Guide to DECnet–VAX Networking

Order Number: AA–LA47A–TE

April 1988

This book presents a summary of information a VMS user or manager needs to function in a networking environment. It introduces basic DECnet–VAX networking concepts, summarizes user and manager network operations, and describes the procedures for getting on an existing network and keeping the network running.

Revision/Update Information:	This manual is a new manual; it does not supersede any previous manual.
Software Version:	VMS Version 5.0

digital equipment corporation
maynard, massachusetts

April 1988

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

The following are trademarks of Digital Equipment Corporation:

DEC	DIBOL	UNIBUS
DEC/CMS	EduSystem	VAX
DEC/MMS	IAS	VAXcluster
DECnet	MASSBUS	VMS
DECsystem–10	PDP	VT
DECSYSTEM–20	PDT	
DECUS	RSTS	
DECwriter	RSX	digital™

ZK4627

HOW TO ORDER ADDITIONAL DOCUMENTATION
DIRECT MAIL ORDERS

USA & PUERTO RICO*

Digital Equipment Corporation
P.O. Box CS2008
Nashua, New Hampshire
03061

CANADA

Digital Equipment
of Canada Ltd.
100 Herzberg Road
Kanata, Ontario K2K 2A6
Attn: Direct Order Desk

INTERNATIONAL

Digital Equipment Corporation
PSG Business Manager
c/o Digital's local subsidiary
or approved distributor

In Continental USA and Puerto Rico call 800-258-1710.
In New Hampshire, Alaska, and Hawaii call 603-884-6660.
In Canada call 800-267-6215.
*Any prepaid order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575).
Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminster, Massachusetts 01473.

Production Note

This book was produced with the VAX DOCUMENT electronic publishing system, a software tool developed and sold by DIGITAL. In this system, writers use an ASCII text editor to create source files containing text and English-like code; this code labels the structural elements of the document, such as chapters, paragraphs, and tables. The VAX DOCUMENT software, which runs on the VMS operating system, interprets the code to format the text, generate a table of contents and index, and paginate the entire document. Writers can print the document on the terminal or line printer, or they can use DIGITAL-supported devices, such as the LN03 laser printer and PostScript™ printers (PrintServer 40 or LN03R ScriptPrinter), to produce a typeset-quality copy containing integrated graphics.

™ PostScript is a trademark of Adobe Systems, Inc.

Contents

FIGURES

TABLES

Preface

The *Guide to DECnet–VAX Networking* provides an introduction to networking on a VMS system. The book presents basic networking concepts, describes briefly how the DECnet network operates, explains how the VMS system uses DECnet–VAX software to access the DECnet network, and examines the different network environments.

This Guide summarizes and uses examples to illustrate everyday network operations, as well as network application development. It explains how to access a system on an existing DECnet network. In addition, it specifies a complete procedure for installing DECnet–VAX on a VMS system, including the steps for establishing an asynchronous DECnet connection. The manual also provides network maintenance and troubleshooting suggestions.

Who Should Read This Guide?

This Guide is designed to be read by anyone who wants to know about networking on VMS operating systems. The Guide does not assume that a reader is familiar with networking. Therefore, it may be particularly useful to anyone new to networking (for example, a new VMS user) or new to DECnet–VAX (for example, an experienced VMS user or manager who is not accustomed to working in a networking environment). Additionally, experienced users of DECnet–VAX should find the Guide of interest because it provides an overview of networking capabilities.

This Guide addresses three categories of readers: general users of the network, advanced network users and programmers, and managers of individual systems or whole networks. A reader may play different roles at different times.

Most readers will make general use of a network for everyday operations (for example, to access files on other systems and send electronic mail). General users may require little knowledge of the way data is transferred from one system to another and may not be concerned with setting up the network.

Some general users may also be advanced users, doing system programming in a network environment or developing network application programs. Advanced programmers may be interested in the way the network functions and how they can use specific networking features.

On some VMS systems (for example, VAXstations) general users may also be the managers of their own systems, and may want to connect them to an existing network. In a large network, one of the system managers may also serve as the manager of the entire network.

Whether you are carrying out daily operations on a VMS system, writing VMS application programs, or managing a VMS system, you can use this Guide to become familiar with the ways the network can enhance your working environment.

What Information Does the Guide Provide?

The initial chapters of the book try to answer the first questions a new user might ask: What is networking? What is DECnet–VAX? What does a DECnet network look like? What can I do over the network? What does the manager of a system have to do? Later chapters of the book attempt to answer some basic startup questions: How can I log in to a network? How can I connect my system to a network? What do I do if I run into network problems?

In response to these questions, the Guide provides the following information:

- Chapter 1: A basic introduction to the concept of networking and a summary explanation of how the DECnet network works. How DECnet–VAX relates to the VMS operating system. What the DECnet network looks like (a series of diagrams showing examples of various DECnet configurations).
- Chapter 2: A summary of the functions VMS users can perform over the network. Descriptions of the basic operations a general user can carry out, with examples of file-handling commands. A summary description of the operations an advanced user or system programmer can perform, illustrated by programming examples. A summary of the responsibilities of the manager of a single system or a whole network.
- Chapter 3: A set of basic instructions for getting on an existing network. How a user logs in to an existing network node. How a system manager brings a VMS system up as a node on an existing network, including a complete procedure for installing DECnet–VAX on the system. A summary of considerations involved in establishing a network.
- Chapter 4: Information useful in keeping the network running, including monitoring tools, test procedures, most frequently encountered message displays, and elementary troubleshooting procedures.

Road Map to the Contents of This Manual

Reading this whole manual will provide an introduction to all aspects of networking on a VMS system. However, many readers assume specific roles on a VMS system and may not be interested in certain topics in the Guide. Therefore, the following suggestions are offered to help different categories of readers select the sections of the manual that they might want to read:

- **General User:** Figure 1 lists sections of the Guide that may be most relevant to general user tasks such as file operations and sending mail over the network.
- **Advanced User:** Figure 2 lists sections of interest to users who perform more advanced network operations and programmers who want to develop applications that communicate between different systems.
- **System/Network Manager:** Figure 3 lists sections that provide general networking information for managers and specific procedures for establishing and managing a system on the network.

Figure 1 Sections of the Guide of Interest to General Users of the Network

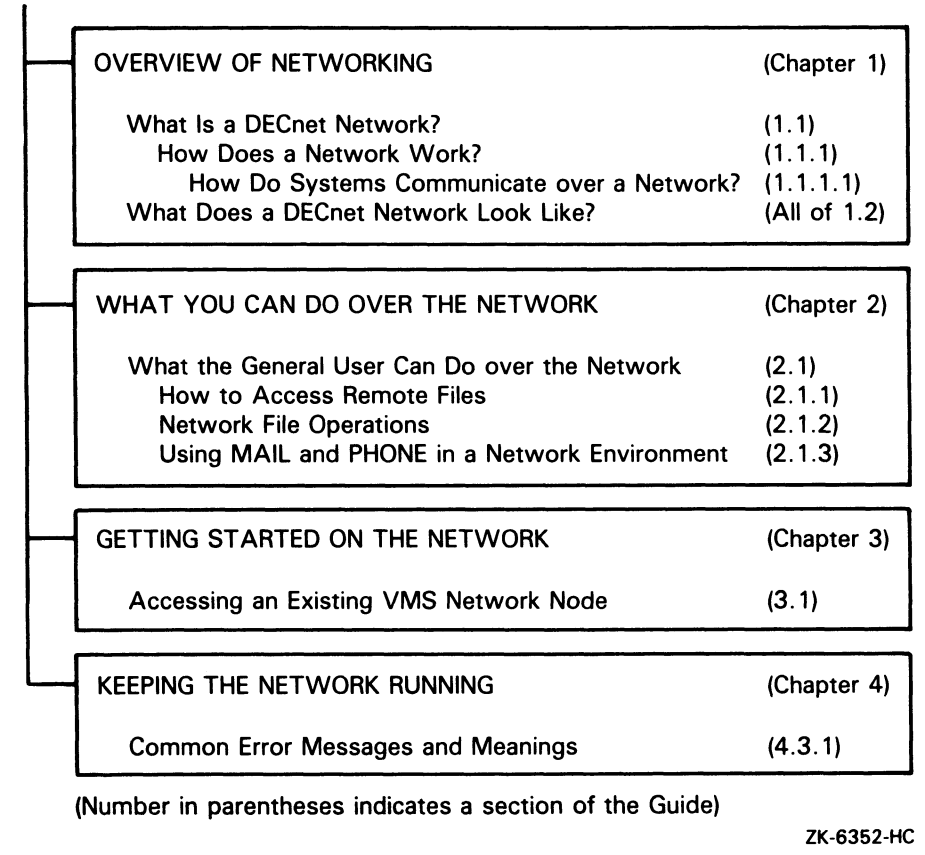


Figure 2 Sections of the Guide of Interest to Advanced Users of the Network

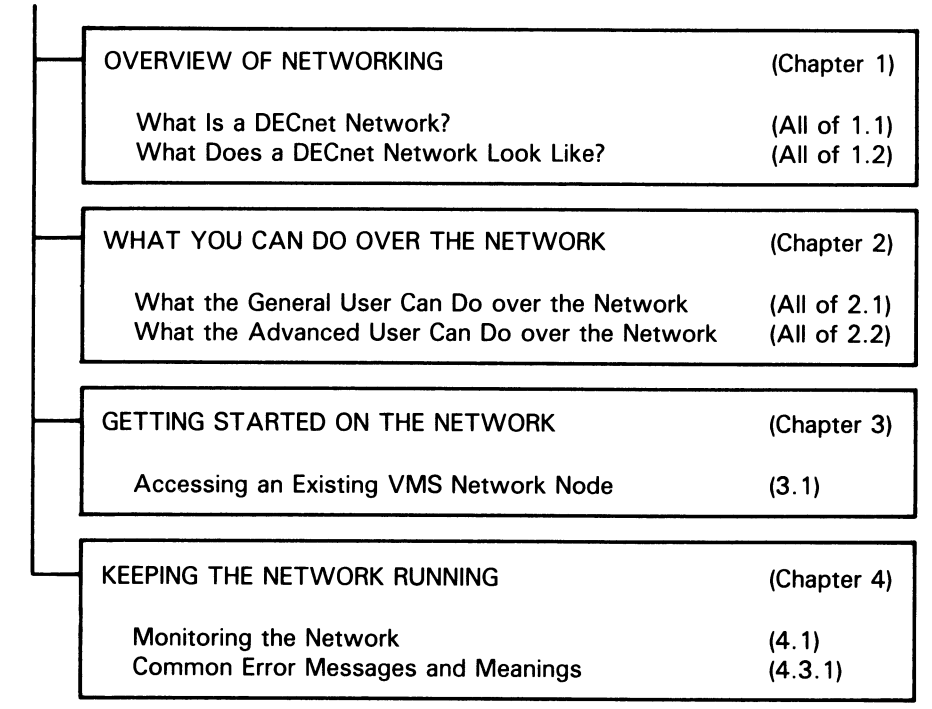
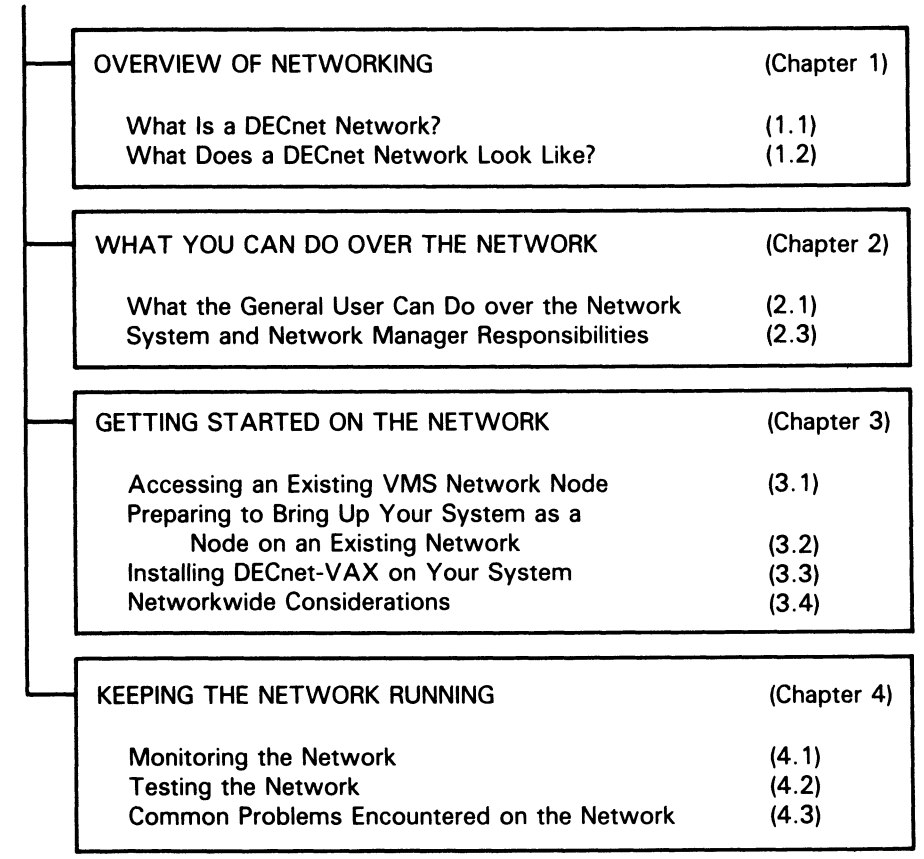


Figure 3 Sections of the Guide of Interest to System and Network Managers



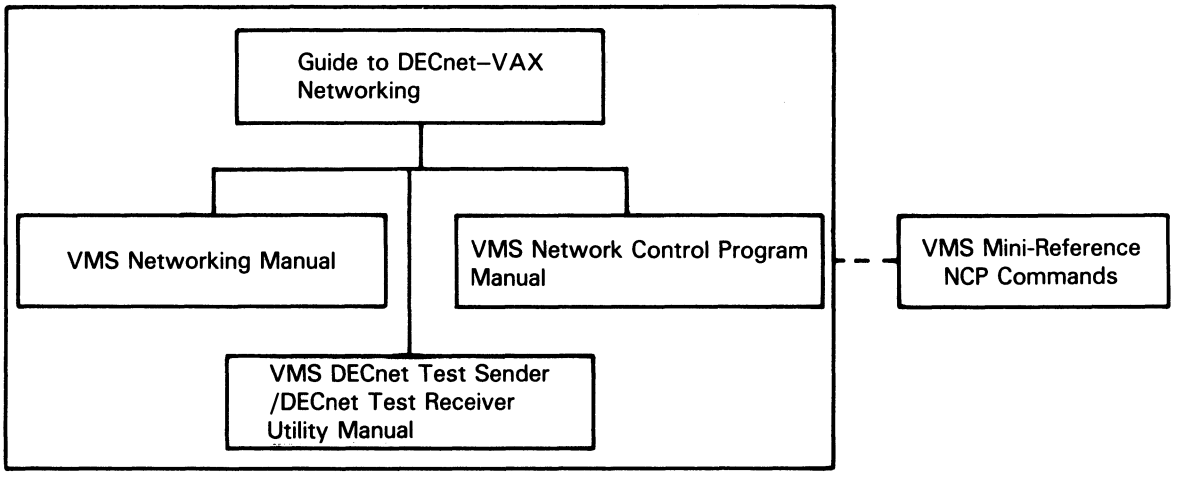
(Number in parentheses indicates a section of the Guide)

ZK-6354-HC

Where to Find More Information on Networking

For detailed information on using DECnet–VAX, refer to the manuals in the Networking Volume of the VMS System Management Kit in the VMS documentation set.

NETWORKING VOLUME:



ZK-6365-HC

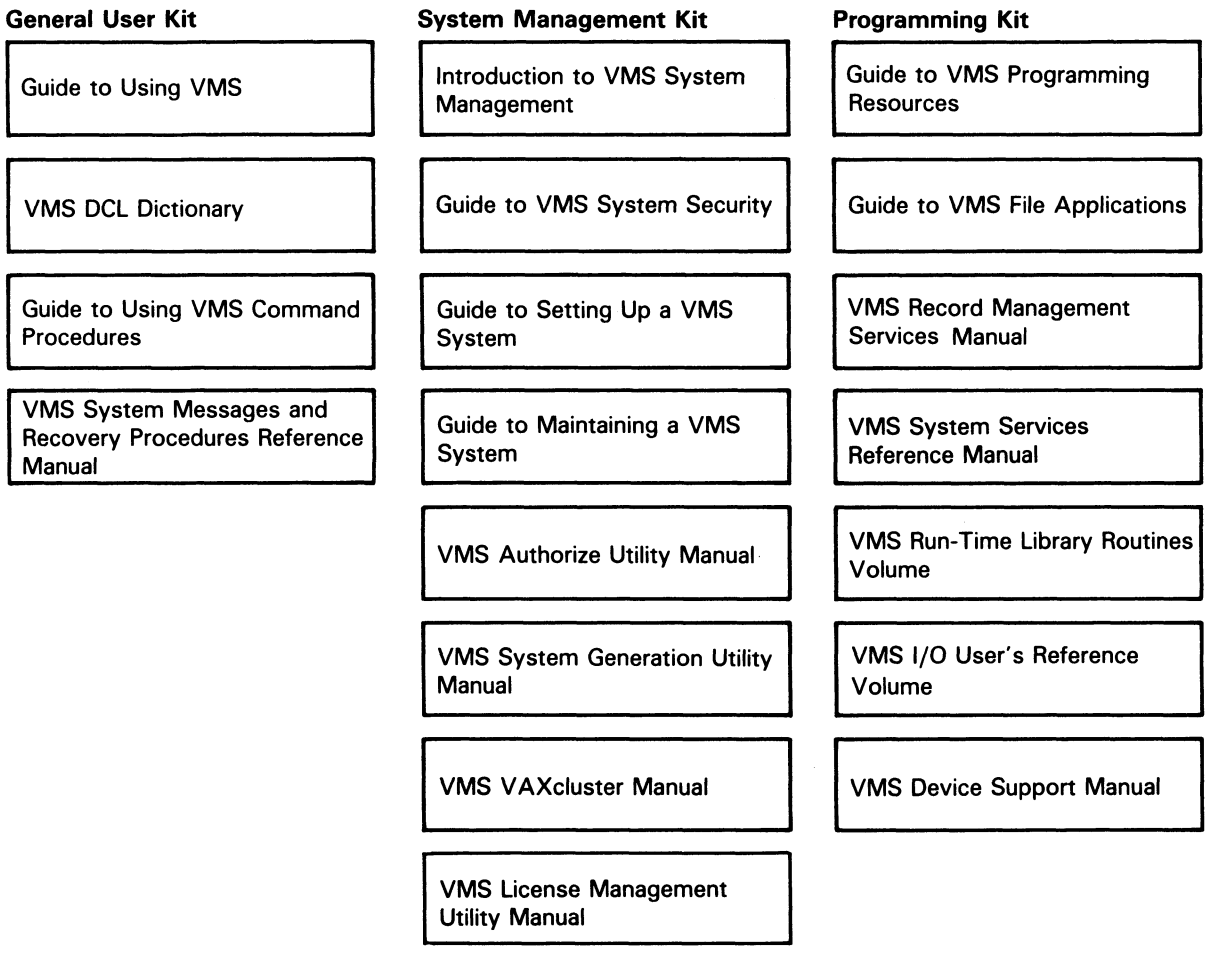
The manuals in the Networking Volume provide the following information:

- The *Guide to DECnet–VAX Networking* provides an introduction to networking on a VMS system and summarizes basic networking operations and procedures.
- The *VMS Networking Manual* includes conceptual and usage information for VMS system and network managers and for DECnet–VAX users and programmers.
- The *VMS Network Control Program Manual* provides usage information for the Network Control Program (NCP) Utility.
- The *VMS DECnet Test Sender/DECnet Test Receiver Utility Manual* provides usage information for the DECnet Test Sender/Receiver (DTS/DTR) Utility.

Additionally, the following manuals available to all VMS users include information directly related to the Networking Volume:

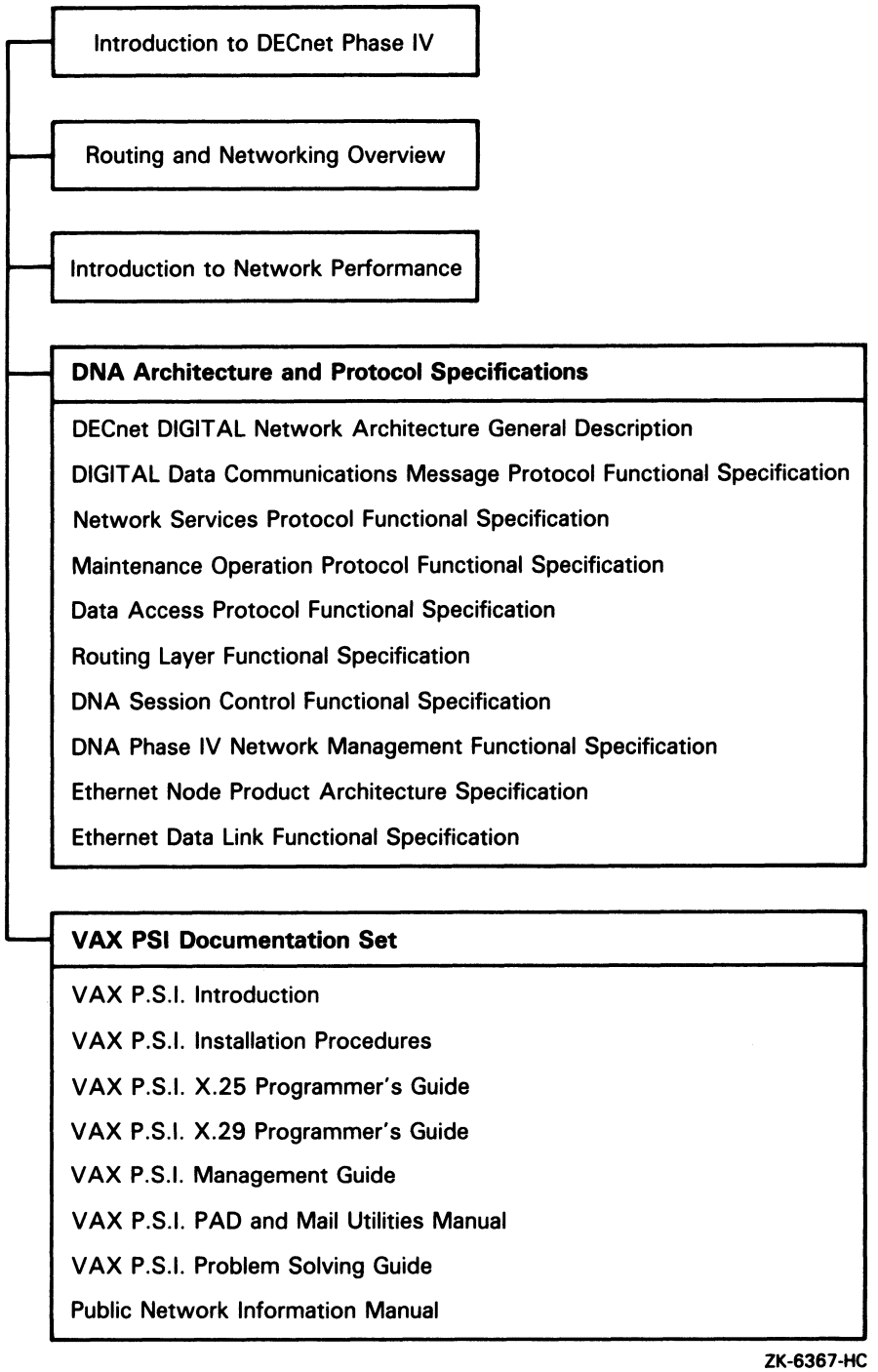
- The *VMS Mini-Reference* includes a quick-reference summary of NCP command formats.
- The *VMS Version 5.0 New Features Manual* describes new networking features supported for the VMS Version 5.0 release.
- The *VMS Version 5.0 Release Notes* includes DECnet–VAX release notes.

In addition to the manuals in the Networking Volume, the following manuals and volumes in the VMS documentation set also include networking information.



ZK-6366-HC

Other DIGITAL documentation that is not part of the VMS documentation set contains additional information on the DECnet network. This documentation (shown in the following diagram) includes DECnet networking overview manuals, DNA specifications, and the VAX PSI documentation. The DNA architecture and protocol specifications define DIGITAL Network Architecture (DNA) protocols to which all implementations of DECnet adhere. The VAX PSI manuals describe the VAX Packetnet System Interface (PSI) software that permits VMS users to communicate over a packet switching network. For more information on these documents, contact your local DIGITAL representative.


Introduction to DECnet Phase IV
Routing and Networking Overview
Introduction to Network Performance
DNA Architecture and Protocol Specifications
DECnet DIGITAL Network Architecture General Description
DIGITAL Data Communications Message Protocol Functional Specification
Network Services Protocol Functional Specification
Maintenance Operation Protocol Functional Specification
Data Access Protocol Functional Specification
Routing Layer Functional Specification
DNA Session Control Functional Specification
DNA Phase IV Network Management Functional Specification
Ethernet Node Product Architecture Specification
Ethernet Data Link Functional Specification
VAX PSI Documentation Set
VAX P.S.I. Introduction
VAX P.S.I. Installation Procedures
VAX P.S.I. X.25 Programmer's Guide
VAX P.S.I. X.29 Programmer's Guide
VAX P.S.I. Management Guide
VAX P.S.I. PAD and Mail Utilities Manual
VAX P.S.I. Problem Solving Guide
Public Network Information Manual
ZK-6367-HC

Conventions Used in This Manual

Convention	Meaning
SET LINE line-id COST cost	Example command words are shown in a command line in capital letters, and they must be entered as shown. Arguments are shown in command lines as lowercase letters. (In this case, you replace the argument shown in the command format with the precise information requested.)
SET KNOWN LINES ALL SE KN LINE ALL	You can abbreviate any command verb to its fewest unique letters. (In this manual, however, commands, formats, and examples use the complete command words.)
NCP> EXIT	Command examples show all output lines or prompting characters that the system prints or displays in black letters. This document uses red lettering to indicate all user-entered commands.
RET	In examples, a key name (usually abbreviated) shown within a box indicates that you press a key on the keyboard; in text, a key name is not enclosed in a box. In this example, the key is the RETURN key. (Note that the RETURN key is not usually shown in syntax statements or in all examples; however, assume that you must press the RETURN key after entering a command or responding to a prompt.)
CTRL/C	A key combination, shown in uppercase with a slash separating two key names, indicates that you hold down the first key while you press the second key. For example, the key combination CTRL/C indicates that you hold down the key labeled CTRL while you press the key labeled C. In examples, a key combination is enclosed in a box.

1 Overview of DECnet–VAX Networking

A VMS operating system can be linked to other systems in a network, greatly expanding the power and capability available on the system. Within a DECnet network, all DIGITAL systems can communicate with each other, and, through special communications products, with selected systems of other vendors. A VMS system participates in a DECnet network through its networking interface, DECnet–VAX.

This chapter provides an introduction to the DECnet network and DECnet–VAX. It also presents an overview of basic networking concepts and explains briefly how a DECnet network operates. Descriptions of typical networking environments indicate the variety of ways in which networks can be set up.

1.1 What Is a DECnet Network?

DECnet is the collective name for the family of communications products (software and hardware) that allow DIGITAL operating systems to participate in a network. A VMS operating system can use its networking software interface, DECnet–VAX, to become part of a DECnet network. As a part of a network, a VMS system can communicate with other VMS systems running on the full range of VAX processors, as well as with a wide range of non-VMS systems that use DECnet software.

All systems connected to a DECnet network are peers or equals. Systems can communicate with each other without having to go through a central or master system. Any system in the network can communicate with any other system in the network, not merely with those systems to which it is directly attached. Network users can gain access to software facilities that do not exist on their particular system, and can communicate freely over the whole network.

A DECnet network links computers into flexible configurations to exchange information, share resources, and perform *distributed processing*. DECnet distributed processing capabilities allow information to be originated anywhere in the network. VMS systems can be placed at locations where they are required while still having access to the facilities of other widely dispersed systems. Access to the network is available wherever it is needed: executive offices, factory floors, laboratories, field locations. Information can be exchanged between all parts of an organization or institution efficiently in a stable, integrated networking environment. An entire organization can be connected into a single unit by means of the network.

1.1.1 How Does a DECnet Network Work?

A DECnet network consists of two or more computing systems linked for the purpose of exchanging information and sharing resources. Network activity involves the flow of information between the systems. Data originated on one system is routed through the network until it reaches its destination on another system.

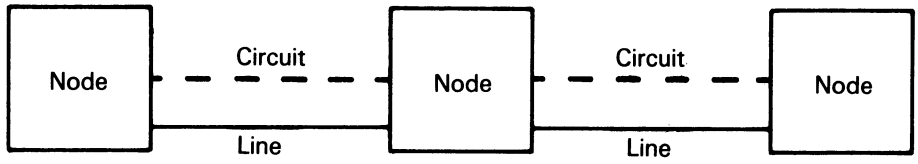
Overview of DECnet–VAX Networking

1.1 What Is a DECnet Network?

1.1.1.1 How Do Systems Communicate over a Network?

Each system on the network is called a *node*. Every node has a unique name and address. Nodes in the network are connected by *lines* over which *circuits* operate (see Figure 1–1). A line is a physical path over which data passes from one node to another in the network. (The path may be over a cable or telephone line, or possibly a microwave or satellite link.) A circuit can be thought of as a higher-level logical connection that operates over the physical connection. The circuit is the communications data path that carries information from one node to another. All input and output (I/O) activity between nodes occurs over circuits. Multiple users can use each circuit. A node can be designed to have active circuits operating over a number of lines that connect the node to the other nodes in the network.

Figure 1–1 Network Nodes, Circuits and Lines



A computing system can run many different processes and programs. For two processes to communicate with each other, they have to have a way to establish contact and exchange data. DECnet permits computer processes running on the same or different nodes to communicate with each other over *logical links*. A logical link connects two processes and carries a stream of two-way communications traffic between the processes over one or more circuits.

The process or program to which a logical link is connected is called an *object*. On a VMS node, some objects are DECnet–VAX system programs (for example, the MAIL object); other objects can be user-written programs. For two programs to communicate over the network, the program on one node establishes a logical link with the object on the other node.

1.1.1.2 How Does the Network Route Messages?

In a DECnet network, the process of directing a data message from a source node to a destination node is called *routing*. The route the data travels over the circuits in the network is called the *path*.

Messages can be exchanged between any two nodes in the DECnet network, even if they are not directly connected to each other. In order for nodes that are not directly connected to be able to communicate, an intervening node along the data path must forward the data received from the source to the destination. Intervening nodes that receive data and forward it to another node are known as routing nodes (or *routers*). Nodes that cannot forward data are called *end nodes*. Both routers and end nodes can send messages to and receive messages from other nodes, but only the router can forward messages on behalf of another node. A router can have more than one active circuit connecting it to the network; an end node can only have one.

