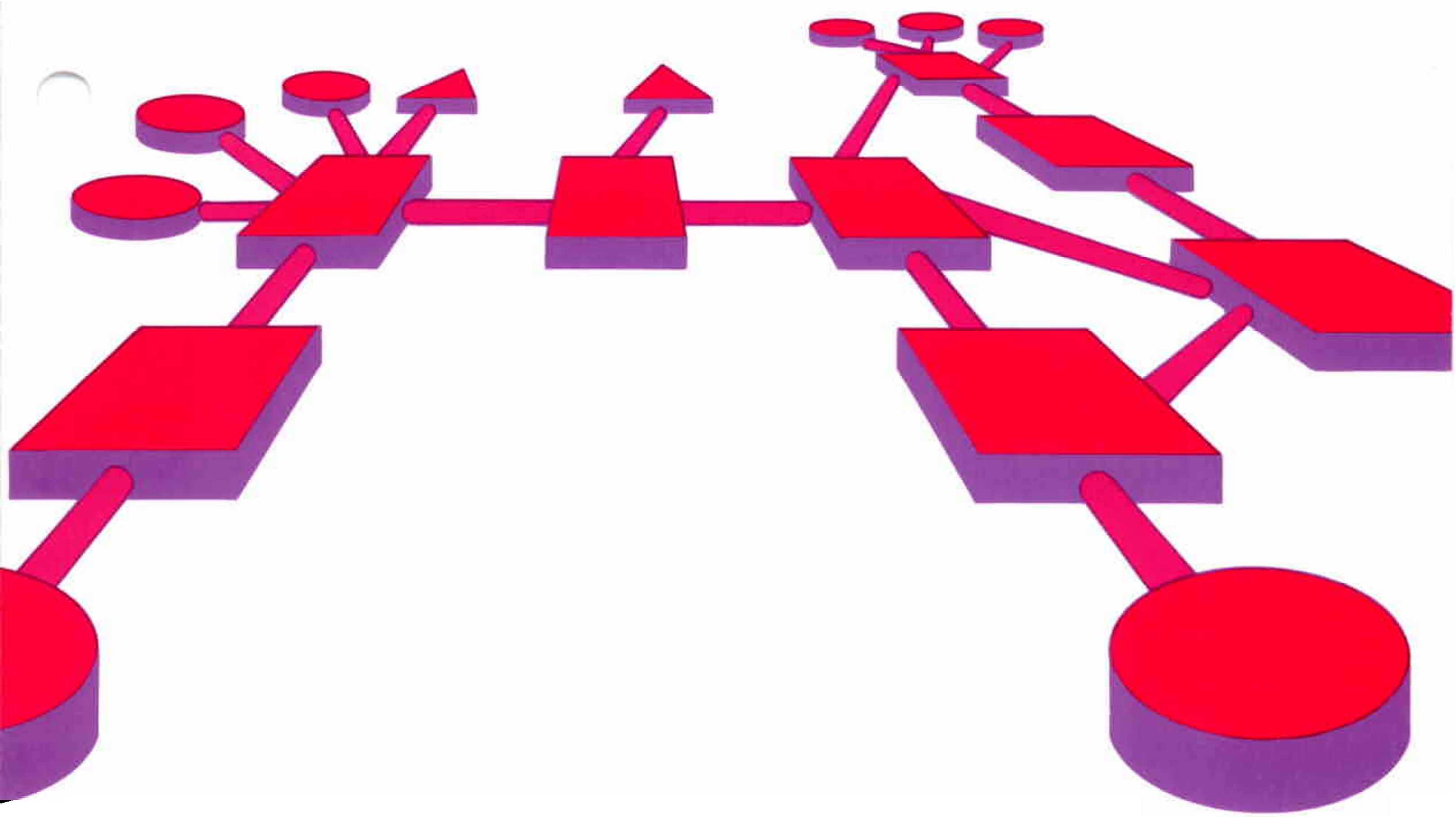
Order No. AA-K180A-TK

DECnet
DIGITAL Network Architecture

Transport
Functional Specification

Version 1.3.0



DECnet
DIGITAL Network Architecture
(Phase III)

Transport
Functional Specification

Order No. AA-K180A-TK
Version 1.3.0

October 1980

This document specifies the functions, interfaces, and protocols for implementing Transport. Transport is that part of the DIGITAL Network Architecture that models the software controlling the routing of messages within DECnet communications networks.

To order additional copies of this document, contact your local Digital Equipment Corporation Sales Office.

digital equipment corporation · maynard, massachusetts

First Printing, October 1980

This material may be changed without notice by Digital Equipment Corporation, and Digital Equipment Corporation is not responsible for any errors which may appear herein.

The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

CONTENTS

CONTENTS (Cont.)

FIGURES

CONTENTS (Cont.)

Page

TABLES

1.0 INTRODUCTION

This document describes the structure, functions, interfaces, protocols, and algorithms for implementing Transport. Transport is the part of the DIGITAL Network Architecture that models the software (or hardware) controlling the routing of messages, called packets, within DECnet communications networks.

A DECnet network is a family of software modules, data bases, and hardware components typically used to tie DIGITAL systems together for resource sharing, distributed computation, or remote system communication. DECnet network implementations follow the DIGITAL Network Architecture (DNA) model.

DNA is a layered structure. Modules in each layer perform distinct functions. Equivalent modules within the same layer in both the same and different nodes communicate using protocols. A node is an implementation of the DNA Session Control layer (Section 2.2). Usually a single computer is associated with one node. Protocols are the messages exchanged by modules and the rules governing the message exchanges. Modules in functionally different layers of DNA interface using either subroutine calls or a system-dependent method. This document describes these interfaces in the format of calls to subroutines.

The routing described in this document is Phase III DECnet routing. It is the major function of the Transport layer. DIGITAL's routing is intended for users with networks consisting of leased lines and local multi-connected topologies (via twisted pair, coaxial cable, and so on).

Phase III DECnet supports two types of routing and is also compatible with Phase II DECnet. Thus there are three types of nodes in terms of routing:

1. Routing nodes. These deliver packets to other nodes, receive packets from other nodes, and route packets from other source nodes through to other destination nodes.

2. Nonrouting nodes. These deliver packets to other nodes and receive packets from other nodes, but do not route packets through.

3. Phase II nodes. These deliver packets to, and receive packets from, adjacent nodes only.

Networks that include nonrouting and Phase II nodes are restricted in how the nodes can be interconnected and which pairs can communicate. Appendices B and G explain these restrictions.

A glossary at the end of this document defines many Transport terms.

This document is intended for readers familiar with computer communications and with DECnet. The primary audience is those who are implementing DECnet systems. However, it may also be of interest to those who want to know the details of the Transport design. Other DNA Phase III functional specifications are:

DNA Data Access Protocol (DAP) Functional Specification, Version 5.6.0, Order No. AA-K177A-TK

DNA Digital Data Communications Message Protocol (DDCMP) Functional Specification, Version 4.1.0, Order No. AA-K175A-TK

DNA Maintenance Operations Protocol (MOP) Functional Specification, Version 2.1.0, Order No. AA-K178A-TK

DNA Network Management Functional Specification, Version 2.0.0, Order No. AA-K181A-TK

DNA Network Services (NSP) Functional Specification, Version 3.2.0, Order No. AA-K176A-TK

DNA Session Control Functional Specification, Version 1.0.0, Order No. AA-K182A-TK

The following overview document for these specifications is an introduction to the structure and functions of DNA.

DNA General Description, Order No. AA-K179A-TK

2.0 FUNCTIONAL DESCRIPTION

Transport routes messages in DECnet networks and manages the message packet flow. A packet is a unit of data to be routed from a source node to a destination node. Transport also prevents old packets from corrupting the Network Services layer. The Transport layer consists of two sublayers:

1. **Transport** control. The Transport control sublayer supplies full-duplex packet transmission between any pair of nodes. It is independent of the Data Link layer below it, and masks the physical and topological characteristics of the network from higher layers. It consists of the following components:

   - Routing
   - Congestion control
   - Packet lifetime control

2. **Transport initialization.** The Transport initialization sublayer masks the characteristics of the Data Link layers from the Transport control sublayer. It consists of the following components:

   - Initialization
   - Physical line monitor

   The Transport initialization sublayer controls the Data Link layer and is Data Link layer dependent.

The Transport components provide the following functions. Section 2.5 gives a more detailed description.

Routing. The routing function determines packet paths. A path is the sequence of connected nodes between a source node and a destination node. When Transport receives a packet, the routing component refers to a data base that is periodically updated by Transport modules in adjacent nodes. The routing component uses information in this data base to determine if a path to a destination exists, and, if so, what the next line in the path is. The routing component then forwards the packet to its destination. If more than one path exists to a destination, the routing component ascertains the best path.

The combined knowledge of all the Transport modules of all the Phase III nodes in a network ascertain the existence of a path, and route the packet to its destination. The routing component at a routing node has the following specific functions:

- It extracts and interprets the route header in a packet.
- It performs packet forwarding based on the destination.
- It manages the characteristics of the path. If a node or line fails on a path, it finds a bypass or set of lines to re-establish the path.
- It interfaces with the Transport initialization layer to receive reports concerning a line or node that has failed or the subsequent recovery of a line or node.

- It returns packets addressed to unreachable nodes to the Network Services layer (NSP), if requested to do so by NSP. A node is unreachable if the path to it exceeds the maximum hops of the network. A hop is the logical distance between two adjacent nodes. Maximum hops is a Transport parameter that is equal to the maximum path length in the network.

- It delivers packets between Phase II nodes and Phase III routing nodes. (A Phase II node can deliver packets to and receive packets from an adjacent Phase III node only.)

Congestion Control. Congestion control manages the buffers at each packet switching node (that is, at each node that permits route-through).

Packet Lifetime Control. Packet lifetime control supplies the following functions:

- It bounds the number of nodes a packet can visit.

- It detects the halting of adjacent nodes, thereby bounding the time a packet can spend in a node that is halted before continuing.

Initialization. The initialization component supplies the following functions:

- It identifies the adjacent node and the adjacent node's Transport layer.

- It performs node verification, if required.

Physical Line Monitor. This component monitors errors detected by the Data Link layer.

2.1 Design Scope

Transport supports the following design requirements:

1. **Deliverability.** The routing component accepts and delivers packets addressed to reachable destinations and rejects packets addressed to unreachable destinations.

2. **Adaptability.** The routing component adapts to topological changes, but not to traffic changes. (Topological changes are changes in the configuration of active lines and nodes in a network. Traffic changes are changes in the physical load on lines in a network.)

3. **Promptness.** The periods of adaptation to topological changes in the network are a reasonable function of the network diameter (that is, the maximum logical distance between network nodes) and line speeds.

4. **Efficiency.** Transport is both processing and core efficient. It does not create excessive routing line overhead.

5. **Robustness.** Transport recovers from transient errors such as lost or temporarily incorrect routing messages. Transport tolerates imprecise parameter settings.

6. **Stability.** The routing component stabilizes in finite time to "good routes," provided no continuous topological changes or continuous data base corruptions occur.

7. **Operator control.** An operator can control many routing functions via parameter changes, and inspect parameters, counters, and routes. Routing, however, will not depend on operator input for correct behavior.

8. **Simplicity.** Transport is sufficiently simple to permit performance tuning and failure isolation.

9. **Maintainability.** Transport provides mechanisms to detect, isolate, and repair most common errors that may affect the routing computation and data bases.

10. **Verification of compatibility.** Transport initialization prevents incompatible routing algorithms from coexisting in the network.

11. **Heterogeneity.** Transport operates over a mixture of network node types, communication lines, and topologies.

12. **Support of subsets.** Transport allows nodes to support a subset of the routing functions.

13. **Extensibility.** Transport accommodates increased routing functions, leaving earlier functions as a subset.

14. **Evolution.** Transport allows orderly transition from algorithm to algorithm without shutting down an entire network.

15. **Deadlock Prevention.** The congestion control component prevents deadlock, the condition in which Transport fails to deliver data.

16. **Independence.** Congestion control does not depend on routing for effective operation.

17. **Duplicate message reduction.** The packet lifetime control algorithm significantly reduces the risk of the user receiving duplicate messages.

The following are not within the scope of Phase III Transport:

1. **Large networks.** The routing component does not support networks of hundreds of nodes efficiently.

2. **Traffic adaptation.** The routing component does not react to traffic flow automatically.

3. **Traffic service classes.** The routing component does not distinguish among different classes of traffic in route determination.

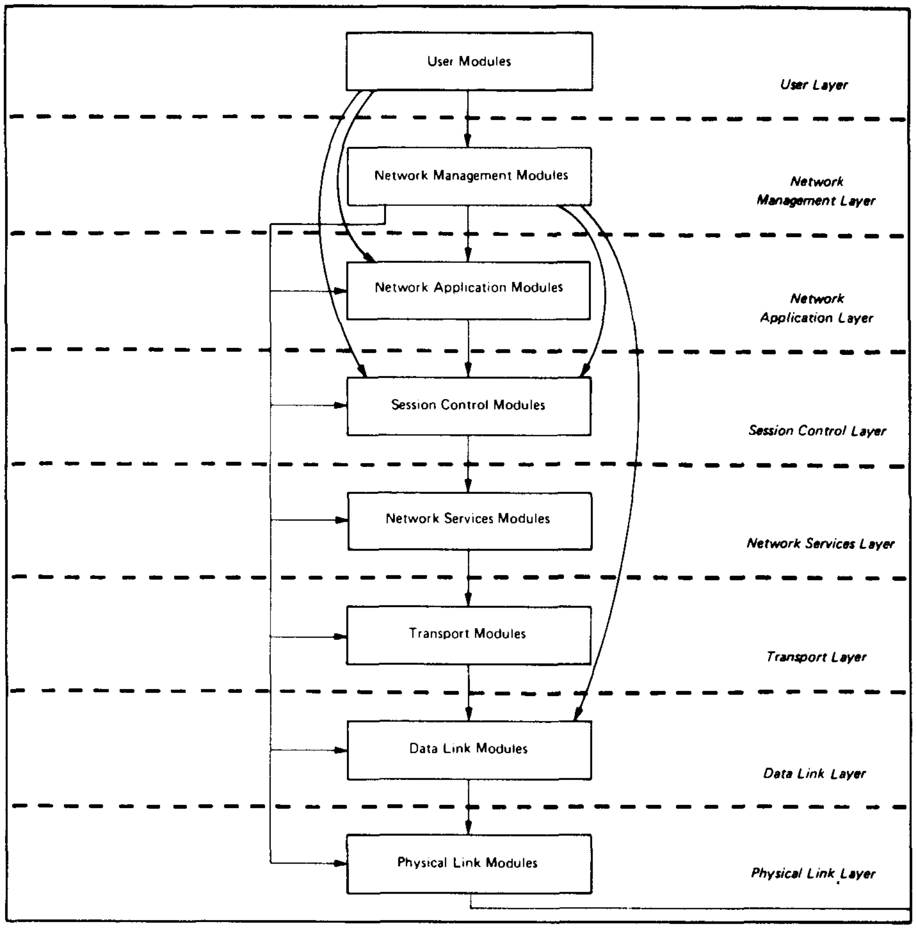
4. **Source-destination routing.** The routing component does not determine routes by source as well as destination.

