Digital's Networks: An Architecture With A Future

digital

Digital's Networks:
An Architecture With A Future

digital

Digital Equipment Corporation makes no representation that the interconnection of its products in the manner described herein will not infringe on existing or future patent rights, nor do the descriptions contained herein imply the granting of license to make, use, or sell equipment constructed in accordance with this description.

The information in this book is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The following are trademarks of Digital Equipment Corporation: ALL-IN-ONE, DEC, DECnet, DECsystem-20, DECUS, the Digital logo, Micro/RSX, MicroVAX I, Observer, PACKETNET, PDP, Professional, RSTS/E, RSX, TOPS-20, UNIBUS, VAX, and VT100.

IBM is a registered trademark of International Business Machines.

Table of Contents

Preface ..... *VII*

CHAPTER 1 · Introducing and Defining DECnet and DNA 1
What is DECnet? 1
Uses of a DECnet Network 4
Digital Network Architecture 5
Interfaces 6
Protocols 10
Importance of a Network Architecture 13
DECnet Capabilities 14
Topology Alternatives 16
DECnet Phases 19

CHAPTER 2 · DECnet Concepts 21
Nodes 21
Lines and Circuits 24
Transmission Modes 26
Data Link Protocols 27
DDCMP Fuctional Description 28
DDCMP Messages 31
Data Messages 31
Control Messages 32
Maintenance Messages 34
DDCMP Operation 34
Typical Message Exchange 35
Maintenance Mode 37
Ethernet Functional Description 37
Ethernet Messages 41
Ethernet Operation 44
Routing 47
Routing Terms 49
Routing Algorithms and Databases 52
Routing Features 53
How Routing Works on an Ethernet LAN 55

Routing Messages 56
Routing Operation 59
Data Flow 64
Data Flow at the Source Node 66
Data Flow Across the Network to the Destination Node 68
Data Flow at the Destination Node 69

CHAPTER 3 · DECnet Configurations 71
DECnet Environments 72
Wide Area Networks 72
Local Area Networks 74
Node Characteristics 81
Phase III and Phase IV Nodes 82
Routing Capabilities 82
General Purpose and Dedicated Nodes 84
Line and Circuit Characteristics 88
The Total Picture 89

CHAPTER 4 · Common Mechanisms in DECnet Functions 92
Logical Links 92
When Are Logical Links Required? 93
How to Create a Logical Link 94
Behind the Scenes—DECnet Software and Logical Links 95
End-to-End Communication Layer and NSP 95
Session Control Layer 99
Requesting a Connection 101
Receiving a Connect Request 101
Session Control Messages 103
How the Program Identifies Logical Links 103
Multiple Logical Links Within a Program 104
Data Types Within a Logical Link 105
Segmentation and Reassembly of Data 105
Error Control 106

Flow Control 108
Access Control 109
User Transparency 111

CHAPTER 5 · Task-to-Task Communications 112
DECnet Task-to-Task Calls 112
Requesting a Logical Link 113
Object Types and Names 116
Accepting/Rejecting a Logical Link Request 117
Sending and Receiving Data 119
Terminating the Link 121

CHAPTER 6 · Remote File and Record Access 122
The Data Access Protocol (DAP) Interface 123
Programming Remote File Access 126
File System Capabilities 128
Initiating Remote Access 130
The File Specification 131
Access-Control Information 131
File Characteristics 132
Accessing Remote Files From a Terminal 133
Access Control 134
File Protection 134
Remote File Specifications 135
Remote Command File Submission 135

CHAPTER 7 · Network Terminal Facilities 136
Interactive Terminal-to-Terminal Communcations 136
Network Virtual Terminal Facilities 138
Network Virtual Terminal 139
Network Virtual Terminal Operation 146
Local Area Transport (LAT) Protocol 148
Relationship Between LAT and CTERM 149

CHAPTER 8 · Internetwork Communications 151
X.25 Communicatons 151
DTEs and DCEs 153
X.25 Circuits 153
X.25 Gateway Access Protocol 155
X.25 Gateway Access Modules 157
X.25 Access Methods 165
CCITT Recommendations for X.3, X.28, and X.29 168
DECnet/SNA Communications 169
DECnet/SNA Gateway Access Functions 170
DECnet/SNA Gateway Access Modules 173
DECnet/SNA Circuits 177
Loading the Gateway Node Software 178

CHAPTER 9 · How DECnet Supports Applications ..179
DECnet's Task-to-Task Capability in an Application Environment 181
Task-to-Task Communication: Order Entry and Production Scheduling 182
Network and Application Interaction 183
DECnet's File Transfer Capability in an Application Environment 185
User-Initiated File Transfer: Market Research and Sales Analysis 185
Program-Initiated File Transfer: Quality Control and Vendor Analysis 188
DECnet's Remote File Access Capability in an Application Environment 189
Network Virtual Terminals 189
Short-Term Cash Management 190
DECnet's Terminal Facilities in an Application Environment .191
Order Entry and Production Scheduling 192
Market Research 193
Short-Term Cash Management 194
DECnet as a Consideration in Relocation Planning 196

CHAPTER 10 · Network System Management 198
Network Management Utilities 199
Network Management Messages 201
Planning for Node Generation 205
Generating Network Software 208
Defining Configuration and Other Static Parameters 209
Node Addresses and Names 209
Node Verification Passwords 209
Network Object Parameters 210
Routing Parameters 211
Line Identification 212
Circuit Parameters 212
Transmission Mode 216
Operating a Node 216
Controlling the State of a Node 216
Controlling Line or Circuit State 217
Monitoring Node Activity 218
Downline Loading 220
Downline—Loading Definitions 221
Downline—Loading Database Parameters 221
Performing a Downline Load 222
Downline—Loading and Checkpointing RSX-IIS Tasks 224

CHAPTER 11 · Monitoring and Testing DECnet Performance 228
Integrated Testing Tools 229
Loopback Testing 229
Upline Dump 237
Monitoring Network Operation with Observer 250

Glossary 256
Index 271

PREFACE

Distributed data processing is a computing philosophy Digital introduced and popularized. For over 20 years, Digital has been developing products that bring computing power right into people's work areas, whether these be offices, labs, or factory floors. Digital has made it possible for people to use computing resources from terminals on their desks, rather than depend on a computer in an isolated room. The corporation's wide range of communications products testifies to its commitment to bringing computing power to user locations, where the resources are most needed.

Introduction to DECnet and the Digital Network Architecture (DNA) discusses DNA–the model of structure and function upon which Digital's communications products are based–and the various implementations of DECnet–Digital's family of networking software. This handbook introduces readers to the capabilities of the following new products:

- DECnet-VAX
- DECnet-RSX (RSX-11M, RSX-11M-PLUS, and RSX-11S)
- DECnet-20
- DECnet-10

Since 1975, Digital has been augmenting DNA. The architecture supports a broad range of applications and a variety of network topologies. (A network topology is a particular configuration of nodes and lines.) This handbook summarizes DNA design, structure, and functional specifications. The current DNA version is Phase IV.

The handbook introduces readers to the following new Phase IV concepts and products:

- PRO/DECnet
- Ethernet and related hardware
- DECnet/SNA Gateway (for VAX/VMS and RSX-11 systems)
- DECnet Terminal Server
- DECnet Router Server
- DECnet Router/X.25 Gateway
- Observer

Introduction to DECnet and the Digital Network Architecture (DNA) describes the concepts and capabilities of DECnet networks. A DECnet network consists of two or more Digital computer systems, enhanced with DECnet software and linked by physical channels or communication lines. Systems in a DECnet network can communicate with each other and share resources. Products are also available that enable DECnet systems to communicate with other network entities, such as IBM SNA mainframes and packet-switched data networks (PSDNs). DECnet networks can be local area or wide area configurations or configurations that include wide area and local area networks.

All implementations of DECnet (DECnet VAX, PRO/DECnet, and DECnet-RSX, for example) embody the same network concepts. The specific capabilities of a DECnet network depend on the type of system participating in the network and on the network's application. The objectives of this handbook are:

- To describe the common DNA network concepts behind all implementations of DECnet
- To identify the major network functions that DECnet provides

One of the primary aspects of a DECnet network is the peer relationship among its systems. Network operation is efficient, since any system in the network can initiate and accept communications without the services of a controlling system. This handbook describes the relative ease and speed with which network users can access the resources of any system in a peer-oriented network.

This handbook is intended for readers with a knowledge of communications technology who want to understand the DNA structure and who want to learn about the concepts and capabilities of DECnet systems. It assumes that the reader is familiar with Digital operating system concepts, but not with DECnet. Typical readers will include the personnel at the site of a newly installed DECnet system who can read this manual to learn about the kind of work DECnet enables them to perform. Another group of readers will include system managers and designers who are thinking about using DECnet to expand the capabilities of their existing Digital computer systems. And, finally, the handbook is also intended for system managers and designers who do not yet use Digital systems but who are considering the implementation of a computer network.

Introduction to DECnet and the Digital Network Architecture (DNA) consists of three parts, encompassing eleven chapters:

- The first part, Chapters 1 to 4, introduces DECnet concepts and general configuration guidelines for DECnet networks.
- The second part, Chapters 5 to 9, defines specific network functions and explains the mechanisms that programs and users at terminals can employ to implement those functions. This part of the handbook also presents examples of how these capabilities can be used in an application environment.

- The third part, Chapters 10 and 11, introduces DECnet functions related to the management of the systems that make up a DECnet network.

Chapter 1 · Introducing and Defining DECnet and DNA

This handbook introduces readers to the *Digital Network Architecture (DNA)*–the framework of specifications within which Digital designs its communications products–and to *DECnet*–the family of networking products that Digital has implemented in accordance with DNA specifications. Readers will learn how and where DECnet is used and what capabilities it offers. In addition, they will understand the importance of a network architecture such as DNA.

There are several implementations of DECnet, one for each operating system from Digital. This handbook discusses features and capabilities that are common to all DECnet implementations. For details on specific implementations, refer to the documentation accompanying the implementations.

· What is DECnet?

DECnet is a family of software and hardware communications products that enables Digital operating systems to participate in a network environment. A network is an entity of two or more computer systems that are connected by physical links (cable, microwave, or satellite, for example) for the purpose of exchanging data and sharing resources. Every system in the network is called a node. Figure 1.1 illustrates an example of a DECnet network.

This handbook refers to two types of network users:

(a) individuals who employ the network to perform such high-level functions as accessing a file on a remote node or updating a database on a remote system. In this handbook, these individuals are called "users." They usually have little knowledge of the details of how data is transferred from one node to another and are not concerned with the technicalities of network performance.

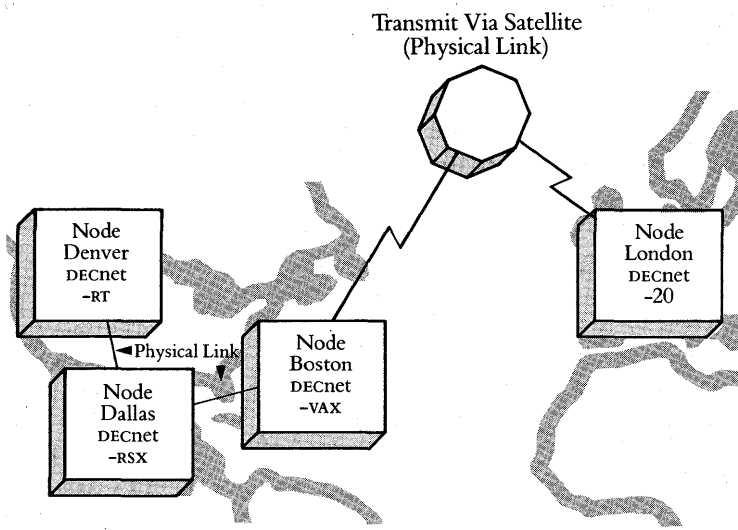


Figure 1.1
A DECnet Network

(b) individuals responsible for configuring and maintaining the network, for monitoring network performance, and for ensuring a smooth-running network environment. These individuals are called "network managers." Occasionally, the manual refers to "system managers," individuals who are responsible for the operation of the individual nodes making up a network. Although network managers are primarily responsible for implementing and maintaining networks, system managers can contribute greatly to high-quality network performance by closely monitoring their specific nodes.

DECnet permits great flexibility in configuring a network. A DECnet network can start with two nodes and be expanded to include over 64,000 nodes. Network managers can configure a *local area network,* with nodes located in a building or spanning a cluster of buildings, and expand the configuration to a *wide area network,* with nodes distributed in different cities across the country and even in different countries. Optionally, very large DECnet networks (with over 1,023 nodes) can be configured as interrelated groups of nodes or "areas." DECnet can support up to 63 areas with up to 1,023 nodes in each area. Area-based networks are particularly effective in large organizations; each department can have a separate area in a corporate-wide network.

The nodes in a DECnet network can run any of Digital's many operating systems. A DECnet network can comprise any combination of VAX systems running VAX/VMS; MicroVAX I systems running MicroVMS or VAXELN; PDP-11 systems running RSX-11M, RSX-11S, RSX-11M-PLUS, RSTS/E, or RT-11; DECSYSTEM-20s and DECsystem-10s running TOPS-20 and TOPS-10 respectively; and the Professional 350 personal computers running the P/OS operating system.

A very important feature of a DECnet network is that its nodes have a peer relationship with each other. This means that every node in the network can communicate with every other node without having to consult a central controlling node. The advantage of such a decentralized network is that each node can be equally responsive to user requests. A network user at one node can easily gain access to applications and facilities that exist on other nodes. Since every data exchange and remote access operation does not have to pass through a central node, communication overhead is reduced and network performance increases in efficiency.

Decentralization also provides a great deal of flexibility in the way networks can be configured. Chapter 3 describes the type of configurations possible in a decentralized network.

- **Uses of a DECnet Network**

It is DECnet that enables various Digital systems in an organization, which are fulfilling differing application requirements, to communicate among themselves. It lets them exchange data and share resources, thereby enabling the implementation of an efficient data processing operation.

Networked systems can share expensive resources such as laser printers and mass storage devices. Costly peripherals, therefore, don't have to be duplicated.

Systems in a network can exchange information so that fewer copies of a given file can be maintained. That way, users can be assured that they are accessing the most current version of a file. In addition, a network can speed up information flow within an organization, since file transfers between systems are rapid.

In a network, data that has been entered interactively at a personal computer can be sent to update a database on a large system. The resources of the large system are not tied up in interactive data entry and are available for complex tasks requiring extensive CPU power.

Networks make it possible for small systems to take advantage of the greater computing power of larger systems. Tasks that need extensive computation can be sent to large systems for quick execution. Conversely, large systems can take advantage of smaller systems by offloading certain applications onto machines designed for those specific jobs.

From their terminals, network users have access to a wide expanse of resources. They can access the computing power or data resources of any node on their DECnet network. They can also access the resources of non-Digital systems on other networks. DECnet supports gateways that enable Digital systems to communicate with systems from other vendors. The DECnet/SNA Gateway enables nodes in a DECnet environment to communicate with systems in an IBM System Network Architecture (SNA) environment. The DECnet Router/X.25 Gateway enables DECnet nodes to communicate with Digital and non-Digital systems implementing the X.25 protocol and connected to packet-switched data networks (PSDNs). See Chapter 8 for further explanation of communication with foreign-vendor networks.

The gateways enable organizations to maintain vendor independence and take advantage of the capabilities of systems from different manufacturers. At the same time, networks make it possible for the different systems in an organization to exchange information and share resources.

Gateways offload communication functions like routing and protocol translation from network nodes, freeing up the latter to process user applications with greater efficiency. See Chapter 3 for further details.

- **Digital Network Architecture**

The Digital Network Architecture (DNA) is the framework for all Digital communications products. The International Standards Organization (ISO), a group responsible for providing industry standards, has created an architectural model for open systems interconnection (OSI). Figure 1.2A shows that DNA and the ISO model are similar. DNA and the ISO model have evolved from common roots.

Architecture Layers	
ISO	DNA
Application	User
	Network Management
Presentation	Network Application
Session	Session Control
Transport	End-to-End Communication
Network	Routing
Data Link	Data Link
Physical Link	Physical Link

Figure 1.2A
ISO and DNA Models

DNA includes the specifications that govern the interrelationship of the various software components making up DECnet. There are two kinds of intercomponent relationship that DNA defines: interfaces and protocols.

Interfaces

Interfaces are definitions of specific functional boundaries between DECnet software components residing within a single node. The

boundaries are structured as a hierarchical set of layers (shown in Figure 1.2A). DECnet software is arranged, according to function, as modules within these layers (see Figure 1.2B). DNA layers, from highest to lowest, are:

User Layer–The user layer encompasses user-written programs and user-level services that access the network, network services that directly support user and application tasks, and overall system management. The most common services accessed by users at this level include resource sharing, file transfers, remote file access, database management, and network management. All are described in greater detail in subsequent chapters.

Network Management Layer–The Network Management layer defines the functions used by operators and network managers to plan, control, and maintain DECnet networks. This layer interfaces with all the DNA layers below it to gather data on network performance and change network operational parameters. Functions include downline loading of remote nodes and loopback network line testing. Chapters 10 and 11 discuss network management in greater detail.

Network Application Layer–The Network Application layer, which contains user- and Digital-supplied modules, defines network functions used by the User and Network Management layers. The most important DECnet functions currently operating in this layer are remote file access, file transfer, remote terminal capability (including the network virtual terminal function), access to X.25 connections using a Packetnet System Interface (PSI) or X.25/X.29 extension package, and access to SNA Gateways. This layer also defines a universal I/O language within DNA so that resource sharing among heterogeneous systems is simplified. Chapters 6 through 8 discuss the functions of this layer.

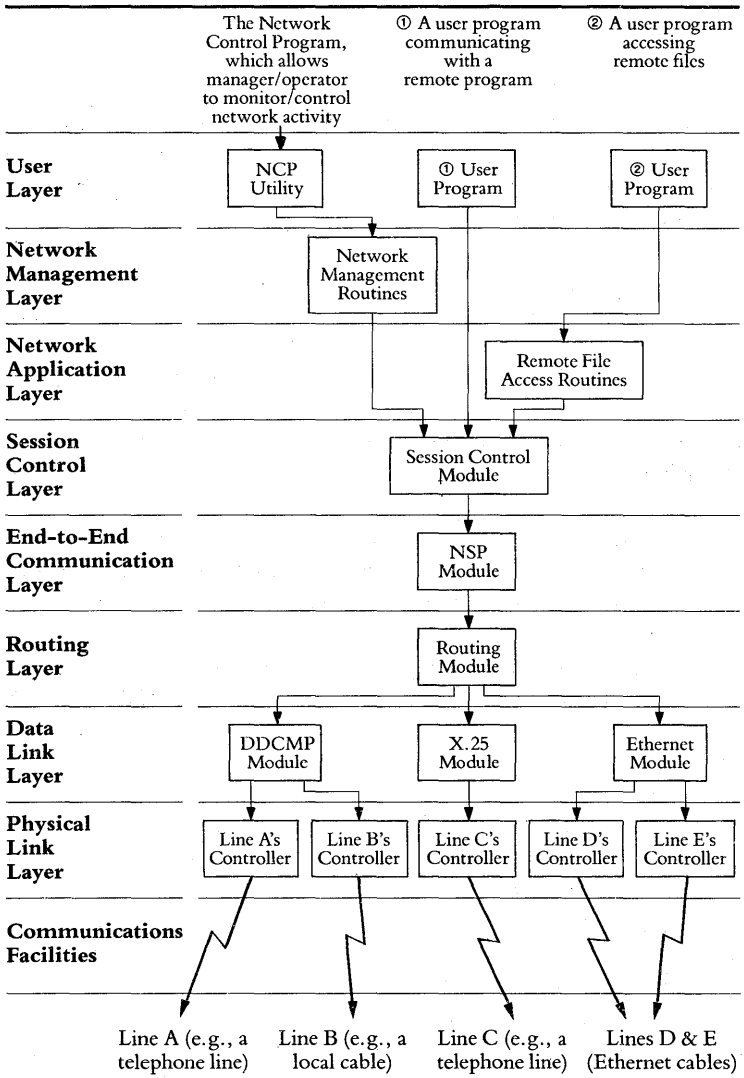


Figure 1.2B
DNA Modules Resident in a Typical DECnet Node

Session Control Layer–The Session Control layer defines the system-dependent aspects of process-to-process communications. The functions of this layer include name-to-address translation, process addressing, and access control. See Chapter 4 on common mechanisms in DECnet functions for a further explanation of the role of the Session Control layer.

End-to-End Communication Layer–The End-to-End Communication layer software handles the system-independent aspects of communications. These include connection management, data flow control, end-to-end error control, and segmentation/reassembly of user messages. See the discussion on common mechanisms in DECnet functions in Chapter 4 for a discussion of these concepts.

Routing Layer–The Routing layer software defines the mechanism for transporting or routing user data from the sending node to the receiving node. Modules in this layer also provide network congestion control and packet lifetime control. See Chapter 2 for an explanation of these concepts.

Data Link Layer–Modules in the Data Link layer define a mechanism for creating an error-free communication path between adjacent nodes, whether they are connected by an X.25 link, an Ethernet link, or a Digital Data Communications Message Protocol (DDCMP) link (point-to-point or multipoint, as described in Chapter 3). See Chapter 2 for a discussion of DDCMP and Ethernet and Chapter 8 for a discussion of X.25.

Physical Link Layer–The Physical Link layer defines the manner in which device drivers and communications hardware (modems and lines, for example) should be implemented to move data over a transmission line. The functions of modules in this layer include monitoring channel signals, clocking on the channel, handling interrupts from the hardware, and informing the Data Link layer when a transmission has been completed.

A module can use the services provided by modules in a lower layer and can provide services to modules in a higher layer. A module cannot access services offered by a module at a higher layer. The Network Management layer modules are the only ones that can interface directly with modules in each of the lower layers for access and control purposes.

In building-block fashion, DNA causes each layer of software to build on the services offered by the modules in a lower layer. DECnet has been designed in accordance with these rules so that high-level network functions, like program-to-program communication, occur in a predictable and regulated fashion on each DECnet node.

Protocols

Modules with equivalent functions in the same layer, but residing in different nodes, communicate using protocols. A protocol is both a set of messages and the rules for exchanging the messages. DNA defines the message formats and rules very specifically. Protocols provide a common code of behavior that two computer systems in a network can use to understand the role each is playing in communications and information exchanges. Figure 1.3 illustrates the relationship between protocols and interfaces. Figure 1.4 illustrates protocol communication between two nodes.

DNA does not define protocols for the User Layer. In this layer, programs must include the means to interpret received data (this

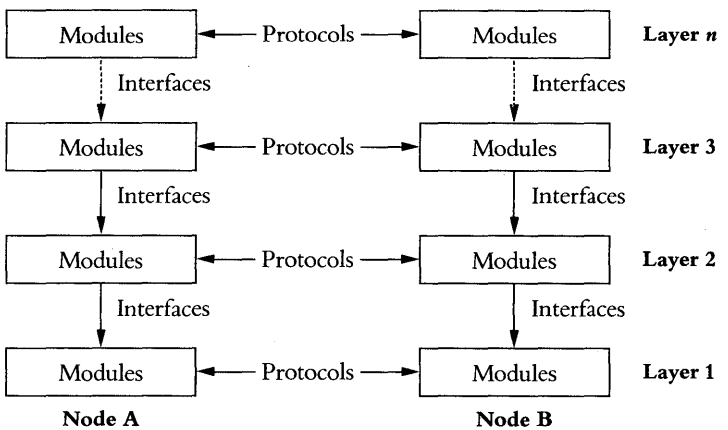


Figure 1.3
Relationship between Protocols and Interfaces

process is similar to interpreting the contents of a record on disk). Some DNA layers support more than one function and therefore include multiple protocols. An example is the Data Link layer, which supports the DDCMP, X.25, and Ethernet protocols. The protocols that DNA defines are not exclusive; customers can substitute their own, if they choose, as long as these protocols are implemented consistently by equivalent modules throughout the network.

The DNA framework of interfaces and protocols is transparent to most user activities. Users need not concern themselves with how data moves through the network.

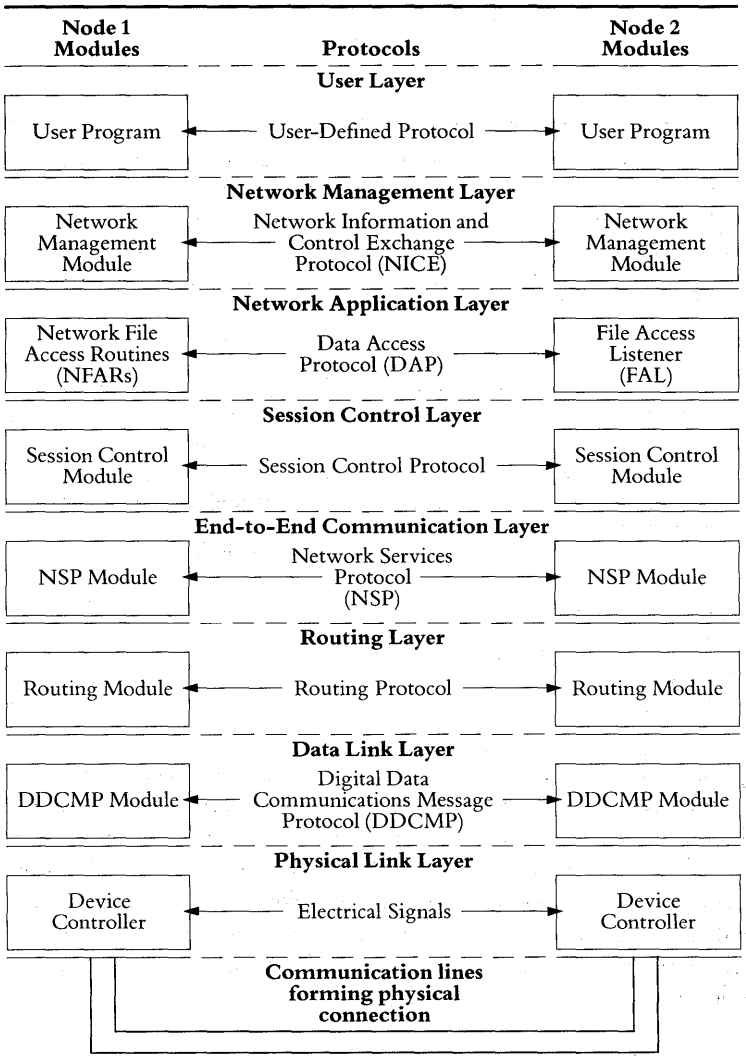


Figure 1.4
Protocol Communications between Two Nodes

Importance of a Network Architecture

DNA is an open-ended architecture designed to absorb new and increasingly efficient communications technologies. It is highly adaptable to different communications situations and supports a broad range of user applications and functions. Digital's adherence to a network architecture in the design of its communications products results in several important advantages for network customers.

Communications technology is changing continually. Future phases of DNA may model additional layers, additional modules within layers, or alternative modules for certain layers. With the wide range of available DNA facilities, organizations will be able to implement networks that are closely tailored to meet specific application needs *without incurring the expense of custom-designed networks.* Currently, DNA's Data Link layer supports three protocols–DDCMP, X.25, and Ethernet. Network managers can implement the protocol best suited to meet their organization's application needs. For example, one organization's network of systems located within a limited geographical area may have a very high traffic volume among nodes. In this case, the Ethernet protocol over an Ethernet cable would be a good solution. For a network with medium traffic volume among geographically dispersed systems, the X.25 protocol might be a more cost-effective solution.

A second advantage of a network architecture is that it specifies common communication mechanisms and user interfaces to which computer systems of different types must adhere when communicating. Without such a networking standard, data exchange and resource sharing among the variety of available operating systems, communication devices, and computer hardware would be highly difficult to effect.

Managing the different systems in a network would also be a very complicated task in the absence of a network architecture. Network management, error recording, and maintenance are easier when there are standard procedures for error detection, recording, isolation, recovery, and repair. An architecture provides this standardization so that network management can take place consistently across different systems. Standardization enables a network manager to manage an unattended remote system from a local node, even if the remote node is running a different operating system than that of the local node.

Yet another benefit of a network architecture is that it enables system-independent functions to be designed specifically enough to be implemented in hardware for improved performance. For example, the DEUNA communications device implements the Ethernet data link protocol; the DMR11, DMP11, and DMV11 implement the DDCMP data link protocol; and the KMS11-BD implements the X.25 frame-level protocol.