A router maintains an information database about the availability of paths to the destination node and keeps it up-to-date by regularly exchanging routing information with other routers. The routing information includes the *cost* and the number of *hops* involved in sending data down a path to a destination node. The circuit cost is a number that the system manager assigns to a circuit between two nodes; the path cost is the sum of the circuit costs along the path to a given node. A hop is the distance between two directly connected nodes; the path length is the number of hops along the path between two nodes.

The router uses current information from its database to choose a data path through the network. The router determines the path to the destination based on the least cost. By changing the cost of a circuit, the manager of a network node can affect the flow of data through the network.

DECnet performs "adaptive routing"– that is, routing that adapts to changing conditions in the network. DECnet selects the best path currently available from the source to the destination. If network conditions change and the primary path becomes unavailable, DECnet redirects the data over the next best alternative path. DECnet automatically reroutes messages if a circuit becomes disabled or a lower-cost path becomes available.

Because adaptive routing in a DECnet network permits messages to be routed over the most cost-effective path currently available, a general user of the network need not be concerned with the path to the destination. The user need only specify the name of the remote node with which he or she wishes to communicate.

1.1.1.3 How Large Can the Network Be?

A DECnet network can vary in size from a small to a very large network. A typical small network might consist of two to four nodes. A maximum of 1023 nodes is possible in an undivided DECnet network; an optimum number is approximately 300 to 500 nodes, depending on the network topology (the way the nodes and lines are arranged in the network).

Very large DECnet networks can be divided into multiple *areas*: up to 63 areas, each containing a maximum of 1023 nodes. In a multiple-area network, the network manager groups nodes into separate areas, with each area functioning as a subnetwork. Nodes in any area can communicate with nodes in other areas. DECnet supports routing within each area and a second, higher level of routing that links the areas, resulting in less routing traffic throughout the network. Nodes that perform routing within a single area are referred to as *level 1 routers*; nodes that perform routing between areas as well as within their own area are called *level 2 routers* (or *area routers*).

1.1.1.4 How Is the DECnet Software Design Structured?

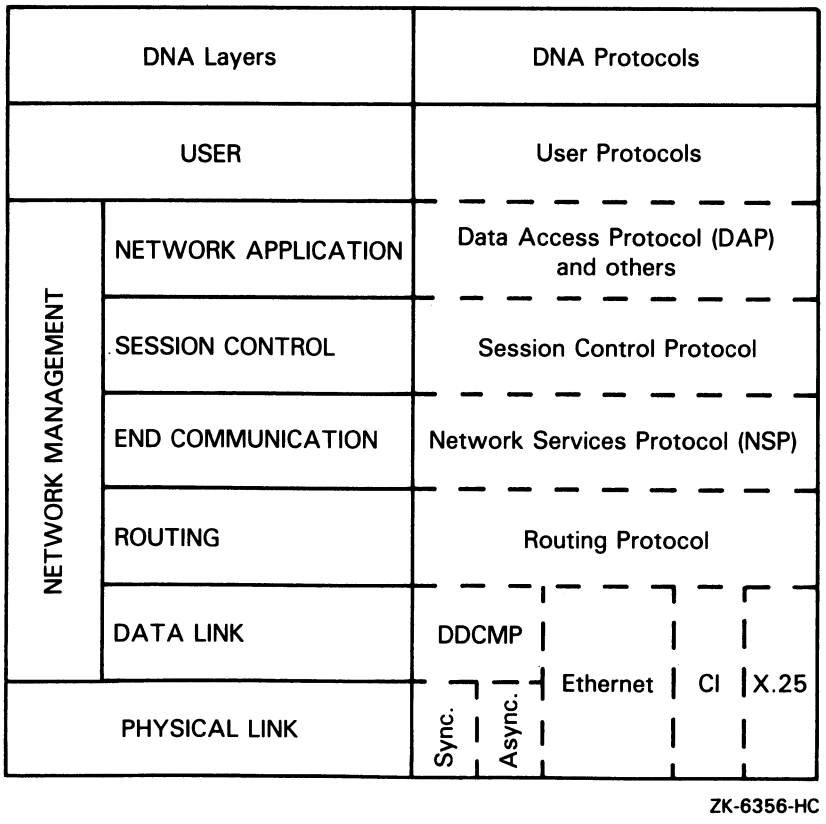
DECnet software design is based on the DIGITAL Network Architecture (DNA), which follows industry standards. The structured design permits a DECnet network to be extended easily and to incorporate new developments in data communications. DECnet nodes can communicate with any system that supports the same DECnet protocols.

DNA specifications govern the interrelationship of the components that make up the DECnet software. The specific functional boundaries between DECnet software components residing at each node are structured as a hierarchical set of layers. Each DNA layer is a client of the next lower layer and does not function independently.

DNA specifies the functional layers in which DECnet software is arranged on each node, and the communications *protocols* through which the corresponding layers at different nodes communicate with each other. Each protocol is a set of messages with specific formats and the rules for exchanging the messages. Protocols govern the operation of a communications link.

Figure 1–2 illustrates DNA layers and the related DNA protocols that provide DECnet network functions at each layer. DECnet functions are described in this manual. The types of lines that can be configured using DECnet data link protocols are discussed briefly later in this chapter. For a complete description of DNA, refer to the DNA specifications.

Figure 1–2 DECnet Network Architecture (DNA) Layers and Protocols



1.1.2 How Does DECnet–VAX Serve as the VMS Network Interface?

DECnet–VAX is the implementation of DECnet software that allows a VMS operating system to function as a network node. As the VMS network interface, DECnet–VAX supports both the protocols necessary for communicating over the network and the functions necessary for configuring, controlling, and monitoring the network.

DECnet–VAX networking software can be configured on any VMS operating system running on any VAX processor. In a DECnet network, a DECnet–VAX node can communicate with all other DECnet–VAX nodes in the network or with any other operating system that supports DECnet. In addition, a DECnet–VAX node can use a *packet switching* network to communicate with nodes on other networks, and can use gateways and other communications software and hardware products to communicate with foreign vendor systems.

DECnet–VAX is tightly coupled to VMS. It is completely integrated into the operating system and provides a natural extension of local input/output operations to remote systems. VMS users can use the network almost transparently. Implementing network applications on VMS is straightforward, and network operations are efficient. Because DECnet–VAX is a part of the VMS system, you can use DECnet–VAX interfaces as standard parts of a local, standalone VMS operating system (not connected to a network). For example, you can develop application programs that communicate directly with each other at the task level on your VMS system, and then use them in a network environment without modification. Before you can bring up your system as a node in a multinode environment, you must have a DECnet–VAX license and register a DECnet–VAX key on your system.

1.2 What Does a DECnet Network Look Like?

DECnet allows users to plan computer networks of any size and arrangement, from a few workstations linked together in one room to a very large network of powerful computers distributed around the world. The DECnet network is designed to permit growth without disruption. The network can grow from a minimum of two nodes to a maximum of over 64,000 nodes. DECnet configurations are flexible and can be expanded easily. Nodes can be located wherever required. Individual nodes can be added or relocated without impact on existing nodes or interruption of network operation.

DECnet supports many different kinds of network connections. Nodes located in a building or a complex of buildings can be connected in a *local area network* (LAN). The network can be expanded to include nodes at more geographically dispersed locations, connected in a *wide area network* (WAN). In addition, systems on a DECnet network can use other DIGITAL communications products to communicate with certain non-DECnet systems and networks in an integrated network environment.

Described below are the systems that can be connected in a DECnet network, the types of communications media used to link the systems, and the variety of network environments in which the systems can be configured.

1.2.1 What Systems Can Communicate over the Network?

DIGITAL communications software can be used to permit DIGITAL systems to communicate with each other, and, optionally, with some non-DIGITAL systems.

A VMS system, brought up as a DECnet–VAX node on the network, can communicate directly with any other VMS systems and with any other

DIGITAL system connected to the same network. All members of the VAX family of processors on which VMS systems are running can be linked, including:

- The smallest VAX processors (such as the desktop workstation, the VAXstation 2000)
- Other small VMS systems (such as the MicroVAX 2000)
- Mid-sized VMS systems (such as the VAX 8500–series processors)
- The largest VMS systems (such as the VAX 8600– and VAX 8800–series processors)

Because all VMS systems are compatible, the VMS user can maintain a consistent computing environment, and can carry out most networking operations without concern for the way the network operates.

A VMS system can also communicate with any other DIGITAL operating system on the network. For example, a VMS system running DECnet–VAX software can communicate with an ULTRIX™ system running DECnet-ULTRIX software, an RSX system running DECnet-RSX software, and a Professional 300–series system that uses PRO/DECnet software.

Several types of personal computers can join the DECnet network as end nodes. For example, DIGITAL's Rainbow 100 personal computer uses DECnet-Rainbow software to be part of the network. The VAXmate personal computer/workstation uses DECnet–VAXmate to connect to the network. Certain IBM™ personal computers can participate as DECnet end nodes by means of DECnet-DOS software.

DECnet–VAX nodes can use VAX PSI software to access directly a packet switching network (such as Telenet™ or TYMNET™) or can access the packet switching network by means of an X25router. Nodes and terminals connected to a packet switching network can use that network to communicate with each other. Packet switching networks are often used for communication over very long distances involving common carriers and satellite links.

Through special interconnect products, such as gateways and emulators, DECnet nodes can communicate with non-DIGITAL systems and networks. The DECnet/SNA gateway permits a DECnet network to connect to an IBM System Network Architecture (SNA) network. Other interconnect products permit DIGITAL systems to communicate with other vendor systems.

1.2.2 What Communications Media Does DECnet Use?

Nodes in a DECnet network can be linked by various types of data transmission media. Local area network configurations use a segment of coaxial cable or several segments of coaxial cable joined together; the cable is called the Ethernet cable. For certain environments, a thin, flexible cable called ThinWire™ Ethernet cable may be used.

™ ULTRIX is a trademark of Digital Equipment Corporation.
™ IBM is a trademark of International Business Machines Corporation.
™ Telenet is a trademark of GTE Telenet Communication Corporation.
™ TYMNET is a trademark of Tymnet, Inc.
™ ThinWire is a trademark of Digital Equipment Corporation.

Wide area networks use dedicated lines, telephone lines, microwave and satellite links, and fiber optic links. Telephone lines may be leased to provide for permanent connections, or may be used as dialup lines for specific periods of time. Communication over telephone lines normally involves the use of *modems* at each end of the connection to perform conversion between the digital signals used by the computer and the analog signals used on the telephone line. For a microwave link, a message is converted into microwave signals at the transmitting site and reconverted at the receiving location, which can be some distance away. Satellite links are usually used for very long-distance communication, such as transoceanic communication.

1.2.3 What Kinds of Network Environments Are Supported?

DECnet networks support a variety of network connections, permitting computers to be linked in flexible configurations. The basic kinds of environments in which a network can be configured are the local area network and the wide area network. A local area network provides for communications within a limited geographical area, while a wide area network permits long-distance communication. The two kinds of environments can be integrated into a single large network.

1.2.3.1 Local Area Networks

A local area network provides a high-speed communications *channel* designed to connect information processing equipment in a limited area such as a room, a building, or a cluster of buildings (for example, a campus). The DIGITAL local area network uses the Ethernet: a single, shared network channel. All nodes have equal access to the Ethernet channel. Because the Ethernet is a multiaccess device, new nodes can be added without affecting existing nodes on the Ethernet.

An Ethernet is a coaxial cable, to which each system or device is connected by a single line. In an office or other area where personal computers and workstations are located, ThinWire Ethernet cabling is usually used. The Ethernet supports a very high rate of data transmission (up to 10 million bits per second) in a limited area. The standard limit on the distance between any two nodes on the Ethernet is 1.74 miles (2.8 kilometers). An Ethernet can support up to 1,023 nodes.

Local area networks can be configured in a variety of arrangements. Two Ethernets can be connected by means of a bridge, a relay that controls network traffic between the Ethernets it connects. Use of a bridge can extend a local area network beyond the distance limitation imposed on a single Ethernet. Routers can also be used to connect two Ethernets. In addition, routing nodes on an Ethernet can be connected to wide area network nodes to form a large, integrated network.

Individual systems can either be connected directly to an Ethernet or gain access to an Ethernet by means of a local area interconnect device, the DELNI, that serves as a concentrator, grouping systems together.

Individual users can optionally gain access to the nodes in a local area network through a terminal server, if one is connected to the network. A user at a terminal connected to the terminal server can access any VMS or non-VMS service node that implements the local area transport (LAT) protocol and is known to the server. A user logged into a node by means of a terminal server can perform the same functions as a user logged in on a terminal directly connected to the node.

1.2 What Does a DECnet Network Look Like?

VMS nodes in a VAXcluster (a group of VMS systems organized to share processor and storage resources) require DECnet–VAX connections. Each node in a VAXcluster can be connected to an Ethernet that provides the DECnet data link for the cluster. If an Ethernet is not available, the computer interconnect (CI) used by the cluster can be configured to be the DECnet data link between the cluster nodes.

In a Local Area VAXcluster, each node is connected to an Ethernet cable; this Ethernet serves as the DECnet data link for the nodes in the Local Area VAXcluster. Noncluster nodes can also be connected to the same Ethernet. In addition, the Local Area VAXcluster can be connected to another Ethernet through any router on the cluster.

Figure 1–3 illustrates a small Ethernet configuration of four nodes. Three MicroVAX-based end nodes and one VMS router are connected to the Ethernet.

Figure 1–3 Example of a Small Local Area Network Configuration

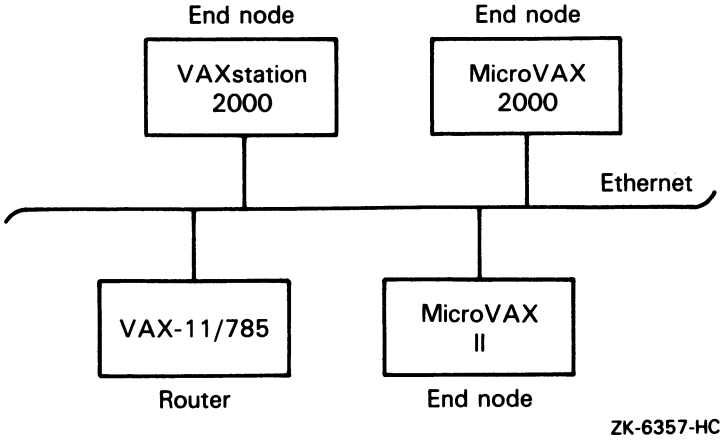
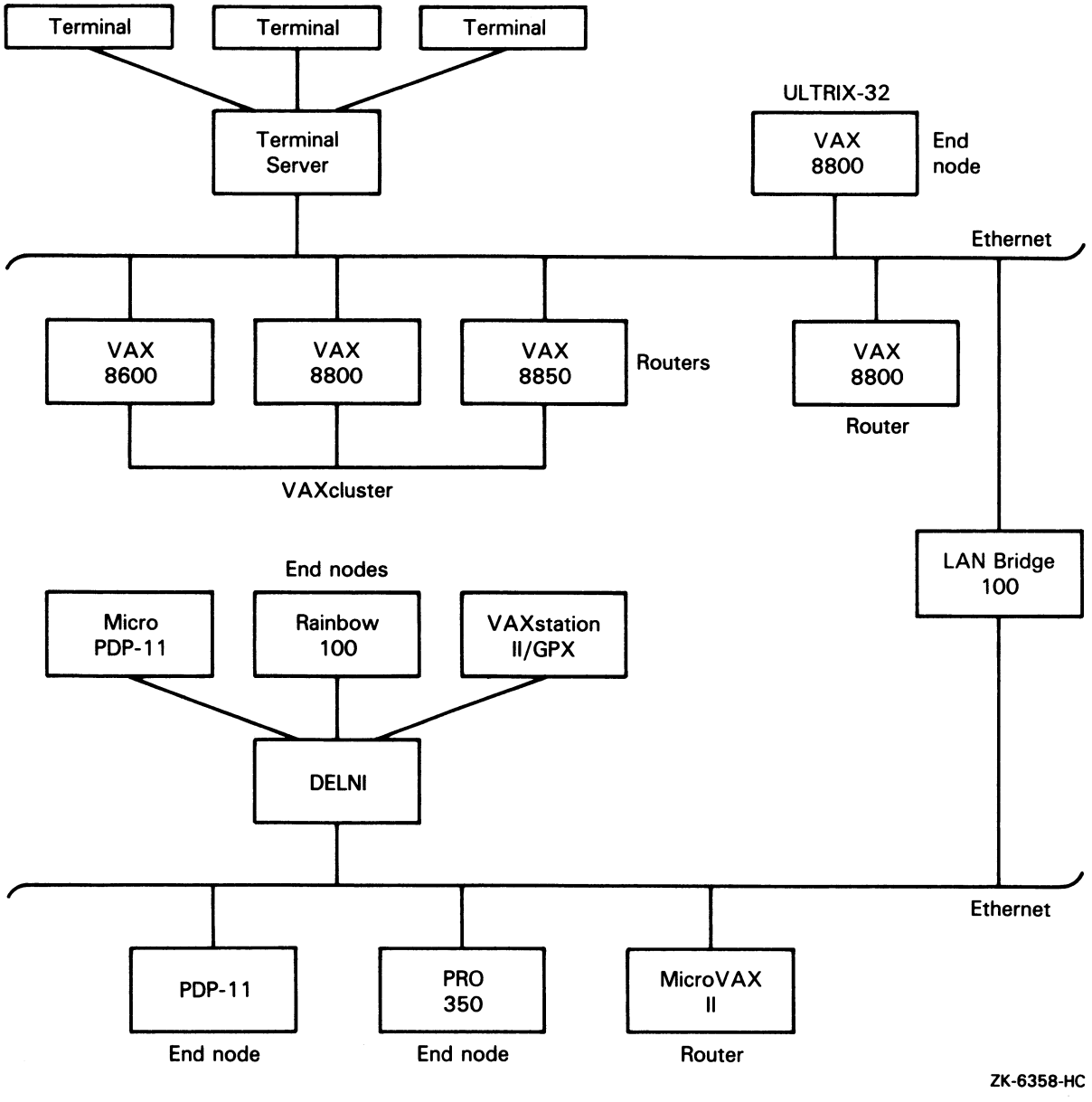


Figure 1–4 shows a larger local area network configuration in which two Ethernets are connected by a LAN bridge. Various kinds of operating systems, including the nodes in a VAXcluster, are connected directly to the Ethernet. In the figure, a group of small systems is connected to the Ethernet by means of a DELNI device. Individual terminal users can gain access to Ethernet nodes through a terminal server.

Figure 1–4 Example of a Large Local Area Network Configuration



1.2.3.2 Wide Area Networks

A wide area network provides for communication over broader geographic areas. DECnet supports long-distance communication with systems located anywhere in the world. The network can be configured to suit the needs of the user. A wide variety of communications media can be used: examples include dedicated, leased and dialup lines, and microwave and satellite links. Nodes in a wide area network can be connected by *point-to-point* lines or through packet switching networks.

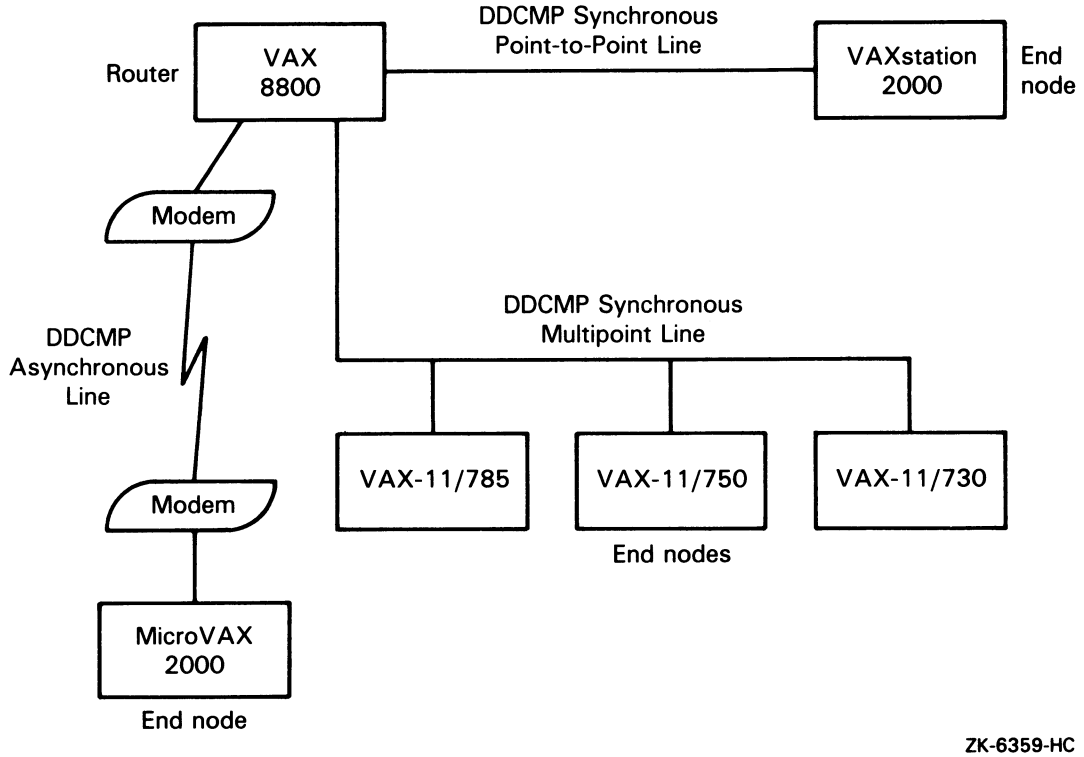
1.2 What Does a DECnet Network Look Like?

DECnet–VAX offers comprehensive wide area network support and long-haul connectivity over point-to-point and *multipoint* connections:

- Point-to-point connections, which use the Digital Data Communications Message Protocol (DDCMP), are *synchronous* or *asynchronous*. Synchronous devices provide high-speed connections over dedicated lines or telephone lines (using modems). Asynchronous devices provide low-speed, low-cost connections over terminal lines that are switched on for network use either permanently (a static connection) or temporarily (a dynamic connection). For example, a user on a MicroVAX can configure a dialup line (a telephone line) to another computer as a dynamic asynchronous DECnet line for the duration of a telephone call.
- A multipoint connection is a special form of point-to-point line: two or more nodes connected by a synchronous DDCMP communications channel, with one node controlling the channel.

Figure 1–5 illustrates the different kinds of DDCMP connections.

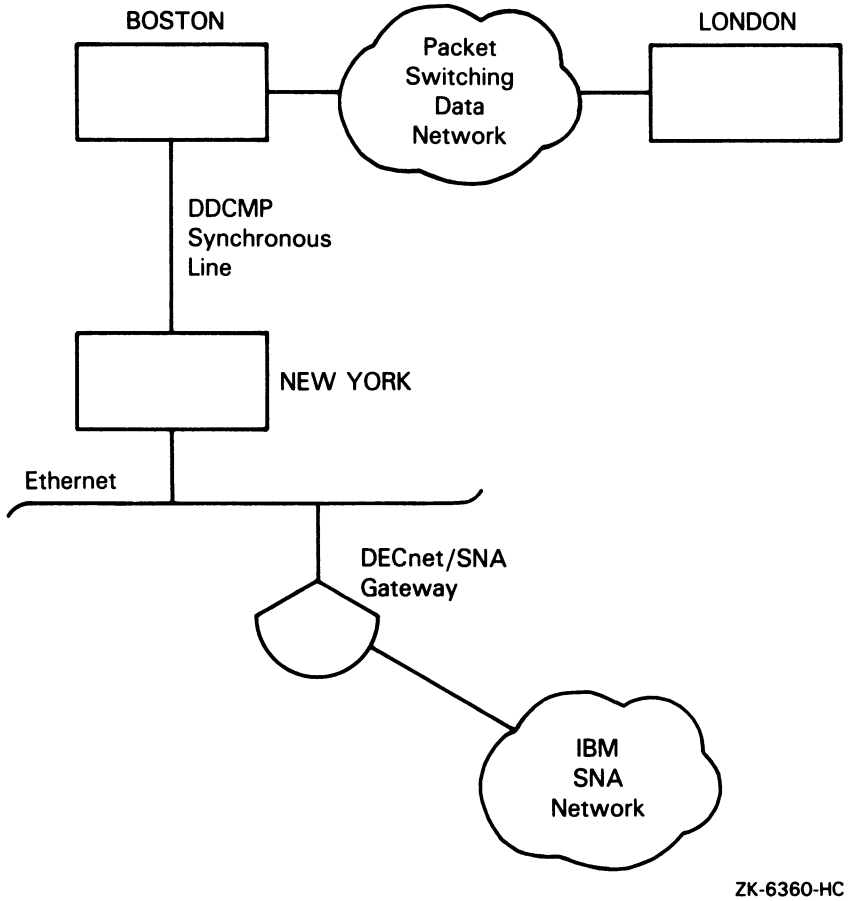
Figure 1–5 Examples of DDCMP Connections



DECnet–VAX supports worldwide communications through packet switching networks and gateways. A DECnet–VAX node can be connected to a packet switching data network (either directly using VAX PSI software or through an X25router) to establish communication with a remote node. Packet switching networks, such as TYMNET and Telenet, provide communication services between nodes on the same or different networks, often in widely dispersed geographic areas connected by satellite links. A DECnet–VAX node connected to an Ethernet can also use a DECnet/SNA gateway on the same Ethernet to communicate with IBM systems in an SNA network.

Figure 1–6 shows various wide area network connections. The figure illustrates the use of a DDCMP synchronous line to connect two VMS nodes at different locations (Boston and New York). Nodes BOSTON and LONDON are both configured to use VAX PSI software, permitting the nodes to be connected directly to an X.25 packet switching data network. Through the packet switching data network, node BOSTON can communicate with node LONDON. The figure also shows how the VMS node located in New York can communicate with IBM systems on an SNA network by means of a DECnet/SNA gateway.

Figure 1–6 Examples of Wide Area Network Connections



1.2.3.3 Integrated Networks

DECnet local area networks and wide area networks can be integrated to provide comprehensive network support. Wide area network connections can be used to connect individual local area networks, and can provide access to non-DIGITAL systems.

1.2 What Does a DECnet Network Look Like?

VMS systems can be configured to use the full possibilities of integrated DECnet networks. Figure 1–7 is an example of a large DECnet configuration that illustrates a variety of ways in which VMS operating systems on VAX processors can be connected to the network. The figure indicates whether a particular VMS node is a router or an end node. The figure also shows a terminal server connected to the Ethernet; individual terminal users can log in to any node on the extended Ethernet by means of the terminal server, provided the server is aware of the service offered by the node.

The DECnet configuration shown includes two Ethernet cables linked by a bridge. Two clusters of VMS nodes are attached to the Ethernet:

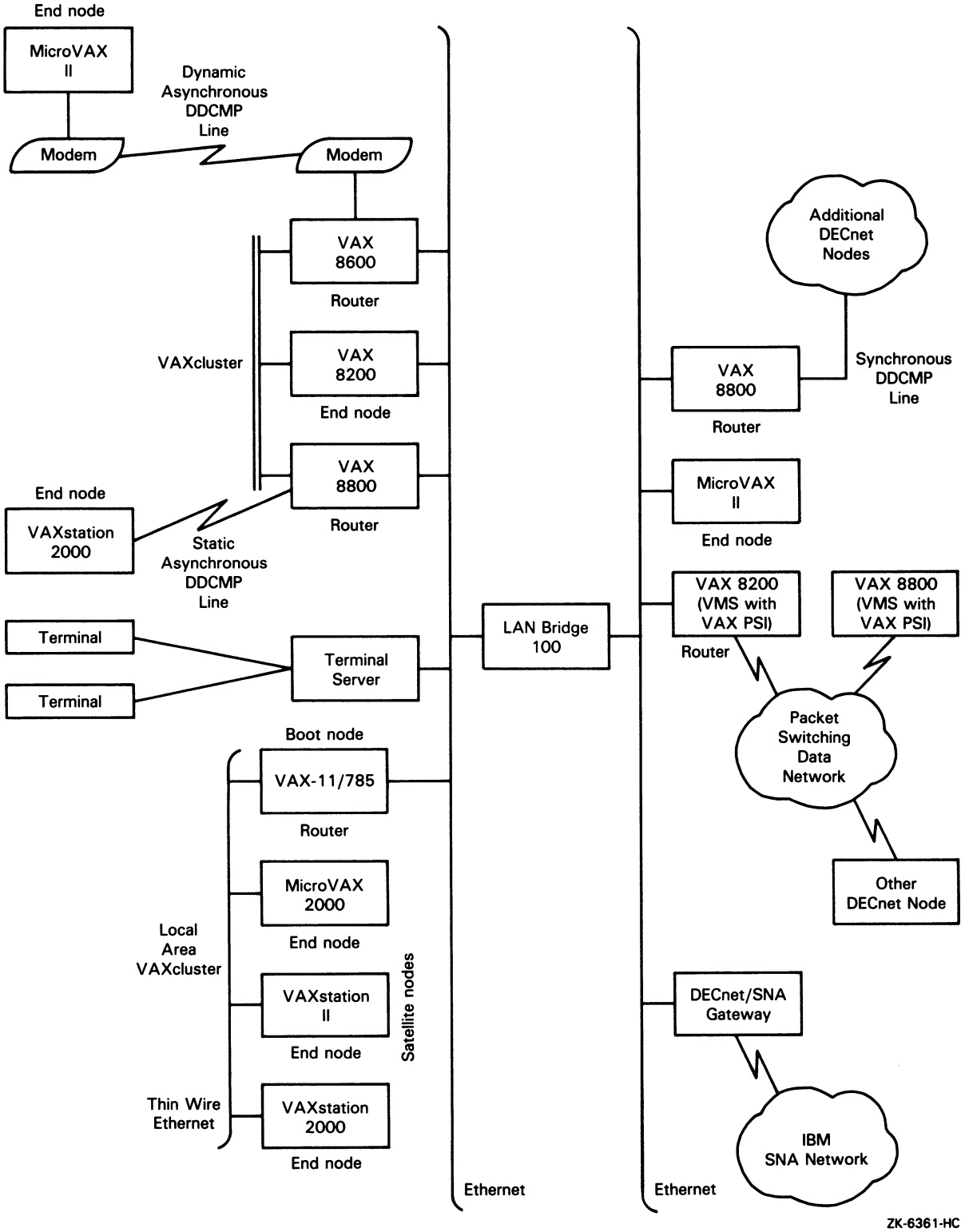
- A VAXcluster that includes three large VMS systems
- A Local Area VAXcluster that includes four smaller VMS systems (linked by their own ThinWire Ethernet)

In Figure 1–7, VMS routers and end nodes that are not members of a cluster are also connected to the Ethernet. The network diagram shows how DECnet–VAX point-to-point connections are integrated with the Ethernet multiaccess configuration:

- A synchronous DDCMP point-to-point line provides a connection to additional DECnet nodes at a remote location.
- Asynchronous DDCMP lines provide permanent (static) and temporary (dynamic) connections between VMS nodes in the network.