5. **Gross operator failure.** Transport does not attempt to protect the network from operator actions, such as removing a line or a node, that may disconnect the network.

6. **Guaranteed delivery.** The congestion control component does not guarantee delivery of all offered packets.

2.2 Relationship to DIGITAL Network Architecture

The DIGITAL Network Architecture (DNA) is a model that defines the functional requirements of all DECnet implementations. The model is a layered one. The Transport layer lies between the Network Services layer and the Data Link layer, as shown in Figure 1.



Horizontal arrows show direct access for control and examination of parameters, counters, etc. Vertical and curved arrows show interfaces between layers for normal user operations such as file access, down-line load, up-line dump, end-to-end looping, and logical link usage.

Figure 1 Relationship of Transport to DNA

A brief description of each DNA layer follows:

1. **User layer.** The highest layer, the User layer supports user services and programs.

2. **Network Management layer.** Modules in the Network Management layer provide user control over and access to network parameters and counters. These modules also furnish up-line dumping, down-line loading, and testing functions. This layer is the only layer that has direct access to each lower layer for control purposes.

3. **Network Application layer.** Modules in the Network Application layer support network functions, such as remote file access and file transfer, used by the two higher layers.

4. **Session Control layer.** The Session Control layer defines the system-dependent aspects of logical link communication, which allows controlled data movement between network nodes.

5. **Network Services layer.** The Network Services layer defines the system-independent aspects of logical link communication.

6. **Transport layer.** Modules in the Transport layer route messages between source and destination nodes.

7. **Data Link layer.** The Data Link layer defines the protocol concerning data integrity and physical channel management.

8. **Physical Link layer.** The Physical Link layer encompasses a part of the device driver for each communications device plus the communications hardware itself. The hardware includes interface devices, modems, and the communication lines. This layer controls the end-to-end transmission of data.

Each DNA layer uses the services of the layer below it. In addition, Network Management provides a control interface to all the DNA layers below it. User and Network Application layer modules can interface directly with Session Control.

2.3 Transport Layer Environment Requirements

The Transport layer requires guarantees from the operating system, the Network Services layer, and the Data Link layer.

The required operating system guarantees are:

1. Priority scheduling such that Transport receives minimum processing guarantees

2. A quota of buffers to Transport sufficient to perform routing and packet lifetime control functions

3. Access to a timer or notification of specific timer expiration

The Network Services layer must guarantee the return of a buffer within a specified amount of time. Otherwise, Transport may discard packets received for Network Services.

The required Data Link layer guarantees are:

1. Provision that both source and destination nodes complete start-up before message exchange can occur

2. Detection of remote start-up

3. Provision that no old messages be received after start-up is complete

4. Masking of transient errors in order to prevent packet data corruption

5. Provision for not duplicating or corrupting packets

6. Packet sequentiality ensuring that, if a packet has been received, all previously sent packets have been received

7. Reporting of failures and degraded line conditions

2.4 Transport Characteristics

The Transport layer possesses the following characteristics:

- **Variable delay.** There is a variable delay time. Delay is defined as the time between receipt of a packet from Network Services at a source node and delivery of that packet to Network Services at a destination node.

- **Nonsequential delivery.** Transport does not guarantee delivery of packets to Network Services at the destination node in the same sequence in which they were received from Network Services at the source node.

- **Packet integrity.** Transport will not duplicate, modify, or misdeliver a packet.

2.5 Transport Control Functional Organization

The Transport control sublayer components described at the beginning of Section 2 can be broken down into more specific functional components. Figure 2 shows the relationship of these components and their functions. These are described briefly here and in detail in Sections 4, 5, and 6.

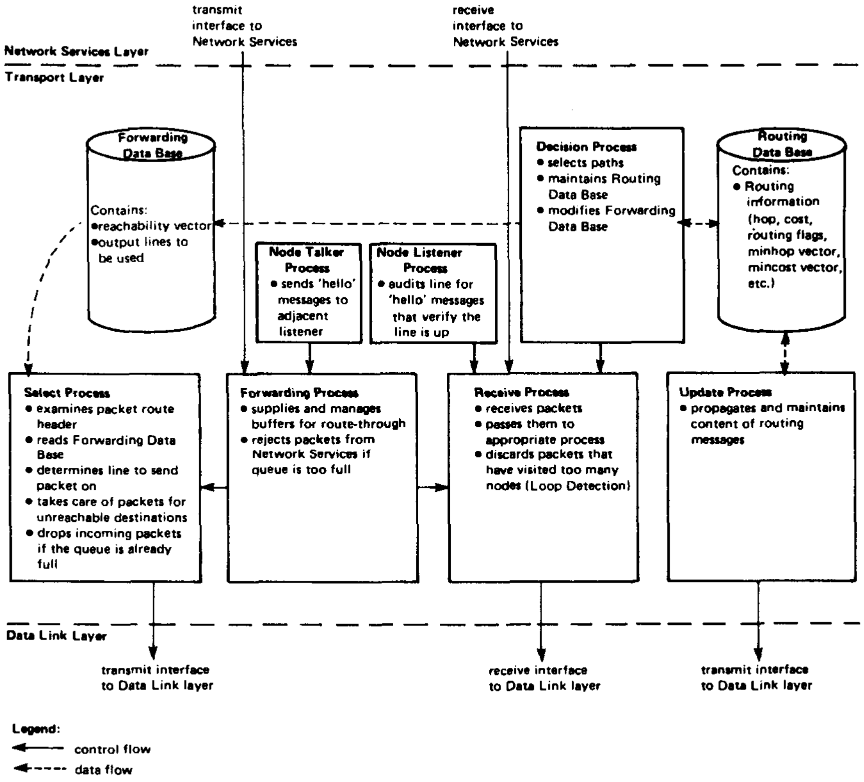

transmit interface to Network Services
receive interface to Network Services
Network Services Layer
Transport Layer
Forwarding Data Base
Contains:
•reachability vector
•output lines to be used
Decision Process
• selects paths
• maintains Routing Data Base
• modifies Forwarding Data Base
Routing Data Base
Contains:
• Routing information (hop, cost, routing flags, minhop vector, mincost vector, etc.)
Node Talker Process
• sends 'hello' messages to adjacent listener
Node Listener Process
• audits line for 'hello' messages that verify the line is up
Select Process
• examines packet route header
• reads Forwarding Data Base
• determines line to send packet on
• takes care of packets for unreachable destinations
• drops incoming packets if the queue is already full
Forwarding Process
• supplies and manages buffers for route-through
• rejects packets from Network Services if queue is too full
Receive Process
• receives packets
• passes them to appropriate process
• discards packets that have visited too many nodes (Loop Detection)
Update Process
• propagates and maintains content of routing messages
Data Link Layer
transmit interface to Data Link layer
receive interface to Data Link layer
transmit interface to Data Link layer
Legend:
control flow
data flow

Figure 2 Transport Components and Their Functions

2.5.1 **Routing** - The routing processes and data bases are:

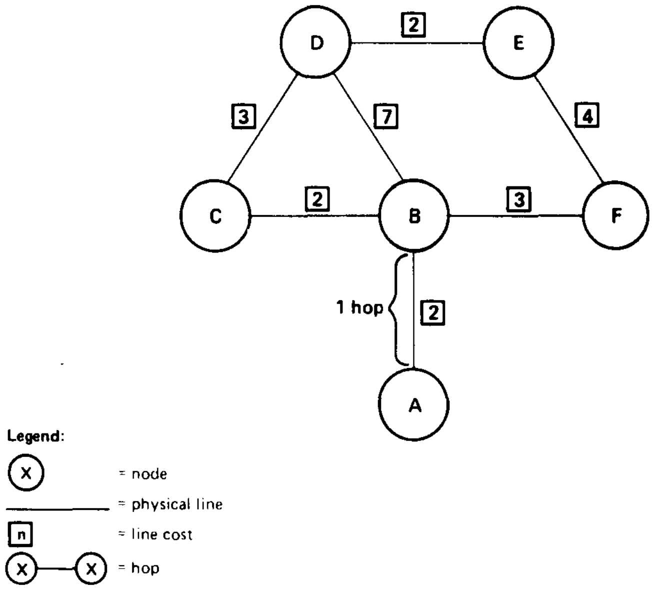
- Decision process (Section 2.5.1.1)
- Update process (Section 2.5.1.2)
- Forwarding process (Section 2.5.1.3)
- Receive and select processes (Section 2.5.1.4)
- Routing data base (specified in Section 4)
- Forwarding data base (specified in Section 4)

2.5.1.1 **Decision Process** - This process selects routes to each destination in the network. It consists of a connectivity algorithm that maintains path lengths and a traffic assignment algorithm that maintains path costs. Path length is the sum of the hops along a path between two nodes. Line cost is a positive integer value associated with using a line, and path cost is the sum of the line costs along a path between two nodes.

When a routing node receives a Routing message (a type of Transport Control message) from an adjacent node, the routing node executes the two decision algorithms. Execution of the two algorithms results in the determination of lines along which to forward packets and possibly the conclusion that one or more particular destination nodes are unreachable.

The system manager must set several of the parameters in the routing data base that the decision process uses. These include line cost, maximum cost, maximum address, maximum lines, and maximum hops. The values of the cost parameters are arbitrary. Appendix F suggests an algorithm for determining line costs. The values of the other parameters depend on the specific topology of the network. If these values are not set correctly, the decision algorithms will not work correctly.

Figure 3 shows a sample network and depicts some of the routing terms. The glossary contains definitions of these and other Transport terms. In this figure, the maximum hops for node A is 4.



Node A wants to send a packet to Node D. There are three possible paths.		
PATH	**PATH COST**	**PATH LENGTH**
Ⓐ to Ⓑ, Ⓑ to Ⓒ, Ⓒ to Ⓓ	$\boxed{2}+\boxed{2}+\boxed{3} = 7^*$	**3 hops**
Ⓐ to Ⓑ, Ⓑ to Ⓓ	$\boxed{2}+\boxed{7} = 9$	**2 hops**
Ⓐ to Ⓑ, Ⓑ to Ⓕ, Ⓕ to Ⓔ, Ⓔ to Ⓓ	$\boxed{2}+\boxed{3}+\boxed{5}+\boxed{2} = 12$	**4 hops**

*7 is the lowest path cost; Node A therefore routes the packet to Node D via this path.

Figure 3 Routing Terms

2.5.1.2 Update Process - This process constructs and propagates Routing messages. A Routing message contains path cost and path length for all destinations. The update process sends Routing messages to adjacent nodes after determining that certain conditions are met. General characteristics of the update process are:

- Routing messages are sent to adjacent nodes only.
- Routing messages contain information on all destination nodes.
- Routing message transmission is event-driven with periodic backup.
- The routing update algorithm maintains a lower limit on routing line overhead.

2.5.1.3 **Forwarding Process** - This process supplies and manages the buffers necessary to support packet route-through to all destinations.

2.5.1.4 **Receive and Select Processes** - The receive process inspects a packet's route header and dispatches the packet to an appropriate Transport control component or to Network Services. The select process performs the simple table lookup to select the output line for the packet. When a destination is unreachable, select either returns the packet to the sender or discards the packet, depending on the option exercised in the packet route header.

2.5.2 Congestion Control: Transmit **Management Component** - This manages buffers by limiting the maximum number of packets on a queue for a line. Transmit management regulates the ratio of packets received directly from Network Services to route-through packets. Transmit management also checks the packet size for each packet received.

2.5.3 **Packet Lifetime Control** - This packet lifetime control component requires three processes:

1. Loop detector (Section 2.5.3.1)

2. Node listener (Section 2.5.3.2)

3. Node talker (Section 2.5.3.3)

2.5.3.1 **Loop Detector** - This process prevents excessive packet looping. It counts the number of nodes a packet has visited and removes a packet when it exceeds the visit limit.

2.5.3.2 **Node Listener** - This process determines that a minimum amount of activity has occurred between this node and an adjacent node. It also determines if the identity of the adjacent node has changed. Violations of the minimum activity audit result in the declaration that the line between the nodes is down.

2.5.3.3 **Node Talker** - This process provides the minimum activity for each adjacent node listener. It places an artificial load on the physical line so failures can be detected. The node talker and listener provide for detection of adjacent Transport halt and adjacent node identity change.

3.0 INTERFACES

This section describes the three external Transport interfaces:

1. Network Management layer interface (Section 3.1)
2. Data Link layer interface (Section 3.2)
3. Network Services layer interface (Section 3.3)

In addition, this section describes the single internal Transport interface, Transport initialization (Section 3.4).

The interfaces take the format of calls to subroutines, as follows:

```
FUNCTION (input ;  output)
```

Each call represents a specific function. An implementation is not required to code the interface as calls to subroutines.

The following symbols are used throughout the document:

< >	not equal to
<=	less than or equal to
>=	greater than or equal to
SQRT	the square root of

3.1 Network Management Layer Interface

This interface allows Network Management to control and observe the Transport layer interactively. Network Management can exert indirect control over the Transport layer via parameter changes. The following Network Management functions form a set of primitive functions that can be used to construct more complex functions.