- **DECnet Capabilities**

DECnet enables users and applications to perform a number of high-level network functions. These include:

Task-to-task communication–Cooperating programs running under different operating systems and written in different languages can exchange data. For example, PDP-11 medical analysis systems can transfer data to a central VAX system without an operator's commands. See Chapter 5 for further details on task-to-task communication.

Remote file and record access–Programs and users can access files that reside on remote nodes. Remote file access facilities enable the

transfer of files between two nodes, the manipulation of files (opening a file, deleting data from or appending data to a file) on remote nodes, and the submission of command files for execution to a remote node in order to gain access to that node's resources

Consult Chapter 6 for additional information on remote file and record access.

Terminal-to-terminal communication–On some DECnet implementations, a user at one terminal can use a DECnet utility to send messages to users at other terminals in the network. Refer to Chapter 7 for further explanation.

Network terminal communication–From their terminals, users can log on to a remote node and execute commands as if they were typing on a local terminal at that node. DECnet Phase III supports this capability if the remote node is running the same operating system as the user's local node. DECnet Phase IV supports this capability even if the operating system on the remote node is different from that on the local node.

The Phase IV capability is known as *network virtual terminal.* This capability is particularly useful for program development. From their terminals, programmers can draw on the resources of any system in the network. Depending on the application they are developing, they can log on to the system best suited to their task. For example, for rapid compilation, they can submit source-code files to a high-performance VAX/VMS system. See Chapter 7 for more details on network terminal communication.

Problem isolation and Network management–Network managers can use DECnet commands and procedures to generate and define nodes

and to isolate network problems. By performing loopback tests, trace facilities, and dump analyzers (see Chapters 10 and 11 for further explanation of network management functions), network managers can monitor network nodes and keep abreast of network performance in order to identify and resolve potential difficulties before they become crises. Users can thus be assured of an error-free network.

Downline loading–With this DECnet capability, an entire software system or task developed on a node with adequate peripheral and memory support can be sent to target systems for execution. For example, programmers in a manufacturing setting can develop new process monitoring and control parameters on a central system and then transfer the parameters to smaller target systems located on the factory floor.

Upline dumping–When a target system begins to fail, DECnet's upline dumping capability automatically sends a system-image dump upline to an adjacent, larger system. For example, an RSX-11S DECnet node that has crashed can upline dump a copy of its system image to an adjacent RSX-11M DECnet node for interpretation. When the failing node is restored, it can be downline loaded to resume operation.

- **Topology Alternatives**

DECnet provides considerable flexibility in configuring a network. A wide area network can include nodes distributed across the country and even in different parts of the world. Depending on an organization's application requirements, a network manager can decide where the nodes will be located, which nodes will be directly connected to each other, and what physical channels (leased lines, coaxial cable, and PSDN links, for example) these connections will use.

Greatly facilitating the configuration of wide area networks is a DECnet feature known as *routing.* Routing eliminates the need for every node in a DECnet network to be physically connected to every other node in the network. Communication hardware and line costs are greatly reduced. With the reduction in the number of physical lines, the risk of downtime due to channel failure also decreases.

Routing ensures that information from a source node is delivered to a specified destination node even if the information has to travel through several intervening nodes. In the event a line fails, DECnet's *adaptive routing* feature finds an available alternative path.

DECnet Phase IV offers *area routing*, the method by which DECnet can support very large networks (up to 63 areas with up to 1,023 nodes in each area). In an area-based network, two types of routing mechanisms operate: *Level 1 routing* is the mechanism responsible for communication *within* an area; *Level 2 routing* is responsible for communication *between* areas. Level 1 and Level 2 routing enable a node in one area to exchange data and share resources with any node in any other area of the area-based network. Chapter 2 includes a comprehensive discussion of routing in networks with and without area routing.

The most recent version of DECnet, Phase IV, incorporates the *Ethernet baseband local area network* specification. What this means is that systems running DECnet Phase IV can be attached to a coaxial cable in order to communicate at high speeds (a maximum cable data rate of 10 Mbits per second). Using DECnet facilities, individuals on an Ethernet network can perform any of the high-level network functions described above.

In addition to enabling high-speed communications, Ethernet networks support a variety of communication servers (see Chapter 3 for details) that greatly reduce the cost of implementing a baseband local area network and significantly increase network performance. Communication servers are special purpose or dedicated nodes that perform communications functions for the other nodes in the network, thereby freeing the latter of communications overhead.

DECnet Phase IV also incorporates the *X.25 protocol* for communication over *packet-switched data networks* (PSDNs). A DECnet wide area network can, therefore, include systems linked through a PSDN. X.25 is a protocol recommended by the International Telephone and Telegraph Consultative Committee for data communications over common-carrier lines. By linking geographically dispersed systems through a PSDN, the cost of data communications can be reduced significantly. The tariffs for a packet-switched data network are based on the volume of data sent, rather than the distance or connect time between communicating systems. Cost savings with PSDN links can be substantial, especially when there is medium traffic volume among systems.

Through one physical link to a PSDN, logical links among many systems can be created. See Chapter 2 for an explanation of logical links. Users need not worry about how the information is routed through the PSDN; the carrier that supplies the PSDN is responsible for ensuring that data is routed to the correct destination.

The incorporation of the X.25 protocol in DECnet Phase IV gives users of packet-switched data networks easy access to all high-level DECnet functions. PSDN users need not write any additional code for their application programs in order to perform network functions such as file transfer, remote resource access, network virtual terminal, and program-to-program communication between DECnet Phase IV systems linked across a PSDN.

The *Digital Data Communications Message Protocol (DDCMP)* is the third protocol that DECnet Phase IV incorporates at the Data Link level of DNA. Depending on what is best for their applications, network managers thus have the option of configuring networks with links that support DDCMP, X.25, or the Ethernet protocol. Chapter 2 discusses these protocols in further detail.

In applications wherein peer relationship among nodes is not critical, organizations can save dramatically on expenses for lines and communication hardware by implementing a *multipoint configuration.* In such a configuration, several nodes are dropped from a single line with a central node controlling access to that line. (See chapter 3 for additional information on multipoint lines.)

DECnet Phases

Digital has been developing networking products based on DNA since 1975. This development effort has occurred in a series of phases. So far, there have been four DECnet phases. The products in each new phase are fully compatible with those produced in the previous phase. The sections below list the capabilities of Phase III and Phase IV DECnet nodes. (Appendix B in the *Overview of Digital Networking Products* contains a brief history of DECnet and discussions of earlier DECnet phases.)

Phase III

Functions that DECnet Phase III systems can perform include task-to-task communication, remote file access, terminal-to-terminal communication, network terminal communication (between systems running the same operating system–homogeneous systems), network management, downline loading, upline dumping, and loopback testing.

In addition, Phase III systems can perform message routing, multipoint communication, and communications with packet-switched data networks using X.25 protocols.

The following DECnet products provide Phase III capabilities: DECnet-20, DECnet-10, DECnet/E, and DECnet-RT.

Phase IV

Phase IV includes additional capabilities that establish Digital as a leading supplier of products for multivendor network environments. In addition to being backward-compatible with Phase III, Phase IV features include:

Data link independence, to increase data link options for customers in implementing networks. DECnet Phase IV-supported data link protocols are X.25, DDCMP, and Ethernet.

Dedicated communications server products that offload communications overhead from general purpose nodes (see Chapter 3).

Increased network routing support allowing over 64,000 nodes to be configured in a network.

Enhanced network management capabilities (see Chapters 10 and 11).

Gateway support for communications with nodes using X.25 protocol over a packet-switched data network (see Chapter 8). Gateway support for communications with IBM systems in an SNA network (see Chapter 8).

DECnet-VAX, DECnet-RSX (for 11M, 11M-PLUS, 11S, and Micro/RSX systems), DECnet-20, and PRO/DECnet have already implemented Phase IV capabilities. PRO/DECnet enables a Professional 350 personal computer to be connected to an Ethernet local area network to exchange information and share resources with other Phase IV nodes. Through a DECnet Router Server (see Chapter 3), the Professional 350 on an Ethernet local area network can communicate with Phase III and Phase IV systems that are not on the Ethernet.

Network activity involves the flow of information between *nodes* across *lines* and *circuits.* Data originating at a node is *routed* through the network until it reaches a specified destination node.

Nodes, lines, circuits, and routing are some major DECnet concepts. Knowledge of them is helpful in understanding how data flows through the network. A knowledge of routing, in particular, is essential in appreciating the flexibility DECnet permits users and network managers in performing their tasks.

• Nodes

A node is a network entity–an identifiable unit capable of processing, sending, and receiving network information. A node consists of an autonomous software entity, such as an operating system, with processors and other associated hardware that implement DNA specifications. Every node in a DECnet network has a unique numeric address. This address designates the location of a node in the network, just as a street address designates the location of a building in a town.

In area-based networks, a node's address includes an area number and a node number. The area number specifies the area in which the node is located, and the node number identifies the node within the specified area. A node address in an area-based network is a 16-bit number. The first 6 bits are reserved for the area number, and the next 10 bits for the node number. This division of the 16-bit node address limits the maximum number of areas in an area-based network to 63 ($2^6 - 1 = 63$), and the maximum number of nodes in an area to 1,023 ($2^{10} - 1 = 1{,}023$).

Data traveling from one node to another is enveloped by control information, which includes the address of the sending node, called the source address, and the address of the receiving node, called the destination address. (Data enveloped by control information is a *packet.*) These addresses inform nodes intercepting the

packet en route to its destination where the packet is going. Node addresses ensure that a data packet can always be identified in terms of the node that is sending it and the node that is receiving it, regardless of the path the packet travels. (See below, under *Routing*, for a discussion of the routing mechanism that determines data packet paths.)

Most users find it difficult to remember numeric addresses, particularly when there are many nodes in a network. Software in the Session Control layer of DECnet eliminates this difficulty. The system manager for each node in the network specifies names for all the nodes with which that node expects to communicate. These names are defined and stored in a database on that node and are meaningful only to the Session Control software running on that node.

Against the names in a node's database, Session Control software stores the unique numeric addresses associated with the names. A single node can be known by several names, since system managers for the different nodes in a network can assign a different node name for the same remote node. Consider a node in San Francisco. This node communicates with nodes in Boston, New York, Los Angeles, and New Mexico. Users at these other nodes access the San Francisco-based node by different names. Although a remote node can have several names, it has only *one* address. This address is consistent throughout the network, regardless of the name by which the node is accessed.

A Session Control module in each node maintains a node-name mapping table that stores the unique node addresses associated with the names of all the nodes with which that node communicates. Although the node-name mapping tables in the Boston, New York, Los Angeles, and New Mexico nodes will contain

different names to refer to the node in San Francisco, the tables in each of these four nodes will have the same address against the name by which they refer to the San Francisco node.

When a user requests connection to a node and refers to it by name, the Session Control module consults the node-name mapping table to determine the address of the destination node. It passes on this information to the End-to-End Communication Layer, which creates logical links between systems (see Chapter 4). For incoming connect requests from the End-to-End Communication Layer, the Session Control module uses the node-name mapping table to identify the node from which the request originated.

Session Control software enables users to refer to nodes by names that are meaningful to them (and which they can therefore remember easily). By means of the node-name mapping table, Session Control software finds the correct address of a node so that data arrives at its proper destination.

When system managers initially generate or configure their node into a network, they usually assign the name and address by which the node is to be known in the network. For small networks, these assignments can be hand-typed into each system. For large networks, an up-to-date master list of node addresses and names can be maintained on a particular node so that other nodes in the network can simply copy the information from the master file. (Existing nodes in the network exchange their node names and addresses with each newly configured node.)

Other node characteristics, such as routing capabilities and dedicated functions, are discussed in Chapter 3.

- **Lines and Circuits**

A line is a physical data path from a DECnet node to another DECnet node or from a gateway to an IBM SNA network or packet-switched data network (PSDN). A line is connected to a node by a hardware controller that drives or controls the line. Lines that use the public telephone network are often connected to a modem. Lines can be either dedicated (hard-wired) or dialup. Lines can be attached to a local area network cable or configured in point-to-point or multipoint configurations (see Chapter 3 for details on the latter two configurations).

Lines form the *physical* connections between adjacent nodes. There is a higher level of internodal connection called circuits. Circuits are *logical* connections that carry information between two nodes and operate over the physical medium of lines. Each point-to-point line has one circuit associated with it, and each tributary on a multipoint line has a circuit (see Figure 2.1). A single physical connection to an X.25 packet-switched data network (PSDN) can support many virtual circuits.

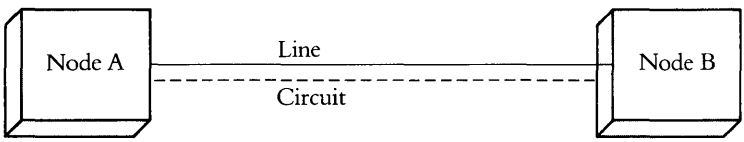
An Ethernet baseband network consists of one physical line with multiple circuits. Each circuit corresponds to a particular Ethernet protocol type. An Ethernet protocol type identifies the type of operation performed over a circuit. Figure 2.1 shows four Ethernet protocol types:

DNA routing protocol enables the operation of the DECnet routing mechanism and is required for all DECnet communications.

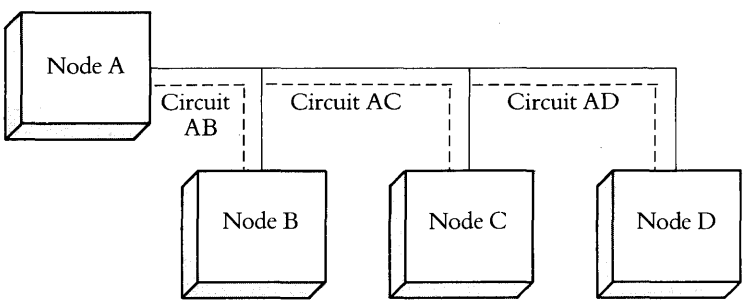
DNA remote console protocol provides simple internodal terminal communications used for diagnostic and maintenance procedures.

DNA dump/load protocol enables operators/managers to load system software remotely in nodes that do not have mass storage (for example, communication servers).

Point-to-Point Connection



Multipoint Connections



Ethernet Connections

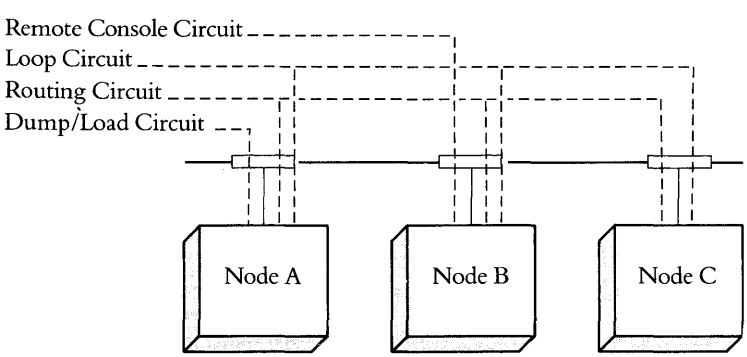


Figure 2.1
Circuits

Loopback protocol enables the performance of loop tests and is required for any type of testing procedure (see Chapter 11).

There are other protocol types not shown in Figure 2.1. Refer to the bibliography in the *Overview of Digital Networking Products* for a list of Ethernet documentation that further explains protocol types.

All communication activity in the network is considered in terms of circuits. DECnet's routing mechanism establishes a succession of circuits between a number of intervening nodes to get data from one node to another. These circuit pathways enable nodes to send data to other nodes that are not physically adjacent to them.

Transmission Modes

Information is transmitted over DECnet lines in half-duplex or full-duplex mode. In half-duplex mode, the line can transmit data in either direction, but only in one direction at any given time. Data cannot be sent and received over the line simultaneously. In full-duplex mode, the line can transmit data in both directions simultaneously, thereby making it possible for a node to send and receive data at the same time. The concepts of half duplex and full duplex relate to the physical line only.

Information flow at the higher level of the circuit always appears to be full-duplex. User data passes through the various DNA layers until it reaches the routing mechanism in the Routing layer. On the basis of predefined information, the routing mechanism determines the circuit over which the data will be transmitted to an adjacent node. The data then passes down to the Data Link layer.

Once the data leaves the Routing layer, the user services and the routing mechanism assume that it has been transmitted over the specified circuit. However, if the data cannot be transmitted over the designated physical line because it is a half-duplex line already carrying data, or, in the case of an Ethernet transmission, because the channel is not free, software in the Routing layer will buffer the data until it can be transmitted. The user is not aware of when the data is actually buffered or when it is sent out directly on a free line.

- **Data Link Protocols**

Currently, there are three protocols residing in the DNA Data Link layer:

Digital Data Communications Message Protocol (DDCMP)–a byte-oriented protocol

Ethernet protocol–Carrier Sense Multiple Access with Collision Detect (CSMA/CD) with physical channel encoding and operating over a coaxial cable

X.25 Levels 2 and 3–operating over Level 1 of the CCITT X.25 recommendation, which defines a standard interface between data terminal equipment (DTE), such as a DECnet node, and the data circuit terminating equipment (DCE) of a packet-switched data network.

This chapter will describe the DDCMP and Ethernet protocols. For additional details on the X.25 protocol, see Chapter 8.

The primary advantage of DECnet's support of three data link protocols is that network managers can configure networks based

on the data link protocol that best suits their organization's application requirements. Regardless of the data link protocol they implement, network managers can be assured of error-free transmission of data, since a primary function of the Data Link layer is to ensure the integrity of transmitted data.

DDCMP Functional Description

There are three general types of data link protocols: byte-oriented, character-oriented, and bit-oriented. DDCMP is a *byte-oriented protocol.* Such a protocol provides a count of the number of bytes that are sent in the data portion of each message. A character-oriented protocol uses special ASCII characters to indicate the beginning of a message and the end of a block of text, and a bit-oriented protocol uses flags to frame data sent in undefined lengths. Neither the character-oriented nor the bit-oriented protocol contains provisions for checking whether all the transmitted data has arrived at its destination. The advantage of a byte count in a byte-oriented protocol is that it facilitates checking on the part of the receiving node to see whether all transmitted data has been received.

DDCMP was designed in 1974 specifically for the Digital Network Architecture. DDCMP is functionally similar to HDLC–High-level Data Link Control–which was adopted in 1975 by the International Standards Organization. (HDLC is a bit-oriented protocol, however.) Another type of data link protocol that is commonly implemented is BISYNC (binary synchronous), which is character-oriented.

DDCMP is a general purpose protocol. It operates on a variety of communications devices, from the very small to the very large. DDCMP makes maximum use of channel bandwidth and handles data transparency efficiently (data transparency is the capability of

receiving, without misinterpretation, data containing bit patterns that resemble protocol control characters). Character-oriented protocols cannot handle transparent data as efficiently as byte- or bit-oriented protocols. Other major features of DDCMP include error recording to warn of impending channel failure on degraded lines and a simplified mode for bootstrapping and testing functions.

DDCMP transmits data grouped into physical blocks known as data messages and provides a mechanism for exchanging error-free messages. This mechanism works in the following manner: DDCMP assigns a number to each data message, beginning with the number one (after each initialization) and incremented by one for each subsequent data message. In addition, DDCMP places a 16-bit cyclic redundancy check (CRC-16) error-detection polynomial at the end of each data message transmitted.

The receiving DDCMP module checks for errors and, if there are none, returns an acknowledgement that it has received the message. Acknowledgement is efficient since the receiving DDCMP module does not have to acknowledge each message sent. Acknowledgement of data message *n* implies acknowledgement of all data messages up to and including data message *n*. If the receiving DDCMP module detects an error, it uses time-outs and control messages to resynchronize and trigger retransmission.

DDCMP Features–DDCMP is a powerful and versatile data link protocol. It has been designed to provide an error-free communications path between adjacent nodes. Features include the ability to

- Obtain data from the Physical Link layer in blocks of 8-bit bytes.
- Sequence data by message numbers.

- Send up to 255 messages without waiting for individual acknowledgement of each successive message. This capability is known as *pipelining*.
- Operate independently of channel bit width (serial or parallel) and transmission characteristics (asynchronous or synchronous).
- Operate with a wide variety of communications hardware and modems.
- Detect errors by means of CRC-16 message trailers.
- Retransmit to correct errors.
- Achieve optimum performance with techniques such as pipelining, *piggybacking* (sending an acknowledgement within a returned data message), and implying a postive acknowledgement of previous messages by negative acknowledgement of current message (see below, under DDCMP Control Messages).
- Operate in half-duplex and full-duplex modes.
- Support point-to-point and multipoint communications.
- Synchronize transmission and reception on byte and message level.
- Frame (construct envelopes) data messages.
- Provide a maintenance mode for diagnostic testing and bootstrapping functions.
- Provide data transparency.
- Notify the other end of the link when restarting or initializing.
- Maintain error counters.
- Record the occurrence of events for automatic error reporting to the user.

DDCMP Messages

There are three types of DDCMP messages: data, control, and maintenance. Data messages consist of user data. Control messages return acknowledgements and other control information to ensure data integrity and error-free transmission. Maintenance messages consist of information for downline loading, upline dumping, link testing, or controlling a remotely located, adjacent system.

• Data Messages

DDCMP formats all messages received from the Routing layer to be sent across the physical link into a data message format (Figure 2.2). The data message format ensures proper handling and error checking of both the header information and the data being sent. (In the message format figures in this chapter, the numbers below each message field indicate the length of the field in bits.)

Data Message Format

SOH	COUNT	FLAGS	RESP	NUM	ADDR	BLKCK1	DATA	BLKCK2
8	14	2	8	8	8	16	8*n*	16

SOH	=	the numbered data message identifier
COUNT	=	the byte count field
FLAGS	=	the link flags
RESP	=	the response number
NUM	=	the transmit number
ADDR	=	the station address field
BLKCK1	=	the block check on the numbered message header
DATA	=	the *n*-byte data field, where $0<n=\text{COUNT}<2^{14}$
BLKCK2	=	the block check on the data field

Figure 2.2
DDCMP Data Message Format

- **Control Messages**

DDCMP has five control messages that carry link control information, transmission status, and initialization notification between DDCMP modules. All the control messages operate to ensure an error-free link between sending and receiving DDCMP modules:

Acknowledge Message (ACK) acknowledges the receipt of correctly numbered data messages that have passed the CRC-16 check. The ACK message is used when no numbered data messages are to be sent in the reverse direction. The ACK message conveys the same information as the RESP field (see Figure 2.3) in numbered data messages.

Negative Acknowledgement Message (NAK) passes error information from the DDCMP data receiving module to the DDCMP data sending module. The NAKTYPE field (see Figure 2.3) indicates the cause of the error. The NAK message servers two purposes: it acknowledges receipt of all previously transmitted messages with a number less than the current message number received and it notifies the sender of error conditions relating to the current message.

Reply to Message Number (REP) is a message from the data sender to the data receiver requesting a received-message status. The data sender issues a REP message when it has sent a data message, has not yet received acknowledgement of that message, and the time allocated of an acknowledgement has expired.

Start Message (STRT) establishes initial contact and synchronization on a DDCMP link. A DDCMP module sends this message during link startup or reinitialization.

Start Acknowledge Message (STACK) is the response to a STRT message. It tells a receiving DDCMP module that the transmitting module has completed reinitialization.

Figure 2.3 shows the formats of the DDCMP control messages.

Acknowledge Message (ACK) Format

ENQ	ACKTYPE	ACKSUB	FLAGS	RESP	FILL	ADDR	BLKCK3
8	8	6	2	8	8	6	16

Negative Acknowledge Message (NAK) Format

ENQ	NAKTYPE	REASON	FLAGS	RESP	FILL	ADDR	BLKCK3

Reply to Message Number (REP) Format

ENQ	REPTYPE	REPSUB	FLAGS	FILL	NUM	ADDR	BLKCK3

Start Message (STRT) Format

ENQ	STRTTYPE	STRTSUB	FLAGS	FILL	FILL	ADDR	BLKCK3

Start Acknowledge Message (STACK) Format

ENQ	STCKTYPE	STCKSUB	FLAGS	FILL	FILL	ADDR	BLKCK3

ENQ	=	the control message identifier
ACKTYPE	=	the ACK message type with a value of 1
NAKTYPE	=	the NAK message type with a value of 2
REPTYPE	=	the REP message type with a value of 3
STRTTYPE	=	the STRT message type with a value of 6
STCKTYPE	=	the STACK message type with a value of 7
ACKSUB	=	the ACK subtype with a value of 0
REASON	=	the NAK error reason
REPSUB	=	the REP subtype with a value of 0
STRTSUB	=	the STRT subtype with a value of 0
STCKSUB	=	the STACK subtype with a value of 0
FLAGS	=	the link flags
RESP	=	the response number used to acknowledge received messages that checked out to be correct
FILL	=	a fill byte with a value of 0
ADDR	=	the tributary address field
BLKCK3	=	the control message block check

Figure 2.3
DDCMP Control Message Formats

- **Maintenance Messages**

Figure 2.4 illustrates the format of the maintenance message, which is used in DDCMP maintenance mode. The maintenance message is a DDCMP envelope for data controlling downline loading, upline dumping, link testing, and controlling an unattended computer system. The DNA protocol used in performing maintenance functions is the Maintenance Operation Protocol (MOP). MOP messages (see Chapter 10 for a description) are sent within the DDCMP maintenace message.

DDCMP Operation

The DDCMP module has three functional components: framing, link management, and message exchange.

Framing–The framing component locates the beginning and end of a message received from a transmitting DDCMP module. Framing involves locating and locking onto, or synchronizing with, a

Maintenance Message Format

DLE	COUNT	FLAGS	FILL	FILL	ADDR	BLKCK1	DATA	BLCKCK2
8	14	2	8	8	8	16	8*n*	16

DLE	=	the maintenance message identifier
COUNT	=	the byte count field
FLAGS	=	the link flags
FILL	=	a fill byte with a value of 0
ADDR	=	the tributary address field
BLCKCK1	=	the header block check on fields DLE through ADDR
DATA	=	the *n*-byte data field, where $0<n=\text{COUNT}<2^{14}$
BLKCK2	=	the block check on the DATA field

Figure 2.4
DDCMP Maintenance Message Format

certain bit, byte, or message and then receiving subsequent bits, bytes, or messages at the same rate as that at which they come in.

At the Physical Link level, modems and communications interfaces synchronize bits. The DDCMP framing component synchronizes bytes by locating a certain 8-bit window in the bit stream. On asynchronous links, DDCMP uses start/stop transmission techniques to synchronize bytes. On synchronous links, DDCMP searches for a SYN character. Byte synchronization is inherent in 8-bit multiple parallel links. DDCMP synchronizes messages by searching for one of the three special starting bytes (SOH, for data messages, ENQ for control messages, and DLE for maintenance messages) after achieving byte synchronization. To maintain message synchronization, DDCMP counts out fixed-length headers and, when required, counts out variable-length data based on the count field of the header.

Link Management–This component controls transmission and reception on links connected to two or more transmitting systems and/or receiving systems in a given direction. Link management controls the direction of data flow on half-duplex links and, with the use of link flags, controls the selection of tributary stations on multipoint links. In addition, link management uses selective addressing to control the receipt of data on multipoint links.

Message Exchange–This component transfers data correctly and in sequence over a link. Message exchange operates at the message level (after framing has been accomplished) to exchange both data and control messages.

Typical Message Exchange

DDCMP is a *positive-acknowledgement-with-retransmission* protocol. This means that, for each data message correctly received and passed to the Routing layer, DDCMP returns a positive acknowl-

edgement. Such an acknowledgement is either an Acknowledge Message (ACK) or a piggybacked acknowledgement in the response (RESP) field of a data message.

If a DDCMP module receives a message out of sequence or with an error detected by the CRC, DDCMP does not pass the data on to the Routing layer. Typically, DDCMP does not acknowledge this message. Eventually, a time-out occurs, and one of two things could happen: the data sending DDCMP module retransmits the message or the data receiving DDCMP module sends a NAK to the data sending module.

The transmission of DDCMP data messages includes the following steps:

1. The transmitter increments the message number and puts the number, n, in the data message. This message is transmitted within the required framing envelope. A timer is started.

2. The receiver frames and receives the message, checks the received CRC value against a computer CRC value, and compares the message number with the number it expects to receive. If the message checks out correctly, the receiver returns a positive acknowledgement (ACK) with that number, passes the message to the Routing layer, and increments the next expected number to $n + 1$. If the message fails to check out, the receiver ignores the message or sends a NAK.

