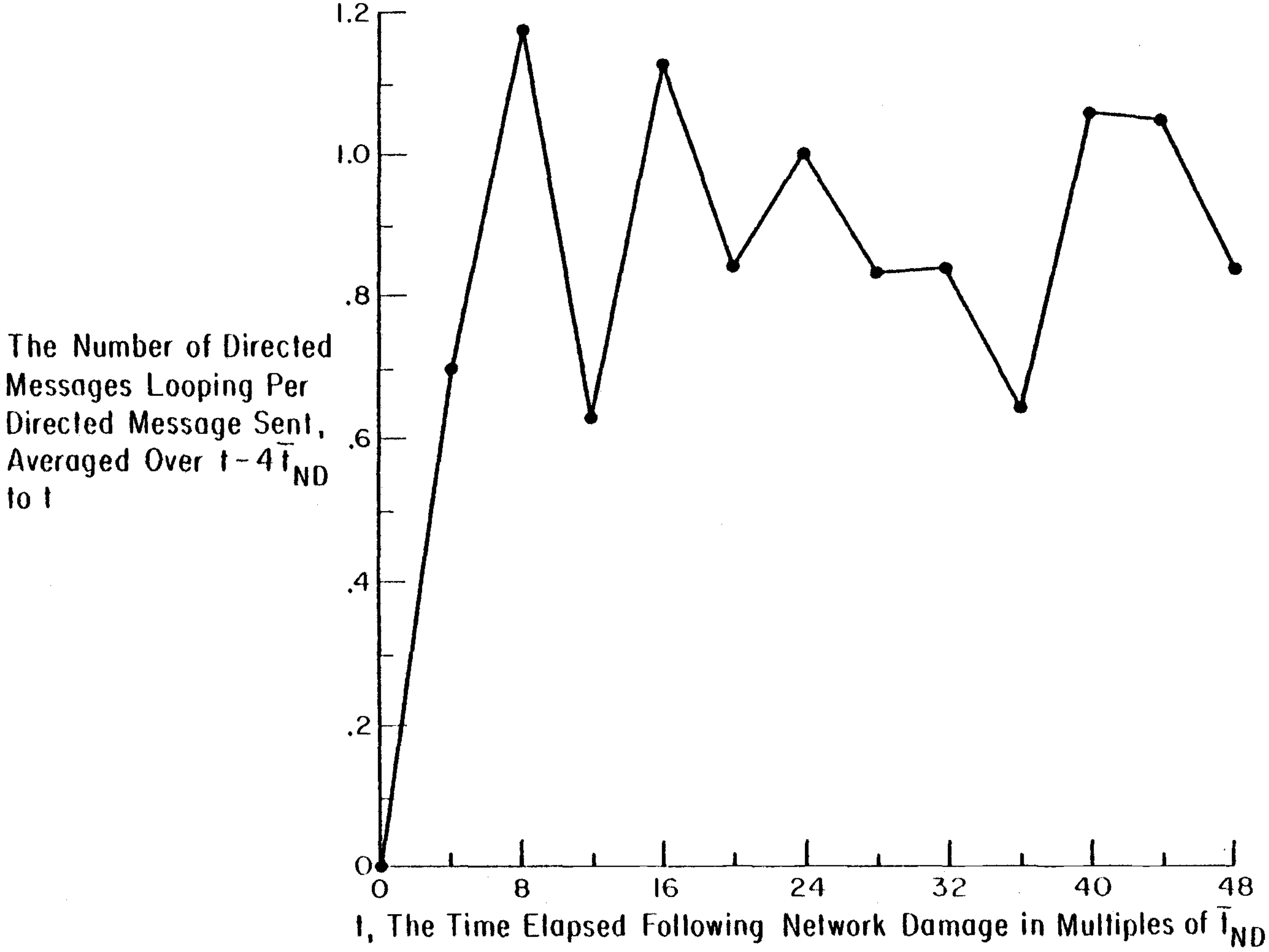


Figure 4. 44. RRPS algorithm, six links jammed.

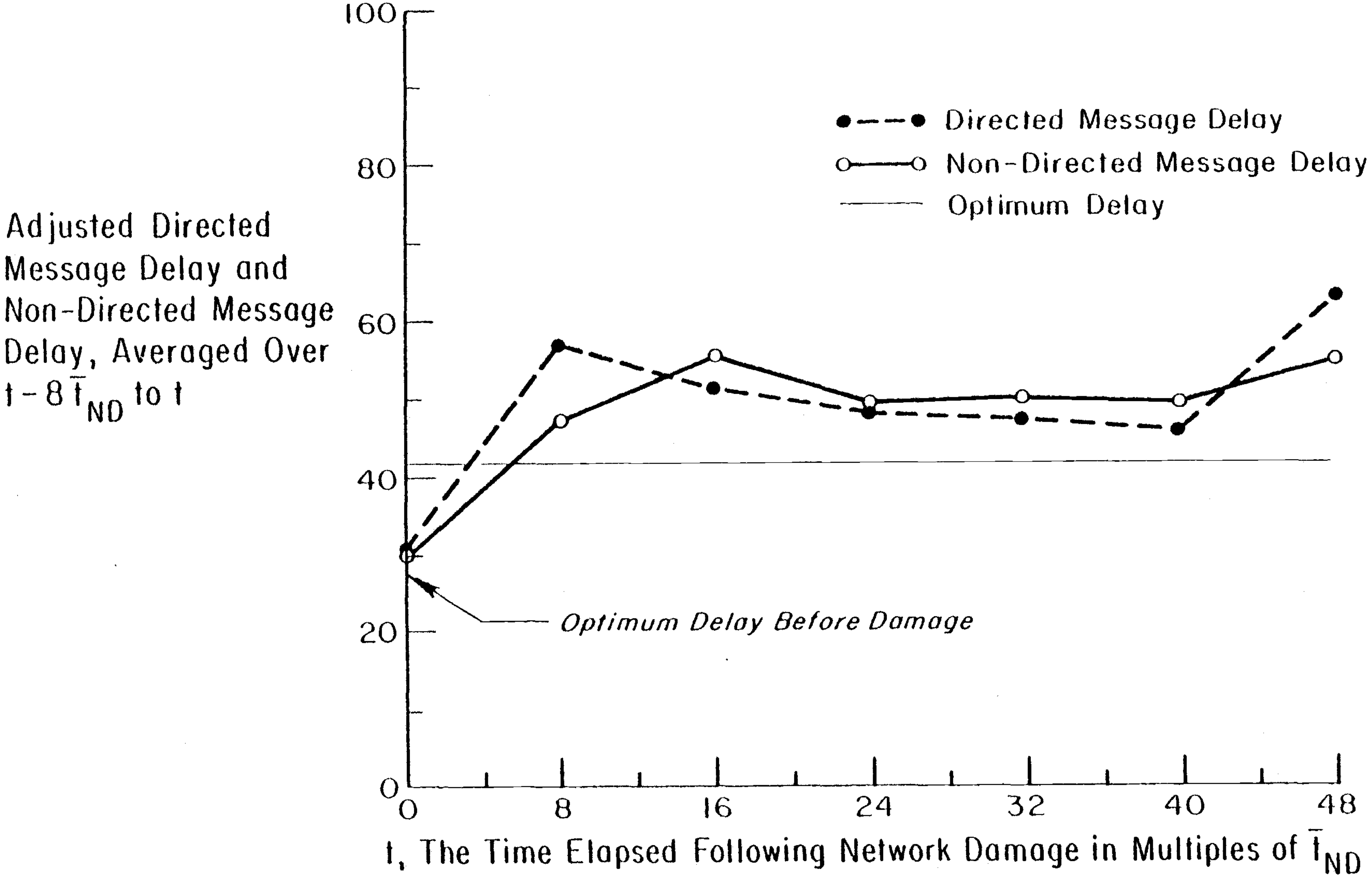


Figure 4.45. RRPS algorithm, six links jammed.

selected by the B table all being equal. The flooding is different from the non-directed message flooding in that the message is sent out of all the node's ports, including the one in which it came. One side effect of this flooding is that a large amount of looping occurs. In fact, unless the directed message is at its source node, sending it out of all three ports results in one copy being returned to the node from which it had just arrived, immediately resulting in ping-ponging, a form of looping. This could be eliminated by adding additional complexity to the routing algorithm and not allowing a directed message to be sent out of its incoming port.

The extra traffic caused by the directed message flooding does not seriously contribute to the network congestion since the misdirected copies very quickly ping-pong, and hence are quickly discarded. Once a copy of the message reaches a node which does have a reciprocal path to the destination, it is no longer flooded, but is routed out of a single port. Thus, the directed message flooding tends to be local in nature.

The adjusted message delay again shows a gradual increase from its pre-damage value. No large jump is seen since there are so few lost messages. Once the messages are no longer being lost, the directed message delay compares very favorably with the non-directed message delay. This is because the flooding of the directed messages tends to route them over the shortest paths, and, significantly, with this severe case of jamming, there are not a lot of alternate routes, so the possibility of their taking a suboptimal path is low.

Reciprocal Path Search Algorithm with Delay Vectors

Each of the previous algorithms was designed to search out and use reciprocal paths, i.e., paths over which messages can travel in both directions. The Reciprocal Path Search Algorithm with Delay Vectors (RPSDV) is designed not only to search out and use the reciprocal paths, but also to search out non-reciprocal paths and use them for routing directed messages. This involves locating non-reciprocal (i.e., jammed) links which can be used for routing directed messages, and disseminating information on these links to all the network nodes so that they can decide if the links are useful to them for routing purposes. In this way, each node will have information on a unique path to a destination node whenever such a path exists, reciprocal or non-reciprocal.

For the purpose of storing information on both the reciprocal and non-reciprocal paths, each node will maintain two tables, the A table and the Delay Vector Table. The A table is the same routing table as that used previously by the Reciprocal Path Search algorithm and stores information on reciprocal paths. It is updated by flooded messages arriving over reciprocal paths. The Delay Vector Table is a new table and stores information on non-reciprocal paths. It is updated by special non-directed messages called "delay vectors." These delay vectors disseminate throughout the network information on particular non-reciprocal links, providing estimates of the delay required to reach destination nodes via the non-reciprocal link, as well as information defining the particular non-reciprocal link in question.

A typical Delay Vector Table is shown in Figure 4.46. It has a column for each destination node, and three rows. The first row's entry, d_{1n}, contains the delay estimate to node n via the non-reciprocal link defined by the d_{2n} and d_{3n} entries. To understand the second and third row entries, consider Figure 4.47 which shows a non-reciprocal link connecting nodes A and B. For reasons which shall become evident later, node B is called the intermediate node and node A is called the creation node. The non-reciprocal link is defined as the link connecting nodes A and B, with the direction of transmission in the direction from the intermediate node, B, toward the creation node, A. Delay Vector Table entry d_{2n} is used to store the identity of the intermediate node, and entry d_{3n} is used to store the identity of the creation node.

d_{1n} = The delay estimate to node n when the directed message is routed via the link connecting the intermediate and creation nodes.

d_{2n} = The intermediate node.

d_{3n} = The creation node.

Figure 4. 46. Delay Vector Table.

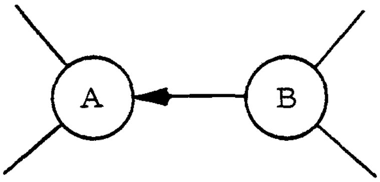


Figure 4. 47. Non-reciprocal link.

Note that the Delay Vector Table only stores a delay estimate to a destination node over a particular non-reciprocal link, along with the identity of the nodes which the link connects. In order to use this information, the node must be able to determine how to reach node B, from whence a message can be sent across the link to node A and on to its destination. The only table which the node can use to route messages to node B is the A table. (This implies that the node must have a reciprocal path to node B. Algorithms could be created eliminating this requirement by creating a third table which would keep track of any non-reciprocal links needed to reach B, along with information on how to use these links to actually reach B.

However, using the third table to keep track of the non-reciprocal links needed to reach B provides for the entry into the algorithm of many choices for route selection, creating the possibility of increasing looping and thus actually degrading the algorithm's performance.) In order to use the Delay Vector Table entries, a node must use the A table to route the message to node B, and from there over the link and ultimately to its destination.

The fact that the A table has an entry for node B, however, does not mean that this entry corresponds to a current valid reciprocal path to B. This was clearly established in the simulation tests of the six links jammed case for the Reciprocal Path Search algorithm. It was observed there that if changes in network structure which eliminate all reciprocal paths between node X and node B occur, node X's A table will still have an entry for node B, but this entry will cease to be updated from then on. This results in the A table entries routing the messages over the old, no longer good, paths. Node X can reasonably evaluate if the A table entries are good only by keeping track of whether or not they are still being updated. For this purpose, node X will keep track of the elapsed time since the A table entry for node B was last updated. If the elapsed time exceeds some set time limit, T_R, then node X will assume that the A table is not currently being updated and therefore no reciprocal path exists to node B. In this event, the A table entries will be considered "dead." If, on the other hand, the elapsed time since the last update is less than T_R, then node X concludes that it does indeed have a reciprocal path to node B, and the corresponding A table entries are considered "alive." To be properly able to determine if the A table entries are truly alive or dead, each node in the network must be required to keep the maximum time elapsed between non-directed message transmissions, $T_{ND_{max}}$, within the limit T_R.

It is also of interest for a node to be able to determine if the Delay Vector Table entry is valid. The fact that the Delay Vector Table contains a delay estimate does not mean that the estimate is valid. If the Delay Vector Table entries have not recently been updated, it is possible that the network structure has changed such that the destination node is no longer reachable via the non-reciprocal link specified by the intermediate and creation node entries. Thus, node X will also keep track of the time elapsed since the last Delay Vector Table update for each destination (column). If this elapsed time exceeds some set time limit, T_{NR}, the node X will conclude that the Delay Vector Table entries for that destination are no longer valid and that a non-reciprocal path no longer exists to it. In this situation, the Delay Vector Table entries for the particular destination shall be considered dead. If on the other hand, the elapsed time is less than T_{NR}, then node X will assume that the non-reciprocal path does exist to the destination and the Delay Vector Table entries shall be considered alive.

Node X can properly determine if the Delay Vector Table entries are alive or dead if the delay vectors are sent out on a regular basis. For this reason, the delay vectors were all sent out at regular intervals of duration T_{DV}, where T_{DV} is less than T_{NR}, during simulations of this algorithm.

The values of $T_{ND_{max}}$, T_R and T_{NR} which were used in the simulation tests are given in Table 4.8. The value of T_{DV} was varied and will be specified for each individual test as it is discussed.

Table 4.8. Values of delay vector parameters used in the transient simulation tests of the new routing protocols.

Parameter	Value
$T_{ND_{max}}$	$4\bar{t}_{ND}$
T_R	$1.5T_{ND_{max}}$
T_{NR}	$1.5T_{DV}$

Delay Vector Creation

As was previously stated, the Delay Vector Tables are updated using special non-directed messages called "delay vectors" which are disseminated throughout the network. These delay vectors are created by one of the nodes connected by the non-reciprocal link.

Consider again the situation depicted in Figure 4.47. Nodes A and B are joined by a non-reciprocal link. Node B should easily be able to detect that it is being jammed, and then inform A of this fact by transmitting a special message directly over the link in its good direction. (The special message could be transmitted over some other longer route if this is required for any reason.) Node A, upon receiving this information from node B, will begin creating and transmitting delay vectors. The purpose of the delay vectors is to inform the entire network of the existence of this non-reciprocal link and to provide estimates of the delays which would result when the messages are routed to destination nodes via the non-reciprocal link. Node A must create the delay vector since it is the only node of the two which has knowledge of which nodes are reachable once the link has been traversed, and thus is the only choice for determining and attaching delay estimates.

The algorithm which node A uses to attach the delay estimates to the delay vector is explained in Figures 4.48 and 4.49.

Node A attempts to attach a delay estimate for each node in the network except for the intermediate node B. Suppose that node A is attempting to determine a delay estimate for node n. It first checks to see if either its A table or Delay Vector Table entries for

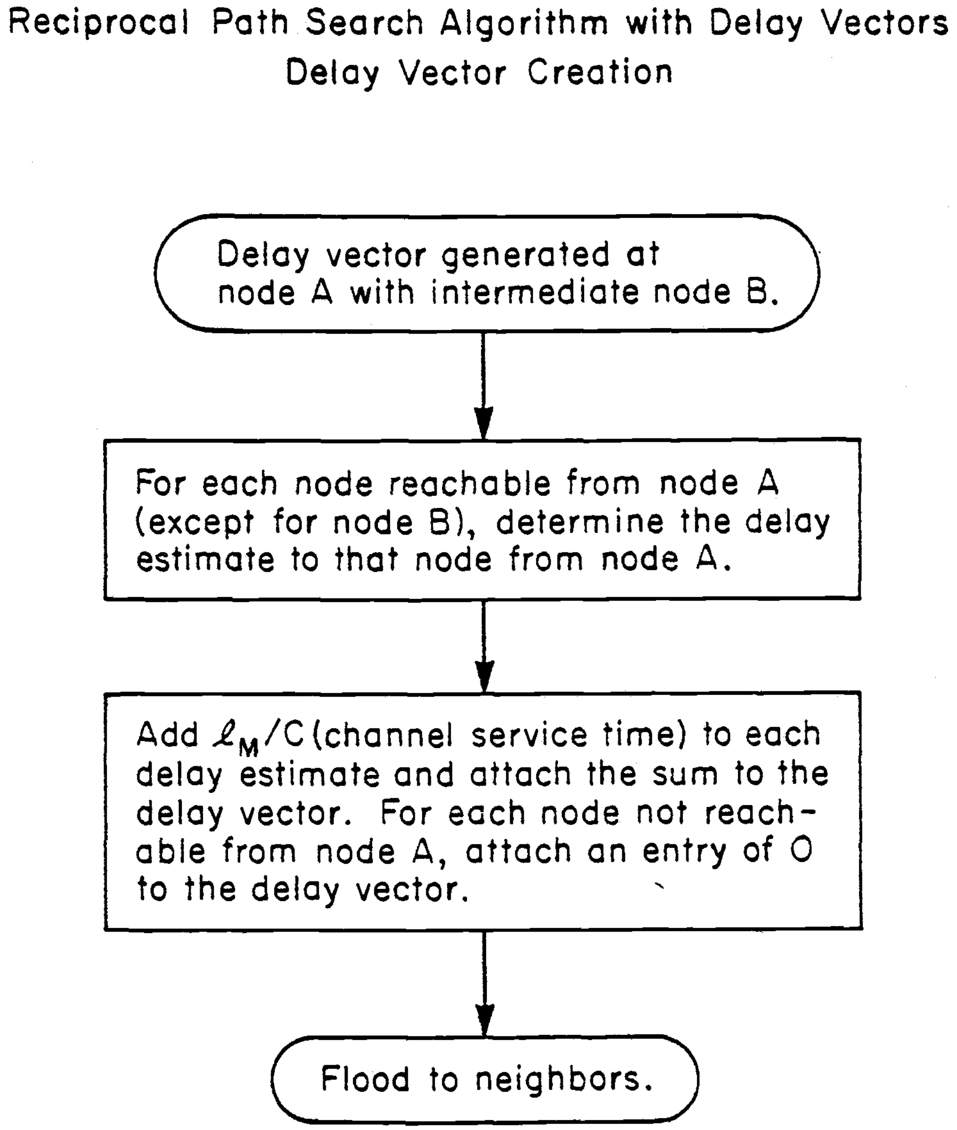

Reciprocal Path Search Algorithm with Delay Vectors
Delay Vector Creation
Delay vector generated at node A with intermediate node B.
For each node reachable from node A (except for node B), determine the delay estimate to that node from node A.
Add ℓ_M/C (channel service time) to each delay estimate and attach the sum to the delay vector. For each node not reachable from node A, attach an entry of 0 to the delay vector.
Flood to neighbors.

Figure 4.48. Conceptual block diagram.

Reciprocal Path Search Algorithm with Delay Vectors
Delay Vector Creation

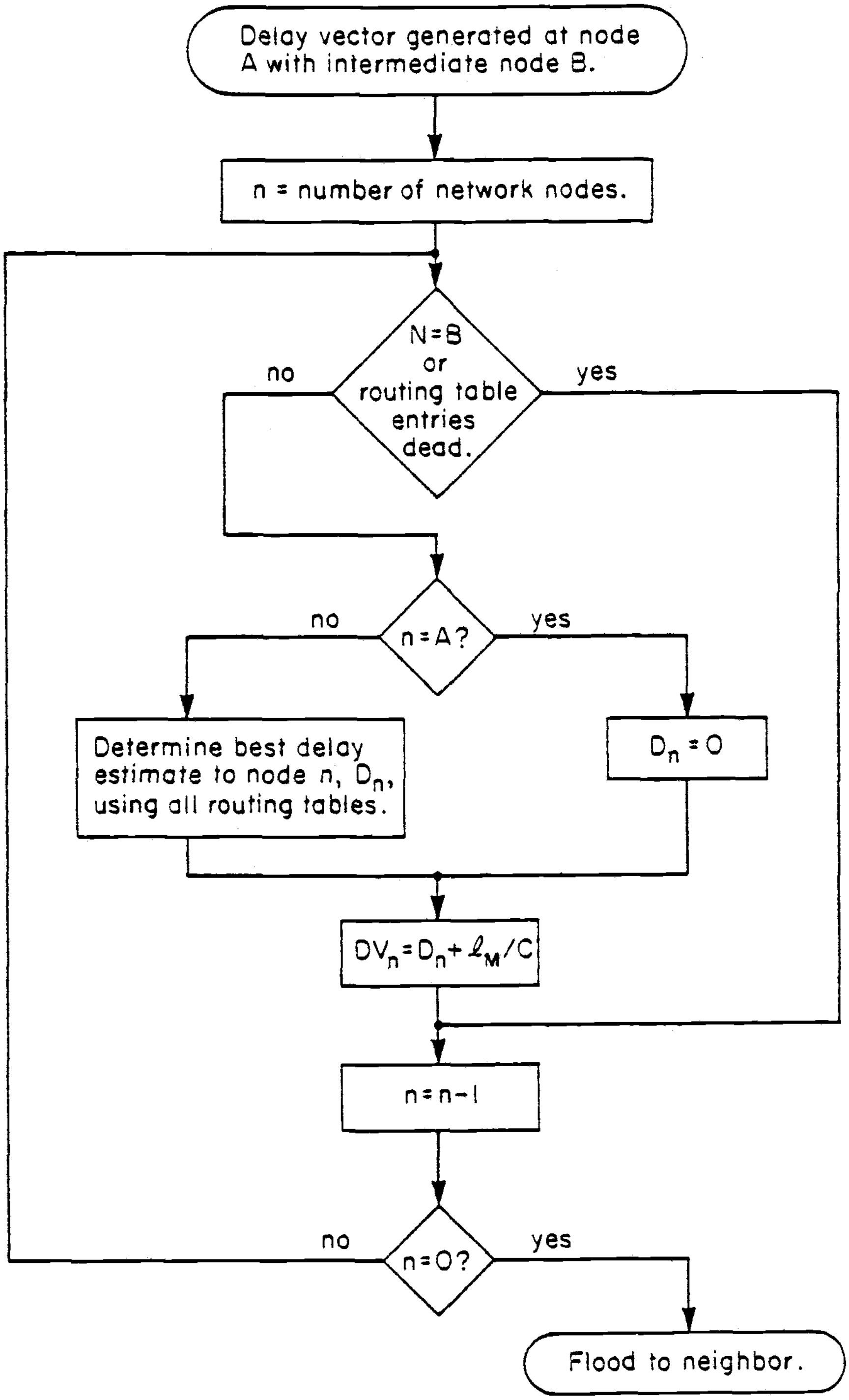

Delay vector generated at node A with intermediate node B.
n = number of network nodes.
N=B or routing table entries dead.
no
yes
n=A?
no
yes
Determine best delay estimate to node n, D_n, using all routing tables.
$D_n = 0$
$DV_n = D_n + \ell_M/C$
n=n-1
n=0?
no
yes
Flood to neighbor.

Figure 4.49. Detailed block diagram.

node n are alive. If they are not alive, node A concludes that it has no path to node n, and merely enters a zero into the delay vector to indicate this. (If the delay vector has a predefined location for each delay estimate, then something must be entered into that location to signal that no estimate is available. Using predefined locations will allow for a shorter delay vector than would be required if each delay estimate needs to be accompanied with the identity of the node which the delay estimate is for.) If, on the other hand, either or both of the A table and Delay Vector Table entries are alive, node A will use these tables to determine the best delay estimate to node n, add ℓ_M/C to it, and attach the updated estimate to the delay vector. Adding ℓ_M/C to the delay estimate determined at node A produces an estimate of the delay to the destination node from node B.

The fact that node A can draw its delay estimates from its Delay Vector Table means that the non-reciprocal link connecting B with A can be used in estimating delays over paths including other non-reciprocal links. This enables any destination node to be reached without flooding even if the only path to it contains several non-reciprocal links. After node A has either attached a delay estimate or a zero for each destination node, it also attaches its own address and the address of the intermediate node B, and floods the delay vector to all of its neighbors. This delay vector will continue to be flooded by each node receiving it, thus disseminating its information throughout the network.

Delay Vector Table Update Routine

The nodes retransmit delay vectors just as they do any other non-directed message, but they also use them to update their Delay Vector Tables.

Figures 4.50 and 4.51 describe how the nodes update their Delay Vector Tables. Suppose a delay vector with creation node A and intermediate node B arrives at node X. Node X first checks to see if there is a reciprocal path from node X to node B. If the reciprocal path does not exist, i.e., the A table entries for node B are all dead, then node X does not update the Delay Vector Table and merely returns the delay vector to the node's non-directed message handling routine for further flooding. However, if a reciprocal path does exist to node B, node X then consults its A table to determine D_B, the delay estimate from node X to node B. Knowing that a path exists to node B and knowing the delay to node B, node X can use the delay vector to update its Delay Vector Table.

For column n in the Delay Vector Table, node X checks to see if $d_{2n} = B$ and $d_{3n} = A$. If so, the Delay Vector Table entry for destination node n was last updated by the delay vector for the

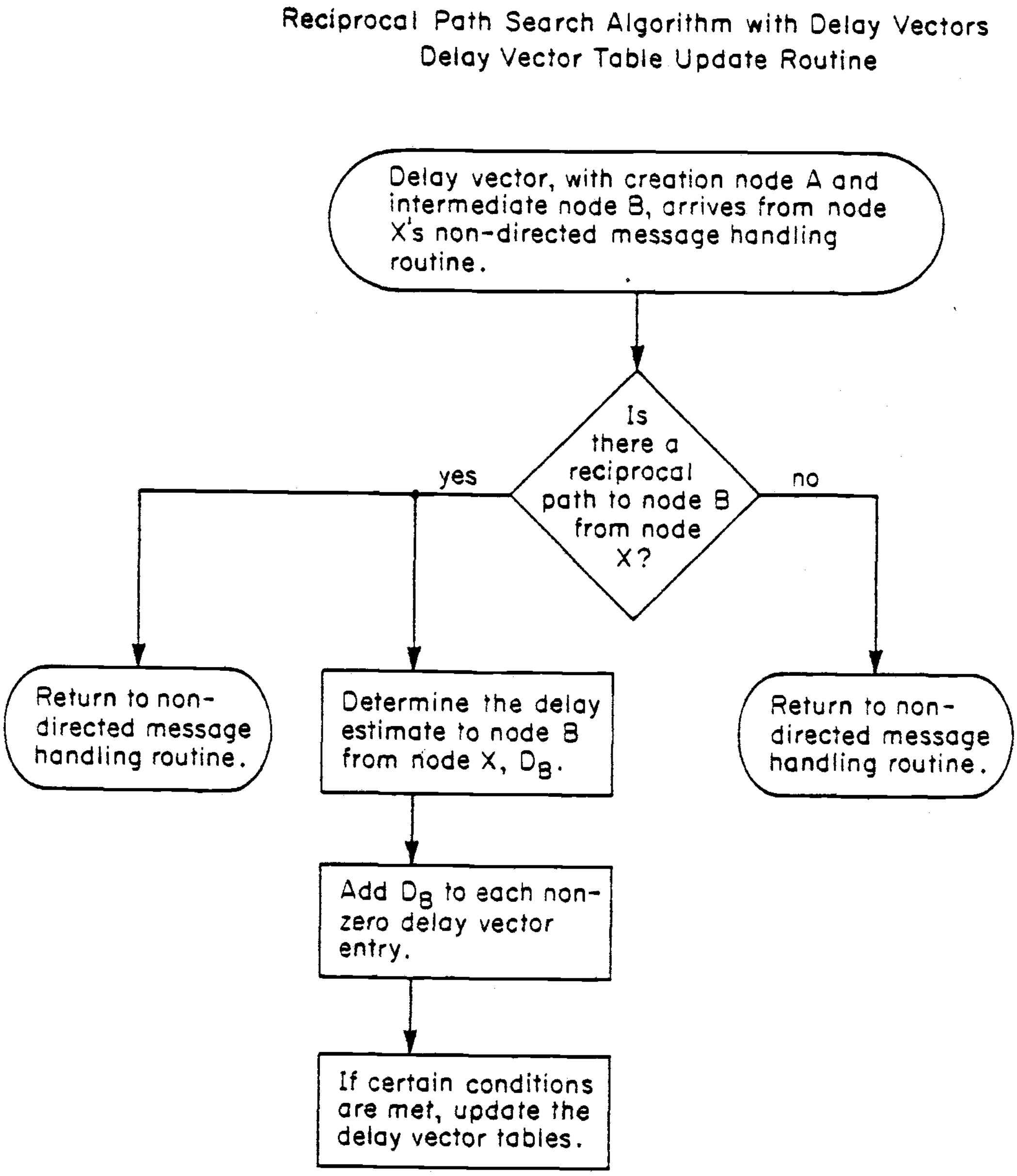

Reciprocal Path Search Algorithm with Delay Vectors
Delay Vector Table Update Routine
Delay vector, with creation node A and intermediate node B, arrives from node X's non-directed message handling routine.
Is there a reciprocal path to node B from node X?
yes
no
Return to non-directed message handling routine.
Determine the delay estimate to node B from node X, D_B.
Return to non-directed message handling routine.
Add D_B to each non-zero delay vector entry.
If certain conditions are met, update the delay vector tables.

Figure 4.50. Conceptual block diagram.

Reciprocal Path Search Algorithm with Delay Vectors
Delay Vector Table Update Routine

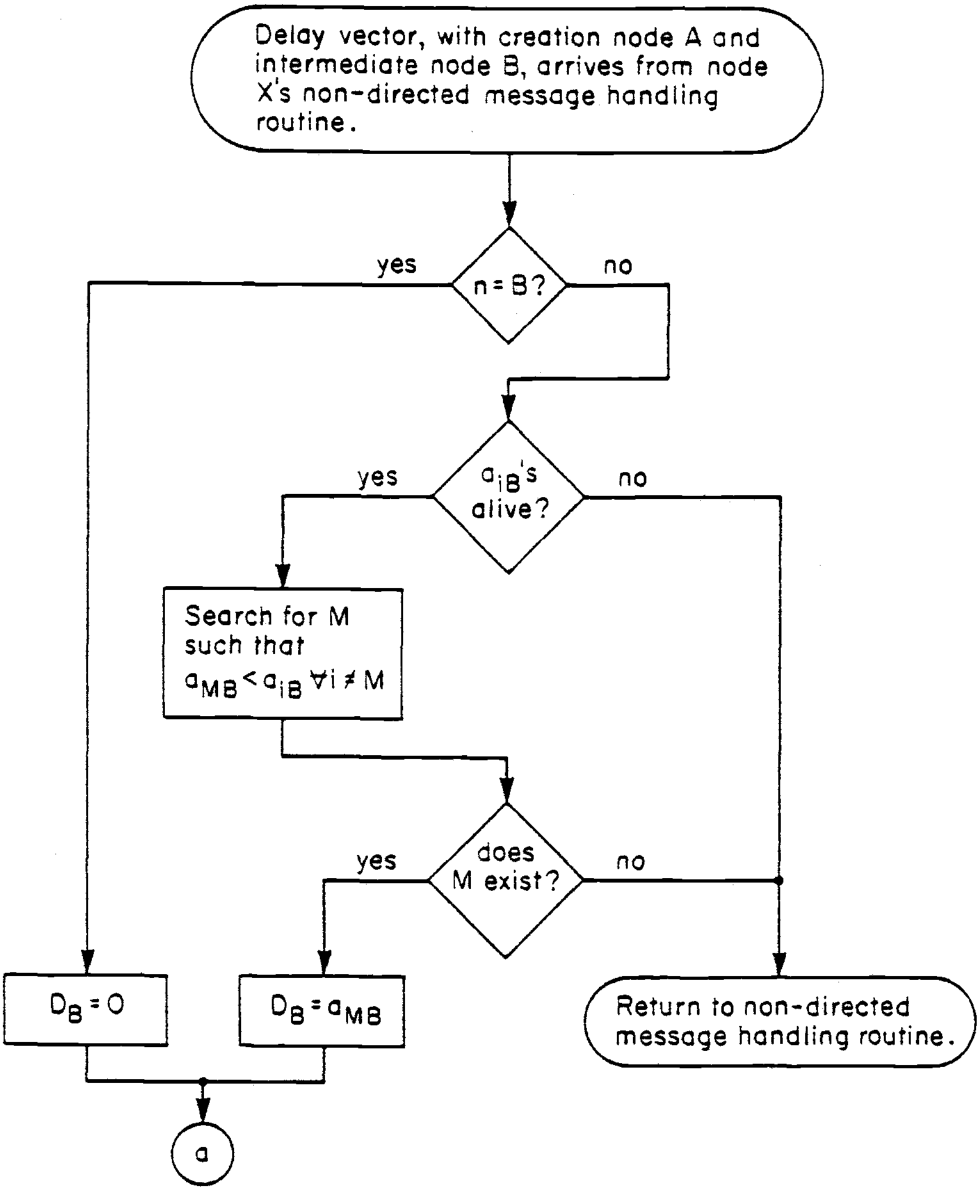

Delay vector, with creation node A and intermediate node B, arrives from node X's non-directed message handling routine.
n = B?
yes
no
a_{iB}'s alive?
yes
no
Search for M such that $a_{MB} < a_{iB}\ \forall i \neq M$
does M exist?
yes
no
$D_B = 0$
$D_B = a_{MB}$
Return to non-directed message handling routine.
a

Figure 4. 51. Detailed block diagram.

Reciprocal Path Search Algorithm with Delay Vectors
Delay Vector Table Update Routine (cont.)

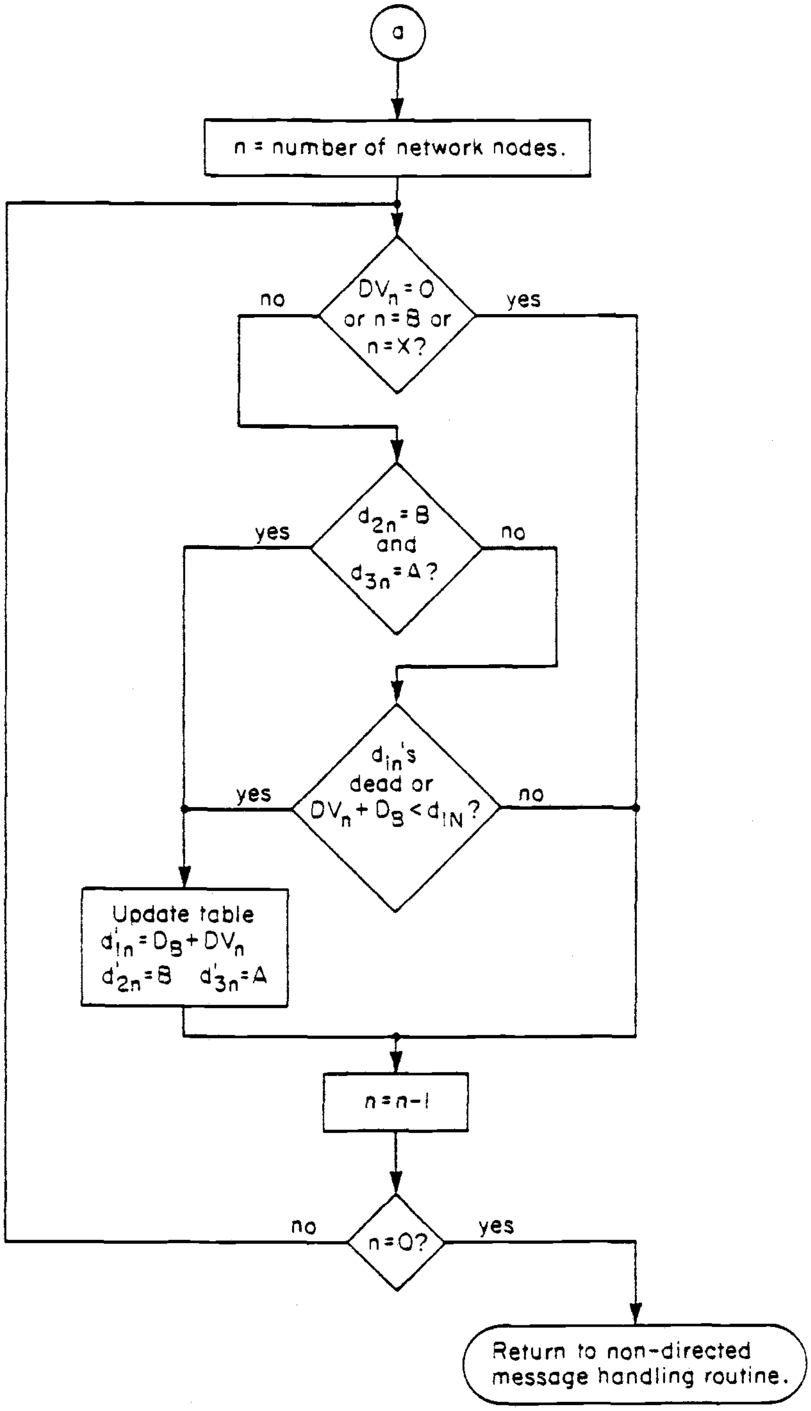

a
n = number of network nodes.
no
$DV_n = 0$ or $n = B$ or $n = X$?
yes
yes
$d_{2n} = B$ and $d_{3n} = A$?
no
yes
d_{in}'s dead or $DV_n + D_B < d_{IN}$?
no
Update table
$d'_{1n} = D_B + DV_n$
$d'_{2n} = B$ $d'_{3n} = A$
$n = n - 1$
no
$n = 0$?
yes
Return to non-directed message handling routine.

Figure 4.51. Continued.

same non-reciprocal link, and thus it is again updated using the formula

$$d'_{1n} = D_B + DV_n \tag{4.14}$$

where

d'_{1n} = the new Delay Vector Table estimate to node n.

D_B = the delay estimate to node B from node X.

DV_n = the delay vector entry for node n.

The entries for d_{2n} and d_{3n} are of course left unchanged. If $d_{2n} \neq B$ or $d_{3n} \neq A$, meaning the Delay Vector Table entry was last updated by a delay vector for a different non-reciprocal link, node X checks further to determine if the d_{in}'s are dead or if $(DV_n + D_B)$ is less than d_{1n}. If either of these conditions are met, the new delay vector entries are an improvement over the existing entries, and the Delay Vector Table is updated using Equation (4. 14) and

$$d'_{2n} = A \tag{4.15}$$

$$d'_{3n} = B \tag{4.16}$$

After node X has completed this procedure for each Delay Vector Table column, the delay vector is returned, in its original form, to the node's regular non-directed message handling routine for further flooding to other nodes. (An alternate approach would be to immediately upon reception flood the delay vector to other nodes, using a copy of it to update the Delay Vector Table.)

Non-Directed Message Handling

The non-directed message handling procedure is described in Figures 4.52 and 4.53. This procedure is exactly the same as in the Reciprocal Path Search algorithm except the routine now checks to see if the non-directed message is a delay vector, and if so, sends it to the Delay Vector Table update routine. Note that these figures assume that the delay vectors are also used to update the A table just as is done with any other non-directed message. This involves only the requirement that the delay vector contain information allowing an estimate of the delay elapsed since it left its source node to be computed.

Directed Message Handling

The directed message handling procedure explained in Figures 4.54 and 4.55, is very different from the procedures used by the previous routing algorithms. Suppose a message arrives at node X directed for node D. Node X first checks to see if it is the first copy received. If it is the first copy, node X then checks to see if either the A table or Delay Vector Table entries are alive. If neither

Reciprocal Path Search Algorithm with Delay Vectors
Non-Directed Message Handling

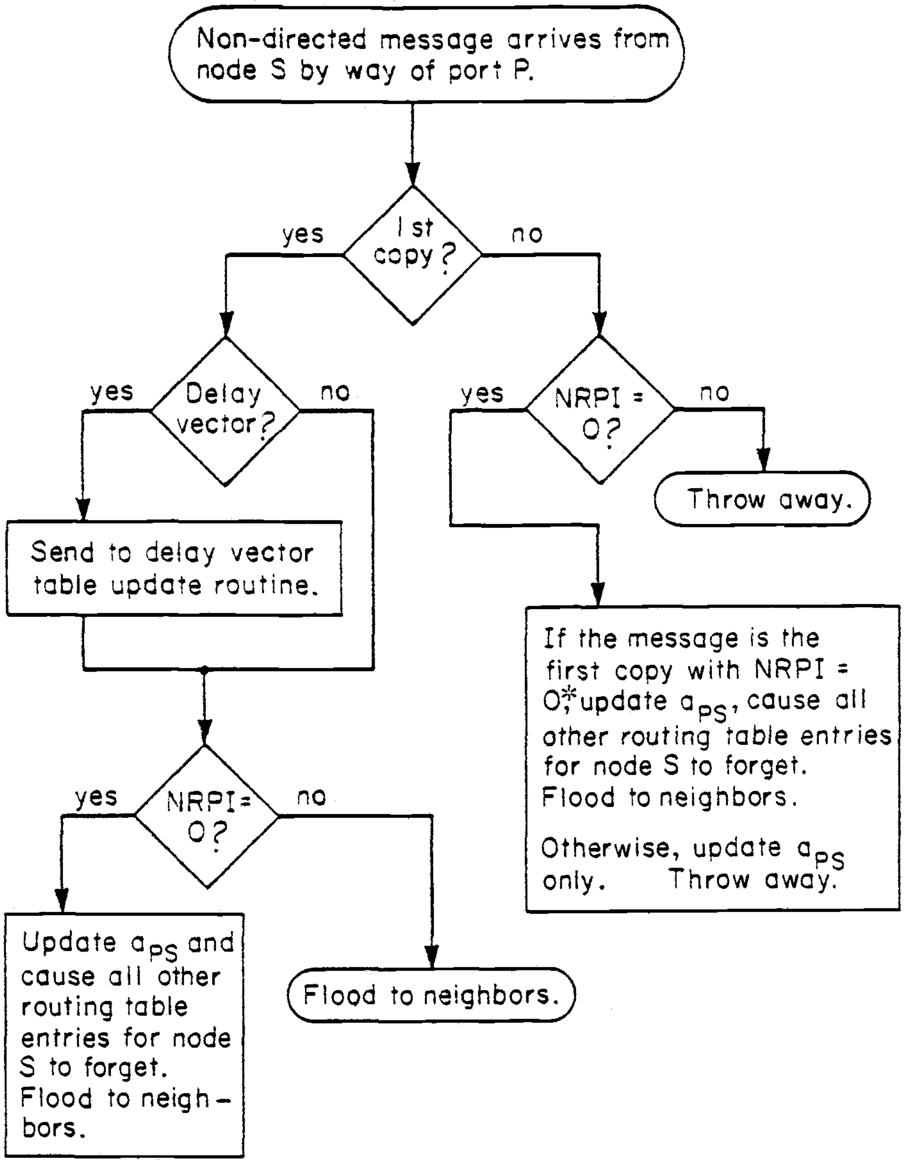

Non-directed message arrives from node S by way of port P.
1st copy?
yes
no
Delay vector?
yes
no
NRPI = O?
yes
no
Throw away.
Send to delay vector table update routine.
If the message is the first copy with NRPI = O,* update a_{PS}, cause all other routing table entries for node S to forget. Flood to neighbors.
Otherwise, update a_{PS} only. Throw away.
NRPI= O?
yes
no
Update a_{PS} and cause all other routing table entries for node S to forget. Flood to neighbors.
Flood to neighbors.