READ NODE STATE (destination; reachability)

Returns: destination reachable

destination unreachable

This function returns information indicating whether or not a packet can get to a destination.

READ NODE TRANSPORT-STATE (; Transport-state)

Returns: Transport is initialized

Transport is terminated and requires problem correction and initialization

This function allows the user to observe whether Transport has been initialized or terminated.

READ NODE HOPS (destination; hops)

Returns: 0 for self

1 for an adjacent node

n where 2 <= n <= [maximum hops] for a reachable destination

m where m < 0 for an unreachable destination

This function returns the number of hops between self and destination along the path with the shortest length.

READ NODE COST (destination; cost)

Returns: n where 1 <= n <= [maximum path cost] for a reachable destination

m where m < 0 for an unreachable destination

This function returns the sum of the line costs between self and destination along the path with the least cost.

READ NODE LINE (destination; line)

Returns: 0 for Self

n where 1 <= n <= [number of output lines] for a reachable destination

m where m < 0 for an unreachable destination

This function returns the output line used to get to the destination.

READ LINE STATE (line; state)

Return: the line state (Section 7)

This function returns the state of the line.

READ LINE COST (line; cost)

Return: the local cost to use this line

This function returns the cost assigned to the line.

READ LINE COUNTERS (line; error counters)

Returns: the values of the Transport line counters

This function returns the current values for the counters associated with this line. Transport maintains one line counter per physical link for each of the line counters, defined in Appendix E. The line counters are as follows:

- Transmit ("route through") packets received
- Transmit packets sent
- Arriving packets received (output packets from other nodes addressed to this node)
- Departing packets sent (input packets from this node addressed to other nodes)
- Transit congestion loss
- Line down
- Initialization failure

READ LINE BLOCKSIZE (line; blocksize)

Return: the Data Link blocksize

This function returns the blocksize that the Data Link layer requires.

READ NODE TYPE (; type)

Returns: 0 routing

1 nonrouting

2 Phase II

This function returns the adjacent node type.

READ NODE MAXIMUM HOPS (; maximum hops)

Return: the current value of the maximum hops

This function returns the current value for the maximum hops parameter. Refer to the glossary for a comprehensive definition of maximum hops.

READ NODE MAXIMUM COST (; maximum cost)

Return: the current value of the maximum path cost

This function returns the current maximum path cost for the Transport module. Refer to the glossary for a comprehensive definition of maximum cost.

READ NODE MAXIMUM ADDRESS (; maximum address)

Return: the current value set for maximum address

This function returns the highest number of any node that can be addressed by this node.

READ NODE MAXIMUM LINES (; maximum lines)

Return: the current value of maximum lines

This function returns the maximum number of lines that the Transport module can support.

READ NODE ROUTING VERSION (; routing version)

Return: the current routing version

This function returns the current routing version, ECO, and user ECO.

READ NODE ROUTING TIMER (; routing timer)

Return: the current value of the routing timer

This function returns the current value of the routing timer that causes a routing message to be generated.

READ NODE MAXIMUM VISITS (; maximum visits)

Return: the current value for maximum visits

This function returns the current value for the routing parameter that determines the maximum number of nodes a packet can visit. Refer to the glossary for a comprehensive definition of maximum visits.

READ NODE COUNTERS (; node counters)

Returns: the current values of the counters associated with the node

This function returns the current values of the node counters maintained by Transport. Appendix E describes these counters in detail. The counters are as follows:

- Node unreachable packet loss
- Aged packet loss
- Node out-of-range packet loss
- Oversized packet loss
- Packet format error
- Partial routing update loss
- Verification reject

SET LINE STATE (line, ON or OFF; status)

Returns: error value if line nonexistent

success

A line set to ON causes the issuance of a start to the Data Link layer. A line set to OFF causes the issuance of a stop to the Data Link layer.

SET LINE COST (line, positive value; status)

Returns: error value if line nonexistent or cost value nonpositive or beyond maximum cost

success

This function changes the line cost the routing algorithm uses.

SET NODE ROUTING PARAMETER (routing parameter value; status)

Value: One of the following routing parameters, described above with the READ functions:

- Maximum hops
- Maximum cost
- Maximum address
- Maximum lines
- Maximum visits
- Routing timer

Returns: error message if value improper or routing parameter non-existent

success

SET NODE STATE (transport identification, ON or OFF)

Returns: none

With the ON parameter, this function forces Transport to initialize all its data bases, and sets the line to an appropriate state.

With the OFF parameter, this function forces a Transport termination, issues a stop for each line, and places an OFF event in Transport's internal event queue. Appendix E describes events. Transport layer termination has the following effects:

1. The decision process cannot process any events.

2. The update process cannot send any routing messages.

3. Network Services interface requests cannot be processed.

4. The select and receive processes cannot forward packets.

ZERO COUNTERS

Returns: none

This function sets line and node counters associated with Transport to zero.

READ EVENT (; event)

Return: the oldest event in Transport's internal event queue

Transport maintains an internal event queue into which it places events. Appendix E describes events. This function reads the oldest event in the queue.

CLEAR EVENTS

Returns: none

This function clears all events from Transport's internal event queue.

3.2 Data Link Layer Interface

This interface, between Transport's initialization layer and the Data Link layer, consists of commands to and responses from the Data Link layer. The interface supports the exchange of data, control, and error information. Data is information to be sent or received by the Data Link layer protocol. Its description usually consists of a starting buffer address and a length or character count, or a chain of addresses and counts. The control information starts and stops the protocol. The error information reports line conditions. The functions of the interface, described as calls, are as follows:

TRANSMIT (line, buffer)

Returns: none

This function gives a message to the Data Link layer for transmission.

CHECK TRANSMIT BUFFER (buffer)

Returns: buffer is queued

buffer is returned to Transport

This function returns information about the transmit buffer.

INITIALIZE LINK (line)

Returns: none

This function causes the Data Link protocol to initialize the link.

STOP LINK (line)

Returns: none

This function halts the Data Link operation on the specified line.

STATUS (line; status)

Returns: off

running

initializing

This function returns the Data Link state of the line. If the Data Link modules provide additional states (for example, a maintenance state), they are treated as the OFF state.

STATUS ERROR COUNTERS (line; error counters)

Returns: error counter values (value,status)

This function returns the values of the Data Link error counters, and notification if error thresholds have been exceeded.

SUPPLY RECEIVE BUFFER (buffer; status)

Returns: buffer accepted

buffer rejected

This function provides an empty buffer to the Data Link modules for receipt of the next sequential message.

CHECK RECEIVE BUFFER (buffer; status)

Returns: no packet received

packet received, buffer returned

This function returns the above information about the receive buffer.

3.3 Network Services Layer Interface

This interface, between the Network Services layer and the Transport layer, consists of commands to and responses from the Transport layer. The commands and responses exchange data and control information.

Data is information that Transport sends or receives. The data description is a destination address, source address, buffer address and length of data. Destination and source addresses are two-byte integer numbers in the range of 1 to the number of nodes in the network. Transport uses node addresses only, not node names, which are resolved at a higher layer.

Control information starts or stops transmission and reception of data and regulates the data flow to a reachable destination.

The functions of the interface, described as calls, are as follows:

TRANSMIT (source, destination, return flag, channel, buffer)

Returns: buffer is queued

buffer is not queued and is returned to Network Services

This function sends a packet.

The return flag indicates whether or not Network Services wants the packet returned if the destination is unreachable or becomes unreachable before Transport can deliver the packet. If the flag is set to "true" (Boolean), Transport attempts to return the contents of the buffer to Network Services as a "received packet." If the flag is not set, Transport discards the packet. Transport may return the buffer after Network Services issues the CHECK TRANSMIT BUFFER call (described next).

Channel, selected by Network Services, is either unspecified or a valid line number. For line level loopback testing, Network Services must specify a line number; otherwise, Transport determines the channel.

CHECK TRANSMIT BUFFER (buffer)

Returns: buffer queued

buffer returned to Network Services

This function checks the status of a previously queued transmit buffer. It returns the buffer to Network Services after any of the following:

- The buffer is copied into a Transport buffer.
- The packet is transmitted.
- The packet is discarded because the destination is unreachable and the return flag is not set.
- The packet contents are transferred to a receive buffer because the destination is unreachable and the return flag is set.

SUPPLY RECEIVE BUFFER (buffer)

Returns: buffer queued for receive by Transport

buffer not queued for receive by Transport

This function queues a receive buffer to Transport.

CHECK RECEIVE BUFFER (source, destination, channel, buffer)

Returns: buffer remains queued by Transport

buffer returned to Network Services with source and destination node addresses (buffer contains a normal packet)

buffer returned to Network Services (buffer contains a "return to sender" packet -- NSP Functional Specification)

This function checks the status of a previously queued receive buffer. It returns the buffer if the packet was received or if the node is unreachable and the return flag is set. The channel variable returns a valid line number (or a value for an internal link) for each received packet.

OPEN (source)

Returns: none

This function identifies a Network Services module as an active Transport user at a node and defines its node address. An open must occur before Network Services can transmit or receive. Source is a node address.

CLOSE (source)

Returns: none

This function removes a Network Services module as a Transport user at a node. Source is a node address.

3.4 Transport Initialization Interface

This interface, between the Transport control sublayer and the Transport initialization sublayer, supports the routing events defined in Section 4.4.2. The interface consists of commands to and responses from the Transport initialization sublayer.

TRANSMIT (line, buffer)

Returns: none

This function transmits a buffer containing a packet.

CHECK TRANSMIT BUFFER (buffer; status)

Returns: buffer queued

buffer returned to user

This function polls a buffer containing a packet that has been sent with the TRANSMIT function. If the packet has been transmitted, the buffer is returned to Transport control. If the packet has not yet been transmitted, a message is returned indicating that the buffer is queued.

STATUS (line; status)

Returns: off (line rejected)

initializing

line accepted by Transport initialization

running; current value of line cost

This function returns the status of the line. The off, initializing, and running states correspond to Data Link layer line states (see Section 3.2).

REINITIALIZE (line)

Returns: none

This function turns the line off and initializes the link in a manner that prohibits race conditions from occurring.

SUPPLY RECEIVE BUFFER (buffer; status)

Returns: buffer accepted

buffer rejected

This function provides a receive buffer to Transport initialization so that it can receive a packet.

CHECK RECEIVE BUFFER (buffer; status)

Returns: no packet received

packet received, buffer returned

This function polls the status of a buffer that Transport control has just supplied with the SUPPLY RECEIVE BUFFER function. Upon receiving a packet into the buffer, Transport initialization returns the buffer to Transport control.

SUPPLY LINE UP COMPLETE (line)

Returns: none

This function informs Transport initialization that the decision process (Section 4.4) recognizes that a line is up. (The process has completed its line up event algorithm.)

SUPPLY LINE DOWN COMPLETE (line)

Returns: none

This function informs Transport initialization that the decision process recognizes that a line is down. (The process has completed its line down event algorithm.)

4.0 DETAILED ROUTING SPECIFICATION

As noted in Section 2.5.1, the routing function consists of the following data bases and processes:

- Routing data base
- Forwarding data base
- Decision process
- Update process
- Forwarding process
- Select process
- Receive process

The relationship among these components is illustrated in Figure 2 (Section 2).

This section specifies each of these components in detail. To facilitate explanation, symbols represent parameters available to each routing node. Table 1 describes these symbols.

Table 1
Routing Parameters Available to Each Routing Node

Symbol	Definition
NN	Number of nodes in network (maximum address)
NLN	Number of lines supported by this node (maximum lines)
Maxh	Maximum hops
Maxc	Maximum cost
Maxl	Maximum cost assignable to a line
Infh	Infinite path length
Infc	Infinite path cost
Tl	Background frequency timer; maximum time period for exchanging routing messages with adjacent node
T2	Rate control frequency timer: minimum time period before another routing message can be sent.
q	Designation of undefined line

Appendix C contains required and suggested settings for routing parameters. The following relationships exist between the above parameters:

```
NN   >= actual maximum network address
Maxh >= actual maximum network path length
Maxc >= (actual maximum network path length) X (Maxl)
Infh >  Maxh
Infc >  Maxc
Tl   >> T2
```

4.1 Routing Data Base

The routing data base contains routing data summarized in Table 2. The Network Management SET NODE STATE ON function sets the data base to its initial values. This is a prerequisite to normal node operation. Table 2 also shows the initial values of the data.

Table 2
Routing Data Base

Symbol	Description	Initial Value
Hop	Network connectivity matrix	Infh
Minhop	Network minimum connectivity vector	Infh
Lcv	Line cost vector	Positive integer in range 1-Maxl
Cost	Traffic assignment matrix	Infc
Mincost	Minimum traffic assignment vector	Infc
Srm	Send routing message flags	0
Self	Self vector	0
Tid	Transport identification	*
Nty	Node type vector	*
Nid	Node identification vector	*

* This value is specified later in this section.

A description of each element of the data base follows.

Network connectivity matrix (Hop). This contains information on the path length to each destination over each line. An entry consists of Hop(i,j),

where:

Hop(i,j) Represents the path length from this Transport layer to the destination, with the following values:

Value	Meaning
0	Self
1	Adjacent node
2-Maxh	Other reachable nodes
Infh	Unreachable node

i Is a decimal integer in the range 1-NN representing a destination address.

j Is a decimal integer in the range 0-NLN representing the line to destination i.

Minimum network connectivity vector (Minhop). This summarizes the path length information contained in the Hop table. An entry consists of Minhop(i),

where:

Minhop(i) Represents the smallest path length via any line from this Transport layer to the destination node, with the following values:

Value	Meaning
0	Smallest path length to Transport users at this node
1	Smallest path length to adjacent node
2-Maxh	Smallest path length to other reachable nodes
Infh	Node unreachable

i Is a decimal integer in the range 1-NN representing a destination address.

Line cost vector (Lcv). This contains information on the perceived cost of using a line. An entry consists of Lcv(j),

where:

Lcv(j) Is a positive integer in the range 1-Maxl representing the cost associated with using line j.

j Is a decimal integer in the range 1-NLN representing a line.