3. The transmitter then follows one of three procedures:
If the transmitter receives a positive acknowledgement, it checks the number received to see if it is an acknowledgement of receipt of an outstanding message (a message that has been awaiting transmission). If this is the case, the transmitting DDCMP module notifies the Routing layer of successful transmission of that

message as well as of any previous lower-numbered outstanding messages. If all outstanding messages have been acknowledged, the timer is stopped. If one or more message remains outstanding, the timer is restarted.

If the transmitter receives nothing, the timer expires. The transmitter sends a REP control message to initiate error recovery.

If the transmitter receives a negative acknowledgement, it retransmits the message and all higher-numbered messages.

The transmitter can send several data messages before requiring that the first one be acknowledged. Acknowledgement of the highest-numbered message implies acknowledgement of all lower-numbered messages. A negative acknowledgement implies postive acknowledgement of any previously transmitted, lower-numbered messages.

Figure 2.5 shows a message exchange involving positive acknowledgement and pipelining. Figure 2.6 shows error recovery from a NAK.

Maintenance Mode

Maintenance mode uses the DDCMP framing and link management components but not the message exchange component. Sequencing or acknowledgement, if required, must be handled within the data fields of the maintenance messages and is part of a higher-level protocol.

Ethernet Functional Description

Computer systems and other digital devices on an Ethernet local area network can exchange data at a speed of up to 10 Mbits per second. An Ethernet local area network can be located in one building or can span a cluster of buildings. The incorporation of the Ethernet specification in Phase IV DNA means that systems running DECnet Phase IV can be connected directly to an Ethernet

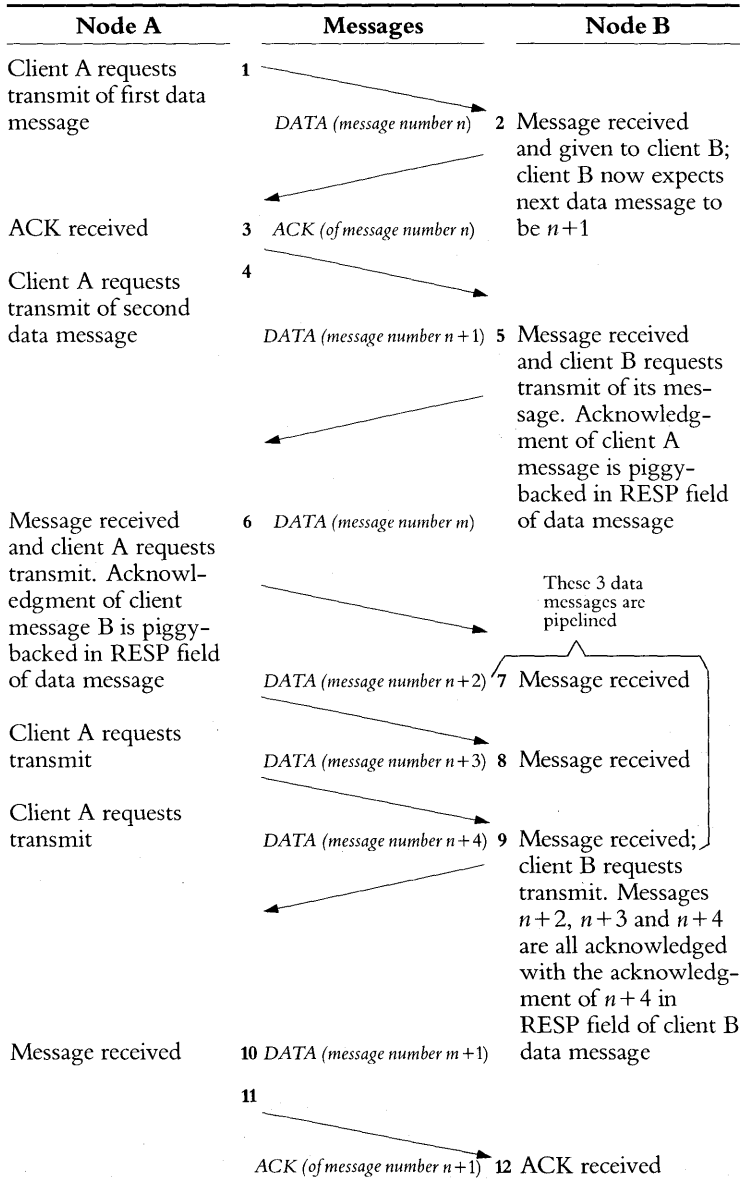


Figure 2.5
Typical Message Exchange, with Positive Acknowledgement, Piggybacking, and Pipelining

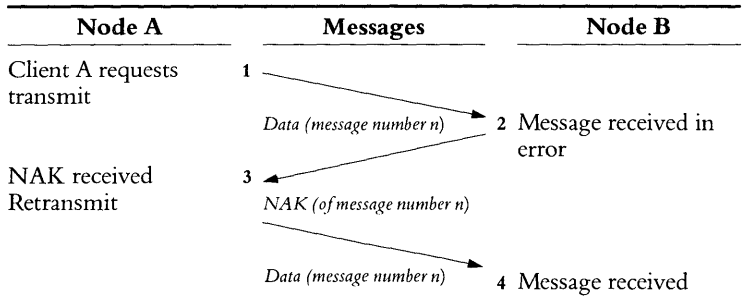


Figure 2.6
DDCMP Message Exchange, showing Error Recovery

coaxial cable to perform high-level network functions (file transfer and remote resource access, for example) at the high speed supported by the Ethernet cable.

The Ethernet local area network specification is the result of an extensive collaborative effort by Digital Equipment Corporation, Intel Corporation, and Xerox Corporation. The specification includes a Physical layer and a Data Link layer.

The primary characteristics of the Physical layer are: a data rate of 10 million bits per second, a maximum station (node) separation of 2.8 kilometers, a maximum of 1,024 nodes or stations on a single Etherent LAN, a shielded coaxial-cable medium using Manchester-encoded digital baseband signaling, and support of a bus structure in the shape of a branching tree.

The characteristics of the Data Link layer include multiaccess network control, in which access to the channel is fairly distributed to all nodes, and regulation of channel access through the *Carrier Sense Multiple Access with Collision Detect (CSMA/CD)* technique.

The functions of the Ethernet Data Link include data encapsulation/decapsulation and link management:

Data Encapsulation/Decapsulation comprises framing, addressing, and error detection. Framing defines the format of message packets (the different fields of information within packets) that are broadcast over the local area network. In addition, framing constructs packets from data supplied by the nodes through the higher layers, disassembles network messages, and supplies data to the higher layer protocols of the node. Addressing handles source and destination addresses, and error detection detects physical channel transmission errors.

Link Management comprises channel allocation and channel access. Channel allocation is responsible for amount of channel use, which is determined by the definition of packet size. Channel access is controlled by the CSMA/CD technique, part of which is carried out in each of the two layers. The Data Link layer responds to the channel or carrier sensing of the Physical layer by deferring transmission in case of traffic, transmitting in the absence of traffic, and backing off and resending in the case of a collision.

Figure 2.7 shows the two functional components of the Data Link layer, the Data Encapsulation sublayer, and the Link Management sublayer.

The Ethernet Data Link layer provides a best-effort delivery service. It does not provide an error-control facility to recover from transmission errors. In DNA, the End-to-End Communication layer is responsible for the error-control functions that are essential for reliable communications. Ethernet local area networks do not require additional error control at the data link level since the Ethernet physical channel has an inherently low error rate.

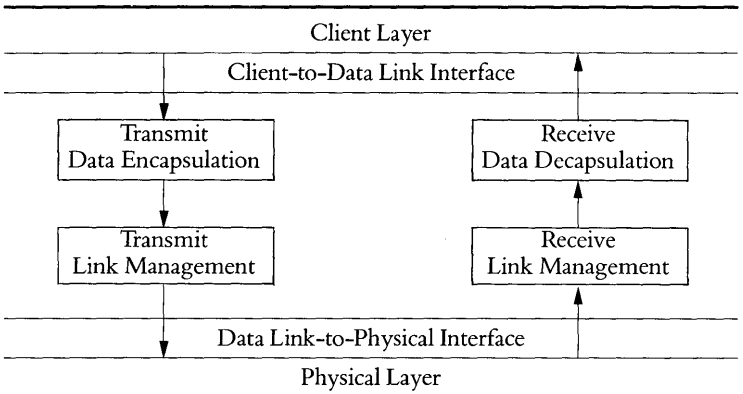


Figure 2.7
Ethernet Data Link Layer Functions

Ethernet Messages

The Ethernet Data Link layer has one type of message, the *frame*. The data encapsulation function of the Data Link layer comprises the construction and processing of frames. The frame format (see Figure 2.8) reflects the data encapsulation functions of framing, addressing, and error detection in the following manner:

Framing: no explicit framing information is contained in the format since the necessary framing cues are present in the interface to the Physical layer.

Addressing: there are two address fields to identify the source and destination nodes of the frame. Both address fields contain 48 bits. In DNA Phase IV, each network node has a 16-bit node address. The 48-bit Ethernet data link address of a DNA Phase IV node is derived

Ethernet Data Link Frame Format

DESTINATION	SOURCE	TYPE	DATA	FCS
48	48	16	$8n$	32

DESTINATION = the destination data link address. A data link address is one of three types:

- *Physical Address*: The unique address associated with a particular station on the Ethernet. The 48 bit field permits a station to have a unique address over all Ethernets. In DNA Phase IV, each network node has a 16 bit node address. The 48 bit Ethernet data link address of a DNA Phase IV node is derived by prefixing the 16 bit address with a 32 bit prefix assigned to DNA Phase IV nodes. Thus, DNA Phase IV addresses are unique over a single DNA network, which may include multiple Ethernets, DDCMP links, and X.25 links.
- *Multicast Address*: A multi-destination address associated by higher level convention with a group of logically related stations on an Ethernet. DNA assigns multicast addresses to the group of all Ethernet End nodes and to the group of all Ethernet Routers.
- *Broadcast Address*: A distinguished, predefined address which denotes the set of all stations on an Ethernet.

SOURCE = the source data link address. This field always contains the physical address of the station transmitting a frame on the Ethernet.

TYPE = the type field. The type field is reserved for use by higher level protocols to identify the higher level protocol associated with the frame, permitting multiple higher-level protocols to coexist in the same Ethernet. Ethernet type field values are assigned to the DNA Phase IV Routing protocol and to the DNA Maintenance Operation protocols.

DATA = the data field. The Ethernet data field contains higher level protocol data and is $8n$ bits long, where $46 \leq n \leq 1500$. Full transparency is provided, in the sense that any arbitrary sequence of 8 bit bytes may appear in the data field. The minimum length of the data field ensures that all frames occupy the channel long enough for reliable collision detection.

FCS = the frame check sequence. This field contains the CRC-32 polynomial check on the rest of the frame.

Figure 2.8
Ethernet Data Link Frame Format

by attaching to the 16-bit address a 32-bit prefix assigned to DNA Phase IV nodes. DNA Phase IV addresses are unique over a single DNA network that can include multiple Ethernet networks, DDCMP links, and X.25 links.

Error Detection: a Frame Check Sequence field is used to check the accuracy of the information contained in each transmitted packet.

In Figure 2.8, the numbers below each field indicate the length of the field in bits.

DNA defines a standard convention for padding higher-level protocols to the minimum Ethernet data field size. Higher-level protocol data is preceded by a 16-bit byte count and followed by any required pad bytes. When this convention is in effect, Ethernet frames have the format shown in Figure 2.9

Ethernet Frame Format with Padding

DESTINATION	SOURCE	TYPE	COUNT	DATA	PAD	FCS
48	48	16	16	$8n$	$8m$	32

COUNT	=	the 16 bit count of DATA bytes, where COUNT = n
DATA	=	the data field. The Ethernet data field contains higher level protocol data and is $8n$ bits long, where $0 \leq n \leq 1498$. Full transparency is provided, in the sense that any arbitrary sequence of 8 bit bytes may appear in the data field.
PAD	=	zero or more bytes of zeros, where $m = \max(0, 44 - n)$.

Figure 2.9
Ethernet Frame Format with Padding

Ethernet Operation

The higher layers of a node (layers above the Data Link layer) initiate transmission requests. These layers are referred to as the *client layers*, since the Data Link layer provides them with transmission and reception services.

Transmission Without Contention The client layer requesting transmission passes data, source and destination addresses, and data format type to the Transmit Data Encapsulation component of the Data Link layer. This component constructs a frame from the client-supplied information, with or without padding (depending on what the client has specified). The Data Link layer then appends a frame-check sequence for error detection and hands the frame to the Transmit Link Management component for transmission.

Before proceeding to transmit, Transmit Link Management attempts to avoid contention with other traffic on the channel by monitoring the carrier-sense signal of the Physical layer and by deferring to any passing traffic. The Physical layer is able to detect carrier–in this case, the electrical baseband signal that is present on the coaxial cable–whenever there is a frame being transmitted.

When the channel is clear, the Data Link layer passes the frame to be transmitted to the Physical layer as a serial stream of bits. Before transmitting any of the bits in the frame, the Physical layer sends an encoded preamble that allows the receivers of all the nodes on the channel to synchronize their clocks. The Physical layer then begins translating the bits of the frame into Manchester-phase-encoded form and generates electrical signals onto the coaxial cable that represent the bits in the frame.

The Physical layer of a transmitting node monitors the channel during the entire time it is transmitting (it is the energy level or the electrical-signal level of the channel that the Physical layer monitors). If there is a collision, the electrical-signal level increases. The Physical layer detects this increase. In a contention-free transmission, the *collision-detect* signal remains off. When the transmission has been completed without contention, the Data Link layer informs the client layer and awaits the next frame for transmission.

Reception Without Contention When a node begins to transmit in the absence of traffic on the channel, the Physical layers of all other nodes on the network sense its carrier signal and alert their respective Data Link layers so that there are no conflicting attempts to transmit. At that point, all other nodes on the network become receiving nodes and listen to the transmitted signal. The Physical layer of each receiving node synchronizes to the incoming preamble, receives the encoded bits from the cable, and translates the phase-encoded signal back into binary data, discarding the preamble.

Meanwhile, the Receive Link Management component of the Data Link layer has been waiting for the incoming bits to be delivered, since it was notified of a carrier signal. Receive Link Management collects bits from the Physical layer as long as the carrier-sense signal remains on. When the carrier-sense signal goes off, the frame is passed to the Receive Data Decapsulation component for processing.

The receiving node's Receive Data Decapsulation component checks the frame's destination address field to determine whether the frame is intended for that node. If it is, its type field is checked to decide which client module (Routing or Network Manage-

ment, for example) should process the incoming frame. If the transmitting client module has specified that padding is in use, the frame is decapsulated appropriately. The contents of the frame are then passed to the receiving client layer with an appropriate status code. The status code is generated by inspecting the frame check sequence to detect any damage to the frame and by checking for proper byte-boundary alignment at the end of the frame. If the destination address portion of the frame indicates that it is not intended for that node, the bit stream to the Data Link layer is halted after the preamble and destination address, and the frame is not accepted.

Collisions: Handling of Contention–It is possible that two or more nodes could determine that the channel is idle and attempt to transmit at or about the same instant. When this happens, their transmissions overlap and interfere with one another. The resulting contention is called a *collision*. A node can experience a collision only during the initial part of its transmission (the *collision window*), before its transmitted signal has had time to propagate to all parts of the Ethernet channel. Once the window has passed, the node or station is said to have *acquired the channel*, since all properly functioning stations would have detected the carrier signal by that time and would have deferred transmission.

In the event of a collision, the Physical layer first notices the interference on the channel and turns on the collision-detect signal. The Transmit Link Management of the Data Link layer notices this signal and initiates collision-handling procedures. First, the frame being transmitted is permitted to continue for a brief interval (called a *jam*), to ensure that the collision is noticed by other transmitting nodes involved in the collision. Then Transmit Link Management terminates the transmission, or *backs off*, and

schedules a retransmission attempt for a randomly selected time in the near future. When repeated collisions occur, retransmission attempts are also repeated, accordingly. Since repeated collisions indicate a busy channel, Transmit Link Management attempts to adjust to the channel load by voluntarily delaying its own retransmissions to reduce its load on the channel. Eventually, either the transmission is successful, or the attempt is abandoned with the assumption that the channel has failed or has become overloaded.

At all receiving nodes, the bits from a collision are received and decoded by the Physical layer as if they were bits from a valid frame (this happens because the collision detect signal is not acknowledged in receiving nodes). The Data Link layer's Receive Link Management component can distinguish between a collision fragment and a valid frame, since collision fragments are always shorter than the shortest valid frame. Receive Link Management does not accept collision fragments.

- **Routing**

Routing is the DNA layer whose function it is to determine the path or *route* along which a packet of data travels to its destination. A path consists of the sequence of circuits and nodes between a source node and a destination node. Users at a node need only specify the names of remote nodes with which they wish to communicate. Routing takes care of the details, ensuring that data travels to its appropriate destination. The path the data takes through the network is entirely transparent to users. In many cases, there are multiple paths between network nodes (see Figure 2.10), some more efficient than others.

In area-based networks, data travels within and between areas. Regardless of the number of areas between source and destination nodes, routing nodes forward a packet of data to its intended

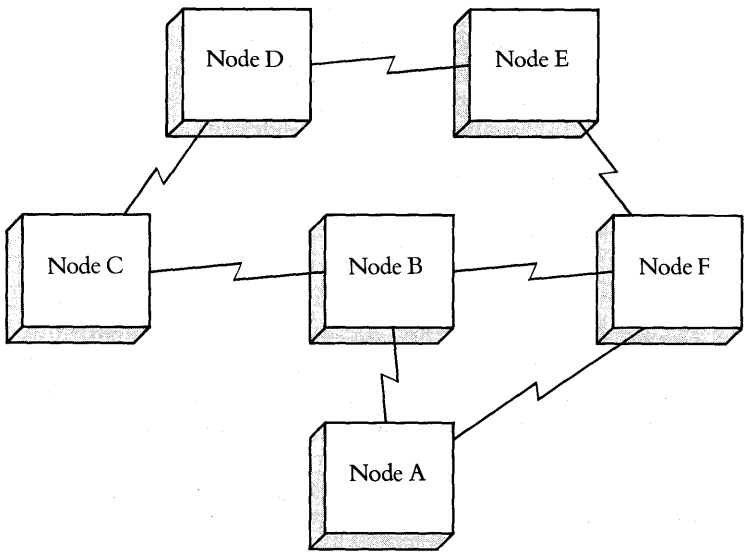


Figure 2.10
Routing Paths

location. *Router* is a general term for any node in a network that can receive and forward messages or packets addressed to other nodes in the network.

There are two types of routers in area-based networks: Level 1 routers, which are responsible for communication within an area, and Level 2 routers, which are responsible for communication between areas.

A Level 1 router operates in the same way as traditional Phase III and Phase IV DECnet routing nodes. It handles routing within an area and keeps itself informed of the state of all nodes in that area,

but it does not concern itself about nodes outside its area. When a node in one area (area 5, for example) wishes to communicate with a node in another area (area 7, for example), a Level 1 router in area 5 forwards the request to the nearest Level 2 router in the same area.

All inter-area communication takes place through Level 2 routers. Each Level 2 router keeps track of the least-cost path to each area in the network, as well as the state of the nodes within its area.

The Routing layer delivers packets on a best-effort basis. What this means is that the layer provides no guarantees against packets being lost, duplicated, or delivered out of order. A higher layer of DNA, the End-to-End Communication layer, provides these assurances. The Routing layer selects routes on the basis of network topology and operator-assigned circuit costs (see below, under *Routing Terms*) and adapts to changes in network topology (like finding an alternate path if a circuit or node fails).

Routing Terms

The total distance between a source node and a destination node is called the *path length.* The path length is measured in *hops.* A hop is the logical distance between two adjacent nodes. A path never exceeds a maximum number of hops; this number is a value that a network manager sets or one that is determined by the DECnet implementation (Phase III or Phase IV).

Routing allows the network manager to assign a *cost* to each circuit connected to a node. Cost is an arbitrary integer (within the size limits of the cost field) that is used to determine the best path for a packet. The cost of each path between a source node and a destination node is the sum of positive integer values assigned to the circuits that compose the path.

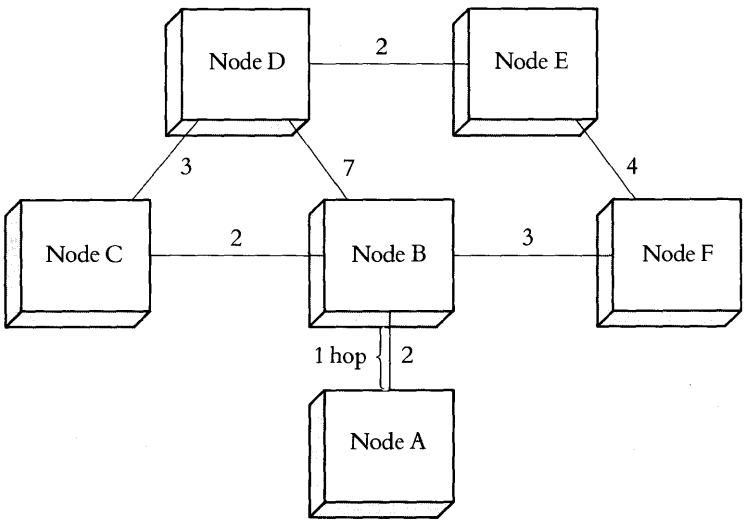
The Routing layer routes packets on the path of least cost, even if this is not the path with the fewest hops. The DNA Routing Functional Specification does not dictate how to assign circuit costs, but it does suggest a cost-assignment algorithm based on circuit bandwidth (since packets take the path with the lowest cost, a network manager could assign low-cost figures for circuits with high bandwidth and high-cost figures for circuits with low bandwidth).

The design of the Routing layer gives network managers great flexibility in configuring networks. Node locations and the assignment of circuit costs and maximum-hop values are left entirely to a network manager's discretion. Thus, a DECnet network can be tailored to meet the varying needs of differing application requirements.

Figure 2.11 illustrates some Routing terms. The glossary contains definitions of these and other Routing terms (including network diameter, maximum path length, maximum visits, maximum path cost, and maximum cost).

Routing parameters like circuit costs and maximum path lengths are defined by network managers when installing a node into a DECnet network. These values can also be modified after a DECnet system has been installed to improve network performance or to reflect changes in network configuration. The values of routing parameters should be determined only after careful consideration of their effects on the local node and on the network as a whole.

When a node is up and running, an operator can dynamically increase or decrease the cost for each circuit defined for that node. Altering circuit costs can result in alteration of packet-routing paths. The ability to change circuit costs dynamically can be an



Node A wants to send a packet to Node D. There are three possible paths.

Path	Path Cost	Path Length
A to B B to C C to D	2 + 2 + 3 = 7★	3 hops
A to B B to D	2 + 7 = 9	2 hops
A to B B to F F to E E to D	2 + 3 + 4 + 2 = 11	4 hops

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

Figure 2.11
Routing Terms

advantage when network performance suffers due to excessively high levels of traffic on low-cost circuits. (The Routing module itself does not automatically adjust to traffic flow, since packets always travel on the path with the lowest cost.)

Routing Algorithms and Databases

Routing algorithms operate on the routing parameters that system managers assign for every node in the network. One Routing algorithm calculates the path length and path cost to every possible destination via every circuit defined for that node. Another algorithm determines which circuit represents the least costly path to each destination in the network. The algorithms operate on locally available values (circuit costs and path lengths, for example) as well as on values supplied by all adjacent nodes.

Each node maintains routing databases in which are stored the results of the algorithms. In an area-based network, the Level 1 routers maintain routing databases that are similar to the routing databases in traditional Phase III and Phase IV DECnet nodes. The routing database of a Level 1 router includes information on path length and path cost to every node and Level 2 router in its area. A Level 2 router makes use of a second database–an area routing table–to determine the least-cost path to the other areas in the network.

Upon receiving a packet to be transmitted or forwarded, the Routing layer first consults the databases to find the least costly path to the packet's destination and then sends the packet via the selected circuit. A packet is discarded or returned to the sender if its destination is not accessible via any circuit known to the Routing module.

When the packet arrives at the next adjacent node in the path, the Routing module there also chooses the least costly path for the

packet, according to the local node's routing databases. The Routing module performs the services for packets being routed through the local node, as it does for packets that are originating at the local node.

Each Routing module executes the algorithms over and over again in response to events of different kinds on the network. If a physical line somewhere in the network goes down, the Routing algorithms recalculate the path length and cost to destinations affected by the line failure.

Whenever the algorithms reexecute, each node reveals the contents of its databases to all adjacent nodes, which use that information to update their own databases. In this way, changes affecting path lengths and costs ripple back and forth through the network so that all the routing databases are automatically updated to reflect the current state of the network.

In an area-based network, when a Level 1 router receives a packet addressed to a node in another area, it uses Level 1 routing to send the packet to the nearest Level 2 router. That router forwards the packet through Level 2 routers to a Level 2 router in the destination area. The destination Level 2 router then sends the packet through Level 1 routers in the destination area to the destination node.

Routing Features

In addition to determining packet paths, path length, and path cost, the routing mechanism performs other functions that ensure efficient data transfer over the network. These include:

Forwarding packets–If a packet is addressed to the local node, Routing delivers it to the End-to-End Communication layer. If a packet is addressed to a remote node, Routing forwards it to the

adjacent node in the path. In large area-based networks, Level 1 routing forwards packets between nodes within an area; Level 2 routing forwards packets between areas.

Adapting to topology changes–If a circuit or node in a path fails, Routing finds an available alternative path.

Adapting to different kinds of circuits–Routing operates over paths consisting of multiple types of Data Link circuits, including DDCMP point-to-point and multipoint lines, Ethernet links, and X.25 virtual circuits.

Periodically updating Routing modules on other nodes–Routing modules in other nodes are periodically updated to reflect any topology changes (circuits going down or up and operators modifying routing parameters, for example. See above).

Returning packets addressed to unreachable nodes–Routing returns packets addressed to unreachable nodes to the End-to-End Communication layer, thereby keeping it informed of the status of packet transmissions.

Limiting the number of packets queuing up for transmission on individual circuits–This prevents a circuit from becoming overloaded.

Managing buffers–Routing manages the buffers at nodes that are capable of routing packets from a remote source to a remote destination.

Regulating the ratio of packets to be forwarded through a node with those that are generated on that node–This achieves a balance between incoming and outgoing traffic.

Tracking the number of nodes a route-through packet has visited and discarding packets that have exceeded a predefined limit–This ensures that packets can never loop endlessly through the network tying up network resources.

Performing node verification–Routing will exchange verification passwords with an adjacent node if so requested by the Network Management layer.

Maintaining counters and gathering event data for network management purposes–This enables a network manager to identify and locate network problems such as traffic congestion.

See below, under *Routing Operation,* for details on these features.

How Routing Works on an Ethernet LAN

When a message is sent to an Ethernet node from a Phase III or Phase IV node that is not on the Ethernet or from a node on a different Ethernet network, a special routing node on the Ethernet–the DECnet Router Server (see Chapter 3)–or a host system routing node receives the message. Either the DECnet Router Server or the host system routing node submits the message to the Ethernet channel. (A DECnet Router/X.25 Gateway can perform this function, as well). The message travels along the Ethernet channel until it reaches the destination node (the destination node can recognize and retrieve messages addressed to it). If a message originates on an Ethernet and is addressed to another node on the same Ethernet, a routing node is not necessary. The source node transmits the message directly to the destination node as if there were a point-to-point connection between the nodes.

When the X.25 protocol is implemented through a gateway node (a VAX node or a DECnet Router/X.25 Gateway) rather than through data link mapping (discussed in Chapter 8), DECnet routing is not used over the PSDN. The PSDN handles all routing functions that occur within its boundaries. (See Chapter 8 for an explanation of X.25 implementations.)

DECnet supports area routing on an Ethernet channel. Groups of Ethernet systems may thus be segregated into separate areas. The routing overhead resulting from such a configuration does not make it a cost-effective solution, however. Nodes that would normally communicate directly with one another over the Ethernet cable now have to communicate through intervening Level 1 and Level 2 routers.

Routing Messages

There are two types of Routing messages: data packets and control messages. Data packets carry data to and from the End-to-End Communication layer. Routing adds a *packet route header* to messages sent by the End-to-End Communication layer. Routing modules in adjacent nodes exchange control messages to initialize the Routing layer, maintain routing data, and monitor the status of adjacent nodes.