Figure 4.52. Conceptual block diagram.

[*] All previous copies had NRPI = 1.

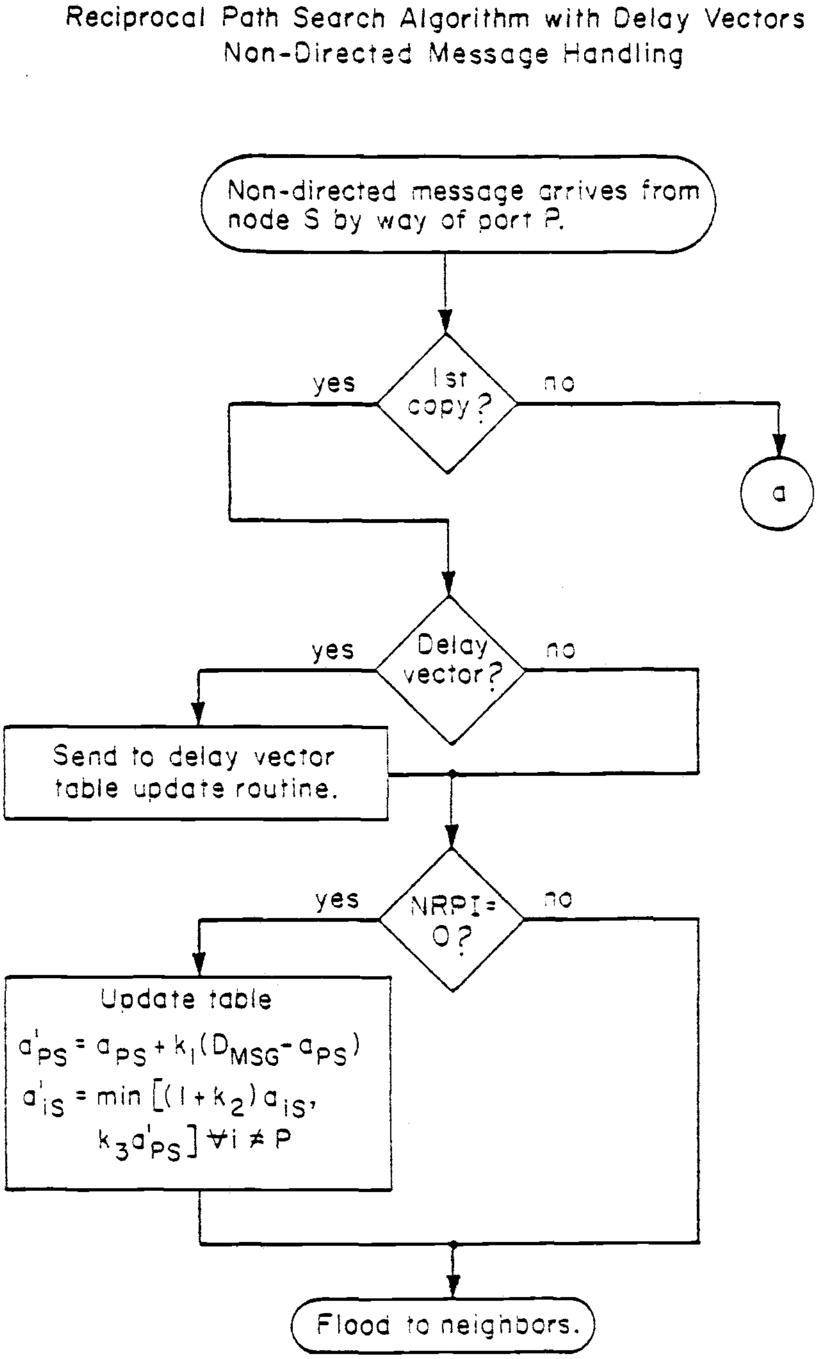

Reciprocal Path Search Algorithm with Delay Vectors
Non-Directed Message Handling
Non-directed message arrives from node S by way of port P.
1st copy?
yes
no
a
Delay vector?
yes
no
Send to delay vector table update routine.
NRPI= 0?
yes
no
Update table
a'PS = aPS + k1(DMSG - aPS)
a'iS = min [(1+k2)aiS, k3a'PS] ∀i ≠ P
Flood to neighbors.

Figure 4. 53. Detailed block diagram.

Reciprocal Path Search Algorithms with Delay Vectors
Non-Directed Message Handling (cont.)

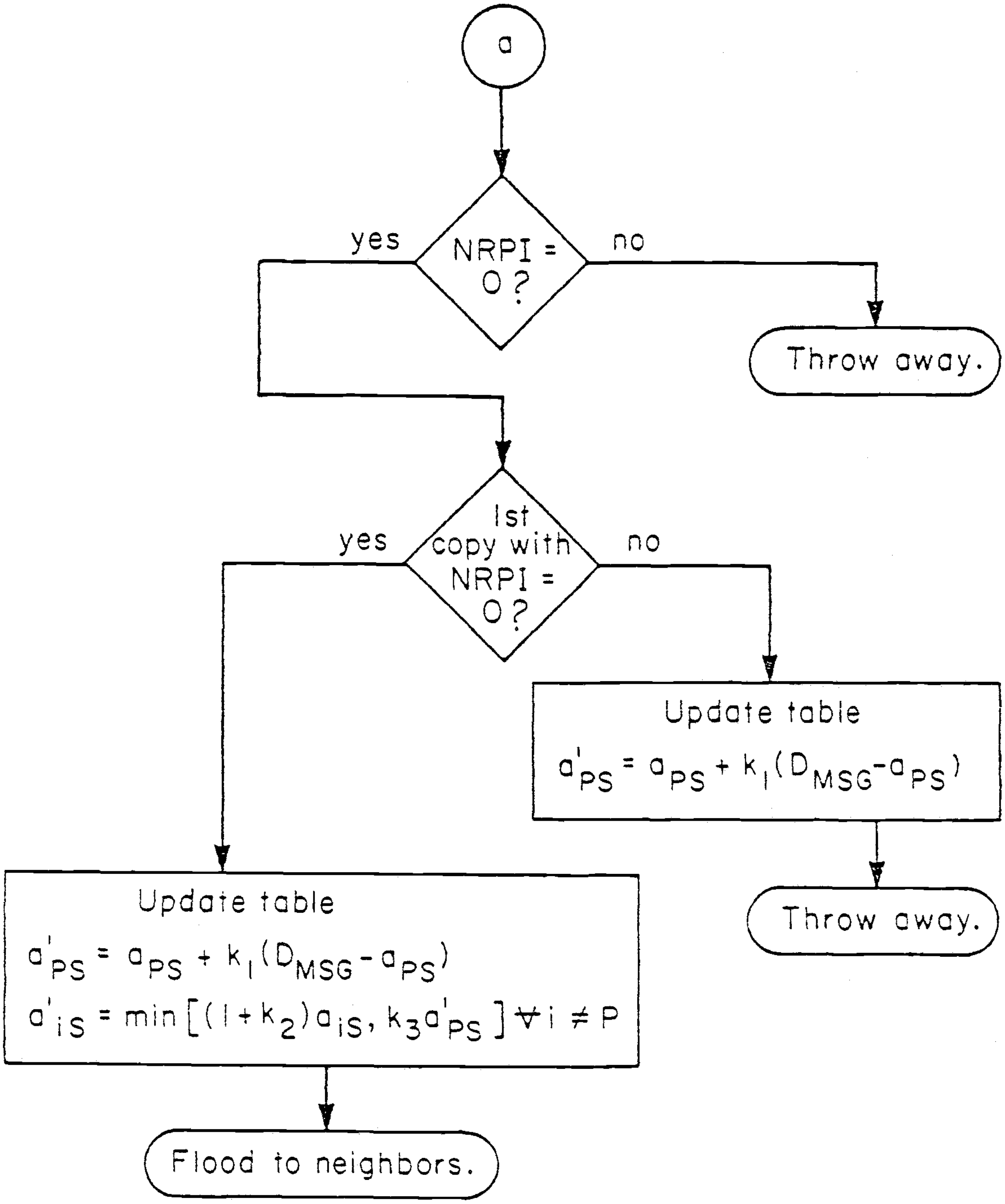

a
NRPI = 0?
yes
no
Throw away.
1st copy with NRPI = 0?
yes
no
Update table
$a'_{PS} = a_{PS} + k_1(D_{MSG} - a_{PS})$
Throw away.
Update table
$a'_{PS} = a_{PS} + k_1(D_{MSG} - a_{PS})$
$a'_{iS} = \min[(1+k_2)a_{iS}, k_3 a'_{PS}] \forall i \neq P$
Flood to neighbors.

Figure 4.53. Continued.

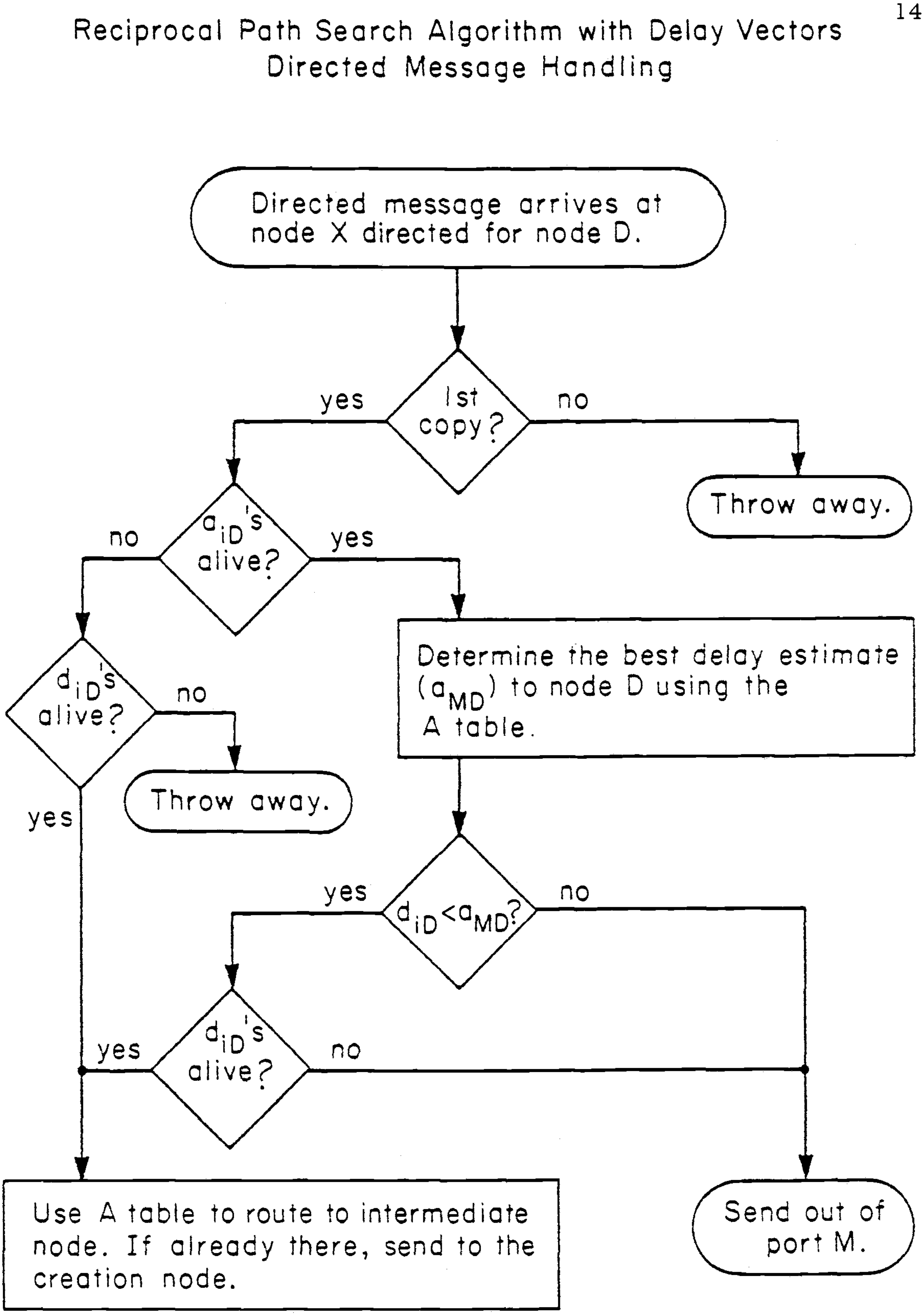

Reciprocal Path Search Algorithm with Delay Vectors
Directed Message Handling
Directed message arrives at node X directed for node D.
1st copy?
yes
no
Throw away.
a_{iD}'s alive?
no
yes
Determine the best delay estimate (a_{MD}) to node D using the A table.
d_{iD}'s alive?
no
Throw away.
yes
$d_{iD}<a_{MD}$?
yes
no
d_{iD}'s alive?
yes
no
Use A table to route to intermediate node. If already there, send to the creation node.
Send out of port M.

Figure 4. 54. Conceptual block diagram.

Reciprocal Path Search Algorithm with Delay Vectors
Directed Message Handling

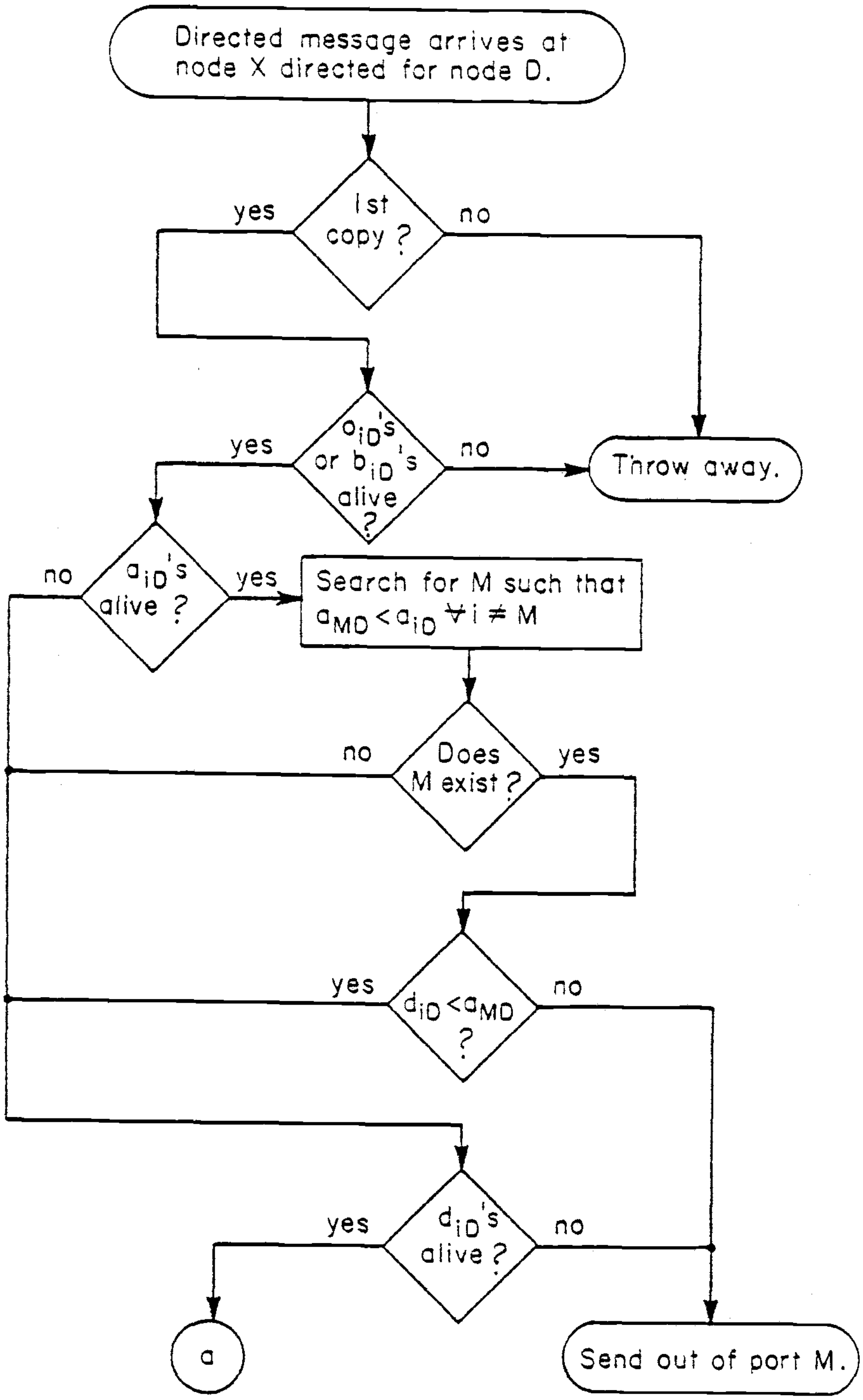

Directed message arrives at node X directed for node D.
1st copy ?
yes
no
a_{iD}'s or b_{iD}'s alive ?
yes
no
Throw away.
no
a_{iD}'s alive ?
yes
Search for M such that $a_{MD} < a_{iD} \forall i \neq M$
no
Does M exist ?
yes
yes
$d_{iD} < a_{MD}$?
no
yes
d_{iD}'s alive ?
no
a
Send out of port M.

Figure 4. 55. Detailed block diagram.

Reciprocal Path Search Algorithm with Delay Vectors
Directed Message Handling (cont.)

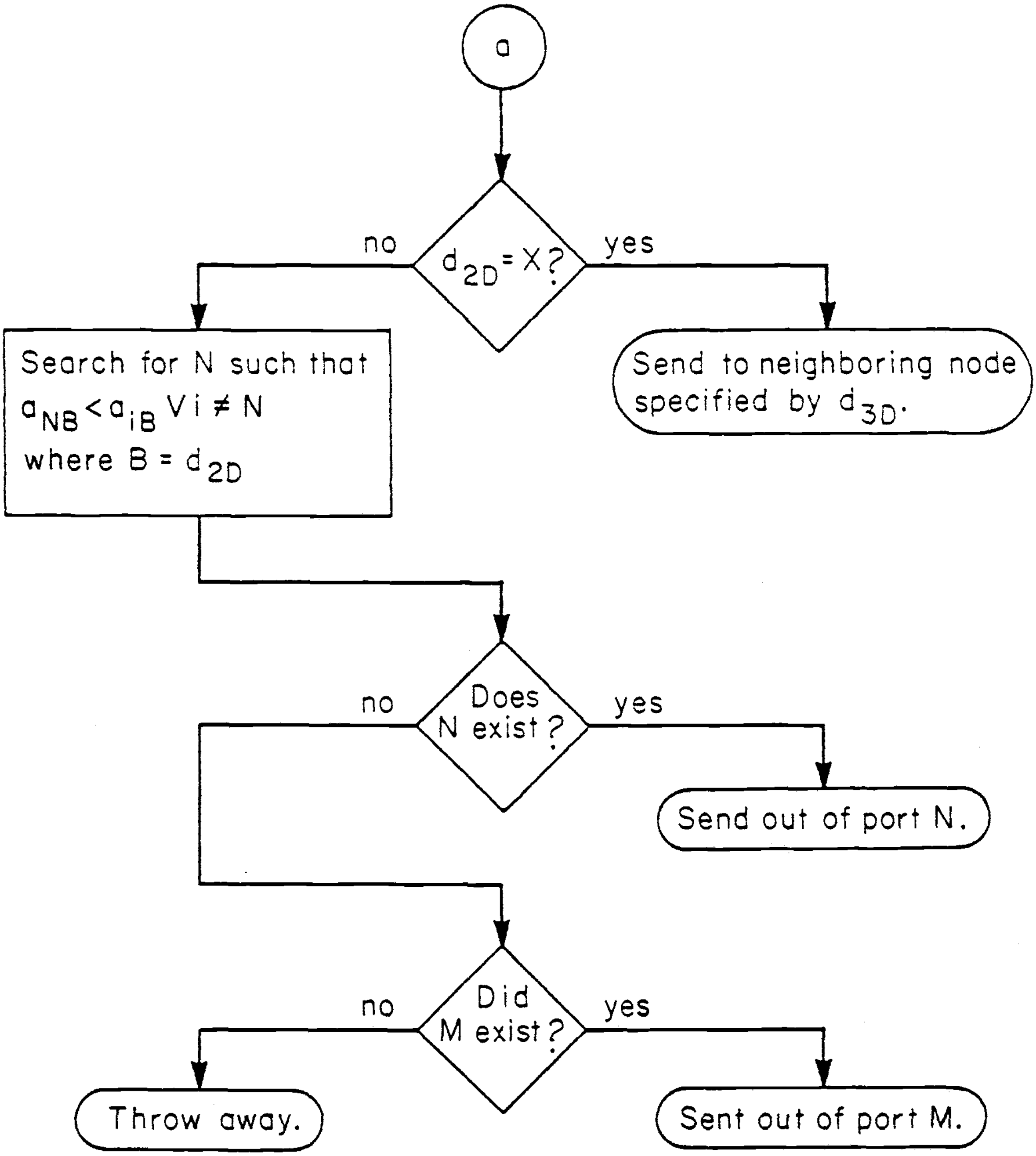

a
no
$d_{2D} = X$?
yes
Search for N such that $a_{NB} < a_{iB} \forall i \neq N$ where $B = d_{2D}$
Send to neighboring node specified by d_{3D}.
no
Does N exist?
yes
Send out of port N.
no
Did M exist?
yes
Throw away.
Sent out of port M.

Figure 4.55. Continued.

table has live entried for node D, node X decides that it does not have a path to node D and discards the message. This is a valid conclusion since in steady state the lack of either live Delay Vector Table or live A table entries for a particular destination does in fact mean that there is no path to that destination, reciprocal or non-reciprocal. Thus the use of delay vectors allows a node to sense when a path does not exist to a destination, and avoid congesting the network with transmissions of directed messages which have no hope of being delivered. However, shortly following network damage, a node may occasionally conclude (due to dead table entries) that no path exists to a destination when in fact one does exist, but the information on that path has not yet propagated throughout the network. This will result in that message not being sent out so it will be a lost message.

If there are live entries for node D in either or both the A table and Delay Vector Table, node X then uses these tables to route the message toward its destination.

The first step in routing the message is to check the A table to see if it has a live entry for node D. If it does, node X determines the best delay estimate to node D contained in the A table. If this estimate, a_{MD}, is less than d_{1D}, or d_{1D} is dead, the message is sent out of port M, along a reciprocal path to node D. If, however, d_{1D} is alive and less than a_{MD}, then node X uses the A table to send the message to the non-reciprocal link defined by d_{2n} and d_{3n}, since the route via this composite path represents a shorter delay path to the destination. Node X uses the A table to route the message to the intermediate node specified by d_{2n}. When the message reaches node d_{2n} (which could be node X) then this node routes the message across the non-reciprocal link directly to the creation node, d_{3n} (assuming no further changes in routing tables occur before the message reaches d_{2n}). This portion of the algorithm assumes that each node knows the identity of its immediate neighbors. This is the only knowledge of the network structure that the algorithm requires.

If the A table entries are dead and the Delay Vector Table entries are alive, node X turns immediately to the Delay Vector Table and uses it to route the message as described above.

Three Links Destroyed

The results for the three links destroyed case are shown in Figures 4.56, 4.57, and 4.58. Again as expected, these results are essentially the same as those obtained with the Simple Backwards Learning and Reciprocal Path Search algorithms. This algorithm only differs from these two algorithms when jamming occurs. For the three links destroyed case, delay vectors were not even sent out

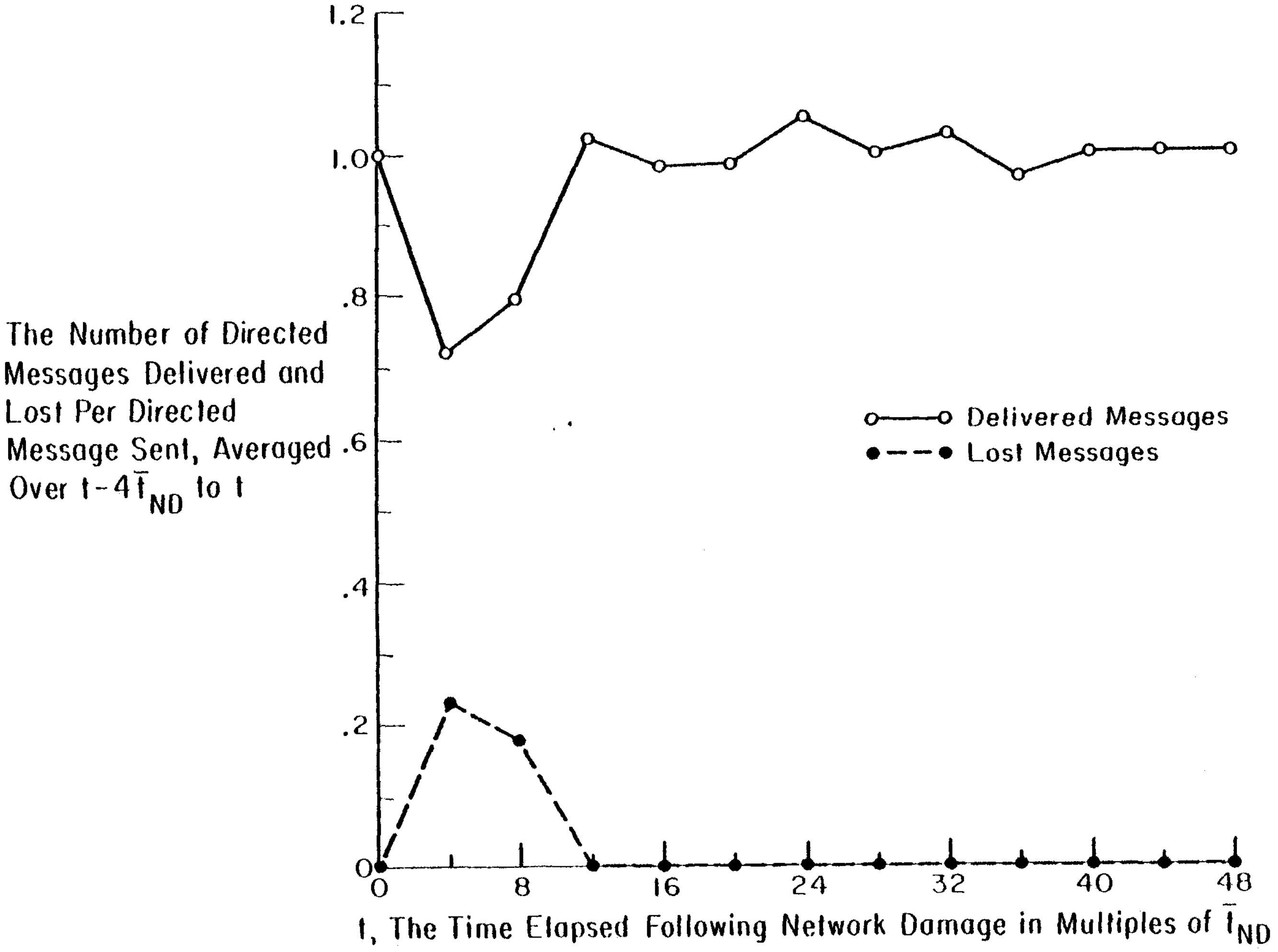


Figure 4.56. RPSDV algorithm, three links destroyed.

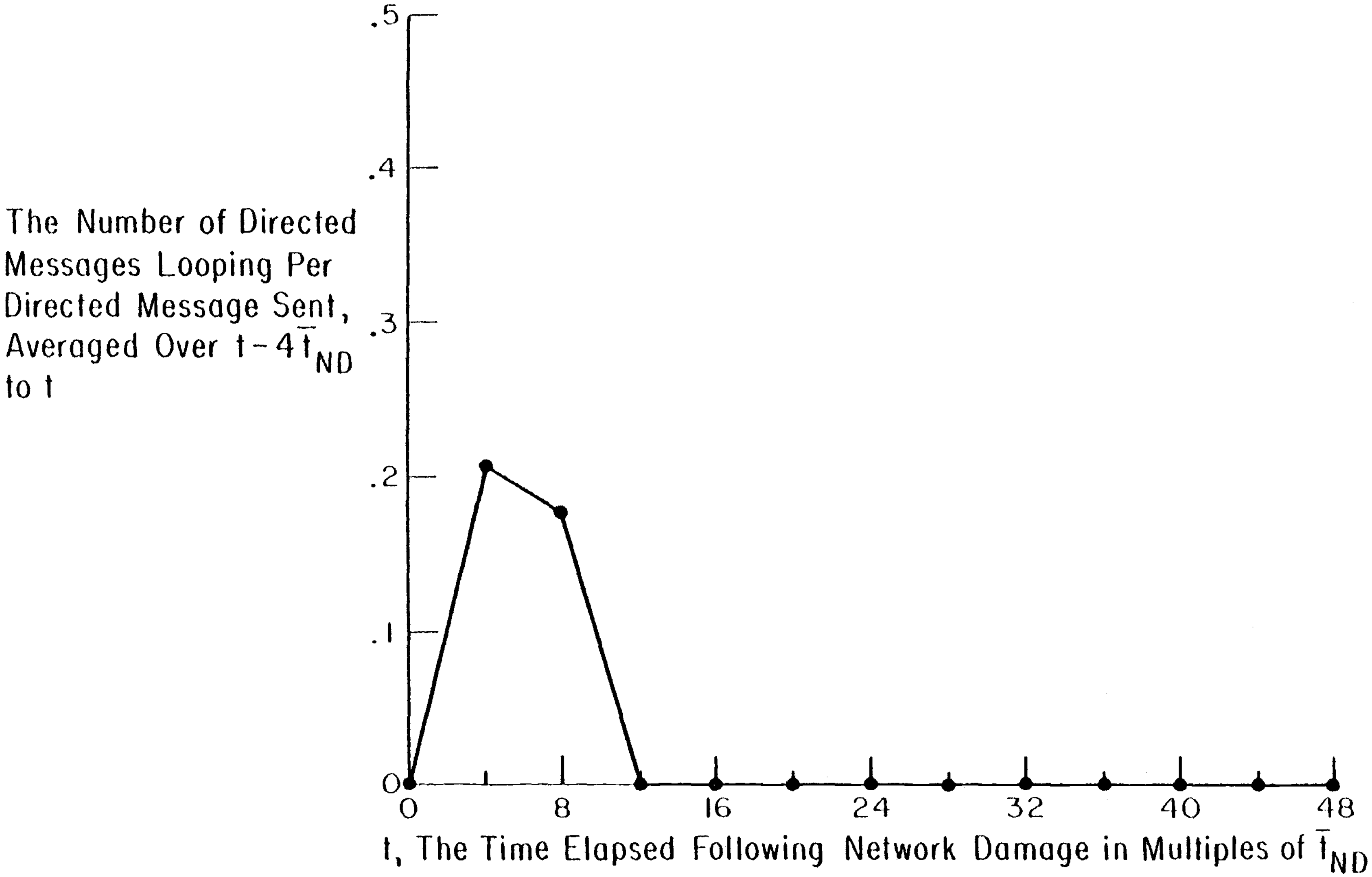


Figure 4.57. RPSDV algorithm, three links destroyed.

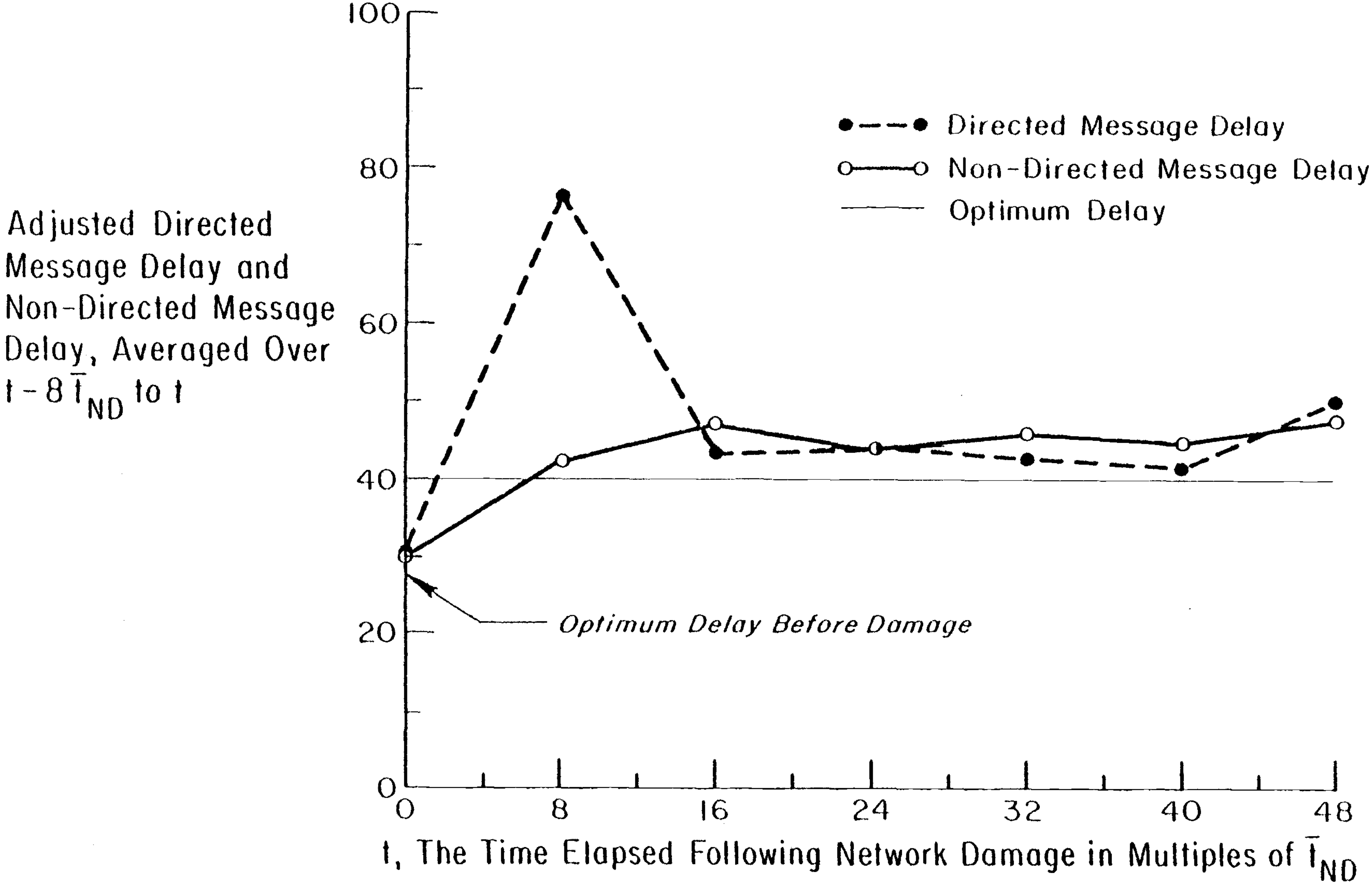


Figure 4. 58. RPSDV algorithm, three links destroyed.

since they are only used when there are non-reciprocal links present in the network.

The results show a temporary period during which messages are lost and looping. After this period, the algorithm completely adapts, and no further messages are lost. The adjusted directed message delay again has an initial period when delays are large due to the penalty imposed for lost messages. Once the algorithm has adapted and messages are no longer being lost, the directed message delay is essentially the same as the non-directed message delay, indicating that the directed messages are being routed over shortest paths.

Three Links Jammed

This case of network damage was tested for values of $T_{DV} = 4\bar{t}_{ND}$ and $T_{DV} = 2\bar{t}_{ND}$.

The results for the $T_{DV} = 4\bar{t}_{ND}$ case are given in Figures 4.59, 4.60, and 4.61. These results again show a temporary period when messages are lost and looping. This period lasts for about $12\bar{t}_{ND}$. When compared to the Reciprocal Path Search algorithm (without delay vectors), it is clear that the use of delay vectors results in an increase in lost and looping messages. This is primarily the result of early bad Delay Vector Table entries. When the network is initially damaged, the delay vectors are constructed using the entries in the A table. These early A table entries are incorrect,

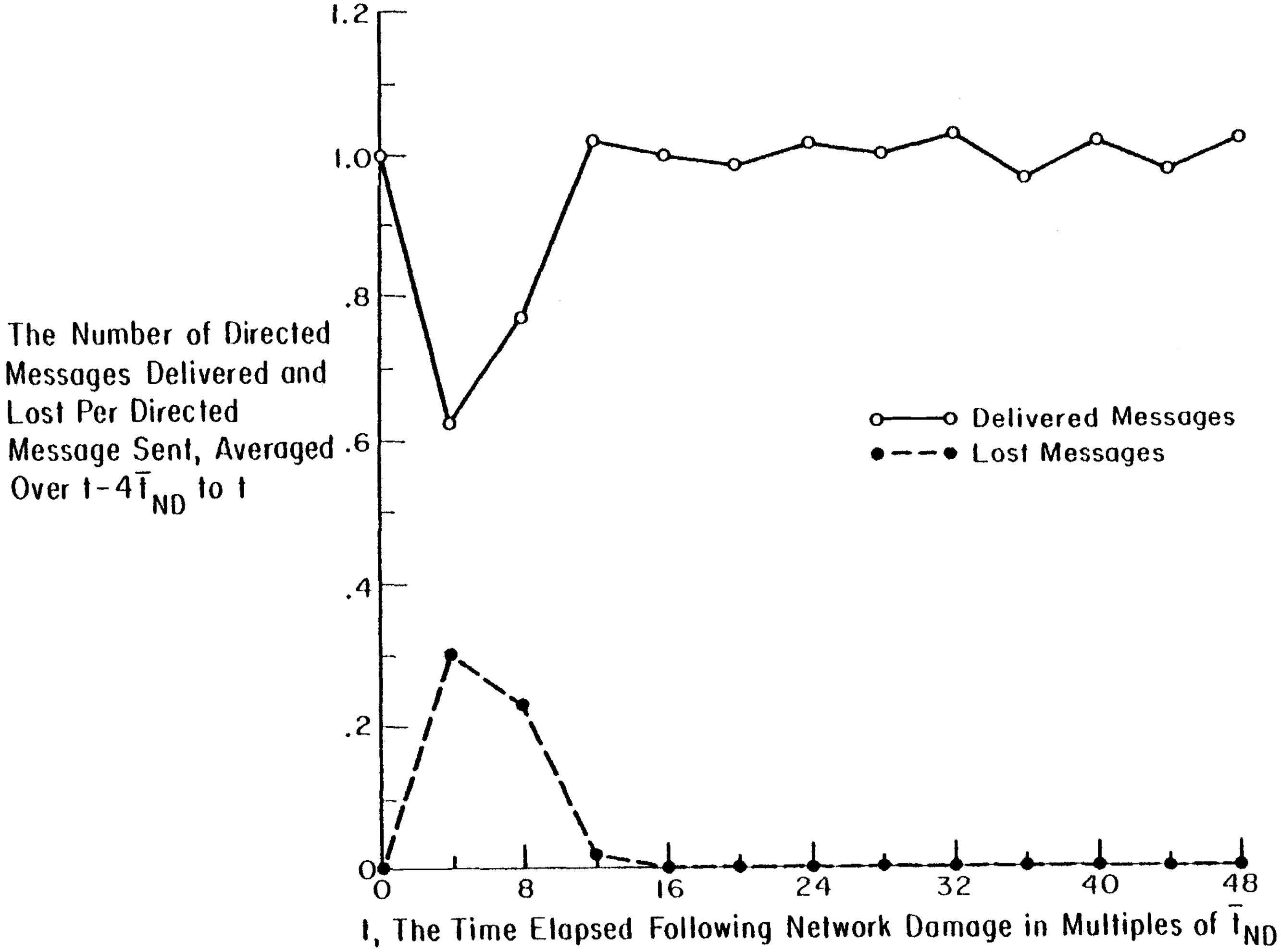


Figure 4.59. RPSDV algorithm with $T_{DV} = 4\bar{t}_{ND}$, three links jammed.

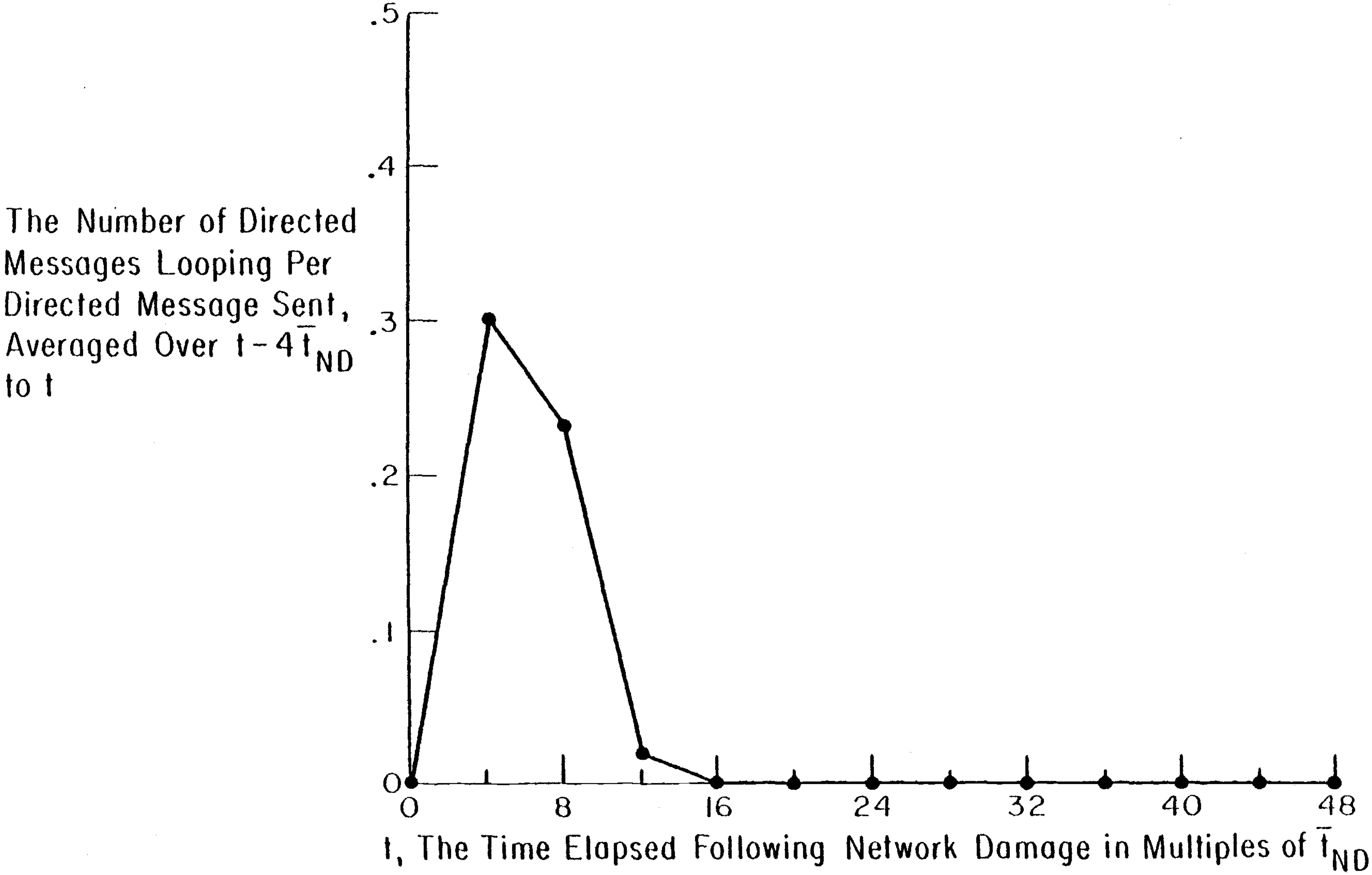


Figure 4.60. RPSDV algorithm with $T_{DV} = 4\bar{t}_{ND}$, three links jammed.