Traffic assignment matrix (Cost). This contains information on the expected path cost to each destination over each line. This information determines which line to use for traffic to a destination. An entry consists of Cost(i,j),

where:

Cost(i,j) Represents the path cost from this Transport layer to the destination, with the following values:

Value	Meaning
0	Expected path cost to this node
1-Maxc	Expected path cost to other reachable nodes
Infc	Unreachable nodes

i Is a decimal integer in the range 1-NN representing a destination address.

j Is a decimal integer in the range 1-NLN representing the line to destination i.

Minimum traffic assignment vector (Mincost). This summarizes the path cost information contained in the Cost table. An entry consists of Mincost(i),

where:

Mincost(i) Represents the smallest cost from this Transport layer to the destination, with the following values:

Value	Meaning
0	Smallest cost to this node
1-Maxc	Smallest cost to other reachable nodes
Infc	Unreachable node

i Is a decimal integer in the range 1-NN representing a destination address.

Send routing message flags (Srm). The Srm flags extend permission to the update process to send a routing message to an adjacent node. The update process determines the actual propagation rules. An entry consists of Srm(j),

where:

Srm(j) Indicates whether or not a routing message should be sent to the adjacent node, with the following values:

Value	Meaning
0	Do not send routing message.
1	Send routing message.

j Is a decimal in the range 1-NLN representing the line on which to send the message to the adjacent node.

Self vector. This contains the node address identity of the user of Transport at this node. An entry consists of Self(i),

where:

Self(i) Contains the node address identity of a Transport user.

(i) Is a decimal in the range 1-NN representing the node address of a Transport user.

Transport Identification (Tid). This contains the Transport initialization value from the Network Management **SET NODE STATE ON** call. The value is a two-byte integer.

Node type vector (Nty). This contains the node type (routing, nonrouting, or Phase II) of an adjacent node. An entry consists of Nty(j),

where:

Nty(j) Contains the value received from the Transport initialization message (Section 8) most recently received. The value indicates whether the adjacent node is routing, nonrouting, or Phase II.

j Is a decimal integer in the range 1-NLN representing the line from this node to the adjacent node.

Node identification vector (Nid). This represents the node identification of an adjacent node. An entry consists of Nid(j),

where:

Nid(j) Contains the value received from the SRCNODE field of the Transport initialization message most recently received.

j Is a decimal integer from 1-NLN representing the line from this node to the adjacent node.

Appendix D contains some routing data base examples.

4.2 Forwarding Data Base

The information in this data base indicates whether or not a destination is reachable and , if reachable, what line to use to get there. This data base consists of two vectors, as follows:

Reachability vector (Reach). This indicates whether or not the destination is reachable. An entry consists of Reach(i),

where:

Reach Is a reachability value consisting of:

Value	Meaning
1	Reachable
0	Unreachable

i Is a decimal in the range 1-NN representing the destination node address.

Output lines (OL). This identifies the line on which to forward a packet to a destination. An entry consists of OL(i),

where:

OL(i) Represents the line to be used when forwarding a packet to destination i. OL contains one of the following values:

Value	Meaning
0	Deliver to a Transport user at this node
1-NLN	A line on which to forward a packet

i Is a decimal in the range 1-NN representing a destination node address.

4.3 Data Base Protection

A memory failure, a corrupted routing message, or a software error can corrupt a routing data base. Such corruptions cause a transient disruption of packet delivery. If the corruption is transient, the routing data bases stabilize to correct routes. If the corruption is continuous, the routing data bases remain in a transient incorrect state.

Two conditions are necessary for self-stabilization:

1. Transport must periodically propagate routing messages.

2. Column 0 of the Cost and Hop matrices (in other words, the values relating to self) cannot be corrupted.

4.4 Decision Process

The decision process selects paths and maintains the routing and forwarding data bases. This process consists of a controller and two algorithms. The following events serve as input to the decision process:

- Line down
- Line up
- Routing message received
- Operator command to change line cost
- Operator command to change parameters Maxh or Maxc
- Operator command to identify Transport user (owner)
- Operator command to remove Transport user
- Timer expiration

The decision process produces the following output:

- Modifications to the routing data base
- Modifications to the forwarding data base

4.4.1 **Decision Controller** - The controller performs the following functions:

- Performs buffer management necessary to receive routing messages.

- Supports Transport initialization (Section 7).

- Checks for valid routing message. A valid routing message has:

    1. A valid checksum

    2. A valid Transport control header (Section 8)

  If the routing message is not valid, then a line down event is generated, and the routing message is discarded and recorded. A valid routing message is processed through end of message or end of table. If the routing message is too long (that is, length beyond end of table), then the controller examines the overrun. The data in the overrun portion of the message must contain values corresponding to Infh in the hop fields and to Infc in the cost fields. If not, then the controller finds the highest existing NN and records the error.

- Updates the forwarding data base

4.4.2 **Decision Algorithms** - This module supports two algorithms: one for connectivity, the minimum hop algorithm; and the other for traffic assignment, the minimum cost algorithm. The minimum hop algorithm changes the reachability vector, Reach. The minimum cost algorithm changes the output line vector, OL, and performs a Boolean "or" operation on the Reach vector.

The following subroutines represent the decision algorithms. They are followed by a description of the action that the decision module takes for each event received.

In both the subroutines and the list of event actions, the symbols i and j have the following meanings:

i destination address, a unique integer from 1-NN.

j line number, a unique integer from 1-NLN.

Subroutine: Rowmin(M,I,minimum)

```
Matrix M
Integer minimum
Integer I
     minimum = 'big number'
     For each column J from 0 to NLN do
     BEGIN
          IF (M(I,J) < minimum ) then
               minimum = M(I,J)
               OL(I) = J
          ENDIF
     END
```

The routine above determines the minimum for row I of Matrix M and stores line number OL(I).

Subroutine: Minimize(I,M,V,P1,P2)

```
Integer I
Matrix M
Vector V
Parameters P1, P2
     Rowmin(M,I,minimum)
     IF (minimum > P1) then minimum = P2
     IF (V(I) < > minimum ) then
          V(I) = minimum
     ENDIF
```

This routine determines entries for vector V, containing the minimum of each row of matrix M.

Subroutine: Routes

```
INTEGER OLD_HOP, OLD_COST
For each row i from 1 to NN do
Begin
OLD_HOP = MINHOP (i)
OLD_COST = MINCOST (i)
Minimize (row i, Hop, Minhop, Maxh, Infh)
Minimize (row i, Cost, Mincost, Maxc, Infc)
If (Minhop(i) = Infh or Mincost(i) = Infc)
  Then
    Begin
    Reach(i) = False
    Minhop(i) = Infh
  Mincost(i) = Infc
    End
  Else Reach(i) = True
Endiff
If(MINHOP(i) < > OLDHOP or MINCOST(i) < > OLDCOST
Then set each Srm FLAG
End
```

This routine determines the reachability and output line for each destination and sets the routing flags if appropriate. The order of invocations of Minimize in this subroutine is very important. If some nodes in a network perform them in one order, and other nodes in another order, then incorrect (circular) routing could result.

Subroutine: Check

```
BEGIN
     For each I from 1 to NN
     IF Self(i) = i
          Check that Hop (Self(i),0) = 0 and Cost
          (Self(i),0) = 0
     ELSE
          Check that Hop(i,0) = Infh and Cost(i,0) = Infc
     ENDIF
     IF either check fails, then
          Terminate Transport
     ENDIF
   IF both checks are successful, then
          Exit
     ENDIF
END
```

This routine detects any corruption of column 0 in the Hop and Cost matrices.

Event	Action
A. line j down	0. Call Check. 1. Set each entry in column "j" of Hop matrix to Infh. 2. Set each entry in column "j" of Cost matrix to Infc. 3. Call Routes. 4. Supply "line down complete" to initialization sublayer.

(continued on next page)

Event	Action
B. line j up	0. Call Check. 1. IF (Nty(j) = "small"), then "k" is neighbor at end of line "j" Hop(k,j) = 1 IF Lcv(j) is not positive, then Transport terminates. Cost(k,j) = Lcv(j) ENDIF ENDIF 2. Set each Srm flag. 3. Call Routes. 4. Supply "line up complete" to initialization sublayer.
C. routing message received on line j	0. Call Check. 1. Copy hop subfield of routing message onto column "j" of Hop matrix. 2. Add 1 to each entry in column "j" of Hop matrix. 3. Copy cost subfield of routing message onto column "j" of Cost matrix. 4. If Lcv(j) is not positive, then Transport terminates. Otherwise, add Lcv(j) to each entry in column "j" of Cost matrix. 5. Call Routes.
D. line j cost change	0. Call Check. 1. Calculate the difference between the new cost and the old cost for line j. Note that the new cost and the old cost must both be positive, otherwise Transport terminates. 2. Add this difference to each entry in column "j" of Cost matrix. 3. Call Routes.
E. Maxh or Maxc change	0. Call Check.

(continued on next page)

Event	Action
F. Open(k) issued to identify Transport user k	0. Call Check. 1. Self(k) = k 2. Hop(k,0) = Cost(k,0) = 0 3. Call Routes.
G. Close(k) issued to remove Transport user k	0. Call Check. 1. Self(k) = q 2. Hop(k,0) = Infh 3. Cost(k,0) = Infh 4. Call Routes.
H. Timer expires	0. Call Check. 1. Set each Srm flag. 2. Call Routes.

4.5 Update Process

The update process propagates routing messages and determines their content. It consists of an algorithm and a format module. The update process accepts the following as input:

- The minimum hop vector, Minhop
- The minimum cost vector, Mincost

The update process produces a routing message for an adjacent node as output.

4.5.1 **Update Algorithm** - The following algorithm supports routing message propagation. The algorithm is based on the send routing message flags, Srm, and the use of two timers, T1 and T2 per line. For further discussion on suggested T1 and T2 settings, see Appendix C.

A routing message is sent to an adjacent node when:

A buffer is available from the quota given to the routing process for update use.

and either

T1 has elapsed since the last transmission of a routing message on this line.

or

The Srm flag is set for this line and at least T2 has elapsed since the last transmission of a routing message.

4.5.2 Update Formatter - This module constructs the routing message from Minhop and Mincost vectors.

4.6 Forwarding Process

The forwarding process supplies and manages the buffers necessary for route-through. Packets are discarded if buffer thresholds are exceeded.

4.7 Receive Process

The receive process receives packets from the Data Link layer. It then passes the packet to the appropriate process, as follows:

Packet Type	Process
Routing message	Decision process
Hello message	Node Listener process
Packet for Self	Network Services layer
Packet for other destination	Forwarding process

4.8 Select Process

The select process examines the destination field in the packet route header, reads the forwarding data base, and determines the appropriate line on which to transmit the packet. If the destination is unreachable or if the address exceeds the range of the forwarding data base, then the select process either returns the packet to the sender or discards the packet, depending on the indicated option in the packet route header. If the packet is to be returned to the sender, then the "return to sender request" bit in the packet route header is turned off and the "return to sender" bit is turned on. Also, the source and destination fields are exchanged. If the destination is unreachable and the "return to sender" bit is on, then the packet is discarded.

5.0 DETAILED CONGESTION CONTROL SPECIFICATION

The transmit management subroutine handles congestion control. Transmit management consists of the following components:

- **Square root limiter.** Reduces buffer occupancy time per packet by using a square root limiter algorithm (Appendix F). The square root limiter also queues packets for an output line, and prevents buffer deadlock by discarding packets when the buffer pool is exhausted. Section 5.1 specifies the square root limiter process.

- **Input packet limiter.** Limits input packet traffic when necessary to ensure that transit packets are not rejected. An input packet (called an arriving packet by Network Management) is a packet from Network Services at this node. A transit packet is a packet from another node to be routed through to another destination node. (Section 5.2)

- **Flusher.** Flushes packets queued for a line that has gone down. (Section 5.3)

- **Packet size checker.** Resolves differences between packet size and Data Link receive blocksize. (Section 5.4)

5.1 Square Root Limiter

The square root limiter discards a transit packet or rejects an input packet when the output line queue exceeds the discard threshold, Ud. Ud is given as follows:

Ud = NB/ SQRT(NLN)

where:

NB = Number of Transport buffers for all output lines.

NLN = Number of active output lines.

An algorithm to calculate the square root limit, Ud, appears in Appendix F. Using this algorithm reduces switch congestion and allows for increased buffer availability by reducing buffer occupancy per packet.

5.2 Input Packet Limiter

The input packet limiter first distinguishes between input packets and transit packets. It then imposes a limit on the number of buffers that input packets can occupy on a per line basis. In times of heavy load, input packets may be rejected while transit packets continue to be routed. This is done because input packets have a relatively short wait, whereas transit packets, if rejected, have a long wait -- a retransmission period.

The input packet limiter accepts as input:

- A packet received from Network Services

- A transmit complete from the Data Link layer for a Network Services packet

The input packet limiter produces the following as output:

- Packet accepted
- Packet rejected
- Modifications to input counter

There is an input counter, N, and an input packet limit, IPL, for each active output line. Each N is initialized to 0. Each IPL is initialized to the number of buffers necessary to prevent the line from idling.

5.3 Flusher

The flusher accepts a packet and checks the line state. If the line state is RUN, the flusher accepts the packet and passes it to the parameter resolver component. If the line state is not RUN, then the flusher discards all packets queued on this line.

5.4 Packet Size Checker

The packet size checker checks the size of each packet that it is about to queue on an output line. This includes packets from both the Network Services layer at this node and the Data Link layer coming from other nodes. When the packet size exceeds the Data Link receive blocksize (established during Transport initialization), the packet size checker discards the packet and records the event.

6.0 DETAILED PACKET LIFETIME CONTROL SPECIFICATION

The packet lifetime control component consists of the following:

- Packet lifetime control data base (Section 6.1)
- Node listener process (Section 6.2)
- Node talker process (Section 6.3)
- Loop detector process (Section 6.4)

6.1 Packet Lifetime Control Data Base

The packet lifetime control data base contains the following parameters:

Maxv The maximum number of nodes that a packet may visit.