Packet Route Header–The Routing layer of a node uses one of two formats for data packet routing headers, depending on whether the adjacent node is on an Ethernet or on some other type of circuit. When the adjacent node is an Ethernet end node (an Ethernet node that does not have the ability to receive and forward messages intended for other nodes), the Routing layer uses the *Ethernet Endnode Data Packet* format. For any other type of

Phase IV Data Packet Routing Header Format

RTFLG	DSTNODE	SRCNODE	FORWARD
8	16	16	8

Ethernet Endnode Data Packet Routing Header Format

RTFLG	DSTNODE	SRCNODE	RES	FORWARD	RES
8	64	64	8	8	16

RTFLG = the set of flags used by the routing nodes, including:
- Return to sender flag (indicates whether or not the packet is being returned)
- Return to sender request flag (indicates whether to discard or try to return packet)
- Intra-Ethernet Packet (indicates to the Ethernet Endnode receiving this packet that the source of the packet is on the same Ethernet, and can be communicated with directly).

DSTNODE = the destination node address
SRCNODE = the source node address
FORWARD = the number of nodes this packet can visit
RES = a reserved field

Figure 2.12
Packet Route Header Formats

adjacent node, the Routing layer uses the *Phase IV Data Packet* format. The Ethernet Endnode Data Packet format has expanded addressing and reserved fields, currently not used in DECnet Phase IV, for later expansion.

Figure 2.12 shows the data packet routing header formats. The numbers below the fields in this and in the next figure indicate the length of the fields in bits.

Routing Control Messages–A node's Routing layer uses six types of Routing control messages. It uses one type, the *Routing message,* on all types of circuits. It uses two types of Routing control messages–the *Ethernet Endnode Hello message* and the *Ethernet Router Hello message*–for Ethernet circuits only. (See above, under *Routing*, for definition of Router). And it uses the three other types of Routing control messages–the *Hello and Test message,* the *Initialization message,* and the *Verification message*–only on non-Ethernet circuits.

The *Routing message* provides the information necessary for updating the routing database of an adjacent node. This message carries path cost and path-length values for a set of destinations.

The *Ethernet Router Hello message* is used for the initialization and periodic monitoring of Routers on an Ethernet circuit. Through a multicast operation, each Etherent router periodically broadcasts an Ethernet Hello message to all other nodes (routers and end nodes) on the same Ethernet circuit. The message contains a list of all routers on the Ethernet circuit from which the sending router has recently received Ethernet Router Hello messages. By exchanging Ethernet Router Hello messages, all routers remain informed of the status of the other routers on the circuit. The message also provides input to an algorithm that selects a single router on the Ethernet circuit to be the *designated router* for that Ethernet. The designated router assists two end nodes in discovering that they are both on the same Ethernet. It does this by setting a special intra-Ethernet bit in a packet that it forwards from one end node to the other. Subsequent communications between the end nodes can be direct, without router intervention.

The *Ethernet Endnode Hello message* is used for the initialization and periodic monitoring of end nodes on an Ethernet circuit. By

means of a multicast operation, each Ethernet end node periodically broadcasts an Ethernet Endnode Hello message to all routers on the Ethernet circuit. The routers use this message to maintain the status (up or down) of end nodes on the Ethernet.

The *Hello and Test message* tests an adjacent node to determine if an adjacency is operational (an *adjacency* is a circuit and node pair). The Routing layer of a node sends this message periodically on non-Ethernet circuits in the absence of other traffic. When a node's Routing layer receives this message or any other valid message from an adjacent node, the Routing layer starts or restarts a timer. If the timer expires before the Routing layer receives another message from the adjacent node, Routing considers the adjacent node down.

The Routing layer of a node sends an *Initialization message* when initializing a non-Ethernet circuit. The message contains information about the node type, required verification, maximum Data Link layer receive block size, and Routing version.

A node's Routing layer sends a *Verification message* if, on a non-Ethernet circuit, the initialization message requires that node verification accompany the initializing process.

Figure 2.13 illustrates the Routing control message formats.

Routing Operation

The Routing layer of a node consists of two sublayers with associated functional components. The *Routing Control Sublayer* performs routing, congestion control, and packet-lifetime-control functions. The *Routing Initialization Sublayer* performs the initialization function.

Routing Message Format

CTLFLG	SCRNODE	RTGINFO	CHECKSUM
16	16	16*n*	16

Ethernet Router Hello Message Format

CTLFLG	VERS	ID	INFO	LIST
8	24	48	64	56*m*

Ethernet Endnode Hello Message Format

CTLFLG	VERS	ID	INFO
8	24	48	168

Hello and Test Message Format

CTLFLG	SRCNODE	TEST DATA
16	16	8-128

Initialization Message Format

CTLFLG	SRCNODE	INITFO
16	16	56

Verification Message Format

CTLFLG	SRCNODE	FCNVAL
16	16	8-64

CTLFLG = Routing control flag, with the following types:
- Initialization message
- Verification message
- Hello and Test message
- Routing message
- Ethernet Router Hello message
- Ethernet Endnode Hello message

SRCNODE = Identification of source node's Routing Module

RTGINFO = Path length and path cost to a set of *n* destinations

CHECKSUM = One's complement add check on routing information

VERS = Routing module version number

ID = Identification of node sending message

INFO = Node type (Router or Endnode), maximum Data Link layer receive block size, hello timer

LIST = list of known Routers on the Ethernet circuit (Each of the *m* entries consists of a 7 bit priority field for selecting the Designated Router, 1 bit to indicate two-way connectivity, and the 48 bit identification of a Router)

TEST DATA = sequence of up to 128 bytes of data to test the circuit

INITFO = node type, required Verification message, maximum Data Link layer receive block size, Routing version.

FCNVAL = type-dependent verification information; function value.

Figure 2.13
Routing Control Message Formats

Routing–The routing function is made up of five processes: decision, update, forwarding, and receive.

The *decision* process selects routes to each destination node in the network. This process uses a connectivity algorithm that maintains path lengths and a traffic-assignment algorithm that maintains path costs. When a routing node receives a Routing message, the routing node executes the connectivity and traffic-assignment algorithms. This execution results in updates to the databases that the routing node uses to determine packet routes.

The *update* process constructs and propagates Routing messages. It sends Routing messages to adjacent nodes whenever the decision process requires it to do so, and it also periodically sends Routing messages to adjacent nodes to ensure the integrity of the routing databases.

The *forwarding* process supplies and manages the buffers required to support packet route-through to all destinations. It looks up a table to select the output circuit for a packet. If a destination is unreachable, the forwarding process either returns the packet to the sender or discards it, depending on the option flagged in the packet route header.

The *receive* process inspects a packet's route header. If the packet is intended for the local node, the receive process dispatches it to the local End-to-End Communication layer. If the packet is intended for another destination, the receive process dispatches it to the Routing layer of the next node in the packet's path.

Congestion Control–This function consists of a single process: *transmit management*. This process manages buffers by limiting the maximum number of packets on a queue for a circuit. If a queue for

a particular circuit reaches a predetermined threshold, additional packets for that queue are discarded to prevent congestion. Transmit management also regulates the ratio of packets received directly from the End-to-End Communication layer to route-through packets. This regulation prevents locally generated packets from degrading a node's route-through service.

Packet Lifetime Control–This function prevents excessive looping by discarding packets that have visited too many nodes.

Initialization and Circuit Monitor–By means of the Routing initialization function, which is both a startup and periodic procedure, the Routing layer of a node ascertains the identities of neighboring nodes and adapts to the characteristics of the circuits to these neighboring nodes.

On an Ethernet circuit, the Ethernet Endnode Hello message and the Ethernet Router Hello message perform initialization and monitoring functions (see above, under specific Routing message types for a description).

On DDCMP circuits (point-to-point and multipoint), the Initialization message, the Verification message, and the Hello and Test message perform initialization and monitoring. In addition to the functions these messages perform, described above, the Initialization message adapts the Routing layer's operation to the block size of the physical line that the Data Link layer manages.

On X.25 virtual circuits (permanent and switched), the Initialization and Verification messages work in the same manner that they do on DDCMP circuits. The difference is that on X.25 circuits, the Routing layer uses a Data Link block size that is taken from the X.25 data packet size selected when the virtual circuit is initialized.

X.25 circuits are initialized when the Network Management layer commands the Routing layer to do so. The Routing Initialization Sublayer contains a database, maintained by Network Management, that includes a mapping between Routing layer circuits and X.25 packet-level permanent virtual circuit identifiers (in the case of permanent virtual circuits), and between Routing layer circuits and virtual call parameters (in the case of switched virtual circuits). The database contains items such as DTE subaddress range, maximum packet size, and the number of times failed calls should be reattempted.

When placing a virtual call, the Routing Initialization Sublayer provides the necessary parameters to the X.25 packet-level component of the Data Link layer, which places the call to the remote Routing layer at the supplied foreign DTE address. If the remote DTE accepts the call, the Routing Initialization function proceeds with circuit initialization.

The Routing Initialization module listens for incoming virtual calls managed by the X.25 packet-level component of the Data Link layer. When informed of an incoming call by the X.25 packet-level component, Routing Initialization executes an algorithm to determine whether to accept the call, reject it, or leave it pending in case some other DNA module (such as the X.25 Gateway Server module–see Chapter 8) wishes to process the call. The algorithm uses the mapping database described above and makes its decision by comparing the foreign DTE subaddress in the incoming call packet to the DTE subaddress range in its database. If the decision is to accept the call, initialization proceeds as described above.

Once an X.25 circuit is established, the Routing Initialization Sublayer performs additional functions not performed for

Ethernet or DDCMP circuits. These functions, which maximize the performance and integrity of X.25 circuits, include:

Blocking and Deblocking Routing Layer Messages–Since communication over X.25 circuits is in the form of packets, Routing Initialization will block and deblock DNA datagrams into X.25 packets in order to minimize the number of packets transmitted over the virtual circuit.

Fragmenting and Reassembling Routing Layer Messages–If the X.25 virtual circuit packet size is too small to contain an entire Routing layer message, Routing Initialization will fragment and reassemble Routing layer messages so that they fit in the available X.25 packet size.

Checksums of X.25 Packets–To insure the integrity of transmitted data, the Routing Initialization Sublayer appends a checksum to every data packet transmitted over the virtual circuit.

As it does for other circuits, the Routing Initialization Sublayer monitors X.25 circuits for non-recoverable errors, including X.25 resets, restarts, and clears. If any of these events occurs, Routing Initialization declares the circuit down, informs the Routing Control Sublayer of the condition, and attempts to reinitialize the circuit.

Figure 2.14 summarizes the operation of the Routing layer.

- **Data Flow**

The primary purpose of a network is to pass data from a source node to a destination node. Data traveling from one node in a network to another node passes from a source process in the User

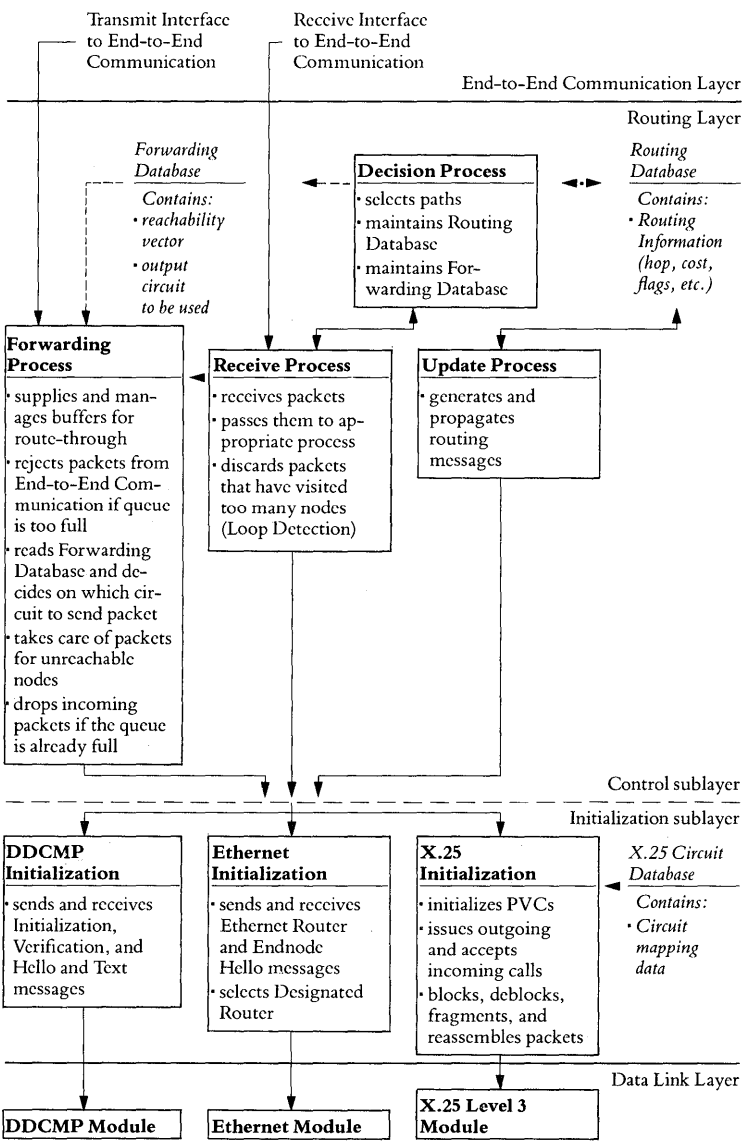


Figure 2.14
Routing Layer Components and their Functions

layer down through each layer of the DNA hierarchy of the source node, before being transmitted across a line. If the destination node is not adjacent to the source node, the data travels up to the Routing Layer of the adjacent node, where it is *routed* (or *switched*), sent back down through the two layers, and transmitted across the next line in the path. The data travels in this manner until it reaches its destination node. At this node, the data passes up the hierarchy of layers to the destination process in the User layer.

As the data passes down through the several layers in the source node, it acquires control information. Each layer adds control information to the packet of data it receives from the layer above it. The control information by which data is enveloped as it travels over a line ensures that it is transmitted with no errors and that it arrives at the proper destination. Figure 2.15 illustrates what happens to data as it passes through the DNA layers at a node. In this example, the Network Management layer is not involved.

In the following scenario a user attempts to form a logical link with another user. The requesting user passes initial connection data to the destination user. The numbered steps correspond with the numbers in Figure 2.16 following this explanation.

Data Flow at the Source Node

1. The source user requests a connection to the destination user and passes connect data.

2. The Session Control module receives the data, maps the destination node name to a numerical address, and places the data in the next transmit buffer, adding control information to the message. The message then passes to the End-to-End Communication layer.

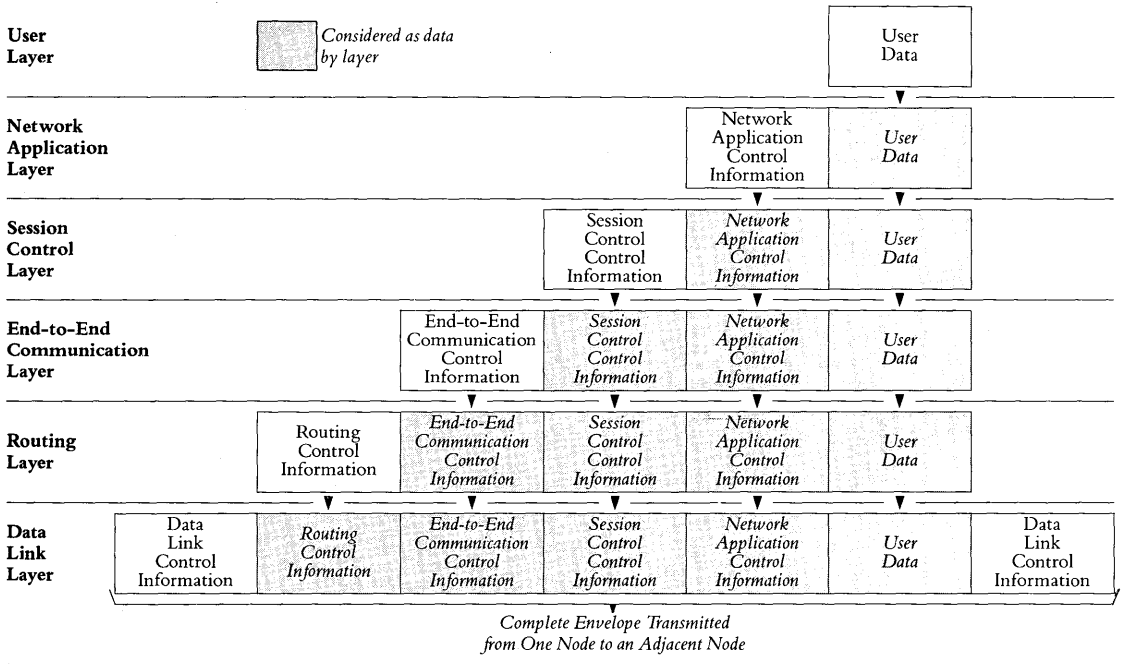


Figure 2.15
Information Building as Data Traverses DNA Layers

3. The End-to-End Communication layer adds its control information and passes the message (now called a datagram) to the Routing layer.

4. The Routing module adds a header consisting of the destination- and source-node addresses and selects an outgoing circuit for the message based on routing information. Routing then passes the message (now called a packet) to the Data Link layer, specifying the outgoing circuit address and, if necessary, the station address of the receiving node on the circuit.

5. The Data Link module adds its protocol header, consisting of framing, synchronizing, addressing, and control information, and its protocol trailer consisting of a cyclic redundancy check (CRC). The frame is now enveloped for transmission.

6. The Physical layer module transmits the enveloped message over the physical line.

Data Flow Across the Network to the Destination Node

7. The enveloped data message arrives at the next adjacent node. The Physical layer module receives the message and passes it to the Data Link layer.

8. The Data Link module checks the packet for bit errors in transmission. On links with a significant probability of transmission errors, the Data Link protocol performs error correction. DDCMP and X.25 provide error protection by retransmitting. On links with a low probability of transmission error–Ethernet links, for example–packets received with errors are discarded, with higher layers providing error recovery. The Data Link header and trailer are removed from a correctly received packet, which is then passed up to the Routing layer.

9. The Routing layer checks the destination address in the header. If the address is not that of the local node, Routing selects the next outgoing circuit from its routing table and passes the message back to the Data Link layer. The Routing layer has routed the message on to the next line in its path. The message proceeds as in steps 5 and 6 above.

10. The message proceeds through the network, switching at routing nodes, until it reaches the Routing layer of the node with the same address as the destination address in the message.

Data Flow at the Destination Node

11. The packet passes to the Routing layer of the destination node as described in steps 7 and 8, above. The destination Routing module removes the Routing header and passes the datagram to the End-to-End Communication layer.

12. The End-to-End Communication layer module examines the End-to-End Communication layer header in the datagram. If the module has the resources to form a new logical link, it passes the connect data, without the End-to-End Communication layer control information, to the Session Control layer.

13. The Session Control module performs any necessary access control functions (to ensure if the incoming connect request is valid and has the privilege to be given access) and passes the message to the appropriate process in the User layer after removing Session Control header information.

14. The destination process interprets the data according to whatever higher-level protocol is being used.

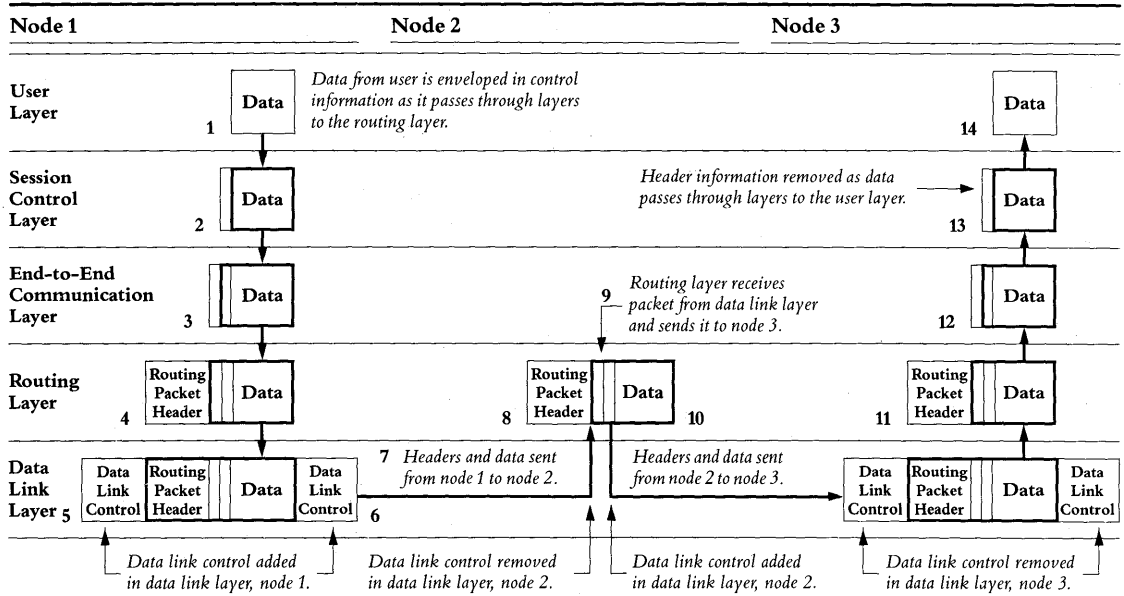


Figure 2.16
Data Flow

Chapter 3 · DECnet Configurations

This chapter discusses how nodes and lines (presented in Chapter 2) can be arranged in various configurations.

The previous chapter explained how the routing mechanism enables nodes that are not linked by a direct physical line between them to communicate. Routing permits great flexibility in the way that nodes and lines can be combined in networks, since every node in the network does not have to be connected by a direct physical line to every other node.

The major factors affecting network configuration are requirements of the applications that will use the network and capabilities of the network components (environment, node characteristics, and line and circuit characteristics). These capabilities will dictate the limits within which networks may be configured.

Network environment is one of the capabilities affecting configuration. There are two basic environments: wide area networking environment, with nodes distributed over a vast geographical area (across a country or even across different countries), and local area networking environment, with nodes distributed in a limited geographical area (a building or cluster of buildings). Wide area networks can include local area networks (see Figure 3.9). Within wide area and local area networks, nodes can be configured in several ways, depending on application requirements.

Node characteristics, such as the ability of the node to receive and forward messages intended for other nodes and the type of network services it can perform, greatly affect how a network manager can position it in a network configuration. Whether the node implements DNA Phase III or Phase IV software is also an important consideration in determining how it will be configured in the network.

Lines and their associated circuits can be arranged in bus structures, point-to-point arrangements, or multipoint configurations.

This chapter on configurations does not describe configuration details for specific DECnet systems or give definite specifications for configuring a network. Rather, it is a general discussion on the types of configurations that are possible using DECnet. For details on how to configure a specific system into a network, refer to the system manager's guide for that particular DECnet system.

• DECnet Environments

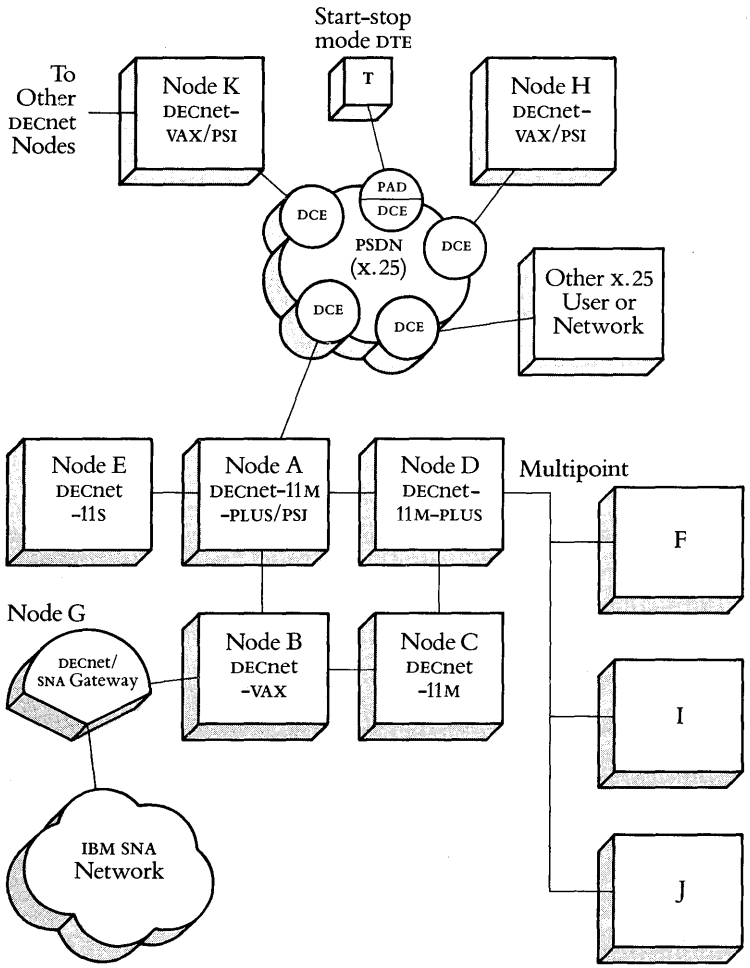
There are two basic environments for DECnet configurations: wide area networks and local area networks (LANs). Figure 3.1 shows a wide area network and Figure 3.2 shows a local area network. Each type can be configured with a variety of nodes and links. However, only Phase IV DECnet nodes can be directly connected to a local area network cable.

Performance and response time are better over a LAN than over a wide area network. From an end user's perspective, this is the only noticeable difference between the two types of network. All other aspects of communication between nodes appear to the user to be identical in either network.

Wide Area Networks

A wide area network is composed of nodes connected by individual communications links configured in various patterns, as shown in Figure 3.1. In general, wide area networks are used for long-distance communications. Nodes in wide area networks can be located many thousands of miles apart.

In a wide area network, messages can be transported over dialup, leased, or switched lines. Wide area networks typically use



PSDN = Packet Switched Data Network

DCE = Data circuit-terminating equipment, a PSDN switching node

DTE = Data terminal equipment, a user computer (packet mode) or a terminal using the PSDN (start-stop mode)

Figure 3.1
Wide Area Network Configuration

common carriers, like the telephone network, to transport messages over most or part of the distance. Because wide area networks use the telephone system for most communications, modems or modem equivalents are generally needed on each end of a communications link to convert digital data to analog form, and vice versa, and to permit transmission over long distances.

DECnet supports transmissions ranging from 1,200 bits per second to 56 Kbytes per second for remote connections over leased or dialup lines and over packet-switched data networks. A network manager can specify different transmission speeds for the various communications lines in a wide area network. Transmission speeds depend on the supported hardware devices and the type of lines used. Specifying transmission speeds gives a network manager some leeway in distributing workloads and adjusting performance levels for various network components.

See below under *Node Characteristics* and *Line and Circuit Characteristics* for guidelines on how to combine nodes in wide area configurations. Any node restrictions in wide area configurations, such as nodes that can be connected only to a local area network, are also identified in these sections.

Examples of wide area networks include private packet-switched networks such as the data networks developed by many companies, packet-switched data networks such as public data networks using the X.25 interface (Telenet and Transpac, for example), and leased-line networks.