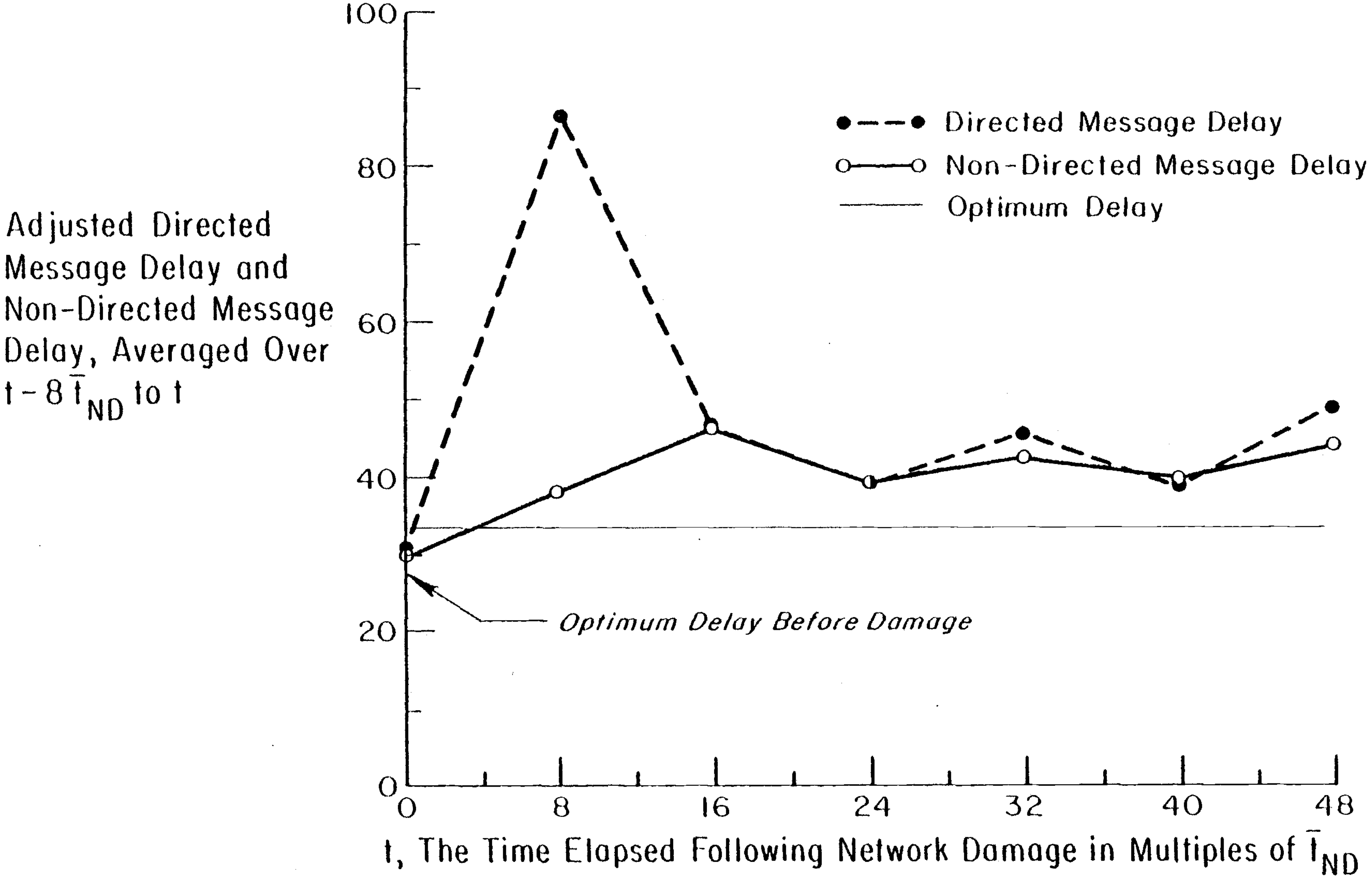


Figure 4.61. RPSDV algorithm with $T_{DV} = 4\bar{t}_{ND}$, three links jammed.

thus causing incorrect entries to work their way into the Delay Vector Tables. Since the Delay Vector Tables are also used for routing directed messages, these incorrect entries result in an increase in lost and looping messages. Once the A tables adapt, the new correct entries are able to work their way into the Delay Vector Tables since the delay vectors are sent out on a regular basis, and thus are able to correct the earlier bad entries. There is a lag, however, between the time when the A tables adapt and when the delay vectors are able to completely correct the Delay Vector Table entries.

The results for the directed message delay again show the initial large value due to the lost messages. Following this initial period, however, the directed message delay compares favorably with the non-directed message delay. This indicates that the directed messages are being routed over shortest paths. This is an improvement over the corresponding results with the Reciprocal Path Search algorithm. The Reciprocal Path Search algorithm did not use delay vectors and routed messages only over reciprocal paths, resulting in some messages taking longer than necessary routes. The use of the delay vectors enables the directed messages to be routed over non-reciprocal links and thus take the shortest routes to the destination nodes. Thus, although the use of the delay vectors causes an early increase in the number of lost messages, they also allow the messages to take improved routes and thus decrease the steady state delay.

The results for the case with $T_{DV} = 2\bar{t}_{ND}$ are given in Figures 4.62, 4.63, and 4.64. Again there is the initial period of lost and looping messages after which the algorithm completely adapts. However, the period of lost messages decreases to $8\bar{t}_{ND}$ and fewer messages are lost than when the delay vectors were sent out every $4\bar{t}_{ND}$. This improvement can be attributed to the increased rate of delay vector transmissions more quickly purging the Delay Vector Tables of their early bad entries. There are still, however, more lost and looping messages than without the use of delay vectors.

Following the usual initial large delays due to the penalty imposed for lost messages, the directed message delay again compares favorably with the non-directed message delay indicating the directed messages are taking the shortest routes to their destinations. There is a slight increase in these delays when compared to the $T_{DV} = 4\bar{t}_{ND}$ case since the increased rate of delay vector transmission adds some extra congestion in the network. This increase in delay is very slight.

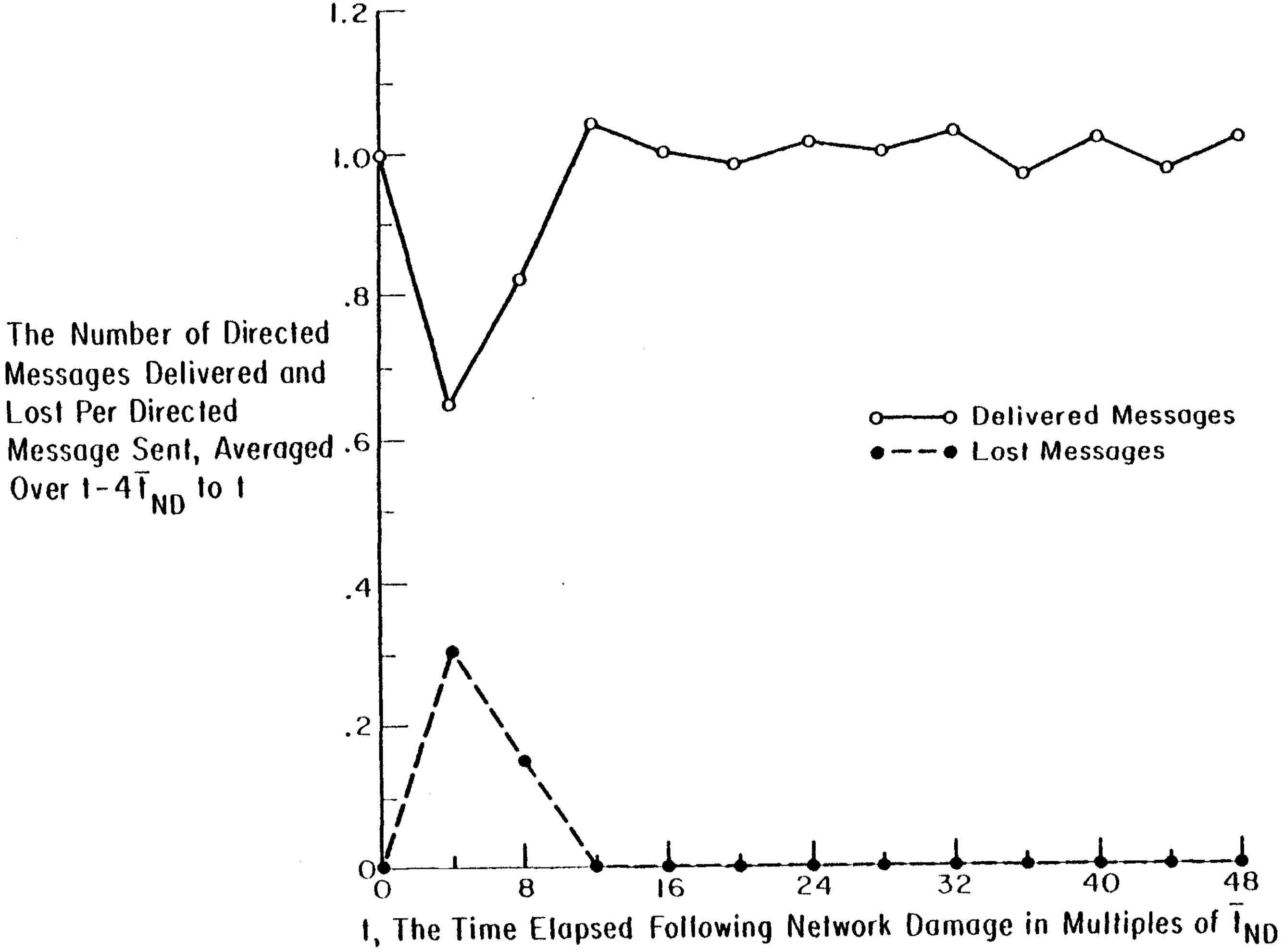


Figure 4.62. RPSDV algorithm with $T_{DV} = 2\bar{t}_{ND}$, three links jammed.

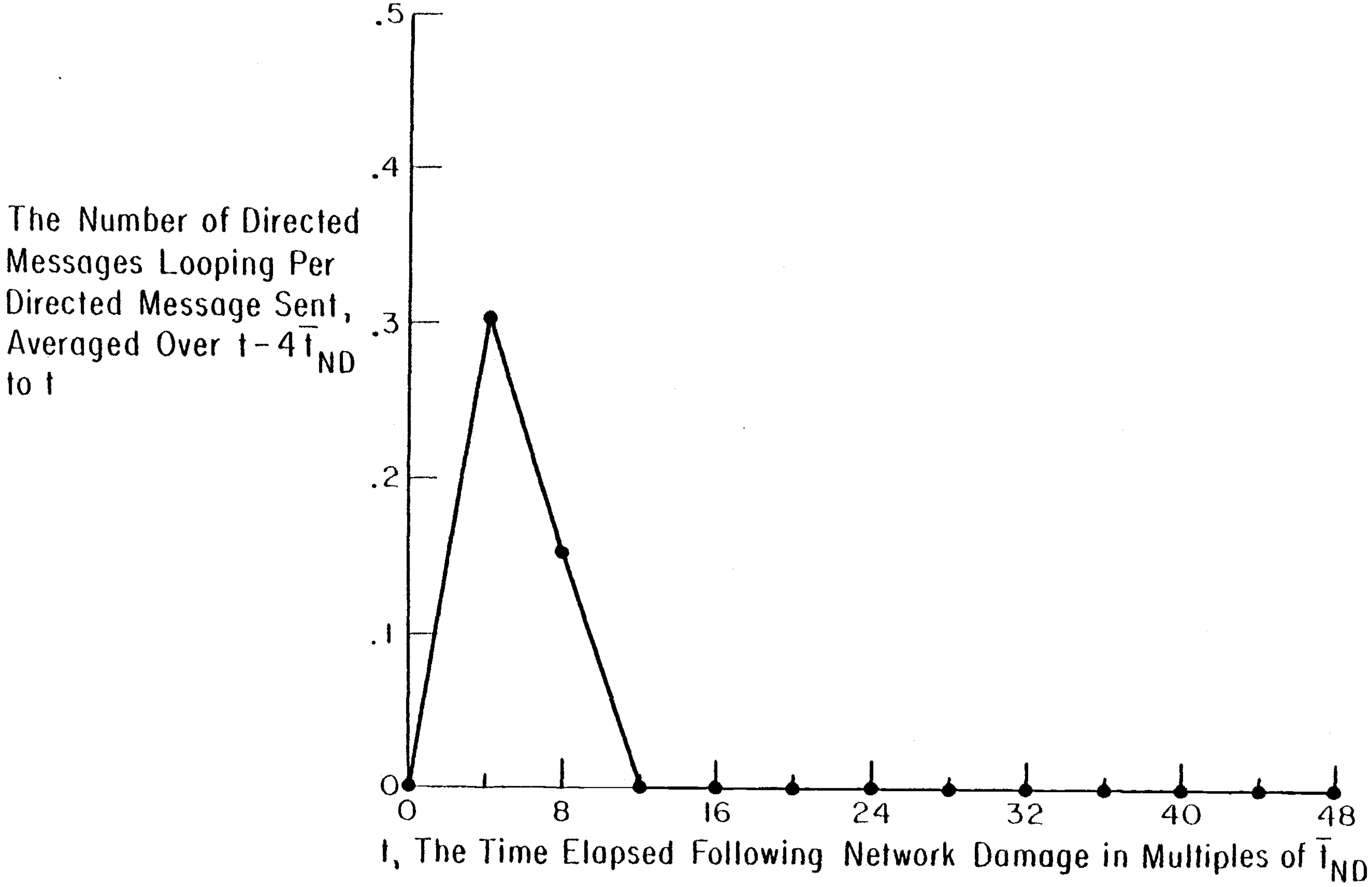


Figure 4. 63. RPSDV algorithm with $T_{DV} = 2\bar{t}_{ND}$, three links jammed.

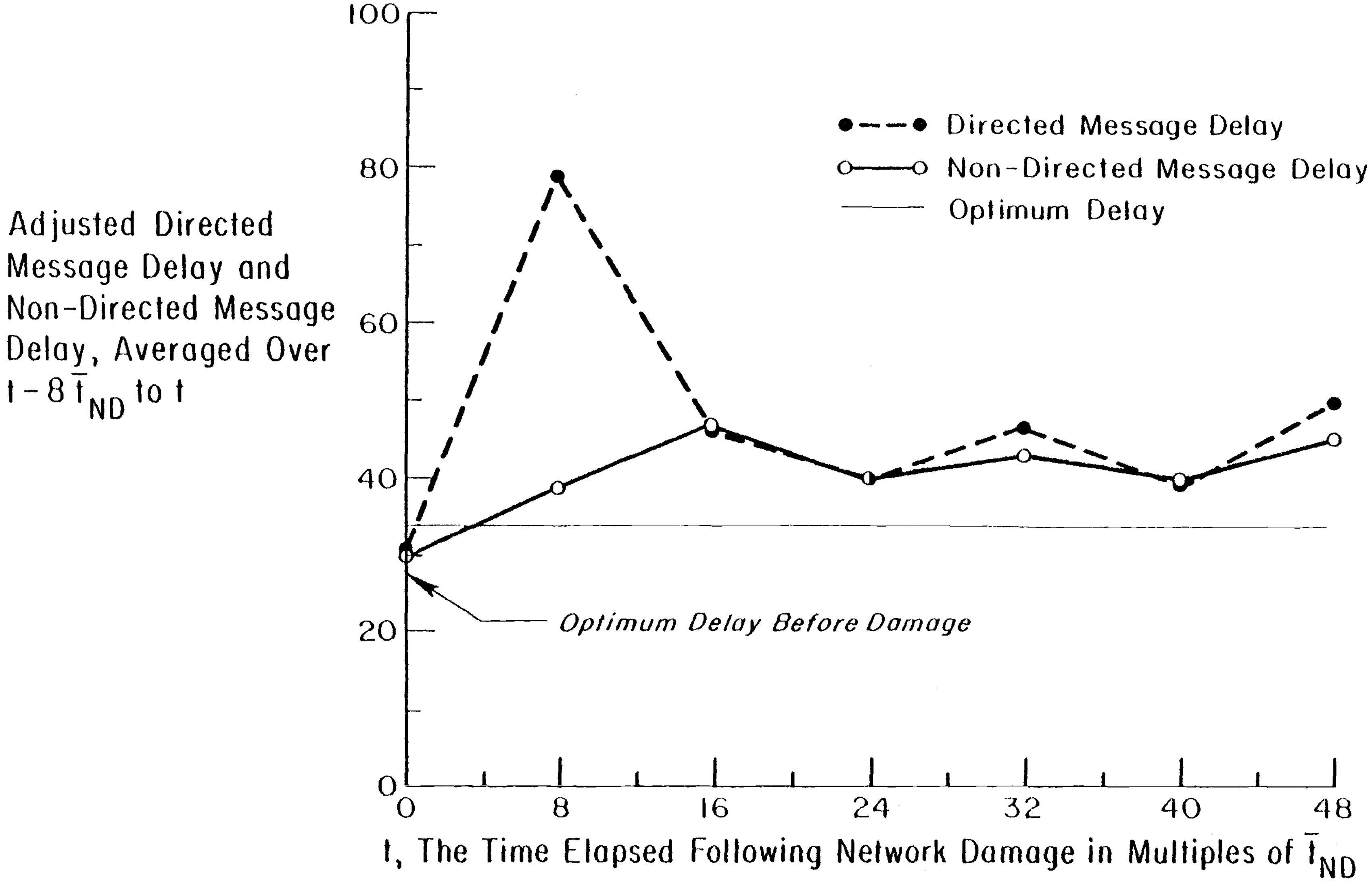


Figure 4.64. RPSDV algorithm with $T_{DV} = 2\bar{t}_{ND}$, three links jammed.

Six Links Jammed

Again, this case of network damage was tested for values of $T_{DV} = 4\bar{t}_{ND}$ and $T_{DV} = 2\bar{t}_{ND}$.

The results for the $T_{DV} = 4\bar{t}_{ND}$ case are given in Figures 4.65, 4.66, and 4.67. These results are a dramatic improvement over the Reciprocal Path Search algorithm, which does not use delay vectors. Without the use of delay vectors to locate the non-reciprocal links, the previous algorithm was unable to adapt to the six links jammed case and message were lost and looping continuously after the damage occurred. However, the Reciprocal Path Search Algorithm with Delay Vectors is able to completely adapt to this case of network damage. It is interesting to note that messages were lost for a period of $16\bar{t}_{ND}$ and they looped for only a period of $8\bar{t}_{ND}$. The messages are lost for a greater period because it takes several iterations of the delay vector transmissions for information on paths which contain several non-reciprocal links to completely propagate throughout the network. In general, if a source node-destination node pair has three links between them which suddenly become non-reciprocal at closely spaced instants in time (as is done in our simulations), it takes three iterations of the delay vector transmissions before complete information on the path is communicated to the source node. In the mean time, if this path with three non-reciprocal

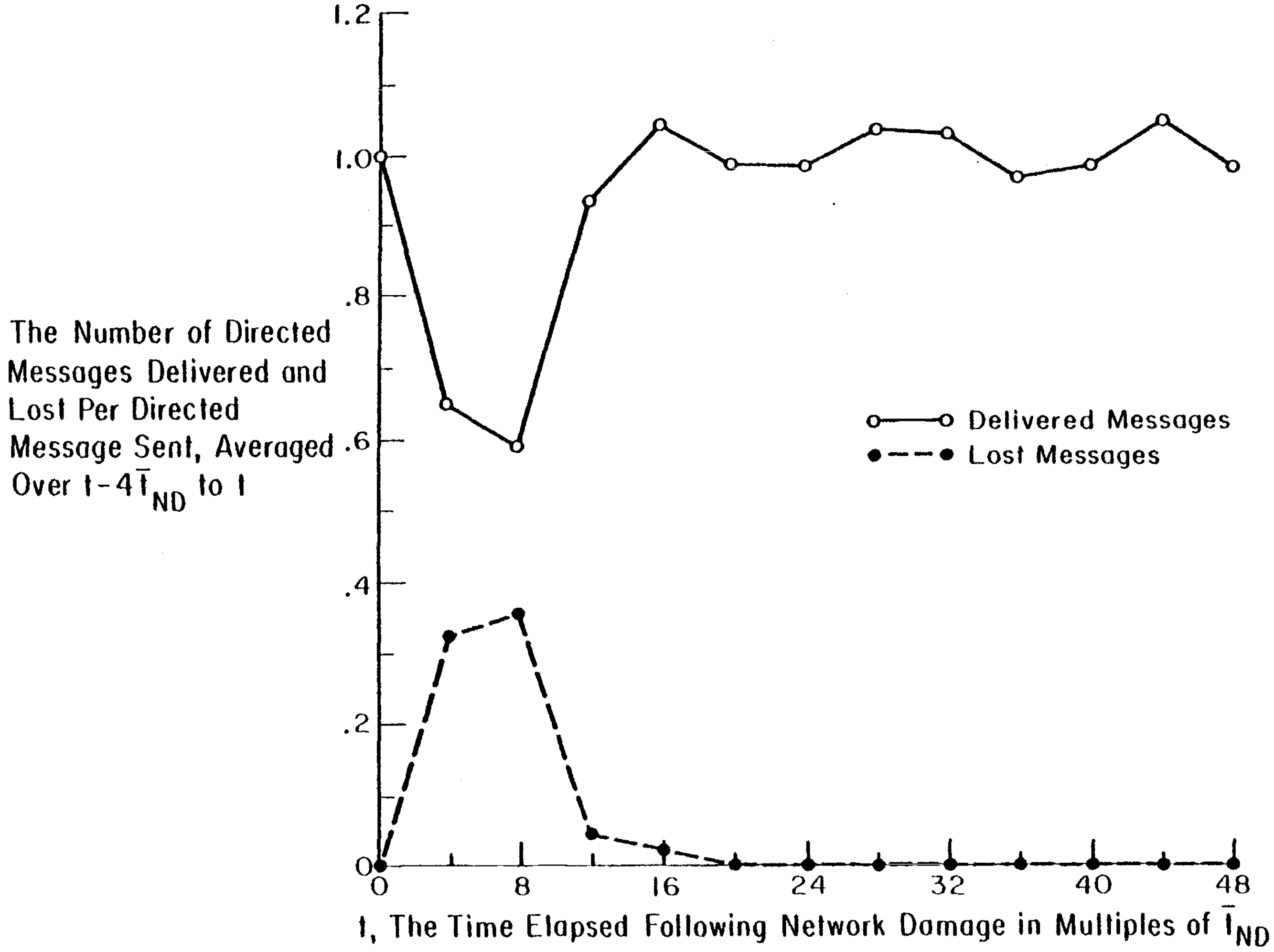


Figure 4.65. RPSDV algorithm with $T_{DV} = 4\bar{t}_{ND}$, six links jammed.

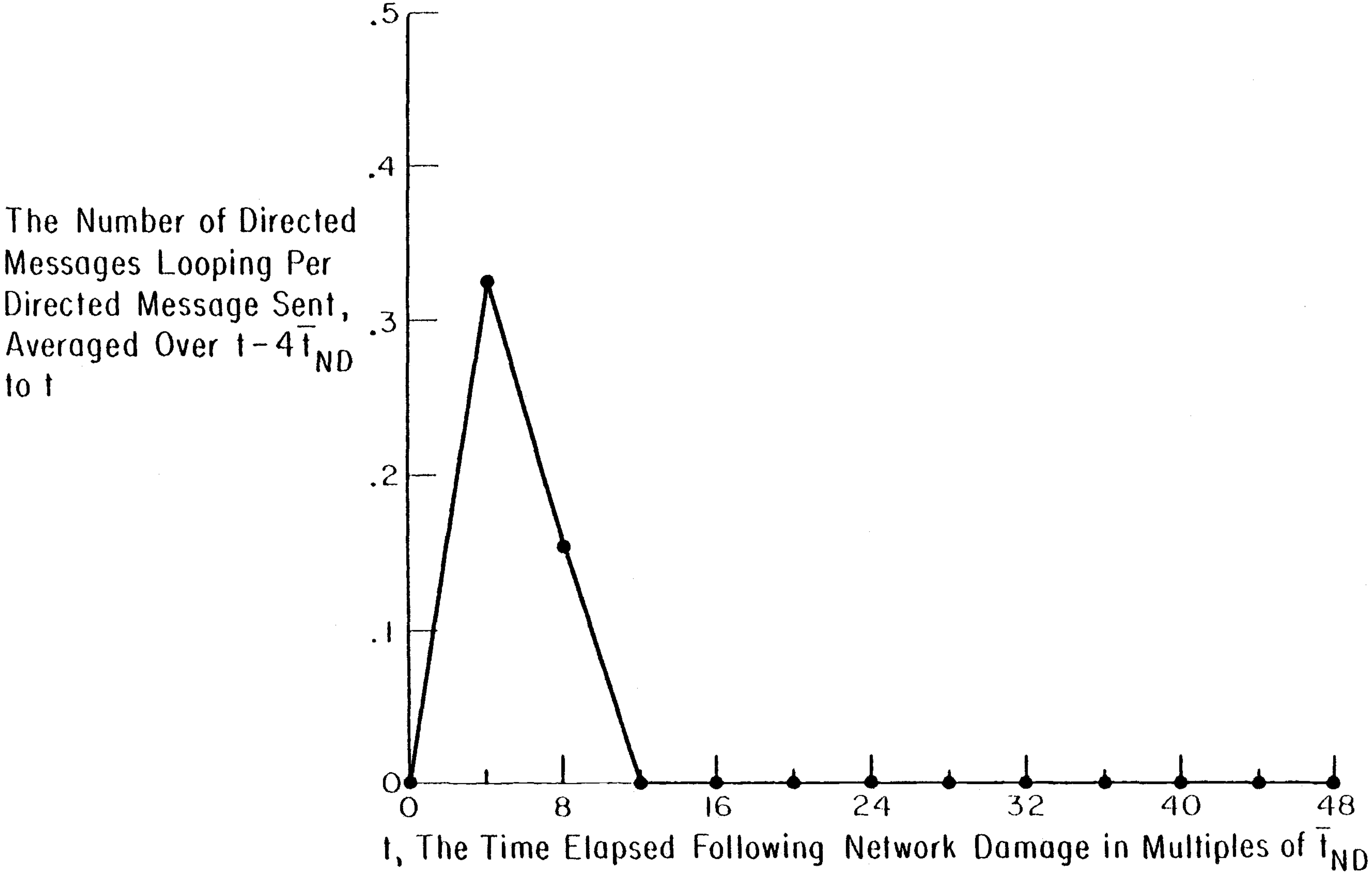


Figure 4.66. RPSDV algorithm with $T_{DV} = 4\bar{t}_{ND}$, six links jammed.

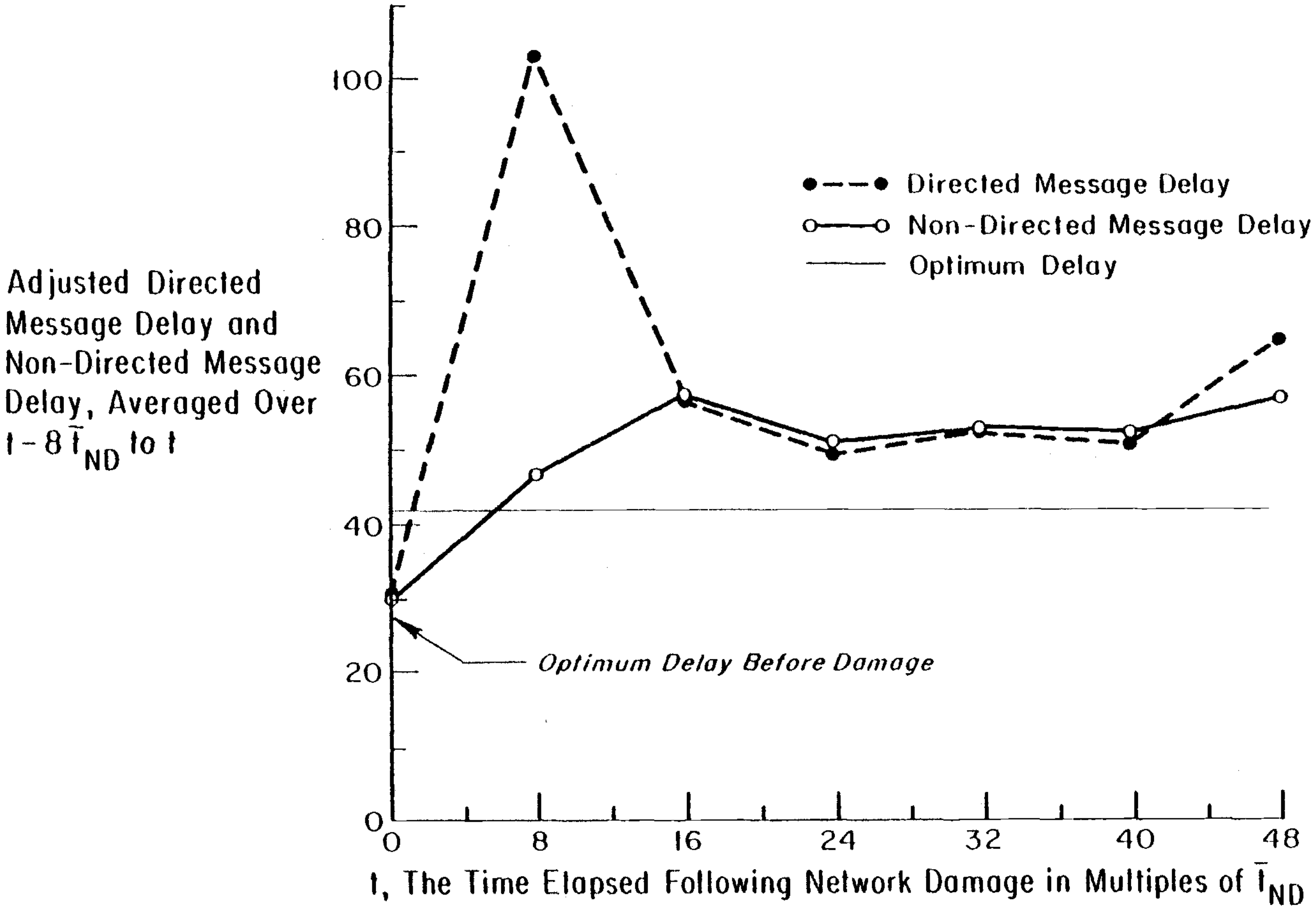


Figure 4.67. RPSDV algorithm with $T_{DV} = 4\bar{t}_{ND}$, six links jammed.

links is the only path between the source and destination nodes, the source node may conclude that there is no path to the destination node, and discard any messages to be sent to it. This process is speeded up if the delay vectors are sent out more frequently.

Following an initial period of large delay due to lost messages, the directed message delay becomes essentially equal to the non-directed message delay. This indicates that the Reciprocal Path Search Algorithm with Delay Vectors is able to route the messages over the shortest paths.

Figures 4.68, 4.69, and 4.70 give the results when $T_{DV} = 2\bar{t}_{ND}$. These results show that the algorithm adapts more quickly than with $T_{DV} = 4\bar{t}_{ND}$. The algorithm adapts in about $8\bar{t}_{ND}$ versus the $16\bar{t}_{ND}$ with the less frequent delay vector transmissions. This is a result of the more frequent transmissions purging the Delay Vector Tables of bad entries more quickly, and also speeding up the process of disseminating throughout the network complete information on all paths.

The adjusted directed message delay again shows the initial large value due to the lost messages. Once the algorithm has adapted, however, the directed message delay is essentially the same as the non-directed message delay, again indicating that the algorithm is routing the messages over the shortest paths. The increased congestion causes by the more frequent delay vector transmissions

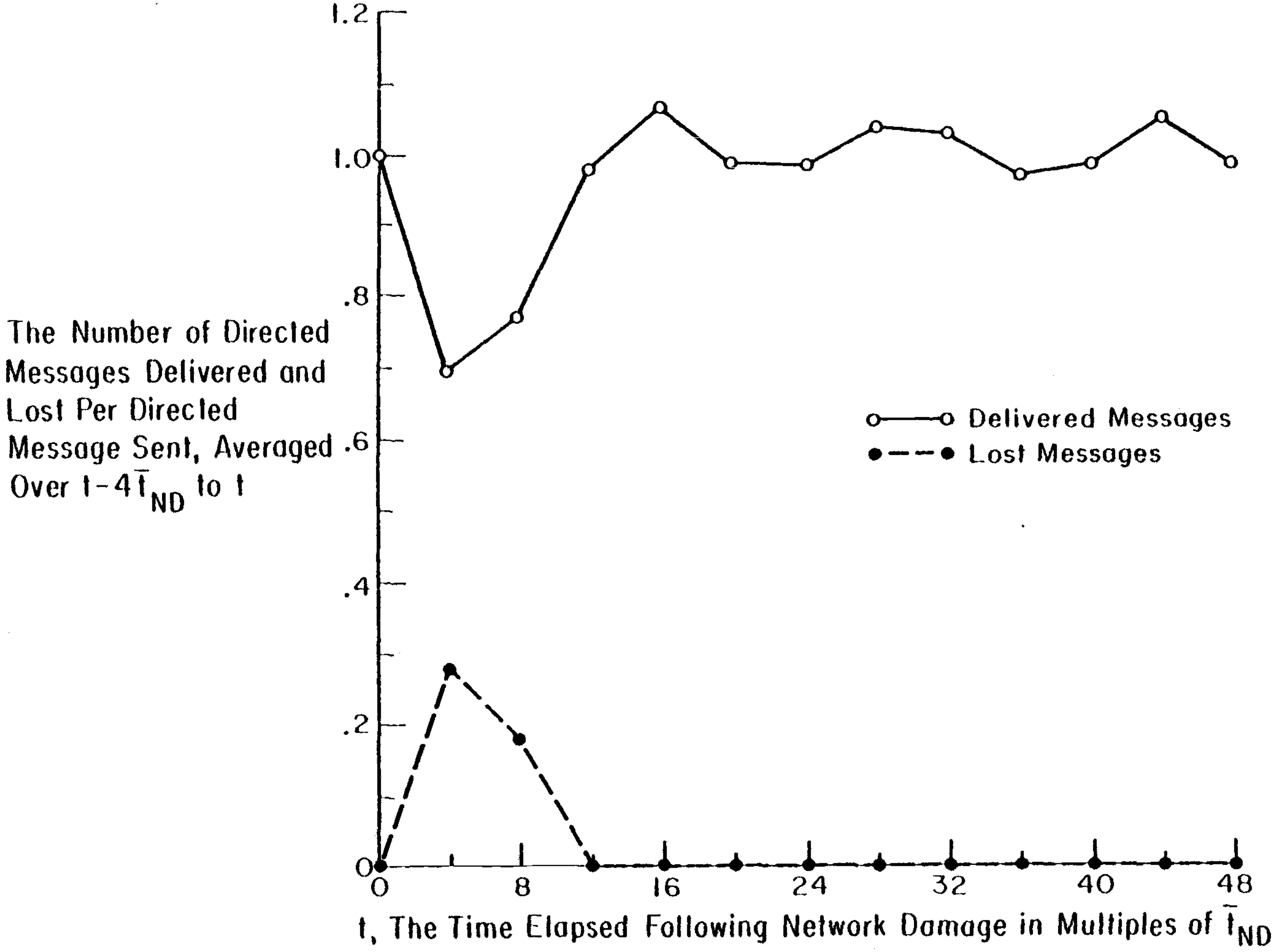


Figure 4.68. RPSDV algorithm with $T_{DV} = 2\bar{t}_{ND}$, six links jammed.

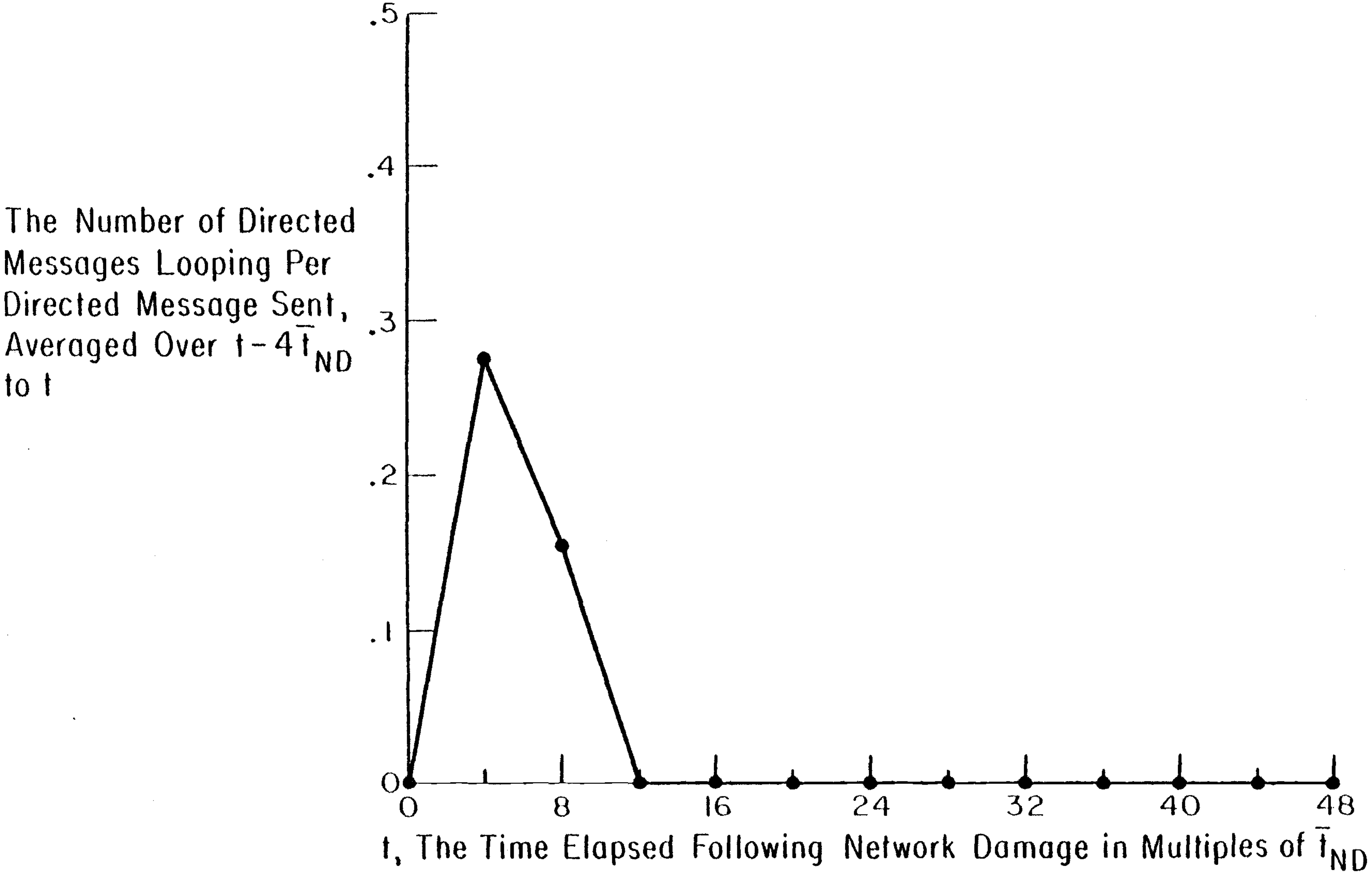


Figure 4.69. RPSDV algorithm with $T_{DV} = 2\bar{t}_{ND}$, six links jammed.

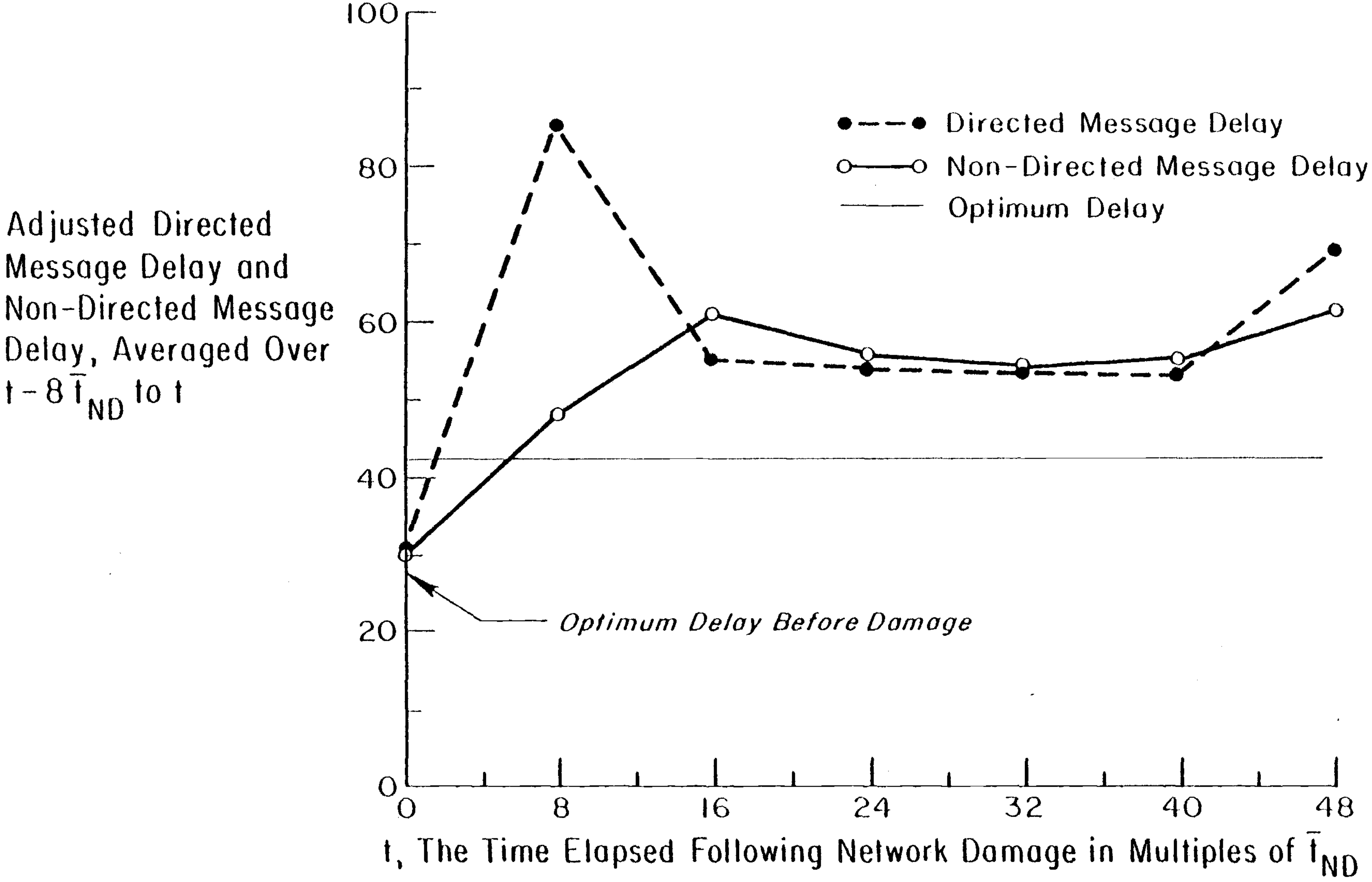


Figure 4.70. RPSDV algorithm with $T_{DV} = 2\bar{t}_{ND}$, six links jammed.

results in a slight increase in directed and non-directed message delays when compared to the $T_{DV} = 4\overline{t}_{ND}$ case.

In summary, it is evident that for severe cases of network damage, the Reciprocal Path Search Algorithm with Delay Vectors is a large improvement over the Reciprocal Path Search algorithm, which does not use delay vectors. Since the use of delay vectors is necessary for the Reciprocal Path Search algorithm to completely adapt to severe cases of network jamming, they are an essential addition to the algorithm.