In the figure, VMS nodes on either Ethernet can communicate with nodes at distant locations (through a packet switching data network) and can access an IBM SNA network by means of a gateway node.

Figure 1–7 Example of a Large Integrated DECnet Configuration



2 What You Can Do over the Network

This chapter summarizes the functions that VMS general users, advanced users, and system managers can perform in a DECnet network environment. It describes the file and mail operations that a general user can conduct over the network. It explains how an advanced user can develop command procedures and application programs to run over the network. The chapter also outlines the responsibilities of the system manager of each network node, and indicates briefly how a system manager could serve as overall manager for a larger DECnet network.

2.1 What the General User Can Do over the Network

As a VMS user, you can use the DECnet network in an almost transparent manner. You can perform network operations as a natural extension of the input/output operations you perform on your own local system, because DECnet–VAX networking capability is completely integrated into the VMS operating system.

You can carry out general user operations over the network as easily as at the local node. Most DCL file-handling commands permit you to access files on remote systems in the same way as files on your own system. You can use DCL commands to perform the following operations over the network:

- You can log in to another network node on which you have an account.
- You can access common or public directories or databases located on any node on the network.
- You can display locally the contents of remote directories and files to which you have access.
- You can copy files from node to node or append files on one node to a file on another node.
- You can print files at the remote node where they reside, copy them to a remote printing device, or copy them to the local node for printing.
- Using a VMS editor, you can access and edit a file on a remote node.
- You can create a new file in a remote directory.
- With the appropriate access, you can delete or purge files from directories, search files, and compare the contents of files on different nodes.
- You can perform sort and merge operations on remote files.
- You can analyze the structure of files, convert their organization and format, and dump their contents in a specific format.
- You can back up local files by using a remote saveset.
- You can create, display, and delete logical names for nodes and devices that are to be included in remote file specifications.

What You Can Do over the Network

2.1 What the General User Can Do over the Network

Any user on a VMS node can send electronic mail to, and receive mail from, a user on any other node in the DECnet network, by means of the Mail Utility. Users can also communicate interactively over the network by means of the Phone Utility.

2.1.1 How to Gain Access to the Network

You can perform general network operations if you have an account on a VMS system that is connected to a DECnet network and have the minimum privileges TMPMBX and NETMBX. To gain access to the network, all you have to do is to log in to your account on the VMS system. (For a complete description of the procedure for accessing the network, see Section 3.1.)

Before you perform an operation over the network, you can check the availability of the network by entering the DCL command SHOW NETWORK. If you are connected to the network, the command display shows the name and address of your node. If your system is not connected to the network, the display indicates that the network is not currently available. (The SHOW NETWORK command display is described in detail in Section 3.1.1.)

After you log in to a network node, you may be able to log in to other nodes on the same network. If you are authorized to access an account on another VMS node or a non-VMS node that supports the DECnet remote command terminal facility (as described in Section 3.1.2), you can log in to that node over the network. To log in to the other node, enter the DCL command SET HOST, specifying the remote node name, and follow the login procedure the remote node uses. Once you are logged in to the remote node, you can perform all general user operations on that system as though it were the local node.

2.1.2 How to Access Remote Files

When you want to access a file on another node in the network, use a remote file specification that includes the name of the remote node on which the file resides. This section describes the format of the remote file specification and the access controls that affect access to a remote file. The section also explains how to use logical names in specifying remote directories and files, and how to specify remote files located in VAXclusters.

2.1.2.1 Remote File Specification Format

You can use VMS DCL commands to access remote files by simply including in the file specification the name of the remote node on which the file is located. To access a file that is protected against general access, include in the file specification access control information (a user name and password and, optionally, an account name) indicating that you are authorized to access the file (see Section 2.1.2.2).

The usual format for a remote file specification is as follows:

```
node"username password"::device:[directory]filename.type;version
```

For example, to identify the file EXAMPLE.LIS residing in the directory INFORMATION on device DBA1 at node BOSTON, use the following file specification:

```
BOSTON::DBA1:[INFORMATION]EXAMPLE.LIS
```

If a file resides on a non-VMS system, enclose the name of the file in quotation marks. For example, to access a file on an ULTRIX node named U32, use the following file specification:

```
U32::"/user/information/example.lis"
```

2.1.2.2 Remote File Access Controls

The owner of a file can use the SET PROTECTION command to protect a file or directory against access over the network. One way to access a protected remote file is to supply access control information in the file specification. If the user name is BLACK and the password is LX2431, you could use the following file specification:

```
BOSTON"BLACK LX2431"::DBA1:[INFORMATION]EXAMPLE.LIS
```

To avoid using a password in a file specification to be transmitted over the network, you may want to have a *proxy* account at the remote node that permits you to access specific directories and files as though you were a local user. If you have proxy access to a remote file, you can omit the access control information when specifying that file name, even if the file is protected against outside access. For example, use this file specification to access the file TASKS.LIS in the directory PROJECT on device WORK at remote node TUCSON, at which you have a proxy account:

```
TUCSON::WORK:[PROJECT]TASKS.LIS
```

(For more information on the use of proxies over the network, see Section 3.3.7.1.)

If the remote node system manager has established a default DECnet account for the remote node (as described in Section 3.3.7.2), you can specify a null access control string in the remote file specification to invoke the default DECnet account. For example, the following file specification causes the file TRENDS.DAT at remote node LONDON to be accessed using the default DECnet account:

```
LONDON"":DBA0:[MARKET]TRENDS.DAT
```

When you access a remote file, a process at the remote system actually performs the file access on your behalf. The remote process follows the rules normally used to access files on that system. The rights and privileges that the remote process uses to access the file depend on the user name supplied. The user name can be one of the following (in order of precedence):

1. A user name that you supply explicitly in an access control string included in the remote file specification. (If you specify a null access control string, the user name in the default DECnet account is used.)
2. The user name in any proxy account that you have at the remote node.
3. The user name in the default DECnet account at the remote node (by default, that user name is DECNET).

2.1.2.3 Logical Names in Remote File Specifications

For convenience, you may want to use logical names to define portions of remote file specifications. A logical name can represent a remote node name, a device name, and, optionally, a directory name. You can use the DCL command DEFINE to create a logical name in the process logical name table. The following example equates the logical name CITY to the equivalence name BOSTON::DBA1: and then uses the logical name in a remote file specification:

```
$ DEFINE CITY BOSTON::DBA1:
$ TYPE CITY:[INFORMATION]EXAMPLE.LIS
```

To represent an access control string in the partial file specification, enclose the access control string in quotation marks, as in the following example:

```
$ DEFINE CITY "BOSTON""BLACK LX2431"""::DBA1:"
```

You can also use the logical name commands ASSIGN and DEASSIGN to indicate portions of remote file specifications. Logical name translation is performed locally and does not affect the network file operation.

If you do not specify the name of a device or directory in a remote file specification, the remote system applies a default name.

2.1.2.4 VAXcluster File Specifications

In a VAXcluster, cluster members must have DECnet–VAX connections, but directories or files can be shared by users on different nodes in the cluster without actually using DECnet. In a cluster, disk volumes are mounted on all cluster nodes, and node names need not be used to access directories or files.

If you are on a noncluster node, use a normal remote file specification to access cluster directories and files. In the remote file specification, you can include the name of a node on which the directory or file resides, or, if the cluster uses an *alias node name*, you can include that alias. The alias node name is a special clusterwide node name that permits outside users to address the cluster as though it were a single node. For example, the cluster alias node name SOURCE is used by nodes A, B and C in a VAXcluster. To display the contents of the public directory COMMON on device DATA in the cluster, enter the following command:

```
$ DIRECTORY SOURCE::DATA:[COMMON]
```

2.1.3 Network File Operations

Most DCL commands used to perform file-handling operations are supported over the network. The following paragraphs illustrate ways in which you can use DCL commands in a network environment.

2.1.3.1 Displaying Remote Directories and Files

To list the files in a remote directory, use the DIRECTORY command. For example, the following command lists all files cataloged in the directory [COMMENTS.PUBLIC] at remote node ALBANY:

```
$ DIRECTORY ALBANY::DISK1:[COMMENTS.PUBLIC]
```

You can also use the DIRECTORY command to list all versions of a particular file in a remote directory, or all attributes of a specific file or files.

The TYPE command enables you to display on your current output device (such as your terminal screen) the contents of one or more remote files to which you have access. For example, to display the contents of the unprotected file MEMBERS.LIS in directory [CLUB] at remote node ALBANY, use the following command:

```
$ TYPE ALBANY::DISK1:[CLUB]MEMBERS
```

If the remote directory is on the default device, you can omit the device name in the TYPE command. For example, the following command causes the file USERDATA.LIS in directory [ACCOUNT] on the default device at node NWYORK to be displayed at the local terminal:

```
$ TYPE NWYORK::[ACCOUNT]USERDATA
```

In a distributed processing environment, information of interest to a number of users on the network may be stored in central directories or databases accessible to everyone on the network. To make access to a public directory easier, define a logical name for the public directory. For example, you can use the logical name PUBLIC to define the public directory [COMMENTS.PUBLIC] at remote node ALBANY:

```
$ DEFINE PUBLIC ALBANY::DISK1:[COMMENTS.PUBLIC]
```

Users logged in to any node on the network can then access the file APPROVALS.TXT located in the directory [COMMENTS.PUBLIC]. For example:

```
$ DIRECTORY PUBLIC
$ TYPE PUBLIC::APPROVALS.TXT
```

They can also copy the file to the local node and then print it, as described in the next section.

2.1.3.2 Copying and Printing Remote Files

Use the COPY command to copy a file from one node to another. As part of the copy operation, you can create a new file from existing files. In the following examples, the owners of the remote directories specified have given other users access to the directories and files. This command copies the file TEST.DAT on node BOSTON to a new file of the same name on remote node LONDON:

```
$ COPY BOSTON::DISK:TEST.DAT;2
_To: LONDON::DBA0:[PRODUCT]TEST.DAT
```

Both of the following commands copy the file NAMES.LIS from the local node to a file with the same name at remote node ALBANY:

```
$ COPY NAMES.LIS ALBANY::DISK1:[CLUB]NAMES.LIS
$ COPY NAMES.LIS ALBANY::DISK1:[CLUB]
```

The following command is the wildcard form of the COPY command. The latest versions of all files in the user's default directory at the local node are copied to the remote node NWYORK.

```
$ COPY *.* NWYORK::[ACCOUNT]*.*
```

The next command copies two local files, USER1.DAT and USER2.DAT, to a single new file, USERS.DAT, at remote node NWYORK:

```
$ COPY USER1.DAT,USER2.DAT NWYORK::[ACCOUNT]USERS.DAT
```

To specify explicitly the access privileges of a particular account on a remote node when copying a file from that node, include the user name and password for that account in the file specification, as in the following example:

```
$ COPY NWYORK"BROWN CHECKING"::[STATISTICS]SUMMARY.LIS *
```

Use the APPEND command to add the contents of one or more input files to the end of an output file located at a different node. For example, to add the contents of the files DATA1.LIS and DATA2.LIS at remote node NWYORK to the end of the file RESULTS.LIS at node LONDON, use the following command:

```
$ APPEND NWYORK"BROWN CHECKING"::[TESTING]DATA1.LIS,DATA2.LIS
_To: LONDON::DBA0:[PRODUCT]RESULTS.LIS
```

To queue a file for printing at the remote node on which it exists, specify the remote file specification in the PRINT/REMOTE command. The PRINT /REMOTE command does not copy the file to the remote node; you must enter a separate COPY command if the file does not reside at the remote node on which it is to be printed. For example, the following commands cause the local file REPORT.LIS to be copied to the default DECnet directory at remote node TUCSON and queued for printing at node TUCSON.

```
$ COPY REPORT.LIS TUCSON::WORK:[PROJECT]
$ PRINT/REMOTE TUCSON::REPORT
```

Alternatively, you can copy a file to a printing device on the remote system. If the device is spooled (for example, a line printer) the file will be temporarily stored on disk and entered in the print queue for the device. For example, this command performs the same operation as the two previous commands, causing the local file to be printed at the line printer at node TUCSON under the default nonprivileged DECnet account:

```
$ COPY REPORT.LIS TUCSON::LPA0:
```

Note that the /REMOTE qualifier is required in the PRINT command whenever a remote file is specified, and you cannot include any other qualifiers in this command. The PRINT/REMOTE command supplies a default file type of LIS.

2.1.3.3 Creating and Editing Remote Files

Using a VMS editor, you can create a file at a remote node and modify the file. You can also edit an existing file on a remote node. To perform editing on a remote file, simply include the node name of the remote file when you invoke the editor. If the file is protected against outside access and you do not have a proxy account at the remote node, you must specify an appropriate access control string as part of the remote file specification in the edit command.

To invoke the EDT editor to perform editing on the file STORY.TXT in the directory [MANUSCRIPT] on remote node ZURICH on which you have a proxy account, enter the following command:

```
$ EDIT ZURICH::[MANUSCRIPT]STORY.TXT
```

You can then perform all normal editing operations on the file STORY.TXT. In the same way, you can invoke any other VMS-supported editor to edit a remote file.

To create a sequential disk file on a remote node without invoking an editor, you can use the DCL command CREATE. For example, use this command to create a sequential file named INPUT.DAT on remote node NWYORK:

```
$ CREATE NWYORK"BROWN CHECKING"::[STATISTICS]INPUT.DAT
1,046,214
2,307,625
1,988,723
^Z
$
```

2.1.3.4 Deleting and Purging Remote Files

If you have access to a remote directory, you can delete or purge files from the directory. Use the DELETE command to delete one or more files from a mass storage volume on a remote node. In the DELETE command, you must supply an explicit version number in any file specification that is not enclosed in quotation marks. If you want to delete the highest numbered version, specify a version number of zero (;0) or null (;). To delete all versions, specify the wildcard character as the version number (;*).

For example, to delete the file DETAILS.LIS;2 in the directory INFORMATION at remote node BOSTON, use this command:

```
$ DELETE BOSTON"BLACK LX2431"::DBA1:[INFORMATION]DETAILS.LIS;2
```

If you have a proxy account that allows you full access to the directory SUGGESTIONS on remote node TUCSON, you can delete all files in that directory, as follows:

```
$ DELETE TUCSON::WORK:[SUGGESTIONS]*.*;*
```

The PURGE command deletes all but the highest-numbered version or versions of one or more files residing at a remote node. To purge all but the two highest-numbered versions of each file of the type LIS in the directory PROPOSALS at remote node TUCSON, use this command:

```
$ PURGE TUCSON::WORK:[PROPOSALS]*.LIS/KEEP=2
```

2.1.3.5 Searching, Comparing and Sorting Remote Files

DCL commands permit you to search remote files for specific information, compare two remote files to determine any differences, and sort and merge remote files.

Use the SEARCH command to search one or more remote files for a specified string or strings. The command lists all occurrences of the string. The following SEARCH command causes the files MEMBERS.LIS and DATA.LIS at remote node ALBANY to be searched for all occurrences of the character string NAME.

```
$ SEARCH ALBANY::DISK1:[CLUB]MEMBERS.LIS,DATA.LIS
_String(s):   NAME
```

To compare the contents of two files (either of which can be local or remote) on a record-by-record basis, use the DIFFERENCES command. The command produces an output file listing any differences. For example, this command compares the two highest versions of the file TEST.DAT in the nonprivileged default DECnet account on remote node BOSTON:

```
$ DIFFERENCES BOSTON::TEST.DAT
```

The following command compares two remote files and displays any differences found. The first file is TEST.DAT at remote node BOSTON and the second file is TEST.DAT at remote node LONDON.

```
$ DIFFERENCES BOSTON::TEST.DAT LONDON::DBA0:[PRODUCT]TEST.DAT
```

To perform sort and merge operations on remote files, invoke the VMS Sort /Merge Utility. Use the SORT command to reorder records in a remote file and create a new output file (or an address file that you can use to access the reordered records). Use the MERGE command to combine two or more sorted files into a single output file that the utility program creates. The files to be combined must be similarly sorted, but can reside at different VMS nodes.

For example, the following command requests a default alphanumeric sort of the records in the file RANDOM.FIL at remote node BOSTON. The SORT program sorts the records on the basis of the contents of the first seven characters in each record and writes the sorted list into the output file ALPHANM.SRT created in the default directory at the local node.

```
$ SORT/KEY=(POSITION:1,SIZE:7) -
_$ BOSTON::DBA1:[RECORDS]RANDOM.FIL ALPHANM.SRT
```

The example of a merge command shown below causes two identically sorted files, FILE1.SRT and FILE2.SRT, on the directory PROJECT at remote node TUCSON to be merged into an output file. This output file, MERGEFILE.DAT, is created at the current default directory at the local node. The input file qualifier /CHECK_SEQUENCE is specified to ensure that the input files are sorted in the correct order.

```
$ MERGE/KEY=(POSITION:1,SIZE:30) -
_$ TUCSON::WORK:[PROJECT]FILE1.SRT,FILE2.SRT/CHECK_SEQUENCE -
_$ MERGEFILE.DAT
```

2.1.3.6 Examining Remote Files and Records

DCL commands that analyze the internal structure of certain files, convert the organization and record format of files, and dump the contents of files in specific data formats are supported in a network environment.

Use the ANALYZE/RMS_FILE command to analyze the internal structure of a remote VAX RMS file. Optionally, you can use the command to generate an FDL (File Definition Language) file. Command qualifiers permit you to check the file structure for errors, obtain statistics on the file structure and use, or enter interactive mode to explore the structure of the file. The following command analyzes the structure of the file RUN.DAT at remote node TUCSON:

```
$ ANALYZE/RMS_FILE TUCSON::WORK:[PRODUCTION]RUN.DAT
```

The CONVERT command allows you to transfer records from a source data file to a second data file, that can differ in file organization and format from the first. You can use this command to transfer files to or from a remote node while altering file attributes. If the output file exists, the Convert Utility changes the organization and format of the data from the input file to that of the output file. If the output file does not exist, the utility creates it from the file attributes specified in an FDL file. You can also use the CONVERT command to copy files to a remote node or to retrieve them without modifying file attributes. However, CONVERT transfers a file record by record, not using block I/O.

The following command causes records from the file SALES.TMP at the local node to be added sequentially to the end of the output file SALES.CMD at remote node BOSTON. The file SALES.TMP is sequential with variable-length record format, and the file SALES.CMD is sequential with stream record format. When the Convert Utility loads records from the input file to the output file, it changes the record format.

```
$ CONVERT/APPEND SALES.TMP BOSTON::DBA1:[RECORDS]SALES.CMD
```

Use the DUMP/RECORDS command to display the contents of remote files in ASCII, hexadecimal, decimal, or octal representation. The DUMP command qualifiers /ALLOCATED and /BLOCKS are not supported in the network context. The following command dumps the contents of the file CALC.DAT, which resides at remote node BOSTON. The command formats the output both in octal words and in character strings, and queues the output to the system printer at the local node.

```
$ DUMP/RECORDS/OCTAL/WORD
_File(s):  BOSTON::DBA1:[RECORDS]CALC.DAT/PRINTER
```

2.1.3.7 Backing Up Files over the Network

You can use the BACKUP command to save local files in a BACKUP saveset residing on a remote VMS node. You can also use this command to restore on the local node files that were previously saved in a saveset on a remote VMS node. Use BACKUP/LIST to display the names and attributes of files cataloged in a remote saveset. The BACKUP saveset cannot be on magnetic tape.

The following command saves the files in the directory SCHED on disk DB1 at the local node in the BACKUP saveset SCH.BCK at remote node MIAMI. The /SAVE_SET qualifier is required to identify the output specifier as a saveset on a Files-11 medium.

```
$ BACKUP
_From:  DB1:[SCHED]*.*
_To:    MIAMI::DBA2:[SAVE]SCH.BCK/SAVE_SET
```

2.1.3.8 Error Messages Displayed During Remote File Operations

When you enter a DCL command to perform a network file operation that does not complete successfully, one or more error messages are displayed. Typically, the sequence of error messages includes a primary error message generated by the DCL command interpreter and a secondary error message generated by VAX Record Management Services (RMS). These messages can optionally be followed by an additional error message associated with the secondary RMS error (from a facility involved in the network file operation).

For example, an attempt to copy the local file INDEX.DAT to TEMP.DAT at remote node SYDNEY failed because SYDNEY is a non-VMS node that does not support indexed files. The following error messages were generated by the DCL command interpreter, RMS, and the remote File Access Listener (FAL) file server utility.

```
$ COPY INDEX.DAT SYDNEY::TEMP.DAT

%COPY-E-OPENOUT, error opening _SYDNEY::TEMP.DAT; as output
-RMS-F-SUPPORT, network operation not supported
-FAL-F-ORG, file organization field rejected
```

2.1.4 Using MAIL and PHONE in a Network Environment

You can use electronic mail to exchange messages with any user on the network. Mail can be exchanged between all VMS and non-VMS nodes that support the Mail Utility, including nodes connected to the DECnet network by means of gateways or packet switching networks.

Any VMS user on the network can send electronic mail to, and receive mail from, any other user on the network by means of the Mail Utility. To address the mail message to the intended recipient on the remote node, you normally use the format *nodename::username*. DECnet–VAX translates the node name to a node address before transmitting the message over the network. At the receiving end, DECnet–VAX translates the address back into a node name before displaying the message to the recipient. For example, to send mail to user SMITH on remote node NWYORK, invoke mail and enter the following:

```
$ MAIL
MAIL> SEND
To:     NWYORK::SMITH
```

If the network connection to the remote node is not available, you receive the following message:

```
_SYSTEM-F-UNREACHABLE, remote node is not currently reachable
```

When someone on a remote node sends you mail, the sender is identified by node name as well as user name. For example, if user JONES at node BOSTON sends you a mail message over the network, the sender is identified in the following way:

```
From:     BOSTON::JONES
```

2.1 What the General User Can Do over the Network

If you are logged in to a network node, you may receive notification of mail arriving from a remote node. The notice displayed on your screen is in the form

```
New mail on node 'local-nodename' from 'remote-nodename::username'
```

For example, if user JONES on node BOSTON sends you mail on node PURPLE, this notice is displayed on your screen:

```
New mail on node PURPLE from BOSTON::JONES
```

The Mail Utility normally permits cluster alias node names and addresses to be used for incoming and outgoing messages. The use of the cluster alias permits MAIL to treat a cluster as though it were a single node. When you send mail to someone whose node is a member of a cluster that uses an alias node name, you can specify either the cluster alias or the user's node name.

If you are logged in to a VAXcluster node that uses an alias node name, the Mail Utility uses the alias node name, rather than the name of your individual node, to identify any mail message you send. An incoming reply directed to the alias node name is given to any active node in the cluster and then delivered to your mail file. Consequently, if you are on a cluster node that uses an alias, the notification of incoming mail on your screen indicates the name of the cluster node that received the message.

For example, user ALLEN is logged in to node PURPLE in a cluster that uses the alias CLUST1. When he receives a reply to a message he sent user JONES at remote node BOSTON, the message will indicate the cluster node name CLUST1 (rather than the node name PURPLE), as in the following:

```
From:  BOSTON::JONES
To:    CLUST1::ALLEN
```

Additionally, the mail notification that user ALLEN receives may indicate that the mail was received at a different node (for example, GREEN) in the same cluster, as in the following:

```
New mail on node GREEN from BOSTON::JONES
```

You can send either messages or files over the network. If you are composing a long message for transmission to a remote node, you may prefer to use an editor to create the message file and then invoke MAIL to transmit the file. This method permits you to avoid the possibility of losing the network connection before you complete your message.

To contact VMS users over the network, you can also use the Phone Utility, which allows you to have an "online" conversation with a user on another VMS node that supports PHONE. To address a user on a remote node, use the format *nodename::username*. Your outgoing connection identifies your local node. During your conversation, PHONE creates a number of incoming links addressed to your node. You should not use a cluster alias node name with the Phone Utility because links addressed to a cluster alias node name can be assigned to any node in the cluster.

2.2 What the Advanced User Can Do over the Network

Advanced DECnet–VAX users and programmers can write command procedures and programs that make the network seem transparent to the user. Accessing a file on a remote node is conceptually the same as accessing a file on the local system. To access a remote file in a command procedure or application program, you need only include in your file specification the name of the remote node and any required access control information.

You can execute command procedures locally that access remote files. You can also submit command procedures for batch execution on remote nodes. Additionally, you can prepare command procedures that cause tasks to be executed at remote nodes.

Developing an application program to run on the network does not require any special changes to the program, because DECnet–VAX is integrated within the VMS operating system. DECnet–VAX provides the user with access to network capabilities through higher-level languages, and through VMS Record Management Services (RMS) and system services, as follows:

- You can access remote files by means of standard I/O statements in higher-level language programs (for example, programs written in FORTRAN, BASIC, PL/1, Pascal, COBOL and C). Regardless of the higher-level language in which a program is written, you can access remote files exactly as you would access local files.
- You can access remote files within MACRO programs by means of standard RMS or system service calls; no DECnet-specific calls are required.

Task-to-task communications, a feature common to all DECnet implementations, allows two application programs running on the same or different operating systems to communicate with each other regardless of the programming languages used. Examples of network applications are distributed processing applications, transaction processing applications, and applications providing connection to servers.

2.2.1 Remote Command Procedures

Within command procedures, you can access remote files as though they were local files. You can use command procedures in a network environment as follows:

- Within command procedures to be executed locally, you can use DCL commands to open and close remote files, and read and write records in these files, using the same qualifiers as for local files. You can also use lexical functions that return information about remote files.
- You can submit DCL command procedures residing on remote nodes for execution as batch jobs on those nodes.
- You can write command procedures to cause tasks to be run at remote nodes.

The following sections summarize ways that you can use command procedures over the network.

2.2.1.1 Accessing Remote Files in Command Procedures

In a command procedure to be executed locally, you can use DCL commands that access remote files. You can use the same command and file qualifiers in the DCL commands OPEN, CLOSE, READ, and WRITE that you would normally use in command procedures to access local files.

Use the OPEN command to open a file for reading or writing and the CLOSE comand to close that file. Use the READ command to read a single record from a specified remote input file, and the WRITE command to write a record to a specified output file. The following example from a command procedure indicates how to write a single line of text to the file EXAMPLE.LIS at remote node BOSTON:

```
$ OPEN/WRITE OUTPUT_FILE BOSTON::DBA1:[INFORMATION]EXAMPLE.LIS
$ WRITE OUTPUT_FILE "Preliminary examples to be supplied"
```

DCL command procedures that are executed locally can include lexical functions that return information about remote files. You can use the following functions to parse or search a remote file or to obtain information about the attributes of the file:

F$FILE_ATTRIBUTES	Returns attribute information about a remote file
F$PARSE	Returns a partial or a full file specification for a remote node
F$SEARCH	Returns the full file specification for the next remote file that matches the given wildcard file specification

2.2.1.2 Submitting Command Procedures for Remote Execution

To submit a command procedure for execution at a remote node, use the DCL command, SUBMIT/REMOTE. This command causes a command procedure residing at a remote node to be entered in the batch job queue for execution at the remote node. The SUBMIT/REMOTE command does not copy the files to the remote node; you must enter a separate COPY command if the file is not already located at the remote node. The /REMOTE qualifier is required in the SUBMIT command if a node name is included in the file specification. No other qualifiers are allowed in the SUBMIT/REMOTE command. The default file type is COM.

For example, enter the following command to submit a command procedure for execution at remote node TUCSON. The command procedure is SCHEDULE.COM in the directory PROJECT on device WORK at node TUCSON.

```
$ SUBMIT/REMOTE TUCSON::WORK:[PROJECT]SCHEDULE
```

If the command procedure JOB.COM resides at the local node, enter the following command to cause the file to be copied to node TUCSON and executed as a batch job:

```
$ COPY JOB.COM TUCSON::WORK:[PROJECT]
$ SUBMIT TUCSON::WORK:[PROJECT]JOB.COM/REMOTE
```

2.2.1.3 Using Command Procedures to Run Remote Tasks

You can specify in a DCL command the name of a command procedure that causes a task to be executed at a remote node. In the command, the file name of the remote command procedure is represented as a task. One form of task specification string that you can use to identify a remote task is as follows:

```
node-specification::"TASK=taskname"
```

Additional information on specifying tasks to be executed at remote nodes is given in Section 2.2.2.

You can use the DCL command TYPE to execute a command procedure at a remote node. In the following example, the TYPE command causes the command procedure SHOWSUM.COM to be run at remote node NWYORK:

```
$ TYPE NWYORK"BROWN CHECKING"::"TASK=SHOWSUM"
```

The following example shows a command procedure that allows you to execute a DCL command at a remote node and have the output displayed at the local node. The command procedure, SHOWUS.COM, causes the command SHOW USER to be executed at the remote node. The SHOWUS.COM command procedure resides at the remote node in the SYS$LOGIN directory of whatever account is accessed.

```
$!            S H O W U S . C O M
$ if f$mode() .eqs. "NETWORK" then define/user sys$output sys$net
$ show user
```

You can then enter a TYPE command to display locally the results of executing the command procedure at the remote node. For example, use this command to request execution of SHOWUS.COM, which resides in the default DECnet account at remote node BOSTON:

```
$ TYPE BOSTON::"TASK=SHOWUS"
```

The resulting list of interactive users at node BOSTON is displayed at your local node.

```
        VAX/VMS Interactive Users
         15-APR-1988 13:50:13.30
   Total number of interactive users = 3
Username     Process Name      PID     Terminal
SMITH        SMITH         0000002C  TXA1:
JONES        JONES_1       00000030  TXA3:
JONES        JONES         00000032  RTA1:
```

2.2.2 Network Applications Using Task-to-Task Communication

DECnet task-to-task communication allows an application program on a network node to exchange data with another program running on a remote node. Task-to-task communications can be transparent or nontransparent. Transparent communication allows users to move data across the network without necessarily knowing they are using DECnet software. Nontransparent communication involves using network-specific features in the communication process.

For DECnet–VAX, task-to-task communication is performed as though a remote file were being accessed. In DECnet terms, a task is an image running in the context of a process. A special quoted string, the task specification string, is used in a remote node specification to indicate the remote task to which you attempt to connect. The task can be identified in the program

either by name or object number. A user-defined task is usually identified by network object number 0, but it can optionally be assigned a nonzero object number, in the range from 128 to 255. A nonzero object number can be specified without a task name. (Specific network services are also identified by nonzero object numbers; for example, 27 represents the Mail Utility object.)