T3 The hello frequency timer. It supports the periodic exchange of hello messages. Hello messages are packet lifetime activity packets used for dead node recognition.

T4 The listening period timer. If no packets or Transport control messages are received in this period, the adjacent node is considered down.

6.2 Node Listener Process

The node listener process detects node failures. The process consists of a controller and an algorithm module. The node listener accepts the following as input:

- Hello message received
- Any message received
- Timer pulse received

The node listener process produces the following as output:

- Modifications to the line data base
- Modifications to the packet lifetime data base

6.2.1 **Node Listener Controller** - The node listener controller manages the buffers necessary for receiving hello messages. It also checks for valid hello messages. A valid hello message contains a valid Transport control header and valid data. If the hello message is not valid, then a line down event is generated, and the hello message is recorded and discarded.

6.2.2 **Node Listener Algorithm** - The node listener algorithm determines when the adjacent node is no longer talking. Such a node is considered down. Consequently, the line is reinitialized. The following table describes the algorithm for handling each node listener event.

Event	Action
A. Hello message or any message received.	1. Reset T4 2. Check Image data
B. Timer pulse	1. Decrement T4 2. IF (T4 <= 0), then Reset T4 Set line state to "line rejected." ENDIF

6.3 Node Talker Process

The node talker propagates hello messages. It sends a hello message to an adjacent node when:

- T3 has elapsed since the last transmission of any message.
- A line has come up.

The node talker can be interlocked with the decision and update processes. When either decision or update fails, then the node talker ceases.

6.4 Loop Detector Process

The loop detector limits the number of nodes that a packet can visit. It increments the node visit field in the packet route header by one. (Section 8 describes the packet route header. The node visit field is used for counting the number of nodes visited by this packet.) The loop detector discards the packet if this number exceeds the maximum node visit limit, Maxv. Note that the parameter Maxv must always be greater than or equal to the parameter Maxh.

The following table describes the algorithm the loop detector executes when it receives a packet.

Event	Action
A. Packet received	1. Add 1 to node visit field in packet route header. 2. IF ((node visit > Maxv) and ("return to sender" is not set)), then Discard packet and record ENDIF IF ((node visit > 2 * Maxv) and ("return to sender" is set)), then Discard packet and record ENDIF

7.0 TRANSPORT INITIALIZATION SUBLAYER

Transport initialization is a start-up procedure between two adjacent nodes. The procedure involves exchanging Transport Initialization messages and possibly Transport Verification messages. This exchange identifies the nodes to each other and provides additional node information. Section 8 describes the messages. This section describes:

- The Transport initialization line states (Section 7.1)
- The Transport initialization line events (Section 7.2)
- The Transport initialization operation and message requirements (Section 7.3)
- The Transport initialization state table and diagram (Section 7.4)

7.1 Transport Initialization Line States

The Transport initialization line states are:

(Symbol) State	Description
(RU) RUN	Transport can use the line to transmit packets between two nodes.
(LR) LINE REJECTED	The line is degraded. To avoid excessive packet delay the line will be declared down. The routing decision process has not yet processed a line down event.
(DS) DATA LINK START	The line is undergoing Data Link layer initialization.
(TI) TRANSPORT INITIALIZE	The line has successfully undergone Data Link initialization and is waiting to receive a Transport Initialization message.
(TV) TRANSPORT VERIFY	A valid Transport Initialization message has been received for this line and the line requires verification.
(TC) TRANSPORT COMPLETE	Transport has completed a valid exchange of Transport Initialization and possibly Transport Verification messages.
(OF) OFF	Transport cannot use the line. The routing decision process has not yet processed a line down event.
(HA) HALT	Transport cannot use the line. A line down event is required.

7.2 Transport Initialization Line Events

The Transport initialization line events are as follows:

(Symbol)	Description
(nti)	Transport received a valid new Transport Initialization message.
(ntv)	Transport received a valid new Transport Verification message.
(oti)	Transport received a valid NSP Node Initialization message.
(tt)	Transport timed out.
(sc)	Transport received a start cumplete notification (in other words, a transition from the initializing state to the running state) from the Data Link layer.
(ste)	Transport received a start notification (in other words, a transition from any state to the stop state) or threshold error notification from the Data Link layer.
(opo)	Operator turned line on.
(opf)	Operator turned line off.
(im)	Transport received an invalid Transport Initialization message or an unexpected message.
(rc)	Transport received a reject complete from the line rejection component of the line monitor.
(ldc)	Transport initialization received a line down complete event from the decision process in the Transport control sublayer.
(luc)	Transport initialization received a line up complete event from the decision process in the Transport control sublayer.

When the Data Link layer has initialized, a timer starts. If the timer expires before the line accepted state is reached, then the line is reinitialized. If the timer expires after the line accepted state is reached, then the timer is ignored.

7.3 Transport Initialization Operation and Message Requirements

Transport initialization performs the following actions:

1. Issue reinitialize command to the Data Link layer and start timer.
2. Issue stop to the Data Link layer.
3. Send a valid Transport Initialization message.
4. Send a valid Transport Verification message.

A valid Transport Initialization message has the following characteristics:

- A valid Transport control header with node address less than or equal to this node's NN
- A received Data Link blocksize greater than or equal to the maximum (routing message size, hello message size, 246)
- An acceptable routing version

A valid Transport Verification message has a value that agrees with the function value.

7.4 Transport Initialization State Table and Diagram

The following table shows all the possible state transitions from Transport's viewpoint at a single node. It also shows the events that cause the state changes and the actions Transport initialization takes, if any, upon the occurrence of an event. The numbers in the "actions" column correspond to those in the list of actions in Section 7.3.

Table 3
Transport Initialization State Table

This table shows each possible new state and action relating to the occurrence of each event in each state. The actions are shown by a slash (/) followed by the number of the action. A dash (-) signifies no action. Section 7.3 describes the actions by number.

Event	Old State							
	RU	LR	DS	TI	TV	TC	OF	HA
nti	LR/-	LR/-	DS/-	*	DS/1	DS/1	OF/-	HA/-
ntv	LR/-	LR/-	DS/-	DS/1	TC/-	DS/1	OF/-	HA/-
oti	LR/-	LR/-	DS/-	RU/-	DS/1	DS/1	OF/-	HA/-
tt	RU/-	LR/-	DS/1	DS/1	DS/1	TC/-	OF/-	HA/-
sc	LR/-	LR/-	TI/3	DS/1	DS/1	DS/1	OF/-	HA/-
ste	LR/-	LR/-	DS/1	DS/1	DS/1	DS/1	OF/-	HA/-
opo	RU/-	LR/-	DS/-	TI/-	TV/-	TC/-	LR/-	DS/1
opf	OF/2	OF/-	HA/2	HA/2	HA/2	HA/2	OF/-	HA/-
im	LR/-	LR/-	DS/-	DS/1	DS/1	DS/1	OF/-	HA/-
rc	LR/-	LR/-	DS/-	TI/-	TV/-	DS/1	OF/-	HA/-
ldc	RU/-	DS/1	DS/-	TI/-	TV/-	TC/-	HA/-	HA/-
luc	RU/-	LR/-	DS/-	TI/-	TV/-	RU/-	OF/-	HA/-

* NOTE

There are four possible neW state/action sets for this transition, as follows:

1. Action: 4; New state: TV; Verification requested in received message; verification required by this node.

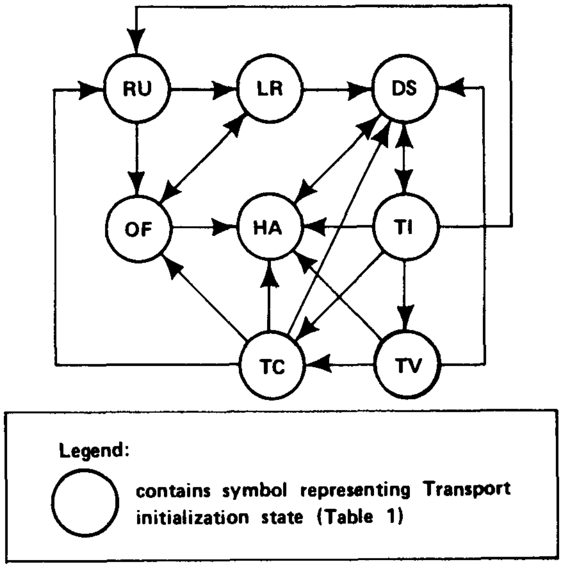
2. Action: 4; New state: TC; Verification requested in received message; verification not required by this node.

3. Action: -; New state: TV; Verification not requested in received message; verification required by this node.

4. Action: -; New state: TC; Verification not requested in received message; verification not required by this node.

The routing decision process generates line down events in the states LR and OF. It generates a line up event in the state TC.

Figure 4, following, shows the Transport state transitions.


RU
LR
DS
OF
HA
TI
TC
TV
Legend:
contains symbol representing Transport initialization state (Table 1)

Note: These state transitions are not guaranteed.

Figure 4 Transport State Transitions

7.5 Line Rejection

The line rejection subroutine determines the sustained unusability of a line. Normal line outages, impulse noise, and so on do not force a line's rejection. Instead, this subroutine seeks the continued unavailability of a line.

The line rejection subroutine monitors a line's effective throughput. A line is rejected when this throughput falls below a threshold. A line can also be rejected by the initialization process through the receipt of error threshold reports.

7.5.1 Line Rejection Parameters and Variables - The line rejection parameters and variables reside in the line data base. The parameters are as follows:

Tpt The number of packets that must be transmitted in a given interval. If this throughput is not met and the line is busy then the line is reinitialized.

T5 The interval in which Tpt packets must be sent.

The line rejection variables are as follows:

Pc The packet counter vector, which counts the number of packets sent during the last period. An entry, Pc(j), represents the number of completed transmissions for line "j" in this last period.

Qa The actual queue size vector. An entry, Qa(j), represents the actual queue size for line "j" at the time of the last observation. Each entry can assume any integer value from 0 to Ud, the discard threshold (Section 5.1).

Op The time interval between successive observations of a line for line load estimations.

7.5.2 **Line Rejection Algorithm** - The rejection algorithm attempts to maintain a minimum throughput for a line as follows:

```
If line queue is non-empty
     Send Tpt packets in T5 seconds
Else reinitialize the line
Endif
```

Execute the following algorithm every Op seconds:

```
Decrement T5
If (T5 <= 0) Then
     If (Pc > 0) Then
          Issue reject complete, rc.
     Else
          Reset T5
          If (Qa < > 0) then
               Pc = 0
          El Pc = Tpt
          Endif
     Endif
Else (T5 > 0)
     If Pc = 0 and Qa < > 0 Then
           Reset T5
           Pc = Tpt
     Endif
Endif
```

Whenever a transmission is complete, execute the following algorithm:

```
Remove packet from queue.
If (Pc > 0) Then
     Pc = Pc - 1
Endif
If (Qa = 0) Then
     Pc = 0
Endif
```

8.0 MESSAGES

This section describes the message formats of the Transport protocol. There are two types of Transport messages:

- Packet route header -- This is used for NSP segments, which may require forwarding (Section 8.2).
- Transport control -- These control Transport routing and initialization functions. They do not require forwarding (Section 8.3).

8.1 Message Format Notation

The following notation is used to describe the messages:

FIELD (LENGTH) : CODING Description of field

where:

FIELD Is the name of the field.

(LENGTH) Is the length of the field, one of:

1. A number meaning the number of 8-bit bytes.
2. A number followed by a "B" meaning the number of bits.
3. The letters "EX-n" meaning extensible field, with n being a number that specifies the maximum length of 8-bit bytes, where the high order bit of each byte indicates whether the next byte is part of the same field. A one (1) means the next byte is part of the same field. The low order seven bits are information bits.
4. The letters "I-n" meaning an image field, with n being a number that specifies the maximum length of 8-bit bytes in the image. The image is preceded by a 1-byte count of the length of the remainder of the field. Image fields are variable length and may be null (count = 0). All eight bits of each byte are information bits.

CODING Represents the type of coding used, one of:

1. B = Binary.
2. BM = Bit map. Each bit has independent meaning.
3. C = Constant.
4. NULL = Interpretation is data dependent.

Fields in separate messages with identical names are the same field and have identical meanings. All numeric values are decimal unless otherwi3e noted. All header fields and data bytes are transmitted low order or least significant bit first on the data line unless otherwise noted. Multiple byte fields are transmitted low order or least significant byte first.

8.2 Packet Route Header

The packet route header has the following format:

RTFLG	DSTNODE	SRCNODE	FORWARD

RTFLG (EX) : BM — Is the set of flags used by the routing nodes. The format of this field is as follows:

Bit:	7	6	5	4	3	2	1	0
Set to:	0	EV	0	RTS	RQR	0	1	0

EV (1B) : B — Is the routing evolution flag, where:

0 Routing node Transport route header

1 Phase II node ASCII route header

RTS (1B) : B — Is the "return to sender" flag, where:

0 Indicates this packet is not on a return trip. If the destination is unreachable, refer to RQR flag (next).

1 Indicates this packet is being returned. If its destination is unreachable, then discard this packet.

RQR (1B) : B — Is the "return to sender request" flag, where:

0 Indicates discard the packet if the destination is unreachable.

1 Indicates try to return the packet to sender if the destination is unreachable. No return guarantee is possible since no path may exist.

DSTNODE (2) : BM — Is the destination node address, a unique integer from 1 to NN. Bit 16 is reserved.

SRCNODE (2) : BM — Is the source node address, a unique integer from 1 to NN. Bit 16 is reserved.

FORWARD (EX) : BM — Is information useful in the forwarding of the message. The format of this field is as follows:

Bit:	7	6	5	0
Set to:	0	0	VISIT	

VISIT (6B) : BM — Contains the count of the number of nodes visited by this packet.

An invalid packet route header is one in which:

- SRCNODE or DSTNODE is not in the range 1 to NN.
- Reserved bits or extension bits are set.
- RQR and RTS are set.

Discard short packets (in other words, packets with less than a packet route header) and packets with an invalid route header.