Rapid Reciprocal Path Search Algorithm with Delay Vectors

This algorithm combines the Rapid Reciprocal Path Search algorithm with the use of delay vectors to create an algorithm which adapts very quickly to network damage, yet has the capability to use non-reciprocal links for routing directed messages.

The Rapid Reciprocal Path Search Algorithm with Delay Vectors (RRPSDV) stores routing information in four tables. The A, B, and C tables contain information on reciprocal paths while the Delay Vector Table stores information on non-reciprocal paths. The techniques for creating delay vectors and using them to update the Delay Vector Tables are exactly the same as in the Reciprocal Path Search Algorithm with Delay Vectors, and will not be repeated here.

Non-Directed Message Handling

The non-directed message handling procedure is described in Figures 4.71 and 4.72. This procedure is exactly the same as in the Rapid Reciprocal Path Search algorithm, except that the algorithm now checks to see if the non-directed message is a delay vector, and if so, sends it to the Delay Vector Table update routine.

Directed Message Handling

The directed message handling procedure is described in Figures 4.73 and 4.74. This procedure is very similar to the procedure used in the Reciprocal Path Search Algorithm with Delay Vectors. There are, however, some important differences in how the messages are routed. First, the delay estimates for the reciprocal path are now drawn from either the A or C table, depending upon the value of the B table entry. The delay estimate to node j by way of port i is thus e_{ij}, where

$$e_{ij} = \begin{cases} a_{ij} & \text{if} \quad b_{ij} = 0 \\ c_{ij} & \text{if} \quad b_{ij} > 0 \end{cases} \qquad (4.17)$$

Another difference is that the algorithm is written so that for cases of link destruction messages do not loop. This is the same effect discussed previously in the Rapid Reciprocal Path Search

Rapid Reciprocal Path Search Algorithm with Delay Vectors
Non-Directed Message Handling

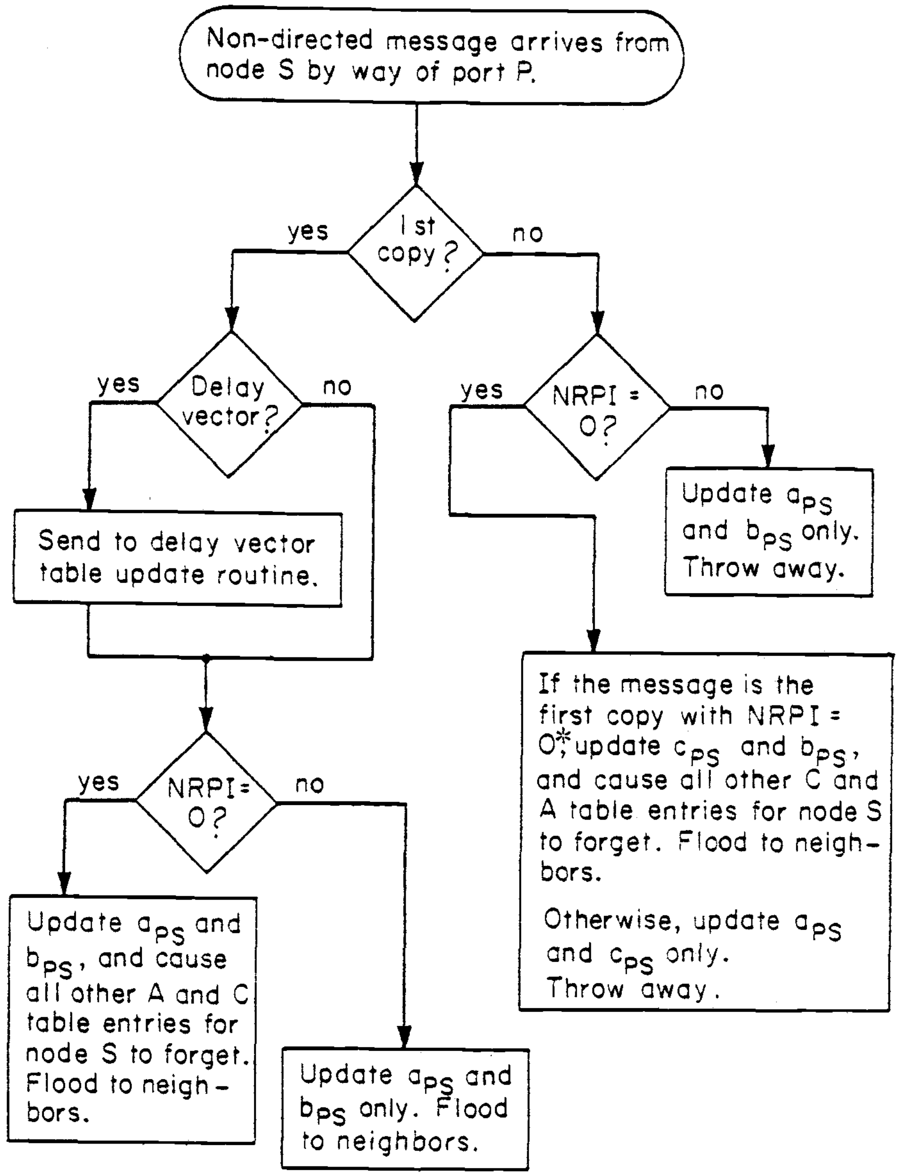

Non-directed message arrives from node S by way of port P.
1 st copy?
yes
no
Delay vector?
yes
no
NRPI = O?
yes
no
Send to delay vector table update routine.
Update a_{PS} and b_{PS} only. Throw away.
NRPI = O?
yes
no
If the message is the first copy with NRPI = O*; update c_{PS} and b_{PS}, and cause all other C and A table entries for node S to forget. Flood to neighbors.
Otherwise, update a_{PS} and c_{PS} only. Throw away.
Update a_{PS} and b_{PS}, and cause all other A and C table entries for node S to forget. Flood to neighbors.
Update a_{PS} and b_{PS} only. Flood to neighbors.

Figure 4.71. Conceptual block diagram.

*All previous copies had NRPI = 1.

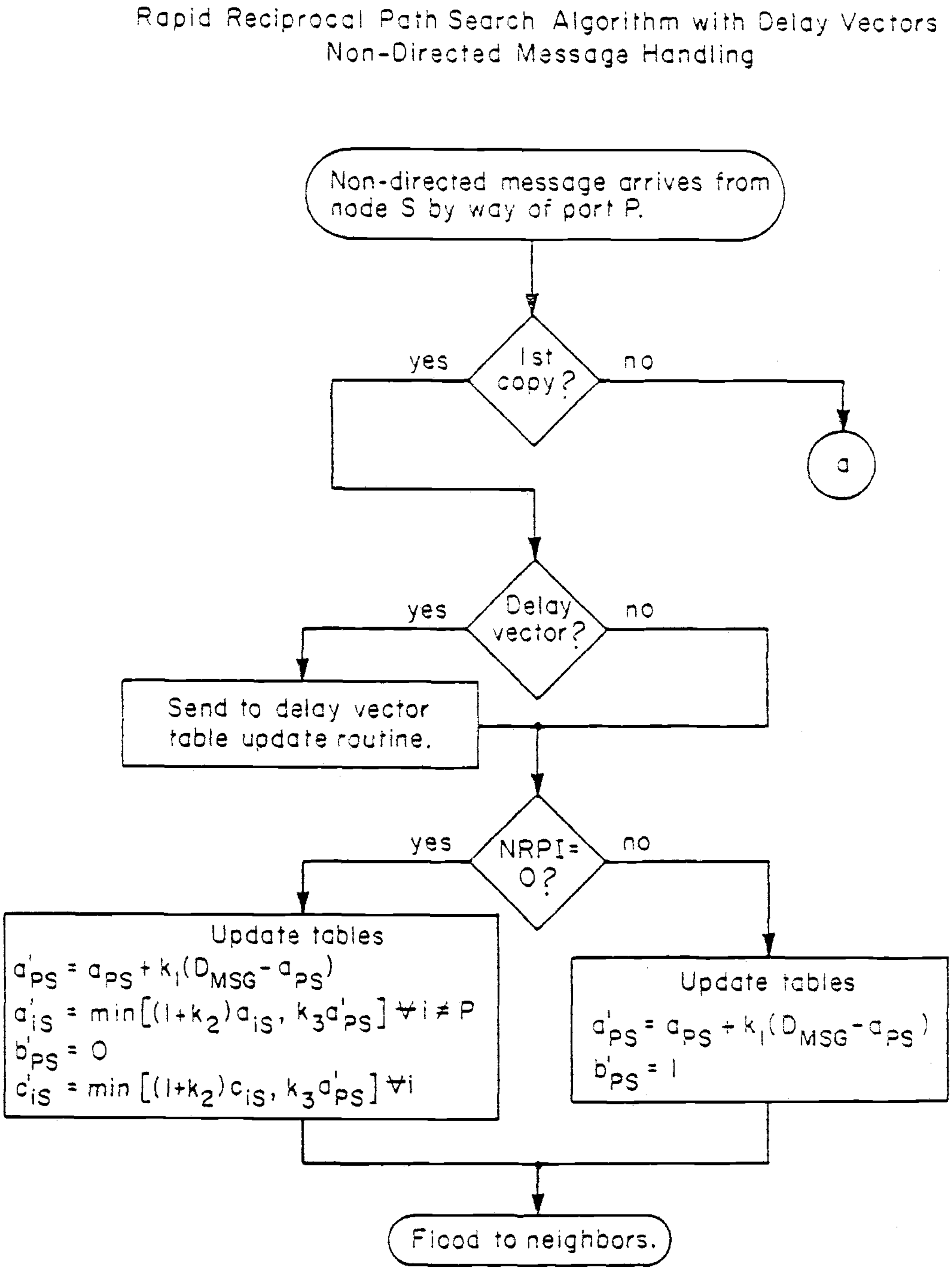

Rapid Reciprocal Path Search Algorithm with Delay Vectors
Non-Directed Message Handling
Non-directed message arrives from node S by way of port P.
yes
1st copy?
no
a
yes
Delay vector?
no
Send to delay vector table update routine.
yes
NRPI= 0?
no
Update tables
$a'_{PS} = a_{PS} + k_1(D_{MSG} - a_{PS})$
$a'_{iS} = \min[(1+k_2)a_{iS}, k_3 a'_{PS}] \forall i \neq P$
$b'_{PS} = 0$
$c'_{iS} = \min[(1+k_2)c_{iS}, k_3 a'_{PS}] \forall i$
Update tables
$a'_{PS} = a_{PS} + k_1(D_{MSG} - a_{PS})$
$b'_{PS} = 1$
Flood to neighbors.

Figure 4. 72. Detailed block diagram.

Rapid Reciprocal Path Search Algorithm with Delay Vectors
Non-Directed Message Handling (cont.)

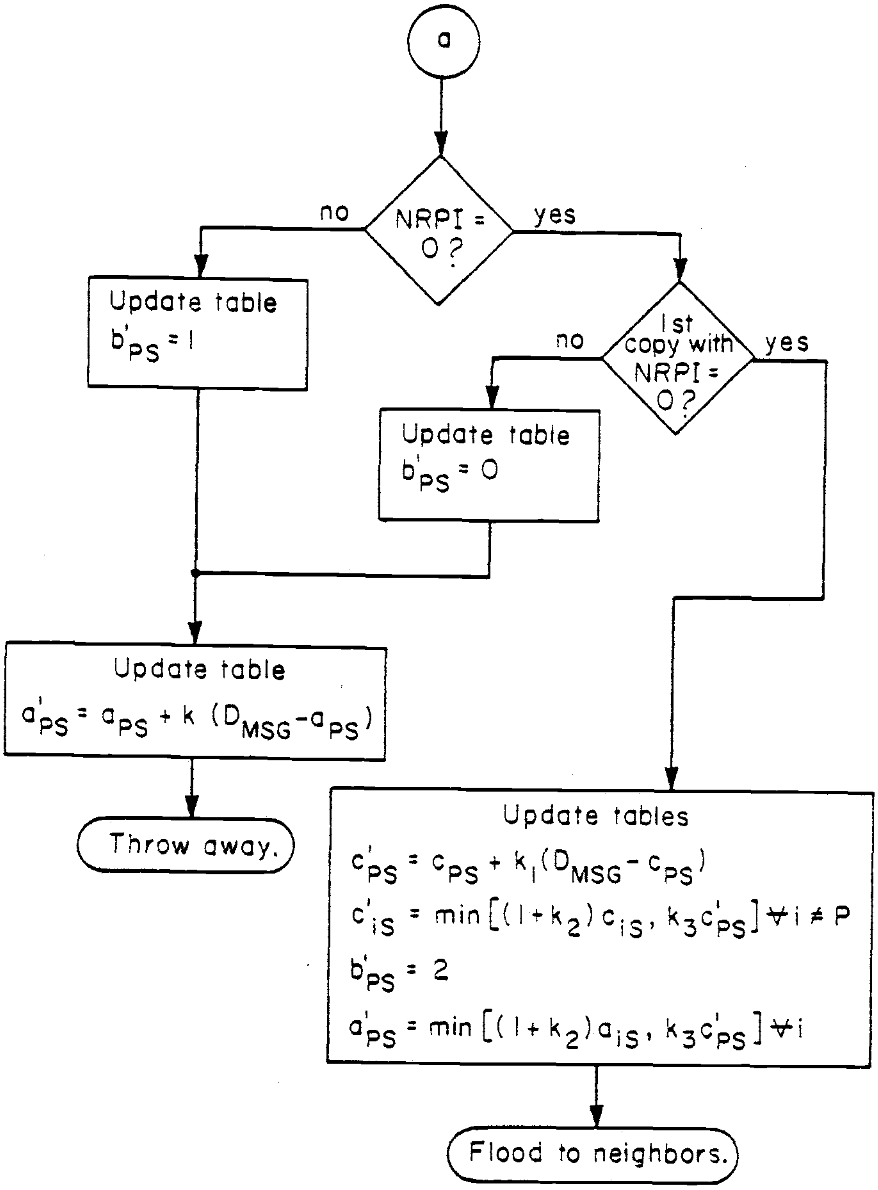

a
NRPI = 0?
no
yes
Update table $b'_{PS} = 1$
1st copy with NRPI = 0?
no
yes
Update table $b'_{PS} = 0$
Update table
$a'_{PS} = a_{PS} + k\ (D_{MSG} - a_{PS})$
Throw away.
Update tables
$c'_{PS} = c_{PS} + k_1(D_{MSG} - c_{PS})$
$c'_{iS} = \min[(1+k_2)c_{iS}, k_3c'_{PS}]\ \forall i \neq P$
$b'_{PS} = 2$
$a'_{PS} = \min[(1+k_2)a_{iS}, k_3c'_{PS}]\ \forall i$
Flood to neighbors.

Figure 4. 72. Continued.

Rapid Reciprocal Path Search Algorithm with Delay Vectors
Directed Message Handling

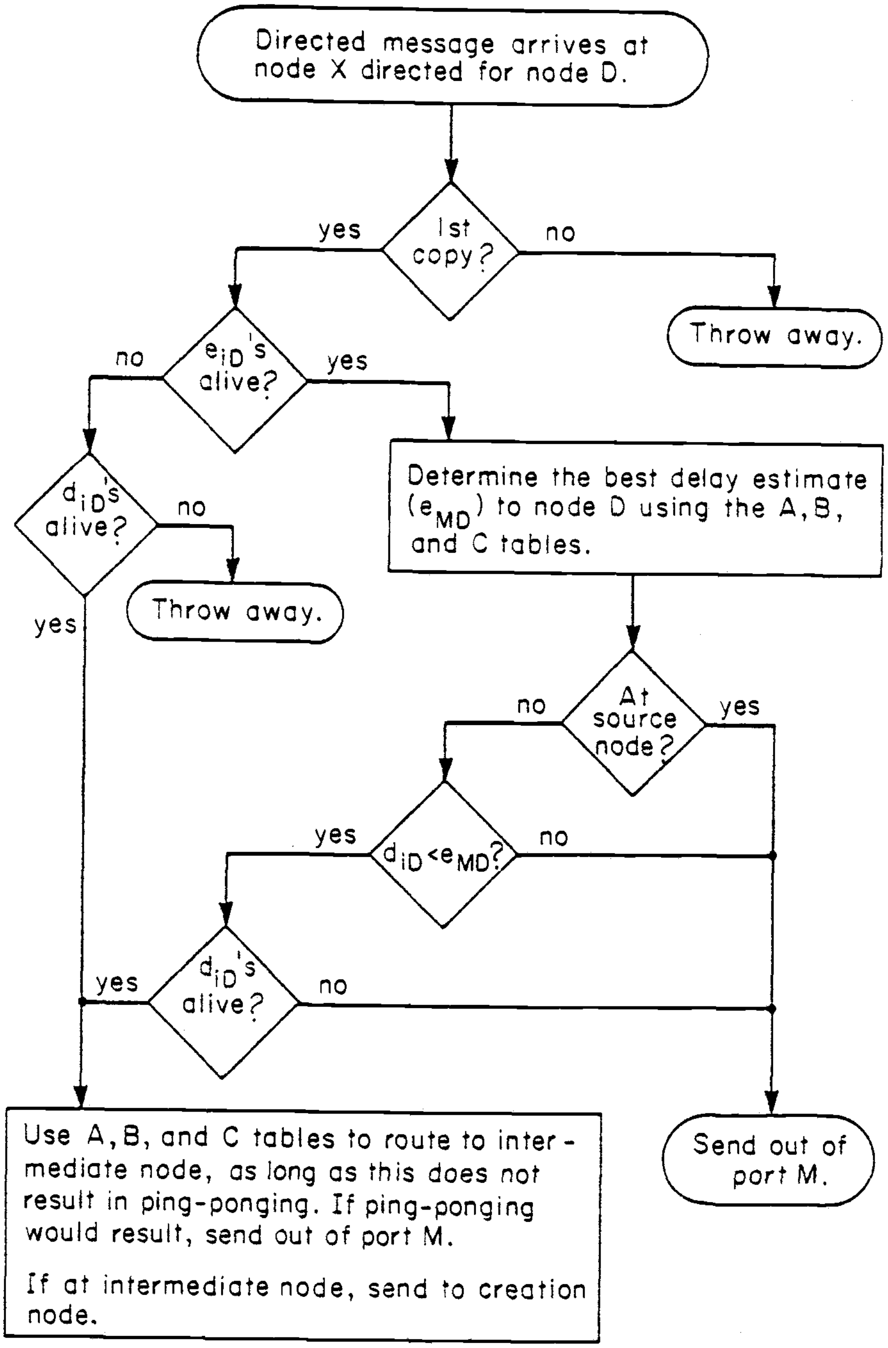

Directed message arrives at node X directed for node D.
1st copy?
yes
no
Throw away.
e_{iD}'s alive?
no
yes
d_{iD}'s alive?
no
Throw away.
yes
Determine the best delay estimate (e_{MD}) to node D using the A, B, and C tables.
At source node?
no
yes
$d_{iD} < e_{MD}$?
yes
no
d_{iD}'s alive?
yes
no
Use A, B, and C tables to route to intermediate node, as long as this does not result in ping-ponging. If ping-ponging would result, send out of port M.
If at intermediate node, send to creation node.
Send out of port M.

Figure 4.73. Conceptual block diagram.

Rapid Reciprocal Path Search Algorithm with Delay Vectors
Directed Message Handling

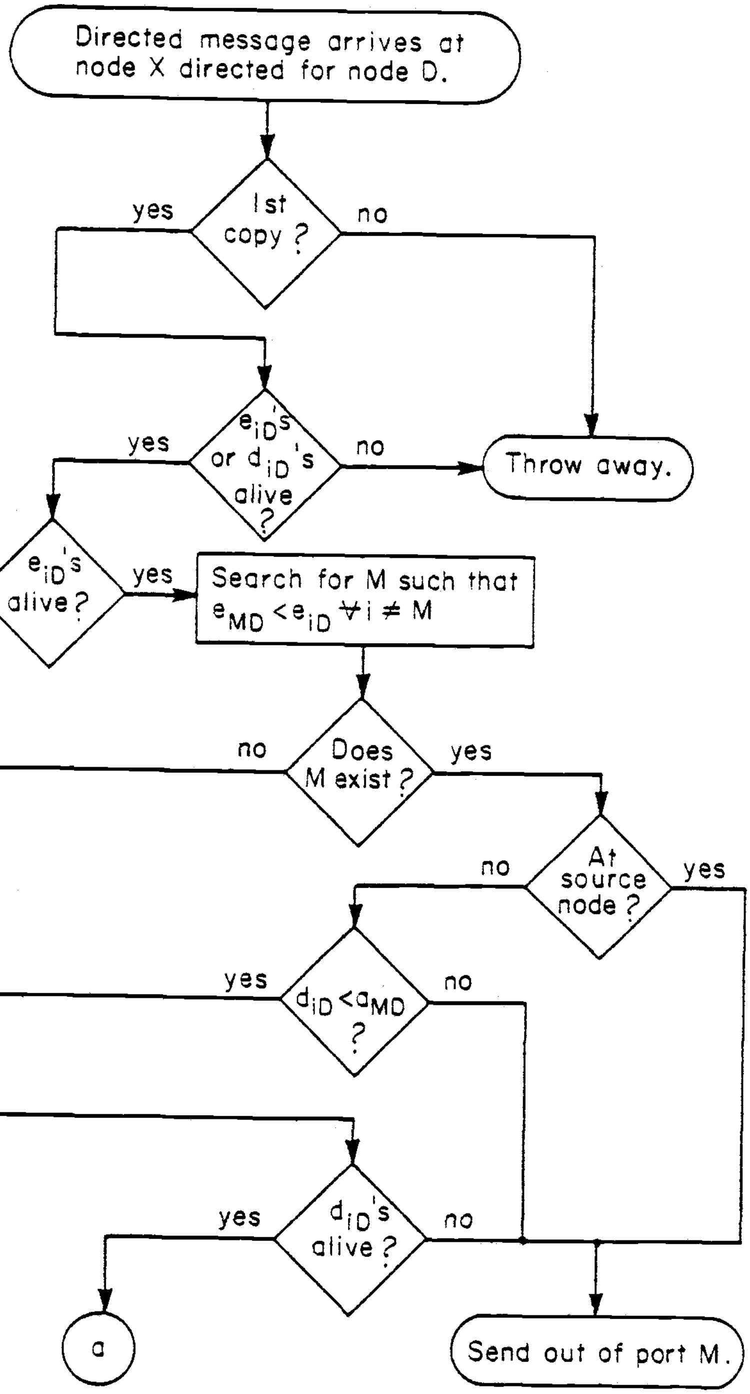


Figure 4.74. Detailed block diagram.

Rapid Reciprocal Path Search Algorithm with Delay Vectors
Directed Message Handling (cont.)

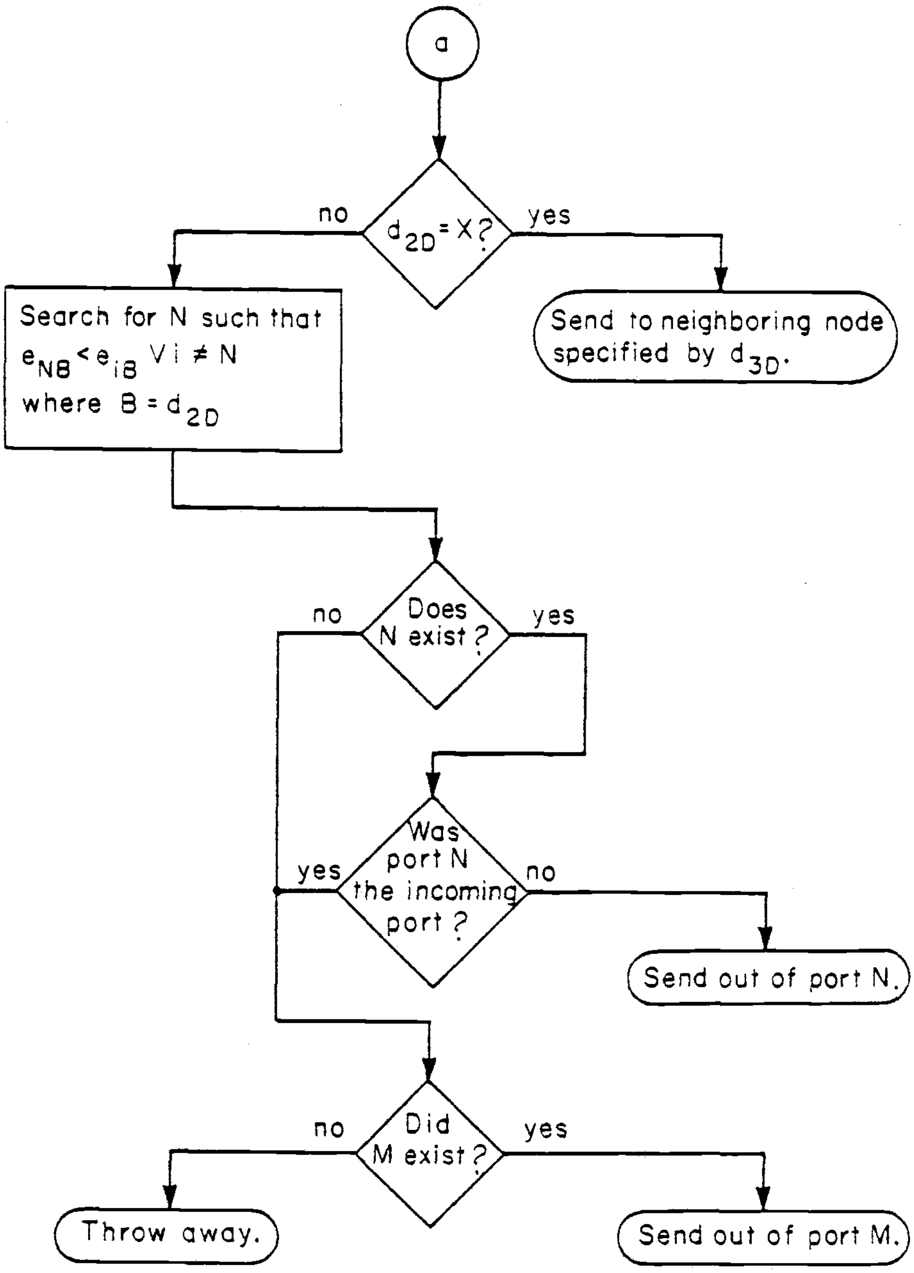

a
$d_{2D} = X$?
no
yes
Search for N such that $e_{NB} < e_{iB}$ ∀ i ≠ N where B = d_{2D}
Send to neighboring node specified by d_{3D}.
Does N exist?
no
yes
Was port N the incoming port?
yes
no
Send out of port N.
Did M exist?
no
yes
Throw away.
Send out of port M.

Figure 4.74. Continued.

algorithm. The looping which occurs is primarily due to messages sent to a node which was previously, but is no longer, connected to the destination. For the case of link jamming, the b_{ij}'s select delay estimates which tell the disconnected node to send the message back to the nodes to which it is still connected, resulting in ping-ponging, and occasionally allowing the message to proceed on to its destination. However, the b_{ij}'s are updated only when jamming occurs. Thus, for the case of link destruction, the b_{ij}'s select delay estimates which direct the disconnected node to transmit the message over the destroyed link, resulting in the message not being sent out at all and preventing ping-ponging (a form of looping). Even though the messages do not loop, they are still lost, resulting in very little change in performance.

The above modifications to the routing algorithm used previously by the Reciprocal Path Search Algorithm with Delay Vectors are rather minor, and are reflective of the use of the A, B, and C tables for routing over reciprocal links instead of just using the A table. The next two modifications, however, are major alterations. Preliminary simulation runs indicated that unless some restrictions are placed upon the use of the Delay Vector Table entries for routing purposes, an excessive amount of looping occurs, eliminating the quick adaptability which is a prime goal of this algorithm. The looping problem necessitated that the nodes be allowed to use the Delay

Vector Table entries for routing purposes only when using them provides a clear benefit. Therefore, nodes are prevented from using the Delay Vector Table entries, even though they indicate a shorter path exists over a non-reciprocal link, when using them results in sending the message out of the same port in which it arrived. In this case, the node falls back on the use of the A, B, and C tables. This restriction makes clear sense, since allowing a message to be transmitted out of the port in which it came will result in ping-ponging, and the loss of the message. The preliminary simulation runs also indicated that a second modification of preventing a source node for a directed message from using its Delay Vector Table for routing directed messages further reduced ping-ponging.

There appear to be some basic factors in the Rapid Reciprocal Path Search algorithm which make it difficult to combine with the use of delay vectors. The delay vectors draw many of their delay estimates from the A and C tables. Following network damage, estimates derived from the A and C tables are, for any one node, excellent indicators of which port should be taken to reach the destination node. However, whereas the delay estimates for each output port may show the correct rankings of these ports, their absolute magnitudes may vary significantly from node to node. Therefore, from node to node, the relative magnitudes of the e_{ij}'s may not compare well.

However, the delay vectors draw their delay estimates from the e_{ij}'s at node X and store them in node Y's Delay Vector Table, where they are compared with Y's e_{ij}'s. This leads to incorrect routing decisions, and thus appears to be the underlying reason why the restrictions had to be placed on the use of the Delay Vector Tables. Besides the ping-pong and source node restrictions, it was found that reducing the size of the limiting constant, k_3, to three also improved the algorithm's performance apparently by reducing the discrepancy in the relative e_{ij} magnitudes from node to node.

Three Links Destroyed

The results for this case of network damage are given in Figures 4.75 and 4.76. The results are essentially the same as for the Rapid Reciprocal Path Search algorithm. This algorithm differs significantly from the Rapid Reciprocal Path Search algorithm only when jamming occurs. For the three links destroyed case, delay vectors are not even sent out since the network contains no non-reciprocal links.

The results show the usual period when messages are temporarily lost, after which the algorithm completely adapts. The adjusted directed message delay again has an initial period during which it is large because of the lost messages. Once the algorithm has adapted, the directed message delay is essentially the same as

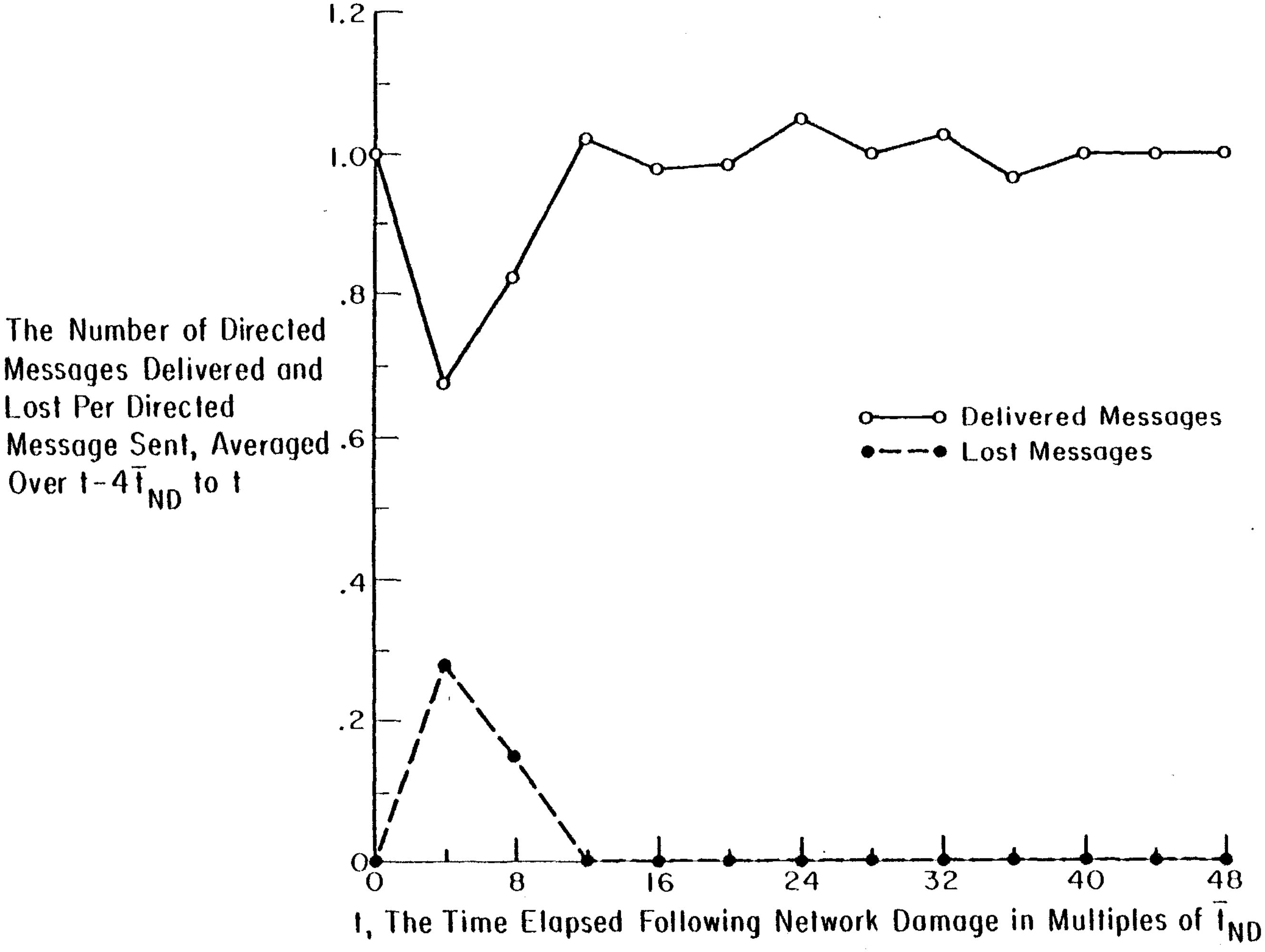


Figure 4.75. RRPSDV algorithm, three links destroyed.

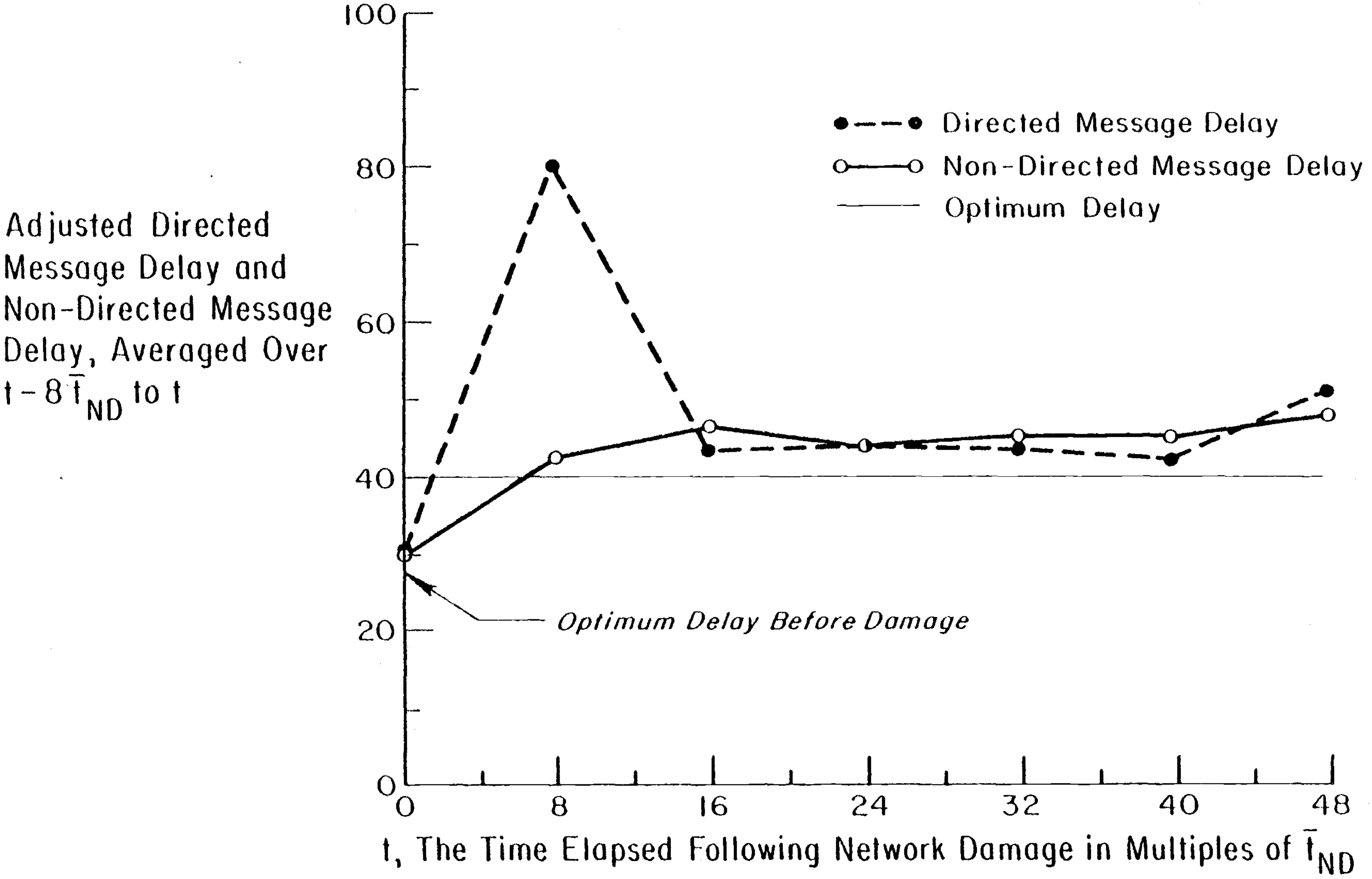


Figure 4.76. RRPSDV algorithm, three links destroyed.

the non-directed message delay, indicating that the directed messages are being routed over shortest paths.

Three Links Jammed

This algorithm was tested only for the case $T_{DV} = 4\overline{t}_{ND}$. The results for the three links jammed case are given in Figures 4.77, 4.78, and 4.79. Immediately following network damage, the algorithm initially appears to adapt very quickly. However, it had some difficulty completely adapting, which is evidenced by the additional messages being lost at $t = 32\overline{t}_{ND}$ and $44\overline{t}_{ND}$. Even with this trouble in completely adapting, the algorithm performed much better for this case of damage than did the Reciprocal Path Search Algorithm with Delay Vectors, although it did lose a few more messages than did the Rapid Reciprocal Path Search algorithm.

The adjusted directed message delay shows a gradual rise from its pre-damage value. Again, there is no large initial jump since so few messages were lost. The adjusted directed message delay then settles out to a value which is generally larger than the non-directed message delay, indicating that the Delay Vector Table restrictions are preventing some of the directed messages from taking shortest routes to their destinations. Although messages are still being lost after the initial adjustment period, so few are lost that the effect on the

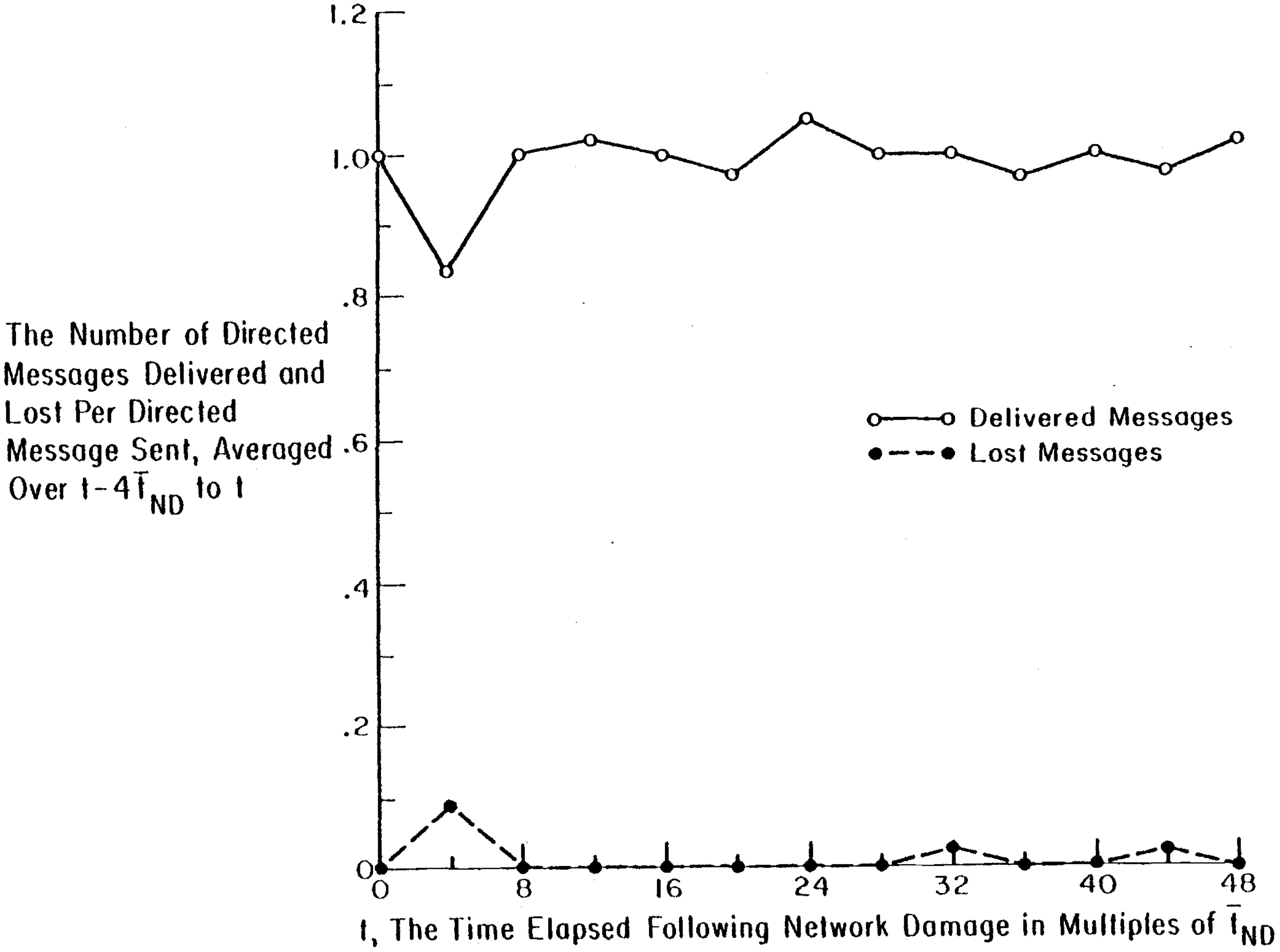


Figure 4.77. RRPSDV algorithm, three links jammed.