The format of the task specification string can be any of the following (note that *n* is any nonzero object number):

```
node-specification::"TASK=taskname"
node-specification::"0=taskname"
node-specification::"n="
```

For example, each of the following specifications identifies the remote task TEST2 with object number 0:

```
BOSTON::"TASK=TEST2"
BOSTON::"0=TEST2"
```

In transparent task-to-task communication applications, a higher-level language task can access a remote task and exchange information. To access the remote task, the higher-level language program uses standard I/O statements that include a remote file specification identifying the task. MACRO programs access remote tasks in the same way. Transparent communication at the system-service level provides all the basic functions necessary for two tasks to exchange messages over the network, without requiring any DECnet-specific calls.

Nontransparent task-to-task communication extends this basic set of functions to allow a nontransparent task to receive multiple inbound connections and to use additional network protocol features such as optional user data and interrupt messages. The nontransparent program written in MACRO includes additional system service and I/O functions supported by DECnet–VAX. Nontransparent task-to-task communication allows you to coordinate a more controlled communication environment for exchanging information.

Example 2–1 illustrates a nontransparent communications application, written in the VAX C language. The example permits a user at one node to submit an inquiry to a database at a remote node and receive a response. The application consists of two nontransparent task-to-task programs. The first program is the database request program on the local node; the other is the database server program on the remote node. A user at the local node provides input in the form of a name key to the requesting program. This program transmits the key to the database server program, which executes a database inquiry and sends the response back to the user.

In the example, DB_REQUESTER is a nontransparent source task on the local node that communicates with a nontransparent target task, DB_SERVER, on node BIGRED. The task DB_SERVER executes a database inquiry at the target node using key information that is input at the originating node. The source task uses a network connect block (NCB) and assigns a channel to establish communication with the target task. DB_SERVER declares itself to be a network object to permit it to receive more than one connection request at a time. DB_SERVER also uses a mailbox to receive notifications of network status.

Note that nonnetwork code is held to a minimum in Example 2–1. The complete programs are available on the VMS system distribution medium. To access the programs, specify the file names SYS$EXAMPLES:DB_REQUESTER.C and SYS$EXAMPLES:DB_SERVER.C.

Example 2–1 Example of a Nontransparent Communication Program

```
/*      .TITLE  DB_REQUESTER - Database request program                 *
 *      .IDENT  'X-1'                                                   */
/* ABSTRACT:    DB_REQUESTER                                           */

/* This program demonstrates how to perform nontransparent             *
 * task-to-task communication with a known network object, DB_SERVER, on *
 * a remote node.  The program accepts key information (a name) from the *
 * user and sends it to the DB_SERVER process at the target node.      *
 * DB_SERVER executes a database inquiry based on the key information and *
 * returns a response to the requester.                                */

    struct net_con_blk {       /*  Network Connect Block (NCB) */
        char node[8];
        char task_delimiter;
        char task[15];
        short int resrvd;
        char ncb_end;
    } ncb = { "BIGRED::",'"',"TASK=DB_SERVER/",0,'"' };

    static $DESCRIPTOR (net_device, "_NET:");

main ()

/* After initialization, this program processes requests entered by the  *
 * user at the name prompt until CTRL/Z is entered.  Processing is       *
 * synchronous.  After accepting each name input, the program sends a    *
 * request to DB_SERVER over the logical link, then waits until it receives *
 * and displays a response before prompting for another name.          */
{
    return_status = initialization () ;
    if ( return_status & STS$M_SUCCESS ) {
        return_status = inquire_name () ;
        while ( return_status & STS$M_SUCCESS ) {
            return_status = issue_request () ;
            if ( return_status & STS$M_SUCCESS ) {
                return_status = rcv_and_disp_response ();
                if ( return_status & STS$M_SUCCESS )
                   return_status = inquire_name () ;
        }   }
        if ( return_status != RMS$_EOF )
            exit ( return_status );
    }
    else
       exit ( return_status );
}
```

Example 2–1 Cont'd. on next page

Example 2–1 (Cont.) Example of a Nontransparent Communication Program

```
initialization ()

/* Set up descriptors and establish communication with the DB_SERVER      *
 * process.  Requesting a logical link over DECnet using nontransparent   *
 * communications involves three steps:                                   *
 *   1. Create a temporary mailbox (optional).                            *
 *   2. Assign a channel to the _NET device and associate the temporary   *
 *      mailbox with it.                                                  *
 *   3. Issue a QIO with a function code of IO$_ACCESS and the P2         *
 *      parameter containing the address of the descriptor of the network *
 *      connect block (NCB).                                              *
 * In this program, steps 1 and 2 are combined through the use of the     *
 * Run-Time Library routine LIB$ASN_WTH_MBX.                              */

{
    return_status = lib$asn_wth_mbx ( &net_device, &max_msg, &buf_quo,
                                      &net_chan, &mbx_chan );
    if ( return_status & STS$M_SUCCESS ) {
        return_status = sys$qiow ( 0, net_chan, IO$_ACCESS, &iosb, 0, 0, 0,
                                   &ncb_desc, 0, 0, 0, 0 );
        if ( return_status & STS$M_SUCCESS )
            return_status = iosb.status;
    }
    return return_status;
}
inquire_name ()    /*  Prompt the user for the query name. */
{
    return lib$get_input ( &inq_name, &input_prompt, 0 );
}

issue_request ()   /* Issue writes, (IO$_WRITEVBLK), over the logical link.*/
{
    return_status = sys$qiow ( 0, net_chan, IO$_WRITEVBLK, &iosb, 0, 0,
                               &buffer.name, inq_name.dsc$w_length, 0, 0, 0, 0 );
    if ( return_status & STS$M_SUCCESS )
        return_status = iosb.status;
    return return_status;
}
rcv_and_disp_response ()

/* This module waits on a read over the logical link to DB_SERVER for a    *
 * response.  After receiving the response, it formats the buffer and      *
 * displays the contents at the user's terminal.  If the server encounters *
 * any errors on the request, the status of the buffer reflects the        *
 * condition.                                                              */
```

Example 2–1 Cont'd. on next page

Example 2–1 (Cont.) Example of a Nontransparent Communication Program

```
{
    return_status = sys$qiow ( 0, net_chan, IO$_READVBLK, &iosb, 0, 0,
                               &buffer, buff_len, 0, 0, 0, 0 );
    if ( return_status & STS$M_SUCCESS ) {
        return_status = iosb.status;
        if ( return_status & STS$M_SUCCESS ) {
            if ( buffer.status & STS$M_SUCCESS ) {
                disp_desc.dsc$w_length = MAX_DISPLAY;
                return_status = sys$fao ( &fao_ctrl, &disp_desc.dsc$w_length,
                        &disp_desc, MAX_NAME, &buffer.name, MAX_ACCOUNT,
                        &buffer.account, MAX_PHONE, &buffer.phone, MAX_ADDRESS,
                        &buffer.address, MAX_LOCATION, &buffer.location );
                if ( return_status & STS$M_SUCCESS )
                    return_status = lib$put_output ( &disp_desc );
                }
            else {
                msgvec.msg_code = buffer.status;
                return_status = sys$putmsg ( &msgvec, 0, 0, 0 );
    }   }   }
    return return_status;
}
/*      .TITLE  DB_SERVER - Database server program                        *
 *      .IDENT  'X-1'                                                      */
/* ABSTRACT:    DB_SERVER                                                  */

/* This program demonstrates how to declare a known network object and     *
 * service and manage multiple connect requests.  It is a simple           *
 * implementation of task-to-task communication using DECnet               *
 * nontransparently.  DB_SERVER accesses the database USER_DB, based on    *
 * the name key supplied in the request buffer, and sends a response to    *
 * the requesting node.                                                    */

    struct {
        char func;
        int terminator;
    } nfb = { NFB$C_DECLNAME, 0 };

    static $DESCRIPTOR ( net_device, "_NET:" );
    static $DESCRIPTOR ( object_name, "DB_SERVER" );

main ()

/* After initialization, DB_SERVER processes requests until a NETSHUT      *
 * message is received in the network command mailbox.  Asynchronous       *
 * processing, which permits several logical links to be serviced          *
 * concurrently, is achieved through the use of asynchronous system traps  *
 * (ASTs).                                                                 */
```

Example 2–1 Cont'd. on next page

Example 2–1 (Cont.) Example of a Nontransparent Communication Program

```
{
    return_status = initialization () ;
    while ( return_status & STS$M_SUCCESS && !net_shut ) {
        return_status = remque_buffer ( LIVE_QUE, &cur_buff ) ;
        if ( return_status & STS$M_SUCCESS ) {
            switch ( cur_buff->cmplt_stat.ast.type ) {
                case NET_RD  : return_status = process_request ();  break;
                case NET_WRT : return_status = process_response (); break;
                case NET_CMD : return_status = process_netcmd ();   break;
                default      : return_status = SS$_BADPARAM;        break;
        }   }
        else if  ( return_status == LIB$_QUEWASEMP )
            return_status = sys$hiber ();
    }
    exit ( return_status );
}

initialization ()

/*  Set up descriptors, initialize variables, queues, and buffers, declare  *
/*  the process as a network receiver, open the database, and issue a read  *
/*  to the network command mailbox.                                         */
{
    return_status = initialize_variables ();
    if ( return_status & STS$M_SUCCESS ) {
        return_status = declare_network_object ();
        if ( return_status & STS$M_SUCCESS ) {
            return_status = open_database ();
            if ( return_status & STS$M_SUCCESS )
                return_status = issue_netcmd_read ();
    }   }
    return return_status;
}
initialize_variables ()  /*  Insert buffers into FREE_QUE for later use.  */
{
}

declare_network_object ()

/* Declare the DB_SERVER process as a known network object to allow the    *
 * reception of multiple inbound connect requests.  The three steps are    *
 *   1.  Create a temporary mailbox to receive network command messages.   *
 *   2.  Assign a channel to the _NET device and associate the temporary   *
 *       mailbox with it.                                                  *
 *   3.  Issue a QIO with a function code of IO$_ACPCONTROL, the P1        *
 *       parameter set to the address of the network function block (NFB), *
 *       and the P2 parameter set to the address of the descriptor         *
 *       containing the object name to be declared.                        */
```

Example 2–1 Cont'd. on next page

Example 2–1 (Cont.) Example of a Nontransparent Communication Program

```
{
    return_status = sys$crembx ( TEMP_MBX, &mbx_chan, MAX_MSG, BUF_QUO,
                                 0, 0, &netcmd_mbx );
    if ( return_status & STS$M_SUCCESS ) {
        return_status = sys$assign ( &net_device, &netdcl_chan, 0,
                                     &netcmd_mbx );
        if ( return_status & STS$M_SUCCESS ) {
            return_status = sys$qiow (  0, netdcl_chan, IO$_ACPCONTROL, &iosb,
                                        0, 0, &nfb_desc, &object_name, 0, 0, 0, 0 );
            if ( return_status & STS$M_SUCCESS )
                return_status = iosb.status;
    }   }
    return return_status;
}

open_database ()
/*       Initialize the FAB and RAB before opening and connecting a stream. */
{
    return_status = sys$open ( &db_fab );
    if ( return_status & STS$M_SUCCESS )
        return_status = sys$connect ( &db_rab );
    return  return_status;
}

process_request ()
/*  A database query request has been received.  Read the database and   *
 *  issue a write.                                                       */
{
    index = cur_buff->cmplt_stat.ast.ndx;
    if ( cur_buff->iosb.status & STS$M_SUCCESS ) {
        return_status = read_database ();
        if ( return_status & STS$M_SUCCESS )
            return_status = issue_link_write ( index );
    }
    else {

        /*  The I/O completed with a failure.  Check for network failure   *
         *  and await the DECnet mailbox message if it was a network       *
         *  failure.                                                       */

        return_status = check_fatal_io ( cur_buff->iosb.status );
        if ( return_status & STS$M_SUCCESS )
            return_status = insque_buffer ( FREE_QUE, cur_buff );
    }
    return return_status;
}

process_response ()
/*  The response to a user's request has completed.  Issue another read for *
 *  the next request.                                                        */
{
    index = cur_buff->cmplt_stat.ast.ndx;
    if ( cur_buff->iosb.status & STS$M_SUCCESS )
        return_status = issue_link_read ( index );
    else {

        /*  The I/O completed with a failure.  Check for network failure   *
         *  and await the DECnet mailbox message if it was a network       *
         *  failure.                                                       */
```

Example 2–1 Cont'd. on next page

Example 2–1 (Cont.) Example of a Nontransparent Communication Program

```
        return_status = check_fatal_io ( cur_buff->iosb.status );
        if ( return_status & STS$M_SUCCESS )
            return_status = insque_buffer ( FREE_QUE, cur_buff );
    }
    return return_status;
}

process_netcmd ()

/* A network command message was received in the network command mailbox.  *
 * Each request is dispatched into one of four groups:                     *
 *   link_failure    A failure has occurred on the link.                   *
 *   establish_link  A connect initiate message was received.              *
 *   not_used        A message that is not used in this program was        *
 *                   received.                                             *
 *   shutdown        A NETSHUT message was received, indicating that DECnet*
 *                   is shutting down and the program should terminate.    */
{
    if ( cur_buff->iosb.status & STS$M_SUCCESS ) {
        unit = cur_buff->msg_area.mbx_msg.unit;
        switch ( cur_buff->msg_area.mbx_msg.msg ) {
            case MSG$_ABORT      : return_status = link_failure(unit);   break;
            case MSG$_CONFIRM    : return_status = not_used();           break;
            case MSG$_CONNECT    : return_status = establish_link();     break;
            case MSG$_DISCON     : return_status = link_failure(unit);   break;
            case MSG$_EXIT       : return_status = link_failure(unit);   break;
            case MSG$_INTMSG     : return_status = not_used();           break;
            case MSG$_PATHLOST   : return_status = link_failure(unit);   break;
            case MSG$_PROTOCOL   : return_status = link_failure(unit);   break;
            case MSG$_REJECT     : return_status = not_used();           break;
            case MSG$_THIRDPARTY : return_status = link_failure(unit);   break;
            case MSG$_TIMEOUT    : return_status = link_failure(unit);   break;
            case MSG$_NETSHUT    : return_status = shutdown ();          break;
            default              : return_status = SS$_BADPARAM;         break;
    }   }
    else return_status = cur_buff->iosb.status;

    return return_status;
}

link_failure (unit)

/*  Cleanup of a logical link involves these steps:                        *
 *    1.  Find the link in the link control table (LCT) based on the unit. *
 *    2.  Deassign the channel.  Otherwise, the link remains in a closed   *
 *        state and the _NET device remains allocated.                     *
 *    3.  Release the lct entry.                                           *
 *    4.  Insert the buffer into the FREE_QUE queue for later use.         */
```

Example 2–1 Cont'd. on next page

Example 2–1 (Cont.) Example of a Nontransparent Communication Program

```
{
    return_status = find_lct ( unit, &index );
    if ( return_status & STS$M_SUCCESS ) {
        return_status = sys$dassgn ( lct[index].channel );
        if ( return_status & STS$M_SUCCESS ) {
            return_status = release_lct ( index );
            return_status = insque_buffer ( FREE_QUE, cur_buff );
    }   }
    else {
        return_status = SS$_NORMAL;
        return_status = insque_buffer ( FREE_QUE, cur_buff );
    }
    return return_status;
}

not_used ()  /*  The network command message is not used in this program.  */
{
    return insque_buffer ( FREE_QUE, cur_buff );
}

establish_link ()

/* To establish a logical link, the incoming request must be accepted.     *
 * Five steps are required to accept a request:                            *
 *   1.  Copy the information portion of the NCB into a buffer.            *
 *   2.  Allocate a link control table entry.                              *
 *   3.  Assign a channel to the _NET device and associate the network     *
 *       command mailbox with the channel.                                 *
 *   4.  Get the unit number of the _NET channel to identify the network   *
 *       command messages.                                                 *
 *   5.  Issue to the _NET channel a QIO with the IO$_ACCESS function code *
 *       and the P2 parameter set to the address of the copied NCB         *
 *       information.                                                      */
{
    i = cur_buff->msg_area.mbx_msg.name_info[0] + 1;
    info_cnt = cur_buff->msg_area.mbx_msg.name_info[i];
    start = i + 1;
    for ( i = 0; i < info_cnt; i++ )
        ncb[i] = cur_buff->msg_area.mbx_msg.name_info[start+i];
    ncb_desc.dsc$w_length = info_cnt;
```

Example 2–1 Cont'd. on next page

Example 2–1 (Cont.) Example of a Nontransparent Communication Program

```
    return_status = allocate_lct ( &index );
    if ( return_status & STS$M_SUCCESS ){
        return_status = sys$assign ( &net_device, &lct[index].channel,
                                     0, &netcmd_mbx );
        if ( return_status & STS$M_SUCCESS ) {
            getdvi_itm.ret_info = &unit;
            return_status = sys$getdviw ( 0, lct[index].channel, 0,
                                          &getdvi_itm, &iosb, 0, 0, 0 );
            if ( return_status & STS$M_SUCCESS ) {
                return_status = iosb.status;
                if ( return_status & STS$M_SUCCESS ) {
                    lct[index].unit = unit;
                    return_status = sys$qiow ( 0, lct[index].channel,
                         IO$_ACCESS, &iosb, 0, 0, 0, &ncb_desc, 0, 0, 0, 0 );
                    if ( return_status & STS$M_SUCCESS ) {
                        if ( iosb.status & STS$M_SUCCESS ) {
                            lct[index].cur_buff = cur_buff;
                            return_status = issue_link_read ( index );
                        }
                        else {
                            return_status = check_fatal_accept ( iosb.status );
                            if ( return_status & STS$M_SUCCESS )
                                return_status = link_failure (lct[index].unit);
    }   }   }   }   }   }
    else {
        func = IO$_ACCESS | IO$M_ABORT;
        return_status = sys$qiow ( 0, netdcl_chan, func,  &iosb, 0, 0, 0,
                                   &ncb_desc, 0, 0, 0, 0 );
        if ( return_status & STS$M_SUCCESS )
            return_status = insque_buffer ( FREE_QUE, cur_buff );
    }
    return return_status;
}

shutdown ()

/*  A NETSHUT message was received in the network command mailbox.  Set a  *
 *  flag to drop through the main processing loop and terminate.            */
{
    net_shut = 1;
    return insque_buffer ( FREE_QUE, cur_buff );
}

issue_netcmd_read ()

/*  Issue a read on the network command mailbox.  The ASTID will inform     *
 *  the AST routine which buffer completed.                                 */
{
    netcmd_buff->cmplt_stat.astid=NET_CMD;
    return  sys$qio ( 0, mbx_chan, IO$_READVBLK, &netcmd_buff->iosb,
                      &ast_routine, NET_CMD, &netcmd_buff->msg_area.mbx_msg,
                      mbx_msg_len, 0, 0, 0, 0 ) ;
}
```

Example 2–1 Cont'd. on next page

Example 2–1 (Cont.) Example of a Nontransparent Communication Program

```
issue_link_read ( index )

/*  Issue an asynchronous read on a logical link.  The ASTID will inform   *
 *  the AST routine which buffer completed.                                 */
{
    cur_buff->cmplt_stat.ast.ndx = index;
    cur_buff->cmplt_stat.ast.type = NET_RD;
    return sys$qio ( 0, lct[index].channel, IO$_READVBLK, &cur_buff->iosb,
                     &ast_routine, cur_buff->cmplt_stat.astid,
                     &cur_buff->msg_area.data,
                     sizeof(cur_buff->msg_area.data), 0, 0, 0, 0 );
}

issue_link_write ( index )

/*  Issue an asynchronous write on a logical link.  The ASTID will inform  *
 *  the AST routine which buffer completed.                                 */
{
    cur_buff->cmplt_stat.ast.ndx = index;
    cur_buff->cmplt_stat.ast.type = NET_WRT;
    return sys$qio ( 0, lct[index].channel, IO$_WRITEVBLK, &cur_buff->iosb,
                     &ast_routine, cur_buff->cmplt_stat.astid,
                     &cur_buff->msg_area.data,
                     sizeof (cur_buff->msg_area.data), 0, 0, 0, 0 );
}

read_database ()

/*  Read the database using the name field of the request buffer.  The     *
 *  status of the read request is placed in the status field of the buffer *
 *  for return to the requester.                                            */
{
    db_rab.rab$l_kbf = &cur_buff->msg_area.data.name;
    db_rab.rab$l_ubf = &cur_buff->msg_area.data;
    return_status = sys$get ( &db_rab );
    cur_buff->msg_area.data.status = return_status;
    if ( return_status == RMS$_RNF )
        return_status = SS$_NORMAL;
    return return_status;
}

find_lct ( unit, pindex )
/*    Find a link control table entry based on the unit.  */
{
    return_status = SS$_NORMAL;
    found = 0;
    for ( i = 0; i < MAX_LINKS && !found; i++ ) {
        if ( lct[i].unit == unit )
            found = 1;
    }
    if ( !found )
        return_status = LIB$_NOTFOU;
    else
        *pindex = i-1;

    return return_status;
}
```

Example 2–1 Cont'd. on next page

Example 2–1 (Cont.) Example of a Nontransparent Communication Program

```
release_lct ( index )

/*  Release a link control table (LCT) entry.  Clear the LCT entry and    *
 *  clear the corresponding bit in the allocation mask.  The mask is used *
 *  to scan the entire table in one instruction.                          */
{
    lct_alloc_mask ^= ( 1 << index );
    lct[index].unit = 0;
    lct[index].channel = 0;
    lct[index].cur_buff = 0;
    return SS$_NORMAL;
}

allocate_lct ( pindex )

/*  Allocate the first available link control table (LCT) entry.  Then    *
 *  set the corresponding bit in the allocation mask.                     */
{
    return_status =  lib$ffc ( &start_pos, &mask_siz, &lct_alloc_mask, pindex);
    if ( return_status & STS$M_SUCCESS )
        lct_alloc_mask |= ( 1 << *pindex );
    return return_status;
}

remque_buffer ( queue, cur_buff )

/*  Insert the buffer into the specified queue, FREE_QUE or LIVE_QUE.     *
 *  Relative queues are used because of the interlock supplied in the RTL *
 *  routine.  Virtual address queues could be used, but AST disabling     *
 *  would be necessary to eliminate queue skews.                          */
{
    return lib$remqhi ( &que_hdr[queue].flink, cur_buff, &rtry );
}

insque_buffer ( queue, cur_buff )
/*  Insert a buffer into the specified queue, FREE_QUE or LIVE_QUE. */
{
    return lib$insqti ( cur_buff, &que_hdr[queue].flink, &rtry );
}

check_fatal_io ( io_stat )
{
     if ( io_stat != SS$_BUFFEROVF && io_stat != SS$_FILNOTACC &&
          io_stat != SS$_INSFMEM)
         io_stat = SS$_NORMAL;
     return io_stat;
}

check_fatal_accept ( io_stat )
{
    if ( io_stat != SS$_DEVALLOC && io_stat != SS$_INSFMEM &&
         io_stat != SS$_IVDEVNAM )
        io_stat = SS$_NORMAL;

    return io_stat;
}
```

Example 2–1 Cont'd. on next page

Example 2–1 (Cont.) Example of a Nontransparent Communication Program

```
ast_routine ( ast_param )

/* AST routine.  This module handles all asynchronous QIOs issued by the   *
 * main process.  The ast_parameter passed to this module governs the       *
 * processing.  The ast_parameter is actually split into three parts:       *
 *   ast_type   The type of QIO completion (NET_RD, NET_WRT, NET_CMD)       *
 *   index      The corresponding link control table entry (only valid for *
 *              NET_RD and NET_WRT)                                         *
 *   unused     Free space for later use                                    *
 * Buffers are inserted into the LIVE_QUE queue for processing in the main *
 * module.  NET_RD AND NET_WRT QIOs work in pairs serially and are          *
 * reissued in the main processing module.  NET_CMD QIOs are reissued       *
 * immediately to accommodate multiple network requests.                    */
{
    switch ( ast_param.type ) {
        case NET_RD  : return_status = insque_buffer ( LIVE_QUE,
                                       lct[ast_param.ndx].cur_buff );   break;
        case NET_WRT : return_status = insque_buffer ( LIVE_QUE,
                                       lct[ast_param.ndx].cur_buff );   break;
        case NET_CMD : return_status = que_and_reissue ();              break;
        default      : return_status = SS$_BADPARAM;                    break;
    }

    if (return_status & STS$M_SUCCESS)  {
        return_status = sys$wake ( 0, 0 );
        if ( return_status != SS$_NORMAL )
            exit ( return_status );
    }
    else
        exit ( return_status );
}

que_and_reissue ()
{
    return_status = insque_buffer ( LIVE_QUE, netcmd_buff );
    if ( return_status & STS$M_SUCCESS ) {
        return_status = remque_buffer ( FREE_QUE, &netcmd_buff );
        if ( return_status & STS$M_SUCCESS )
            return_status = issue_netcmd_read ();
    }

    return return_status;
}
```

2.3 System and Network Manager Responsibilities

The system manager of a DECnet–VAX node is responsible for establishing the system as a node in the network, and controlling and monitoring the node. Configuring your system as a network node requires supplying information at the local node about network components, including the characteristics of the local node, remote nodes, circuits, lines, and objects. This information constitutes what is called the *configuration database* for the local node. Each node in the network has such a database. As manager of your system, you supply information about the configuration database using the Network Control Program (NCP) Utility.

2.3 System and Network Manager Responsibilities

If you are either configuring a DECnet–VAX node for the first time or rebuilding the configuration database for your local node, you can use the interactive NETCONFIG.COM procedure to configure your node automatically. Once you bring up your DECnet–VAX node and verify its connection to the network, you can use the NCP Utility to control and monitor local network operation, and to test network software operation. See Chapter 3 for the procedure for bringing up your node on the network, and Chapter 4 for the techniques for maintaining the network.

Planning for configuration of your node in an existing network usually involves coordinating with the system managers of other nodes in the network or with the manager of the network (if a manager has been designated) to ensure uniform parameter settings.

To create a new network, two or more system managers should connect their systems by means of communications lines; each manager should then bring up his or her system as a network node.

A system manager of a network node may be called upon to provide DECnet–VAX host services for other DECnet nodes. Host services include loading system images and programs *downline* to unattended remote nodes, and receiving for interpretation *upline* dumps of system images from nodes that have crashed. For example, DECnet–VAX permits you to load an operating system file image or a terminal server image downline to a target node. Another DECnet–VAX host service involves connecting to an unattended remote node (for example, a diskless communications server) to act as its console.

For a larger network, one person, who may be the manager of a network node, is usually designated as the manager of the network. The network manager is responsible for planning, building and fine tuning a whole network to run with maximum efficiency. The network manager makes networkwide configuration decisions, such as the kinds of paths to be established, which nodes should be routers or end nodes, and whether the network should be divided into areas. The network manager also sets values for network parameters that should be the same across the network.

Managing a large network usually involves regular monitoring to detect patterns of usage and error conditions on the network, and performing remote configuration of the network to control traffic patterns and accommodate network growth. System and network managers also perform maintenance procedures, to prevent serious problems from developing, and carry out troubleshooting procedures, to resolve problems quickly. Using network software, the manager can obtain statistics on network usage and routing parameters. Network logging files provide error statistics useful in diagnosing potential problems. NCP commands display the status of nodes, lines and circuits in the network.

Some of the considerations involved in developing a network are summarized at the end of Chapter 3. Maintenance and troubleshooting procedures are summarized in Chapter 4. For a complete description of managing and maintaining VMS systems in a DECnet network, refer to the *VMS Networking Manual*. NCP commands are specified in the *VMS Network Control Program Manual*.

3 Getting Started on the Network

How you gain access to the DECnet network is related to the role you are performing on the VMS system. Depending on your role, you can:

- **Log in to an existing network node:** If you are a general user with an account on a VMS system that is connected to an existing network, you have access to the network as soon as you log in to your account, provided you have the privileges NETMBX and TMPMBX. (See the description in Section 3.1.)
- **Bring up your VMS system as a network node:** If you are the manager of a VMS system, you can physically connect your system to an existing DECnet network by means of a communications line, and bring your system up as a network node by performing the DECnet–VAX installation procedure. The DECnet–VAX installation procedure you perform on your system involves registering the DECnet–VAX key using the VMS License Management Utility, configuring your node as part of the network, starting the network, and verifying that you are connected to the network. (Section 3.2 and Section 3.3 describe the steps involved.)
- **Create a new network:** If there is no existing network to which you can connect, you can cooperate with the managers of other systems to create a new network. A network is formed when two or more systems are connected by communications lines and each system is brought up as a network node. For larger networks, the system manager of one node may also manage the network. (Section 3.4 indicates some considerations involved in establishing a large network.)

3.1 Accessing an Existing VMS Network Node

A VMS operating system on a VAX computer connected to an existing DECnet network is called a DECnet–VAX node (or VMS node). Once your VMS node has been brought up on the network, you can access the network simply by logging in to the node. The node at which you log in is called the local node; other nodes on the network are called remote nodes.

The following section describes how to log in to a VMS node and how to determine the way your node relates to the network. The section also discusses how you can log in to another node on the network from your local node.

3.1.1 Logging In to a Network Node

If you are a VMS user with an account on a system that is a node on a DECnet network, you can gain access to the network by logging in to your account on the node, provided you have the privileges required by the network. The minimum privileges required to access the network for general user operations are TMPMBX and NETMBX. To verify that you have these privileges, enter the DCL command SHOW PROCESS/PRIVILEGES. For example, enter the following:

```
$ SHOW PROCESS/PRIVILEGES

1-JUN-1988 09:08:24.08 TTA8:  User: JONES

Process privileges:

TMPMBX              may create temporary mailbox
NETMBX              may create network device
```

If necessary, ask the system manager to provide you with the required privileges.

To log in to the VMS system, follow the standard VMS login procedure. Turn on the terminal and press the RETURN key, or connect to the VMS system through a terminal server. Respond to the prompts for user name and password. The DCL prompt ($, by default) should appear, indicating that you are logged in to the system at DCL command level.

You can now perform all general user operations on the network, including sending and receiving mail and accessing remote files. Network user operations are described in Chapter 2.

To display the name of your local node, use the DCL command SHOW LOGICAL SYS$NODE. For example, to display the name of the local node ORANGE, enter the following:

```
$ SHOW LOGICAL SYS$NODE

"SYS$NODE" = "ORANGE::" (LNM$SYSTEM_TABLE)
```

To learn how your node relates to the rest of the network, you can enter the DCL command SHOW NETWORK. The command display includes the address and name of your local node. If your node is a routing node, the SHOW NETWORK display lists the other nodes (in your area) to which your node has access and provides routing information about the path to each node. In a multiple-area network, the display also indicates the path to the area router through which your node has access to nodes in other areas. (See Chapter 1 for a discussion of routing over the network.)