Local Area Networks

Local area networks (LANs) are privately owned networks that offer reliable high-speed communication channels. They are optimized for connecting information-processing equipment in a limited geographical area, such as an office, a building, or a

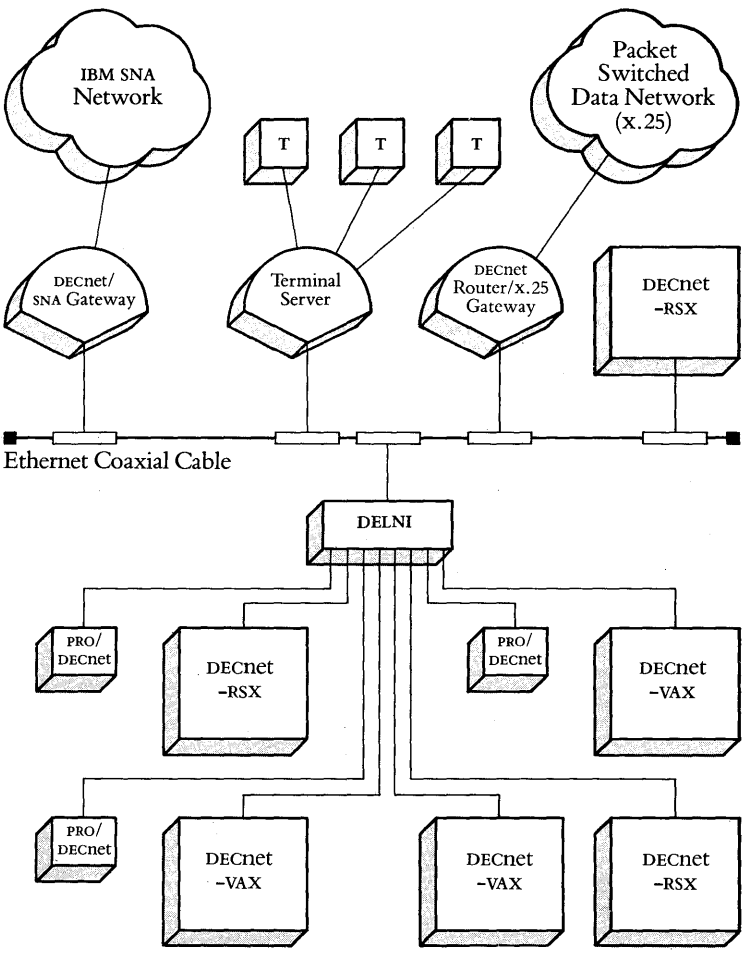


Figure 3.2
An Ethernet Local Area Network

complex of buildings. LANs can be designed with a variety of technologies and arranged in different configurations. Consequently, they vary significantly with respect to their transmission speeds, the distances they span, and the capabilities and services they offer.

An Ethernet baseband LAN consists of a segment of coaxial cable or several segments of coaxial cable joined together, with each segment ranging in length from 20.4 to 500 meters. The Ethernet cable is terminated at both ends (see Figure 3.3).

A Phase IV DECnet Ethernet network supports up to 1,023 nodes. Ethernet supports a bus topology, with each node attached to the cable by a single line. The channel bandwidth of an Ethernet cable is 10 Mbytes. The controller on the host system and host-system bottlenecks (caused by disk I/O, for example) limit CPU throughput to considerably less than 10 Mbytes. Typically, each node can achieve a throughput between 400 and 1,400 Kbytes.

Nodes that support a UNIBUS structure, such as PDP-11 and VAX nodes, connect to the Ethernet cable by means of a network adapter called the DEUNA. Nodes with a Q-BUS structure, such as MicroVAX and MICRO-11 systems, connect using a network adapter called the DEQNA. The DECNA controller connects the Professional 350 personal computer to the Ethernet cable. The DEUNA, DEQNA, and DECNA adapters, which are installed as

Figure 3.3
Ethernet Cable

communications devices in a DECnet node, incorporate hardware module boards, a distribution panel, and cabling. All three adapters connect to Digital's H4000 transceiver physically and electrically by a transceiver cable (see Figure 3.4 for DEUNA and DEQNA connections). A maximum of 100 transceivers can be used on a single 500-meter segment of Ethernet cable.

A communications controller called the DELNI enables the connection of up to eight nodes or devices to each H4000 transceiver. The DELNI can also be used in a stand-alone configuration, unconnected to the Ethernet coaxial cable. In this configuration, the DELNI supports up to eight devices (systems, not terminals) using standard Ethernet transceiver cables up to 50 meters away. The DELNI and connected devices form a local area network of their own. The DELNI can also be arranged in a hierarchical stand-alone configuration that is not connected to a main Ethernet segment. Here, the DELNI supports up to 64 devices using Ethernet transceiver cables up to 50 meters away. Configurations using the DELNI are cost-effective for LANs that span a limited area. When an

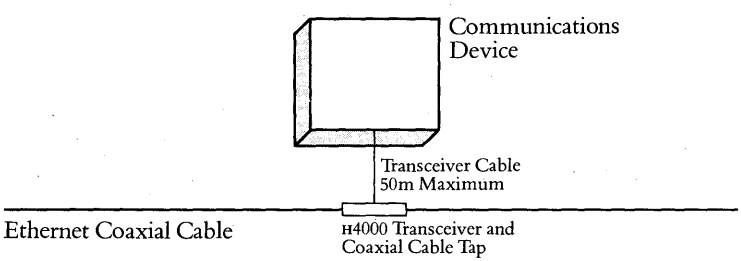


Figure 3.4
Ethernet Hardware

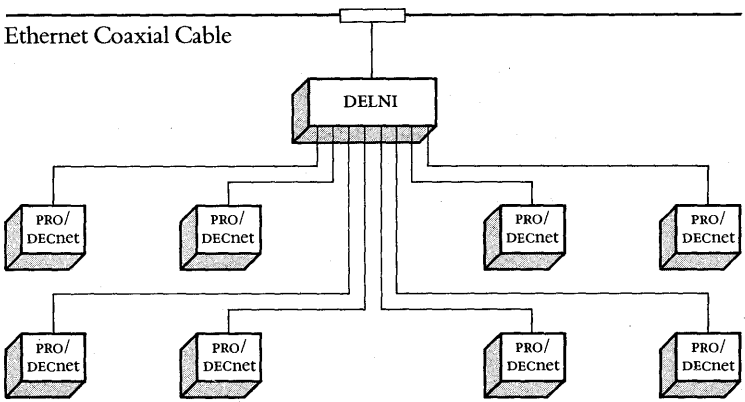


Figure 3.5
DELNI

organization's networking needs grow, a larger Ethernet LAN can be easily implemeted by connecting the DELNI configuration to a coaxial cable via an H4000 transceiver. Figure 3.5 shows PRO/DECnet nodes on a DELNI that is attached to an Ethernet cable.

Segments of coaxial cable can be connected to extend an Ethernet LAN beyond the 500-meter limit of a single segment. Two cable segments are joined using a device called a repeater (see Figure 3.6). A repeater enables the two cable segments to function as if they were one cable. It amplifies transmission signals and passes data packets between two coaxial cable segments without filtering signals. Digital local area networks can use two types of repeaters:

Local repeater–Connects two coaxial cable segments within a limited geographical distance (within a building, for example).

There can be a maximum of 100 meters between the two cable segments, and each cable segment can be a maximum of 500 meters long.

Remote repeater–Connects two coaxial cable segments spanning a large distance (between two buildings, for example). The remote repeater consists of two local repeaters connected by a fiber-optic link up to 1,000 meters long.

Both local and remote repeaters connect to an H4000 transceiver on the coaxial cable using transceiver cables (see Figure 3.6). No more than two repeaters can be placed between any two nodes on a single Ethernet.

Other characteristics of Ethernet configurations are:

- Transceiver cables connecting nodes to the Ethernet cannot exceed 50 meters.
- Cable segments must be terminated in 50 ohms.
- The maximum distance between any two nodes is 2,800 meters (this includes the length of the Ethernet cable, the transceiver cables, a local repeater, and a remote repeater).

All DECnet nodes directly connected to an Ethernet cable must be Phase IV nodes. However, Phase III nodes or Phase IV nodes that are not on the Ethernet can gain access to Ethernet resources through a DECnet Router Server, a DECnet Router/X.25 Gateway, or a full-function Ethernet node acting as a Phase IV Router.

All the server nodes described below are supported on the Ethernet. (There is a DECnet/SNA Gateway implementation that supports DECnet/SNA communications in a wide area network configuration as well.) Server nodes are special purpose nodes on

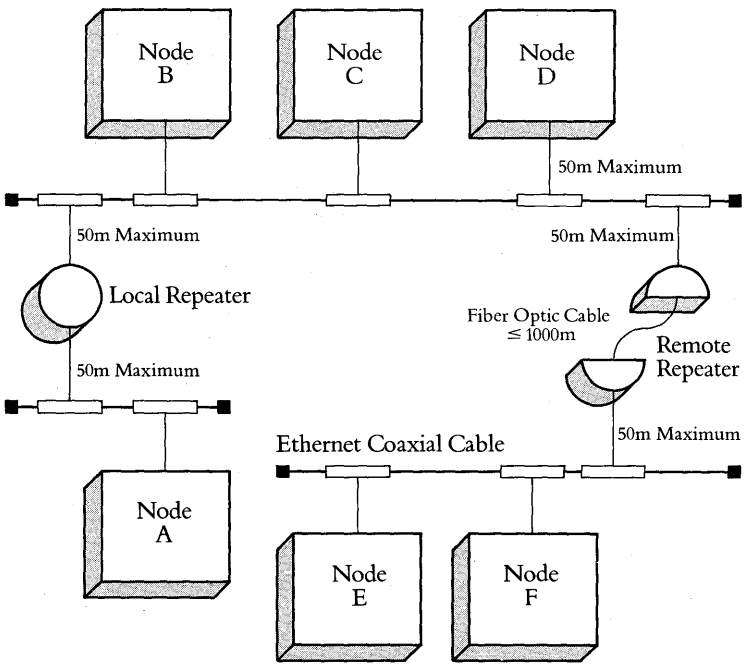


Figure 3.6
Repeaters

the Ethernet that perform communication functions for the other nodes, thereby freeing the latter of communication processing. Server nodes are identical in hardware design. The software that distinguishes them and prepares them for their special communications functions is downline-loaded from a designated host node on the same Ethernet LAN. A host node is a full-capability DECnet node whose services are required in the startup procedures or

operations of another node. Documentation for each server node includes installation information describing the requirements of a host node.

One type of DECnet system, PRO/DECnet, can participate in network activities only as an Ethernet node. (PRO/DECnet software enables a Professional 350 series personal computer to participate as a node in a DECnet Ethernet network. PRO/DECnet is distinguished by a menu-driven user interface. Refer to the PRO/DECnet manual set for more specific information.) Once connected to an Ethernet cable, however, PRO/DECnet can access all the resources available to any other Phase IV DECnet node. Routers and servers enable the Professional 350 running PRO/DECnet to communicate with systems on another Ethernet and with remote Phase IV and Phase III systems not on an Ethernet network (see Figure 3.9).

Nodes connected to an Ethernet cable can connect with wide area networks, other local area networks, packet-switched data networks that implement the X.25 protocol, and systems in IBM SNA networks. Figure 3.9 shows that X.25 and DECnet/SNA Gateways can connect directly to the Ethernet cable.

For more information on Ethernet local area networks, refer to the following Digital manuals:

- *Introduction to Local Area Networks*
- *Networks and Communications Buyer's Guide*

- **Node Characteristics**

Characteristics of a node that affect network configurations are whether it supports Phase III or Phase IV DNA implementation, what its routing capabilities are, and whether it is a dedicated or general purpose node.

Phase III And Phase IV Nodes

DECnet products have been developed in a series of phases. Not all Digital operating systems implement the same phase of DECnet, nor do they all implement the full capabilities of each phase. Chapter 1 discusses the DECnet phases and capabilities currently supported by each Digital operating system.

In planning a network configuration, the phase of DECnet that is installed on a node is important, since Phase IV DECnet extends the networking capabilities of a node beyond what is offered in Phase III. Only Phase IV nodes can connect directly to an Ethernet cable. Phase III nodes can communicate with Ethernet nodes through a DECnet Router Server or a Phase IV node. The Phase IV area routing mechanism supports networks of over 64,000 nodes (refer to *Routing* in Chapter 2 for details), while Phase III supports up to 255 nodes.

Communications servers, described below, all implement Phase IV software. Servers are important to configuration plans because they offload functions, such as routing decisions (Chapter 2) and virtual terminal capabilities (Chapter 7), that are normally performed by a general function node. An Ethernet LAN configured with a DECnet Router Server could support a number of nodes without routing software (such nodes are called *end nodes*). The router would provide routing functions for these end nodes.

Throughout this handbook, differences between Phase III and Phase IV will be identified and discussed as they affect various aspects of network operations.

Routing Capabilities

From the point of view of routing capabilities, there are two types of DECnet nodes: end nodes and full-function nodes.

End node–End nodes do not have route-through capability. This means that they cannot receive and forward messages intended for other nodes. An end node can send packets to an adjacent node. If an end node has multiple circuits to one or to several adjacent nodes, only one of those circuits can be active at a time.

If a node adjacent to an end node is a full-function node, its route-through facilities can be used by the end node to establish connections with other nodes in the network. Nodes A, D, and H in Figure 3.7 are end nodes.

Full-function node–A full-function node can send packets from itself to any other Phase IV node, regardless of its location in the network. It can receive packets from any other Phase IV node and it can route packets intended for other nodes. Full-function nodes can have multiple circuits actively communicating with one or several nodes at the same time. Full-function nodes also maintain an up-to-date map of the network in their database. The map provides routing information, such as the node address of each accessible node in the network and circuit costs. This information is used by the routing mechanism to determine the most cost-effective paths for sending data to a destination node. Data destined for a node that is not adjacent to the sending node can be routed though one or more full-function node. Nodes B, C, E, F, and G in Figure 3.7 are full-function nodes.

Level 1 and Level 2 routers in area-based networks are full-function nodes. A Level 1 router routes packets between nodes within its area. When it receives a packet that is intended for a node in another area, the Level 1 router forwards the packet to the nearest Level 2 router in its area. The Level 2 router then routes the packet through other Level 2 routers to a Level 2 router in the destination area, which forwards it to a Level 1 router in that area. Level 1 routers in the destination area forward the packet to the destination node.

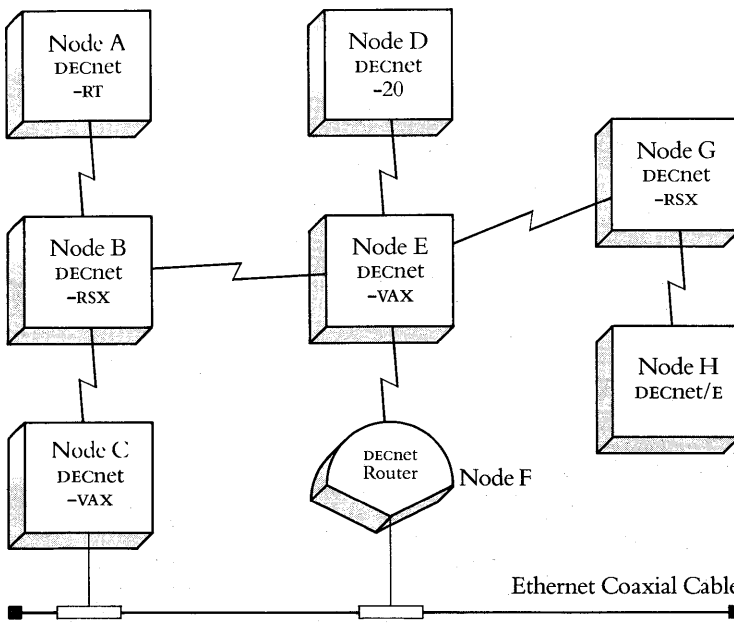


Figure 3.7
Node Routing Capabilities

General Purpose And Dedicated Nodes

Also affecting network configuration is whether or not particular nodes in the network are general function or special purpose. Dedicated or special purpose nodes have specific functions that can service the needs of many other nodes. Figure 3.8 shows dedicated DECnet nodes providing routing services to a group of nodes on an Ethernet cable. Also in the figure are gateways that connect Ethernet LANs to X.25 and IBM/SNA networks.

Most of the dedicated nodes discussed in this handbook must be configured on an Ethernet cable. The one exception is an implementation of the DECnet/SNA Gateway, which can also be config-

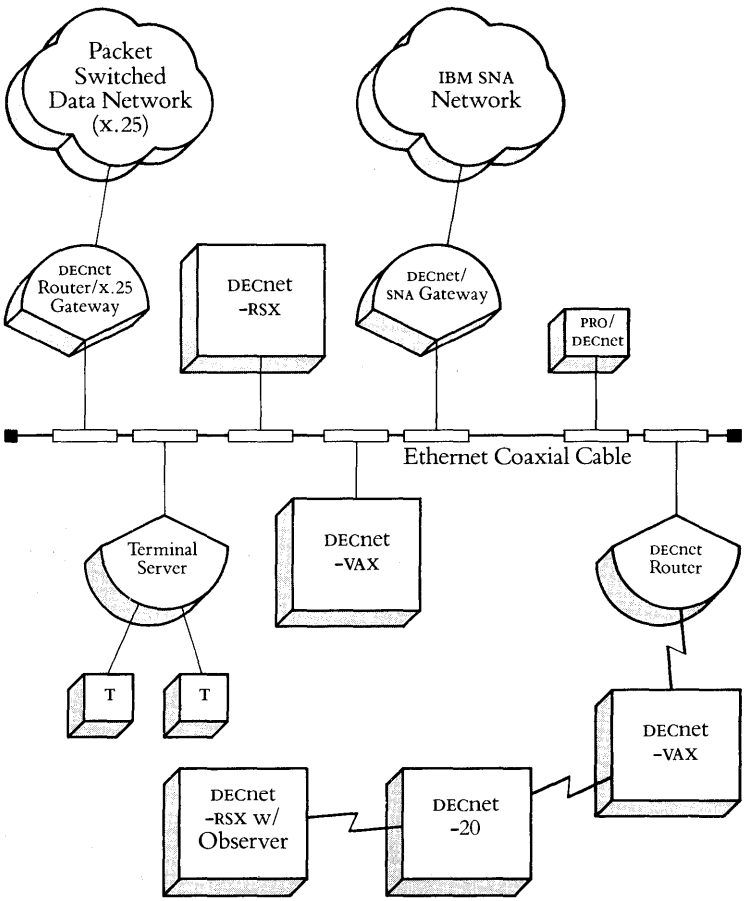


Figure 3.8
General Purpose and Dedicated Nodes

ured in a wide area network. General purpose nodes can be configured in a wide area or local area network environment.

DECnet nodes performing dedicated communications functions are called communications servers. With the exception of the Terminal Server, the communications servers implement Phase IV DECnet software. (Because these servers are dedicated to a specific function, they cannot be accessed as regular Phase IV nodes for general user functions.) The Terminal Server implements the Local Area Transport (LAT) protocol (see Chapter 7 for details).

DECnet communications servers must be downline-loaded by a DECnet host node on the Ethernet cable. (The exception to this rule is the wide area version of the DECnet/SNA Gateway. A DECnet/SNA Gateway intended for a wide area network can be located in a wide area network and loaded directly from a floppy disk.) DECnet communications servers perform the following functions:

DECnet Router Server–This is Phase IV DECnet node that performs routing functions only. The DECnet Router Server handles the routing of traffic between nodes on two Ethernets and between nodes on an Ethernet network and remote Phase IV or Phase III nodes that are not on the Ethernet. The DECnet router offloads routing functions from general function host nodes so that their resources can be used more efficiently to support user applications.

Terminal Server–This server enables the connection of several terminals to a host system on an Ethernet network (see Chapter 7 for details on the LAT protocol operating over the Ethernet terminal-host link). Through the use of a single command, users on terminals connected to this server can establish a nonblocked logical connection to any Phase IV node on the Ethernet network.

Since the terminal server provides access to multiple hosts, it potentially reduces the number of direct terminal connections to individual hosts within a LAN. Thus, power and packaging

requirements on the hosts are reduced. The terminal server allows terminals to be distributed close to where they are used, thereby reducing cabling costs and complexity. By locating the terminal server with the terminal users, the task of managing the multiple cables running from the terminals to the server can be simplified.

Network reliability increases with the terminal server. When a host with many terminal connections fails, all the terminals connected to it become inoperative. However, if a host to which terminals are logically connected via a terminal server fails, the terminals are not inoperative and can access other hosts on the network.

DECnet Router/X.25 Gateway–This gateway makes possible communication between DECnet nodes on an Ethernet and DECnet nodes connected to an X.25 PSDN. DECnet-VAX nodes on an Ethernet can communicate through this gateway with non-Digital systems that support the X.25 protocol and with asynchronous terminals that are physically connected to an X.25 network (see Chapter 8 for details on additional XEP software that the DECnet-VAX nodes require). And terminals on the Ethernet that are logically connected to VAX systems with XEP software can access remote systems connected to a PSDN. Through the gateway, every node on the Ethernet network has access to DECnet nodes on the X.25 PSDN, minus the expense and problems of reliability associated with direct connection of every system to the X.25 network.

The DECnet Router/X.25 Gateway also provides users with all the capabilities of the DECnet Router for making point-to-point connections between systems observing the Digital Data Communications Message Protocol (DDCMP).

DECnet/SNA Gateway–This server enables users to derive the complementary benefits of Digital and IBM computing environments. There are two DECnet/SNA gateway products: one for wide

area networks and one for local area networks. Through either of the DECnet/SNA Gateway products, Digital systems on an Ethernet or non-Ethernet DECnet network can communicate with IBM host systems in a Systems Network Architecture (SNA) network. Through the DECnet/SNA gateway, users within a DECnet network can perform Remote Job Entry, 3270 Terminal Emulation, and Application Program Interface (for task-to-task communication) into the IBM SNA network. These functions can be easily installed and integrated into existing user procedures, thereby providing a transparent and bidirectional flow of information between the Digital and IBM networking environments. A gateway management capability enables a DECnet network manager to monitor and control gateway performance.

General purpose nodes perform a variety of high-level network functions (file transfer, remote resource access, task-to-task communication, and network virtual terminal, for example) that are discussed in Chapters 4 through 9 of this handbook. All Phase IV general purpose nodes, with the exception of a PRO/DECnet node, can run in either a wide area or an Ethernet environment, unless otherwise specified.

- **Line and Circuit Characteristics**

Network nodes in a wide area network can be connected using *point-to-point* or *multipoint* lines and circuits. Ethernet nodes are connected by means of a bus topology. The type of line/circuit between two nodes can have a marked effect on access capabilities. For instance, data can travel faster on a point-to-point line/circuit than over a multipoint line/circuit.

A point-to-point line connects two nodes using a single circuit. In a point-to-point connection, nodes always have access to the communications path linking them.

A multipoint line is shared by more than two nodes (see Figure 2.1). Each node communicates over the line using a separate circuit.

One node on the line, called the control station, controls access to the shared communications path; the other nodes on the line are known as tributaries.

A channel-access method known as polling determines when each tributary can use the communications path in a multipoint connection. In polling, the control station sends a message to each tributary in an order prescribed by the control-station software. The poll message asks if the tributary has any data to send. If the tributary does have data to send, the control station allows the tributary to transmit. When transmission is completed, the communications path is freed, and the control station polls the next tributary. If a tributary has no data to send when it is polled, the control station passes it and polls the next tributary. If data is sent to a tributary from another tributary or system in the network, it must first pass through the control station, which delivers the data (using the routing mechanism) to the specified tributary when the communications path is freed.

A multipoint connection can be more economical than a point-to-point one, as there are several stations communicating over a single line. However, a node may not always have access to the communications path when needed.

Chapter 2 discusses the access method and transmission and reception of data over Ethernet links.

- **The Total Picture**

Figure 3.9 shows an example of a large network that includes a wide area network and a local area Ethernet network. In such a network environment, there are a variety of configurations that can be implemented to serve a wide range of application needs. All nodes in the network can communicate with each other and combine their resources to perform network activities as an active, cohesive unit.

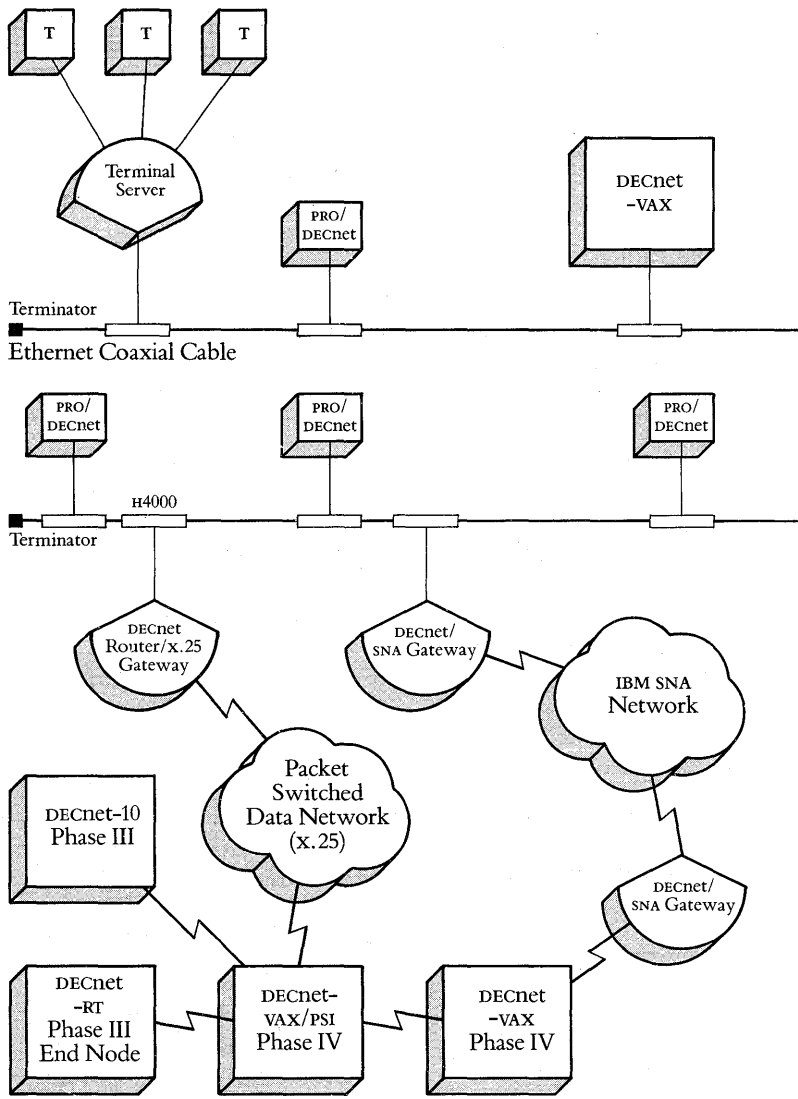
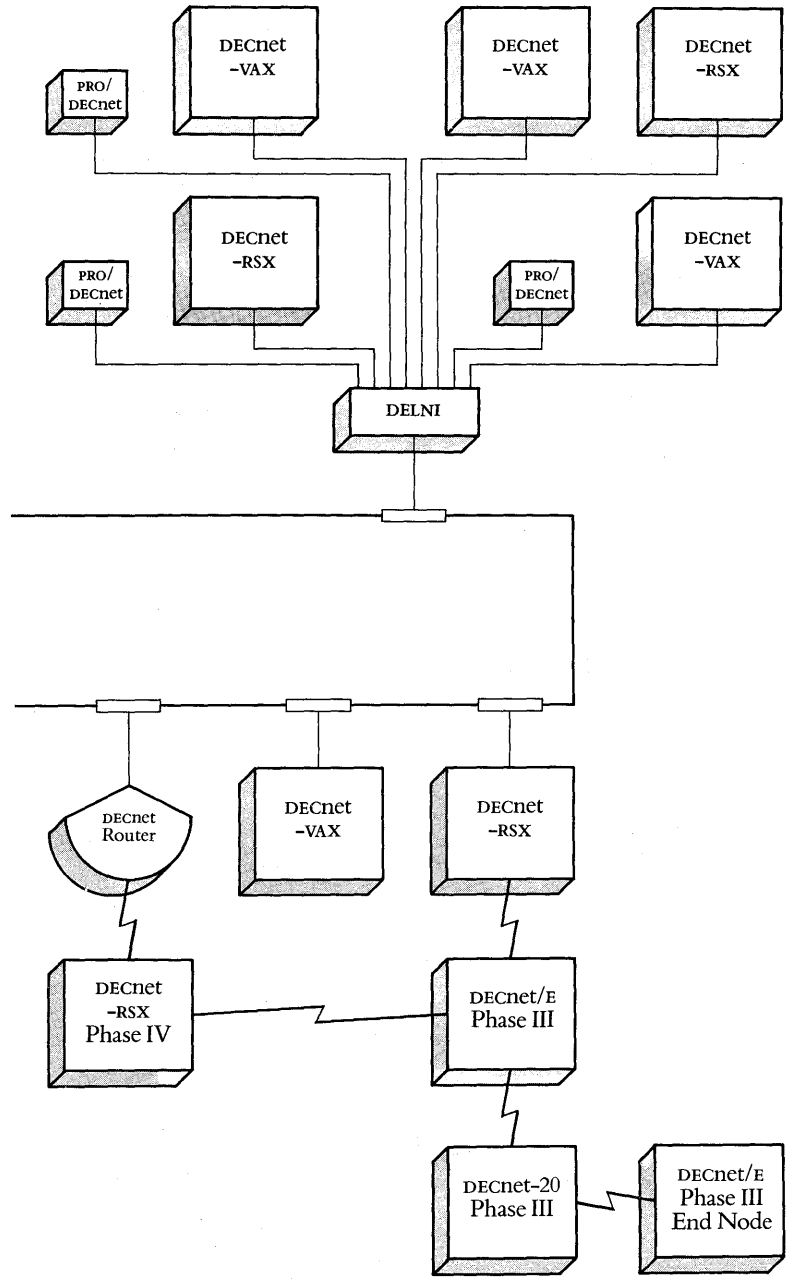


Figure 3.9
The Total Picture


DECnet
-VAX
DECnet
-VAX
DECnet
-RSX
PRO/
DECnet
DECnet
-RSX
DECnet
-VAX
PRO/
DECnet
PRO/
DECnet
DELNI
DECnet
Router
DECnet
-VAX
DECnet
-RSX
DECnet
-RSX
Phase IV
DECnet/E
Phase III
DECnet-20
Phase III
DECnet/E
Phase III
End Node

Once a DECnet system has been configured in a network (Chapter 3) and installed (Chapter 10), it is available to users for a variety of high-level network functions, including:

- Task-to-task communication (Chapter 5)
- Remote file and record access and file transfer (Chapter 6)
- Terminal communications (Chapter 7)

Chapter 8 describes how users can perform these functions over foreign-vendor networks, and Chapter 9 describes application environments that implement these functions.