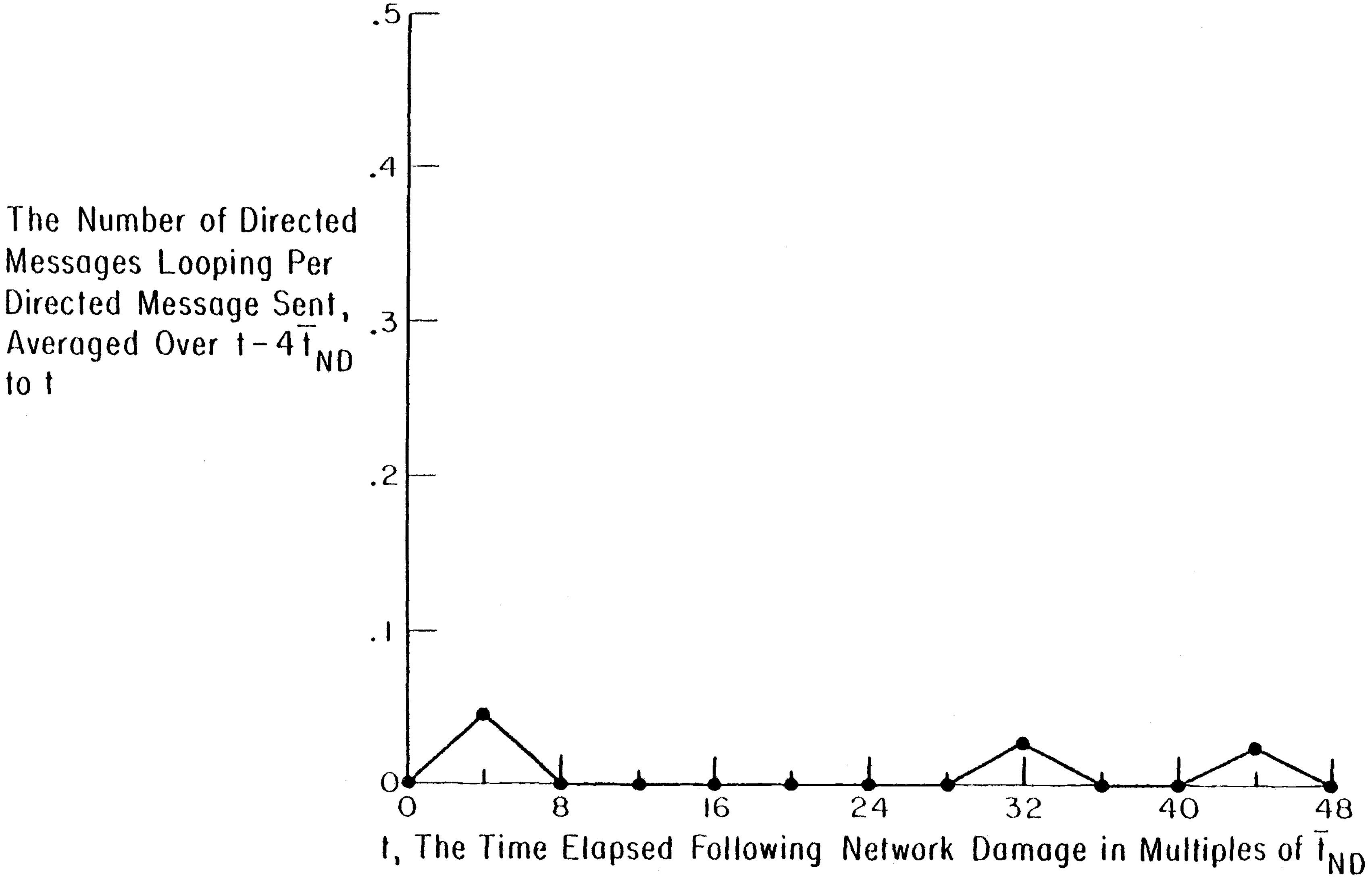


Figure 4.78. RRPSDV algorithm, three links jammed.

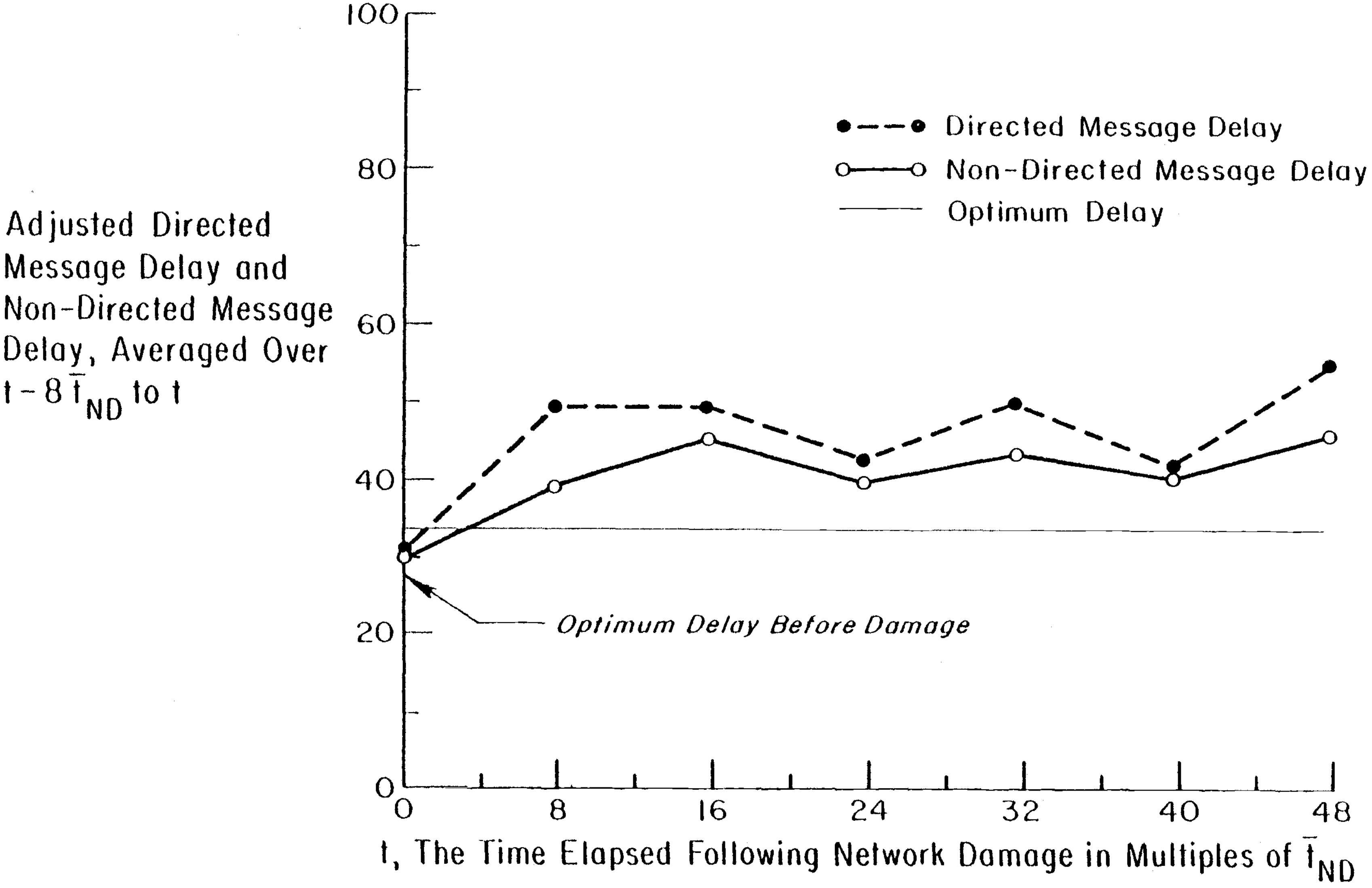


Figure 4.79. RRPSDV algorithm, three links jammed.

adjusted directed message delay of the penalty imposed for lost messages is not noticeable.

Appendix A contains plots of the simulation results when the restrictions on the use of the Delay Vector Tables are removed. Figures A. 1, A. 2, and A. 3 show the results when the source node restriction is removed. Figures A. 4, A. 5, and A. 6 show the performance when the ping-pong and source node restrictions are both removed. Figures A. 7, A. 8, and A. 9 show the results when the source node and ping-pong restrictions are removed, and the limiting constant, k_3, is increased from three to ten. It is evident that a progressive degredation in performance results as the restrictions are removed and k_3 is increased. It is important that this algorithm lose as few messages as possible since this is the only advantage which it has over the Reciprocal Path Search Algorithm with Delay Vectors. Without quick adaptability, this algorithm offers no improvement while being much more complex.

Six Links Jammed

This algorithm was again tested only for $T_{DV} = 4\bar{t}_{ND}$. The results for the six links jammed case are given in Figures 4.80, 4.81, and 4.82. These results again show an initial period when messages are lost and looping. A few messages are lost even after the looping has ceased, and even while messages are looping many more are being lost. This is a direct result of the algorithm preventing directed messages from being flooded out. Since this algorithm

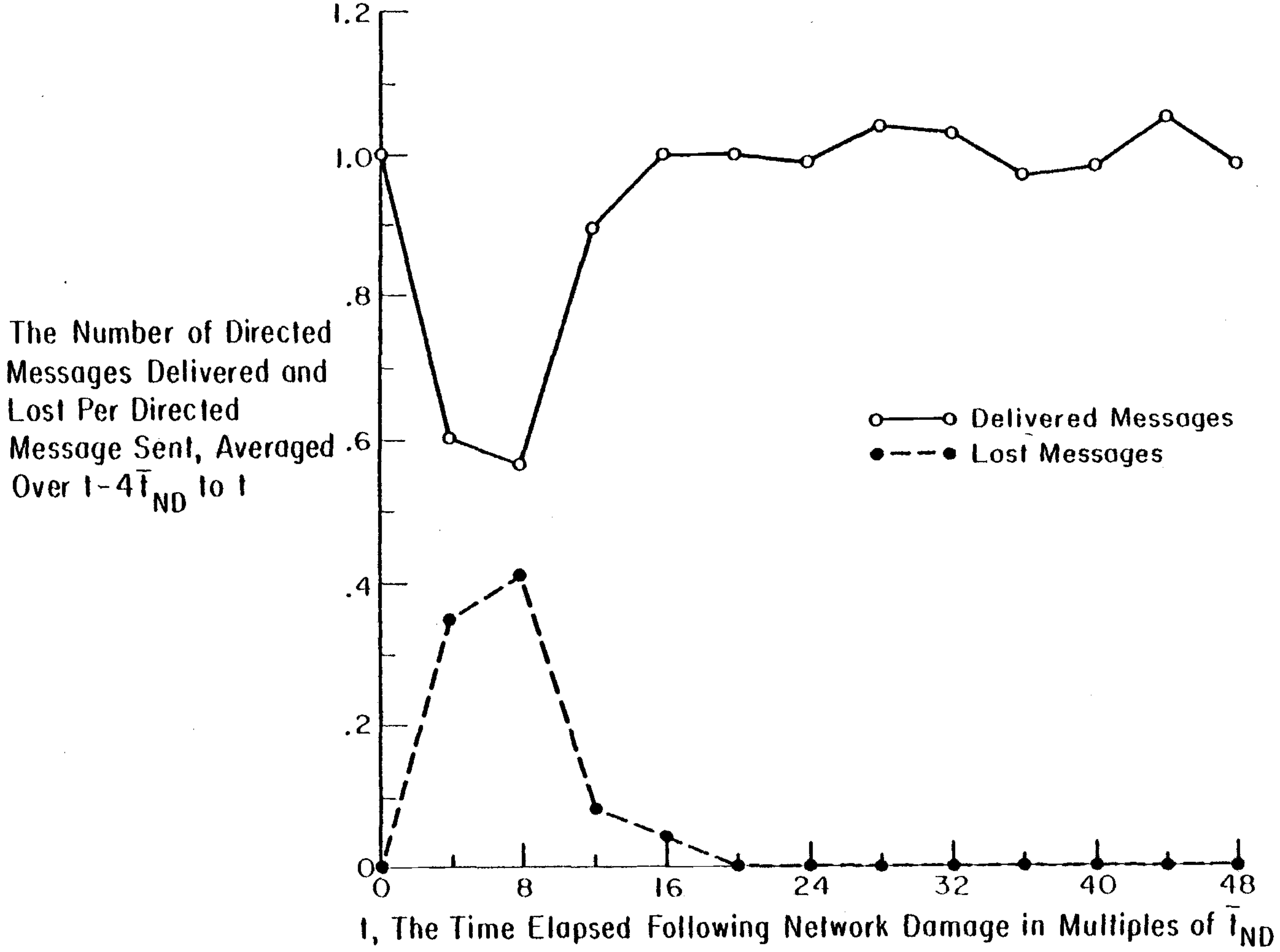


Figure 4.80. RRPSDV algorithm, six links jammed.

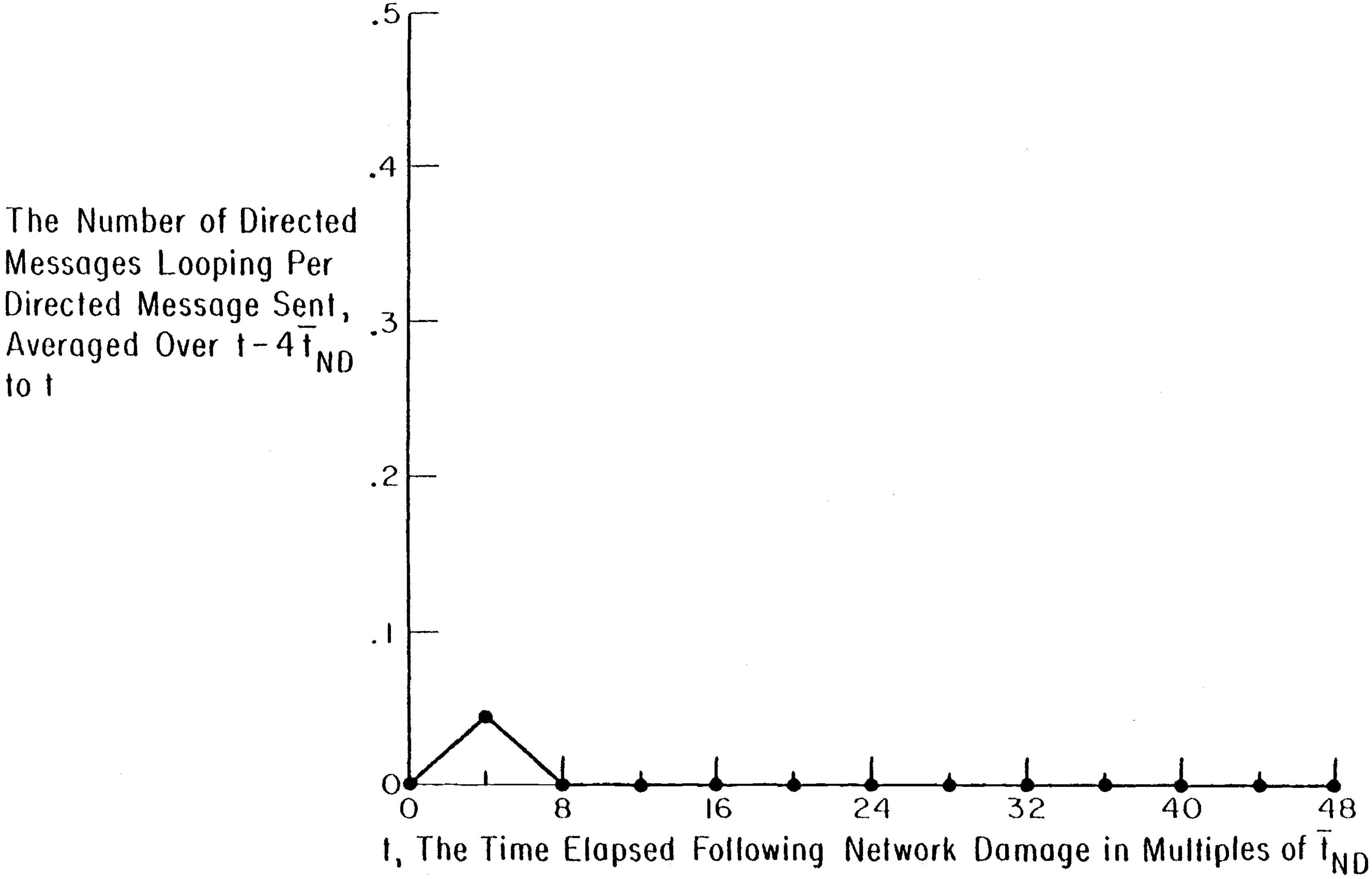


Figure 4.81. RRPSDV algorithm, six links jammed.

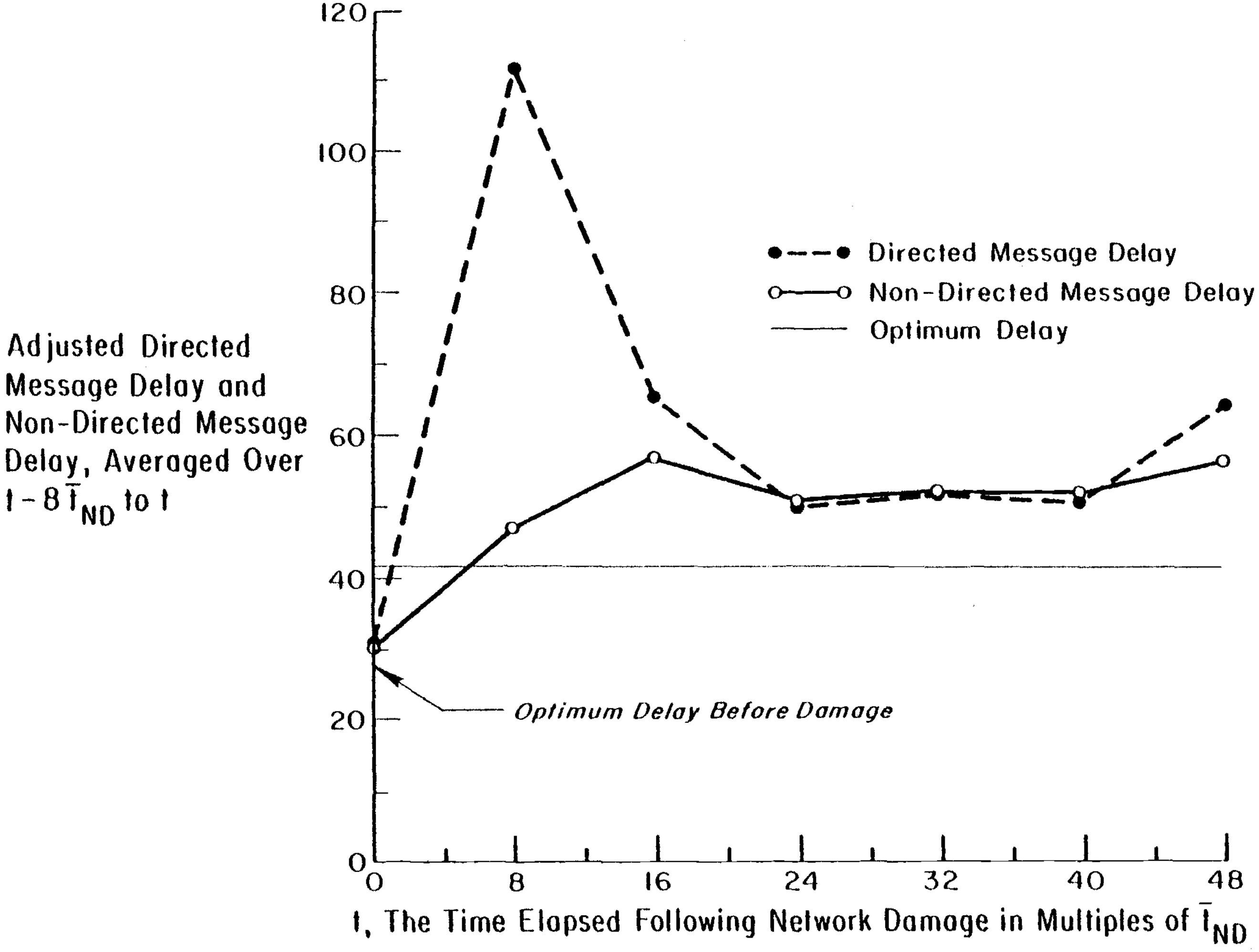


Figure 4.82. RRPSDV algorithm, six links jammed.

has the capability to route messages over both reciprocal and non-reciprocal paths, the node assumes that if information on a path is not available then no path exists. As was explained in detail previously, this is a valid conclusion in steady state and provides the nodes with the capability to detect when they have become disconnected from other network nodes. This prevents directed messages which have no hope of reaching their destination from being transmitted and thus needlessly congesting the network. It is conceivable that, in some situations, this capability can prevent the network from being saturated with directed messages which should never have been transmitted, and thus allow the network to continue functioning when it otherwise would not. This feature also slows down the algorithm's adaptation since it takes time for complete information on the non-reciprocal paths to reach all nodes.

The adjusted directed message delay shows an initial period when delay is much larger than the non-directed message delay due to the lost messages. After the algorithm adjusts, the directed message delay is essentially equal to the non-directed delay indicating that directed messages are being routed over shortest paths. However, a comparison to message delays for the Rapid Reciprocal Path Search algorithm, which does not use delay vectors, shows that the use of the delay vectors actually increased the message delays. Evidently, congestion caused by flooding out the delay vectors produced much more network congestion than was caused by the Rapid Reciprocal Path Search algorithm flooding out the directed messages themselves.

These results indicate that for the case of six links jammed, the use of delay vectors with the Rapid Reciprocal Path Search algorithm actually degrades the algorithm's performance. This does not mean, however, that the Rapid Reciprocal Path Search Algorithm with Delay Vectors would not produce improved performance for other cases of network damage. Its ability to determine when no path, reciprocal or non-reciprocal, exists to a node may prove to be very important when damage such that many node pairs are disconnected from each other occurs. Also, the Rapid Reciprocal Path Search Algorithm with Delay Vectors does have the ability to adapt quickly to some cases of network damage, such as the three links jammed case. Thus, it is a viable algorithm which adapts to all forms of network damage. Further research on refining the efficiency of the algorithm could reduce the extra traffic required enough to make its efficiency more comparable to that of the simpler Rapid Reciprocal Path Search algorithm.

Overhead Considerations

The only routing algorithm presented in this chapter which involves the use of essentially no overhead not otherwise necessary for transmission of data is the Simple Backwards Learning algorithm. This algorithm uses information contained in the non-directed messages to construct routing tables for directed message routing. These non-directed messages were generated and routed independently of the need for them to provide information for routing tables.

The Reciprocal Path Search algorithm forced extra non-directed messages to be generated in order to search out reciprocal paths when non-reciprocal forms of network damage occur. These extra non-directed message transmissions constitute overhead.

The Rapid Reciprocal Path Search algorithm not only forced extra non-directed messages to be generated in order to search out reciprocal paths, but it also flooded out directed messages when no reciprocal paths were found. Both the extra directed message transmissions caused by flooding and the extra non-directed message transmissions constitute overhead.

Delay vectors were used in conjunction with the Reciprocal Path Search algorithm and the Rapid Reciprocal Path Search algorithm in order to search out and use non-reciprocal paths for directed message routing. These delay vectors are special non-directed messages, and constitute more overhead.

The desired effect of the overhead is to increase the number of message deliveries and decrease message delay. These performance measures are of real importance. The overhead is important only in how it influences these performance measures. Thus, the algorithms presented in this chapter should be evaluated primarily in terms of their ultimate ability to deliver messages and to deliver them with a minimum of delay.

The above reservations notwithstanding, the total number of directed and non-directed message transmissions observed during equivalent simulation runs for each of the algorithms are listed in Tables 4. 9 and 4. 10. In general, an increase in message transmissions over those needed with the Simple Backwards Learning algorithm constitutes overhead. The Rapid Reciprocal Path Search algorithm also produces overhead by flooding directed messages, resulting in an increase in the directed message transmissions.

Summary

This chapter has presented and analyzed a new class of directed message routing algorithms for computer communication networks. These algorithms utilize new techniques for searching out and using both reciprocal and non-reciprocal paths for routing directed messages. The particular routing algorithms studied were the Simple Backwards Learning algorithm, the Reciprocal Path Search algorithm, the Rapid Reciprocal Path Search algorithm, the Reciprocal Path Search Algorithm with Delay Vectors, and the Rapid Reciprocal Path Search Algorithm with Delay Vectors.

Table 4.9. Total message transmissions for three links jammed case.

Algorithm	Total Non-Directed Message Transmissions	Total Directed Message Transmissions	Total Transmissions
SBL	11,546	2,710	14,256
RPS	13,237	2,958	16,195
RRPS	13,228	3,006	16,234
RPSDV			
$T_{DV} = 4\bar{t}_{ND}$	13,839	2,619	16,458
$T_{DV} = 2\bar{t}_{ND}$	14,379	2,612	16,991
RRPSDV	13,707	2,601	16,308

SBL = Simple Backwards Learning algorithm.
RPS = Reciprocal Path Search algorithm.
RRPS = Rapid Reciprocal Path Search algorithm.
RPSDV = Reciprocal Path Search Algorithm with Delay Vectors.
RRPSDV = Rapid Reciprocal Path Search Algorithm with Delay Vectors.

Table 4.10. Total message transmissions for the six links jammed case.

Algorithm	Total Non-Directed Message Transmissions	Total Directed Message Transmissions	Total Transmissions
SBL	11,633	2,340	13,973
RPS	11,614	2,503	14,117
RRPS	11,593	3,919	15,512
RPSDV			
$T_{DV} = 4\bar{t}_{ND}$	12,942	2,989	15,931
$T_{DV} = 2\bar{t}_{ND}$	14,097	3,063	17,160
RRPSDV	13,701	2,670	16,371

The Simple Backwards Learning algorithm was first tested to determine its steady state characteristics in an undamaged 13 node network, which were used to develop a performance model. (Since the other routing algorithms differ from Simple Backwards Learning only in how they adapt to network damage, their steady state performance characteristics in the undamaged 13 node network are identical to the performance characteristic of the Simple Backwards Learning algorithm. Thus, only a single performance model is needed to describe the steady state performance of all the routing algorithms in the undamaged 13 node GPSS model of the radar network.) The Simple Backwards Learning algorithm was then tested to determine its adaptability to various cases of network damage. It was determined from these tests that the Simple Backwards Learning algorithm adapts well to reciprocal forms of damage such as total communication link destruction, but it adapts very poorly to non-reciprocal forms of damage such as link jamming.

The Reciprocal Path Search algorithm was designed to adapt to non-reciprocal as well as reciprocal forms of damage. This algorithm searches out reciprocal paths which can be used to route directed messages around the non-reciprocal (jammed) links. The major disadvantage of this algorithm is that it requires the existence of reciprocal paths between nodes, and whenever such paths do not exist, the algorithm fails to adapt.

The Rapid Reciprocal Path Search algorithm is designed to adapt more quickly and more completely than does the Reciprocal Path Search algorithm. This algorithm also searches out reciprocal paths which can be used to route directed messages around the non-reciprocal links, and it accomplishes this much more quickly than does the Reciprocal Path Search algorithm. In addition, this algorithm is able to quickly detect the absence of a reciprocal path between nodes and is thus able to deliver directed messages in this situation by flooding them.

The Reciprocal Path Search Algorithm with Delay Vectors is designed to search out and use both reciprocal and non-reciprocal paths which can be used for directed message routing. Whenever any path exists between two nodes, reciprocal or non-reciprocal, this algorithm is able to locate and use such a path for directed message routing, and is thus able to avoid the directed message flooding which the Rapid Reciprocal Path Search algorithm uses. Also, this algorithm is able to detect when no path exists to a destination node, and thus is able to avoid congesting the network with directed messages which have no hope of being delivered.

The Rapid Reciprocal Path Search Algorithm with Delay Vectors also has the capability to locate and use both reciprocal and non-reciprocal paths for directed message routing, and to detect the absence of any path between nodes. However, it accomplishes these tasks generally more quickly than does the Reciprocal Path Search Algorithm with Delay Vectors.

The results of the adaptability tests show that for the cases of network damage considered in this chapter, the Rapid Reciprocal Path Search algorithm clearly performed the best. However, these cases of network damage are necessarily limited in scope, and were chosen only to provide a wide range of conditions under which to test the algorithms. The routing algorithms presented each possess differing capabilities, and any conclusions regarding which algorithm is best for a particular application depends primarily upon what the performance requirements are for that application. Further refinement of the algorithms presented could also alter some of the comparisons presented.

V. ANALYSIS OF COMBINED ROUTING AND ACKNOWLEDGEMENT PROTOCOLS

Retransmission of messages lost because of incorrect routing or because of errors induced by noisy communication channels is a basic ingredient of computer communication networks (24). The retransmission of a message by the sender may result from the reception of a negative acknowledgement (NAK), or because a positive acknowledgement (ACK) has not been received within a time out period. Positive acknowledgement with retransmission, using a time out at the sender, is a very common acknowledgement scheme (24). The primary benefit of using negative acknowledgements is to reduce redundant retransmissions. However, even when negative acknowledgements are used, positive acknowledgement retransmission, with an appropriately large time out at the sender, will still be required to ensure reliability in the event the negative acknowledgements or retransmissions are lost (25).

The acknowledgements may be sent on a link-by-link or on an end-to-end basis, or both. The end-to-end technique involves the transmission of acknowledgements from a message's destination node to its source node. The link-by-link technique involves the transmission of acknowledgements between the nodes at opposite ends of each communication link.

Only end-to-end acknowledgements will ensure that the lack of reception of a message at its destination due to incorrect routing will be recognized at the source node. Thus, end-to-end acknowledgements are required in any alternate routed packet switched computer communication network which guarantees delivery (this is not true for networks which use fixed routing). Link-by-link acknowledgements are not needed to guarantee directed message delivery, therefore their use needs to be justified on the basis of any performance improvements which they provide (26).

For acknowledgement protocols which use only positive acknowledgements with a time out at the sender, the length of the time out period has a profound effect on both message delay and message throughput (27). A small time out period minimizes average message delay since lost packets are retransmitted prompty. A large retransmission time out period tends to increase message throughput since no communication channel capacity is wasted on unnecessary retransmissions. Sunshine (27) has shown that for various distributions of message delay, an optimum value of time out period can be determined. This optimum value corresponds to a knee on a plot of throughput versus delay for varying time out periods. Larger time out periods than this optimal value will rapidly increase delay with little improvement in throughput. Smaller time out periods will rapidly decrease throughput, with little improvement in delay.

The primary acknowledgement scheme tested in this chapter is the use of positive end-to-end acknowledgements with retransmission if no acknowledgement is received within a specific time out period. Only the directed messages are assumed to need retransmission. The non-directed track reports are updated at frequent intervals, and the occasional loss of a track report is not critical. (The need for retransmission of some types of non-directed messages could possibly be required. However, retransmission does not appear to be necessary for track reports, so the problem of acknowledging and retransmitting non-directed messages will not be treated in this thesis.) Also, the flooding algorithm used to route non-directed messages is very reliable and non-directed messages are never lost because of routing failure (although some non-directed message copies may occasionally be lost as a result of transmission errors). Virtually all of the directed message routing algorithms, however, allowed some lost messages, and thus retransmission of the directed messages is the only way to ensure delivery, even in an error free transmission environment.

Link-by-link acknowledgements will not be used here since the one directional jamming of the communication links may block the reception of the acknowledgements. The end-to-end acknowledgements which will be used can take paths which avoid jamming.

Developing otpimum error control techniques is not the purpose of the performance tests of the combined routing and acknowledgement algorithms, but rather the tests are designed to show that the combined use of acknowledgements and the new routing algorithms produce a complete protocol which will reliably deliver all messages to their destinations.

Two of the new routing algorithms have been selected to be tested with the acknowledgements. These are the Rapid Reciprocal Path Search algorithm, and the Reciprocal Path Search Algorithm with Delay Vectors. Both of these routing algorithms completely adapted to all cases of network damage. The six links jammed case will be used to test the combined algorithms since this case of network damage will provide the most extreme tests of the protocols.

Reciprocal Path Search Algorithm with Delay Vectors and Acknowledgements

End-to-end acknowledgements with retransmission of directed messages after a time out period of T_{PA} was added to the Reciprocal Path Search Algorithm with Delay Vectors. The acknowledgements are assumed to fit into a packet of length ℓ_A which is one-tenth of ℓ_M, the packet length for regular non-directed and directed messages. The acknowledgements are given priority over all other message types so that they can quickly reach the directed message source nodes, and thus prevent excessive congestion caused by unnecessary retransmissions.

The model of the radar network is the same as was used previously in the tests of the routing algorithms. The channel capacities, message lengths, and various routing algorithm parameters used in the simulation tests are given in Table 5. 1.

Table 5. 1. Values of parameters used in the transient simulation tests of the combined routing and acknowledgement protocols.

Parameter	Value
k_1	0.25
k_2	0.25
k_3	10.0
$\bar{t}_D$	300.0
$\bar{t}_{ND}$	300.0
ℓ_M/C	10.0
ℓ_A/C	1.0

Transient Simulation Tests

Transient simulation tests were conducted on the combined acknowledgement and routing protocol to determine its adaptability to the case of six links jammed. The statistics used to evaluate the protocol's performance are as follows:

1. The number of directed messages delivered per directed message original sent.
2. The number of directed message copies (i.e., retransmissions) sent per directed message original sent.
3. The number of directed messages looping per directed message original sent.
4. The directed message and non-directed message delays.

Statistics 1, 2, and 3 are averaged over a window of width $4\bar{t}_{ND}$ and statistic 4 is averaged over a window of width $8\bar{t}_{ND}$. Thus any point on the curves of these statistics provides an indication of the near term performance of the protocol.

The message delivery and looping message statistics are the same as were used previously on the tests of the routing algorithms alone. However, the statistic on directed message copies is unique to the tests of the combined routing and acknowledgement protocol. The statistic on the number of copies sent out is intended to replace the statistic on lost messages used in Chapter IV. When acknowledgements are included in the model, messages are not really lost in a strict sense since copies of all messages are stored at their source nodes. Assuming that sufficient time is allowed for acknowledgements to be returned to the source node, retransmissions of directed messages in excess of the number required as a result of transmission errors are an indication of the routing algorithm's inability to deliver messages.

The statistic on directed and non-directed message delay is essentially the same as in Chapter IV, except for one change. Previously, a penalty was included for each lost message. This penalty is no longer needed since undelivered messages are always retransmitted after the time out period, and after one or several retransmissions, the message will eventually reach its destination and be included in the average.

Figures 5.1, 5.2, 5.3, 5.4, and 5.5 give the simulation results for the six links jammed case with $T_{PA} = 500$. These runs are for a case with 100 percent effective jamming and no transmission errors otherwise, just as in Chapter IV.

These results show a period immediately following the network damage when message delivery is low and messages loop. This period lasts for about $12\bar{t}_{ND}$. Once the routing algorithm has adapted to the damage, there is a period of very large message delivery rate. This large delivery rate is a result of large numbers of retransmitted messages finally reaching their destinations. This is a "catch up" period during which the protocol is reducing the backlog of undelivered messages which accumulated during the preceding period of looping and poor message delivery. Once the catch up period is completed, the delivery rate levels out, and message retransmissions cease,

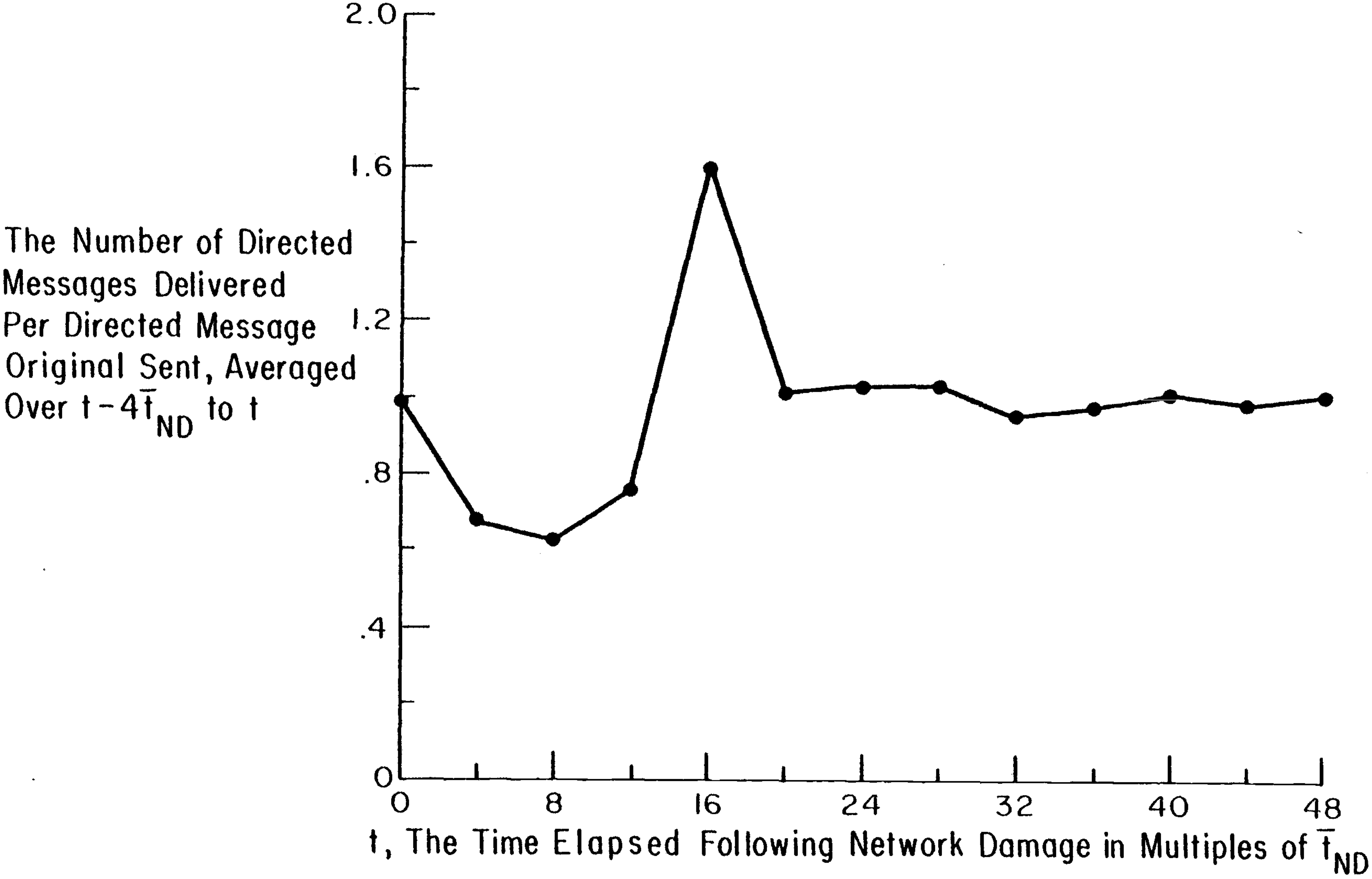


Figure 5.1. RPSDV algorithm with ACK's, $T_{PA} = 500$.

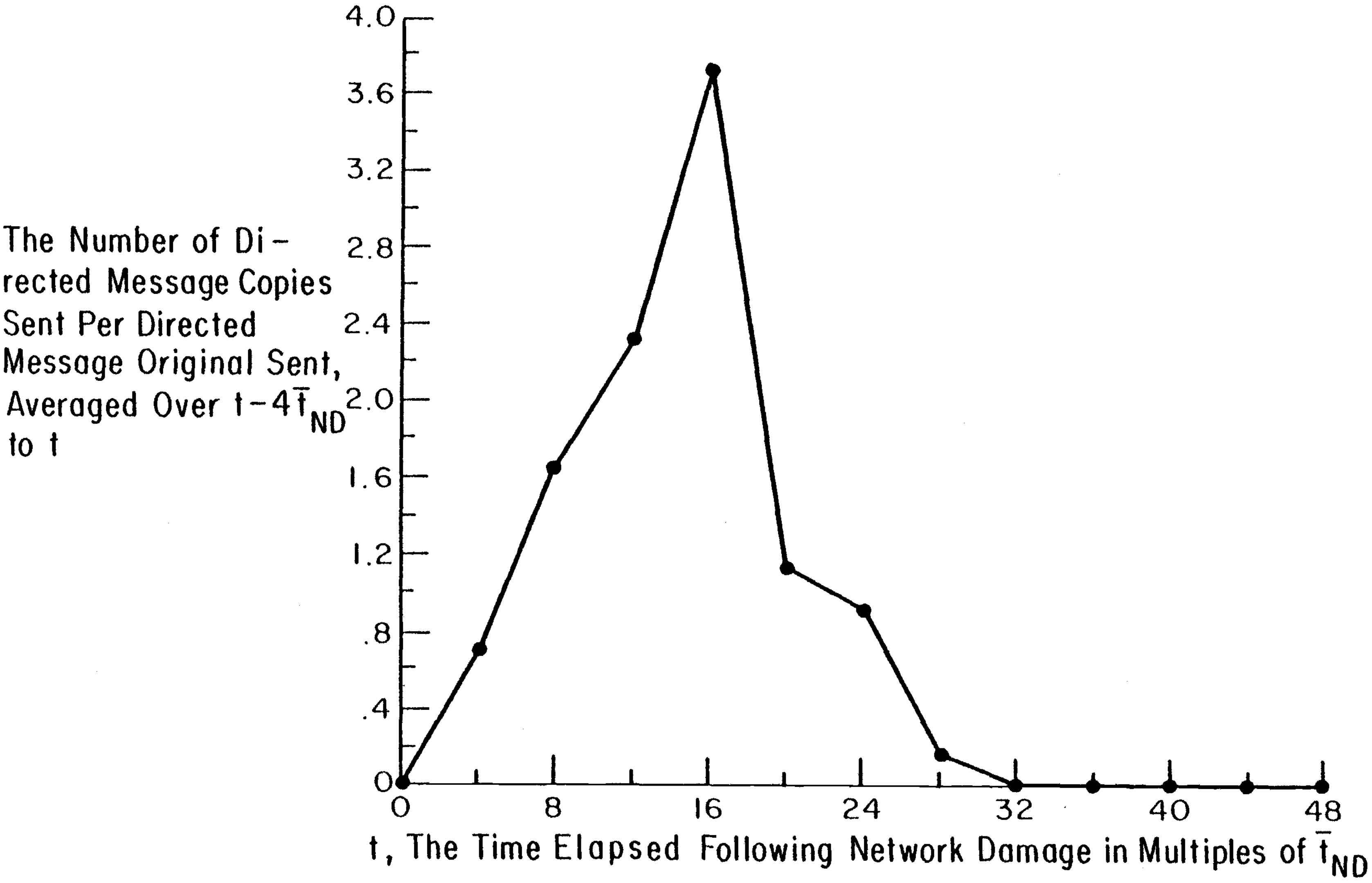


Figure 5.2. RPSDV algorithm with ACK's, $T_{PA} = 500$.

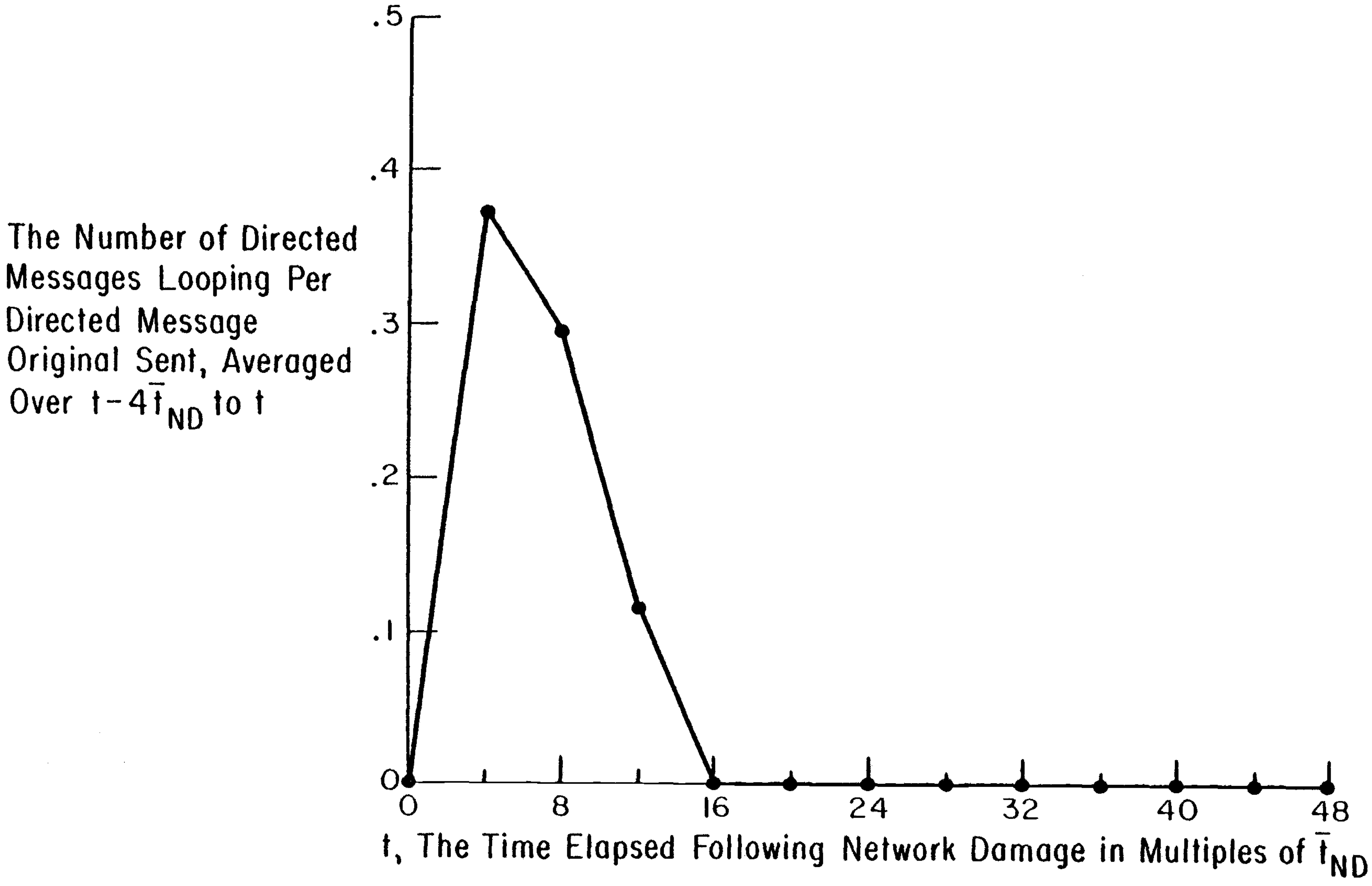


Figure 5.3. RPSDV algorithm with ACK's, $T_{PA} = 500$.