8.3 Transport Control Messages

A control message has the following general format:

CTLFLG	SRCNODE	type-dependent-information

CTLFLG (EX) : BM   Is the Transport control flag, with the following format:

Bit:	7	6	5	4	3	1	0
Set to:	0	0	0	0	TYPE		1

TYPE (3B) : B   Indicates the type of Transport control message, as follows:

0 Transport Initialization message

1 Transport Verification message

2 Transport Hello and Test message

3 Transport Routing message

SRCNODE (2) : B   Is the identification of the source node's Transport, containing the value, Tid (Section 4.1). SRCNODE is less than or equal to NN (Section 4.1).

type-dependent-information   Is as described in Sections 8.3.1 - 4.

8.3.1 **Transport Routing Message** - The Routing type-dependent information is:

RTGINFO	CHECKSUM

RTGINFO : BM Is the Transport routing information, which is a sequence of 16-bit fields in the following format:

Bit:	15	14	10	9	0
Set to:	0	HOPS		COST	

HOPS (5B) : B Is the path length to a destination.

COST (10B) : B Is the path cost to a destination.

The length of this field is determined by the length of the entire message. The word position determines node reference: field 1 refers to destination number 1, and so on.

CHECKSUM (2) : B Is a check on the routing data base, as well as the message. It is a one's complement add, representing the sum of both COST and HOP, where COST and HOP are treated as a 16-bit binary value.

8.3.2 **Transport Hello and Test Message** - The Hello and Test type-dependent information is:

TEST	DATA

TEST DATA (I-128) : B Is a sequence of up to 128 bytes of data used to test the line. Each byte is 252 octal. The node listener does the checking.

8.3.3 **Transport Initialization Message** - The Transport Initialization type-dependent data has the following format:

TIINFO	BLKSIZE	TIVER	RESERVED

TIINFO (EX) : BM Is Transport information on node type and service requests, as follows:

Bit:	7	6	5	4	3	2	1	0
Set to:	0	0	0	0	0	VERIF		NTYPE

NTYPE (2B) : B Is the Transport node type:

0 Reserved
1 Reserved
2 Routing node
3 Nonrouting node

VERIF (1B) : BM Transport Verification message required if this bit is set.

RESERVED (I-64) A reserved field currently containing a count of 0.

BLKSIZE (2) : B   Is the maximum Data Link layer receive block size.

TIVER (3) : B   Is the Transport version, with the following format:

Byte 1 -- version number (1)

Byte 2 -- ECO number (3)

Byte 3 -- user ECO number (0)

8.3.4 **Transport Verification Message** - The type dependent verification information is:

FCNVAL

FCNVAL (I-64) : B   Is the function value.

APPENDIX A

ROUTES, ADDRESSES, AND NAMES

This Appendix explains the relationship between addresses and names in a network.

Transport identifies nodes in a network by unique numbers (addresses). However, it is often more convenient for users to identify nodes by an alphabetic or alphanumeric name. In addition, several users at one node may each wish to identify network nodes by different names. Moreover, users may not want to use only names that are unique within the network. Thus a problem arises as to how to bind node names to node addresses in a network.

The solution is to use names that are unique within a node (locally unique) and addresses that are unique within the network (globally unique). The following rules apply:

- Transport knows nodes only by their addresses.
- All addresses are known throughout the network (global) and are unique (an integer from 1 to NN).
- Names are assigned individually on a node-by-node basis (local) and are unique within a node.
- The local naming function (name to address directory) is not necessarily a one to one function. In other words, alias node names may be assigned to a node name as long as overall local uniqueness is preserved.

This solution has the following advantages:

- Aliases are not global. When networks merge, there is no need to change local aliases.
- It preserves the correspondence between names and addresses.
- It avoids the complex problem of maintaining duplicate copies of a distributed data base in an automatic function.
- It avoids network maintenance problems related to name and address directories. Incorrect directories affect only local users.

This solution imposes some responsibilities on network managers. The local network manager must ensure the local directories preserve uniqueness of names. The central network manager must ensure the directories preserve uniqueness of addresses.

APPENDIX B

ROUTING SUBSETS AND TOPOLOGIES

This Appendix defines routing and nonrouting nodes in terms of their composition, and outlines topological considerations that must be made when planning network configurations.

B.1 Routing and Nonrouting Nodes

Phase III nodes are of two types:

1. **Routing nodes.** These contain the full routing subset of the decision process, the update process, the forwarding process, and the select and receive modules.

2. **Nonrouting nodes.** These contain only the select and receive modules. A nonrouting node supports a single line and hence is an end node.

B.2 Topological Concepts

Network topology involves two concepts: physical connectivity and logical connectivity.

Physical connectivity defines rules for connecting network nodes by physical lines. A node's position in the network may be restricted depending on its node type (routing, nonrouting, or Phase II). Two nodes are physically connected if they are connected by a sequence of active lines.

Logical connectivity defines rules for two nodes being able to communicate. Two nodes are logically connected if they can communicate.

In a network of only Phase III nodes, any two nodes can be logically connected. In a network containing one or more Phase II nodes, some restrictions are imposed. Refer to Appendix G.

B.3 DECnet Topological Rule

In any network of routing and nonrouting nodes, physical connectivity is identical to logical connectivity.

Routing nodes can be physically interconnected in an arbitrary way. Any two physically connected routing nodes are also logically connected.

A nonrouting node can only be an end node adjacent to another nonrouting node, or a routing node.

The first case is a network of two nonrouting nodes, which are also logically connected. In the second case, the node is logically connected to any physically connected node.

B.4 Legal and Illegal Topologies

A network topology is legal if the physical connectivity satisfies the condition stated in Section B.3. Apart from an independent network of Phase II nodes, the most general legal network topology is illustrated in Figure 5. In it all routing and nonrouting nodes are logically connected. All Phase II nodes are logically connected to adjacent nodes only. Any subnetwork (possibly a single node) of a legal network is also a legal network.

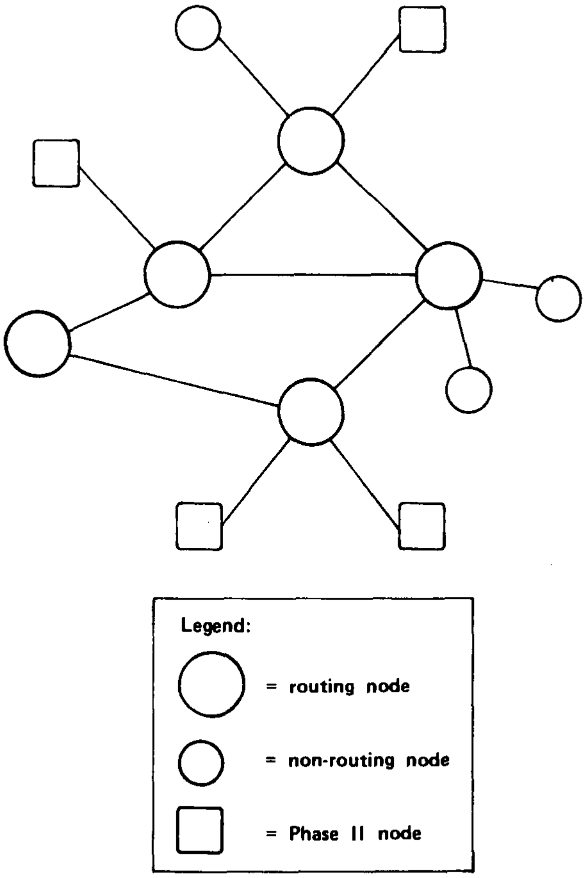


Figure 5 Example of a Legal Network Topology

A network topology may be changed intentionally or unintentionally due to a component failure. Possible network changes are the result of:

- Disconnecting a physical line.
- Shutting a node off, which disconnects all logical links.
- Opening a physical line.
- Starting a node and opening a physical line to another node.

This section discusses conditions for preserving topological legality when the network is altered.

Since any subnetwork of a legal network is also a legal network, disconnecting a line or shutting a node preserves topological legality. However, the network may be partitioned into disconnected subnetworks.

A physical line can legally be established between nodes of the same or different routing types, under the following conditions:

Routing--Routing	If nodes are not adjacent
Routing--Nonrouting	If the nonrouting node is isolated
Nonrouting--Nonrouting	If both nodes are isolated
Nonrouting--Routing	If the nonrouting node is isolated

B.5 Nonrouting Operation

This section describes the routing operation for nonrouting nodes. Section 4.4 describess the routing algorithm for routing nodes.

B.5.1 Select and Receive Modules

The node listener processes a packet upon receipt. The node listener then passes the packet or discards it, depending on whether it is a packet for self or any other message. If the packet is a routing message, the node listener discards it.

When transmitting to a node other than self, the node listener sends the packet out over the only line available. Otherwise, the node listener returns or discards the packet, depending on the packet route header option.

If a remote NSP sends a packet with "return to sender" requested, and there is no adjacent routing node, the node listener returns the packet to NSP. This happens either when the line is down or when it is up to an adjacent nonrouting node.

B.5.2 Interfaces

The NSP and Data Link interfaces are as described in Sections 3.2 and 3.3. Nonrouting nodes support the entire Network Management interface in Section 3.1 except:

READ NODE STATUS
READ NODE HOPS
READ NODE COST
READ NODE LINE
READ LINE COST
SET LINE COST

Since a nonrouting node does not perform the packet aging function, the interface does not need to support the reading or writing of the Maxv parameter.

Note that the source address in the OPEN function must be equal to the Transport identification value, Tid.

APPENDIX C

ROUTING PARAMETER SETTINGS

The following routing parameter settings are required:

1. Infh = 31.
2. Infc = 1023.

Some suggested routing parameter settings are as follows:

1. Maxh = 16.
2. Maxc = 400.
3. Maxl = 25.
4. Tl = 1 minute when line bandwidth x < 9600 bits/second
   15 seconds when x >= 9600 bits/second
5. T2 = Tl /30

Some suggested packet lifetime control parameters are as follows:

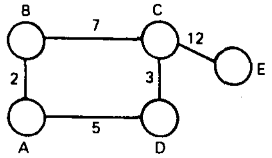
1. Maxv = Maxh + k where 1 < k <= Maxh
2. T3 = 3 seconds
3. T4 = 30 seconds

APPENDIX D

ROUTING EXAMPLES

This Appendix provides some examples of routing data bases and the effect of topological changes on the routing data bases. The examples are specific to a given topology and selected nodes in that topology. The examples do not illustrate actual routing timings, as these vary with line and node characteristics.

Figure 6a shows a routing data base in a static topology. Figure 6b continues with a similar network configuration, but shows a line coming up and the effects of this event on the routing data base. In this example all line costs are assumed to be 1 (one) and therefore do not affect the routing message.



A, B, C, D, E = routing nodes
2, 3, 5, 7, 12 = line costs (same at each end of the line)
DA, DC, DE, AB, AD = lines from nodes D and A; line DE is down

Let infh = 31.
infc = 1023.
maxh = 4.
maxc = 150.

The following diagram shows the routing data bases at nodes D and A:

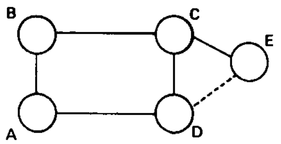
Node D

HOP MATRIX			MINHOP	COST MATRIX			MINCOST	OUTPUT LINE
via: to:	DA	DC		via: to:	DA	DC		
A	1	3	1	A	5	12	5	DA
B	2	2	2	B	7	10	7	DA
C	3	1	1	C	14	3	3	DC
D	–	–	0	D	–	–	0	–
E	4	2	2	E	26	15	15	DC

Node A

HOP MATRIX			MINHOP	COST MATRIX			MINCOST	OUTPUT LINE
via: to:	AB	AD		via: to:	AB	AD		
A	–	–	0	A	–	–	0	–
B	1	3	1	B	2	15	2	AB
C	2	2	2	C	9	8	8	AD
D	3	1	1	D	12	5	5	AD
E	3	3	3	E	21	20	20	AD

Figure 6a Static Topology Routing Data Base Example



A, B, C, D, E = routing nodes
DA, DC, DE = lines from node D
EC, ED = lines from node E

Let Infh = 31.
Maxh = 4.

The following diagram shows the interaction between nodes D and E when link DE comes up. This diagram shows only the hop matrix and associated minhop vector. This corresponds to a network where all line costs are set to 1. In such a case, minhop is the routing message.

① LINK DE COMES UP

Node D

HOP MATRIX				MINHOP	OUTPUT LINE
to: \ via:	DA	DC	DE		
A	1	3	31	1	DA
B	2	2	31	2	DA
C	3	1	31	1	DC
D	–	–	–	–	–
E	4	2	31	2	DE

D sends the routing message [1, 2, 1, 0, 2]
to E.

Node E

HOP MATRIX			MINHOP	OUTPUT LINE
to: \ via:	EC	ED		
A	3	31	3	EC
B	2	31	2	EC
C	1	31	1	EC
D	2	31	2	ED
E	–	–	–	–

E sends the routing message [3, 2, 1, 2, 0]
to D and C.

② EFFECT OF EXCHANGE OF MESSAGE

Node D

HOP MATRIX				MINHOP	OUTPUT LINE
to: \ via:	DA	DC	DE		
A	1	3	4	1	DA
B	2	2	3	2	DA
C	3	1	2	1	DC
D	–	–	–	–	–
E	4	2	1	1	DE

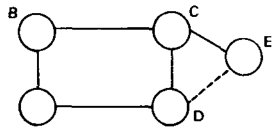
Node E

HOP MATRIX			MINHOP	OUTPUT LINE
to: \ via:	EC	ED		
A	3	2	2	ED
B	2	3	2	EC
C	1	2	1	EC
D	2	1	1	ED
E	–	–	–	–

E sends the routing message [2, 2, 1, 1, 0]
to D and C.

Figure 6b Routing Data Base -- Line Up Example

Figure 6c continues with the example from Figure 6b, but shows the routing message exchanges that occur between nodes A,B,C, and D and the adjustments to the routing data bases resulting from the line up event.