All of the high-level network functions listed above have common mechanisms at work within them, including logical links, segmentation and reassembly of data, error control, flow control, and access control. This chapter discusses these mechanisms.

- **Logical Links**

Every node runs multiple programs and processes. Whether they are in the same node or in different nodes, processes or programs that have to communicate with each other need some way of establishing contact and exchanging data, unhindered by the confusion and interference of other network traffic. To facilitate the orderly flow of information between two programs or processes, DECnet implements a mechanism called a logical link. It is a *temporary* conversation path established between two communicating programs (or processes) in a DECnet network. The communicating programs can reside on nodes located anywhere in the network, since DECnet establishes logical links between nonadjacent as well as adjacent nodes. (Do not confuse logical links with lines or circuits. A line is the physical medium over which data travels; a circuit, which operates over lines, is a logical communication path between adjacent nodes. See Chapter 2.) Figure 4.1 shows typical logical links.

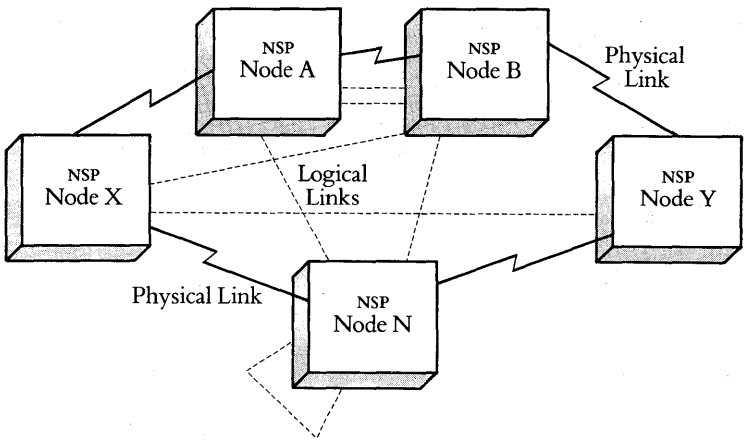


Figure 4.1
Logical Links

When Are Logical Links Required?

Almost all network functions, including those discussed in Chapters 5, 6, 7, and 8, require the services of a logical link between programs. The exceptions are some maintenance functions, such as loopback tests and the downline-loading of certain systems (downline-loading of an RSX-11S DECnet node, for example). Logical links operate in the following types of connection:

- A user program connected to another user program
- A user program connected to a DECnet module
- A DECnet module connected to another DECnet module

In the first type of connection, the application programmer specifies a series of DECnet calls in each user program to establish and direct the operation of the logical link. In the second type of

connection, the programmer includes calls that initiate and control the link in the user program only; the DECnet module automatically handles its end of the link. In the last case, the creation and operation of the logical link is a level removed from the user; user programming is not involved in the connection at all. The two DECnet modules exchange messages based on information supplied by their local system.

How To Create A Logical Link

Logical links are created by an interaction of DECnet calls specified in the two cooperating programs. This interaction is referred to as the "handshake procedure." In this procedure, DECnet calls are passed between the two programs to establish an *agreement to communicate*. Unless both programs agree, a logical link cannot be created.

The handshake procedure takes place in a prescribed order. First, one program, the source program, issues a call to request communication with a second program, the target program. DECnet software in the source program's node carries the connection request to DECnet software in the target program's node. (If the two programs are in the same node, the DECnet software in that node alone is involved in establishing the connection.)

The target program has the option of accepting or rejecting the source program's request for connection. If the target program returns a call accepting the connection request, a logical link is established between the two programs. At this point, source and target distinctions become irrelevant, and both programs can issue calls to send and receive data until either program decides to exit or abort the connection. If the target program returns a call rejecting the connection request, no logical link is created.

The calls required to establish a logical link and to exchange data with another program are described in the programmer's reference manual for each DECnet system. Also listed in the manual are the programming languages that can be used to to issue these calls. Chapter 9 discusses logical link processing in an applications environment.

Behind The Scenes—DECnet Software And Logical Links

The DNA levels primarily responsible for affecting process-to-process communication through logical links are the End-to-End Communication and Session Control layers. End-to-End Communication handles the *system-independent* aspects and Session Control the *system-dependent* aspects of process-to-process communication.

End-to-End Communication Layer and NSP

The modules of the End-to-End Communication layer are responsible for the creation and destruction of logical links. The information passed between two programs in the handshaking procedure and in all communications activity is intercepted and evaluated by the End-to-End Communication modules in each node. Information passes from the local program to the local End-to-End Communication software, from there to the remote End-to-End Communication modules, and finally to the remote program (as shown in Figure 4.2).

The End-to-End Communication modules in the local and remote nodes communicate through the Network Services Protocol (NSP). NSP functions include:

- creation and destruction of logical links.
- guaranteeing the delivery of data and control messages in sequence to a specified destination by means of an error control mechanism.

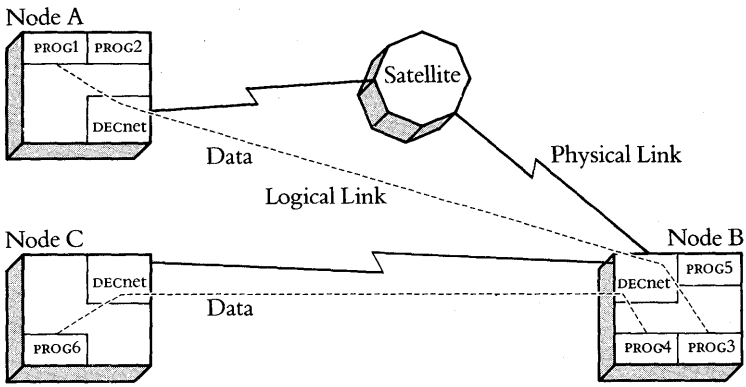


Figure 4.2
DECnet and Logical Links

- managing the movement of interrupt and normal data (see under *Data Types,* below) from transmit buffers, using flow control mechanisms (see below, under *Flow Control*).
- breaking up normal data messages into segments that can be transmitted individually and reassembling these segments in proper sequence upon reception (see *Segmentation and Reassembly of Data,* below).

NSP Messages–There are three types of NSP messages–data, acknowledgment, and control. Table 4.1 summarizes the functions that each NSP message type performs.

NSP establishes, maintains, and destroys logical links by exchanging control messages with other NSP modules or with itself (if the communicating programs or processes are in the same node).

Type	Message	Description
Data	Data Segment	Carries a portion of a Session Control message. (This has been passed to Session Control from higher DNA layers and Session Control has added its own control information, if any.)
Data (also called Other Data)	Interrupt	Carries urgent data, originating from higher DNA layers. It also may contain an optional Data Segment acknowledgment.
	Data Request	Carries data flow control information and optionally a Data Segment acknowledgment (also called Link Service message).
	Interrupt Request	Carries interrupt flow control information and optionally a Data Segment acknowledgment (Link Service message).
Acknowledgment	Data Acknowledgment	Acknowledges receipt of either a Connect Confirm message or one or more Data Segment messages, and optionally an Other Data message.
	Other Data Acknowledgment	Acknowledges receipt of one or more Interrupt, Data Request or Interrupt Request messages, and optionally a Data Segment message.
	Connect	Acknowledges receipt of a Connect Initiate message.
Control	Connect Initiate and Retransmitted Connect Initiate	Carries a logical link connect request from a Session Control module.
	Connect Confirm	Carries a logical link connect acceptance from a Session Control Module.
	Disconnect Initiate	Carries a logical link connect rejection or disconnect request from a Session Control module.
	No Resources	Sent when a Connect Initiate message is received and there are no resources to establish a new logical link (also called Disconnect Confirm message).
	Disconnect Complete	Acknowledges the receipt of a Disconnect Initiate message (also called Disconnect Confirm message).
	No Link	Sent when a message is received for a nonexistent logical link (also called Disconnect Confirm message).
	No Operation	Does nothing.

Table 4.1
NSP Messages

Figure 4.3 shows a typical message exchange. In this figure, an NSP module first initiates a connection, then sends data, and finally disconnects the link based on commands from the Session Control layer. The figure does not show the messages that control data flow (these are illustrated in Figure 4.7, below).

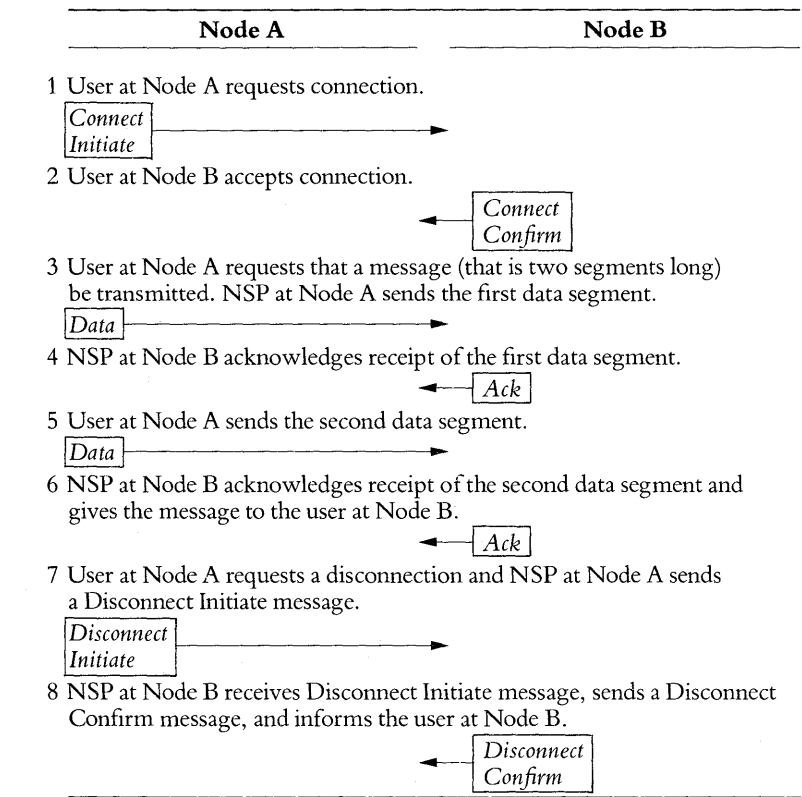


Figure 4.3
Typical Message Exchange between NSP Modules in Two Nodes

Other logical-link functions performed by End-to-End Communication modules on each node include:

- Establishing link addresses to identify each end of a logical link (see below, under *How the Program Identifies Logical Links)*
- Formatting outgoing data for transmission by communication hardware
- Allowing multiple logical links to be created over a single physical line by separating data into logical-link streams (see under *Multiple Logical Links Within a Program,* below)
- Removing incoming data from logical-link streams and delivering it to the appropriate program.

Session Control Layer

The modules in this layer provide *system-dependent* process-to-process communication functions, thereby bridging the gap between the End-to-End Communication layer and the logical-link functions required by processes running under an operating system.

Session Control functions include:

- Mapping node names to node addresses–A Session Control module maintains a node-name mapping table or database that lists the names of nodes with which the local node expects to communicate, as well as the unique network address of each node name in the table. When a local end-user process requests connection to an end-user process on a remote node, the Session Control module consults the table to select the node address of the target program or the number of the physical channel on which the connect request will be transmitted.
- Requesting logical links on behalf of end-user processes (see below, under *Requesting a Connection* for details)–The Session

Control layer passes destination node address and channel number to the End-to-End Communication layer when making logical-link requests for end-user processes.

- Receiving connect requests addressed to end-user processes (see under *Receiving a Connect Request,* below)–For incoming connect requests from the End-to-End Communication layer, a Session Control module uses the node-name mapping table to identify the node from which the request originated. The identity of a node is important in determining the validity of a connect request (see below, under *Access Control).*
- Sending and receiving logical link data–Requests by end-user processes to send and receive data are passed directly by the Session Control layer to the End-to-End Communication layer.
- Disconnecting and aborting a logical link–As in the case of sending and receiving data, requests by end-user processes to disconnect and abort logical links are passed directly by the Session Control layer to the End-to-End Communication layer. When a logical link has been disconnected or aborted, the Session Control layer notifies the end-user process of the event.
- Optionally, monitoring logical links–This function can be used at either end of the logical link to detect a probable network disconnection. The Session Control layer can also optionally detect failure on the part of the End-to-End Communication layer to deliver received data in a timely manner.
  The Session Control layer defines a database containing the states of Session Control and optional default connection timers.
- Identifying end-user processes–A Session Control module executes a system-dependent algorithm to determine if an existing end-user process corresponds to the destination end-user process specified in an incoming connect request. It also performs additional functions related to passing a connect request to an existing end-user process.

- Activating or creating processes–A Session Control module can create a process or activate an existing one to handle an incoming connect request.
- Validating incoming connect requests–A Session Control module uses access-control information included in an incoming connect request to perform system-dependent validation functions (see below, under *Access Control*).

Figure 4.4 shows the relationship of the Session Control modules to end-user processes and to the End-to-End Communication modules.

Requesting a Connection

When it receives a logical-link connect request from an end-user process, the Session Control layer performs four tasks:

- Identifies the destination node address or channel number for the End-to-End Communication layer by using the node-name mapping table.
- Formats connect data for the End-to-End Communication layer.
- Issues a connect request to the End-to-End Communication layer.
- Optionally, starts an outgoing connection timer. If the timer expires before the End-to-End Communication layer issues an acceptance or rejection of the connect request, Session Control disconnects the logical link for the source end-user process.

Receiving a Connect Request

When the Session Control layer detects an incoming connect request from the End-to-End Communication layer, it performs six tasks:

- Parses connect data to obtain such information as source and destination end-user process names and access-control information.

	A Network Node	Comments
User Processes	End User 1, End User 2, End User 3	End users are User, Network Application, and Network Management modules.
System Dependent Modules	Operating System ↔ Session Control –① –②	Session Control is an interface to the End-to-End Communication layer for end user processes. It functions in conjunction with the operating system. ① and ② are data bases used by Session Control.
System Independent Modules	End-to-End Communication Layer ↕ The Network	The End-to-End Communication layer provides the logical link service to Session Control. Its functions are not dependent on individual operating systems.

*Network Management interfaces with Session Control in two ways: (1) to obtain access to the logical link service and (2) to monitor and control Session Control operations.

Figure 4.4
Session Control Interaction with End-User Processes and End to End Communication

- Validates any access-control information.
- Identifies, activates, or creates a destination end-user process.
- Maps the source node's address or channel number to a node name, if there is one.
- Delivers the incoming connect request to the end-user process.
- Optionally, starts an incoming timer when the connect request is delivered. If the timer expires before the end-user process accepts the connect request, the Session Control layer issues a reject to the End-to-End Communication layer.

Session Control Messages

Session Control's message protocol defines messages sent on a logical link as connect data, reject data, and disconnect data. Figure 4.5 shows the formats for the Session Control messages. The numbers below each message field indicate its maximum length in bytes.

How The Program Identifies Logical Links

A program can establish and use more than one logical link at a time. The maximum number of links that can run simultaneously in a single program is usually determined by the programmer or by system restrictions. (The programmer's reference manual for a system includes details on the logical-link requirements of that system.) Since a program can use multiple logical links, there must be some way for it to differentiate among the many logical links that may be running within it. DECnet effects the differentiation by having each program specify a unique link identifier during every handshake procedure to identify the link that may be created. The link identifier defines a link only within the context of a particular program; two cooperating programs do not have to specify the same link identifier for the same link.

Connect Data Message Format

DSTNAME	SCRNAME	MENUVER	RQSTRID	PASSWRD	ACCOUNT	URSDATA
19	19	1	39	39	39	39

Reject/Disconnect Data Message Format

REASON	DATACTL
1	n

DSTNAME = the destination end user name
SRCNAME = the source end user name
MENUVER = the field format and version format
RQSTRID = the source user identification for access verification
PASSWRD = the access verification password
ACCOUNT = the link or service account data
USRDATA = the end user process connect data
REASON = a reason code
DATACTL = user data (length of field determined by the total length of reject or disconnect data received from the End-to-End Communication layer)

Figure 4.5
Session Control Message Formats

End-to-End Communication software on each node takes the link identifier specified by its local program, and both sets of End-to-End Communication modules agree on a pair of link addresses that they associate with a particular link. Link addresses are meaningful only to the End-to-End Communication modules; they are transparent to the end user.

Multiple Logical Links Within A Program

A single program can establish logical links that communicate with several programs, or establish several logical links with the

same program to exchange data intended for different purposes. For example, two programs can establish two logical links between them; one link can be used to transmit transaction data, while the other can be used to transmit control information.

Data Types Within a Logical Link

A logical link can be considered to comprise two separate data *subchannels,* each carrying messages in both directions (a logical link operates in full-duplex mode). One data subchannel carries *normal data,* which is the subject matter of most data exchanges between two programs. User data, such as a list of messages and the responses sent by the receiving program, would constitute normal data. The other data subchannel carries *interrupt messages, data request messages,* and *interrupt request messages.*

To interrupt the flow of normal data, either program can send interrupt data. Interrupt data is high-priority information signaling the occurrence of some event requiring immediate attention. Interrupt data breaks through the normal data flow. The means of delivering interrupt data are system- and program-dependent, but software in the receiving node usually reads it before accepting any normal data that may be waiting to be delivered.

- **Segmentation and Reassembly of Data**

The End-to-End Communication modules that manage each end of a logical link guarantee:

- that all transmitted data is delivered to the destination node in proper sequential order.
- that all data received by the NSP module on the destination node is given to the target program in the proper sequence.

The Routing layer limits the amount of user data that an NSP module can send in one datagram (data without routing information is known as a datagram). Taking normal data from Session Control buffers, the transmitting NSP module breaks it up, if necessary, into segments. To guarantee proper segment sequencing, the NSP module numbers the segments and transmits them along with other control information in a Data Segment message over a link. The receiving NSP module uses the sequence numbers to reassamble the data segments in correct sequence in the receiving node's Session Control buffers.

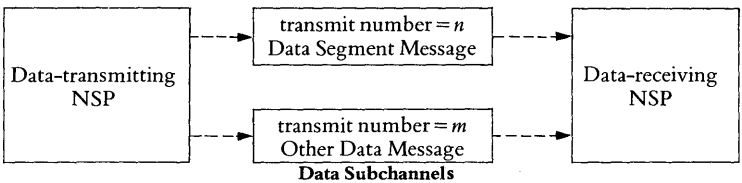
End-to-End Communication software assigns a separate set of transmit numbers to interrupt messages. (Interrupt messages are not segmented, however, because interrupt data is limited in size and always fits in a single datagram). The discrete sets of numbers for normal and interrupt data logically divide them into separate data streams within the logical link.

Error Control

The receiving NSP module acknowledges delivery of the segments by using the transmit numbers for identification. Optionally, a receiving NSP module can negatively acknowledge normal data it has received, if the data has to be discarded (from being irreparably out of order, for example). If the transmitting NSP module receives a negative acknowledgment or fails to receive a positive acknowledgment within a certain period of time, it retransmits the data.

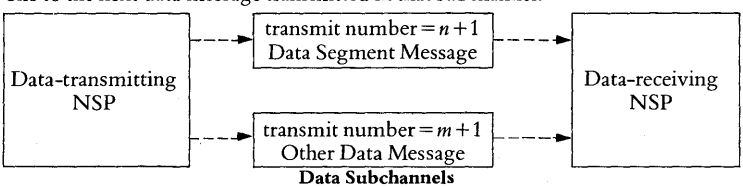
There are variations of detail among the several implementations of DECnet in the execution of segment acknowledgment and retransmission. However, despite the differences, all the DECnet implementations use these mechanisms to guarantee delivery of all transmitted segments and to ensure that the segments are delivered in the proper sequence. Figure 4.6 illustrates data segmentation and acknowledgement.

1 The data-transmitting NSP assigns a transmit number to the message, and starts a timer.



2 If the timer times out, the message is retransmitted.

3 If the timer does not time out, and the flow control mechanism allows another message to be sent, the data-transmitting NSP assigns the transmit number plus one to the next data message transmitted in that subchannel.



4 When the message with the first transmit number is received by the data-receiving NSP, it returns that number as an acknowledgment number within the first acknowledgment.

5 If the next data message transmit number received is *equal to the current acknowledgment number plus one*, the data-receiving NSP accepts the data message, incrementing the acknowledgment number. It then sends the new receive acknowledgment number back to the data-transmitting NSP within an acknowledgment message.

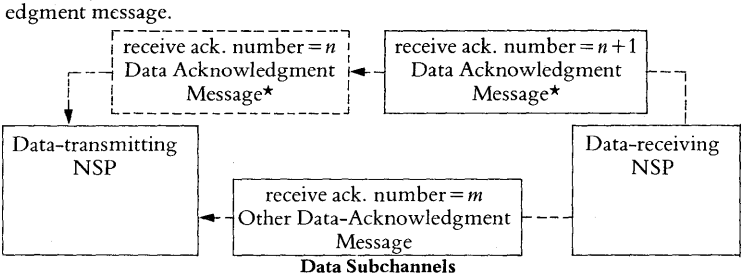


Figure 4.6
Acknowledgement Operation

(Continued)

6 However, if the data-receiving NSP receives a data message transmit number *less than or equal to the current receive acknowledgment number* for that subchannel, the data segment is discarded. The data-receiving NSP sends an acknowledgment back to the data-transmitting NSP. The acknowledgment contains the receive acknowledgment number.

7 If the data-receiving NSP receives a data message transmit number *greater than the current receive acknowledgment number plus one* for that subchannel, the data segment may be held until the preceding segments are received or it may be discarded.

---

*The data-receiving NSP might not send an acknowledgment for each data message received. The receive acknowledgment number implies that all previous numbers were received.

• Flow Control

Network programs and DECnet software both require a certain amount of buffer space for temporary message storage. For example, a transmitting NSP module keeps a copy of every message it sends over a link until the receiver acknowledges receipt of the message. At the program level, buffer space is necessary to hold inbound messages waiting to be processed. DECnet software and programs need buffer space for other purposes as well, depending on the application and the DECnet implementations.

Without some kind of control, message traffic can easily cause available buffer space to overflow. If this happens, communications overhead is incurred, because a message must be present at specified intervals until the destination node can accept it. To prevent this, the programs and NSP modules exercise flow control.

NSP's flow-control mechanisms ensure that data is not lost for lack of buffering capability and that deadlocks do not occur. Both normal and interrupt data are subjected to flow control.

The data receiving part of NSP controls data flow. When a logical link is formed, the NSP modules in each node inform one another of the manner in which they, as data receivers, want to control the flow of normal data. The receiving NSP module can choose not to implement flow control or it can choose one of two types of normal data flow control:

- Segment–the receiver sends a request count of the number of segments it can accept. See Figure 4.7 on Segment Flow Control.
- Message–the receiver sends a request count of the number of Session Control messages it can accept. (Message flow control is obsolescent and will be eliminated at a future date).

The transmitting NSP module uses the receiver's segment request count to determine whether to send data. As an added control, the receiver can always tell the transmitter to stop sending data unconditionally or to start sending data under the normal request count conditions. The receiver also controls interrupt data flow with an interrupt request count.

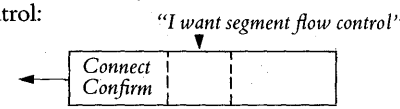
- **Access Control**

The Session Control layer in most DECnet nodes performs access-control functions by examining logical-link requests and preventing unauthorized access of node resources. A program or process on one node that wishes to communicate with a program or process on a remote node must specify information similar to the information needed to log onto a remote system. For example, a program may have to specify an authorized user name or account and password in order to gain entrance to a task on a remote node. Session Control software at the remote node verifies the information and decides whether or not to allow access. In task-to-task communication, users must specify access-control information in

NSP/Node A Data-Transmitting	NSP/Node B Data-Receiving

1 NSP/Node A sends a Connect Initiate to NSP/Node B:

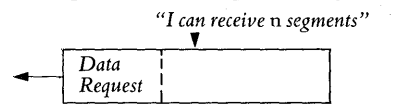
2 NSP/Node B, having received the Connect Initiate message, returns a Connect Confirm message. A field in the message indicates that NSP/Node B expects segment flow control:



3 NSP/Node A's data base has the initial value of 0 for its request count variable for normal data segment flow control:

FLOWrem_dat = 0

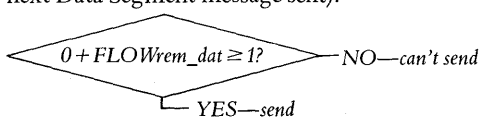
4 NSP/Node B sends a Data Request message containing a flow control value that indicates the number of Data Segment messages NSP/Node B can receive. (NSP/Node B executes an implementation-dependent algorithm to determine this value):



5 NSP/Node A increments FLOWrem_dat by *n:*

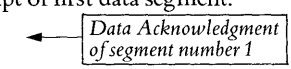
FLOWrem_dat = n

6 NSP/Node A executes algorithm to determine if Data Segment number 1 can be sent (highest acknowledged Data Segment message plus the current request count must be greater than or equal to the number of the next Data Segment message sent):



7 The answer to the above is YES, so NSP/Node A sends Data Segment number 1:

8 NSP/Node B acknowledges receipt of first data segment:



9 NSP/Node A subtracts 1 from its flow control request count:

FLOWrem_dat = n − 1

Figure 4.7
Segment Flow Control Shown in One Direction on a Logical Link

a special parameter when they request a logical link connection (see Chapter 5). In remote file access (see Chapter 6) and in terminal communications, this information must be included in the file specification. (File specifications differ for each type of DECnet system; the programmer's reference manual for a specific DECnet system describes its correct file specification syntax.)

Access-control information may not be required for every network function. To determine what information is needed and when, users can consult the system manager at the remote node to which they are seeking access. Access-control information is initially established during installation, and can be dynamically modified on most systems by the system manager.

- **User Transparency**
  A network user is not aware of the many activities that take place within systems and across links when two programs communicate. The connection between the two programs appears to be direct and without mediation. However, if some problem occurs in the connection, the DECnet software notifies the user by means of an error or status message.

DECnet enables two programs or tasks running on different nodes in a network to exchange data over a logical link. For example, a program in an RSX-11M node can use local DECnet-RSX facilities to communicate with a program running in a DECnet-VAX node. PRO/DECnet does not provide a user interface for task-to-task communication. This chapter may, therefore, not be relevant for PRO/DECnet users.

In most DECnet implementations, performing task-to-task communication is similar to performing input/output (I/O) operations. The logical link between the two communicating programs or tasks is like an I/O channel over which both programs can send and receive data. (Refer to system-specific documentation for details on how a particular operating system implements I/O operations.)

• DECnet Task-To-Task Calls

Programmers include DECnet calls in cooperating programs to enable them to communicate with each other. These calls activate routines requesting the End-to-End Communication modules in each node to perform specific functions, such as the creation and control of a logical link (see Chapter 4 for details on logical links).