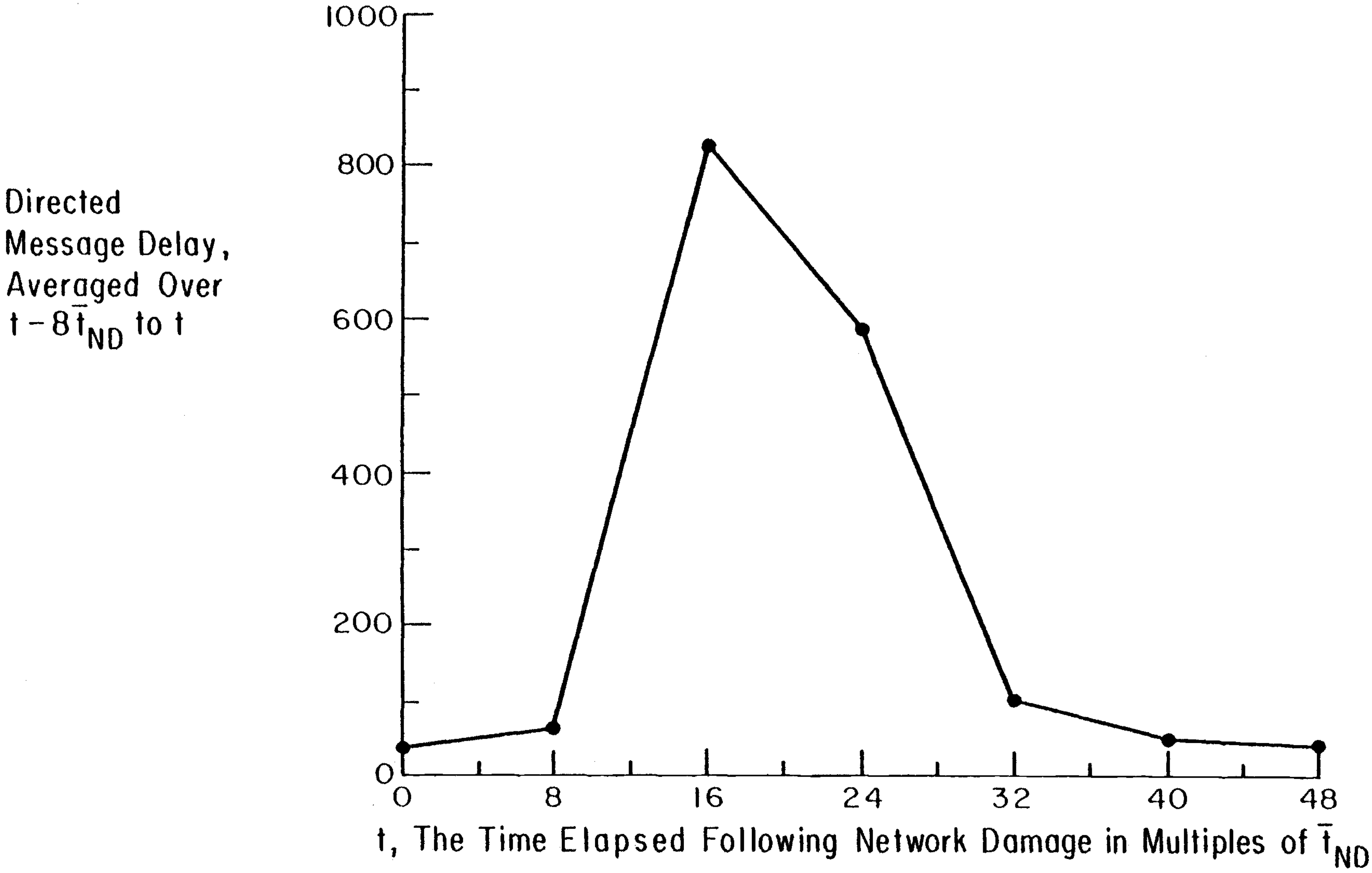


Figure 5.4. RPSDV algorithm with ACK's, T_{PA} = 500.

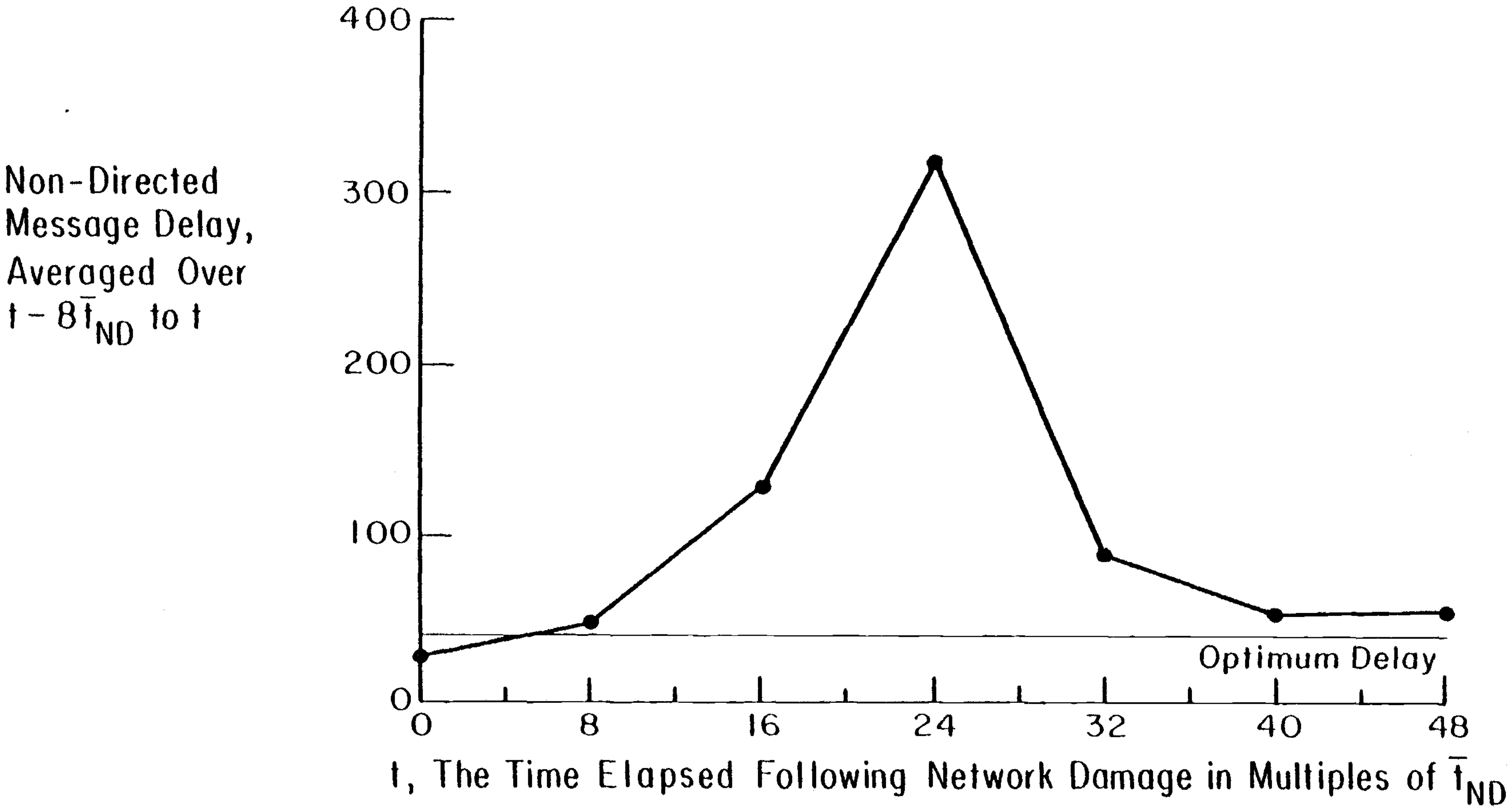


Figure 5.5. RPSDV algorithm with ACK's, $T_{PA} = 500$.

indicating that all the original message transmissions are reaching their destinations.

The directed message delay shows an initial low value while the routing algorithm is still adapting. Once the directed message copies (retransmissions) start arriving, however, the directed message delay takes a large jump which is reflective of the time out period during which these copies had to wait before they were transmitted. The non-directed message delay also shows an increase during this catch up period. The increase in non-directed message delay is reflective of the network congestion resulting from the large number of directed message copies which were transmitted and still on their way to their destinations. Once the copies are completely delivered, the congestion eases, and both the directed and non-directed message delays decrease.

Preliminary simulation tests indicated that without giving the acknowledgements priority, the large amount of congestion which exists during the catch up period can seriously delay their delivery. This impairment in acknowledgement delivery can result in unnecessary directed message retransmissions, further congesting the network, and possibly leading to an unstable condition and saturation of the network with directed message copies.

For the next tests, the GPSS simulation model of the network was improved by adding noise to the communication links and by more accurately modeling the jammed links. Noise was added to the communication links by decreasing the probability of a successful transmission over a good communication link from 100 percent to 95 percent. Improvement in modeling the jammed links was accomplished by allowing 50 percent of the messages transmitted across them to be successfully received. It is expected that even when a link is being jammed, a significant fraction of the messages will be successfully transmitted across it, with the 50 percent fraction used here chosen to severely test the algorithm.

The combined routing and acknowledgement protocols were next tested in the improved model for the case of six links jammed, again with $T_{PA} = 500$. Figures 5.6, 5.7, 5.8, 5.9, and 5.10 give the results for this test.

These results show some similarity to the previous tests on the original model which used 100 percent effective jamming and no transmission errors otherwise. There is an initial period of looping messages and resultant low delivery, followed by a catch up period. The statistics on the directed message retransmissions show a build up of retransmissions during the period of low message delivery, and then a tapering off as the backlog of undelivered messages is reduced. It appears that more time is required to substantially reduce the backlog than in the tests on the original model of the radar network due to

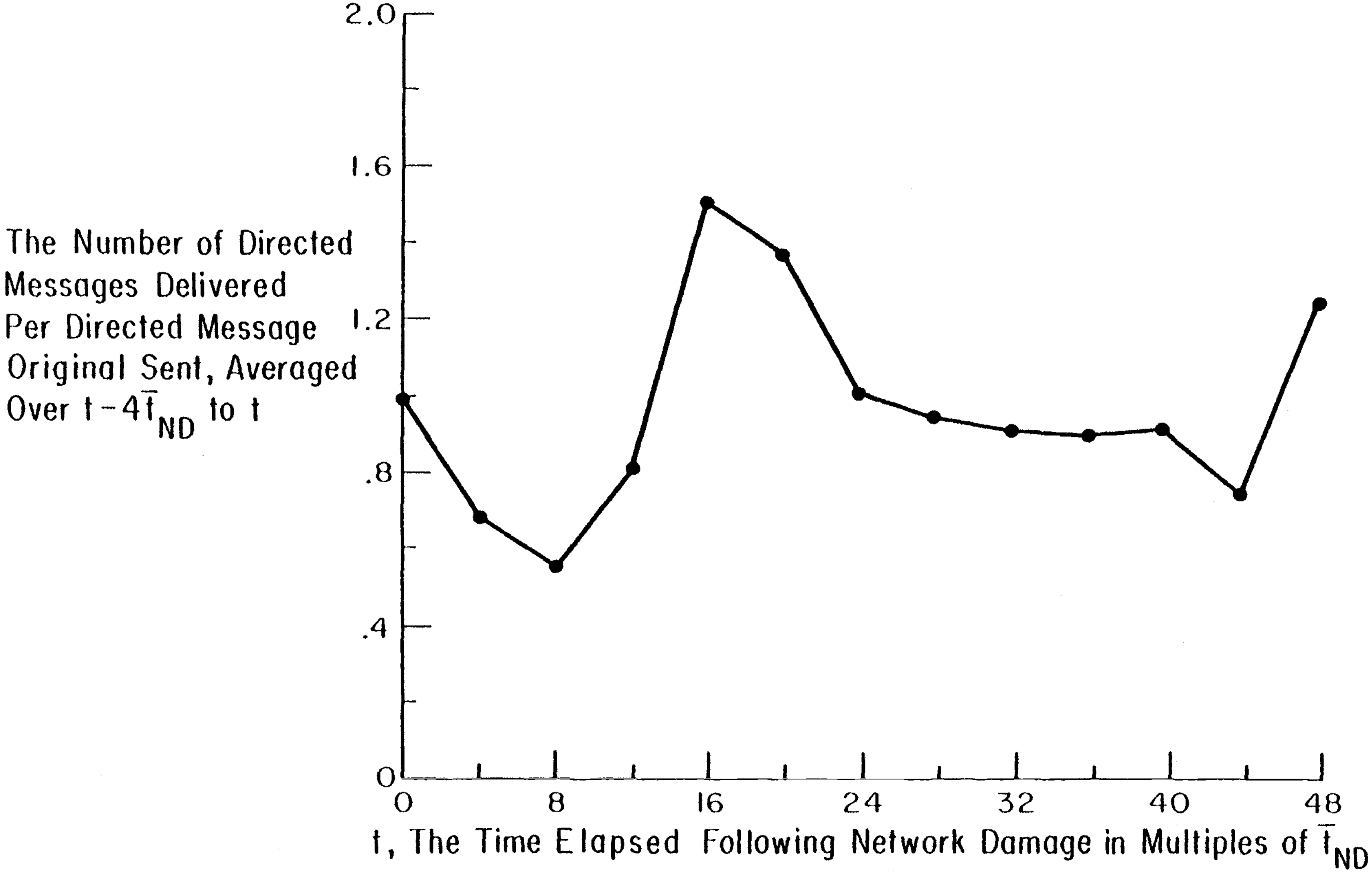


Figure 5. 6. RPSDV algorithm with ACK's, T_{PA} = 500, improved model.

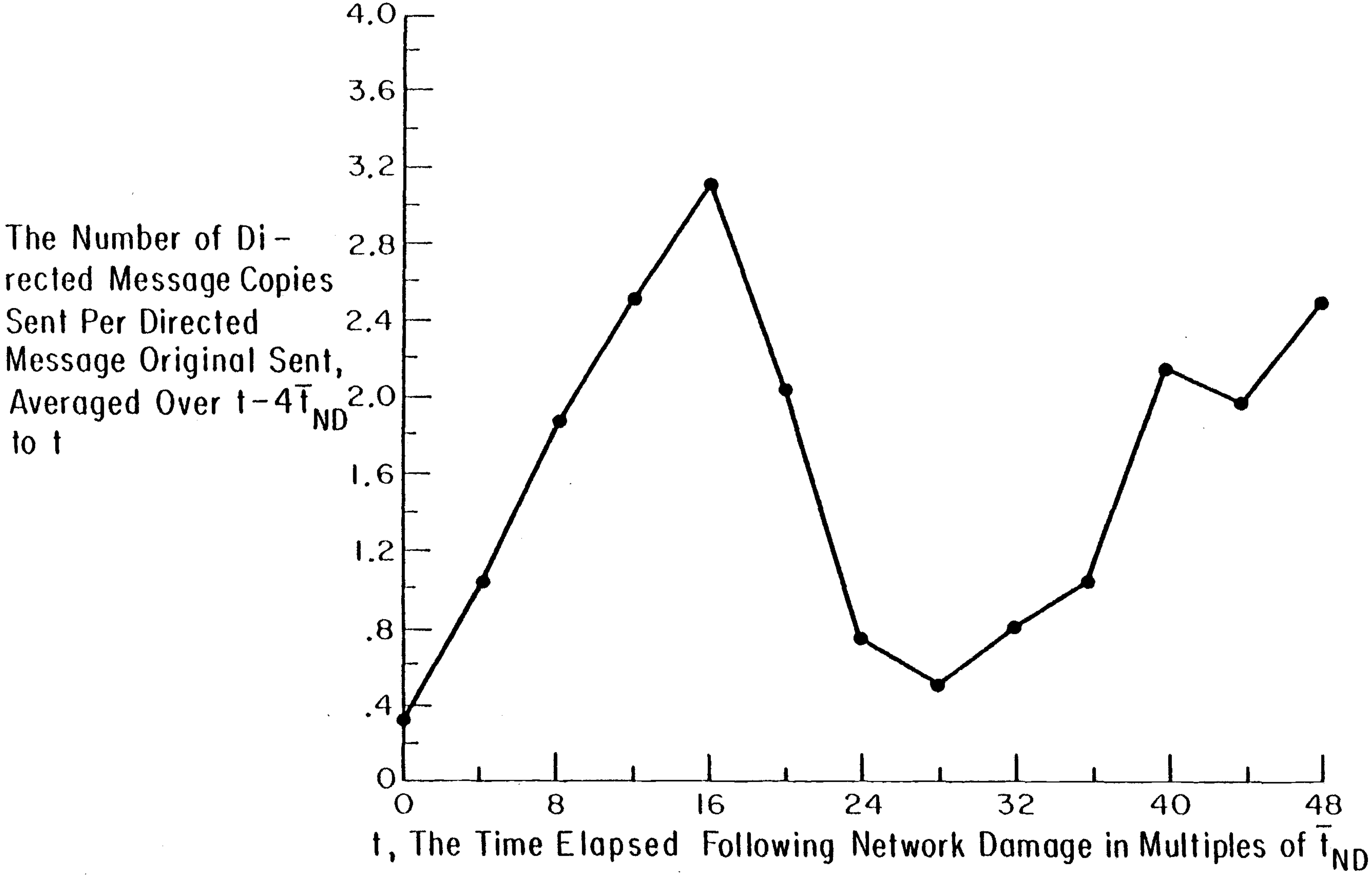


Figure 5.7. RPSDV algorithm with ACK's, T_{PA} = 500, improved model.

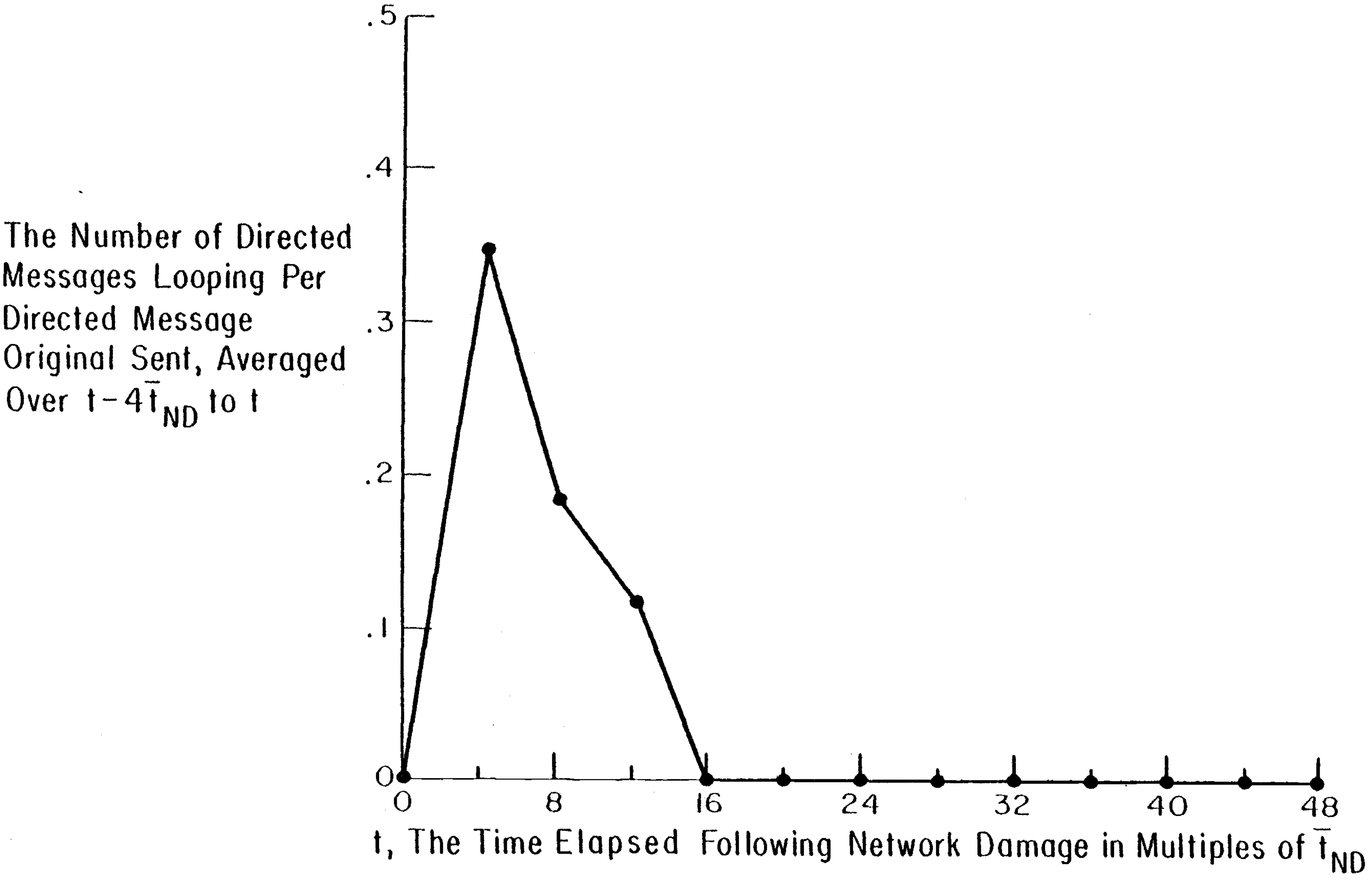


Figure 5.8. RPSDV algorithm with ACK's, $T_{PA} = 500$, improved model.

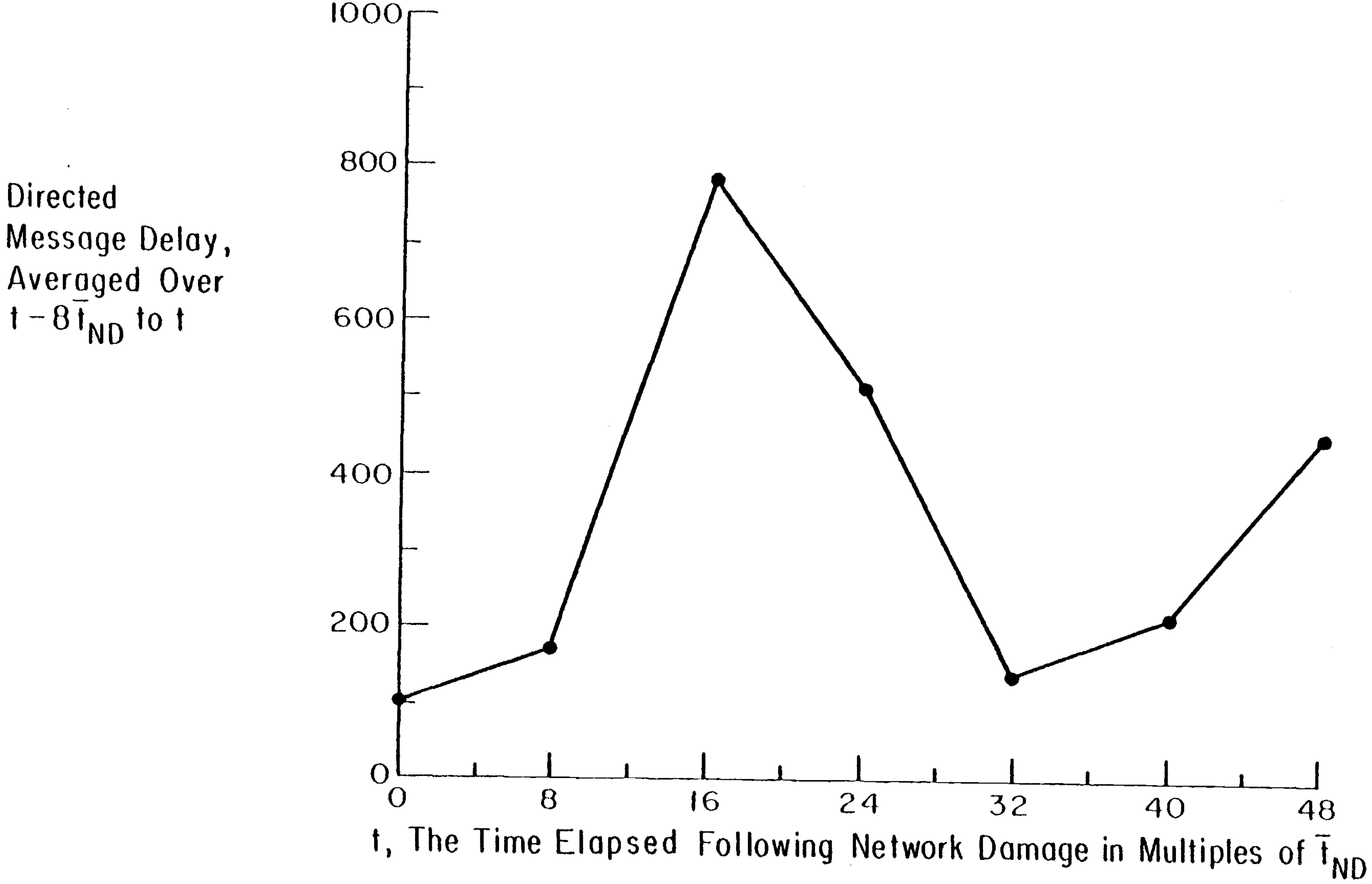


Figure 5.9. RPSDV algorithm with ACK's, $T_{PA} = 500$, improved model.

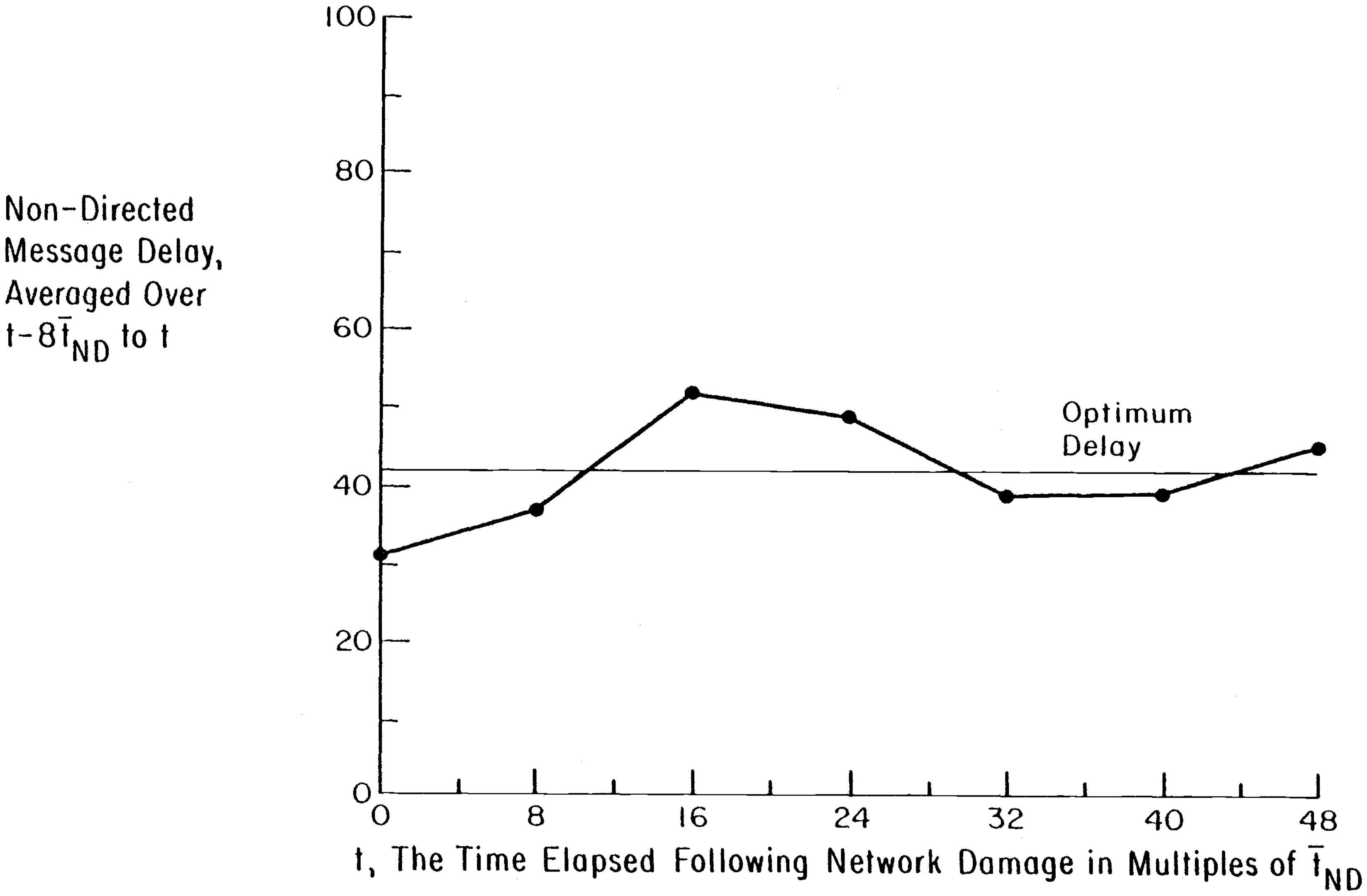


Figure 5.10. RPSDV algorithm with ACK's, T_{PA} = 500, improved model.

the additional loss of directed messages caused by the communication channel noise.

The directed message retransmissions never cease even after the initial large backlog of undelivered messages is substantially reduced. This is a result of two factors. First, directed messages are continuously being lost due to the communication channel noise. Second, the channel noise also prevents some of the nodes from receiving a few of the delay vectors, thus preventing some of the Delay Vector Table entries from being updated before they become dead. Since the six links jammed case, which is being modeled here, has several node pairs which have only non-reciprocal paths between them, the dead Delay Vector Table entries result in some nodes concluding that no path exists to some destinations, and thus discarding messages directed to these nodes. These Delay Vector Table entries do not remain dead, however. Delay vectors are transmitted on a regular basis, and with each round of delay vector transmissions, most of the dead Delay Vector Table entries are updated, allowing many messages to be retransmitted and delivered to their destinations. However, with each round of delay vector transmissions some delay vectors are lost, causing some previously alive entries to become dead in addition to the opposite effect of previously dead entries becoming alive. The result of the Delay Vector Table entries randomly becoming dead and alive is that throughout the simulation run messages are discarded and then later retransmitted. The simulation runs do indicate that the algorithm does reliably ensure that all directed messages eventually reach their destination.

With the rather small network modeled by the GPSS program, the loss of delay vectors due to noisy communication links can be expected on a regular basis. This might not be a problem in a larger network which has many more paths, since the loss of a few copies over noisy communication channels may be compensated for by the many copies which will be produced over multiple paths.

The two factors mentioned above, the directed messages being lost due to both noise and dead Delay Vector Table entries, cause large flucutations in the number of directed messages delivered and retransmitted throughout the run. This is especially evident late in the run when a large number of Delay Vector Table entries suddenly became dead and the delivered message rate took a large dip. The following round of delay vector transmissions was able to correct this situation and again reduce the backlog of undelivered directed messages through a large number of retransmissions.

Turning to the directed message delay, the results show that this statistic again has an initially low value during the period of looping messages and resultant low delivery. Once the routing algorithm adapts, and large numbers of directed message copies (retransmissions) start arriving at their destinations, the directed message delay again takes a large jump which reflects the time out period which the copies had to wait before their transmissions. The non-directed message delay also increases during this catch-up period due to the congestion caused by the large influx of directed message retransmissions. However, it did not increase as much as it did in the original model. Evidently, the continuous loss of directed messages caused by the noisy communication channels and dead Delay Vector Table entries helped keep down the network congestion and thus improved the non-directed message delay. It is also interesting to note that following the initial catch-up period, the non-directed message delay decreases and then fluctuates about the optimum delay line. The optimum delay line represents a steady state value, and the non-directed message delay obtained from the simulation results is transient in nature. Therefore, for uncongested networks, the non-directed message delay can be expected to fluctuate above and below this steady state optimum.

The directed message delay started to increase again near the end of the simulation run as a result of a large number of retransmissions which were eliminating the backlog of undelivered messages caused by the unusually large loss of the delay vectors.

One final simulation run was conducted on the improved model with the Reciprocal Path Search Algorithm with Delay Vectors and acknowledgements. For this test, the time out period T_{PA}, was increased from 500 to 1000. The results for this case are given in Figures 5.11, 5.12, 5.13, 5.14, and 5.15.

These results indicate that, as should be expected, the increase in time out period resulted in an increase in directed message delay while reducing network congestion.

The results again show a period of message looping and resultant poor delivery rate. The statistics on directed message retransmissions show a build up while directed messages are being lost, and then a tapering off as the backlog of undelivered messages is reduced. Again, there are fluctuations in the statistics on messages delivered and retransmitted which are caused by the fluctuations in lost messages as a result of the communication channel noise and the dead Delay Vector Table entries.

The effect of the increased time out period is especially evident in the statistics on message delays. The directed message delay was initially low during the period of looping messages and poor delivery. During the catch up period, the directed message delay again took a jump as a result of the time out period which the directed message copies had to wait before being transmitted. This jump in delay was much larger than was previously observed, due to the increased T_{PA}. After the original backlog of undelivered messages was reduced, the average directed message delay decreased, but it was

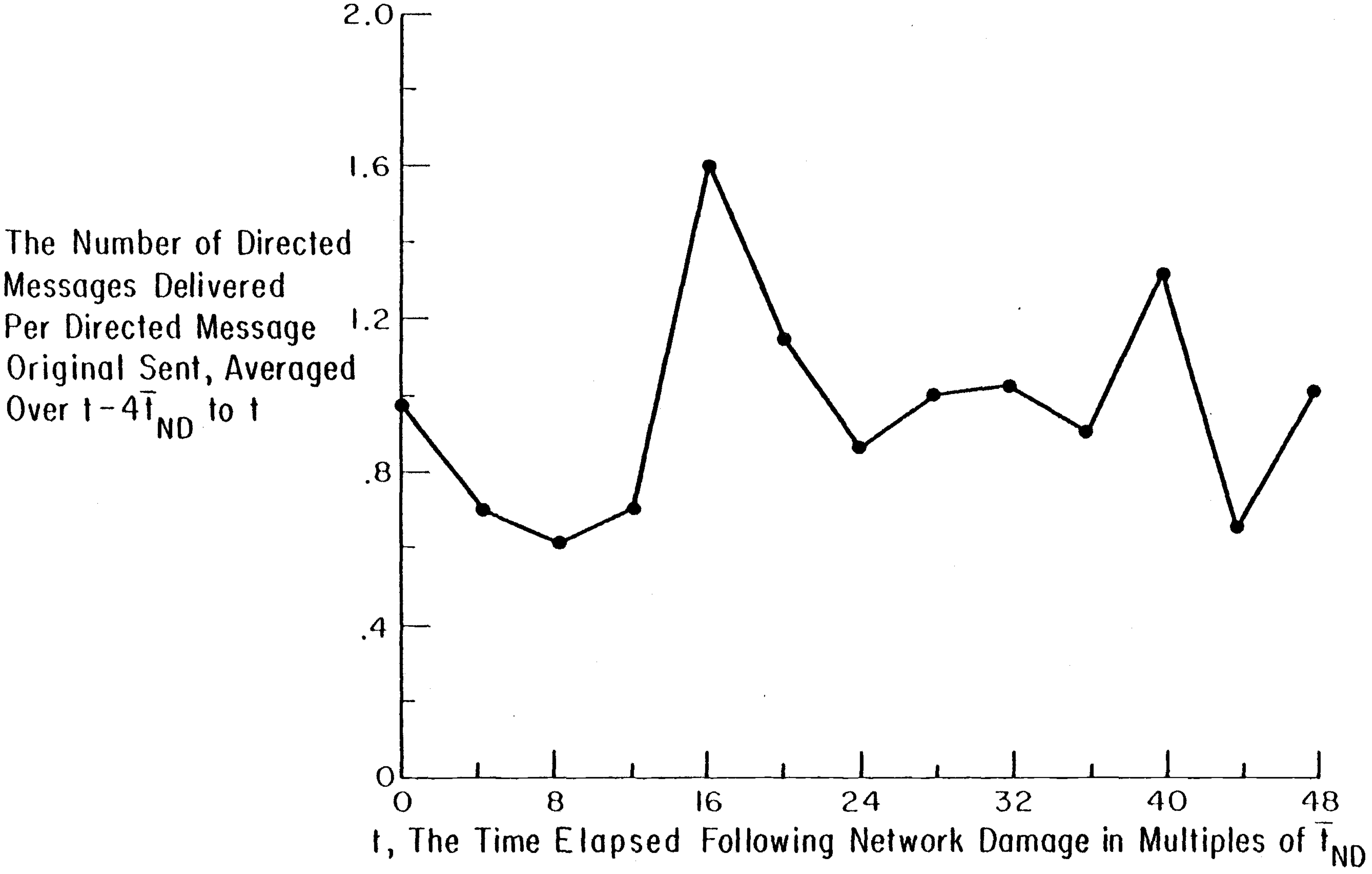


Figure 5.11. RPSDV algorithm with ACK's, T_{PA} = 1000, improved model.

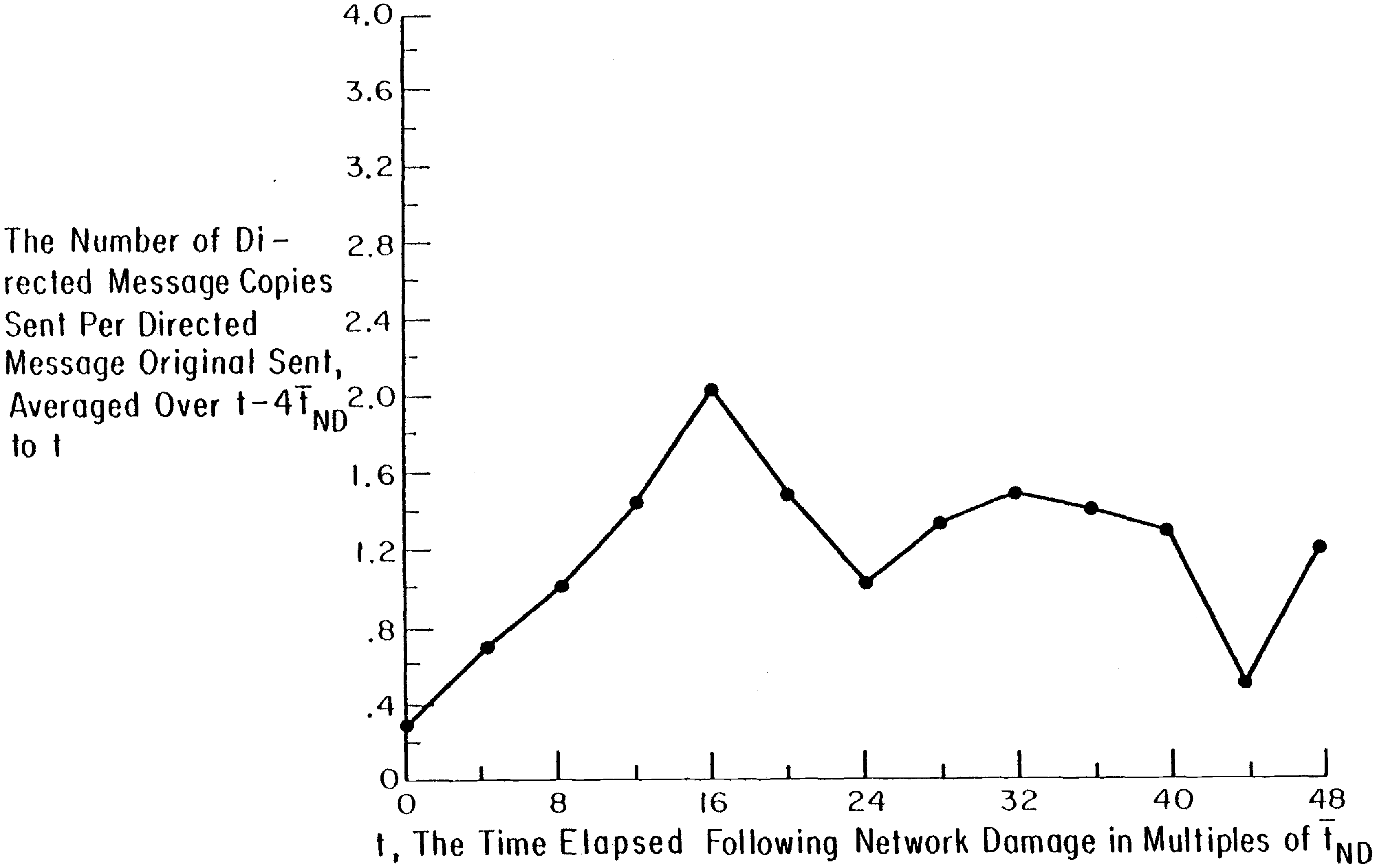


Figure 5. 12. RPSDV algorithm with ACK's, T_{PA} = 1000, improved model.