A, B, C, D, E = routing nodes
AB, AD, BA, BC, CB, CD, CE, DA, DC, DE = lines from nodes A, B, C, and D

The following diagram shows the interaction between nodes A, B, C, and D when a routing message exchange occurs. (*All line costs are assumed to be set to 1.*)

(1) Tables A and C prior to exchange:

Node A

HOP MATRIX			MINHOP	OUTPUT LINE
to: \ via:	AB	AD		
A	–	–	–	–
B	1	3	1	AB
C	2	2	2	AB
D	3	1	1	AD
E	3	3	3	AB

Node C

HOP MATRIX				MINHOP	OUTPUT LINE
to: \ via:	CB	CD	CE		
A	2	2	4	2	CB
B	1	3	3	1	CB
C	–	–	–	–	–
D	3	1	3	1	CD
E	3	3	1	1	CE

(2) A and C *receive* [1, 2, 1, 0, 1] from D as a result of the change in node D's tables as shown in Figure 5b:

Node A

HOP MATRIX			MIN HOP	OUTPUT LINE
to: \ via:	AB	AD		
A	–	–	–	–
B	1	3	1	A
C	2	2	2	A
D	3	1	1	A
E	3	2	2	A

Node C

HOP MATRIX				MINHOP	OUTPUT LINE
to: \ via:	CB	CD	CE		
A	2	2	4	2	C
B	1	3	3	1	C
C	–	–	–	–	–
D	3	1	3	1	C
E	3	3	1	1	C

(3) Tables B and D prior to exchange:

Node B

HOP MATRIX			MINHOP	OUTPUT LINE
to: \ via:	BA	BC		
A	1	3	1	BA
B	–	–	–	–
C	3	1	1	BC
D	2	2	2	BA
E	4	2	2	BC

Node D

HOP MATRIX				MINHOP	OUTPUT LINE
to: \ via:	DA	DC	DE		
A	1	3	4	1	DA
B	2	2	3	2	DA
C	3	1	2	1	DC
D	–	–	–	–	–
E	4	2	1	1	DE

(4) B and D receive [0, 1, 2, 1, 2] from A:

Node B

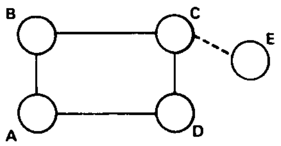
HOP MATRIX			MINHOP	OUTPUT LINE
to: \ via:	BA	BC		
A	1	3	1	BA
B	–	–	–	–
C	3	1	1	BC
D	2	2	2	BA
E	3	2	2	BC

Node D

HOP MATRIX				MINHOP	OUTPUT LINE
to: \ via:	DA	DC	DE		
A	1	3	3	1	DA
B	2	2	3	2	DA
C	3	1	2	1	DC
D	–	–	–	–	–
E	3	2	1	1	DE

Figure 6c Routing Data Base -- Message Exchange Example

Figure 6d, the final example, shows a node being isolated from the network by a line going down. Only the part of the routing data bases relating to the isolated node is shown. The algorithms that cause the changes exemplified are described in Section 4.4.2.



A, B, C, D, E = routing nodes
AB, AD, BA, BC, CB, CD, CE, DA, DC, DE = lines from nodes A, B, C, and D; line DE is down
Maxh = 4.
Infh = 31.

The following diagram exemplifies what happens in routing data bases when a node becomes isolated from the rest of the network. In this case, line CE goes down, isolating node E. Only that part of the routing data bases pertaining to node E is shown. An entry of 31 in the minhop field indicates an unreachable node. Only the element in the routing message referring to E is shown.

(1) Tables A, B, C, and D prior to link CE going down:

Node A

HOP MATRIX			MINHOP	OUTPUT LINE
via: to:	AB	AD		
E	3	3	3	AB

Node B

HOP MATRIX			MINHOP	OUTPUT LINE
via: to:	BA	BC		
E	4	2	2	BC

Node C

HOP MATRIX				MINHOP	OUTPUT LINE
via: to:	CB	CD	CE		
E	3	3	1	1	CE

Node D

HOP MATRIX				MINHOP	OUTPUT LINE
via: to:	DA	DC	DE		
E	4	2	31	2	DC

(2) Link CE goes down:

Node A

HOP MATRIX			MINHOP	OUTPUT LINE
via: to:	AB	AD		
E	3	3	3	AB

Node B

HOP MATRIX			MINHOP	OUTPUT LINE
via: to:	BA	BC		
E	4	2	2	BC

Node C

HOP MATRIX				MINHOP	OUTPUT LINE
via: to:	CB	CD	CE		
E	3	3	31	3	CB

Node D

HOP MATRIX				MINHOP	OUTPUT LINE
via: to:	DA	DC	DE		
E	4	2	31	2	DC

Figure 6d Routing Data Bases -- Example of Isolation of a Node

③ C sends 3 to B and D:

Node A

HOP MATRIX			MINHOP	OUTPUT LINE
via: to:	AB	AD		
E	3	3	3	AB

Node B

HOP MATRIX			MINHOP	OUTPUT LINE
via: to:	BA	BC		
E	4	4	4	BA

Node C

HOP MATRIX				MINHOP	OUTPUT LINE
via: to:	CB	CD	CE		
E	3	3	31	3	CB

Node D

HOP MATRIX				MINHOP	OUTPUT LINE
via: to:	DA	DC	DE		
E	4	4	31	4	DA

④ B sends 4 to A and C; D sends 4 to A and C:

Node A

HOP MATRIX			MINHOP	OUTPUT LINE
via: to:	AB	AD		
E	31	31	31	X

Node B

HOP MATRIX			MINHOP	OUTPUT LINE
via: to:	BA	BC		
E	4	4	4	BA

Node C

HOP MATRIX				MINHOP	OUTPUT LINE
via: to:	CB	CD	CE		
E	31	31	31	31	X

Node D

HOP MATRIX				MINHOP	OUTPUT LINE
via: to:	DA	DC	DE		
E	4	4	31	4	A

⑤ A sends AE 5 to B and D; C sends CE 5 to B and D:

Node A

HOP MATRIX			MINHOP	OUTPUT LINE
via: to:	AB	AD		
E	31	31	31	X

Node B

HOP MATRIX			MINHOP	OUTPUT LINE
via: to:	BA	BC		
E	31	31	31	X

Node C

HOP MATRIX				MINHOP	OUTPUT LINE
via: to:	CB	CD	CE		
E	31	31	31	31	X

Node D

HOP MATRIX				MINHOP	OUTPUT LINE
via: to:	DA	DC	DE		
E	31	31	31	31	X

Figure 6D (Cont.) Routing Data Bases-- Example of Isolation of a Node

APPENDIX E

TRANSPORT COUNTERS AND EVENTS

This Appendix specifies the Transport counters and events. These provide Network Management with a means to detect and isolate certain failures. The events and counters described in this Appendix relate generally to Transport activities, congestion, faults, topological changes, and verification (security) violations. Counters and events related to packet modification, misdelivery, or duplication, which NSP can detect, are not specified here.

In the following discussion, the term fault refers to the cause of a problem. The term error refers to a manifestation of a fault.

Transport counters and events capture sufficient information to detect and isolate single faults. Multiple faults are detected, but only isolated when it is cost effective to do so.

E.1 Source Events

A source event is any event that may result in incrementing a counter or logging an event. Several source events may cause a single counter to increment. Similarly, several source events may produce a single generic event for logging.

Source events are classified in two ways: by event manifestation and by cause. The primary classification is by event manifestation. Within each event manifestation category are subcategories of specific source events. Each of these specific source events has a particular cause. The following three tables describe source event manifestations, causes, and source events. Table 4 defines each event manifestation category. Table 5 defines each source event cause. Table 6 defines each source event along with its event manifestation category and probable cause. The abbreviations in Table 6 are defined in Table 5.

Table 4
Source Event Manifestations

Event Manifestation	**Definition**
Data Movement	This group of source events reflects the movement of data through Transport.
Congestion	This group of source events reflects the discarding of data packets due to Transport congestion.
Data Packet Discarded	This group of source events reflects the discarding of data packets due to a fault.
Message Format Error	The format of the message is in error.
Partial Routing Update Loss	This source event represents the receipt of a routing message that is too long to process so that some information from the message is discarded.
Line Down	This group of source events represents the detection of a physical link failure.
Initialization Failure	This group of source events represents a failure to initialize with an adjacent node.
Verification Reject	This source event represents the receipt of an invalid verification message.
Line Up	This source event represents the successful initialization with an adjacent node.
Node Reachability Change	This group of source events represents a change in node reachability (between reachable and unreachable).

Table 5
Source Event Causes

(Symbol) Cause	Explanation
(A) Activity	The normal activity of Transport in moving data.
(C) Congestion	The resource limit condition, detected by Transport, that causes Transport to discard normal data.
(S) System Fault	Failures of either hardware or software and undetected line failures, excluding line faults.
(O) Operator Fault	Failure of an operator to set parameters correctly for the harmonious operation of multiple nodes. Typical operator faults include a node's maximum address that is too small or an adjacent node's identification that is too large.
(T) Topological Change	Modifications in topology that result in a node changing its reachability status.
(V) Verification Violation	The detected attempt of a node to initialize without providing the expected verification information.
(L) Line Fault	The detected failure of a physical link. A line fault does not necessarily result in a topological change.

NOTE

Some source events may have multiple causes:

Packet loss through aging is caused by topological fault and either:

Operator fault -- Rate control timer, T2, is set too large.

or

Software fault -- Decision or update operates too slowly.

Packet loss through aging is caused by a software fault when:

- Visit field is destroyed in some node or by undetected line error.
- Routing algorithm has not completed and packets are looping.

Packet loss through destination unreachability is caused by an operator fault when:

- Destination does not exist (detected at source node).
- Routing parameters, Maxh or Maxc, are set too small.
- Routing parameter NN (maximum node address) is less than node address.

Packet loss through destination unreachability is caused by a topological fault when:

- Node is really unreachable.
- Node will be unreachable when routing algorithm completes.
- Multiple topological faults cause an increase in actual network diameter and some maximum hop parameters are too small.

Table 6
Source Events

Event Manifestation	Source Event Number	Probable Cause	Definition
Data Movement	1	A	A data packet is received from a line for another node in the network.
	2	A	A data packet from another node in the network is sent over a line.
	3	A	A data packet is received from a line for this node's NSP.
	4	A	A data packet from this node's NSP is sent over a line.
Congestion	5	C	A transit data packet is discarded for congestion reasons.
	6	C	An output packet is discarded due to the inability of NSP to process output packets fast enough.
Data Packet Discarded	7	T	Destination node is unreachable.
	8 9	T S	Packet is too old. Destination node is out of range.
	10	O	Received data packet is too large to forward due to the blocksize of the data link that would be used.
Message Format Error	11	S	Message format is in error.
Partial Routing Update Loss node's NN.	12	O	Received long routing message with reachable node number greater than this node's NN.

(continued on next page)

Table 6 (Cont.)
Source Events

Event Manifestation	Source Event Number	Probable Cause	Definition
Line Down	13	L	Data link synchronization lost.
	14	L	Data link threshold error detected.
	15	L	Line rejection algorithm rejected line (for reason other than threshold error).
	15.1	L	Node listener timeout.
	15.2	L	Node listener received invalid data.
	16	S	Unexpected control (initialization or verification) message received.
	17	S	Routing message received with checksum error.
	18	O	Node id from routing or hello message not the expected one.
Initialization Failure	20	L	Verification message not received in timeout period.
	21	L	Data link synchronization lost.
	22	L	Data link threshold error detected.
	23	O	Version skew detected.
	24	O	Node id in received initialization message too large.
	25	O	Block size in received initialization message too large.
	26	O	Invalid verification seed value in received initialization message.
	27	S	Unexpected message received.
Verification Reject	28	V	Invalid verification received.
Line Up	29	L	Node initialization complete.
Node Reachability	30	T	Node reachable.
Change	31	T	Node unreachable.

Table 6
Source Events

Event Manifestation	Source Event Number	Probable Cause	Definition
Data Movement	1	A	A data packet is received from a line for another node in the network.
	2	A	A data packet from another node in the network is sent over a line.
	3	A	A data packet is received from a line for this node's NSP.
	4	A	A data packet from this node's NSP is sent over a line.
Congestion	5	C	A transit data packet is discarded for congestion reasons.
	6	C	An output packet is discarded due to the inability of NSP to process output packets fast enough.
Data Packet Discarded	7	T	Destination node is unreachable.
	8 9	T S	Packet is too old. Destination node is out of range.
	10	O	Received data packet is too large to forward due to the blocksize of the data link that would be used.
Message Format Error	11	S	Message format is in error.
Partial Routing Update Loss node's NN.	12	O	Received long routing message with reachable node number greater than this node's NN.

(continued on next page)

Table 6 (Cont.)
Source Events

Event Manifestation	Source Event Number	Probable Cause	Definition
Line Down	13	L	Data link synchronization lost.
	14	L	Data link threshold error detected.
	15	L	Line rejection algorithm rejected line (for reason other than threshold error).
	15.1	L	Node listener timeout.
	15.2	L	Node listener received invalid data.
	16	S	Unexpected control (initialization or verification) message received.
	17	S	Routing message received with checksum error.
	18	O	Node id from routing or hello message not the expected one.
Initialization Failure	20	L	Verification message not received in timeout period.
	21	L	Data link synchronization lost.
	22	L	Data link threshold error detected.
	23	O	Version skew detected.
	24	O	Node id in received initialization message too large.
	25	O	Block size in received initialization message too large.
	26	O	Invalid verification seed value in received initialization message.
	27	S	Unexpected message received.
Verification Reject	28	V	Invalid verification received.
Line Up	29	L	Node initialization complete.
Node Reachability	30	T	Node reachable.
Change	31	T	Node unreachable.

E.2 Counters

There are two types of counters -- node and line. Transport maintains one node counter for each of the defined node counters. Transport maintains one line counter per physical link for each of the defined line counters. Node counters count source events attributed to topological changes, faults, and verification violation. Line counters count source events attributed to activity, congestion, and faults.