The following example is the network display for routing node ORANGE connected to Ethernet circuit UNA-0.

```
$ SHOW NETWORK

VAX/VMS Network status for local node 2.5 ORANGE on 1-JUN-1988 10:10:10

The next hop to the nearest area router is node 2.2 VIOLET:

            Node          Links   Cost  Hops  Next Hop to Node

            2.5 ORANGE      0       0     0   Local ->    2.5 ORANGE
            2.2 VIOLET      1       1     1   UNA-0 ->    2.2 VIOLET
            2.3 PURPLE      0       1     1   UNA-0 ->    2.3 PURPLE
            2.4 YELLOW      0       1     1   UNA-0 ->    2.4 YELLOW
```

If your node is an end node, the SHOW NETWORK display identifies your node by name and address, but does not provide a list of accessible nodes. If your end node is connected to a multiaccess Ethernet circuit, however, the display identifies a particular router designated to perform routing services for the end node. The following example shows the display for end node YELLOW on an Ethernet.

```
$ SHOW NETWORK

VAX/VMS Network status for local node 2.4 YELLOW on 1-JUN-1988 10:15:00

This is a nonrouting node, and does not have any network information.
The designated router for YELLOW is node 2.5 ORANGE.
```

If the network is not available at the time you enter the SHOW NETWORK command (for example, if DECnet–VAX is temporarily turned off), the message "Network unavailable" is displayed.

You can obtain additional information about the network by means of the Network Control Program (NCP) Utility. The NCP commands you can use to monitor the network are described in Chapter 4.

3.1.2 Accessing a Remote Node Interactively

You can also log in to other nodes on the network from your local system. DECnet–VAX provides a network command terminal facility that permits you to log in to a remote node on which you have an account, and use the facilities of that system while you are physically connected to the local system. (The network command terminal facility is also referred to as the network virtual terminal facility.)

To access a remote node, enter the DCL command SET HOST and specify the remote node name or address. If the network link cannot be established, you will receive an error message. Otherwise, you can log in using the login procedure required by the remote node (which need not be a VMS node).

For example, user JONES on node ORANGE can access VMS system YELLOW on which he has an account, as follows:

```
$ SET HOST YELLOW
USERNAME: JONES
PASSWORD:
            Welcome to VAX/VMS Version 5.0 on node YELLOW
$
```

If you attempt to gain access to another VMS node using invalid access information, the host system responds with the message "User authorization failure." If you want to try to access the node again, press RETURN and you will again be prompted for a user name and password. After a number of unsuccessful attempts, the link will be disconnected automatically. If you want to abort the procedure, enter CTRL/Z at the user name or password prompt (or enter CTRL/Y twice, as described below).

Once you are logged in to a remote node using the SET HOST command, you can perform any operation on the remote node as though you were logged in to that node locally.

3.1 Accessing an Existing VMS Network Node

You can terminate the remote session in two ways:

- Using the remote system's logout procedure. (On a VMS system, enter the command LOGOUT.)
- By pressing CTRL/Y twice. The remote system responds by asking "Are you repeating ^Y to abort the remote session?" Answering Y or y aborts the remote session.

The message "%REM-S-END, control returned to node _nodename::" is displayed and you are returned to the local node.

If DECnet cannot maintain a connection to the remote node, the remote session is terminated, the message "Path lost to partner" may be displayed, and you are returned to the local node. The path may have been lost because the other node (or an intermediate node on the path to that node) went down temporarily, or a transient network error occurred. If the SHOW NETWORK display indicates that the remote node is still reachable, you can try to log in to that node again. (For a discussion of network messages, see Chapter 4.)

3.2 Preparing to Bring Up Your System as a Node on an Existing Network

If you are the system manager of a VMS system, you can install DECnet–VAX and configure your system as a node on an existing network. You can be connected permanently to the network, or you can optionally choose to establish a temporary connection to the network over a telephone line. A temporary DECnet connection exists only for the duration of the telephone call.

Before you begin the procedure for installing DECnet–VAX on your system, you should check your hardware and connect any required communications lines. You should also prepare your VMS operating system for the network environment and make some basic decisions about how you want to configure your node. These preparations are discussed in the following subsections.

3.2.1 Connecting the Communications Hardware on Your System

A network is a flexible configuration of computers and terminals interconnected by communications lines. You should identify the equipment you need to connect your VAX computer to an existing network. For each connection, this equipment normally includes

- A communications controller device (line device) that contains one or more interface points called ports. (The line device is installed on your processor.)
- A communications line to connect the port to the network.

Consult your DIGITAL sales support representative if you are not familiar with the equipment that you require, or if you need to install such equipment. Following the instructions in the hardware user manuals included with the equipment, you should be able to connect each network communications line to the appropriate port.

3.2 Preparing to Bring Up Your System as a Node on an Existing Network

A VAX computer on which a VMS operating system is running can be connected to the network by means of high-speed lines (such as an Ethernet cable or a synchronous point-to-point line). A VMS system can also be connected to a network by means of a low-speed, low-cost asynchronous line. An asynchronous point-to-point connection can be established over any VMS terminal line between a VMS system and another system (which can be a non-VMS system) that supports the DECnet asynchronous DIGITAL Data Communications Message Protocol (DDCMP). An asynchronous connection can optionally be made over a dialup line (for example, a telephone line) if a modem is used at each end of the connection. A modem is a device that connects the terminal line to the telephone line. Modems may be purchased separately from DIGITAL, along with the appropriate installation documentation.

A VAX processor can have a number of communications ports, depending on the model. For example, the VAXstation II illustrated in Figure 3–1 has four asynchronous terminal ports and one Ethernet port. You can connect one of the terminal ports to a modem attached to a telephone line to be used as an asynchronous DECnet dialup line. You can also connect the Ethernet port by means of a transceiver cable to an H4000 transceiver that is attached to an Ethernet coaxial cable. The possible connections are limited only by the number of devices that your processor can support, as specified in the DECnet–VAX Software Product Description load unit tables, and the devices that you configure for your node. Note that any node with two or more network connections must be a router.

A typical VAXstation 2000 configuration is shown in Figure 3–2. The processor box has an Ethernet port in the upper right corner to which you can connect a ThinWire Ethernet T–connector. Be sure a ThinWire Ethernet cable connection for the VAXstation 2000 is available in your office and that the T–connector is properly terminated.

An example of a large VAX system, a VAX 8800 processor, is shown in Figure 3–3. The system has three network connections:

- Connection to an Ethernet cable by means of an H4000 transceiver
- Connection to a MicroVAX II by means of a static asynchronous DDCMP line
- Connection to a VAX–11/780 at a remote location by means of a synchronous DDCMP line

Figure 3–1 Example of a Communications Hookup for a VAXstation II

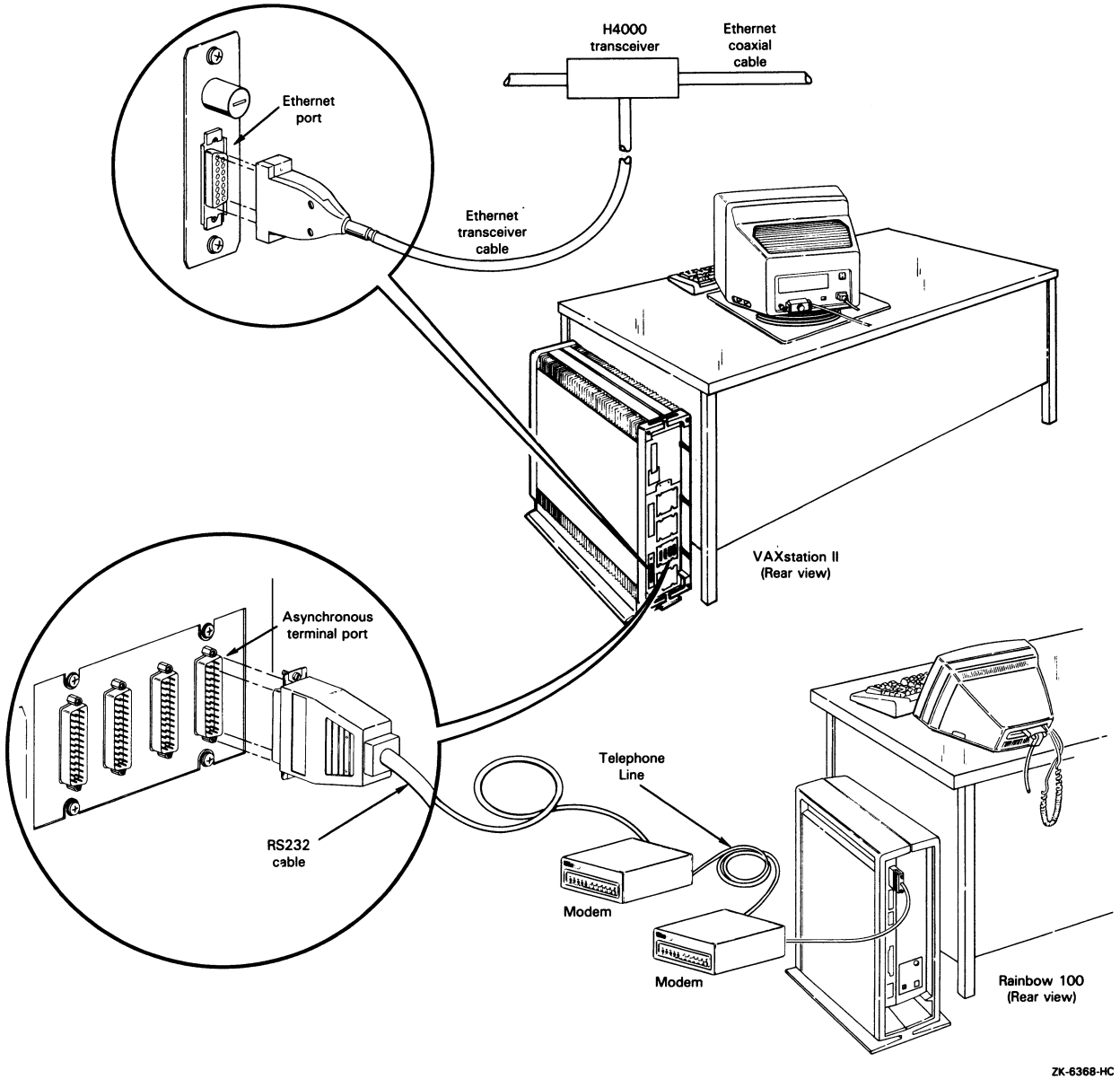


Figure 3–2 Example of a Communications Hookup for a VAXstation 2000

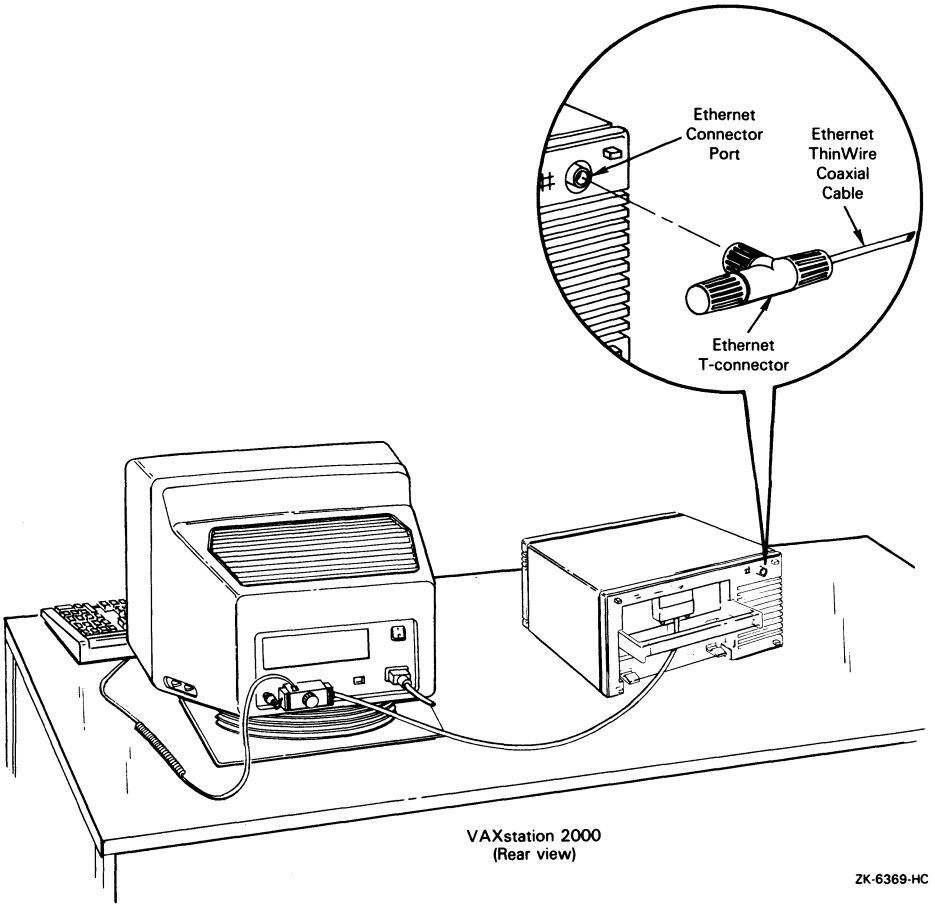
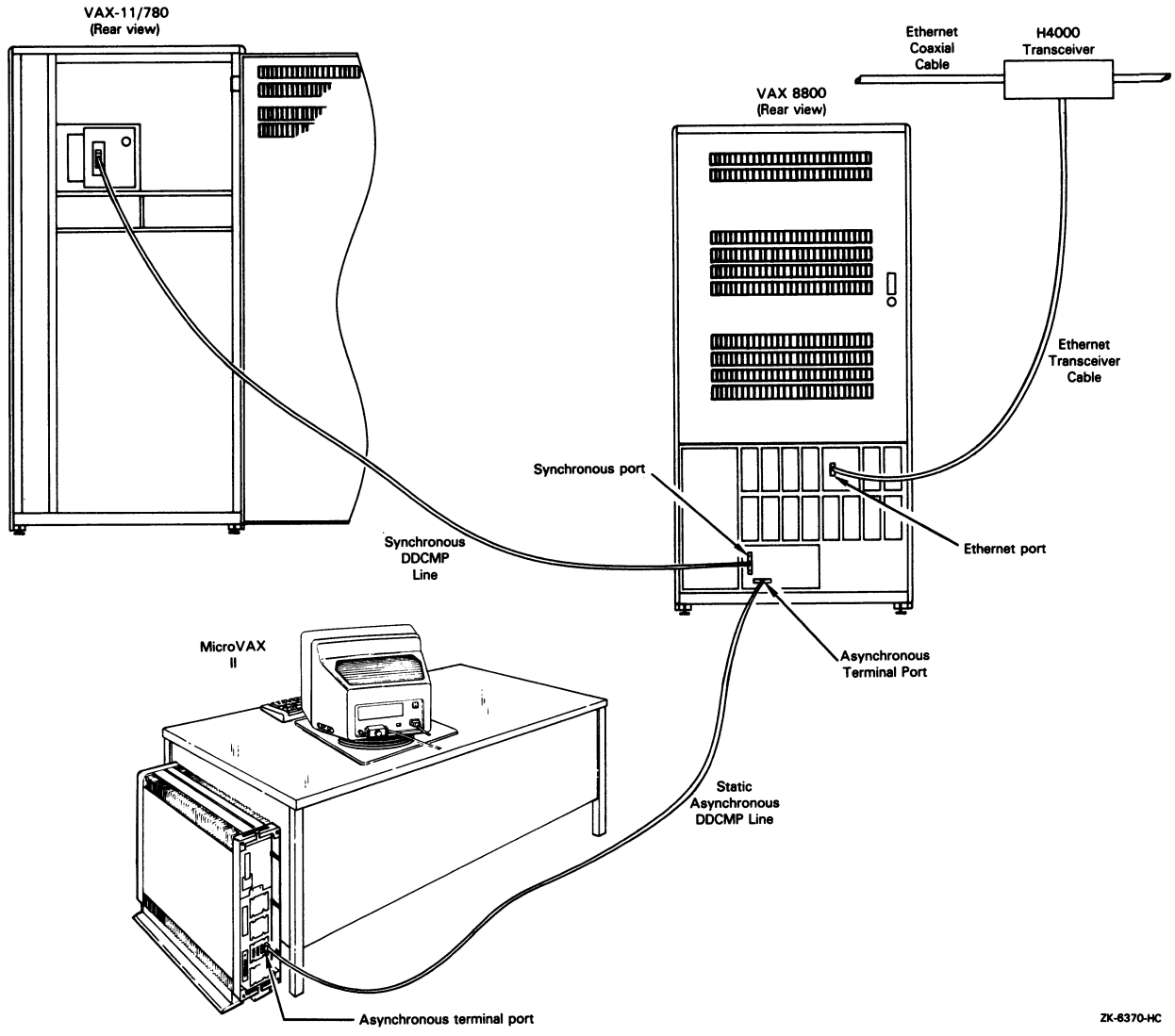


Figure 3–3 Example of a Communications Hookup for a Large VAX Computer



3.2.2 Preparing Your VMS System for the Network Environment

Before you bring up DECnet–VAX on your system, you should take the following steps to prepare your system to function as part of the network:

- Check to see if you have the privileges you need to perform network operations. The minimum privileges that a system manager normally requires to configure and control the network and run network programs are SYSPRV, OPER, TMPMBX, NETMBX and BYPASS. A list of privileges required for network operations appears in Table 3–1.

  Specify the DCL command SHOW PROCESS/PRIVILEGES to determine which of your authorized privileges are currently enabled, and the command SET PROCESS/PRIVILEGES to enable any additional privileges you are authorized to have that are required for network operations.

- Decide whether you wish to establish a default nonprivileged DECnet account and directory. The nonprivileged account is a default DECnet account that DECnet–VAX uses to allow access if the user on a remote network node does not explicitly supply access control information (the user name and password) when requesting a connection to the local node, or if there is no proxy account to use. Note that the default DECnet account is required to use such VMS utilities as MAIL or PHONE over the network. For more information on the default DECnet account, see Section 3.3.7.

  If you want the default account, you can request that the DECnet–VAX configuration procedure, NETCONFIG.COM, establish a default nonprivileged account and directory for your node automatically. (Section 3.3.2.2 describes NETCONFIG.COM.) As an alternative, you can establish a nonprivileged account and directory manually.

- Set up any proxy accounts that you want to establish for your node. A proxy login account allows a user on a remote node on the network to access data by way of a local account on your system. For more information on setting up and using proxy accounts, see Section 3.3.7.1.

- If necessary, tune your VMS system to accommodate DECnet–VAX software. The network manager who establishes network configuration guidelines should provide you with any required information if you need to update VMS system parameters and quotas. See Section 3.4 for a summary description of considerations involved in establishing a network.

Table 3–1 VMS Privileges Required for DECnet–VAX Operations

Privilege	Network Operations
ACNT	Required to start the network; permits you to suppress accounting messages
BYPASS	Permits you to view passwords in the DECnet–VAX databases
CMKRNL	Required to start the network; permits a process to access the VMS kernel
DETACH	Required to start the network; allows you to create detached processes
NETMBX	Required for all network users; needed for any network operation; needed to create a logical link
OPER	Allows you to perform operator functions such as modifying the DECnet–VAX volatile database
TMPMBX	Required for all network users and default DECnet accounts; needed to run NCP and to create a temporary mailbox
SYSNAM	Permits you to declare a name or object number in a user task
SYSPRV	Required to access the DECnet–VAX permanent database

3.2.3 Planning the Configuration of Your DECnet–VAX Node

Before you specify how your node is to be configured as part of an existing network, you should make the following decisions:

- Select a unique node name and node address for your system. If a network manager has been designated for your network, request a node name and address from the network manager. If your node is a member of a VAXcluster, obtain your node name and address from the VAXcluster manager. (The VAXcluster node name must be set in the VMS system parameter SCSNODE and the node address in SCSSYSTEMID. See the *VMS System Generation Utility Manual.*

  Each node in the network is identified by a specific name and a numeric address by which the node is known to other nodes in the network. The node name can be no more than six alphanumeric characters (including at least one alphabetic character). The node address consists of an area number (in the range from 1 to 63, with a default value of 1) and a node number (in the range from 1 to 1023) separated by a period (for example, 2.2).

  If your node is a member of a VAXcluster that uses an alias node identifier (an alias name or address), you can obtain the alias identifier from the VAXcluster manager. An alias node identifier, common to some or all nodes in a cluster, permits remote nodes to treat the cluster as though it were a single node. Individual nodes in a VAXcluster can optionally assume the alias, while retaining their individual node names. You can use the alias adopted by the cluster, as well as your own node name, to communicate with other nodes in the network.

- Determine the node names and addresses of all other nodes in your network to which you wish to connect. To obtain the correct node name and address of each node, you can contact the network manager or, if necessary, the individual system managers of the other nodes. You must enter this remote node information in your network node database. Alternatively, you can determine whether the names and addresses of the nodes that you wish to define are already defined in the network database of another node. If you have the appropriate privileges, you can copy the node database information from a remote node into your node database. (The procedure is described in Section 3.3.2.2.)
- Decide whether your system is to be a router or an end node. If you have a DECnet full function license and the accompanying DECnet–VAX key, you have the option of configuring your system as either a router or an end node. If your DECnet license and key are for the end node capability, you can only configure your system as an end node.
- Determine the types of connections that will be made to the network: Ethernet, synchronous DDCMP, asynchronous DDCMP, or CI connections (CI, computer interconnect, is the medium used in VAXclusters, but it may also be used to provide DECnet–VAX connections for nodes in the cluster). You can use the network configuration procedure NETCONFIG.COM (described in Section 3.3.2) to configure all circuits and lines automatically except for asynchronous circuits and lines, and CI circuits. (The DECnet–VAX installation procedure is given in the next section.)

3.3 Installing DECnet–VAX on Your System

This section describes the procedure for installing DECnet–VAX on your VMS operating system. Use this installation procedure to bring up your system as a node on an existing DECnet network. (If there is no existing DECnet network available, see Section 3.4.)

To perform the installation procedure, you should log in to the SYSTEM account on your node. The DECnet–VAX installation procedure consists of the following steps:

1. Purchase the DECnet–VAX license and the DECnet–VAX key and register the key on your system using the VMS License Management Utility. (See Section 3.3.1.)
2. Configure your DECnet–VAX node and define the remote node names. You can configure your node and turn on the network at your node either automatically or manually. (Follow the steps in Section 3.3.2.)
3. If you plan to use an asynchronous DECnet connection or CI connection, perform any steps needed to establish the connection. (See Section 3.3.3 for the procedure for establishing asynchronous connections. Refer to the *VMS Networking Manual* for information on establishing CI connections.)
4. Verify that your node is connected to the network. (See Section 3.3.4.)

3.3.1 Getting a DECnet–VAX License and Key

To permit your node to communicate with other nodes in the network, you must have a DECnet–VAX license and register a DECnet–VAX key on your system using the VMS License Management Utility. You can purchase either an end node or a full function license and the corresponding key. The end node key permits you to configure your node only as an end node. The full function key permits you to configure your node as either a routing node or an end node. You can also use the full function key to upgrade from end node to full function capability. The procedure for registering the DECnet–VAX key is described in the *VMS Version 5.0 Release Notes*. Refer to the *VMS License Management Utility Manual* for additional information on licensing.

You can register the DECnet–VAX key as the initial step in bringing up the network, or you can register it after performing the automatic configuration of DECnet–VAX (using NETCONFIG.COM), as described below. Be sure to determine whether the key you are registering is for the full function or end node DECnet capability. The full function DECnet–VAX key is DVNETRTG; the end node DECnet–VAX key is DVNETEND.

3.3.2 Configuring Your DECnet–VAX Node

You are now ready to configure your DECnet–VAX system. You can configure the node automatically or manually.

- You can use the automatic configuration procedure when you first bring up the node or when you reconfigure it completely. (See Section 3.3.2.2.)
- You can use manual configuration techniques to bring up a new node, reconfigure a node, or modify an existing configuration. (See Section 3.3.2.1.)

The system manager at each node in the network is responsible for the DECnet–VAX configuration database for the node. The database includes files that describe the local (executor) node and the other nodes in the network with which the local node can communicate, as well as the circuits and lines that connect the local node to the network. The network database also includes information on the logging collection points (such as the logging monitor) to which network events are reported. In addition, DECnet–VAX provides a default object database describing all objects (such as MAIL) known to the network. Each node in the network has such a database.

The configuration database comprises the *volatile database* (the working copy of the database that reflects current network conditions) and the *permanent database* (which provides the initial values for the volatile database when you start the network). Modifications to the volatile database exist only while the network is running. Changes made to the permanent database remain after the network is shut down, but do not affect the current system.

As system manager, you provide network component information, from the point of view of the local node, in the configuration database at the local node. Use the Network Control Program (NCP) to supply this information in

the form of parameter values, which determine how the various components of the network function together. Use NCP DEFINE commands to establish the contents of the permanent database and SET commands to specify the contents of the volatile database. Use PURGE commands to delete permanent database entries and CLEAR commands to delete or reset volatile database entries.

3.3.2.1 Configuring Your Node Manually

You can always configure your node manually; however, you have the option of doing it automatically (as described in the next section) if you are configuring a new node or completely reconfiguring a node.

If you decide to configure your node manually, you must enter NCP commands to establish the permanent configuration database and then turn on the network manually, causing the contents of the permanent database to be entered in the volatile database. A brief explanation of how to use NCP to establish your configuration database manually appears later in this section. (See Section 3.3.6.) For a detailed description of how to configure DECnet–VAX nodes, refer to the *VMS Networking Manual*.

If you decide to configure your node manually, you can optionally create a default nonprivileged DECnet account and directory for your node manually. (Establishment of the default DECnet account is described in Section 3.3.7.2.)

3.3.2.2 Configuring Your Node Automatically

You can use the interactive command procedure NETCONFIG.COM to configure your system automatically. NETCONFIG.COM configures all required permanent database entries except for remote nodes, asynchronous circuits and lines, and CI circuits. NETCONFIG.COM also sets up and turns on event logging. You can also use the command procedure to set up an optional default nonprivileged DECnet account on your system.

Use NETCONFIG.COM only if you are bringing up your node for the first time, or want to reconfigure your node completely, because the procedure purges any existing permanent database entries on your system (except for remote node entries). You must have the privilege SYSPRV to use NETCONFIG.COM to configure your node.

If you decide to use the NETCONFIG.COM command procedure to configure your node automatically, perform the steps given below. Default values appear in brackets [] after certain questions in the interactive dialog. To accept a default, press RETURN.

1 **Log in to the SYSTEM account on your node.**

2 **Invoke NETCONFIG.COM.** Enter the following command at the dollar sign ($) prompt:

```
$ @SYS$MANAGER:NETCONFIG
```

The following message is displayed:

```
                DECnet-VAX network configuration procedure

This procedure will help you define the parameters needed to get DECnet
running on this machine.  You will be shown the changes before they are
executed, in case you wish to perform them manually.
```

3 **Provide the node name.** You will be prompted as follows:

```
What do you want your DECnet node name to be?
```

Enter the node name you have selected (or have been assigned by the network manager). Your node name must be six alphanumeric characters or less, and must be unique among all node names in the network.

(If you are on a VAXcluster node, you must press RETURN to accept the default node name that appears in brackets at the end of the prompt. This default node name is based on the SYSGEN parameter SCSNODE. If no default node name is displayed, exit the procedure and use SYSGEN to set up a value for SCSNODE, then restart the procedure. The DECnet node name of a VAXcluster node must match the value of SCSNODE.)

4 **Provide the node address.** You will be prompted as follows:

```
What do you want your DECnet address to be?
```

Enter the node address you have selected (or been assigned by the network manager). The node address is a numeric value of the following format:

```
area-number.node-number
```

Area-number (1 to 63) designates the area in which the node is grouped and **node-number** (1 to 1023) designates the node's unique address within the area. If you do not specify an area number, the system will supply a default area number (the default value is 1).

(If you are on a VAXcluster node, you must press RETURN to accept the default node address that appears in brackets at the end of the prompt. This default node address is based on the SYSGEN parameter SCSSYSTEMID. If no default node address is displayed, exit the procedure and use SYSGEN to set up a value for SCSSYSTEMID, then restart the procedure. The DECnet node address of a VAXcluster node must match the value of SCSSYSTEMID.)

5 **Specify router or nonrouter status.** You will be prompted as follows:

```
Do you want to operate as a router?  [NO (nonrouting)]
```

Press RETURN to operate as a nonrouter (that is, as an end node). Type YES and press RETURN if you want your system to be a router and if you have registered the DECnet–VAX full function key or will register it before you start up the network.

6 **Set up the nonprivileged DECnet account.** You will be prompted as follows:

```
Do you want a default DECnet account? [YES]
```

Press RETURN to set up the default nonprivileged DECnet account and directory. Type NO and press RETURN if you do not want to set up the account. (For a description of the default nonprivileged DECnet account, see Section 3.3.7.)

7 **Apply the configuration.** The network configuration procedure displays the list of commands necessary to start up your network. (An example showing the commands appears later in this section.)

You will be prompted as follows:

```
Do you want these commands to be executed? [YES]
```

Press RETURN to configure the network; type NO and press RETURN to cancel the configuration operation. If you choose to configure the network, the procedure displays a series of information messages and the following statement:

```
The changes have been made.
```

8 **Turn on the network.** You will then receive the following messages, ending in a prompt:

```
If you have not already registered the DECnet-VAX key, then do so now.
After the key has been registered, you should invoke the procedure
SYS$MANAGER:STARTNET.COM to start up DECnet-VAX with these changes.
(If the key is already registered) Do you want DECnet started? [YES]:
```

You can respond to this prompt in either of the following ways:

- Type NO and press RETURN in response to the last prompt if you need to register the key on your system at this point. Register the key using the VMS License Management Utility. (The DECnet–VAX key registration procedure is described in the *VMS Version 5.0 Release Notes*.) Once the DECnet–VAX key is registered, you can then start up DECnet–VAX manually with these configuration changes by entering the following command:

  ```
  $ @SYS$MANAGER:STARTNET
  ```

  (You can also type NO and press RETURN in response to the last prompt if the key is already registered but you do not want to start the network until a later time.)

- Press RETURN in response to the last prompt if you want to start the network at this time and the key is already registered. The procedure turns on the network and displays the identification numbers of the created processes. When the dollar sign ($) prompt appears, you have successfully configured and turned on the DECnet–VAX network.

  If you want the network to be started automatically each time the VMS operating system is booted, enable the following command in the SYS$MANAGER:SYSTARTUP_V5.COM command procedure (by deleting the exclamation point at the beginning of this command line in the command procedure):

  ```
  $ @SYS$MANAGER:STARTNET
  ```

- Note that you can optionally press RETURN to start the network without the key being registered, if you want to use DECnet–VAX only at your local node. The key is required if you want to establish connections to other nodes in the network.

Example 3–1 shows the interactive dialog that is displayed when you invoke NETCONFIG.COM to configure node PURPLE with address 2.3 as an end node with a default DECnet account. In this example, node PURPLE is connected to Ethernet circuit UNA-0.