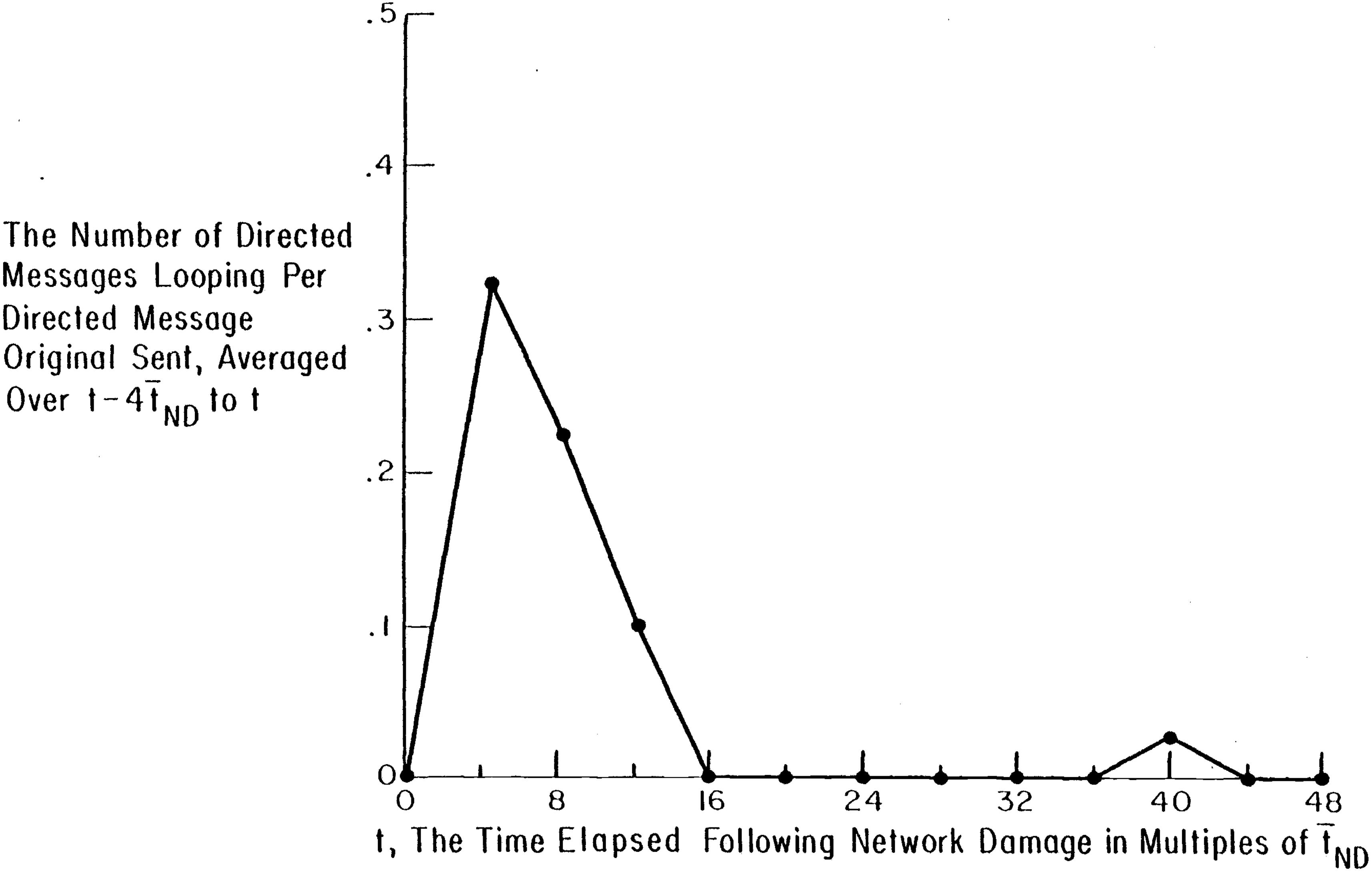


Figure 5.13. RPSDV algorithm with ACK's, T_{PA} = 1000, improved model.

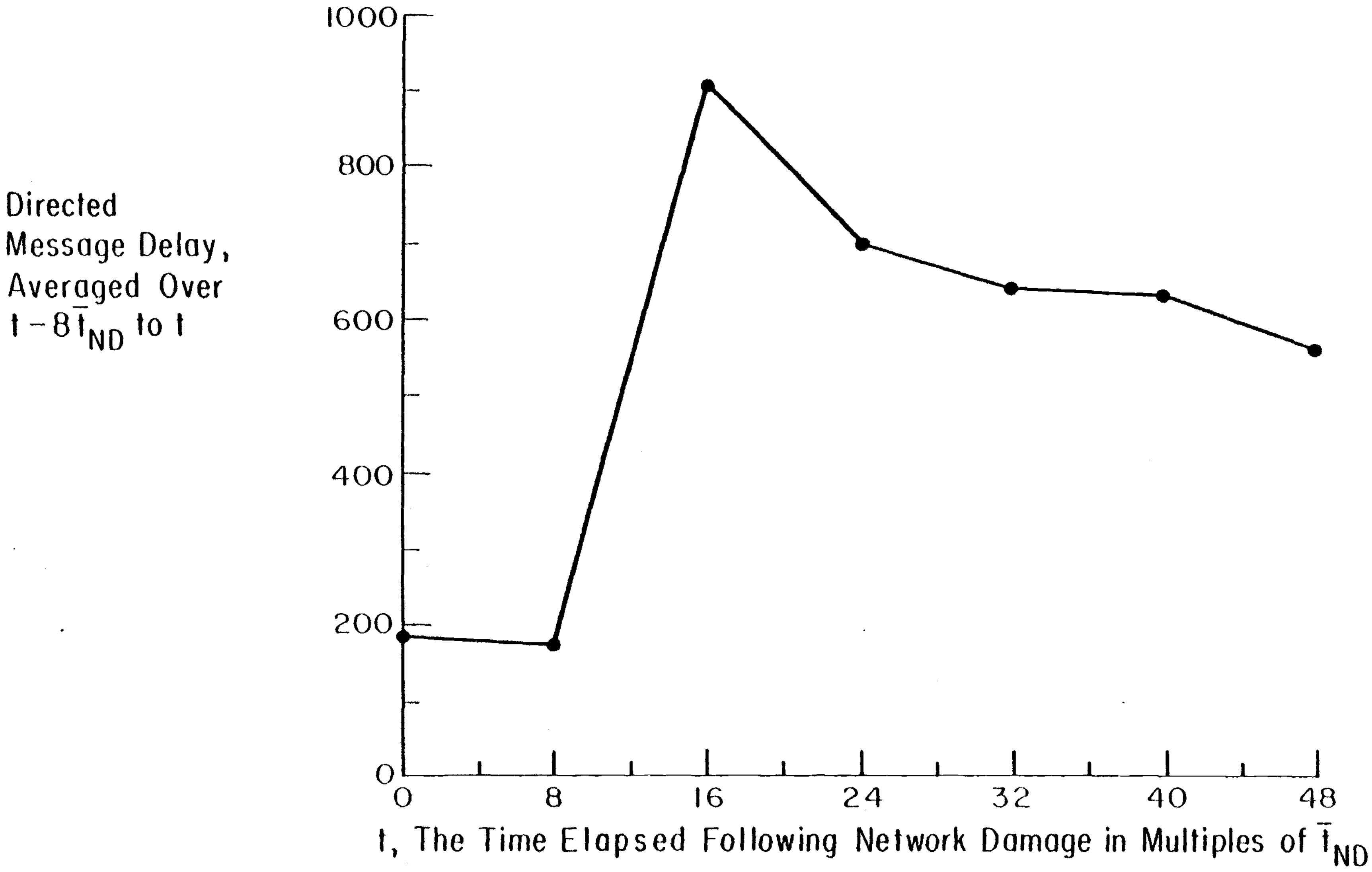


Figure 5. 14. RPSDV algorithm with ACK's, T_{PA} = 1000, improved model.

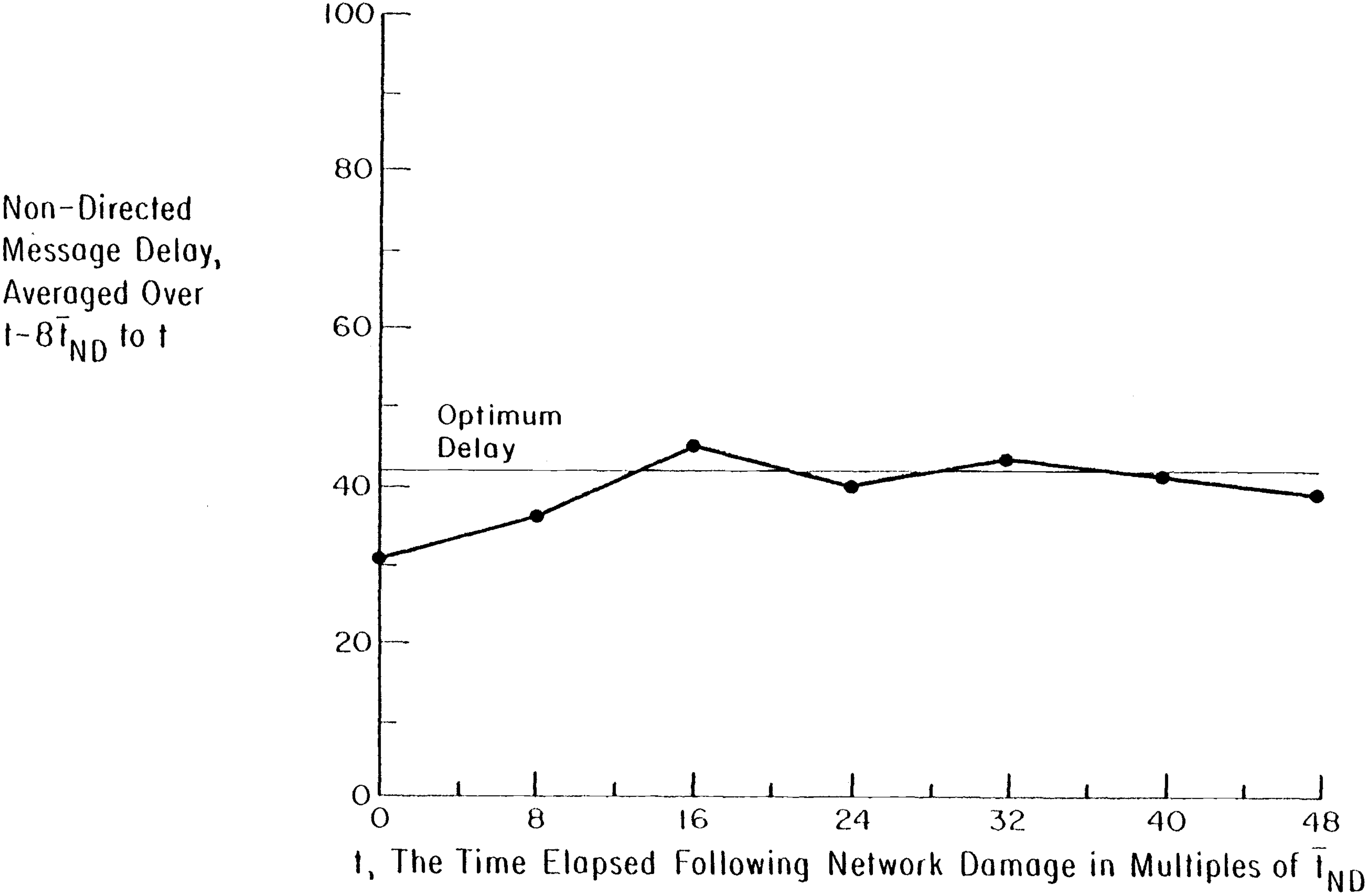


Figure 5.15. RPSDV algorithm with ACK's, $T_{PA} = 1000$, improved model.

still much greater than that in the previous run with the smaller time out period.

The non-directed message delay was generally lower than in the run with $T_{PA} = 500$. The increased time out period resulted in a decrease in network congestion and corresponding decrease in non-directed message delay.

Rapid Reciprocal Path Search Algorithm with Acknowledgements

End-to-end acknowledgements with retransmission of directed messages after a time out period of T_{PA} were also added to the Rapid Reciprocal Path Search algorithm. The acknowledgements were again assumed to fit into a packet of length of ℓ_A which is one tenth of ℓ_M. The acknowledgements were again given priority over all other message types. The various simulation and protocol parameters remain the same as given in Table 5.1.

Transient Simulation Tests

The protocol was tested using the GPSS model of the radar network which was improved by adding noise to the communication links and by more accurately modeling the jammed links. Noise was added to the communication links by decreasing the probability of a successful transmission over a good communication link from 100 percent to 95 percent, and jammed links are modeled by allowing 50 percent of the messages transmitted across them to be successfully received. The results of the tests on the Rapid Reciprocal Path Search algorithm with acknowledgements are given in Figures 5. 16, 5. 17, 5. 18, 5. 19, and 5. 20.

These results show a dramatic improvement in performance when compared to the test results for the Reciprocal Path Search Algorithm with Delay Vectors and acknowledgements. The Rapid Reciprocal Path Search routing algorithm adapts to the network damage so quickly that the acknowledgement portion of the protocol is hardly tested at all. This is why emphasis was placed on the results of testing the acknowledgements with the Reciprocal Path Search Algorithm with Delay Vectors discussed above.

The statistic on message delivery showed no significant drop following the network damage. Similarly, the statistics on directed message retransmissions showed no large peak while the few messages which were originally lost while the routing tables were adapting, were retransmitted. In fact, the directed message retransmissions remained extremely low during the entire simulation run. Once the routing tables had adapted, very few messages were lost. The Rapid Reciprocal Path Search algorithm does not use delay vectors, so it does not have the problems with dead routing table entries which the previous algorithm had.

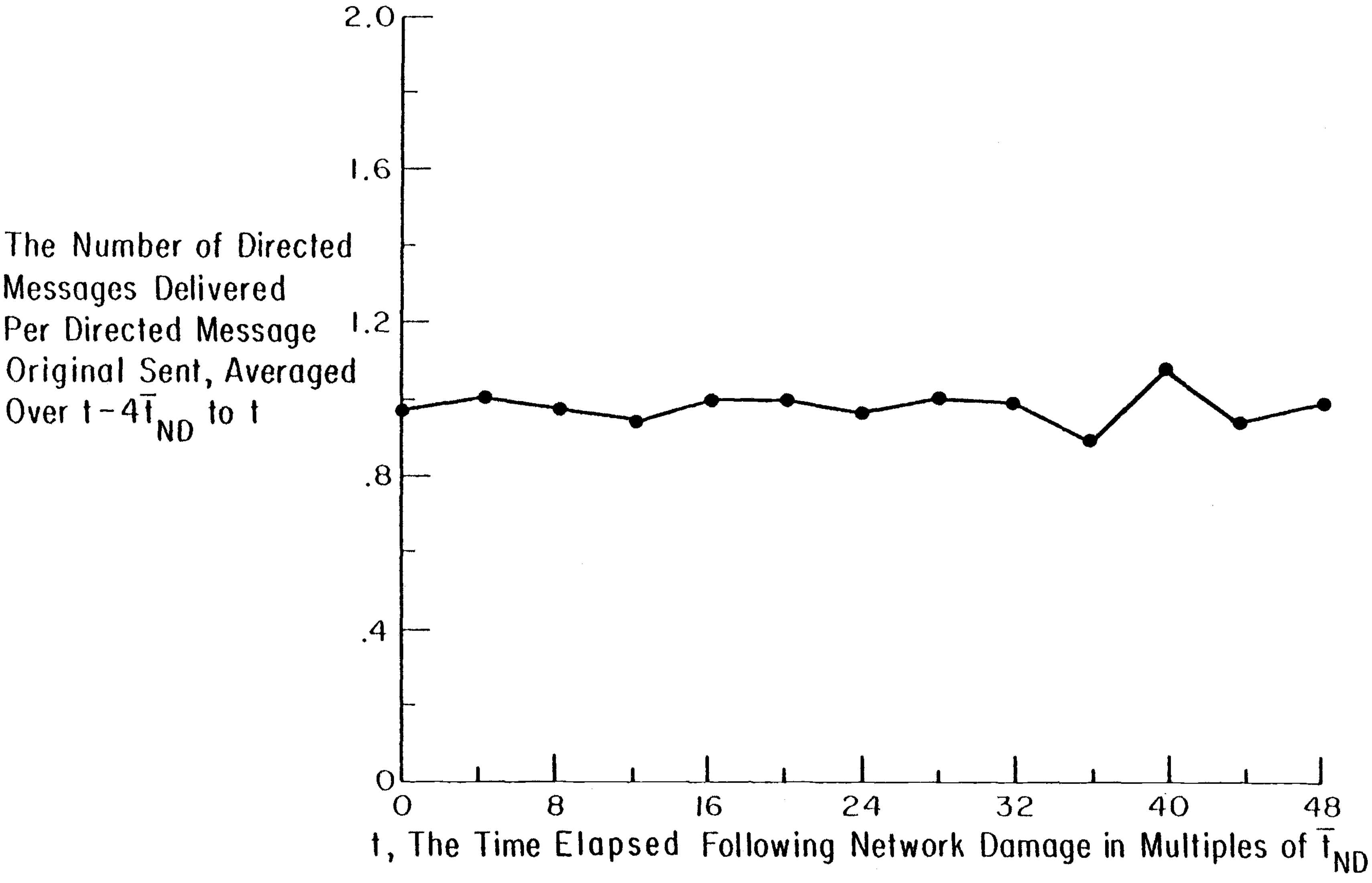


Figure 5.16. RRPS algorithm with ACK's, T_{PA} = 500, improved model.

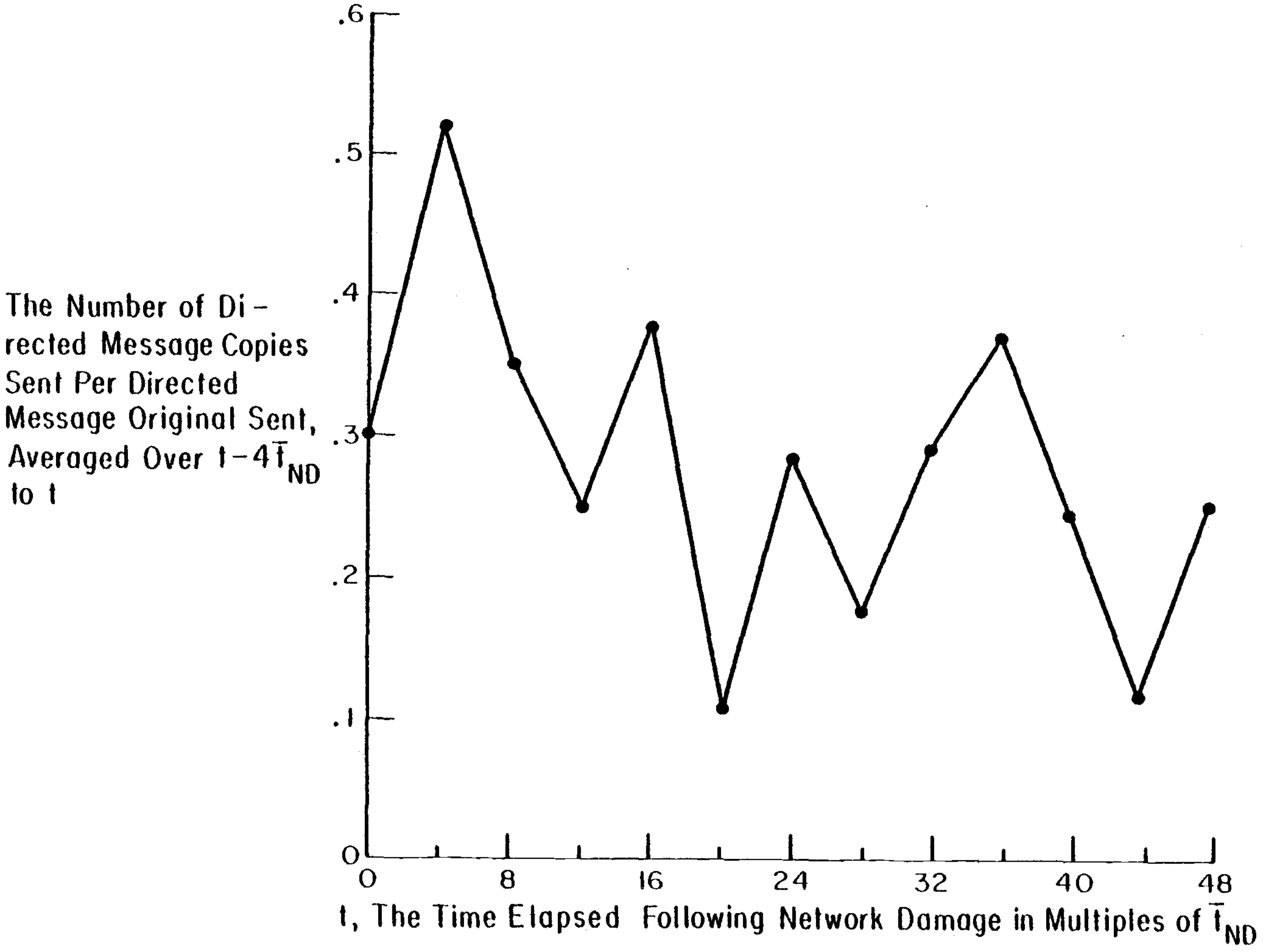


Figure 5. 17. RRPS algorithm with ACK's, T_{PA} = 500, improved model.

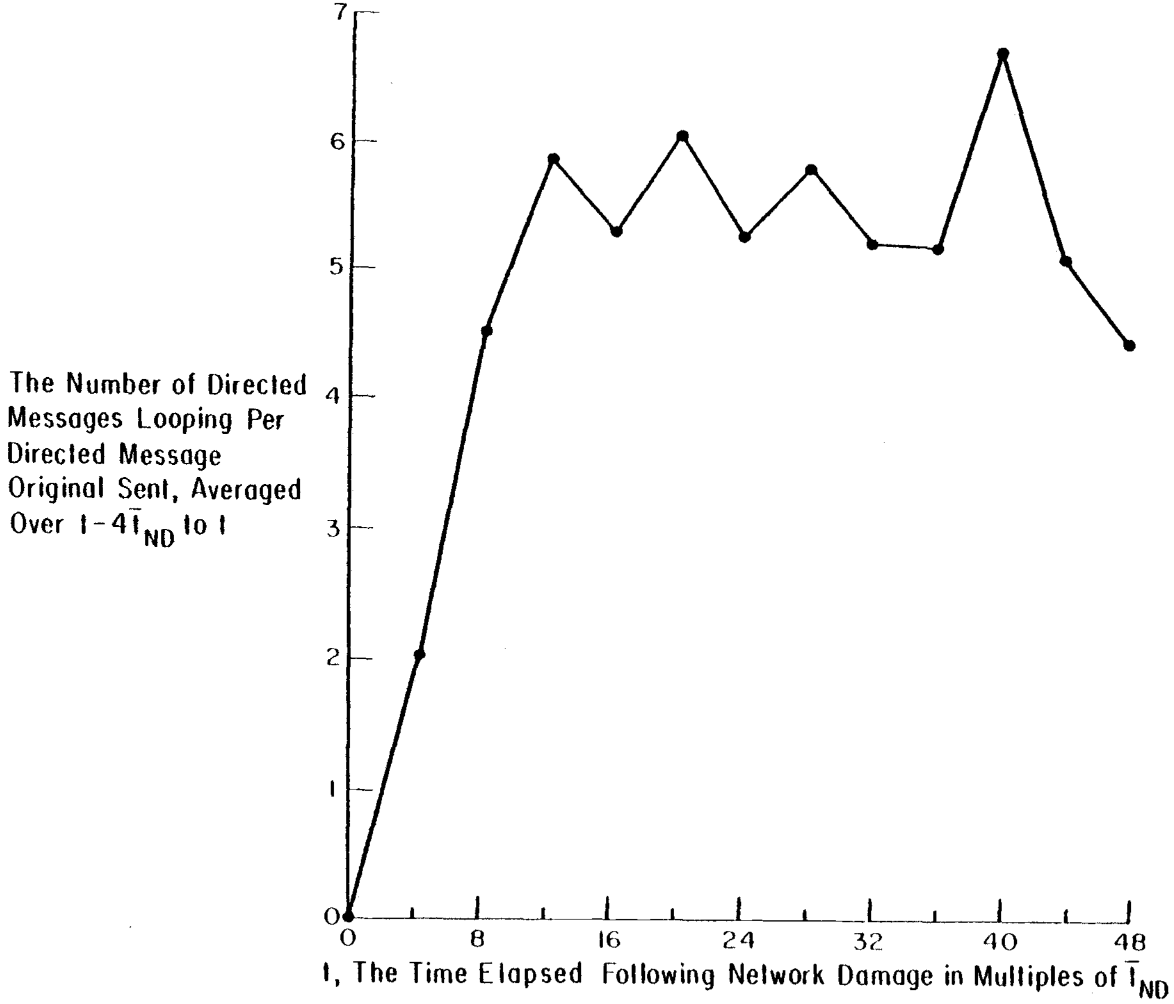


Figure 5. 18. RRPS algorithm with ACK's, T_{PA} = 500, improved model.

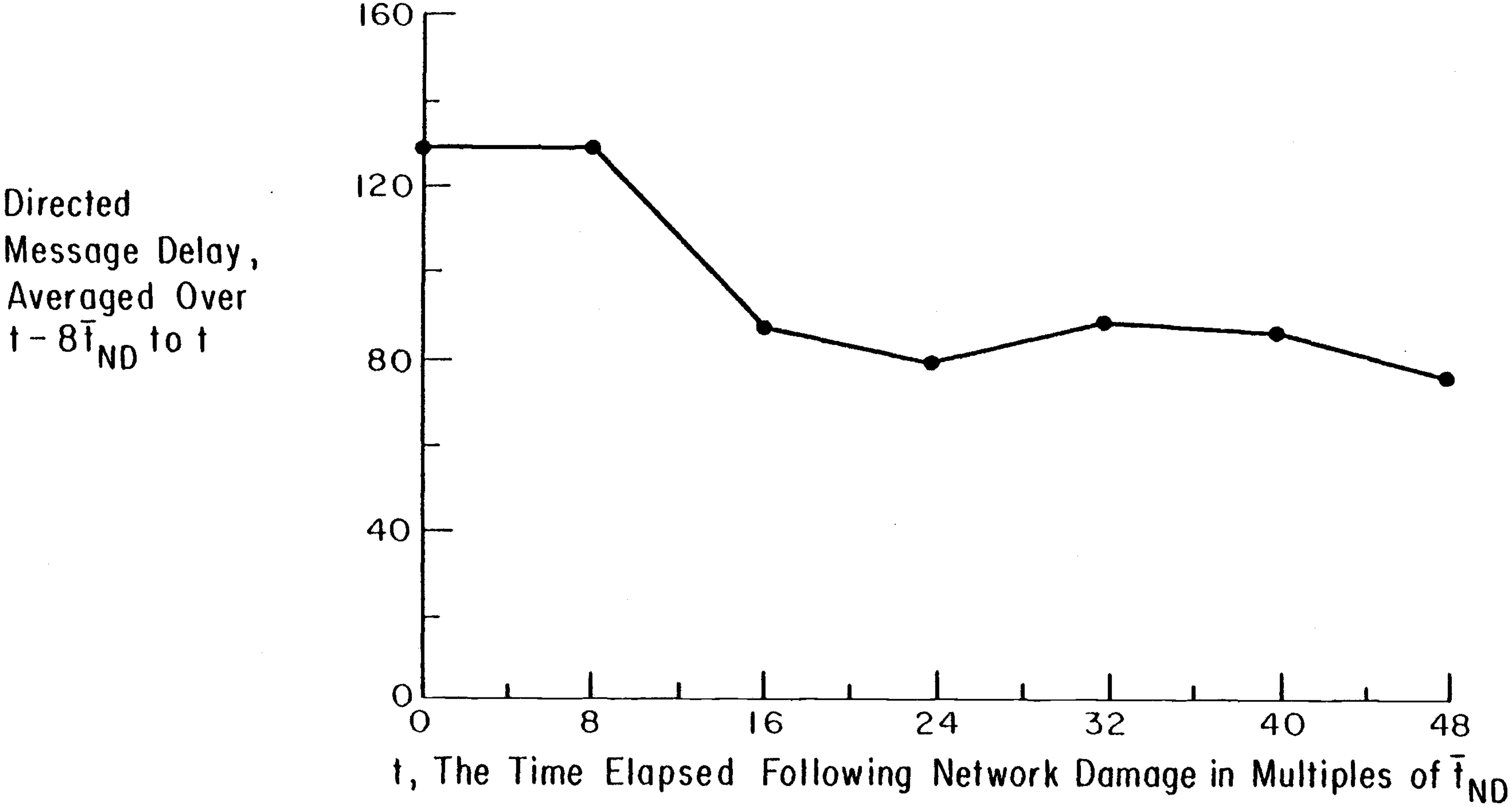


Figure 5. 19. RRPS algorithm with ACK's, T_{PA} = 500, improved model.

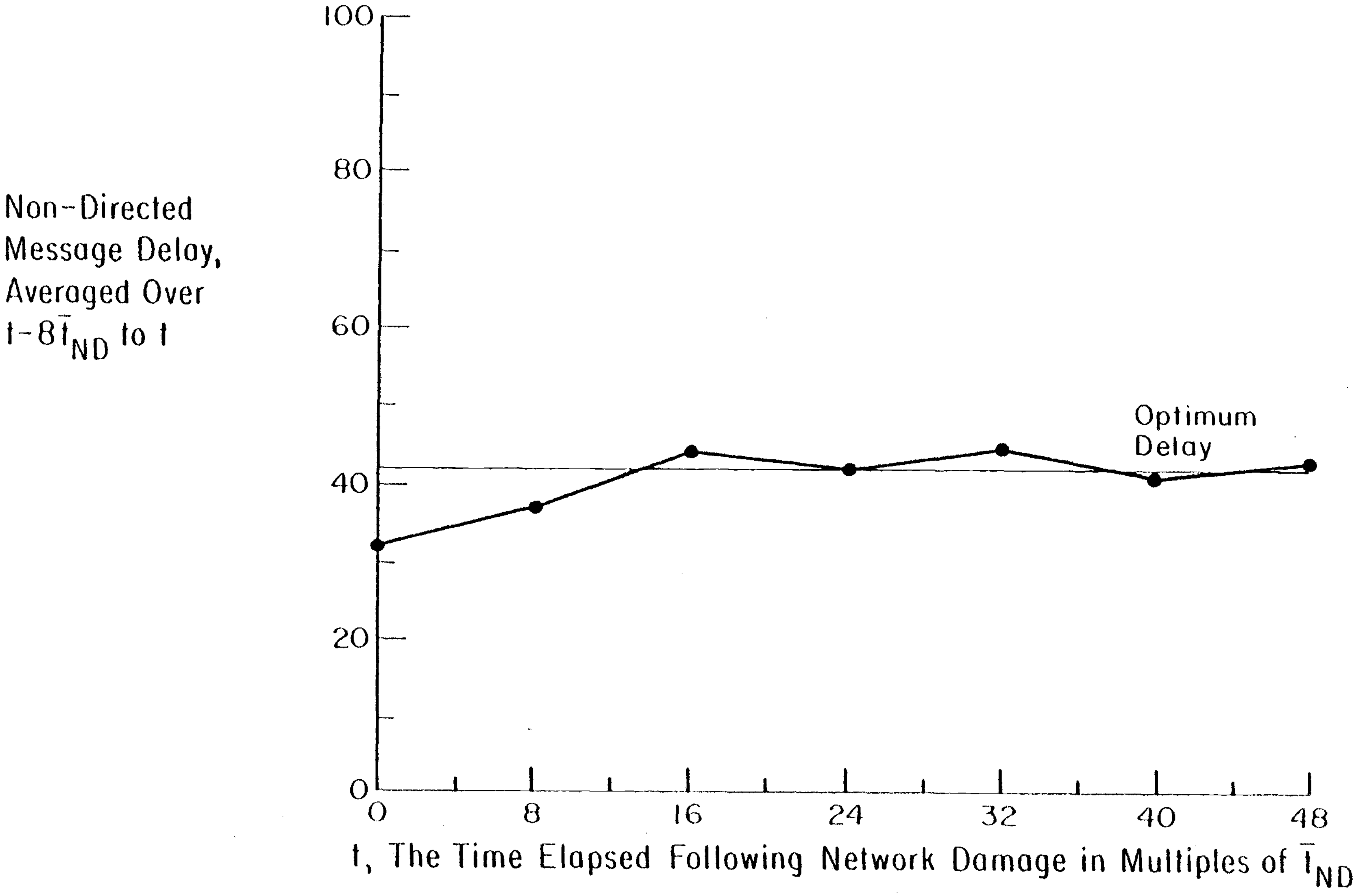


Figure 5.20. RRPS algorithm with ACK's, T_{PA} = 500, improved model.

The statistic on looping messages shows a large increase following the network damage. However, this is characteristic of the Rapid Reciprocal Path Search routing algorithm, and is a result of the flooding of directed messages. The reasons for the directed message flooding are explained in the previous chapter on routing algorithms.

The directed message delay results are somewhat surprising. Following the instant of network damage, the directed message delay remains temporarily steady and then drops and levels out at a value which is actually below its pre-damage value. Apparently, the directed message flooding which occurs after the damage reduces the number of directed messages which are lost due to communication link noise, and thus reduces the directed message delay since fewer messages are included in the average which experienced the time out period delay of 500. It should be remembered, however, that no attempt has been made to select an optimum time out period. Also, these simulation tests have been made using a relatively uncongested network. Thus, the results here should not be considered a conclusive case in favor of directed message flooding.

The non-directed message delay remained low during the entire test. The lack of a significant increase in non-directed message delay indicates that no significant network congestion resulted from the directed message retransmissions.

Summary

Two routing algorithms, the Rapid Reciprocal Path Search algorithm and the Reciprocal Path Search Algorithm with Delay Vectors, have been tested in conjunction with an acknowledgement scheme for the case of six links jammed. The test results indicate that both routing algorithms, when combined with acknowledgement schemes, form complete protocols which reliably deliver messages even with severe network jamming. Thus, the algorithms developed in this thesis constitute a significant new class of protocols for computer communication networks.

The Rapid Reciprocal Path Search algorithm with acknowledgements performed extremely well, indicating that this protocol is an attractive candidate for radar network applications. However, for computer communication networks where jamming and other forms of non-reciprocal network damage are not expected to be a significant problem, some of the other routing algorithms presented in the previous chapter could, when combined with acknowledgements, provide completely adequate protocols with simpler implementation.

VI. CONVENTIONAL NETWORK APPLICATIONS

The most likely candidates for use in conventional computer communication networks among the new routing algorithms are the Simple Backwards Learning algorithm and the Reciprocal Path Search algorithm. The Simple Backwards Learning algorithm adapts to all reciprocal forms of network damage, such as communication link failures in which transmission is prohibited in both directions over the link. This is the most probable form of network damage which can be expected in conventional network applications. For any applications where non-reciprocal forms of damage, such as jamming, can occur significantly often, an algorithm such as the Reciprocal Path Search algorithm or Rapid Reciprocal Path Search algorithm, either without or with the use of delay vectors, needs to be considered.

For the radar network application, the Simple Backwards Learning algorithm requires essentially no overhead since the routing tables are constructed using information contained in the flooded non-directed track report messages, which are generated independently of the need for them to provide information for the routing algorithm. However, in conventional networks, the non-directed track report messages will not exist. Thus, unless some other class of non-directed messages is prevalent, the nodes in a conventional network will need to flood out short "Here I Am" (HIA) messages in order to provide information for the construction of the directed message routing tables. These short HIA messages need only contain the address of the source node and information allowing an estimate of the elapsed delay since the HIA message left its source node to be computed.

In order to put in perspective how much overhead is required by the Simple Backwards Learning algorithm, a comparison between its overhead and the overhead requirements of an ARPA type routing algorithm will be made.

The ARPA routing algorithm was described previously in Chapter II. This algorithm also uses routing tables which are stored at each node. The ARPA routing algorithm uses an exchange of delay estimates to provide information for updating the tables. At set intervals, each node sends to all of its neighbors a list which contains estimates of the shortest delay from it to all the other nodes. This list thus contains a number of entries which is one less than the number of nodes in the network. Upon receiving the list from one of its neighbors, a node adds to each delay estimate a measure of the current delay required to travel from itself to the neighbor from which the list was received, thus forming a new list. This new list then provides that node with an estimate of the minimum delay required to reach all destinations if the message is sent out of the port connecting to the neighbor. The routing table for the node is then constructed by combining the delay estimates derived from the lists received from all of its neighbors.

Consider a computer communication network with n nodes and m ports at each node. For each routing table update, the use of an ARPA type routing algorithm requires each node to transmit m lists, one to each neighbor. Each list will contain n-1 delay estimates. If each delay estimate contains b bits, then the overhead per node required by the ARPA type routing algorithm for each update is

$$m(n-1)b$$

Thus, the total overhead for each update is

$$\begin{aligned} OH_A &= nm(n-1)b \\ &= mb(n^2-n) \end{aligned} \qquad (6.1)$$

Now consider the Simple Backwards Learning algorithm. With this algorithm each routing table update requires each node to flood out a short HIA message. Each node, upon reception of the first copy of an HIA message sends it out of every port except the one in which it was received. Thus, each node sends every HIA message out of at most m-1 ports, except for the source node which sends it out of all m ports. Thus, the maximum number of HIA messages generated per node for every Simple Backwards Learning algorithm update is

$$n(m-1) + 1$$

Hence, the total message transmissions per update is no more than

$$n^2(m-1) + n$$

Each of the HIA messages need to contain the address of the source node and a delay estimate. The message address requires $\log_2 n$ bits and the delay estimate requires b bits. Thus an upper bound on the total overhead for each update of the Simple Backwards Learning algorithm is

$$OH_{SBL} = (\log_2 n + b)(n^2(m-1) + n) \qquad (6.2)$$

Equations (6.1) and (6.2) thus give the overhead requirements for the ARPA and Simple Backwards Learning algorithms, respectively. A cursory examination of the two equations reveals that both overhead requirements increase quadratically as the size of the network increases, and linearly as the number of ports or the number of bits per delay estimate increase. A more detailed comparison of the two overhead requirements is contained in Tables 6.1 and 6.2. Each table compares the two overhead requirements for fixed m, but for varying b and n.

Table 6. 1. A comparison of OH_A and OH_{SBL} for m = 2.

	b = 4		b = 6		b = 8		b = 10	
n	OH_A	OH_{SBL}	OH_A	OH_{SBL}	OH_A	OH_{SBL}	OH_A	OH_{SBL}
4	96	120	144	160	192	200	240	240
8	448	504	672	648	896	792	1120	936
16	1920	2176	2880	2720	3840	3264	4800	3808
32	7936	9504	11904	11616	15872	13728	19840	15840
64	32256	41600	48384	49920	64512	58240	80640	66560
128	130048	181632	195072	214656	260096	247680	325120	280704

Table 6.2. A comparison of OH_A and OH_{SBL} for m = 3.

n	b = 4		b = 6		b = 8		b = 10	
	OH_A	OH_{SBL}	OH_A	OH_{SBL}	OH_A	OH_{SBL}	OH_A	OH_{SBL}
4	144	216	216	288	288	360	360	432
8	672	952	1008	1224	1344	1496	1680	1768
16	2880	4224	4320	5280	5760	6336	7200	7392
32	11904	18720	17856	22880	23808	27040	29760	31200
64	48384	82560	72576	99072	96768	115584	120960	132096
128	195072	361856	292608	427648	390144	493440	487680	559232

Table 6.1 reveals that for $m = 2$, the Simple Backwards Learning algorithm has less overhead for networks which contain eight or more nodes, and which use six or more bits for delay estimates. However, Table 6.2 reveals that for $m = 3$, the ARPA type routing algorithm always uses less overhead. However, the advantage which the Simple Backwards Learning algorithm possesses for $m = 2$, and the advantage which the ARPA type routing algorithm possesses for $m = 3$ is not always very large. Often the overhead requirements of both algorithms are very similar, whether $m = 3$ or $m = 2$, thus emphasizing the importance of other design parameters such as ease of implementation and performance characteristics.

Summary

This chapter has discussed the appropriateness of the routing algorithms developed in this thesis for use in conventional networks. The Simple Backwards Learning algorithm is the most likely candidate for use in conventional networks since it adapts to all forms of reciprocal network damage, the type of damage most conventional networks should expect, and it is relatively simple to implement. For non-reciprocal forms of damage, one of the other routing algorithms, such as the Reciprocal Path Search algorithm or the Rapid Reciprocal Path Search algorithm would most likely be adequate.

The overhead comparisons with an ARPA type routing algorithm indicate that the Simple Backwards Learning algorithm requires less overhead generally for large networks with low connectivity, especially for large values of b, the number of bits required for each delay estimate. Even for some small networks and for some networks with high connectivity, the overhead for the Simple Backwards Learning algorithm remains competitive, particularly for those networks which use many bits in their delay estimates.

VII. AREAS FOR FURTHER RESEARCH

This thesis has developed a new class of directed message routing protocols for computer communication networks. These routing protocols utilize new techniques for searching out and using both reciprocal and non-reciprocal paths for routing directed messages.

Performance tests were conducted in Chapters IV and V on each of five new routing algorithms. These tests were necessarily limited and were designed primarily to assist in developing the new routing algorithms and to analyze their qualitative performance characteristics while adapting to a wide range of network damage. Among the performance characteristics of most interest were whether or not the routing algorithms are able to adapt, their speed of adaptation if they do adapt, and which of the new routing algorithms are able to route messages over optimal paths. The quantitative results obtained from the transient performance tests are not very meaningful, primarily for two reasons. First, the quality of the input data is not good. It is not accurately known what the characteristics of the input traffic will be in an actual radar network. Second, each of the algorithms was tested with only a single simulation run. Thus, a large amount of confidence should not be placed in the exact numerical quantities obtained in the transient tests.

Since the performance tests contained in Chapters IV and V are limited in scope, much further research needs to be done on the new routing algorithms. Areas for further research include investigating what effects the size and topology of a network have on the performance of the routing algorithms; what effects varying the values of k_1, k_2, and k_3 has upon the rate of adaptability; and how the protocols adapt to a greater variety of cases of network damage. It is conceivable that varying these factors may change some of the comparisons made in this thesis.

The algorithms have been tested in a 13 node network since this network contains several alternate paths between nodes (thus providing sufficient capacity for allowing the changes in routes necessary to adapt to changes in network structure), and it is small enough to permit reasonable computing costs. Performance tests should also be conducted using a larger network (a network of 50 or more nodes is the expected size of an actual radar network). In a larger network, it is expected that each routing algorithm will possess the same qualitative characteristics revealed here. However, the size of the network should affect the speed with which each routing algorithm adapts to network damage. A large network will contain both longer paths and more alternate paths than a small network, increasing the amount of time required to disseminate information on reciprocal and non-reciprocal paths throughout the network, thus slowing the

the routing algorithms' adaptability.

The routing algorithms should also be tested using varying values of k_1, k_2, and k_3. The values of these parameters, which remained mostly fixed throughout the performance tests, have a large effect upon the rate of adaptation of the new routing algorithms to network damage. Increasing values of k_1 and k_2 and decreasing values of k_3 should speed up the routing algorithms' adaptability.

The transient tests on the new routing algorithms were also conducted with fixed $\bar{t}_D$ (the mean inter-generation time for directed messages at each node) and $\bar{t}_{ND}$ (the mean inter-generation time for non-directed messages at each node). Further tests should be conducted on the new routing algorithms varying these parameters to determine their performance under different levels of network congestion and varying ratios of directed and non-directed messages.

Tests also need to be conducted on the new routing protocols using a wide variety of cases of network damage. The cases of network damage used in this thesis (three links destroyed, three links jammed, and six links jammed), were selected to provide a wide range of conditions under which to test the routing algorithms. The adaptability of the new routing algorithms needs to be evaluated using cases of network damage which include destroyed nodes, both destroyed and jammed links together, and which include source-destination node pairs which are completely disconnected from each other.

Finally, many refinements can be made to the routing algorithms. An obvious improvement would be to add complexity to the routing algorithms by prohibiting directed messages from leaving a node out of the port in which it arrived (using the routing tables to select a second best port or ports), thus eliminating ping-ponging. This and other refinements could greatly improve the performance of these new routing algorithms, particularly the routing algorithms which include delay vectors.

BIBLIOGRAPHY

1. Wushow Chou, "Computer Communications Networks--The Parts Make up the Whole," National Computer Conference, pp. 119-128, May 1978.

2. David J. Farber, "Networks: An Introduction," Datamation, pp. 36-39, April 1972.

3. Leonard Kleinrock, Queueing Systems Volume II: Computer Applications, John Wiley & Sons, Inc., New York, 1976.

4. John M. McQuillan and Vinton G. Cerf, Tutorial: A Practical View of Computer Communications Protocols, IEEE Computer Society, IEEE Catalog No. EHO 137-0, 1978.

5. W.R. Crowther, et al., "Issues in Packet Switching Network Design," National Computer Conference, pp. 161-175, May 1975.

6. Robert S. Wilkov, "Analysis and Design of Reliable Computer Networks," IEEE Transactions on Communications, Vol. COM-20, No. 3, pp. 660-678, June 1972.

7. S. M. Sussman, et al., "A Survivable Network of Ground Relays for Tactical Data Communication," Report No. WP-21511, The Mitre Corporation, Bedford, Massachusetts, November 1977.

8. L. Cartledge and R. M. O'Donnell, "Description and Performance Evaluation of the Moving Target Detector," Report No. FAA-RD-76-190, Lincoln Laboratory, Lexington, Massachusetts, 8 March 1977.