The form of the programmer-specified DECnet calls depends on the programming language being used; calls can be actual calls or macros or system directives. However, the DECnet task-to-task capability translates each of these call types into the same set of DECnet messages, regardless of the language in which the calls are programmed. For example, whether a source program issues a connect request in FORTRAN or MACRO, the End-to-End Communication modules in the source node send the same type of message to the End-to-End Communication modules in the target node. Thus, when programmers write task-to-task communication calls into communicating programs, they do not have to be concerned

about what programming languages the remote node supports. They do, however, have to know what languages the local node supports, since not all operating systems support the same languages and not all languages support task-to-task communication. The programmer's reference manual for a particular DECnet system lists the programming languages available on that system for task-to-task communication.

DECnet task-to-task calls in communicating programs perform several functions, including requesting a logical link, receiving a logical-link request, and accepting or rejecting a logical-link request. In addition, task-to-task calls send data, receive data, send interrupt data, and receive interrupt data. A task-to-task call also terminates a logical link.

In some cases, a program must issue three separate calls to create a logical link; in other cases, a program needs to issue only one call. The number of calls required to effect any of the task-to-task functions listed above is dependent on the operating system running on the communicating nodes and the programming language being used.

- **Requesting a Logical Link**

The first step in establishing task-to-task communication is requesting a logical link. A programmer does so by issuing connect request calls in accordance with the language conventions of the source node's operating system (issuing a connect request is the first stage in the handshake procedure described in Chapter 4).

The connect request call must include network addressing information that identifies the node and the program sending the message and the node and the program that should receive it. Programmers can specify this addressing information in the source program either by building a special data area (called a *connect block*) or by including an ASCII string called a *network specification.*

Building a Connect Block-Most DECnet task-to-task call sequences include one or more call to build a connect block. A programmer specifies parameters to the connect block call(s) that identify the source and target programs. These parameters include all or part of the following information:

Link identifier–This identifier differentiates the requested logical link from any other logical link currently being used by the source program. At any given time, a program can have multiple logical links, either with a single program or with several programs. Each active logical link of a program has a unique identifier. End-to-End Communication modules in the communicating nodes use the link identifiers to ensure that data is transmitted over the correct logical link and arrives at its intended destination. See Chapter 4 for further details on logical-link identifiers. If the connect request succeeds, the source program uses the link identifier to address data to be sent over the link. The different operating systems implementing DECnet refer to the source program's link identifier by different names. Check the programmer's reference manual for a specific system to determine what the link identifier is called on that system.

Target node identifier–A target node can be identified either by a unique node address that distinguishes the node from all others in the network or by a node name defined by the source node. A Session Control module on the source node translates the target node's name into a unique node address (see Chapter 2 for additional information on translating node names to node addresses).

Object type or name–Object is another term for a network program. A network program or object has a special identifier by which it makes itself known to the local End-to-End Communication

modules. This identifier consists of an object type and/or an object name (see below under the section on object types and object names for details). Figure 5.1 shows how objects are addressed.

Access-control information–This information describes the source program and includes a user identification, a password, and, optionally, an account number. Access-control information is equivalent to the data a user supplies when logging onto a system. In most DECnet implementations, this information is a factor in the target node's decision to accept or reject the connect request.

Optional user data–A source program usually has the option of sending 16 bytes of data to the target program as part of a logical-link connect request.

When programmers finish building the connect block using the call(s), macro(s), or directive(s) appropriate for the operating system and language being used, they can specify the label or address of that connect block in the source program as a parameter to a connect request call. DECnet requires this network addressing information to respond to logical link requests.

Network Specifications–If programmers do not build connect blocks to provide network addressing information for the connect request, they must provide that information in a network specification. (Some DECnet systems require a network specification instead of a connect block. Refer to the documentation for a DECnet system to find out if and how network specifications must be used.) A network specification is an ASCII string specified in the connect request itself. Like the connect block, it includes a target node identifier, access-control information, and a target object type or name. (DECnet-20 network specifications can include optional data.)

- **Object Types and Names**

A program in a node makes itself known to the local End-to-End Communication software by declaring its object type and name. In most DECnet implementations, a program must declare its object type and name to the local End-to-End Communication software in order to be eligible to receive link requests. (In some implementations, a system manager can use a DECnet command at a terminal to declare a program's object type and name. The programmer's reference manual for a system describes how objects are specified on the system.) The object name can be a special network name for the program or it can be the same name by which the program is known to the local operating system.

In a connect request, the source program uses object types and names to specify the target program with which it wants to communicate. Object types can take one of two formats: an object type equal to 0 and an ASCII name, or an object type equal to a positive integer (from 1 to 255) and a null name. The first format identifies a program by name, and the second format identifies a program by numeric object type.

To address a target program, a connect request specifies either a name or an object type, but not both. The first format—object type 0 plus name—is commonly used to address user-written network programs. To use this format in a connect request, a source program must know the target program's declared object name. Note that the maximum length allowed for a name is dictated by the operating system in which the target program runs.

The second format—an object type equal to a positive integer—provides an abbreviated means of identifying a frequently used network function, usually a DECnet module. A specific object type always represents the same generic function within a network,

even if the program that actually performs the function has a different name at each node. For example, the DECnet module that performs the file access function (FAL) might have a different name at every node, but it can always be identified by object type 17 (decimal) or 21 (octal). Thus, a source program can address the FAL function without knowing the FAL program's name in the target node.

Digital reserves a range of object types for DECnet system programs. Types within this range are used consistently across all DECnet implementations to refer to the same functions. The programmer's reference manual for each DECnet implementation lists the object types reserved for DECnet use.

Unreserved types can be used to identify user-written network programs. For example, in a user-written transaction-processing application, each node might have a resident program for recording statistics on transactions performed within the last 24 hours. An application designer could choose an unreserved object type to identify all such programs throughout the network.

Figure 5.1 illustrates the object identifiers for several programs in Nodes A and B.

- **Accepting/Rejecting a Logical Link Request**
  End-to-End Communication software in the source node forwards the connect request to the node specified in the connect block or network specification. At the target node, End-to-End Communication software first checks that the program addressed in the connect request is a valid object and then verifies the source program's identification. If the target program is a known object and any required verification checks out, the target End-to-End Communication software delivers the connect request to the target program.

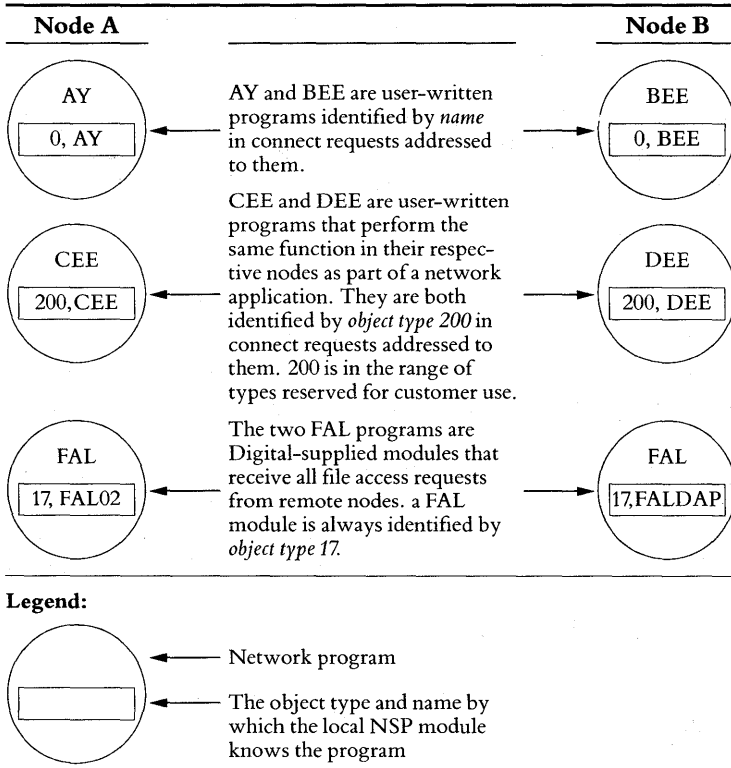


Figure 5.1
Addressing Network Objects

After examining the incoming connect request, the target program either accepts or rejects it. (See system-specific documentation for details on how each system handles connection requests.) End-to-End Communication software in the target node forwards the appropriate response back to the source node. The target program usually has the option of sending 16 bytes of data along with the acceptance or rejection of the link. For example, a response rejecting a connection might include data that tells the source program why the connect request was not accepted. If the target program agrees to the connection, it specifies its own logical-link identifier in a call accepting the connect request.

- **Sending and Receiving Data**

Once the target node accepts the connect request and a logical link has been established, either program can send and receive data using the assigned logical-link identifiers as address parameters in each message. They can send and receive data one at a time or simultaneously.

Programs can send two types of data over a logical link: *normal data* and *interrupt data.* Normal data comprises the subject matter of the dialogue between the programs, and interrupt data conveys special high-priority information.

To convey normal data over the link, a source program issues one or more call to send the data, and a target program issues one or more call to receive it. End-to-End Communication software at the source node will not transmit data unless the target program has already issued a receive call. Each receive call allocates the buffer space that the target program needs to store the data. Figure 5.2 illustrates the various stages of this procedure.

Some DECnet implementations allow the source End-to-End Communication software to transmit data over any logical link, as long as the target End-to-End Communication software has access to enough system buffer space to hold the data. End-to-End Communication software at the target node then delivers the data it has received when the appropriate target program allocates its own buffer space by issuing one or more receive call. Regardless of the implementations involved, End-to-End Communication software in the source and target nodes exchanges link service messages to determine whether the target program is prepared to receive a message. This precaution is part of DECnet's flow-control mechanism (see Chapter 4 for details on flow control).

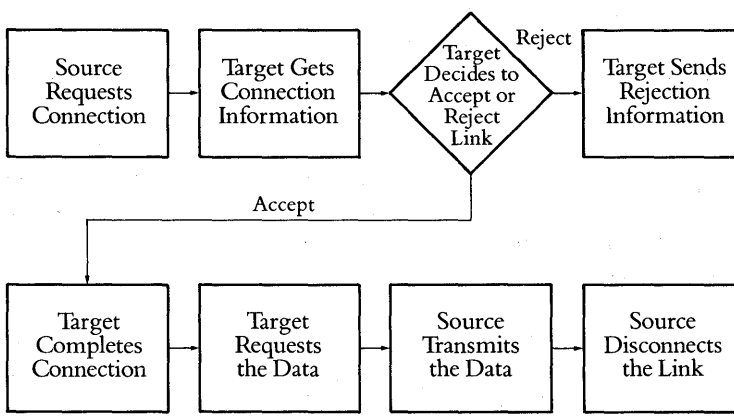


Figure 5.2
Transmitting Normal Data

Interrupt data can consist of up to 16 bytes of information to be delivered immediately to the target program. If End-to-End Communication software in the target node has a queue of normal data that it has already received but not yet processed, the interrupt data is placed either at the head of the queue or in a separate queue that the target program can access without first reading the normal data.

- **Terminating the Link**

Either program can issue a call at any time to terminate the link in one of two ways. One method, which disconnects the link in an orderly fashion, is normally used by a program to terminate a session that has proceeded as expected. All transmissions are completed before the link is dissolved. The programmer must decide which of the two programs disconnects under normal circumstances. The various DECnet user manuals refer to orderly link disconnections as disconnects or synchronous disconnects.

The second method of disconnecting a link forces termination of the link, regardless of whether or not the remote DECnet software has acknowledged previously transmitted data. In most programmer manuals, this method of disconnecting is called aborting the link. When the source node's End-to-End Communication software receives notification to abort a link, it cancels all messages waiting to be transmitted over the link. A program may choose to abort in response to some unusual system event, such as an impending emergency shutdown. When disconnecting or aborting a link, a program can send up to 16 bytes of data to the other program.

Using DECnet, a program in one node can access a file in another node, despite differences in the operating and file systems of the two nodes. DECnet's remote file access capability enables user-written programs that incorporate appropriate DECnet I/O calls to open and close a remote file, create or delete a remote file, and read from or write records to a remote file.

A second application of the remote file access capability is that people who use terminals can run a DECnet utility or issue a system command to manipulate remote files. From their terminals, users can transfer (copy) files to or from remote nodes, delete remote files, submit and execute command files at a remote node, append files to local or remote files, list remote directories, and queue files to a remote printer. This chapter describes program- and user-manipulation of remote files.

Since programs and users of terminals can access and manipulate remote files, a system manager need maintain only one copy of a file. Ensuring that a file is up to date is simpler when there is only one copy of it, rather than several copies on several network nodes.

Like other DECnet functions, remote file access requires the cooperation of two network programs which exchange a series of DECnet messages. A program in one node issues a call requesting remote file access. This call is translated into one or more DECnet message by the local DECnet software, then sent to a DECnet program in the remote node. The program in the remote node receives the file access request messages and translates them into a form recognizable to its file system. As in task-to-task communication, the two programs must establish a logical link before they can begin communicating.

In the context of remote file access, the program that requests remote file access is called the source program, and a DECnet system

program that receives the request is called the target program. Depending on the application, the source or accessing program can be a version of the Network File Transfer (NFT) utility (see below for details), a system command such as the COPY command, or a user-written program that accesses remote files via calls to DECnet subroutines or to file access subroutines provided by the local file system.

The target program is always a version of a DECnet utility called the File Access Listener (FAL), whose role is to receive remote access requests from the network. FAL completes connections initiated by remote accessing programs and translates the incoming requests into calls to the file system at FAL's node. FAL then sends the resulting file data back to the accessing program, at whose node special routines reformat the data to make it conform to local file structures.

File data can flow in either direction between the source program and a target program. The file accessed can reside on a mass-storage device like a disk and can be directed to an I/O device such as a lineprinter or a terminal. The actual file operations that a source program can perform depend on the file systems local to both nodes. The programmer reference manuals for the particular systems involved in remote file access include details on file operations.

- **The Data Access Protocol (DAP) Interface**

The File Access Listener at the target node and the accessing program exchange Data Access Protocol (DAP) messages to perform remote file access operations. DAP resides in the Network Application layer of the DNA architecture and uses the logical-link services of the End-to-End Communication layer. DAP defines a set of messages that controls the execution of remote file access and outlines procedures to accomplish specific file operations. For

instance, to create a file, an accessing program and a remote FAL must exchange a subset of DAP messages in a prescribed sequence.

The Data Access Protocol supports the transfer of files between heterogeneous file systems and supports sequential, relative, and indexed file organizations. It retrieves a file from an input device like a disk, cardreader, or terminal. It stores a file on an output device like a magtape player, lineprinter, or terminal. It permits sequential, random, and indexed sequential (ISAM) access of records. DAP lists directories of remote files and supports the deleting and renaming of these remote files. It allows multiple data streams to be sent over a logical link, recovers from transient errors, and reports fatal errors to the user. DAP also submits and executes remote command files. It permits wildcard file specification for command file execution, sequential file retrieval, file deletion, and file renaming. And it permits an optional file checksum to ensure file integrity.

In a typical DAP dialogue, the first message exchange consists of configuration information regarding the operating and file systems of the source and target nodes and the buffer size. *File Attributes* messages then supply information about the file to be accessed. Following these messages is the *Access Request* message to open a particular file. If data is to be transferred to the accessing program, a data stream is set up at this point. For both sequential and random access file transfer, one control message sets up the data stream (DAP has a file transfer mode that eliminates the need for DAP control messages once file data flow begins). After the file transfer is completed, *Access Complete* messages terminate the data stream. Figure 6.1 shows a DAP message exchange for sequential file retrieval.

DAP has been designed to minimize protocol overhead. Defaults are specified for fields whenever possible. And relatively small file records can be blocked together and sent as one large message.

User Node Message Description	Messages	Remote Node Message Description
Configuration Information (e.g., Buffer Size, OS, File System, DECnet Phase No., and DAP Version Number)	*Configuration Message* →	
	← *Configuration Message*	Configuration Information Returned
File Characteristics (e.g., Type, Block Size and Record Size)	*Attributes Message* →	
Access Request	*Access Message* →	
	← *Attributes Message*	Actual File Characteristics Returned
	← *Acknowledge Message*	File Opened
Set up Data Stream	*Control (Initiate Data Stream) Message* →	
	← *Acknowledge Message*	Data Stream Established
Request Start of Data Transfer and declare mode of Transfer	*Control (Get) Message* →	
	← *Record 1* ⋮ *Record N*	Data Sent in Records
	← *Status Message*	End of File Detected
Request to Terminate file access	*Access Complete Message* →	
	← *Access Complete Response*	Request Completed Successfully

Figure 6.1
DAP Message Exchange (Sequential File Retrieval)

A user program does not handle DAP messages directly. All DECnet implementations include system software that sends and receives DAP messages on behalf of user programs. DAP-speaking DECnet modules use the messages listed in Table 6.1 to accomplish remote file access and transfer.

▪ Programming Remote File Access

To gain access to a remote file, a user program incorporates DECnet I/O calls that activate remote file access subroutines. The function of the subroutines is to build, send, and interpret DECnet file access messages on behalf of user processes. The File Access Listener acts on behalf of the accessing or source user program by receiving I/O requests at the target node and translating them into calls to the file system at that node. Other file access subroutines currently supported by DECnet implementations include:

Network File Transfer (NFT), an interactive utility operating at the User layer of the Digital Network Architecture. NFT interfaces to a DAP-speaking accessing process and provides DAP functions for people who wish to manipulate files on any node in the network from their terminal.

Record Management Services (RMS), the standard file system for many of Digital's operating systems. In DECnet-VAX, DAP functions are incorporated into RMS. Remote file access functions are therefore transparent to accessing user programs on DECnet-VAX systems. A program uses the same I/O calls to access a local file as it does to access a remote one. The only difference between the two access requests is that, to access a remote file, a program includes a node identifier in the specification of the RMS file to be accessed. RMS converts the access request into equivalent DAP messages. These are sent to a remote FAL to complete the access request. To a DECnet-VAX user, RMS remote and local file access are almost identical, the only difference being the specifation of a node name and access-control information for a remote file access.

Message	Function
Configuration	Exchanges system capability and configuration information between DAP-speaking processes. Sent immediately after a logical link is established, this message contains information about the operating system, the file system, protocol version, and buffering capability.
Attributes	Provides information on how data is structured in the file being accessed. The message contains information on file organization, data type, format, record attributes, record length, size, and device characteristics.
Access	Specifies the file name and type of access requested.
Control	Sends Control information to a file system and establishes data streams.
Continue-Transfer	Allows recovery from errors. Used for retry, skip, and abort after an error is reported.
Acknowledge	Acknowledges access commands and Control messages used to establish data streams.
Access Complete	Denotes termination of access.
Data	Transfers file data over the logical link.
Status	Returns status and information on error conditions.
Key Definition Attributes Extension	Specifies key definitions for indexed files.
Allocation Attributes Extension	Specifies the character of the allocation when creating or explicitly extending a file.
Summary Attributes Extension	Returns summary information about a file.
Date and Time Attributes Extension	Specifies time-related information about a file.
Protection Attributes Extension	Specifies file protection codes.
Name	Sends name information when renaming a file or obtaining file directory data.

Table 6.1
DAP Messages

Network File Access Routines (NFARs), a set of FORTRAN-callable subroutines that become part of the user process. NFARs cooperate with a remote FAL, via DAP, to access remote files for user applications. RSX-DECnet uses NFARs to provide remote file access functions.

VAX/VMS Command Language Interpreter, which enables users of terminals on VAX/VMS systems to enter VMS commands interactively, in order to access and manipulate remote files. VMS commands pertaining to remote file access and manipulation interface with RMS to provide network-wide access. No separate NFT utility is therefore required in DECnet-VAX systems.

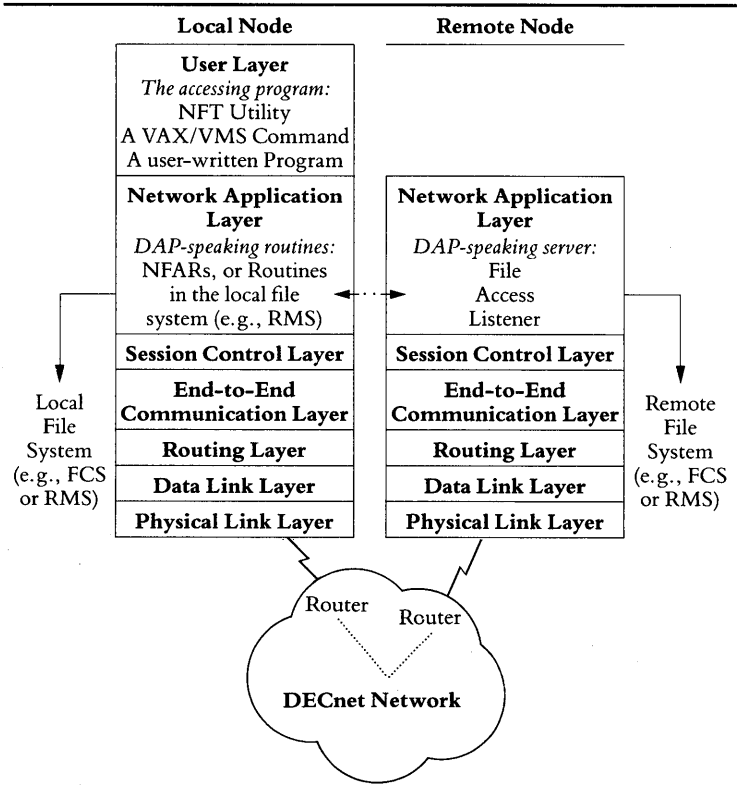
Network Management Modules use DAP services to obtain files for downline-loading other remote nodes and to transfer upline dumps for storage.

Figure 6.2 shows the location of the various file access subroutines in the Digital Network Architecture.

Regardless of the file access subroutine that is used, all DECnet systems allow people who use terminals and user programs to open an existing remote file, create a new file on a remote node, read records from a file, write records to a file, close a file, and delete a file. Besides this, specific DECnet systems allow user programs to manipulate remote files in other ways. The programmer's reference manual for each system discusses what operations are available and how to specify DECnet calls for remote file access. The rest of this chapter discusses aspects of remote file access that apply to all DECnet implementations.

File System Capabilities

A programmer needs to be familiar with the file system resident at the target node. File organization, access modes, and other



Legend:

NFT	Network File Transfer Utility	Router	DECnet node performing Switching
FCS	File Control System	⟋	A Physical Communication Link
RMS	Record Management Services	.....	A path through the network
NFARs	Network File Access Routines	◄···►	DAP message flow
DAP	Data Access Protocol		

Figure 6.2
File Transfer and DECnet File Access Subroutines

characteristics are dependent on the type of file system that each operating system supports. When the local and remote nodes have the same operating and file systems, programming remote access is similar to programming local I/O operations. However, when the access operation bridges different types of operating systems, the programmer faces certain variables and restrictions.

Generally, the theory of the lowest common denominator applies. A source program can perform those functions that its source language provides and that the remote file system supports. For example, a program on a VAX/VMS node cannot use all the remote access functions provided by DECnet-VAX if the target node runs RSX-11M.

By comparing the file system characteristics with the available remote access calls, a programmer can find out the kinds of file operations that are possible between two different DECnet implementations. This information is available in the programmer's reference manual for each DECnet system.

Initiating Remote Access

Like task-to-task communication, remote file access requires a handshake sequence at the beginning of the operation. Not only do NSP modules in the End-to-End Communication layer of the source and target nodes set up a logical link between the source and target programs, they also enable the exchange of DAP file access messages to prepare for the file operation that will be performed over the link.

This extended handshake, which is transparent to the source program, occurs automatically when the program issues a call to open a remote file. The form of the call to open a file varies from system to system and from language to language, but the call

always provides much of the information exchanged in the handshake. Using the information supplied by the call, as well as system-supplied data about the local file system, the remote file access subroutines (described above) generate DAP file access messages addressed to the remote FAL. In response, FAL, the target program, sends back DAP messages to define characteristics of its local file system.

For most remote access operations, the call to open that the source program issues contains a file specification, access-control information, and characteristics of the file to be accessed. Through the call, the source program passes this information to the subroutines in the local file system. Depending on the open call's particular function, the call passes additional information. For example, the call might specify if a file is to be opened for reading or opened for appending.

The File Specification

The file specification identifies the remote file to be accessed. Because the remote file system actually carries out the requested file operation, the programmer must know how the file is identified by users on the node where the file resides. Refer to the programmer's reference manual for the source and target systems for file specification syntax.

Access-Control Information

Access-control information identifies the accessing program to the remote system. It consists of a user identification code or name, a password associated with the user identification, and additional accounting information required by the remote system. If this information matches an account or guest account entry in the remote system's user file, the program gains access to that system's resources. As in specifications of remote files, the syntax of the access-control information must be recognizable to the remote system.

Gaining access to the remote system does not guarantee that requested file operations will succeed. In most Digital operating systems, each file has a protection code that determines the types of access allowed to defined groups of users. The user identification— a code or a name—specified by the accessing program determines the program's group category and, therefore, determines the types of access it can make to each file.

File Characteristics

The characteristics of the file to be accessed include access method, file organization, record attributes, data type, and record format.

Access Method (sequential or random): The source program indicates how it will access the file. All Digital file systems support sequential access, the method in which the records in a file are read or written one after the other. Selected file systems permit random access, which allows the source program to access a specific record without having to read all the records that come before it.

File Organization (sequential, relative, or indexed): Files that are organized sequentially have records arranged one after the other. All types of storage devices support sequential file organization. In files that are organized by the relative method, records within a file are identified by a relative record number. This number identifies the record's position relative to the beginning of the file. Indexed organization (available for RMS files only) is a complex file structure that allows both sequential and random access and uses record keys for identification (using a record key to retrieve a record from an indexed file is similar to consulting the index in a book to find the pages that discuss a particular subject). The keys used to identify individual records are defined at file creation. Only disk devices support relative and indexed file organization.

Record Attributes–This characteristic indicates the type of vertical format control that applies to the file.

Data Type (ASCII or image): Depending on the data's record attributes, DECnet software converts the format of ASCII data. Image data is a stream of bits to which the DECnet software applies no interpretation.

Record format (fixed-length, stream, variable-length, or variable-with-fixed length control—VFC): This characteristic indicates the manner in which records are formatted within the file. A VFC record supported by RMS files only includes a fixed-length control field in addition to the variable-length data portion.

- **Accessing Remote Files From a Terminal**

All DECnet implementations support some kind of terminal-based access to remote files. In some DECnet implementations, system commands, such as COPY, TYPE, and APPEND, enable remote file access from a terminal. Other DECnet implementations use the Network File Transfer (NFT) utility. The programmer's reference manual for a specific DECnet system tells which access method the system supports for terminal-based file access.

All DECnet implementations allow people using terminals to perform several operations on remote files. Users can transfer (copy) a file to or from a remote node or between two remote nodes. They can delete a remote file, and they can submit a local command file for execution at a remote node or execute a command file already existing at a remote node (command files cannot be submitted to or executed at a DECnet-RT node). In addition, users can append one or more local or remote file to an existing local or remote file (on DECnet-20 nodes, supported only for RJE-20 stations). Users of terminals can also obtain listings of remote directories (a directory file lists all the files that reside on a device and that belong to a specific user or category). Further, users can queue one or more file to a lineprinter. The files can be remote and the printer local, or the files can be local and the printer remote (on DECnet-20 nodes, this operation is supported only for RJE-20 stations).

Implementations of DECnet that support system commands rather than NFT also enable users to create, open, and close remote files as well as read to and write from them. Check system-specific documentation for details.

On behalf of the person using the terminal, the access method (NFT or system commands) creates a logical link between itself and the remote File Access Listener. From the input a user types at the terminal, the access method formulates the appropriate Data Access Protocol messages. End-to-End Communication software at the source node then sends these over the link. In turn, FAL at the target node interprets the messages it receives and interfaces with its local file system, in accordance with the terminal user's request. FAL then returns information and any requested file data to the source program (NFT or system commands) by sending back DAP messages. See Chapter 9 for examples of this procedure in an applications environment.

Access Control

To access a remote file, a person using a terminal must supply the same access-control information that an accessing program must include. This information—user identification, password, and optional account data—must be available whenever a file access command refers to a remote node. Refer to system-specific documentation to see how a system handles access information in terminal-based file access.

File Protection

The access-control information that a user at a terminal supplies at the local node determines the user's access rights at the remote node. The remote file system compares the user's identification with the protection code associated with the file to be accessed. The file system carries out a requested access only if the protection code for the file grants access to the identified user.

Remote File Specifications

A specification that describes a file on a remote node must conform to the remote node's syntax rules. If the remote node has a different operating system than the source node, the remote file specification may contain fields or conventions that the source node does not recognize. To prevent the source node from being confused by a foreign syntax, some implementations require the user to enclose the foreign specification in double quotes; some do not. Refer to the programmer's reference manual for a system for specific guidelines.