Example 3–1 Sample NETCONFIG.COM Dialogue

```
        DECnet-VAX network configuration procedure

This procedure will help you define the parameters needed to get
DECnet running on this machine.  You will be shown the changes before
they are actually executed, in case you wish to perform them manually.

What do you want your DECnet node name to be?          :  PURPLE
What do you want your DECnet address to be?            :  2.3
Do you want to operate as a router?   [NO (nonrouting)]:  RET
Do you want a default DECnet account?             [YES]:  RET

         Here are the commands necessary to set up your system.

-----------------------------------

$ RUN SYS$SYSTEM:NCP
    PURGE EXECUTOR ALL
    PURGE KNOWN LINES ALL
    PURGE KNOWN CIRCUITS ALL
    PURGE KNOWN LOGGING ALL
    PURGE KNOWN OBJECTS ALL
    PURGE MODULE CONFIGURATOR KNOWN CIRCUITS ALL
$ DEFINE/USER SYS$OUTPUT NL:
$ DEFINE/USER SYS$ERROR NL:
$ RUN SYS$SYSTEM:NCP     ! Remove existing entry, if any
    PURGE NODE 2.3 ALL
    PURGE NODE PURPLE ALL
$ RUN SYS$SYSTEM:NCP
    DEFINE EXECUTOR ADDRESS 2.3 STATE ON
    DEFINE EXECUTOR NAME PURPLE
    DEFINE EXECUTOR MAXIMUM ADDRESS 1023
    DEFINE EXECUTOR ROUTING TYPE NONROUTING IV
    DEFINE EXECUTOR NONPRIVILEGED USER DECNET
$ DEFINE/USER SYSUAF SYS$SYSTEM:SYSUAF.DAT
$ RUN SYS$SYSTEM:AUTHORIZE
    ADD DECNET /OWNER="DECNET DEFAULT" -
        /PASSWORD="" -
        /UIC=[376,376] /ACCOUNT=DECNET -
        /DEVICE=SYS$SYSDEVICE: /DIRECTORY=[DECNET] -
        /PRIVILEGE=(TMPMBX,NETMBX) -
        /FLAGS=(CAPTIVE) /LGICMD=NL: -
        /NOBATCH /NOINTERACTIVE
$ CREATE/DIRECTORY SYS$SYSDEVICE:[DECNET] /OWNER=[376,376]
$ RUN SYS$SYSTEM:NCP
    DEFINE LINE    UNA-0 STATE ON
    DEFINE CIRCUIT UNA-0 STATE ON COST 1
    DEFINE LOGGING MONITOR STATE ON
    DEFINE LOGGING MONITOR EVENTS 0.0-9
    DEFINE LOGGING MONITOR EVENTS 2.0-1
    DEFINE LOGGING MONITOR EVENTS 4.2-13,15-16,18-19
    DEFINE LOGGING MONITOR EVENTS 5.0-18
    DEFINE LOGGING MONITOR EVENTS 128.0-4

-----------------------------------
```

Example 3–1 Cont'd. on next page

Example 3–1 (Cont.) Sample NETCONFIG.COM Dialogue

```
Do you want these commands to be executed?                    [YES]: RET
The changes have been made.
If you have not already registered the DECnet-VAX key, then do so now.
After the key has been registered, you should invoke the procedure
SYS$MANAGER:STARTNET.COM to start up DECnet-VAX with these changes.

(If the key is already registered) Do you want DECnet started?[YES]: RET
```

9 **Define the other node names.** At the dollar sign ($) prompt, invoke the Network Control Program (NCP) by entering the following command:

```
$ RUN SYS$SYSTEM:NCP
```

For each remote node in the network that you want to identify by node name, enter the following command to define the node address and name in your permanent node database:

```
NCP>DEFINE NODE address NAME name
```

Address is the existing node address in the form *area-number.node-number* and **name** is the node name. If you omit the area number from the node address, the area number of your local node is used. The network manager or the system manager of the remote node you wish to define can provide you with the correct name and address.

If a node that you can access on your network has a node database that contains all the node names and addresses you want to define and you have the appropriate privileges to access that database, you can enter the following command at the NCP prompt (provided the network is turned on):

```
NCP>COPY KNOWN NODES FROM node-id TO PERMANENT
```

In this command, **node-id** is the node name or address of the remote node from which you are copying the information. If you specify the node name, that name must be in your volatile database. All the node names and addresses are copied to your permanent node database from the volatile node database of the remote node.

If your node is a member of a VAXcluster that uses an alias node identifier (an alias node name and address), your node can adopt the alias. Specify the following commands to define the alias node address and name in the permanent node database, and associate the alias identifier with your node:

```
NCP>DEFINE NODE address NAME name
NCP>DEFINE EXECUTOR ALIAS NODE node-id
```

For the **node-id**, you can specify either the alias node address or name that you have defined. Your node can then be identified by the alias node name and address as well as by its unique node name and address when DECnet is running.

Then enter the following commands to create the volatile node database for your node:

```
NCP>SET KNOWN NODES ALL
NCP>EXIT
```

The other nodes on the network should define your node name and node address in their node databases in order to be able to communicate with your node by node name. If a network manager assigned the unique node name and address to your node, the manager can define your node name in an overall network node database.

For additional information on using NCP to tailor the configuration database, refer to Section 3.3.6.

10 **Determine how to proceed.** You have completed the network startup procedure. If you plan to use asynchronous DECnet, continue to the next section, which describes how to establish asynchronous connections.

3.3.3 Establishing Asynchronous DECnet Connections to Other Systems

The automatic network configuration procedure described above does not configure asynchronous lines and circuits. As a VMS system manager, you have the option of connecting your VMS system to another system by means of a low-cost, low-speed asynchronous DECnet line. The two types of asynchronous DECnet connections you can establish are:

- A static asynchronous DECnet connection, which creates a permanent DECnet link to a single remote node. (See Section 3.3.3.1.)
- A dynamic asynchronous DECnet connection, which provides a temporary DECnet link. You can establish dynamic connections to different remote nodes at different times. (See Section 3.3.3.2.)

Note that non-VMS systems that support DECnet asynchronous DDCMP lines can make asynchronous DECnet connections to VMS systems. The asynchronous connection can be between two routers, a router and an end node, or two end nodes. If you are on an end node and want to make an asynchronous connection, it will be your only connection to the network, because an end node can only have one circuit active at a time.

3.3.3.1 Establishing a Static Asynchronous Connection

A static asynchronous DECnet connection is a permanent connection between two nodes. This type of connection can be made in one of two ways:

- The nodes can be connected by a physical line (a null modem cable) attached to a terminal port at each system. No modems are required. You can communicate with the other system at any time.
- The connection can be made over a dialup line using modems at both ends of the line. For example, your VMS system can establish a static asynchronous connection to a remote node over a telephone line.

You can configure your static asynchronous line as soon as you have executed NETCONFIG.COM, and then turn on the network manually. If your system is brought up as a routing node, you can establish a static asynchronous connection at any time, no matter how many network connections you already have.

Follow the steps outlined in this section to establish a static asynchronous connection. For the connection to be successful, the node with which you are creating a DECnet link must also establish an asynchronous DECnet connection with your node. (Note that the line speeds at each end of the connection must be the same.)

1 Log in to the SYSTEM account on your VMS node.

2 Load the asynchronous DDCMP driver, NODRIVER (NOA0). Enter the commands shown below at your terminal (or include them in the SYS$MANAGER:SYSTARTUP_V5.COM command procedure before you boot the system).

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN>  CONNECT NOA0/NOADAPTER
SYSGEN>  EXIT
```

The asynchronous driver must be loaded before any asynchronous connection can be made.

3 To set up the terminal line to become a static asynchronous DECnet line, enter the DCL command SET TERMINAL at your terminal. If there is more than one terminal attached to your VMS system, you must specify a SET TERMINAL command for each terminal line that will be used for a static asynchronous DECnet connection.

- **Nondialup line:** For a nondialup configuration, enter the following SET TERMINAL command to convert a terminal line to a static asynchronous line:

  ```
  $ SET TERMINAL/PERMANENT/PROTOCOL=DDCMP device-name:
  ```

  In this command, **device-name** is the name of the terminal port that is connected to the line that you want to make a static asynchronous DECnet line. (All references to a device in this section refer to the terminal port.)

  For example, to set up a 2400-baud terminal line connected to port TTA0 on your system to be a static asynchronous DECnet line, enter the following command:

  ```
  $ SET TERMINAL/PERMANENT/PROTOCOL=DDCMP TTA0:
  ```

- **Dialup line:** For a dialup configuration, enter the following SET TERMINAL command to convert the terminal line to a static asynchronous DECnet line with modem control.

  ```
  $ SET TERMINAL/PERMANENT/MODEM/NOAUTOBAUD -
  _$ /NOTYPE_AHEAD/PROTOCOL=DDCMP device-name:
  ```

  For example, if the line connected to terminal port TTB0 is connected to a 2400-baud modem, specify this command:

  ```
  $ SET TERMINAL/PERMANENT/MODEM/NOAUTOBAUD -
  _$ /NOTYPE_AHEAD/PROTOCOL=DDCMP TTB0:
  ```

You can ensure that these SET TERMINAL commands will be executed automatically each time the network is started in the future. Modify your SYS$MANAGER:SYSTARTUP_V5.COM command procedure to include all required SET TERMINAL commands before the command @SYS$MANAGER:STARTNET.

4 After configuring your node, configure the asynchronous lines and circuits in the network database. Use NCP commands to define each asynchronous line and accompanying circuit as being in the ON state. (The line and circuit are turned on when

SYS$MANAGER:STARTNET.COM is executed.) Enter the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>DEFINE LINE dev-c-u STATE ON RECEIVE BUFFERS 4 -
     _LINE SPEED  baud-rate
NCP>DEFINE CIRCUIT dev-c-u STATE ON
NCP>EXIT
```

Baud-rate is the speed at which the line sends and receives data. For an asynchronous line or circuit, **dev-c-u** is defined as follows:

dev The first two letters of the asynchronous device name (possible values are TT and TX).

c A decimal number (0 or a positive integer) designating a device's hardware controller. If the third letter of the device name is A, **c** equals 0. If the third letter of the device name is B, **c** equals 1, and so on.

u The unit number of the device name; **u** is always equal to 0 or a positive integer.

(An example is the device identifier TT-0-0, which represents the asynchronous device name TTA0.)

A minimum of four buffers should be allocated for data reception over the line.

For example, to use a line connected to port TTA0 at 2400 baud, run NCP and enter the following commands to define the line and circuit:

```
$ RUN SYS$SYSTEM:NCP
NCP>DEFINE LINE TT-0-0 STATE ON RECEIVE BUFFERS 4 -
     _LINE SPEED  2400
NCP>DEFINE CIRCUIT TT-0-0 STATE ON
NCP>EXIT
```

If the line speed at the other end of the connection is changed after the initial static asynchronous connection is made, you can use the DEFINE LINE command specified above to change the line speed for your end of the connection to match the line speed at the other end. The line speed will be changed the next time the line is turned on.

5 For security over a dialup connection, you can run NCP and establish optional transmit and receive passwords for the local end of the static asynchronous dialup link. The transmit password is the password sent to the other node during connection startup; the receive password is the password expected from the other node during connection startup. You must also use NCP to specify that your asynchronous circuit is to verify the password supplied by the other node. If the correct passwords are not supplied, the asynchronous connection cannot be made.

Although transmit and receive passwords are not mandatory for static asynchronous dialup links, they add to the security of your DECnet connection. Passwords can contain from one to eight alphanumeric characters and must be delimited with quotation marks if they contain spaces. Specify the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>DEFINE CIRCUIT dev-c-u VERIFICATION ENABLED
NCP>DEFINE NODE node-id TRANSMIT PASSWORD transmit-password -
     _RECEIVE PASSWORD receive-password
NCP>EXIT
```

Node-id is the name of the remote node to which your node will be connected.

In the following example, the name of your local node is LOCALA, the name of the remote node is REMOTB, your asynchronous circuit is TT-0-0, the transmit password is PASSA, and the receive password is PASSB. Enter the following on node LOCALA:

```
$ RUN SYS$SYSTEM:NCP
NCP>DEFINE CIRCUIT TT-0-0 VERIFICATION ENABLED
NCP>DEFINE NODE REMOTB TRANSMIT PASSWORD PASSA -
     _RECEIVE PASSWORD PASSB
NCP>EXIT
```

Note that if you have defined passwords for the local end of the link, you must notify the remote node system manager to establish transmit and receive passwords for the remote end of the static asynchronous DECnet dialup link. For the above example, the remote system manager should enter these commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>DEFINE CIRCUIT TT-1-0 VERIFICATION ENABLED
NCP>DEFINE NODE LOCALA TRANSMIT PASSWORD PASSB -
     _RECEIVE PASSWORD PASSA
NCP>EXIT
```

6 If the network is not already on, turn on the network at your node by entering the following command:

```
$ @SYS$MANAGER:STARTNET
```

This command starts the network and causes the permanent database entries defined above to be entered in the volatile database on the running network.

If the network was already running before you began the static asynchronous connection procedure, enter the following commands to cause the permanent database entries to be entered in the volatile database.

```
$ RUN SYS$SYSTEM:NCP
NCP>SET LINE dev-c-u ALL
NCP>SET CIRCUIT dev-c-u ALL
NCP>SET NODE node-id ALL
NCP>EXIT
```

If the line and circuit could not be set on in the volatile database, causing DECnet to fail to gain control of the line, the following error message appears:

```
% NCP-I-NMLRSP, LISTENER RESPONSE - Operation failure
```

If the static asynchronous connection cannot be made, refer to the section on asynchronous connection problems. (See Chapter 4.)

7 If you want to turn off the asynchronous lines temporarily, run NCP and enter the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>SET LINE dev-c-u STATE OFF
NCP>SET CIRCUIT dev-c-u STATE OFF
NCP>CLEAR LINE dev-c-u ALL
NCP>CLEAR CIRCUIT dev-c-u ALL
NCP>EXIT
```

To turn the static asynchronous DECnet line back on, enter the following NCP commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>SET LINE dev-c-u ALL
NCP>SET CIRCUIT dev-c-u ALL
NCP>EXIT
```

8 If you want to switch an asynchronous DECnet line back to a terminal line with DECnet running, you must clear the line and circuit entries from the network volatile database. To clear the entries, enter these commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>SET LINE dev-c-u STATE OFF
NCP>SET CIRCUIT dev-c-u STATE OFF
NCP>CLEAR LINE dev-c-u ALL
NCP>CLEAR CIRCUIT dev-c-u ALL
NCP>EXIT
```

To switch the line for which modem control was not enabled back to a terminal line, use this command:

```
$ SET TERMINAL/PERMANENT/PROTOCOL=NONE device-name:
```

For example, to switch the nondialup line connected to port TTA0 back to a terminal line, enter the command:

```
$ SET TERMINAL/PERMANENT/PROTOCOL=NONE TTA0:
```

To switch the line for which modem control was enabled back to a terminal line, use this command:

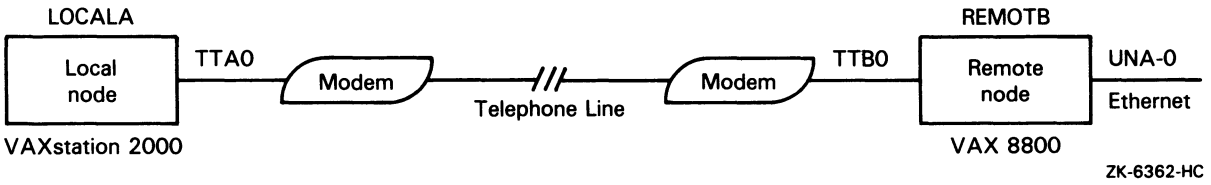
```
$ SET TERMINAL/PERMANENT/MODEM/AUTOBAUD -
_$ /TYPE_AHEAD/PROTOCOL=NONE device-name:
```

For example, to switch the dialup line connected to port TTB0 back to a terminal line, enter the command:

```
$ SET TERMINAL/PERMANENT/MODEM/AUTOBAUD -
_$ /TYPE_AHEAD/PROTOCOL=NONE TTB0:
```

Example 3–2 lists the commands that the system managers of two VMS nodes must put into the permanent databases to cause a static asynchronous connection to be established over the dialup line shown in Figure 3–4 when the network is turned on at each node. Note that if an asynchronous connection is made to a non-VMS system, the commands issued at the remote node must be appropriate for that system.

Figure 3–4 A Typical Static Asynchronous Dialup Connection



Example 3–2 Sample Commands for a Static Asynchronous Dialup Connection

```
Commands issued at the local VMS node (LOCALA):

$ RUN SYS$SYSTEM:SYSGEN
SYSGEN>  CONNECT NOA0/NOADAPTER
SYSGEN>  EXIT
$ SET TERMINAL/PERMANENT/MODEM/DIALUP/NOAUTOBAUD/NOTYPE_AHEAD -
_$ /PROTOCOL=DDCMP TTA0:
$ RUN SYS$SYSTEM:NCP
NCP>DEFINE LINE TT-0-0 STATE ON RECEIVE BUFFERS 4 LINE SPEED 2400
NCP>DEFINE CIRCUIT TT-0-0 STATE ON VERIFICATION ENABLED
NCP>DEFINE NODE REMOTB TRANSMIT PASSWORD PASSB RECEIVE PASSWORD PASSA
NCP>EXIT

Commands issued at the remote VMS node (REMOTB):

$ RUN SYS$SYSTEM:SYSGEN
SYSGEN>  CONNECT NOA0/NOADAPTER
SYSGEN>  EXIT
$ SET TERMINAL/PERMANENT/MODEM/NOAUTOBAUD/NOTYPE_AHEAD -
_$ /PROTOCOL=DDCMP TTB0:
$ RUN SYS$SYSTEM:NCP
NCP>DEFINE LINE TT-1-0 STATE ON RECEIVE BUFFERS 4 LINE SPEED 2400
NCP>DEFINE CIRCUIT TT-1-0 STATE ON VERIFICATION ENABLED
NCP>DEFINE NODE LOCALA TRANSMIT PASSWORD PASSA RECEIVE PASSWORD PASSB
NCP>EXIT
```

3.3.3.2 Establishing a Dynamic Asynchronous Connection

A dynamic asynchronous DECnet connection is a temporary connection between two nodes, normally over a telephone line through the use of modems. The line at each end of the connection can be switched from a terminal line to a dynamic asynchronous DECnet line. Configuration of dynamic asynchronous lines is performed automatically by DECnet during establishment of a dynamic connection. A dynamic asynchronous connection is normally maintained only for the duration of a telephone call.

Note: A dynamic asynchronous connection to a VMS node can be initiated from any VMS or non-VMS node that supports the DECnet asynchronous DDCMP protocol.

On a VMS node, you have the option of performing the initial steps of the dynamic asynchronous connection process (steps 1 and 2 below) before you turn on the network at your node (step 3). The later steps of the process (starting with step 4) must occur when the line is being switched to DECnet.

Follow the steps outlined below to establish a dynamic asynchronous DECnet connection. This procedure assumes the local VMS node is originating the connection and switching the terminal line on for DECnet use. The connection must be to a VMS node on which you have an account with NETMBX privilege. The steps that the system manager at the remote VMS node must perform in order for the dynamic asynchronous DECnet link to be established successfully are also indicated below.

1 Log in to the SYSTEM account and enter the following commands interactively (or include them in the SYS$MANAGER:SYSTARTUP_V5.COM command procedure before you boot the system). These commands load the asynchronous driver NODRIVER (NOA0) and install DYNSWITCH software on your system.

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN>  CONNECT NOA0/NOADAPTER
SYSGEN>  EXIT
$ INSTALL:=$SYS$SYSTEM:INSTALL
$ INSTALL/COMMAND
INSTALL> CREATE SYS$LIBRARY:DYNSWITCH/SHARE -
_ /PROTECT/HEADER/OPEN
INSTALL> EXIT
```

The system manager of the remote VMS node must also enter these commands.

Additionally, the system manager at the remote VMS node must enter the commands given below. These commands enable the use of *virtual terminals* for the terminal line that is to be switched, and set the DISCONNECT characteristic for the terminal line. (The virtual terminal capability permits the process to continue running if the physical terminal you are using becomes disconnected.)

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN>  CONNECT VTA0/NOADAPTER/DRIVER=TTDRIVER
SYSGEN>  EXIT
$ SET TERMINAL/EIGHT_BIT/PERMANENT/MODEM/DIALUP -
_$ /DISCONNECT device-name:
```

Device-name is the name of the terminal port to which the dynamic asynchronous connection is made.

2 You must establish the required transmit password at the originating end of the dynamic asynchronous dialup link. The transmit password is the password sent to the remote node during connection startup. Use NCP to enter a command to define the transmit password for the remote node. The password can contain one to eight alphanumeric characters and should not contain any spaces. Specify the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>DEFINE NODE node-id TRANSMIT PASSWORD password
NCP>EXIT
```

Node-id is the name of the remote node with which your node is forming a connection.

In the following example, the node name of your local node is LOCALA, the transmit password is PASSA, and the remote node with which you are creating a dynamic asynchronous dialup link is REMOTC.

```
$ RUN SYS$SYSTEM:NCP
NCP>DEFINE NODE REMOTC TRANSMIT PASSWORD PASSA
NCP>EXIT
```

For each remote node with which you will create a dynamic asynchronous DECnet dialup link, you must define a transmit password in a separate command.

The system manager for the node at the other end of the connection must define that same password as a receive password for your node (the password expected to be received from your node). The remote system manager should also specify the parameter INBOUND ROUTER or INBOUND ENDNODE, to indicate the type of node (router or end node) that is expected to initiate the dynamic connection. These are the commands the remote manager should enter:

```
$ RUN SYS$SYSTEM:NCP
NCP>DEFINE NODE node-id RECEIVE PASSWORD password INBOUND node-type
NCP>EXIT
```

For example, if your node LOCALA is an end node and your transmit password is PASSA, the manager at REMOTC should issue the following command:

```
$ RUN SYS$SYSTEM:NCP
NCP>DEFINE NODE LOCALA RECEIVE PASSWORD PASSA INBOUND ENDNODE
NCP>EXIT
```

3 DECnet must be running on both nodes for the remaining steps. If you have not already done so, turn on the network by entering the following command (and request that the remote system manager do so also):

```
$ @SYS$MANAGER:STARTNET
```

If the network was already running before you began the dynamic asynchronous connection procedure, enter these commands to cause the permanent database entry to be entered in the volatile database:

```
$ RUN SYS$SYSTEM:NCP
NCP>SET NODE node-id ALL
NCP>EXIT
```

4 The remaining steps can be performed by any VMS user with NETMBX privilege. Log in to your local VMS system and enter the following DCL command on your terminal to cause your process to function as a *terminal emulator* (which makes the remote terminal appear to be a local terminal connection):

```
$ SET HOST/DTE device-name:
```

Device-name is the name of your local terminal port that is connected to the modem. If both systems use modems with autodial capabilities (for example, DF03, DF112 or DF224 modems that support an autodial protocol), you can optionally include the /DIAL qualifier on the SET HOST/DTE command to cause automatic dialing of the modem on the remote node, as follows:

```
$ SET HOST/DTE/DIAL=number device-name:
```

5 If you are not using automatic dialing, dial in to the remote node manually.

6 Once the dialup connection is made and you receive the remote VMS system welcome message, log in to your account on the remote node.

7 While logged in to your account on the remote node, enter the following command to cause the line to be switched to a DECnet line automatically:

```
$ SET TERMINAL/PROTOCOL=DDCMP/SWITCH=DECNET
```

The following message indicates that the DECnet link is being established:

```
%REM-S-END - control returned to local-nodename::
$
```

To check whether the communications link has come up, specify the following command on the local system:

```
$ RUN SYS$SYSTEM:NCP
NCP>SHOW KNOWN CIRCUITS
NCP>EXIT
```

The resulting display should list a circuit identified by the mnemonic TT or TX, depending on the asynchronous device installed on the line, and indicate that it is in the ON state.

When the DCL prompt ($, by default) appears on your terminal screen, you can begin to communicate with the remote node over the asynchronous DECnet connection.

If the dynamic connection is not made successfully, refer to the section on asynchronous connection problems. (See Chapter 4.)

8 As an alternative to switching the terminal line to a DECnet line automatically (as described in the previous step), you can switch the line manually. If you originate a dynamic connection to a VMS node from a non-VMS system, manual switching is required; from a VMS system, it is optional. If you are originating the connection from a non-VMS node, follow system-specific procedures to log in to the remote VMS node by means of terminal emulation.

Once you are logged in to the remote node, two steps are required to perform manual switching:

a. Using your account on the remote VMS node, specify the SET TERMINAL command described in step 7, but add the /MANUAL qualifier:

```
$ SET TERMINAL/PROTOCOL=DDCMP/SWITCH=DECNET/MANUAL
```

You will receive the following message from the remote node indicating the remote system is switching its line to DECnet use:

```
%SET-I-SWINPRG The line you are currently logged over is becoming
              a DECnet line
```

b. You should exit from the terminal emulator and switch your line manually to a DECnet line. The procedure depends on the specific operating system on which you are logged in.

MS-DOS procedure: The following procedure illustrates how an MS-DOS™ system user on a DIGITAL Rainbow computer, who is

™ MS-DOS is a trademark of Microsoft Corporation.

originating a dynamic connection to a VMS system, exits from the terminal emulator and turns on the DECnet line.

- Exit from the terminal emulator poly-TRM™ on the Rainbow by pressing the EXIT key (function key 10).
- When an MS–DOS prompt (E>) is displayed on the screen, enter the following commands to turn on the line to DECnet and verify that the line is up:

```
E> NCP SET LINE STATE ON
E> NCP READ LOGGING
```

  After a short interval, a display should appear, showing that the line state is on and the adjacency is up, indicating that DECnet is started at the MS–DOS node.

VMS procedure: The following example shows how a VMS user originating a dynamic connection would perform this procedure.

- Exit from the terminal emulator by pressing the backslash (\) key and the CTRL key simultaneously on your VMS system.
- Enter the following command to switch your terminal line to a DECnet line manually:

```
$ SET TERMINAL/PROTOCOL=DDCMP TTA0:
```

  TTA0 is the name of the terminal port on the local node.

- Enter NCP commands to turn on the line and circuit connected to your terminal port TTA0 manually, as in the following example:

```
$ RUN SYS$SYSTEM:NCP
NCP>SET LINE TT-0-0 RECEIVE BUFFERS 4 LINE SPEED 2400 STATE ON
NCP>SET CIRCUIT TT-0-0 RECEIVE BUFFERS 4 STATE ON
NCP>EXIT
```

  Asynchronous DECnet is then started on the local VMS node.

9 You can terminate the dynamic asynchronous link in one of two ways:

a. Break the telephone connection.

b. Run NCP and turn off either the asynchronous line or circuit. The two commands you can use are as follows:

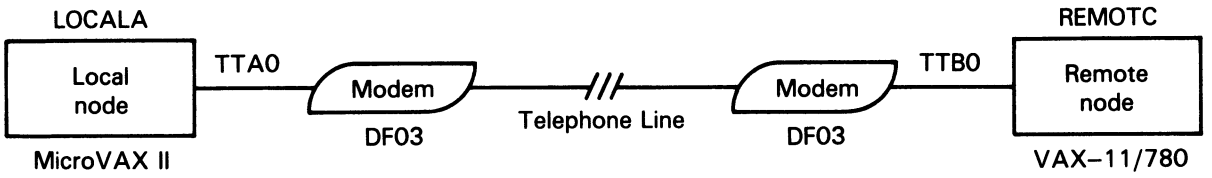
```
$ RUN SYS$SYSTEM:NCP
NCP>SET LINE dev-c-u STATE OFF
NCP>SET CIRCUIT dev-c-u STATE OFF
NCP>EXIT
```

If either of the above NCP commands is entered at the remote node, the line returns to terminal mode immediately. If the command is entered at the local (originating) VMS node, the remote line and circuit remain on for approximately four minutes and then the line returns to terminal mode.

™ poly-TRM is a trademark of Polygon Associates, Inc.

An illustration of the establishment of a dynamic asynchronous connection is shown in Figure 3–5. The commands that must be entered at each end of the connection are shown in Example 3–3.

Figure 3–5 A Typical Dynamic Asynchronous Connection



Example 3–3 Sample Commands for a Dynamic Asynchronous Connection

```
Commands issued at both the local VMS node (LOCALA) and the remote VMS node
(REMOTC):

$ RUN SYS$SYSTEM:SYSGEN
SYSGEN>  CONNECT NOA0/NOADAPTER
SYSGEN>  EXIT
$ INSTALL:=$SYS$SYSTEM:INSTALL
$ INSTALL/COMMAND
INSTALL> CREATE SYS$LIBRARY:DYNSWITCH/SHARE/PROTECT/HEADER/OPEN
INSTALL> EXIT

Commands issued at the remote node (REMOTC):

$ RUN SYS$SYSTEM:SYSGEN
SYSGEN>  CONNECT VTA0/NOADAPTER/DRIVER=TTDRIVER
SYSGEN>  EXIT
$ SET TERMINAL/EIGHT_BIT/PERMANENT/MODEM/DIALUP/DISCONNECT TTB0:
$ RUN SYS$SYSTEM:NCP
NCP>DEFINE NODE LOCALA RECEIVE PASSWORD PASSA INBOUND ENDNODE
NCP>SET NODE LOCALA ALL
NCP>EXIT

Commands issued at the local node (LOCALA):

$ RUN SYS$SYSTEM:NCP
NCP>DEFINE NODE REMOTC TRANSMIT PASSWORD PASSA
NCP>SET NODE REMOTC ALL
NCP>EXIT
$ SET HOST/DTE/DIAL=8556543 TTA0:

! After dialing in automatically to REMOTC, log in to your account on REMOTC.

$ SET TERMINAL/PROTOCOL=DDCMP/SWITCH=DECNET
%REM-S-END - control returned to LOCALA:
$
```

3.3.4 Verifying Successful Connection to the Network

To verify your connection to the network, you can optionally perform the following tests. These tests demonstrate whether your node can communicate with an adjacent node, that is, a node connected to your node by a single physical line. (Your node can reach an adjacent node directly, without data having to be routed through an intermediate node.)

1 Run the User Environmental Test Package (UETP), if it is available, to test DECnet hardware and software on your node and an adjacent node. Log in to the SYSTEST account using the password UETP and enter the following command:

```
$ @SYS$TEST:UETP RET
Run "ALL" UETP phases or a "SUBSET" [ALL]? S RET
You can choose one or more of the following phases:
DEVICE, LOAD, DECNET, CLUSTER
Phase(s): DECNET RET
```

The DECnet phase performs tests on the following:

- The local node (the node on which the UETP software is running)
- All circuits in sequence
- All adjacent nodes, and all circuits in parallel

A test on an adjacent node should normally last no more than two minutes. The DECnet node and circuit counters are zeroed at the beginning of the test to permit failure monitoring. The UETP output displays the condition of the circuit to the adjacent node and any response timeouts or errors indicated by the counters. (For a complete description of the UETP procedure, refer to your processor-specific installation and operations guide.)