9. Electronic Systems Division, USAF, "Statement of Work for Data Systems Requirements Study Related to Future USAF Tactical Air Surveillance and Control Improvements," 1977.

10. G. Deley, et al., "Final Report on Data Systems Requirements Study for Future USAF Advanced Forward Area Tactical Radar Network," General Research Corporation, Santa Barbara, California, October 1978.

11. James P. Gray, "Line Control Procedures," Proceedings of the IEEE, Vol. 60, pp. 1301-1312, November 1972.

12. D.W. Davies and D.L.A. Barber, Communication Networks for Computers, John Wiley & Sons, 1973.

13. Mario Gerla, "Deterministic and Adaptive Routing Policies in Packet-Switched Computer Networks," Third IEEE Data Communications Symposium, pp. 23-28, November 13-15, 1973.

14. B.W. Boehm and R.L. Mobley, "Adaptive Routing Techniques for Distributed Communication Systems," IEEE Trans. Commun., pp. 135-144, December 1964.

15. A.V. Butrimenko, "On Searching for the Shortest Paths Along a Graph During Variations in It," Tech. Cybernetics (USSR), Vol. 6, 1964.

16. Paul Baran, "On Distributed Communication Networks," IEEE Trans. Commun., pp. 1-9, March 1964.

17. Raymond L. Pickholtz and Caldwell McCoy, Jr., "Effects of a Priority Discipline in Routing for Packet-Switched Networks," IEEE Trans. Commun., Vol. COM-24, No. 5, pp. 506-516, May 1976.

18. John M. McQuillan, Gilbert Falk, and Ira Richer, "A Review of the Development and Performance of the ARPANET Routing Algorithm," IEEE Trans. Commun., Vol. COM-26, No. 12, December 1978.

19. Leonard Kleinrock, "Analytic and Simulation Methods in Computer Network Design," Spring Joint Computer Conference, pp. 569-579, 1970.

20. Hiroshi Inose and Tadao Saito, "Theoretical Aspects in the Analysis and Synthesis of Packet Communication Networks," Proceedings of the IEEE, Vol. 66, No. 11, pp. 1409-1422, November 1978.

21. Fouad A. Tobagi, et al., "Modeling and Measurement Techniques in Packet Communication Networks," Proceedings of the IEEE, Vol. 66, No. 11, pp. 1423-1447, November 1978.

22. Mischa Schwartz, Computer Communication Network Design and Analysis, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1977.

23. O. Wech, private communication.

24. S. W. Edge and A. J. Hinchley, "A Survey of End-To-End Retransmission Techniques," Computer Communication Review, Vol. 8, No. 4, pp. 1-18, October 1978.

25. L. Pouzin, "Flow Control in Data Networks--Methods and Tools," ICCC Proceedings, Toronto, p. 467, 1976.

26. Israel Gitman, "Comparison of Hop-By-Hop and End-To-End Acknowledgment Schemes in Computer Communication Networks," IEEE Trans. Commun., pp. 1258-1262, November 1976.

27. Carl A. Sunshine, "Factors in Interprocess Communication Protocol Efficiency for Computer Networks," National Computer Conference, pp. 571-576, 1976.

28. Carson E. Agnew, "On Quadratic Adaptive Routing Algorithms," Communications of the ACM, Vol. 19, No. 1, pp. 18-22, January 1976.

29. R. J. Benice and A. H. Frey, Jr., "An Analysis of Retransmission Systems," IEEE Trans. Commun., pp. 135-145, December 1964.

30. R. J. Benice and A. H. Frey, Jr., "Comparisons of Error Control Techniques," IEEE Trans. Commun., December 1964.

31. Kenneth Boulding, "A Simulated Data Communication Network," Computer Communication Review, Vol. 8, No. 4, pp. 19-29, October 1978.

32. S. Carr, S. Crocker, and V. Cerf, "Host to Host Communication Protocol in the ARPA Network," Spring Joint Computer Conference, 1970.

33. V. Cerf and R. Kahn, "A Protocol for Packet Network Intercommunications," IEEE Trans. Commun., Vol. COM-22, No. 5, pp. 637-648, May 1974.

34. Vinton G. Cerf and Peter T. Kirstein, "Issues in Packet-Network Interconnection," Proceedings of the IEEE, Vol. 66, No. 11, November 1978.

35. W. Chou and H. Frank, "Routing Strategies for Computer Network Design," Symposium on Computer Communication Networks and Teletraffic, Polytechnique Institute of Brooklyn, April 4-6, 1972.

36. Stephen D. Crocker, et al., "Function-Oriented Protocols for the ARPA Computer Network," Spring Joint Computer Conference, pp. 271-278, 1972.

37. Guy Fayolle, Erol Gelenbe, and Guy Pujolle, "An Analytic Evaluation of the Performance of the 'Send and Wait' Protocol," IEEE Trans. Commun., Vol. COM-26, No. 3, pp. 313-318, March 1978.

38. Gerard J. Foschini and Jack Salz, "A Basic Dynamic Routing Problem and Diffusion," IEEE Trans. Commun., Vol. COM-26, No. 3, pp. 320-327, March 1978.

39. Howard Frank, Israel Gitman, and Richard Van Slyke, "Packet Radio Systems--Network Considerations," National Computer Conference, pp. 217-231, May 1975.

40. H. Frank, R.E. Kahn, and L. Kleinrock, "Computer Communication Network Design--Experience with Theory and Practice," AFIPS Conference Proceedings, 40:255-270, SJCC, Atlantic City, New Jersey, 1972.

41. Ivan T. Frisch and Howard Frank, "Computer Communications--How We Got Where We Are," National Computer Conference, pp. 109-117, May 1975.

42. G.L. Fultz and L. Kleinrock, "Adaptive Routing Techniques for Store and Forward Computer-Communication Networks," Proc. Int. Conf. Commun., 39-1 to 39-8, 1971.

43. I. Gitman, R.M. Van Slyke, and H. Frank, "Routing in Packet-Switching Broadcast Radio Networks," IEEE Trans. Commun., pp. 926-927, August 1976.

44. Robert G. Gallager, "A Minimum Delay Routing Algorithm Using Distributed Computation," IEEE Trans. Commun., Vol. COM-25, No. 1, pp. 73-85, January 1977.

45. A. Gaspar and P. Lamm, "A Simulated Data Communication Network," Computer Communication Review, Vol. 8, No. 4, pp. 19-29, October 1978.

46. F.E. Heart, et al., "The Interface Message Processor of the ARPA Network," Spring Joint Computer Conference, 1970.

47. C.E. Houstis, W.J. Kelly, Jr., and B.J. Leon, "Distributed Control for a Variable Length Message Store and Forward System," IEEE Trans. Commun., Vol. COM-26, No. 4, April 1978.

48. Robert E. Kahn, "The Organization of Computer Resources into a Packet Radio Network," IEEE Trans. Commun., Vol. COM-25, No. 1, pp. 169-178, January 1977.

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

50. L. Kleinrock, "Models for Computer Networks," Proceedings of the International Communications Conference, pp. 21-9 to 21-16, University of Colorado, Boulder, June 1969.

51. J. M. McQuillan, "Design Considerations for Routing Algorithms in Computer Networks," Proc. Seventh Hawaii Int. Conf. Sys. Sci., pp. 22-24, January 8-10, 1974.

52. Robert M. Metcalfe and David R. Boggs, "Ethernet: Distributed Packet Switching for Local Computer Networks," Communications of the ACM, Vol. 19, No. 7, pp. 395-404, July 1976.

53. David L. Mills, "An Overview of the Distributed Computer Network," National Computer Conference, pp. 523-531, 1976.

54. H. Miyhara, T. Hasegawa, and Y. Teshigawara, "A Comparative Evaluation of Switching Methods in Computer Communication Networks," Proc. Int. Conf. Commun., June 1975.

55. Louis Pouzin and Hubert Zimmermann, "A Tutorial on Protocols," Proceedings of the IEEE, Vol. 66, No. 11, November 1978.

56. A. Rybczynski, et al., "A New Communication Protocol for Accessing Data Networks--The International Packet-Mode Interface," National Computer Conference, pp. 477-482, 1976.

57. Frits C. Schoute and John M. McQuillan, "A Comparison of Information Policies for Minimum Delay Routing Algorithms," IEEE Trans. Commun., Vol. COM-26, No. 8, pp. 1266-1270, August 1978.

58. Adrian Segall, "The Modeling of Adaptive Routing in Data-Communication Networks," IEEE Trans. Commun., Vol. COM-25, No. 1, pp. 169-178, January 1977.

59. J. D. Spragins, "Advanced Forward Area Tactical Radar Network," Interim Report under AFOSR Grant 77-3339, 15 December 1977.

60. J. D. Spragins, "Potential Technology Breakthroughs in Next Generation Radars," Participant's Final Report for 1976 USAF-ASEE Summer Faculty Research Program, 27 August 1976.

61. Thomas E. Stern, "A Class of Decentralized Routing Algorithms Using Relaxation," IEEE Trans. Commun., Vol. COM-25, No. 10, pp. 1092-1102, October 1977.

62. William D. Tajibnapis, "A Correctness Proof of a Topology Information Maintenance Protocol for a Distributed Computer Network," Communications of the ACM, Vol. 20, No. 7, pp. 477-485, July 1977.

63. Fouad A. Tobagi, Stanley E. Lieberson, and Leonard Kleinrock, "On Measurement Facilities in Packet Radio Systems," National Computer Conference, pp. 589-596, 1976.

APPENDIX A

Supplementary Simulation Results

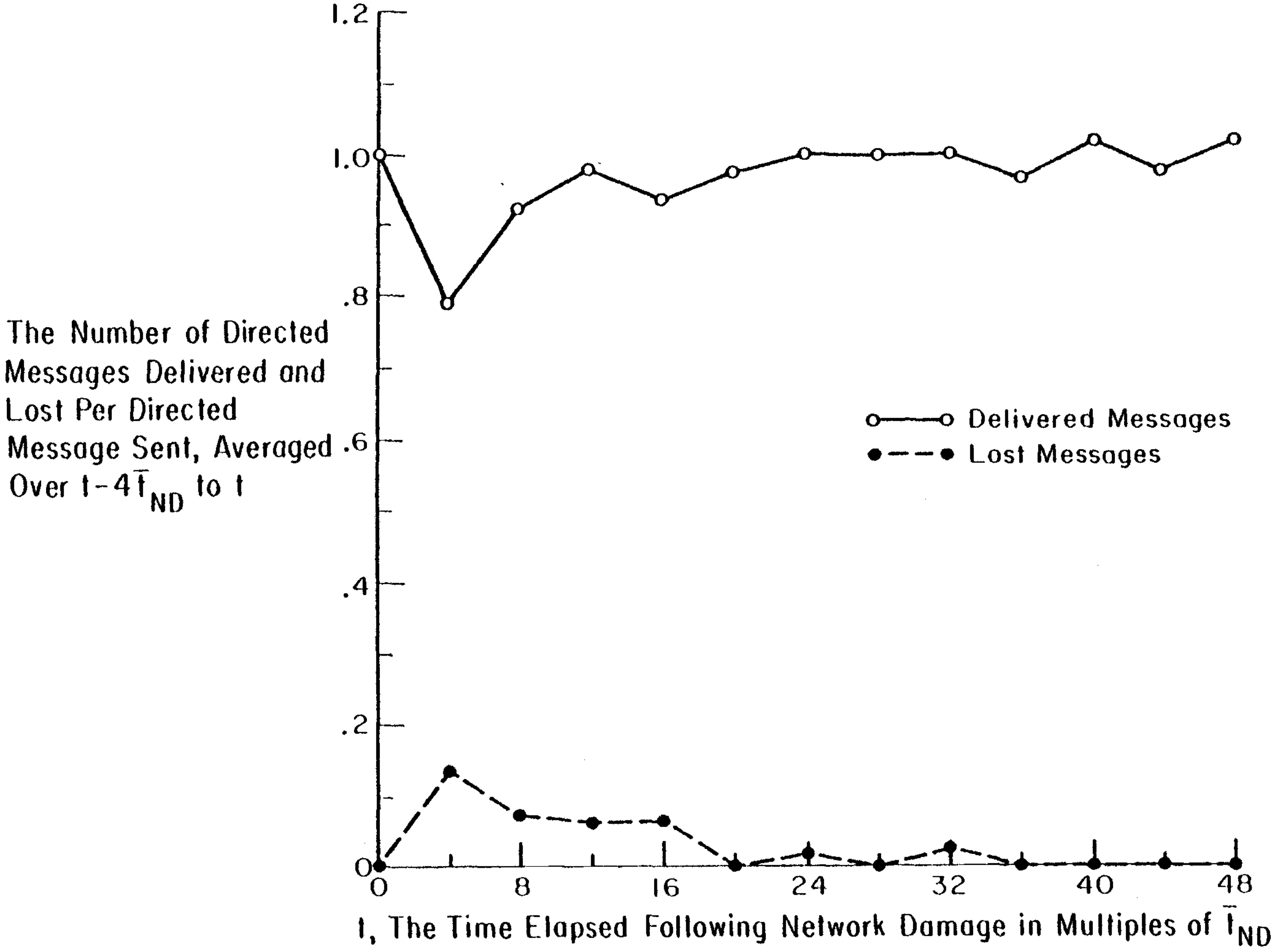


Figure A. 1. RRPSDV algorithm without source node restriction, three links jammed.

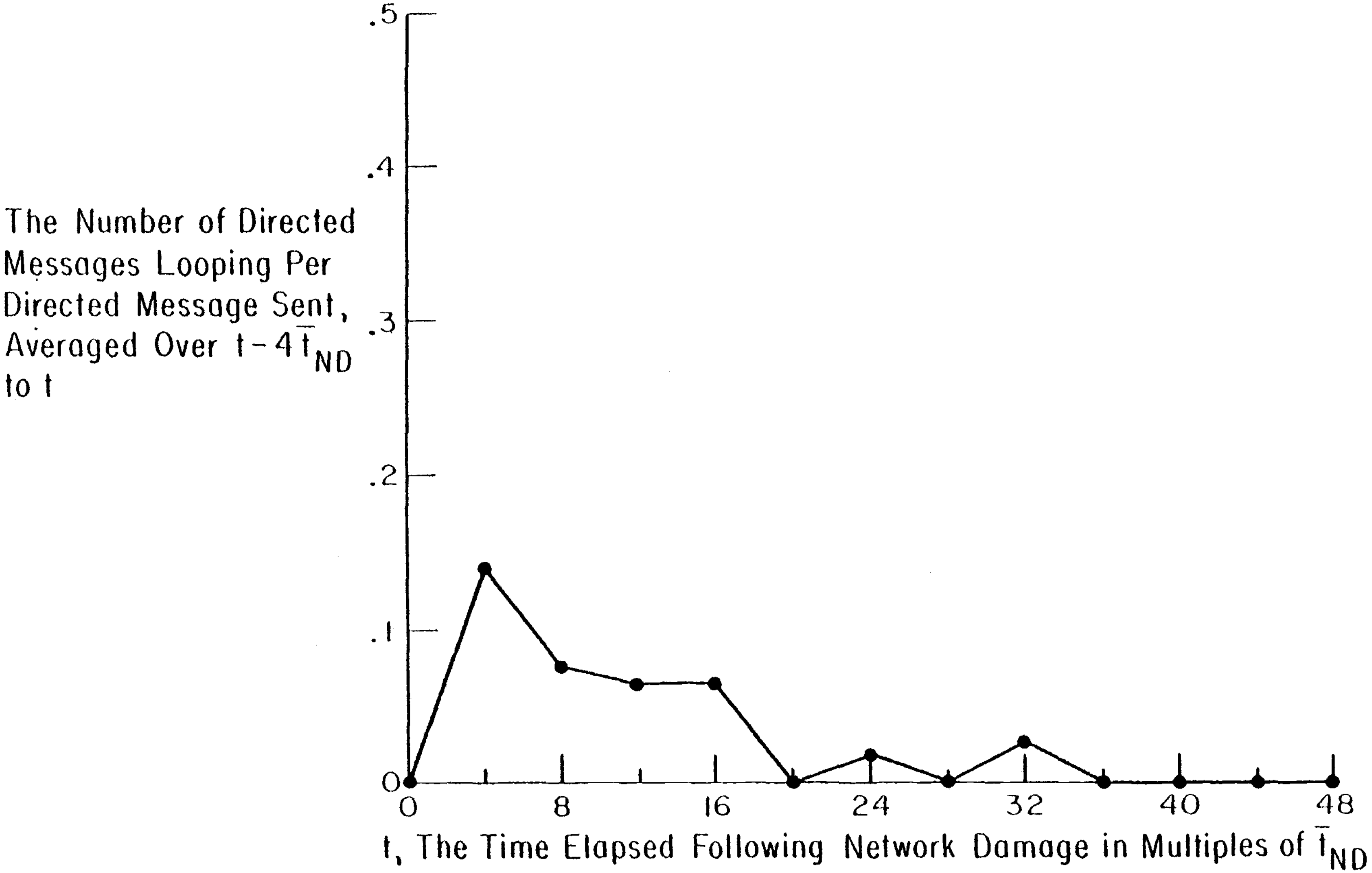


Figure A.2. RRPSDV algorithm without source node restriction, three links jammed.

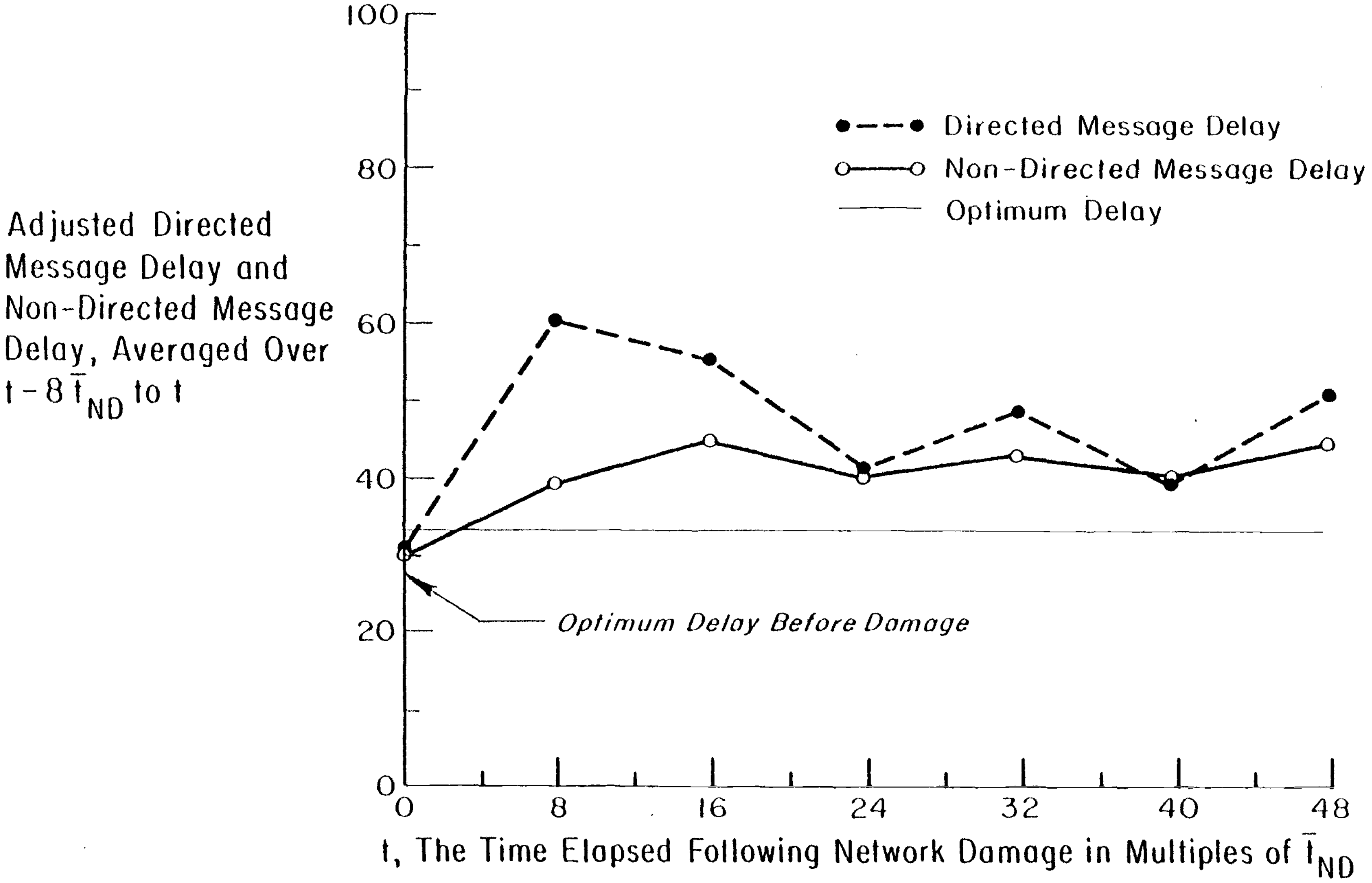


Figure A. 3. RRPSDV algorithm without source node restriction, three links jammed.

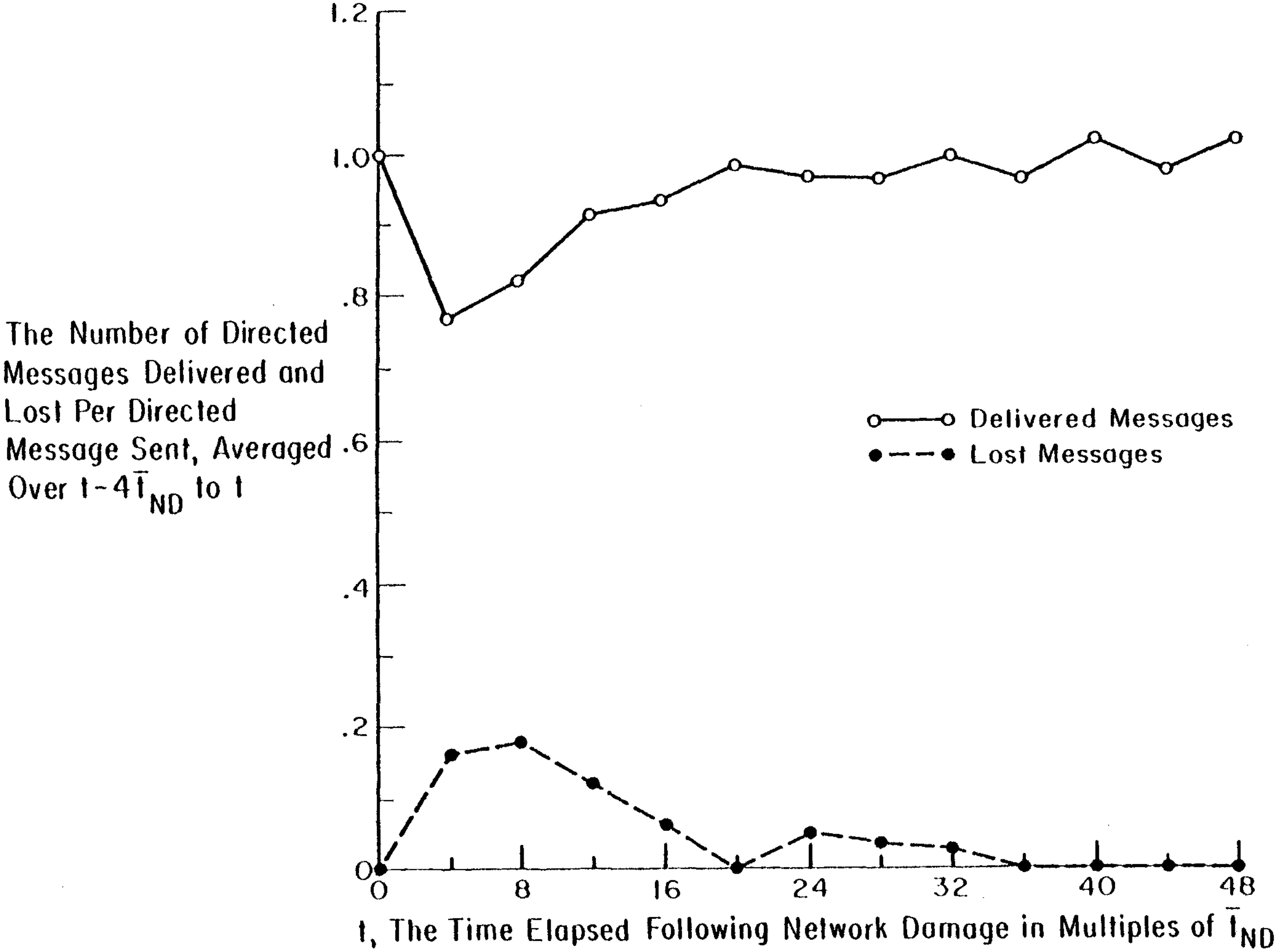


Figure A. 4. RRPSDV algorithm without source node or ping-pong restrictions, three links jammed.

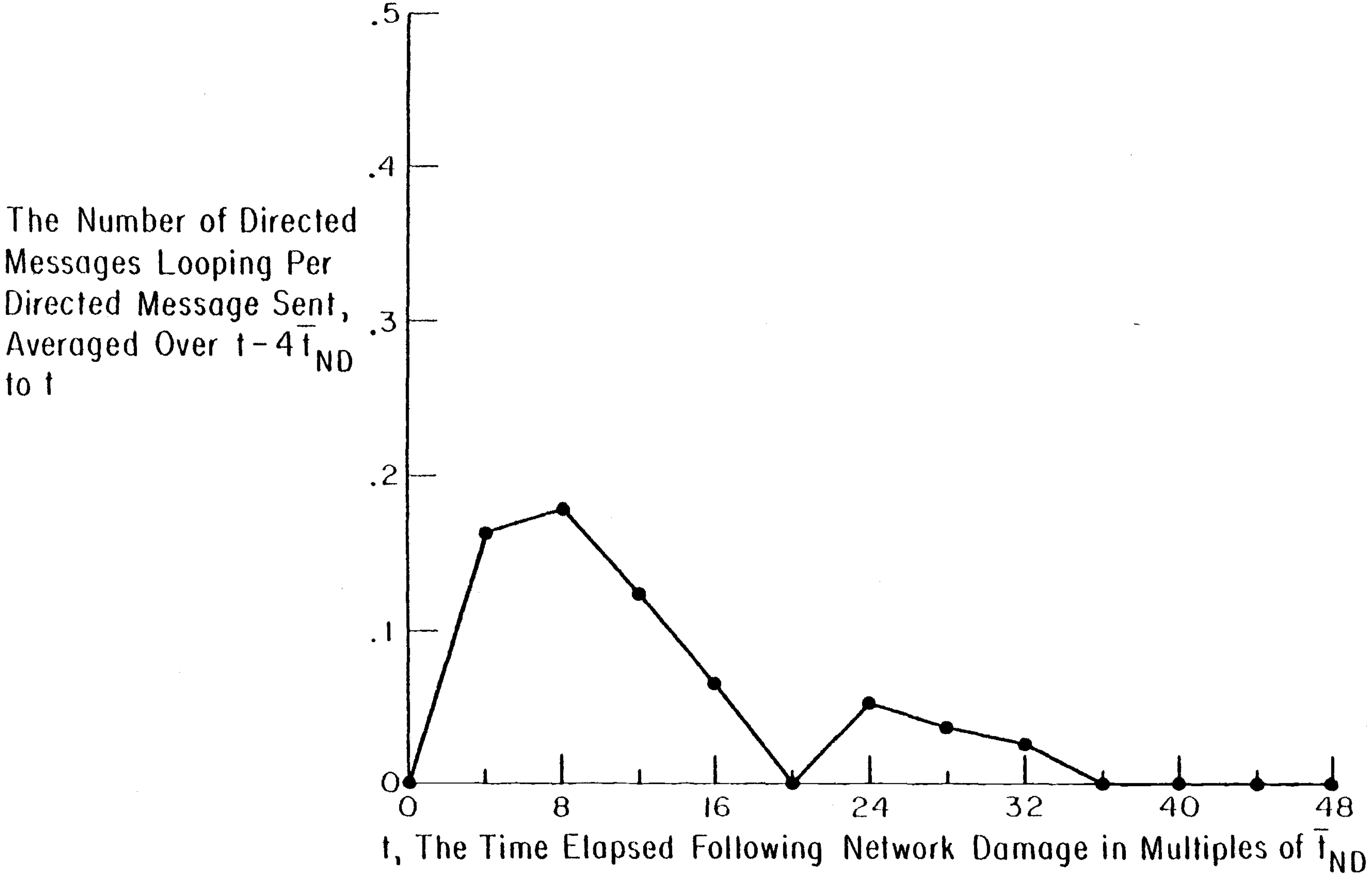


Figure A.5. RRPSDV algorithm without source node or ping-pong restrictions, three links jammed.

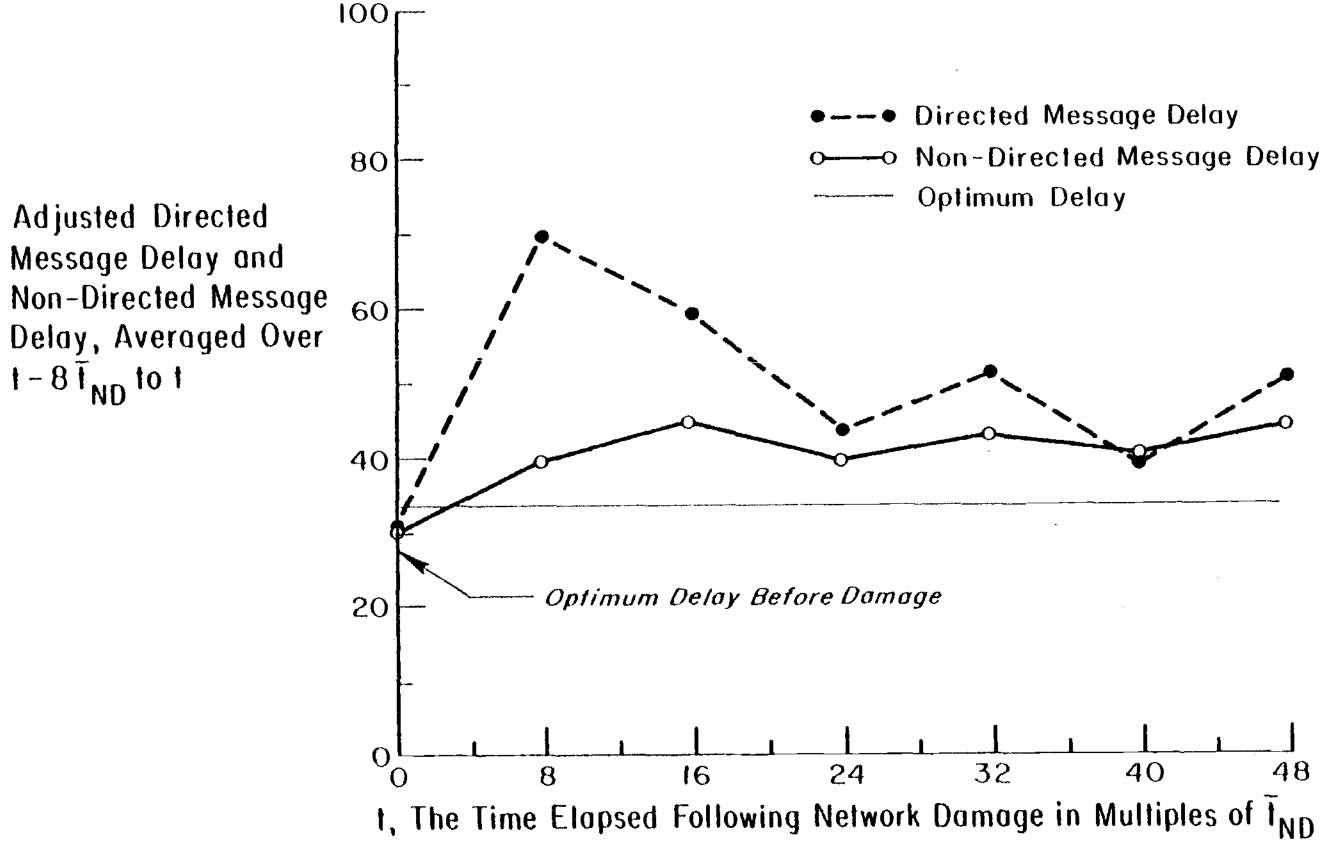


Figure A.6. RRPSDV algorithm without source node or ping-pong restrictions, three links jammed.

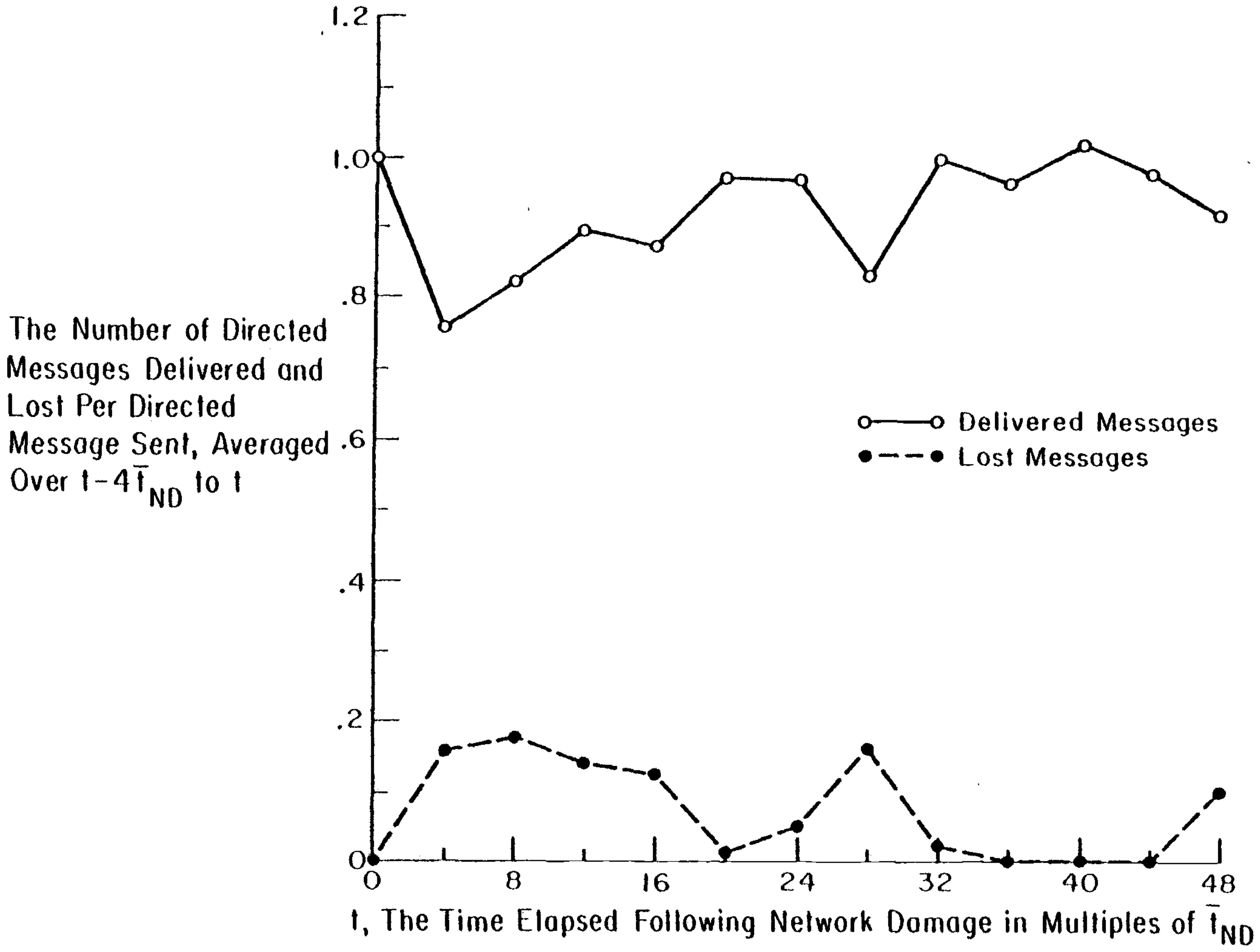


Figure A. 7. RRPSDV algorithm without source node or ping-pong restrictions and k = 10, three links jammed.

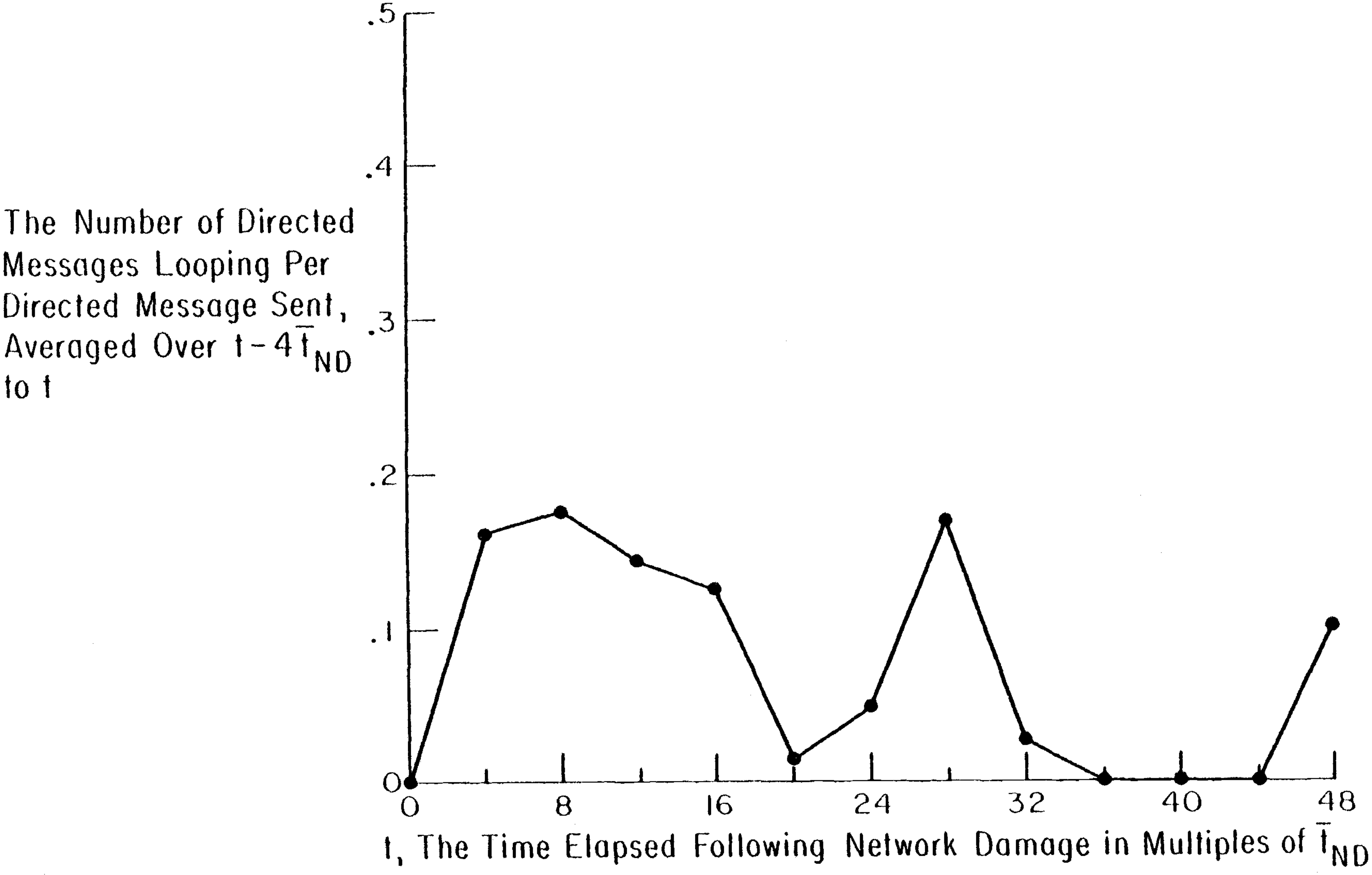


Figure A. 8. RRPSDV algorithm without source node or ping-pong restrictions and k_3 = 10, three links jammed.

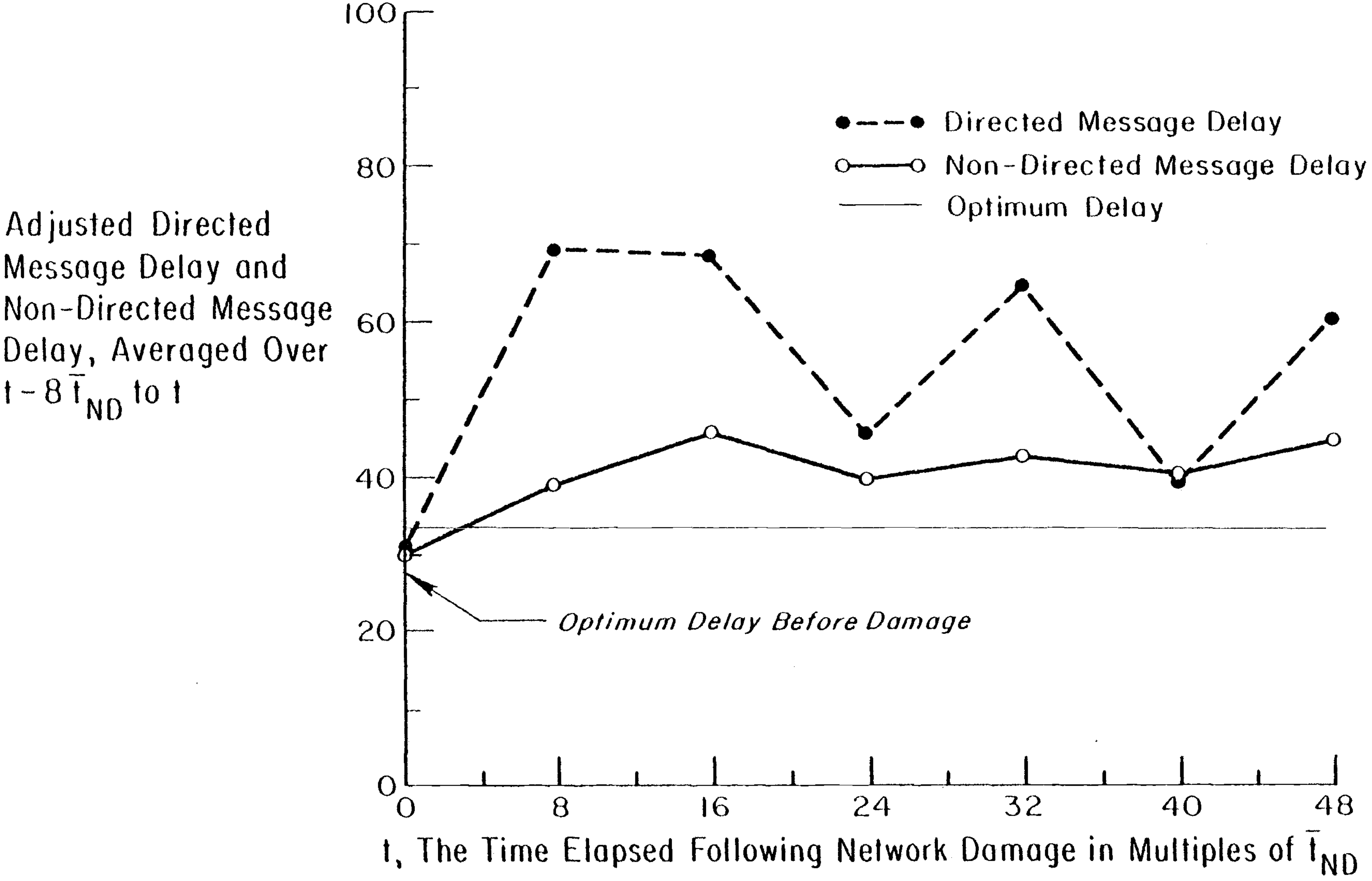


Figure A.9. RRPSDV algorithm without source node or ping-pong restrictions and k_3 = 10, three links jammed.