Tables 7 and 8 define the node and line counters. Each counter relates to a source event number from Table 6, Section E.1. Refer to the glossary for definitions of input, output and transit packets. A packet is received when it passes from the Data Link layer to the Transport layer. A packet is sent when it passes from the Transport layer to the Data Link layer.

Table 7
Node Counters

Counter Name	Counter Width	Source Events Included
node unreachable packet loss	16 bits	7
aged packet loss	8 bits	8
node out-of-range packet loss	8 bits	9
oversized packet loss	8 bits	10
packet format error	8 bits	11
partial routing update loss	8 bits	12
verification reject	8 bits	28

Table 8
Line Counters

Counter Name	Counter Width	Source Events Included
transit packet received	32 bits	1
transit packet sent	32 bits	2
output (arriving) packet received	32 bits	2
input (departing) packet sent	32 bits	3
transit congestion loss*	16 bits	5
output (arriving) congestion loss	16 bits	6
line down	16 bits	13 - 18
initialization failure	16 bits	19 - 27

* Only required in the implementations in which NSP does not guarantee Transport that it will process an output packet (thereby freeing the buffer holding the packet) in a bounded period of time.

E.3 Events

Network Management groups some of the source events (Section E.1) together for logging. The DNA Network Management Functional Specification specifies this logging operation. When a source event to be logged occurs, Transport identifies it by type, time-stamps it, and places it in an internal Transport event queue. If the event queue is full, Transport discards the newest event in the queue and replaces it with an "event(s) lost" event.

Table 9 specifies the Transport events and the source events that trigger them. The source event numbers are from Table 6, Section E.1.

Table 9
Transport Events

Event Type	Source Events	Logged Information
Node unreachable packet loss	7	line, packet header
Aged packet loss	8	packet header
Node out-of-range packet loss	9	line, packet header
Oversized packet loss	10	line, packet header
Message format error	11	line, packet header
Partial routing update loss	12	line, packet header, highest node address
Line down - line fault	13-15.2	line
Line down - software fault	16,17	line, packet header
Line down - operator fault	18	line, packet header, expected node id
Initialization failure - line fault	19-22	line
Initialization failure - operator fault	23-26	line, packet header, received version (23 only)
Initialization failure - software fault	27	line, packet header
Verification reject	28	line, node id from message
Line up	29	line, node id from message
Node reachability change	30,31	node address

NOTE

1. A logged event of a single type that can result from more than one source event also contains a reason code to specify the source event.
2. "Packet header" denotes the first 6 bytes (48 bits) of a Transport message.

APPENDIX F

ALGORITHMS AND MODELS

This appendix describes algorithms and models pertaining to:

- Line cost (Section F.1)
- Square root limiter (Section F.2)
- Buffer management (Section F.3)

F.1 Line Cost Assignment Algorithm

The assignment of cost to lines can reflect both delay and throughput data. Delay data can include transmission delay, propagation delay, processing delay, and retransmission delay. Delay data does not include queueing delay. Throughput data can include line bandwidth, circuit overhead, and processor bandwidth. Throughput data does not include actual traffic overhead. Basically, it is desirable to avoid a line cost assignment algorithm with high sensitivity to traffic fluctuations, thereby producing a condition where routes change to accommodate traffic changes and the new traffic flow causes new route changes, and so on.

A line cost assignment occurs as a result of a node generation or an initialization module. An operator can always override any assignment.

One such assignment is based on line bandwidth and is as follows:

$$F(x) = \begin{cases} 1 \text{ where bandwidth } x >= 100{,}000 \text{ bits/second} \\ [100{,}000/x] \text{ for } 4{,}000 \text{ bits/second} < x < 100{,}000 \text{ bits/sec} \\ 25 \text{ where } x <= 4{,}000 \text{ bits/second} \end{cases}$$

where x is line bandwidth (bits/sec)

F.2 Square Root Limit Algorithm

Calculate the square root threshold as follows:

Let n = number of Transport buffers in the pool
y = number of output lines that are active

Then x = the square root limit, which is the smallest integer greater than or equal to n / SQRT y.

F.3 Buffer Management

When no buffers are available for receiving packets from a line, store and forward deadlock can occur. Deadlock can be avoided by insuring that at least one buffer is available per line, or a buffer can be made available without requiring additional resources. Such deadlock avoidance can require discarding packets.

When receive buffers are not available quickly enough, a line can go down unnecessarily at the Data Link layer. It is much better for Transport to discard a packet than for a line to go down.

Transport should not initialize unless it can obtain at least the minimum number of receive buffers for each line. If an implementation obtains these buffers from a shared system buffer pool, then the minimum number must be permanently allocated from the pool by Transport when it initializes. They can, of course, be returned when Transport halts.

The only time a line may be allowed to go below its minimum number of buffers is when the system can guarantee that a receive buffer can be allocated to the line soon enough in the future to prevent the line from going down. This means that if the system has run out of free buffers and is down to the minimum number of receive buffers for each line, then:

1. A received data packet that would normally be forwarded on another line must be discarded.

2. A received Transport control message can and should be processed.

3. A received data packet for this node should be given to NSP only if NSP is known to be able to return the buffer in a short, bounded period of time. Otherwise, discard the packet.

Compute the minimum number of receive buffers required for a given line from the line speed and an estimate of the maximum time that Transport (or possibly NSP) can take to process a received message.

F.3.1 **Possible Buffer Management Model** - Transport has a common pool of buffers that can transmit or receive. If the implementation of NSP in a system is known to be unable to make the guarantee of short, bounded processing of received data packets, then Transport must limit the number of outstanding , received packets that NSP can hold onto simultaneously (provided that NSP and Transport are sharing a common buffer pool). This is best done by a fixed quota.

Setting this quota to the square root limit used by the congestion control algorithm is acceptable, but other values may be used as well. If a quota is used, then any packets discarded due to the NSP quota being filled must be counted as described in Appendix E.

The Transport buffer quota provided by the system is divided into the following buffer quotas:

1. Decision (0)

2. Update (1 sufficient; 1 per line recommended)

3. Node Listener (0)

4. Node talker (1)

5. Forwarding (at least 1 per line; 12 - 15 per terrestial line and 30 - 35 per satellite link recommended)

6. A separate receive quota for each line (depends on speed of line -- at least 1, 2 or 3 recommended)

   If an implementation is constructed using a single buffer pool that Transport shares with other system processes, and if Transport does not do any actual data moving from one buffer to another, then all buffers containing data to be transmitted are either obtained from NSP or are receive buffers that contain data packets that are being forwarded.

   The rules above and the congestion control algorithms in Section 5 adequately define the use of these buffers. However, note the following:

   First, the square root limit is defined to be the number of buffers available for forwarding divided by the square root of the number of lines. The number of buffers available for forwarding should not include the minimum number of receive buffers, nor should it include NSP's quota, if such a quota exists.

   Second, in such a model, a single buffer beyond the minimum number of receive buffers and a single NSP transmit buffer are sufficient to allow Transport to run correctly without starving a line for receive buffers. In general, for adequate performance additional buffers will also be required.

F.3.2 **Details of Charging and Crediting Against Quotas** - All buffers not free will be charged against a specific quota. The quota will never be exceeded except possibly for a brief instant while a Transport process frees the buffer by consuming the information or by discarding a packet.

A quota is charged for a buffer upon the following events:

1. A free buffer is assigned to the Data Link layer for reception on a specific line.

2. A buffer is moved from one Transport module to another. The receiving quota is charged.

3. A buffer is supplied by NSP that contains input data.

4. A free buffer is seized by a process to send a Transport control message.

A quota is credited for a buffer upon the following events:

1. Transmission of a buffer is completed by the Data Link layer. (The quota is credited whether or not the transmission was successful.)

2. A buffer is moved from one Transport module to another. The sending quota is credited.

3. A process consumes the contents of a Transport control message and returns the empty buffer.

4. A process discards a packet and returns the buffer.

5. NSP issues a successful CHECK RECEIVE command.

APPENDIX G

PHASE II COMPATIBILITY

This Appendix describes how Phase II nodes can communicate with Phase III nodes. Phase II node compatibility has the following goal:

> Any solution must provide a smooth transition such that customers can upgrade one node at a time while never decreasing communication capabilities.

The solution is as follows:

> A Phase II node can communicate only with adjacent nodes. A Phase II node derives no benefits from a routing network until it upgrades to either a routing or nonrouting node subject to the topological restrictions described in Appendix B.

The solution assumes the following Phase II characteristics:

1. Transport Initialization or Verification messages are ignored and dropped. The line is not recycled.

2. Phase II nodes transmit with either no route header or an ASCII route header. An illegal message (routing message or packet with Transport route header) is dropped.

3. Phase II nodes understand that communication is point-to-point.

4. Phase II nodes can process NSP acknowledgment messages in response to NSP Connect Initiate and Connect Confirm messages.

5. Phase II nodes can destroy logical links when a physical line fails.

Communication between Phase II and Phase III nodes is as follows:

> **Initialization.** Upon recognizing a Phase II Node Initialization message, a Phase III node builds and sends the Phase II format Node Initialization message and, if requested, a verification message. The Phase II node is flagged for this line. The routing process receives a line up event and updates the routing and forwarding data bases.

Receiving messages. A packet from a Phase II node is distinguishable from a packet with a Transport route header since a Phase II node transmits packets with either no route header or an ASCII route header. When an ASCII route header is sent, the new node removes the route header by skipping the route flags and two image fields. The destination address is the local node address. The source address is the adjacent node address (established during initialization).

Transmitting messages. A packet from a Phase III node to a Phase II node uses the line selected by the forwarding process. This line can be recognized as having a Phase II node at the other end because this information was obtained during the initialization process. A packet to a Phase II node does not receive a Transport route header.

GLOSSARY

aged packet

A packet that has exceeded the maximum number of visits.

congestion

The condition that arises when there are too many packets to be queued.

datagram

A unit of data passed between Transport and the Network Services layer. When a route header is added, it becomes a packet.

end node

A topological description of a nonrouting node. Since a nonrouting node cannot perform route-through and supports only a single line, it must be an end node. However, it is also possible for a routing node with a single line to be an end node.

error

The manifestation of a fault.

event

Occurrences that are logged for recording by Network Management. Events result from occurrences of source events.

fault

The cause of a problem in the operation of the Transport modules.

hello

The term for the packet lifetime activity used for dead node recognition. The node talker sends hello messages.

hop

The logical distance between two adjacent nodes in a network.

initialization

With regard to Transport, a start-up procedure between two adjacent nodes.

input (departing) packet

A packet from this node's Network Services layer.

line cost

The positive integer value associated with using a line.

logical connectivity

The result of nodes being able to communicate with each other. Used in describing network topologies.

maximum address

The largest possible node address in a DECnet network. This is equal to or less than the number of nodes in the network.

maximum cost

An operator-controllable Transport parameter that defines the point where the routing decision algorithm in a node declares another node unreachable because the cost of the least costly path to the other node is excessive. For correct operation, this parameter must not be less than the maximum path cost of the network.

maximum hops

An operator-controllable Transport parameter that defines the point where the routing decision algorithm in a node declares another node unreachable because the shortest path between the two nodes is too long. For correct operation this parameter must not be less than the network diameter.

maximum path cost

The routing cost between the two nodes of a network having the greatest routing cost, where routing cost is the cost of the least costly path between a given pair of nodes. In Figure 3 the maximum path cost is 9.

maximum path length

The routing distance between the two nodes of a network having the greatest routing distance, where routing distance is the length of the least costly path between a given pair of nodes. In Figure 3 the maximum path length is 4.

maximum visits

An operator-controllable Transport parameter that defines the point where the packet lifetime control algorithm discards a packet which has traversed too many nodes. For correct operation, this parameter must not be less than the maximum path length of the network.

network diameter

The reachability distance between the two nodes of a network having the greatest reachability distance, where reachability distance is the length of the shortest path between a given pair of nodes. In Figure 3 the network diameter is 3.

node

An implementation of Session Control. The interface between the network and a DIGITAL computer system.

nonrouting node

A Phase III DECnet node that contains a subset of routing modules (select process and receive process) and can deliver and receive packets. It is connected to the network by a single line.

output (arriving) packet

A packet whose destination is this node.

packet

A unit of data to be routed from a source node to a destination node. When stripped of its route header and passed to the Network Services layer, it becomes a datagram.

packet looping

Condition where a packet revisits a node.

path

The route a packet takes from source node to destination node. This can be a sequence of connected nodes between two nodes.

path cost

The sum of the line costs along a path between two nodes.

path length

The number of hops along a path between two nodes.

physical connectivity

The result of nodes being attached to each other by active lines used in describing network topologies.

reachable node

A node to which a routing node believes it can direct a packet.

received packet

A packet received by this node's Transport layer from the Data Link layer.

route-through

Packet switching. The directing of packets from source nodes to destination nodes. Routing nodes permit route-through.

routing

Directing data message packets from source nodes to destination nodes.

routing node

A Phase III DECnet node that contains the complete set of Transport modules, and can deliver, receive and route through packets.

sent packet

A packet passed from this node's Transport layer to the Data Link layer.

source event

A specified occurrence in this node's Transport layer that may cause a counter to be incremented or an event to be logged.

topology

The physical arrangement and relationships of interconnected nodes and lines in a network. A legal topology satisfies the requirements of this specification.

transit packet

A packet arriving at this node from a source node and destined for another node.

unreachable node

A node to which a routing node has determined that the path exceed the maximum hops of the network.

visit field

The field in the packet route header that contains the count of the number of nodes visited by this packet.

DECnet DIGITAL
Network Architecture
(Phase III)
AA-K180A-TK

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of reader that you most nearly represent.

- □ Assembly language programmer
- □ Higher-level language programmer
- □ Occasional programmer (experienced)
- □ User with little programming experience
- □ Student programmer
- □ Other (please specify)________________

Name____________________ Date__________

Organization________________________

Street____________________________

City____________ State________ Zip Code________
or
Country

Please cut along this line.

—Do Not Tear - Fold Here and Tape —

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE DOCUMENTATION
146 MAIN STREET ML 5-5/E39
MAYNARD, MASSACHUSETTS 01754

Do Not Tear - Fold Here and Tape

Cut Along Dotted Line