2 Copy a file to an adjacent VMS node (if one is available), copy it back to your node, and compare the two files. Arrange with the system manager of the adjacent node to obtain an account with NETMBX privilege that will permit you to access a directory on that node. Then perform the steps given below. (Refer to Chapter 2 for a description of how to perform file operations over the network.)

a. Create a temporary file for the test. To create the file, copy one of your files and name it TEST1.TMP.

b. Copy your test file to the directory on the remote node.

c. Issue a DIRECTORY command specifying the test file TEST1.TMP in the remote directory, to verify that the file has been copied.

d. Copy the file back to your own node, using the file name TEST2.TMP.

e. Issue a DIRECTORY command specifying the name of the file (TEST2.TMP) in your own directory, to verify that the file has been copied back to your node.

f. Issue a DIFFERENCES command to compare the original file (TEST1.TMP) and the file copied back from the remote node (TEST2.TMP).

g. If the files are the same, delete all test files on both nodes.

The DCL commands in Example 3–4 show how a system manager copies a file (named LOCALFILE.LIS) to TEST1.TMP locally and then copies the TMP file to a remote directory (RNODE::DISK1:[GENERAL]) and back again, comparing the results to verify that the local node is connected to the network. If you use the commands in this example, substitute the name of your own file for the file name LOCALFILE.LIS, and the name of the remote directory to which you have access in place of the directory name (RNODE::DISK1:[GENERAL]).

Example 3–4 Sample Commands to Verify Connection to Another Node

```
$ COPY LOCALFILE.LIS TEST1.TMP
$ COPY TEST1.TMP RNODE::DISK1:[GENERAL]*
$ DIRECTORY RNODE::DISK1:[GENERAL]TEST1.TMP
$ COPY RNODE::DISK1:[GENERAL]TEST1.TMP TEST2.TMP
$ DIRECTORY TEST2.TMP
$ DIFFERENCES TEST1.TMP TEST2.TMP
$ DELETE TEST1.TMP;*,TEST2.TMP;*
$ DELETE RNODE::DISK1:[GENERAL]TEST1.TMP;*
```

3 Send mail to and receive mail from another node that supports the Mail Utility. To carry out this test, arrange with the system manager of a remote node on your network to have access to an account (with WRITE and DELETE privileges) on that node. Then follow the steps given below. (For a discussion of using MAIL over the network, see Chapter 2.)

   a. Using the Mail Utility, send a mail message to the account on the remote node.

   b. Log in to the account on the remote node using the SET HOST command.

   c. Read the mail message as received on the remote node, forward it back to the sender, exit from MAIL on the remote node, and log out of the remote system.

   d. Check that you have received the mail message on your local node.

If these tests are successful and you do not receive an error message from DECnet, you have verified that DECnet–VAX is installed correctly. If you are not able to carry out the verification tests successfully, the problem may be software or hardware errors, or possibly transient network problems, as described below.

- **Software or hardware errors:** If you receive DECnet error messages, refer to the section on common error messages. (See Section 4.3.1.)

  You can review the DECnet–VAX installation procedure to ensure that you followed all instructions in installing the software. You can also check the communications hardware to be sure it is set up and connected properly, according to the instructions in the hardware user manuals.

  To test the hardware, perform controller loopback tests to determine if the communications controller in your processor is working. Also check the controller on the remote node. For a summary description of controller loopback testing, refer to the section on testing the network. (See Chapter 4.) If these tests fail, contact your network manager.

- **Transient network problems:** If you cannot verify connectivity because a node is not reachable or the network is slow to respond, try to rerun the tests at a time when you are sure the nodes are available or when computer usage is not at a peak. DECnet messages indicate when a remote node is not reachable.

3.3.5 Shutting Down and Restarting the Network

The network shuts down automatically as part of the normal VMS system shutdown procedure. If your VMS system is running, you can shut down the network at your local node without destroying any active logical links by entering the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>SET EXECUTOR STATE SHUT
NCP>EXIT
```

When this command sequence is issued, no new links are allowed; when all existing links are disconnected, the network is turned off.

While your VMS system is running, you can stop the network at your node by entering the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>SET EXECUTOR STATE OFF
NCP>EXIT
```

All logical links are terminated immediately and the network is stopped.

To turn the network on manually, specify the following:

```
$ @SYS$MANAGER:STARTNET
```

To start the network if it is not currently active, you must be logged in to the SYSTEM account or have the privileges listed at the beginning of the STARTNET.COM command procedure.

To cause the network to be started each time the VMS operating system is booted, enable the following command in the SYS$MANAGER:SYSTARTUP_V5.COM command procedure:

```
$ @SYS$MANAGER:STARTNET
```

The command is supplied in the command procedure; to enable it, use a text editor to delete the exclamation point at the start of the command line. The network will be turned on automatically as part of the VMS system startup. You will not have to turn on the network again unless you should explicitly shut down the network or remove the network startup invocation from the site-specific startup command procedure.

3.3.6 Using NCP to Create and Tailor the Configuration Database

The system manager is responsible for configuring the node for network operation by supplying information in the DECnet–VAX configuration database about the following network components:

- The local (executor) node
- Remote nodes with which the local node can communicate
- Local circuits
- Local lines
- Network objects
- Network event logging

The configuration database is actually two databases: a permanent database that establishes the default parameter values for node startup, and a volatile database that contains the current parameter values in a functioning network.

You can use the Network Control Program (NCP) Utility to build the network configuration database manually or to modify its contents. If you are configuring the node for the first time, you can use the automatic configuration command procedure, NETCONFIG.COM, to establish parameters needed to get DECnet running. The procedure for using NETCONFIG.COM is described in an earlier section. (See Section 3.3.2.2.)

When you run NCP and enter a command, NCP will prompt you for selected parameters if you do not supply them. NCP also provides a HELP facility with information about each command, which you can access as follows:

```
$ RUN SYS$SYSTEM:NCP
NCP>HELP [topic...]
```

Refer to the *VMS Network Control Program Manual* for a complete description of the NCP command language. For a quick reference guide to NCP command formats, see the *VMS Mini-Reference* manual.

Use NCP SET commands to establish the contents of the volatile database, and DEFINE commands to establish the contents of the permanent database. You must have OPER privilege to change the volatile database and SYSPRV privilege to change the permanent database.

The permanent database information is supplied to the volatile database when the network is started (that is, the STARTNET.COM command procedure is executed). You can also use the ALL parameter with the SET command to cause all permanent database entries for a network component to be loaded into the volatile database.

The basic NCP commands required to define the network components in the permanent configuration database are as follows:

```
$ RUN SYS$SYSTEM:NCP
NCP>DEFINE EXECUTOR
NCP>DEFINE NODE node-id
NCP>DEFINE CIRCUIT circuit-id
NCP>DEFINE LINE line-id
NCP>DEFINE OBJECT object-name
NCP>DEFINE LOGGING MONITOR STATE ON
NCP>DEFINE LOGGING MONITOR EVENTS event-list
NCP>EXIT
```

In an NCP command, **node-id** can be a node name a maximum of six alphanumeric characters long, or a node address in the form *area-number.node-number*, where the area number can be from 1 to 63 and the node number can be from 1 to 1023. If no area number is specified, the area number of the executor node is used. The default area number for the executor is 1. The **object-name** can be up to 12 characters long.

The **circuit-id** and **line-id** values are in the form **dev-c[-u]**, defined as follows:

dev A device name (see Table 3–2)

c A decimal number (0 or a positive integer) designating a device's hardware controller

u The unit number of the device name (0 or a positive integer); included if more than one unit is associated with the controller

Table 3–2 DECnet–VAX Device Names

Circuit or Line Device	Device Name
Ethernet	UNA QNA SVA BNA
Synchronous DDCMP	DMB DMC DMF DMP
Asynchronous DDCMP	TT TX

For example, the circuit and line identification for an Ethernet UNA device is in the form UNA-c (such as UNA-0); an example for a QNA device is QNA-0. An example of a synchronous DDCMP device identifier is DMC-2, and an example of an asynchronous device identifier is TT-1-0.

NCP commands also recognize the plural forms of the network component names: KNOWN NODES, KNOWN CIRCUITS, KNOWN LINES, KNOWN OBJECTS.

To modify the current configuration of your node, you can issue SET commands for any network component. For example, to add circuit and line entries for the Ethernet UNA device (the DEUNA), specify the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>SET LINE UNA-0 STATE ON
NCP>SET CIRCUIT UNA-0 STATE ON
NCP>EXIT
```

To determine the contents of your network configuration database, use the NCP commands LIST and SHOW. The LIST command displays information in the permanent database; the SHOW command displays volatile database entries. (For more information on monitoring the network using the network databases, see Chapter 4.) To delete entries from the configuration database, use the PURGE and CLEAR commands. The PURGE command deletes permanent database entries; the CLEAR command deletes or resets volatile database entries.

For example, to list the permanent name and address of a node, enter the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>LIST NODE node-id
NCP>EXIT
```

To delete a node from the permanent database, enter the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>PURGE NODE node-id ALL
NCP>EXIT
```

Node-id can be either the node name or the node address. You can also delete an individual parameter for a node. For example, to purge the RECEIVE PASSWORD parameter for node PURPLE, enter the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>PURGE NODE PURPLE RECEIVE PASSWORD
NCP>EXIT
```

Because the PURGE command does not affect the volatile (memory-resident) copy of the DECnet database, you can access a node deleted with the PURGE command until DECnet is started again. If you use the CLEAR command to delete a node entry, the node entry will reappear in the volatile database after DECnet is started again.

3.3.7 Providing Security for Your DECnet–VAX Node

Some of the security measures that you can use to protect your files and system in a network environment are summarized in this section.

As manager of a VMS node, you can protect your system against unauthorized access by users on other nodes in the network by setting passwords for any accounts that you may create. Otherwise, users on other nodes could gain full access to your system by using the SET HOST command to log in to one of the accounts on your node. (Logging in to a VMS node using the SET HOST command is described in Section 3.1.2.)

3.3.7.1 Protecting Files and Using Proxy Accounts

As a user on a VMS node, you can protect the files in your directory against access over the network. To set limits on who can access the files in your account, specify the DCL command SET PROTECTION. If your file is protected, a VMS user on a remote node who wants to access your file must be able to specify the user name and password of a local account that has the appropriate privileges to access the file. A remote user to whom you have given this information must then include the authorization information in the form of an access control string, *"username password"*, in the VMS file specification used to access your file:

```
node"username  password"::device:[directory]filename.type;version
```

For example, if Jones on remote node MIAMI wants to obtain a copy of the file TEST.DAT from user Smith's directory (on the device WORK1) at local node BOSTON, he should specify the following command, which includes Smith's password:

```
$ COPY BOSTON"SMITH ABCDEF"::WORK1:[SMITH]TEST.DAT *
```

(See Chapter 2 for a description of network file operations.)

Establishing proxy accounts. As system manager of your node, you can maintain the security of passwords by preventing their transmission over the network. You can permit selected outside users to access particular accounts on your node without having to send any explicit access control information over the network. To do this, you must create a proxy account that allows a remote user to have access privileges on your node without having a private account on your node. If the remote user is assigned a proxy account on your local node that maps into a local user account, the remote user assumes the same access privileges as the owner of the local account. For example, if Jones on a remote node has a proxy account that maps into Smith's account at

local node BOSTON, Jones can copy Smith's file TEST.DAT over the network by specifying the following command:

```
$ COPY BOSTON::WORK1:[SMITH]TEST.DAT *
```

The system manager controls the use of proxy accounts at the local node. Use the Authorize Utility to create and modify the permanent proxy database, NETPROXY.DAT, at your node. Each NETPROXY.DAT entry can map a single remote user to multiple proxy accounts on the local node (one default proxy account and up to 15 additional proxy accounts). The proxy database entry identifies the user by *nodename::username* or *nodename::(group,member)*. (For information on using the Authorize Utility, refer to the *VMS Authorize Utility Manual*.)

For example, to create a NETPROXY.DAT file at local node BOSTON and add a default proxy entry mapping user MARTIN on remote node MIAMI to user ALLEN at the local node, enter the following commands:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF> CREATE/PROXY
UAF> ADD/PROXY MIAMI::MARTIN ALLEN/DEFAULT
UAF> EXIT
```

When DECnet is started up, the information in NETPROXY.DAT is used to construct a volatile proxy database. If changes are made to the permanent proxy database by means of the Authorize Utility, the volatile proxy database is updated automatically.

Similarly, the system manager at a remote node can create and maintain a proxy database of network users having proxy access to specific accounts on that node.

Controlling proxy login access. For proxy login to be successful, one node must be able to initiate proxy login access and the other node must allow proxy login access. To control proxy login for your local (executor) node, use Network Control Program commands to modify the proxy parameters in the executor and object databases. The NCP parameters that specify whether a node can initiate proxy login are the outgoing proxy parameters; the parameters that specify whether a node allows proxy login access are the incoming proxy parameters. By default, both the local node and the remote node can initiate proxy login and allow proxy access.

Defaults for DIGITAL-supplied objects are set in the object database. For example, the object MAIL has outgoing proxy access set by default. To specify or modify the proxy parameter for a network object, use the NCP command SET OBJECT. Use this command to permit outgoing proxy access for a network object:

```
$ RUN SYS$SYSTEM:NCP
NCP>SET OBJECT object-name PROXY OUTGOING
NCP>EXIT
```

Proxy can be disabled on an object-by-object basis by setting the proxy parameter for a particular object to NONE.

To restrict incoming and/or outgoing proxy login access at your local node, specify the NCP commands SET EXECUTOR and SET OBJECT with the appropriate proxy parameters. The proxy setting for the object overrides the executor proxy setting. For example, if incoming proxy login access to your local node is disabled but incoming proxy access to a specific object on the local node is permitted, connection to the object through a proxy account is

allowed. The following commands indicate that proxy access to the object represented by *object-name* is permitted :

```
$ RUN SYS$SYSTEM:NCP
NCP>SET EXECUTOR INCOMING PROXY DISABLED
NCP>SET OBJECT object-name PROXY INCOMING
NCP>EXIT
```

See the *Guide to VMS System Security* and the *VMS Authorize Utility Manual* for more information on establishing proxy accounts and the *VMS Networking Manual* for information on using proxies.

3.3.7.2 Controlling Access to Your Node

In general, the system manager can control access to the local node at three levels:

- **Circuit-level access control:** For point-to-point connections, especially over dialup lines, you can use passwords to verify that the initiating node is authorized to form a connection with your node. Passwords are usually optional for point-to-point connections but are required for dynamic asynchronous connections.

  Each end of a point-to-point circuit can establish a password to transmit to the other node, and specify a password expected from the other node. Before the link is established, each node verifies that it received the expected password from the other node.

  Added security is provided for a dynamic asynchronous connection (which is normally maintained only for the duration of a telephone call): the node requesting the dynamic connection is required to supply a password, but the node receiving the login request is prevented from revealing a password to the requesting node. The steps needed to use passwords in establishing asynchronous DECnet connections are given in Section 3.3.3.

- **Node-level access control:** To control the establishment of logical links with remote nodes, you can specify in your network database access control parameters that indicate which of the following logical link connections are permitted: INCOMING, OUTGOING, BOTH, or NONE. Use the NCP commands that follow to specify access parameters for a specific node, and the executor parameter DEFAULT ACCESS that applies to any node for which a specific access parameter is not specified:

  ```
  $ RUN SYS$SYSTEM:NCP
  NCP>DEFINE NODE node-id ACCESS option
  NCP>DEFINE EXECUTOR DEFAULT ACCESS option
  NCP>EXIT
  ```

- **System-level access control:** When a remote user requests access to the system, the following means of authorization are checked:

  - Is an explicit access control string available?
  - Does the user have a proxy account on the local node?
  - Is there a default nonprivileged DECnet account at the local node?

  If no explicit access control information or proxy account is available, DECnet–VAX will attempt to use a default nonprivileged DECnet account to access the system. The default DECnet account allows users to perform certain network operations, such as the exchange of electronic mail between users on different nodes, without having to supply a name and

password. The default DECnet account is also used for file operations when an access control string is not supplied. For example, it permits remote users to access local files on which the file protection has been set to allow WORLD access. If you do not want remote users accessing your node, do not create a default DECnet account.

You can request the DECnet–VAX configuration command procedure, NETCONFIG.COM, to establish the default nonprivileged DECnet account and directory for you automatically (see Section 3.3.2.2) or you can establish the account and directory manually, as indicated below:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF>ADD NETNONPRIV/PASSWORD=NONPRIV/DEVICE=device-name:-
_ /DIRECTORY=[NETUSER]/UIC=[200,200]/PRIVILEGE=(TMPMBX,NETMBX)-
_ /FLAGS=(CAPTIVE)/NOBATCH/NOINTERACTIVE/LGICMD=NL:
UAF>EXIT
$ CREATE/DIRECTORY device-name:[NETUSER]/OWNER_UIC=[200,200]
```

Device-name is the name of the device on which you have your directory. (The Authorize Utility is described in the *VMS Authorize Utility Manual*.)

If a remote node requests access to an object on the local node but does not supply access control information, any access control information specified for the object in the local network database will be used.

This section is a brief summary of some of the access controls available over the network. For a complete description of DECnet–VAX access controls, see the *VMS Networking Manual*. See also the *Guide to VMS System Security* for additional suggestions on protecting your system.

3.4 Networkwide Considerations

To create a network, the managers of the individual systems involved are responsible for connecting the systems by means of communications lines. The system managers must then configure their nodes and start the network on their systems. For a large network, one manager, called the network manager, may serve as the central focus for the development and establishment of the network.

The initial step in creating a new network is planning the network configuration. The system managers or the network manager, if one is designated, must design the network topology: the arrangement and relationship of nodes, circuits and lines. Some of the decisions involved in designing a large network are as follows:

- Will the network be divided into areas? If so, which nodes are to be in each area?
- Which nodes will be routers and which will be end nodes?
- What will the unique node address of each node be?
- What is the maximum node address possible in the network?

- How will the nodes be connected? What paths will be followed? What lines and circuits will be used? What costs will be assigned to the various circuits? Will satellite links be used?
- If the network includes a VAXcluster, will the nodes in the cluster share a cluster alias (a special node identifier by which the nodes in a VAXcluster can optionally be accessed)?

In addition, the manager of a large network should define the values of NCP parameters that must be the same for all nodes in the network. The network manager should establish uniform values for data transmission and routing control parameters. Routing control parameters control the path that the data being transmitted is likely to take through the network, and are used to minimize traffic congestion at particular nodes in the network. The manager should also define network logging to provide for adequate monitoring and control of network operation.

The network manager should work with the system managers of individual nodes in the network to ensure that all nodes are properly tuned for the network environment. For example, the manager should determine if certain critical routing nodes in large networks need to make adjustments to system parameters to accommodate networking software. Resources that may need adjustment include memory and processor priority.

For a complete description of the process of developing a network, refer to the *VMS Networking Manual.*

4 Keeping the Network Running

Once you have brought up your system as a network node, you can use a variety of software techniques to monitor and test the network. You can also use troubleshooting techniques to resolve problems related to keeping the network running. The tools you can use to monitor the network and the types of tests you can perform on the network are summarized below. Common problems encountered during network operation are indicated, along with advice on troubleshooting. (Refer to the *VMS Networking Manual* for full information on maintaining, controlling and testing the network.)

4.1 Monitoring the Network

You can monitor network activity using software tools. Analyzing the information you collect can help you to determine whether the network is running properly or whether any changes are required to resolve problems or improve performance. Major network monitoring tools include the following:

- NCP display commands you can use to determine the status and characteristics of components in the network. (See Section 4.1.1.)
- NCP counters you can read to obtain error and performance statistics on current network operations. (See Section 4.1.2.)
- Network events logged by DECnet that can be reported to you as they happen. (See Section 4.1.3.)
- Other software tools, such as the Ethernet configurator and the DECnet Test Sender/DECnet Test Receiver (DTS/DTR) Utility, that permit you to learn more about network operation. (See Section 4.1.4.)

4.1.1 Using NCP to Display Information About Network Components

You can use the NCP commands SHOW and LIST to monitor network activity by displaying the following:

- Information about the current condition of network components (using SHOW commands) and the startup values assigned to the components (using LIST commands)
- Counter information about circuits, lines, remote nodes, and the local node (using SHOW COUNTER commands)
- Information about the range of network events being logged by the DECnet event logging facility (using SHOW LOGGING commands)

You do not need any privileges to issue SHOW commands, but you need the privilege SYSPRV to issue LIST commands.

Keeping the Network Running

4.1 Monitoring the Network

Use the SHOW command to monitor the operation of the running network. You can display the characteristics and current status of network circuits, lines and nodes, including the local (executor) node. This information is useful in detecting any changes in the network configuration or operation. For example, if a circuit failure causes some nodes to become unreachable, you can use SHOW commands to check the status of the circuit and the nodes.

In general, the SHOW and LIST commands permit you to indicate what type of information you want NCP to display about the particular component you specify. The display types include the following:

- CHARACTERISTICS: Static information that does not normally change during network operations (for example, the identification of the local node and the circuits connected to the local node, and relevant routing parameters such as circuit cost).
- STATUS: Dynamic information that usually indicates network operation for the running network (for example, the operational state of the local node, circuits, lines and remote nodes).
- SUMMARY: Only the most useful information from both static and dynamic sources; usually a condensed list of information provided for the CHARACTERISTICS and STATUS display types. SUMMARY is the default if you do not specify a display type.
- COUNTERS: Counter information about circuits, lines, remote nodes, and the local node. (See Section 4.1.2 for a description of the use of counters.)
- EVENTS: Information about which network events are currently being logged to which logging collection point. (See Section 4.1.3 for a description of the use of network events.)

When you display information about network components, you can specify either the singular or plural form of the component in the NCP command. Plural forms of component names are KNOWN (all components available to the local node), ACTIVE (all circuits, lines and logging not in the OFF state), and ADJACENT (all nodes directly connected to the local node).

Typical examples of NCP display commands follow:

```
$ RUN SYS$SYSTEM:NCP
NCP>SHOW EXECUTOR CHARACTERISTICS
NCP>SHOW KNOWN LINES STATUS
NCP>SHOW ACTIVE CIRCUITS
NCP>SHOW ADJACENT NODES STATUS
NCP>LIST KNOWN NODES
NCP>EXIT
```

All NCP display commands optionally allow you to direct the information displayed to an output file you specify. For example:

```
$ RUN SYS$SYSTEM:NCP
NCP>SHOW KNOWN NODES TO NODES.DAT
NCP>EXIT
```

This command creates the file NODES.DAT that contains summary information on all nodes known to the local system. If the specified output file exists, NCP appends the display information to that file. The default file type is LIS and the default output file is SYS$OUTPUT.

You can display information about network components on remote nodes using the TELL prefix in the NCP command. The format of the command is TELL *node-id* SHOW *component*. For example, to look at remote node counters, specify the following command sequence:

```
$ RUN SYS$SYSTEM:NCP
NCP>TELL node-id SHOW EXECUTOR COUNTERS
NCP>EXIT
```

4.1.2 Using NCP Counters

You can use NCP commands to display error and performance statistics about network components at any time while the network is running. DECnet software uses counters to collect statistics for the executor node, remote nodes, circuits and lines automatically. To display the contents of counters, use NCP SHOW COUNTER commands, as in the following typical examples of the commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>SHOW EXECUTOR COUNTERS
NCP>SHOW NODE node-id COUNTERS
NCP>SHOW KNOWN CIRCUITS COUNTERS
NCP>SHOW KNOWN LINES COUNTERS
NCP>SHOW LINE line-id COUNTERS
NCP>EXIT
```

For the local node and remote nodes, counter statistics cover such subjects as connection requests, user data traffic, timeouts, and errors. Circuit counters cover such topics as the transmission of data packets over the circuit, timeouts, and errors. Line counters cover such information as the transmission of bytes and data blocks over the line and relevant errors. For a complete summary description of all network counters, including the probable causes of particular types of occurrences, refer to the *VMS Network Control Program Manual.*

Use NCP commands to control counter usage. You may want to reset counters to zero if you are establishing a controlled environment for test purposes. To reset counters to zero, use the NCP command ZERO COUNTERS (the ZERO command requires the OPER privilege). You can zero counters for the executor node and individual nodes, circuits and lines, or all nodes, circuits and lines. In the examples of typical commands that follow, note that the word COUNTERS is optional:

```
$ RUN SYS$SYSTEM:NCP
NCP>ZERO EXECUTOR COUNTERS
NCP>ZERO NODE node-id
NCP>ZERO KNOWN CIRCUITS
NCP>ZERO LINE line-id COUNTERS
NCP>EXIT
```

You can regulate the frequency with which specific counters are logged by issuing the following command sequence:

```
$ RUN SYS$SYSTEM:NCP
NCP>SET component COUNTER TIMER nn
NCP>EXIT
```

The variable *nn* is in seconds. Expiration of the counter timer causes the contents of the counter to be logged and the counter reset to zero. For example, use the following commands to cause a node counter logging event to occur every 600 seconds for the local node:

```
$ RUN SYS$SYSTEM:NCP
NCP>SET EXECUTOR COUNTER TIMER 600
NCP>EXIT
```

4.1.3 Using DECnet Event Logging

Use the DECnet event logging facility to monitor significant network events, such as circuit failures or lost packets, on a continuous basis. Whenever a network error or other meaningful event occurs, the DECnet event logger will log an event message to a terminal or file that you specify. Examples of network events that are logged as they happen include the following:

- Changes in circuit and line states (for example, a circuit failure)
- A node becoming reachable or unreachable
- Circuit and node counter values, logged before the counter is automatically reset to zero
- Errors in data transmission
- Use of invalid data link passwords

Collection and analysis of network events can provide insight into why a particular error condition exists or why network performance may vary.

As events are detected, the event logger sends them to a collection point for analysis. Collection points are called *logging sinks*; they can be located on the local node or at a remote node. Event data can go to one or more sinks. Each of the following types of event sinks handles event data in a slightly different way:

- **Logging monitor.** A program that receives and processes events. Events sent to the logging monitor are displayed on the screen of any terminal declaring itself a "network operator" by means of the Operator Communication (OPCOM) facility. Directing events to the OPCOM terminal is very useful for applications where the operator needs to know what is happening on the network as it happens. For example, it may be useful to know that a circuit is going down as it happens.

  The automatic configuration command procedure (NETCONFIG.COM, described in Chapter 3) enables the logging monitor. The OPCOM process is started when the command procedure SYS$MANAGER:SYSTARTUP_V5.COM is executed. You can enable a terminal as a network operator terminal by specifying the DCL command REPLY/ENABLE=NETWORK. Usually the operator console (OPA0) is one of the OPCOM terminals.

- **Logging console.** A terminal or file that receives events in a readable format. The default logging console is the operator console.

- **Logging file.** A user-specified file that receives events in binary format, possibly for later analysis.

In order for logging to occur at your node, logging must be enabled and the events to be logged must be identified. If you use the automatic configuration command procedure, NETCONFIG.COM, logging will be established automatically. Otherwise, you can use the NCP command SET or DEFINE LOGGING to set the logging sink state to be ON. To identify a remote location for a logging sink, specify the SINK *node-id* parameter in the command. Use one or more separate commands to define the events to be logged. For example, issue the following commands to cause all network events to be logged to OPCOM and displayed at your network operator terminal:

```
$ RUN SYS$SYSTEM:NCP
NCP>SET LOGGING MONITOR STATE ON
NCP>SET LOGGING MONITOR KNOWN EVENTS
NCP>EXIT
```

Alternatively, for each event class, you can specify the specific events to be logged, as follows:

```
$ RUN SYS$SYSTEM:NCP
NCP>SET KNOWN LOGGING EVENTS event-list
NCP>EXIT
```

Events in the event list are identified by class and type (in the form *class.type*). An event *class* refers to the DECnet software functional layer in which the event occurred. (DECnet layers are described in Chapter 1 and summarized in Section 4.3.2.1.) Event classes logged by DECnet include those listed in Table 4–1. The event *type* is a decimal number representing a unique event within the class. You can use the asterisk (*) wildcard character for event types, and you can specify a single event type or a range of types.

Table 4–1 DECnet Event Classes

Event Class	DECnet Functional Layer
0	Network Management
1	Application
2	Session Control
3	End Communication
4	Routing
5	Data Link
6	Physical Link
7	X.25 packet-level events
128–159	VMS system-specific

An example of the command to specify event types 5 through 7 in event class 4 is as follows:

```
$ RUN SYS$SYSTEM:NCP
NCP>DEFINE LOGGING MONITOR EVENTS 4.5-7
NCP>EXIT
```

Additional examples of commands to define logging events are included in the NETCONFIG.COM example shown in Example 3–1. A complete description of network events, including explanations of probable causes of particular events, is given in the *VMS Network Control Program Manual.*

The event message displayed by OPCOM is in the following form:

```
EVENT TYPE  class.type, event-text
From node-address (node name) Occurred (date and time)
component type and identifier
descriptive text
```

An example of a network event message display on the operator terminal at node RED is as follows:

```
%OPCOM, 29-APR-1988, 11:10:09.54, message from user DECnet
DECnet event 4.14, node reachability change
From node 2.5 (RED), 29-APR-1988 11:10:05.16
Node 2.4 (YELLOW), Reachable
```

You can use the SHOW LOGGING command to learn what logging is being performed. For example, to display complete information on all logging being conducted at all nodes, use these commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>SHOW ACTIVE LOGGING KNOWN SINKS
NCP>EXIT
```

To stop monitoring at the network operator terminal temporarily, issue the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>SET LOGGING MONITOR STATE OFF
NCP>CLEAR LOGGING MONITOR ALL
NCP>EXIT
```

Then issue these commands to turn monitoring back on:

```
$ RUN SYS$SYSTEM:NCP
NCP>SET LOGGING MONITOR STATE ON
NCP>EXIT
```

To disable logging at the network operator terminal permanently, enter the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>PURGE LOGGING MONITOR ALL
NCP>EXIT
```

4.1.4 Other Monitoring Tools

Additional software tools are available to view network activity or exercise network operations. The following is a brief list of some of these tools:

- **Ethernet Configurator**. The NCP Ethernet configurator module permits you to obtain a list of all systems on an Ethernet circuit or circuits. The configurator runs as a separate process available to all users on the system once it has been started. To obtain information on the current configuration of nodes on Ethernet circuits, specify the NCP command SHOW MODULE CONFIGURATOR. To start or stop the configurator, specify the SURVEILLANCE ENABLED or DISABLED parameter in the SET MODULE CONFIGURATOR command. (The Ethernet configurator is described in the *VMS Networking Manual*.)

- **DTS/DTR**. The DECnet Test Sender and DECnet Test Receiver programs are cooperating tasks that perform various functions to exercise network task-to-task capabilities. (See the *VMS DECnet Test Sender/DECnet Test Receiver Utility Manual* for a complete description of the DTS/DTR Utility.)
- **VMS Monitor Utility**. You can use the Monitor Utility to monitor DECnet, displaying information about the use of system resources. If you issue the DCL command MONITOR DECNET, the display shows statistics on current DECnet activity at the local node. The information can be of value in determining how much of the system's resources is being used for networking activity. (See the *VMS Monitor Utility Manual.*)

4.2 Testing the Network

You can use the NCP Utility to perform a series of tests to help determine whether the network is operating properly. The tests exercise hardware and networking software at various functional layers. Each test repeatedly sends data through various network components which return the data to its source. These tests are called loopback tests, because the data is looped through a loopback mirror (that is, a cooperating process or device that returns data to the originator).

If messages cannot be looped successfully, or if the data is returned in a corrupted state, an NCP display indicates that the test failed, and includes the reason for the failure and the number of messages not looped.

Loopback tests check the operation of DECnet software at various functional layers. DECnet–VAX software functional layers are indicated in the section on troubleshooting the network. (See Section 4.3.2.1.) DECnet architectural design is described in Chapter 1. For a complete description of network testing, refer to the *VMS Networking Manual.*

Loopback tests can be performed at two levels:

- Node-level loopback tests check the operation of logical links, routing, and other software.
- Circuit-level loopback tests evaluate the operation of circuits. (Note that loopback tests cannot be performed on asynchronous circuits or lines.)

The types of tests and the commands used in each test are summarized briefly below.

4.2.1 Node-Level Loopback Tests

Node-level tests determine whether your DECnet–VAX software can form logical links and exchange data with a DECnet process on a remote node or on your local node. If no error messages are generated, the test is successful. If a test fails, you can perform the next test in sequence.

1 **Remote loopback test:** Verifies local and remote node network operation up to the User layer on the remote node. Loops information to a loopback mirror process on a remote node. The COUNT parameter in the LOOP NODE command specifies the number of messages the test will attempt to send. Use the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>SET LINE line-id STATE ON
NCP>SET CIRCUIT circuit-id STATE ON
NCP>LOOP NODE remote-node COUNT n
NCP>EXIT
```

2 **Local-to-remote loopback test using a loop node name:** Verifies network operation of the local and adjacent nodes over a logical link path on a circuit that you specify. This test checks the operation of software on the local and remote nodes up to and including the Routing layer. Specify a *loop node* name (such as TESTER) to loop information to the local node and a remote node. The loop node name does not identify an actual node; it is a special loopback node name associated with a specific circuit. When DECnet recognizes the special node name, it transmits test data over the circuit associated with the name.

To perform this test, use the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>SET LINE line-id STATE ON
NCP>SET NODE TESTER CIRCUIT circuit-id
NCP>SET CIRCUIT circuit-id STATE ON
NCP>LOOP NODE TESTER COUNT 10
NCP>EXIT
```

3 **Local-to-local loopback test using a loop node name:** Verifies Routing and Data Link layer software at the local node. To start this test, turn off the line, set the device controller to loopback mode, specify a loop node name (TESTER), and turn the line on. Loop the test messages over the circuit associated with the test node. The LENGTH parameter specifies the length of each block to be sent. Use the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>SET LINE line-id STATE OFF
NCP>SET CIRCUIT circuit-id STATE OFF
NCP>SET LINE line-id CONTROLLER LOOPBACK
NCP>SET LINE line-id STATE ON
NCP>SET CIRCUIT line-id STATE ON
NCP>SET NODE TESTER CIRCUIT circuit-id
NCP>LOOP NODE TESTER COUNT 10 LENGTH 32
NCP>EXIT
```

4 **Local-to-local loopback test:** Verifies the operation of local (executor) node software from the User layer to the Routing layer. Test local DECnet software by looping messages to the loopback mirror on the local node using an internal logical link path. Use the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>LOOP EXECUTOR COUNT 10
NCP>EXIT
```

If the node-level tests are successful, return the line and circuit to the working state (as described at the end of the next section). Otherwise, continue with the circuit-level tests.

4.2.2 Circuit-Level Tests

If you are unable to perform a loopback test at the node level, perform circuit-level tests to determine whether the circuit is functioning properly. These tests loop test data through a remote node or through a hardware loopback device on the circuit (called a controller loopback device), depending on the test. (Loopback tests cannot be performed on asynchronous circuits or lines.)

1 **Software loopback test on circuit:** Loop data to an adjacent node on the circuit to test whether the circuit is operational up to the adjacent node's controller. Loop data through the circuit to the adjacent node and back to the local node. Use the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>SET LINE line-id STATE OFF
NCP>SET CIRCUIT circuit-id STATE OFF
NCP>SET LINE line-id CONTROLLER NORMAL
NCP>SET LINE line-id STATE ON
NCP>SET CIRCUIT circuit-id STATE ON
NCP>LOOP CIRCUIT circuit-id COUNT n
NCP>EXIT
```

For an Ethernet circuit, the PHYSICAL ADDRESS parameter must be specified in the LOOP CIRCUIT command.

2 **Controller loopback tests:** If you are unable to communicate with an adjacent node, perform a controller loopback test to determine whether the communications controller on your system is functioning properly. Loop the data to the device on the circuit; the device must be in loopback mode to determine whether the controller works properly. Controller loopback tests are performed differently for point-to-point synchronous circuits and Ethernet multiaccess circuits:

a. **Controller loopback test for non-Ethernet circuit:** Set the communications controller to loopback mode and make the controller loop network messages back to your machine. Use the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>SET LINE line-id STATE OFF
NCP>SET CIRCUIT circuit-id STATE OFF
NCP>SET LINE line-id CONTROLLER LOOPBACK
NCP>SET LINE line-id STATE ON
NCP>SET CIRCUIT line-id STATE ON
NCP>LOOP CIRCUIT circuit-id COUNT 10 LENGTH 32
NCP>EXIT
```

b. **Controller loopback test for Ethernet circuit:** You can test an Ethernet circuit by looping data to the Ethernet interface of a remote node, rather than to the remote node itself. You can run the test concurrent with other DECnet operations on the circuit. The Ethernet

circuit must be on and the SERVICE parameter must be enabled on the circuit. You can test the status of the circuit by causing the circuit to loop messages to itself. Use the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP>SET LINE line-id STATE OFF
NCP>SET CIRCUIT circuit-id STATE OFF
NCP>SET CIRCUIT circuit-id STATE ON SERVICE ENABLED
NCP>SET LINE line-id CONTROLLER LOOPBACK STATE ON
NCP>SET NODE TESTER CIRCUIT circuit-id
NCP>LOOP NODE TESTER
NCP>EXIT
```

After the above tests are completed, you should set the line and circuit back to the working state, as follows:

```
$ RUN SYS$SYSTEM:NCP
NCP>CLEAR LINE line-id
NCP>CLEAR CIRCUIT circuit-id
NCP>SET LINE line-id STATE ON
NCP>SET CIRCUIT circuit-id STATE ON
NCP>EXIT
```

4.3 Common Problems Encountered on the Network

Once you have brought up your system as a network node, you may receive messages related to networking errors. Other problems that can occur at any time during network operation may not result in messages being displayed. This section explains the causes of error messages that may be displayed, suggests troubleshooting techniques that you can use to isolate and identify common network problems, and describes the problems that you may experience in establishing asynchronous connections.

4.3.1 Common Error Messages and Meanings

When you are using DECnet–VAX, you may receive network-related messages indicating software or hardware problems, transient conditions, or errors in your input. The following list displays some common network-related messages, explains what condition may be causing each message, and suggests actions you can take. (All messages displayed on a VMS system, including those generated by DECnet–VAX, are described in the *VMS System Messages and Recovery Procedures Reference Volume*.)

- **NCP-I-INVPVA, invalid parameter value**

  This message is displayed if you specify a parameter value in an NCP command that is not a valid value for the specified parameter. For example, the following command generates this error message:

  ```
  NCP>SET LINE UNA-0 PROTOCOL DDCMP POINT
  _%NCP-I-INVPVA, Invalid parameter value, Protocol
  NCP>
  ```

  The value for the indicated parameter is invalid because the UNA device uses the ETHERNET protocol. The name of the parameter for which the value was invalid is displayed at the end of the error message. Reissue the command with the correct value for the parameter.

- **SYSTEM-I-LINKEXIT, network partner exited**

  This message is displayed if the process on the remote node exited before confirming the logical link to your node. The remote process might have exited prematurely, a timeout may have occurred at the remote node, or there may be a problem in the log file on the remote node. You could either retry the operation or try to read the NETSERVER.LOG file in the directory of the account you are attempting to access at the remote node. (DECnet–VAX automatically creates a NETSERVER.LOG file and places it in the directory for the appropriate account when it receives a connect request.)

- **SYSTEM-F-UNREACHABLE, remote node is not currently reachable**

  This message is displayed when you attempt to connect to a node that is unreachable. For example, when you address a mail message to a user at remote node PURPLE using the Mail Utility, if MAIL cannot create a link to the remote node, you may receive the following message:

  ```
  %MAIL-E-LOGLINK, error creating network link to node PURPLE
  _SYSTEM-F-UNREACHABLE, remote node is not currently reachable
  ```

  You can try to access the remote node again at a later time.

  The message is also displayed even if the remote node does not exist, as long as you have indicated a node address or a node name that you previously defined in your node database.

  You also receive notice that the node is unreachable if the value of the executor parameter MAXIMUM ADDRESS in your network database is lower than the address of the remote node you are attempting to access. Increase the value of the NCP executor parameter MAXIMUM ADDRESS in your database to be at least as high as the highest address of any node that you wish to contact.

- **SYSTEM-F-INVLOGIN, login information invalid at remote node**

  You receive this message if you attempt to access a remote node using an access control string that contains an invalid user name or password, or if you do not specify any access control information and no default DECnet account or proxy account is available at the remote node.

  For example, if you enter this command specifying an invalid user password in the access control string that is part of the file specification, you receive this error message:

  ```
  $ DIRECTORY BOSTON"SMITH GHIJKL"::WORK1:[SMITH]
  %SYSTEM-F-INVLOGIN, login information invalid at remote node
  ```

  Retry the file operation with the correct login information.

- **SYSTEM-F-NOSUCHNODE, remote node is unknown**

  You receive this message if you attempt to issue a command to access a remote node (for example, the DCL command SET HOST) and the remote node represented by **node-id** is not identified in the local volatile database. Verify that the node identifier is correct, enter the node name in your node database, and retry the operation.

- **SYSTEM-F-PATHLOST, path to network partner lost**

  You receive this message if you logged in to another node over the network (for example, using the DCL command SET HOST) and the path to the remote node is lost.

  The path may be lost because of too much network activity or communications problems, or because DECnet was turned off at the remote node. Wait, then check to see if the node is still reachable. If so, try again to log in.

- **SYSTEM-F-SHUT, remote node no longer accepting connects**

  You receive this message if you attempt to access the remote node using a DCL command (such as the SET HOST command) under either of these conditions:

  a. The executor parameter DEFAULT ACCESS on the remote node has been set to NONE. The default access at the remote node must be set to permit incoming and outgoing access before you can connect to the node.

  b. The command SET EXECUTOR STATE SHUT was executed on the remote system. The network must be restarted on the remote node.

- **SYSTEM-F-NOLINKS, maximum network logical links exceeded**

  This message appears if the maximum number of links that the remote node allows has been exceeded. Wait and try again later.

- **SYSTEM-F-NOSUCHOBJ, network object unknown at remote node**

  You receive this message if you attempt to access a network object at a remote node and the object is not specified in the remote node database. For example, if you attempt to use the Phone Utility to reach a node that does not have an entry for the network object PHONE in its configuration database, you receive the above message.

4.3.2 Problems Related to Network Operation

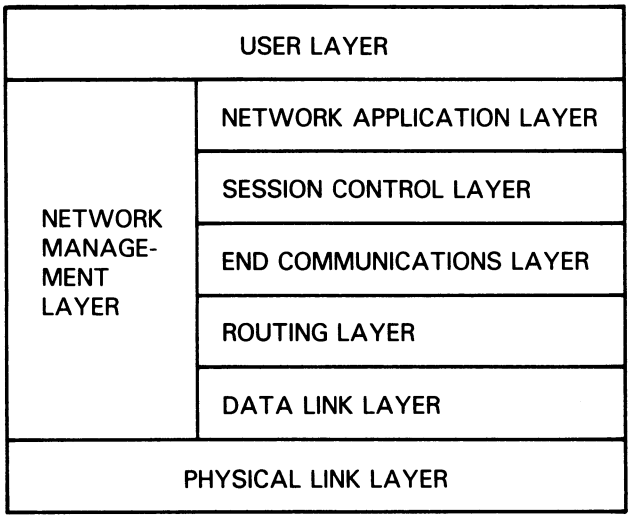
Problems in maintaining the proper functioning of the running network can be difficult to resolve. This section describes the technique for isolating a problem to a particular DECnet software functional layer or layers, and provides troubleshooting suggestions to determine the specific network problem. As system manager of the local node, you may want to consult with the network manager (if one is available for your network) as necessary to resolve these problems. (Refer to the *VMS Networking Manual* for full information on maintaining a DECnet–VAX node.)

4.3.2.1 Troubleshooting Techniques Based On DNA Layers

Techniques for troubleshooting DECnet–VAX problems are based on the layered network design of DECnet–VAX as specified in the DIGITAL Network Architecture (DNA). The DNA layers are illustrated in Figure 4–1. Each layer performs particular services as part of the overall network capability provided at the node. (For more information on DECnet–VAX software design, see Chapter 1.)

During troubleshooting, it is useful to distinguish among the network layers in localizing the cause of a particular problem. For example, some problems are characteristic of the Data Link layer, while others are related to the Routing layer or to the End Communications layer (which provides logical link services).

Figure 4–1 DECnet–VAX Software Design as Based On DNA Layers



4.3.2.2 Network Problems and Suggested Actions

The following discussion of network difficulties identifies typical problems originating at the various layers, and actions you can take to locate the source of the problem. The problems are grouped into those related to data links, routing, and logical links.

Data link problems. Inability to reach an active node is a common problem on the network. The problem could be either a data link problem or a routing problem.

To determine whether the problem is a data link problem, examine both the remote node and the circuit. The data link layer causes data to be moved over physical devices, so it affects only adjacent nodes (an adjacent node is connected to the local node by a single physical line). You can learn whether the unreachable node is an adjacent node and whether the node is available by checking with the network manager or the system manager of the unreachable node.

Also check the state of the circuits (the data link protocol causes a circuit to be in the ON-SYNCHRONIZING state). The problem may be with the data link if the circuit does not start up correctly or is up but the adjacent node is not reachable. (Note that circuit startup may also be affected by incorrect setting of the transmit and receive passwords, as described in the following section on routing problems.)

To locate a data link problem, examine the appropriate counters, line and circuit parameters, and network events.

- Use NCP SHOW commands to display the contents of the circuit and line counters to see if they are reporting errors. (See Section 4.1.2 for an explanation of how to use NCP counters. Descriptions of all counters, including probable causes of particular types of occurrences, appear in the *VMS Network Control Program Manual.*)
- Use NCP SHOW commands to check the values of line and circuit parameters in the network configuration database.
- Then look at the network events DECnet logged for event class 4 (for the Routing layer) and event class 5 (for the Data Link layer) to determine whether any events affecting the data link have occurred. (See Section 4.1.3 for a description of how to use DECnet event logging. Network events are summarized by event class and type in the *VMS Network Control Program Manual;* explanations of probable causes of network events are included as applicable.)

Routing problems. Routing layer problems can involve nodes that are not reachable or circuits that are not stable. The circuit may be up and the adjacent node may be reachable, but one or more intermediate nodes (along the communications path) that should be reachable are not.

To isolate such Routing layer problems, examine the appropriate counters and passwords, and try to check the nodes along the routing path.

- Check the contents of the node and circuit counters at your node and, if possible, arrange for the node and circuit counters at the remote node to be examined.
- Examine network events logged for event class 4 (for the Routing layer).
- Check the settings of the transmit and receive passwords for the local node and the adjacent node to see if they match (these passwords affect circuit startup).
- Finally, you can use NCP commands with the TELL prefix to try to trace the routing path from one node to another, to determine if an intermediate node is down and to examine the parameter values for all nodes on the communications path. (See Section 4.1.1 for an explanation of how to use the TELL prefix in an NCP command.)

If erratic routing behavior occurs (for example, constant changes in the reachability of nodes, or connection to a node other than the one you expect to reach), check whether two or more nodes with the same node address are connected to the network. If routing seems to be functioning properly, you can look at executor parameters related to routing (such as cost and hops). For additional information on routing layer parameters, refer to the *VMS Networking Manual.*

Logical link problems. The End Communications layer, which provides logical link services, can also be the source of common problems. Typical symptoms of logical link problems include

- Link timeout
- Network partner exited
- Invalid account
- Problems with performance and response time

In general, for logical link problems, you can examine the following:

- The default DECnet nonprivileged account and directory on the remote node, to determine if they have been created properly.
- Incoming and outgoing timers at both ends of the logical link, to ensure the links are not timing out prematurely because the timers are set too low.
- The accounting log (using the VMS Accounting Utility), to determine whether the correct process was created or whether a correct process exited prematurely.
- The load on the local and remote nodes, to determine whether the load is preventing the link from being created.
- The path over the network to the remote node. If the circuit is an Ethernet circuit, check the line buffer size parameter to ensure the proper setting.
- The NETSERVER timeouts, by getting someone to examine the NETSERVER.LOG file at the remote node.
- The proxy settings for your node and for the objects being accessed. (To determine the default proxy access setting for your executor node, specify the NCP command SHOW EXECUTOR CHARACTERISTICS. To examine the proxy access setting for network objects, use the NCP command SHOW KNOWN OBJECTS CHARACTERISTICS.)
- The disk quota, to ensure it is sufficient to create the NETSERVER.LOG file.
- The SYS$LOGIN file, to determine whether the file protection is set to WORLD:READ.

If a logical link connection is unsuccessful, the link may have timed out for one of the following reasons:

- A heavily loaded node can cause creation of a logical link to take a long time.
- Incoming and outgoing timers may be set too low.

To prevent link timeouts, you can reset the executor parameters INCOMING TIMER and OUTGOING TIMER to higher values at both nodes.

A logical link problem may cause the message "network partner exited" to be displayed. This message indicates that the remote node exited before the logical link was established. Check the following:

- The networking load on the nodes at each end of the logical connection
- The accounting log on the remote node
- Netserver timeouts on the remote node

If you receive a message indicating an invalid account, check that you have the proper authorization to make the logical link connection. However, an invalid account condition may also be reported by the message "network partner exited." Consequently you should try to to have someone check the NETSERVER.LOG file on the remote node.

If performance and response time over the logical link become degraded, the cause may be too much traffic on a path to the target node. If you encounter this problem, consult with the network manager.

Configuration problems. The main reason for network errors may be improper configuration of the system. Check your DECnet–VAX configuration, and check the communications cables and connections.

4.3.3 Asynchronous Connection Problems

Attempts to establish asynchronous DECnet connections with other nodes can fail for a variety of reasons. This section describes some reasons why you may fail to make a static or dynamic connection.

4.3.3.1 Problems with Static Asynchronous Connections

A static asychronous connection has failed if the static asynchronous DECnet line is started but remains in the ON-STARTING state. To isolate the cause of the problem, check whether the following conditions exist:

- Are the line speeds at both ends of the connection set to the same value?
- If you are using a dialup line, is the modem characteristic set on the terminal? (This must be done before the line is set to asynchronous DDCMP use.)
- Are the two nodes being connected located in the same area in the network (that is, do their node addresses have the same area number) or are both nodes area routers?
- Is the parity on the asynchronous DECnet line set to NONE? If your system is not a VMS system, is the terminal line set to the correct parity?
- Is the terminal line set up to use 8-bit characters?
- If the node already has an active circuit, is the node a routing node?
- If verification is enabled for the circuit, do the passwords set at the two nodes match?

If you are unsuccessful in setting up your terminal line as a static asynchronous DDCMP line, check the following:

- Is the /NOTYPE_AHEAD qualifier specified for your terminal before you attempt to set up the static asynchronous line? If a type-ahead buffer is associated with your terminal, you may not be able to bring up your terminal line as an asynchronous DECnet line until you terminate any process started at the remote node that may own your terminal line.

4.3.3.2 Problems with Dynamic Asynchronous Connections

If dynamic switching is being performed and the asynchronous DECnet connection is not made, first check the following:

- Is DECnet started on both nodes?
- Is the asynchronous DDCMP class driver (NODRIVER) loaded by means of SYS$SYSTEM:SYSGEN at each node?
- Is the dynamic switch image (DYNSWITCH) installed by means of SYS$SYSTEM:INSTALL at each node?
- Are virtual terminals enabled on the remote node and, in particular, for the terminal over which you are logged in to the remote node?

If the dynamic asynchronous lines are started but are left in the "ON - STARTING" state, make the following checks:

- Are the two nodes that are being connected located in the same area (that is, do their addresses have the same area number) or are they both area routers?
- Are the routing initialization passwords (transmit and receive passwords) set appropriately at each node?
- Is the INBOUND parameter for the initiating node set correctly in the node database at the node receiving the connection request?
- Is the parity on the asynchronous DECnet line set to NONE? If your system is not a VMS system, is the terminal line set to the correct parity?
- Is the terminal line at the remote node set up to use 8-bit characters?
- If the node already has an active circuit, is the node defined as a routing node?

Glossary

access control: Validation of requests for connection, login, or file access, to determine whether they can be accepted. User name and password provide the most common means of access control.

active component: A component whose operational state is other than OFF.

adjacent node: A node connected to the local node by a single physical line.

alias node identifier: An optional node name or address, common to some or all nodes in a VAXcluster, that permits the VAXcluster to be treated as a single node.

area: A group of nodes in a network that can run independently as a subnetwork.

area router: A level 2 router.

area routing: A technique for grouping the nodes in a network into areas for routing purposes. Routing in a multiple-area network is on two levels: one level of routing is within an area (called level 1 routing), and a second, higher level of routing is between areas (called level 2 routing).

asynchronous transmission: A mode of data transmission in which the time intervals between transmitted characters may be of unequal length. Asynchronous transmission most commonly occurs over terminal lines.

channel: A means of transmission of data.

characteristics: A display type for the NCP commmands SHOW and LIST. It refers to static information about a networking component that is kept in the configuration database.

circuit: The communication path between nodes. Circuits operate over physical lines and are the medium on which all input/output occurs.

command node: The node from which an NCP command is issued.

component: An element in the network that can be controlled and monitored. Components include lines, circuits, nodes, modules, logging, and objects. Components form part of the NCP command syntax.

configuration database: The combination of both the permanent and the volatile databases. It consists of information about the local node, and all nodes, modules, circuits, lines, and objects in the network.

cost: A numeric value assigned to a circuit that exists between two adjacent nodes. In the DECnet network, packets are routed on paths with the lowest cost.

counters: Performance and error statistics kept for a component, such as a line or a node.

data link mapping (DLM): Capability of using an X.25 virtual circuit as a DECnet data link.

designated router: A routing node on the Ethernet selected to perform routing services on behalf of end nodes.

distributed processing: The technology that enables the distribution of computing power and storage facilities to user work areas where they are needed.

downline system load: A DECnet–VAX function that allows an unattended target node to receive an operating system file image or terminal server file image from another node.

downline task load: A function that allows a remote target node to receive an RSX–11S task from another node.

end node: A node that can receive packets addressed to it and send packets to other nodes, but cannot route packets through from other nodes. Also called a nonrouting node.

event: A network or system-specific occurrence for which the logging component maintains a record.

event class: A particular category of events. Generally, this classification follows the DNA architectural layers; some layers may contain more than one class. Class also includes the identification of system-specific events.

event type: A particular form of event that is unique within an event class.

executor node: The node at which an NCP command actually executes.

hop: The logical distance between two nodes. One hop is the distance from one node to an adjacent node.

host node: For DECnet, a node that provides services for another node (for example, the host node that supplies program image files for a downline load).

known component: The classification for one or more of the same networking components. This classification includes all active and inactive occurrences of the component type. For example, known nodes include all active and inactive nodes in the network.

level 1 router: A node that can send and receive packets, and can route packets from one node to another, only within a single area of a network.

level 2 router: A node that can send and receive packets, and can route packets from one node to another, within its own area and between areas in a network. Also known as an area router.

line: The network management component that provides a distinct physical data path.

local node: The node at which you are physically located.

logical link: A carrier of a single stream of two-way communications traffic between two user-level processes.

logging: The network management component that routes event data to a logging sink such as a console or file.

logging console: A logging sink that is to receive a human-readable record of events. Typically, a logging console is a terminal or a user-specified file.

logging file: A logging sink that is to receive a machine-readable record of events for later retrieval. The logging file is user defined.

logging monitor: A logging sink that is to receive a machine-readable record of events for possible real-time decision making. Typically, the logging monitor is a user-defined program.

logging sink: The device or process that records the network events logged by the event logger.

loop node: A local node that is associated with a particular line and is treated as if it were a remote node. All traffic to the loop node is sent over the associated line.

modem (modulator-demodulator): A device that modulates signals transmitted over communications circuits.

module: A network management component.

multiaccess channel: A medium (for example, Ethernet) on which many transmitters contend for access.

multipoint circuit: A circuit connecting two systems, with one of the systems (the control station) controlling the circuit, and the other system serving as a tributary.

network connect block (NCB): For DECnet, a user-generated data structure used in a nontransparent task to identify a remote task and optionally send user data in calls to request, accept, or reject a logical link connection.

network task: A nontransparent task that is able to process multiple inbound connection requests; that is, it has declared a network name or object number.

node: A network management component that supports DECnet software.

node address: The required, unique, numeric identification of a specific node in the network.

node name: An optional alphanumeric identification associated with a specific node address. A node name must contain at least one alphabetic character.

nonprivileged: In DECnet–VAX terminology, this term means no privileges in addition to NETMBX and TMPMBX. NETMBX is the minimal requirement for any network activity.

nonrouting node: An end node.

object: A DECnet–VAX process that receives a logical link request. If the object type number is not zero, the object performs a specific network function. If the object type number is zero, the object is usually a user-defined image for a special-purpose application.

packet: A unit of data to be routed from a source node to a destination node.

packet switching: A data transmission process, utilizing addressed packets, whereby a channel is occupied only for the duration of transmission of the packet.

packet switching data network (PSDN): A set of equipment and interconnecting links that provides a packet switching communications service to subscribers.

Packetnet System Interface (PSI): The name for the software product that allows DIGITAL operating systems to participate in a packet switching environment.

parameter: An entry in the volatile or permanent database for a network management component.

path: The route a packet takes from source to destination.

path cost: The sum of the circuit costs along a path between two nodes.

path length: The number of hops along a path between two nodes; that is, the number of circuits a packet must travel along to reach its destination.

permanent database: A file containing information about network management components.

point-to-point circuit: A circuit that connects two nodes, operating over a single line.

privileged: In DECnet–VAX terminology, this term means any user privileges in addition to NETMBX and TMPMBX.

protocol: An agreed set of rules governing the operation of a communications link.

proxy login: The procedure that permits a remote user to access a specific account at the local node, without supplying the user name and password.

reachable node: A node to which the local node has a usable communications path.

remote node: To any one node in the network, this node is any other network node.

router: A node that can send and receive packets and can route packets from one node to another. A router may have more than one active circuit.

routing: The network function that determines the path along which data travels to its destination.

routing node: A router (either a level 1 or a level 2 router).

sink node: A node on which a logging sink, such as a file or console, is actually located.

source task: The task that initiates a logical link connection request in a task-to-task communication environment.

state: The functions that are currently valid for a given component. States include line, circuit, local node, module, and logging.

status: A display type for the NCP commands SHOW and LIST. Status refers to dynamic information about a component that is kept in either the volatile or the permanent database.

summary: The default display type for the NCP commands SHOW and LIST. A summary includes the most useful information for a component, selected from the status and characteristics information.

synchronous transmission: A mode of data transmission in which the time of occurrence of each signal representing a bit is related to a fixed time frame.

target node: The node that receives a memory image during a downline load; a node that loops back a test message.

target task: The task that receives and processes a logical link connection request in a task-to-task communication environment.

task: In this manual, the term refers to an image running in the context of a process.

task specifier: Information provided to DECnet–VAX software so that it can complete a logical link connection to a remote task. This information includes the name of the remote node on which the target task runs and the name of the task itself.

terminal emulator: A program that acts as a transparent interface between two ports, making it appear as though a terminal on the local processor is directly connected to a remote processor.

upline dump: A DECnet–VAX function that allows an adjacent unattended node to dump its memory to a file on a VMS system.

virtual terminal: For VMS, a pseudodevice that connects a process to a physical terminal device. The virtual terminal can be disconnected from the physical terminal and reconnected later.

volatile database: A memory image that contains information about network management components.

X.25: A CCITT–recommended standard for communication between devices which use public carrier networks such as packet switching data networks. CCITT, the Comite Consultatif International Telegraphique et Telephonique, is an international consultative committee that sets international communications usage standards.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Reader's Comments

Guide to DECnet–VAX Networking
AA–LA47A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

_____

What I like best about this manual is _____

_____

_____

What I like least about this manual is _____

_____

_____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

--- 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

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01–3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

--- Do Not Tear - Fold Here ---

Cut Along Dotted Line

Reader's Comments

Guide to DECnet–VAX Networking
AA–LA47A–TE

Please use this postage-paid form to comment on this manual. If you require a written reply to a software problem and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Thank you for your assistance.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy (software works as manual says)	☐	☐	☐	☐
Completeness (enough information)	☐	☐	☐	☐
Clarity (easy to understand)	☐	☐	☐	☐
Organization (structure of subject matter)	☐	☐	☐	☐
Figures (useful)	☐	☐	☐	☐
Examples (useful)	☐	☐	☐	☐
Index (ability to find topic)	☐	☐	☐	☐
Page layout (easy to find information)	☐	☐	☐	☐

I would like to see more/less _____

_____

_____

What I like best about this manual is _____

_____

_____

What I like least about this manual is _____

_____

_____

I found the following errors in this manual:

Page	Description
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Additional comments or suggestions to improve this manual:

_____

_____

_____

_____

I am using **Version** _____ of the software this manual describes.

Name/Title _____ Dept. _____

Company _____ Date _____

Mailing Address _____

_____ Phone _____

-- 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

DIGITAL EQUIPMENT CORPORATION
Corporate User Publications—Spit Brook
ZK01–3/J35 110 SPIT BROOK ROAD
NASHUA, NH 03062-9987

-- Do Not Tear - Fold Here ----------

Cut Along Dotted Line