Remote Command File Submission

Most terminal-based file access implementations allow a terminal user to execute a command file in a remote system. A command file contains ASCII command lines equivalent to the command lines a user enters at a terminal. The remote node reads and executes these commands when a user submits the file. See system-specific documentation for details on how a system implements this capability.

Remote command file submission and execution are valuable means of using the resources of a remote system. At the local node, a user can run a text editor to create an ASCII file consisting of commands that conform to the syntax of the remote node's command language. Subsequently, the user can invoke the NFT utility (or system commands) to copy the file to the remote node for execution there.

Digital offers several terminal facilities that enable users at one node to communicate interactively with remote nodes and with terminal users at remote nodes. These facilities include:

- Interactive terminal-to-terminal communication
- Network virtual terminal, allowing direct access to a remote node's operating system
- Remote file access from a terminal

A fourth terminal facility, the Local Area Transport (LAT) protocol, enables terminals connected through a Terminal Server to an Ethernet LAN to establish links with any host on the same Ethernet. In addition, the terminals can connect to remote hosts not on the same Ethernet via an intervening DECnet host on the Ethernet. LAT is not a DECnet protocol.

This chapter discusses terminal-to-terminal communication, direct access to a remote node's operating system, and the LAT protocol. For information on how to access remote files from a terminal, see Chapter 6.

• Interactive Terminal-To-Terminal Communications

Users at two DECnet terminals can communicate interactively through the Phone or TLK utilities:

Phone–Phone is a utility of the VAX/VMS operating system. DECnet can operate on Phone to allow a terminal-to-terminal conversation over the network between two DECnet-VAX terminal users. Phone allows a terminal user to "dial" another terminal user on the same or on a different DECnet-VAX node. In a DIAL command, users specify the name of the target node and the name of the person with whom they wish to talk. If that person is logged on and available, the Phone utility activates the bell function on the remote terminal and displays messages on the terminal screen to notify the person of the awaiting phone call. The person being

called can then elect to answer the call by issuing an ANSWER command. Once the call has been answered, the two people can engage in typed conservation by entering messages at their terminals. At the end of the conversation, either user can issue a command to hang up, thereby disconnecting the connection at both ends.

Phone offers several options that can be implemented to provide terminal users with additional services and information. For example, there is a directory service that lists available users on each DECnet-VAX system supporting Phone.

Terminal users on two nodes can communicate via Phone only if both nodes support the utility. Nodes supporting different utilities cannot communicate at this interactive level.

TLK–TLK is a DECnet utility supported by DECnet-RSX. It enables terminal users on two DECnet-RSX systems to communicate. Through TLK, users can send one-way messages contained in one line (single-line mode) to another terminal user or can engage in an interactive conversation (dialogue mode). To send a message in either mode, users issue a TLK command specifying a node name and a terminal identifier for the terminal with which they wish to "talk." The user at that terminal receives a copy of the message sent, along with the identity of the sending terminal. All messages received are displayed on the terminal screen. In dialogue mode, TLK establishes an interactive connection with the target terminal so that a user there can respond instantly to the TLK messages received.

A single-line-mode message is just one line long, and it's terminated when the sender types a carriage return to transmit the message. To end a "conversation" of dialogue-mode messages, either user can type a [CTRL/Z] or EXIT command.

A system manager can use TLK to send messages read from a TLK command file. TLK command files are useful for sending many messages at once and for storing and sending sets of messages that need to be sent more than once (reminders to users of routine file backup procedures at the end of every day, for example).

All DECnet systems do not support TLK. Refer to system-specific user documentation (accompanying a system) to see if the system supports TLK and, if it does, for details on how to use TLK. Refer to Chapter 9 for an example of interactive terminal communications in an applications environment.

- **Network Virtual Terminal Facilities**

All DECnet systems, with the exception of DECnet-RT, have a utility that logically connects a local terminal to a remote node. With DECnet Phase III, such connections are possible only if the local node runs the same operating system as the remote node. A DECnet-VAX terminal user could use the SET HOST terminal facility to log on to another DECnet-VAX node; a DECnet/E terminal user could communicate with a remote DECnet/E node using the NET utility. Intermediate routing nodes do not have to run the same operating system as the nodes involved in the terminal-to-remote-node connection.

With DECnet Phase IV, logical-terminal-to-remote-node connections can be established even if the local and remote nodes are running different operating systems. The Phase IV utility enabling this is called network virtual terminal (NVT). NVT also enables terminals that are physically connected to a host on an Ethernet to connect logically to any other host node on the same Ethernet and to remote host systems (by using the CTERM protocol).

Programmers are the primary beneficiaries of the network command-terminal facilities. If a node does not have adequate system and storage resources to support program development, programmers at that node can log onto a remote node that does have the adequate resources. They can then develop programs interactively, as if they were local users of that remote node. For example, programmers on a DECnet-11S node would access a larger remote node, since RSX-11S nodes do not have any mass storage.

In Phase IV networks, NVT enables programmers to use the resources of any system in the network. Depending on the applications they are developing, programmers could log onto to the remote nodes best suited to their needs, to develop their programs with increased efficiency.

Terminal users at a node do not see the complex activities that DECnet performs to make it possible for them to log onto a remote node. All that a user needs to know to gain access to a remote node's operating system is its node name. A simple login command gives the user access to the resources of the remote node.

Refer to the user documentation accompanying a system for details on how to use available network command terminal utilities. Although different DECnet systems support different network virtual terminal utilities, all the utilities allow a terminal user issuing appropriate commands to log onto a specified remote node and perform most of the functions that the remote node allows its local users to perform (see Figure 7.1 for an example).

Network Virtual Terminal

The DNA layer responsible for providing NVT services is Network Application, the layer above the Session Control layer and immediately below the User layer. In the Network Application layer, the

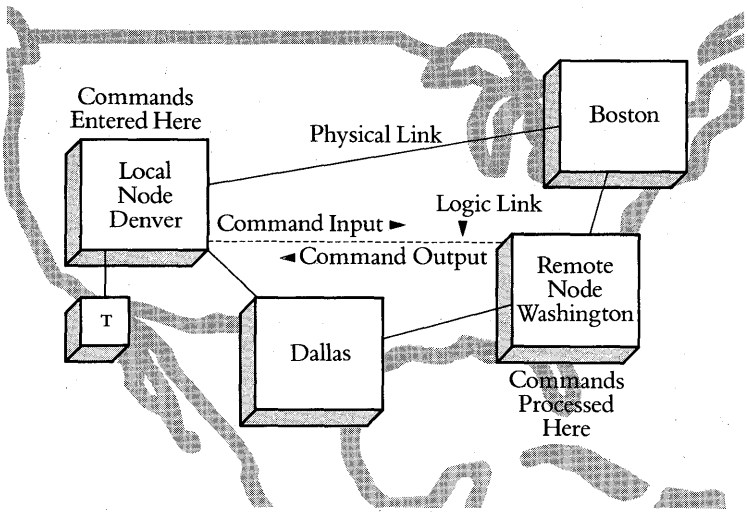


Figure 7.1
Remote Terminal Processing

protocols effecting virtual terminal communications are organized in two sublayers:

- *terminal communication module,* which controls the connections between applications and terminals.
- *command terminal module,* which enables terminals to communicate with Command Language Processors and application programs.

In an Ethernet local area network, NVT enables terminals to connect logically to any host node on the same Ethernet. In a wide area network, terminals on a node can use NVT to connect logically to any remote node in the same network. NVT protocols provide the following additional services:

- Distribution of terminal-handling functions between the two communicating systems.
- Management, at the operating-system level, of terminal input/output functions and terminal characteristics. The description of the command-terminal protocol (see below) includes a list of these functions.
- Offering standard terminal services, and featuring good performance and device independence. Optionally, offering methods of controlling the terminal's behavior in considerable detail.
- Support of high-availability implementations. The protocols contain functions that allow the restart of interrupted communications.

Figure 7.2 depicts how the NVT modules are organized within DNA.

Terminal Communication Protocol–This protocol is the Foundation layer, the lower of the two sublayers, of the NVT service, and is the base protocol for NVT. The terminal communication protocol is responsible for making and breaking connections between applications and terminals. It extends DNA Session Control layer services by establishing logical links between endpoints that are specific to terminal services. The endpoint in the host system is called a portal; the other end, in the server system, is called a logical terminal.

In the context of the NVT function, the host node is the node at which the terminal attempting to make the connection is located; the server system is the remote node with which the terminal establishes the connection.

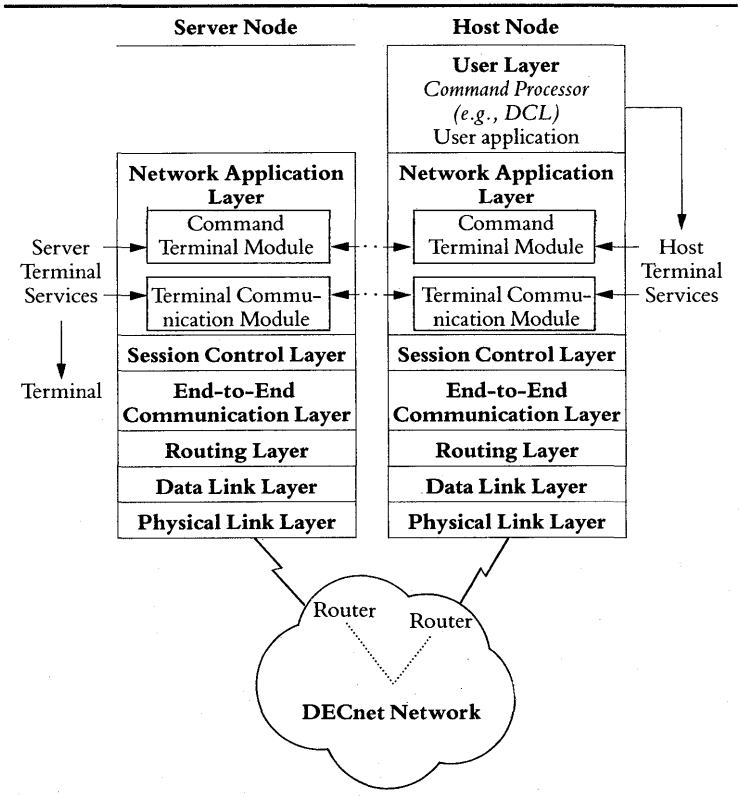


Figure 7.2
Network Virtual Terminal Service

A portal corresponds to a remote terminal identifier in the host, and a connection *binds* that identifier to an actual terminal. The NVT connection is therefore referred to as a *binding*.

In the future, it may become desirable to add new modules to the Network Application layer as alternatives to the command-terminal module. These modules might require their own protocols. In anticipation of such developments, the current implementation of the NVT modules has been designed so that portal, logical terminal, and binding share a state called a *mode*. In the future, *mode management* will enable the terminal communication protocol to select one of the several available higher-level NVT modules and protocols (currently, the only available higher-level module is the command-terminal module) Table 7.1 shows the terminal communication protocol messages and their functions.

Command Terminal Protocol–The functions of this protocol primarly involve command-line input and output. They are general enough to support a broad range of video and hardcopy terminal applications. The input/output functions include:

- Reading a line of input. Line terminators can be specified individually. Although the server system is responsible for echoing and operator editing, the host software has considerable control over these features, including the enabling and disabling of each feature.
- Accepting input even if the host program has not issued a read request and saving input until a read request arrives from the host program (this capability is known as *typeahead*).
- Inputting, or taking action in response to, certain characters immediately as the keys are struck (*out-of-band* character processing). These characters can be specified individually by users.
- Writing a string of output characters. Writing can proceed concurrently with reading, subject to certain synchronization options. The terminal operator can cause output to be discarded.

Message	Function
Bind Request	Requests a Binding; identifies version and type of sending system.
Rebind Request	Requests a rebinding (reestablishes broken communications, for high availability implementations).
Unbind	Requests that a binding be released.
Bind Accept	Accepts a Bind request.
Enter Mode	Requests entry of a new mode (the only mode currently defined is command mode). This selects the command terminal protocol as the higher level protocol.
Exit Mode	Requests that the current mode be exited.
Confirm Mode	Confirms the entry of a new mode.
No Mode	Indicates that the requested mode is not available or confirms an exit mode request.
Data	Carries Data (i.e. command terminal protocol information).

Table 7.1
Terminal Communication Protocol Messages

- Recognizing ANSI standard escape sequences on input and output.
- Cancelling a read request.
- Reading and setting terminal device characteristics.
- Determining how many characters there are in the typeahead and input buffers combined.
- Clearing the typeahead and input buffers.

Message	Function
Initiate	Carries initialization information, as well as protocol and implementation version numbers.
Start Read	Requests that a READ be issued to the terminal.
Read Data	Carries input data from terminal on completion of a read request.
Out-of-Band	Carries out-of-band input data.
Unread	Cancels a prior read request.
Clear Input	Requests that the input and typeahead buffers be cleared.
Write	Requests the output of data to the terminal.
Write Complete	Carries write completion status.
Discard State	Carries a change to the output discard state due to a terminal operator request (via an entered output-discard character).
Read Characteristics	Requests terminal characteristics.
Characteristics	Carries terminal characteristics.
Check Input	Requests input count (number of characters in the typeahead and input buffers combined).
Input Count	Carries input count as requested with Check Input.
Input State	Indicates a change from zero to non-zero or vice-versa in the number of characters in the input and typeahead buffers combined.

Table 7.2
Command Terminal Protocol Messages

Table 7.2 shows the command-terminal protocol messages and their functions.

Network Virtual Terminal Operation

In a typical network command-terminal session, a terminal management module in a server system with an active terminal requests a binding to some host system. When the terminal operator issues a SET HOST command, the terminal management module invokes a terminal communication services function.

The terminal communication services module initiates a logical link to its counterpart in the host system, using DNA Session Control layer services. On discovering the incoming logical link, the host module allocates a portal, thereby beginning the formation of a binding. The host module then accepts the logical link. Once the link has been formed, the server module sends a Bind Request message to the host. A terminal management module in the host, acting on behalf of the host's login process, recognizes the binding request and causes the terminal communication services module to send a Bind Accept message.

Once the binding has been formed, the host takes the initiative and prepares the binding for the command terminal protocol (in other words, enters command mode) with a further exchange of messages. This process takes place in connection with the first terminal I/O request from the login process. The respective protocol modules in the host and server can now begin speaking the command terminal protocol by sending an Initiate message to one another. After initialization, the dialogue of remote terminal service requests and responses ensues.

Terminal service requests normally originate with an application program in the host system. The application program issues requests to the host operating-system terminal services. Host terminal services issue corresponding requests to the host protocol module. The server protocol module reproduces those requests remotely and reissues them to the server-terminal services.

Termination of the session can occur in a number of ways. A typical, orderly shutdown begins with a logout by the application in the host. As a result, the host terminal communication services module sends an Unbind request. The server responds by releasing its resources. Finally, the host disconnects the logical link. Figure 7.3 shows an NVT message exchange.

Host Node Message Description	Messages	Server Node Message Description
	Bind Request Message ◄——	Request a binding with a specified Host
Accept Binding	*Bind Accept Message* ——►	
Establish Command Mode	*Enter Mode Message* ——►	
	Confirm Mode Message ◄——	Server enters command mode
Initialize command terminal protocol	*Initiate Message* ——►	
	◄—— *Initiate Message*	Initialize command terminal protocol
Begin terminal service dialog		
Issue Read request	*Start Read Message* ——►	
	Read Data Message ◄——	Accept and process a line of input
Host terminal software notified of out-of-band character	*Out-of-band Message* ◄——	User types an out-of-band character
	⋮	
Dialog continues		Dialog continues
Application logs off. Release host resources	*Unbind Message* ——►	Release server resources

Figure 7.3
NVT Protocol Message Exchange

• Local Area Transport (LAT) Protocol

LAT is an Ethernet-based virtual circuit protocol that does not use DECnet as a message transport facility. A Terminal Server on an Ethernet LAN implements the LAT protocol to enable terminals that are connected to it to establish a logical link with any host node on the same Ethernet. The host node must implement the host side of the LAT protocol. Figure 7.4 shows the LAT protocol implementation in an Ethernet local area network.

The terminal server and a host node on the Ethernet exchange LAT messages at regular intervals dictated by the circuit timer, which is a parameter that a network manager can set. Characters that a user inputs at a terminal are buffered until the circuit timer expires.

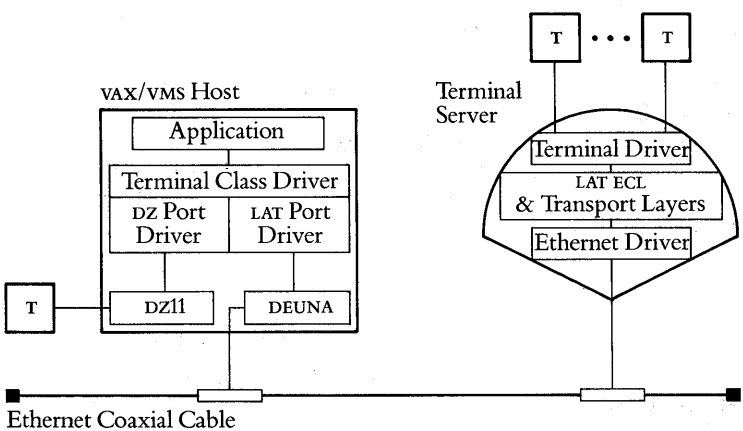


Figure 7.4
LAT Protocol Implementation

When this happens, a message containing the buffered characters is transmitted to the host.

The actions of the Terminal Server are transparent to users. With simple commands, users can establish establish logical connections (sessions) with host nodes and proceed with their work. Terminal Server software enables a user to maintain several sessions at one time. With a user-defined switch character on the terminal keyboard, the user can move quickly from one session to another without repeating the login dialogue each time. One of the primary advantages of the Terminal Server is that it frees host nodes from terminal processing, thereby enabling them to process user applications more efficiently.

Terminal Server software enables the Terminal Server to multiplex access requests from many terminals into one message for each host node on the Ethernet, thereby virtually eliminating blocking (a problem with terminals that are physically connected to host nodes, wherein the number of terminal access requests exceeds the number of host ports).

Relationship Between LAT and CTERM

To establish a logical link with a remote DECnet node that is not on the Ethernet LAN a terminal connected to a terminal server must go through an intervening DECnet host node on the Ethernet. This host node converts the LAT protocol that operates over the Ethernet link to the NVT CTERM protocol that operates over the DECnet link between the host and remote nodes. See Figure 7.5 for an illustration of the relationship between the LAT and CTERM protocols. The dotted line in the figure indicates the communication path established between a terminal on the terminal server and the remote DECnet node.

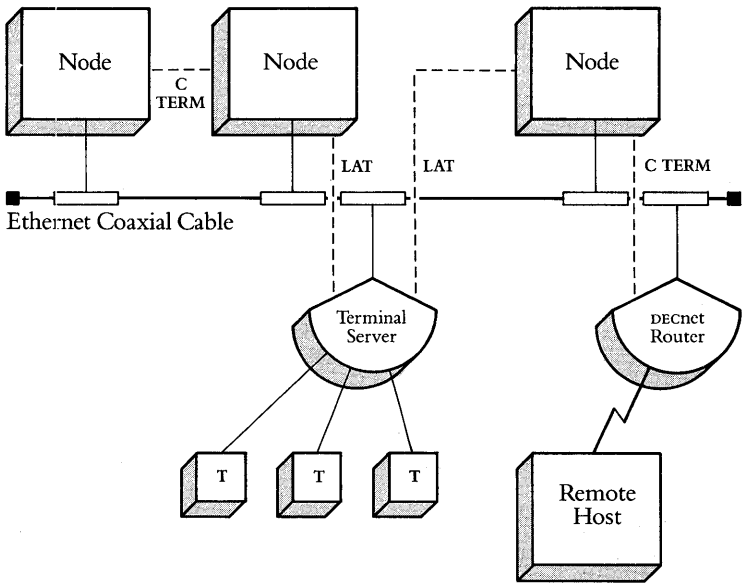


Figure 7.5
LAT and CTERM Operation

Systems in a DECnet network can communicate with systems in non-DECnet networks. Specifically, DECnet nodes can exchange data and share resources with DECnet and non-DECnet nodes over an X.25 packet-switched data network (PSDN) and with systems in an IBM SNA network.

A DECnet node can communicate over a PSDN with other DECnet and non-DECnet systems if it has been installed either with Packetnet System Interface (PSI) software or with the X.25/X.29 extension package that enables the DECnet node to be connected logically to a DECnet Router/X.25 Gateway. (See below for details on these additional software components.)

• X.25 Communications

X.25 is a recommendation of the Comite Consultatif International Telephonique et Telegraphique (CCITT), which defines a standard means for computers to interface with packet-switched data networks. A PSDN is a data communications service offered by common carriers, such as the Postal Telephone and Telegraph Authorities (PTT). Packet-switched data networks are widespread, with their number increasing every year. An organization can realize substantial cost savings by implementing network configurations in which widely dispersed systems communicate over a PSDN (the tariffs for data communications over a PSDN are based on volume of data sent rather than on connect time or distance between communicating systems).

It is mandatory for all public packet-switched data networks to use the X.25 interface. Private carriers, while not required to do so, can choose to run a network service using X.25 standards.

A packet-switched data network is one that receives addressed packets of data from network users and conveys them through internal switching routines to specified receivers. The way the

network delivers the packets is completely transparent to the end user who has no influence over the path the packet takes through the network. The X.25 recommendation specifies the manner in which the data packets get from the user to the PSDN.

The X.25 recommendation defines standards that govern the relationship between users and the PSDN network on the following levels:

Level 1, the physical level–This level defines the mechanical, electrical, functional, and procedural characteristics of the physical link between the user's equipment and the PSDN equipment. In DNA, Level 1 resides in the Physical layer.

Level 2, the frame level–This level defines the link-access procedure for data exchange over the communications link between the user and the PSDN. Level 2 resides in the Data Link layer of DNA.

Level 3, the packet level–This level defines the packet format and procedures for the exchange of packets containing control information and user data between the user and the PSDN. Like Level 2, this level resides in DNA's Data Link layer.

In accordance with the X.25 recommendation, a header containing control and destination information is included as part of each data packet submitted to the PSDN network. (In DECnet implementations, X.25 software, described below, creates this header.) The header information identifies to the PSDN the data sender and the data receiver. Thus, regardless of the kind of system or language used to generate the data message, the PSDN recognizes the header information and routes the packet to its proper destination.

The PSDN interleaves packets from many users over its transmission lines, thereby providing a fast and economical communications

service. The network differentiates between multiple packets on a line by means of a numbering scheme that ensures that the packets get sent to the proper destination in the correct order.

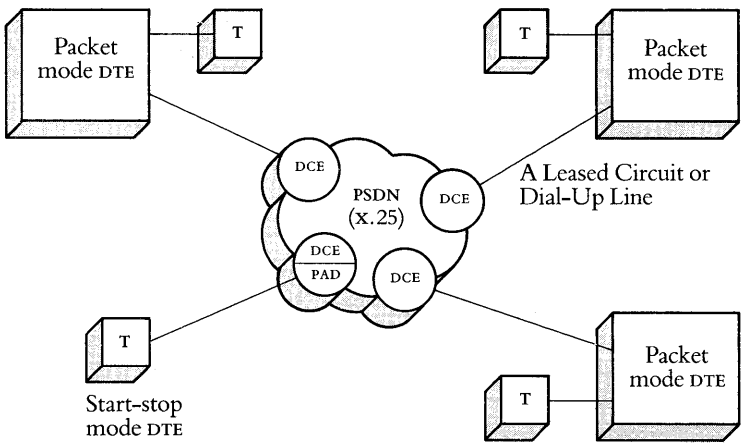
DTEs and DCEs

Each PSDN consists of a number of geographically separated switching nodes that are connected by high-speed links. When an organization leases a circuit from the PTT or other carrier, the circuit physically connects a computer in the organization to one of the PSDN switching nodes. The PSDN switching nodes are called network interfaces or data circuit terminating equipment (DCE). User computers or terminals connected to DCEs are called data terminal equipment (DTE). Computers and block-mode terminals are *packet-mode DTEs*. Asynchronous terminals are *start-stop mode DTEs*. Figure 8.1 illustrates these components.

X.25 Circuits

Two DTEs (for example, system A and system B, with which A wants to communicate over the PSDN) communicate by means of one or more virtual circuit. A virtual circuit is a permanent or temporary logical association set up by the PSDN. A *logical channel* is an association between a DTE and its DCE for a given virtual circuit. Each virtual circuit handles the exchange of data between two specific DTEs. The DTE at each end of a circuit assigns a logical channel number. This number is allocated independently at each end of the circuit (at each DTE-DCE interface). When sending data, a DTE includes a logical channel number to identify the channel and the corresponding circuit to which the data belongs.

When a DTE uses more than one virtual circuit at a time (for example, multiple circuits to the same destination or circuits to several different destinations), the circuits are multiplexed over the physical link between the DTE and the DCE.



PSDN = Packet Switched Data Network
DCE = Data circuit-terminating equipment, a PSDN switching node
DTE = Data terminal equipment, a user computer (packet mode) or a terminal using the PSDN (start-stop mode)

Figure 8.1
Components of a Packet-Switched Data Network

Virtual circuits are either permanent or temporary (switched). A *permanent virtual circuit (PVC)* is analogous to a leased line between a local and a remote DTE. Either DTE can send data over the PVC at any time without issuing calls to set up or break the circuit. When a user subscribes to a PSDN, the administrators of the PSDN allocate the logical channel number for the PVC. A temporary association between two DTEs is called a *switched virtual circuit (SVC)*. A DTE sets

up this type of circuit only when it wants to send data. The sending DTE assigns a channel, identifies the target DTE, and then obtains that DTE's agreement to communicate. When the DTEs have finished exchanging data, one or the other initiates a clearing sequence to terminate the SVC.

If a system uses an X.25 native-mode user interface, discussed below, under X.25 *Access Methods*, application programs running on the system must issue calls to specify or create a virtual circuit with a remote DTE. If the system uses data link mapping (DLM) mode, also described under X.25 *Access Methods*, users need merely include the node and task name of the destination DTE; the DECnet routing mechanism handles the selection of circuits as it would over any DECnet-to-DECnet connection. (With DLM, the user is not aware that the connection with the destination node is occurring over a PSDN.)

X.25 Gateway Access Protocol

This protocol, which resides in DNA's Network Application layer, permits user-written programs in a DECnet node to communicate with user programs in a non-DECnet system across an X.25-based PSDN. The communicating DECnet user program does not have to reside in the same DECnet node that is physically connected to the PSDN.

Table 8.1 lists the messages that the X.25 Gateway Access protocol uses to accomplish remote access to the PSDN.

Gateway Access Operation–Three processes are involved in the X.25 Gateway Access operation: the user process, the X.25 Gateway Server process, and the foreign X.25 DTE process. The user process accesses X.25 packet-level functions by issuing calls to the X.25 Gateway Access module (see below, under X.25 *Gateway Access Modules*). These calls are translated into X.25 Gateway Access

Message	Function
Open	Requests the use of a permanent virtual circuit.
Open Accept	Accepts a permanent virtual circuit Open request and allocates the PVC to the Gateway Access user.
Open Reject	Rejects a Permanent Virtual Circuit Open request, refusing allocation of the PVC to the Gateway Access user.
Outgoing Call	Requests that an outgoing Virtual Call be placed by issuing a Call Request Packet to the X.25 network.
Call Reject	Indicates rejection of an outgoing or incoming virtual call for lack of resources.
Incoming Accept	Indicates acceptance of an outgoing virtual call.
Incoming Call	Indicates an incoming virtual call has been received.
Outgoing Accept	Accepts a previously received incoming virtual call.
Clear Request	Requests that a virtual call be Cleared.
Clear Indication	Indicates a clear request packet has been received from the X.25 network.
Clear Confirm	Indicates a clear confirmation packet has been received from the X.25 network.
Reset Request	Requests that a virtual circuit be reset.
Reset Indication	Indicates that a reset indication packet has been received from the X.25 network.
Reset Confirmation	Indicates a reset confirmation by either the user or the X.25 network.
Reset Marker	Marks the place in the stream of Data messages where a reset occurred.
Data	Contains Data packets outbound from or inbound to a user.
Interrupt	Contains Interrupt packet data outbound from or inbound to a user.
Interrupt Confirmation	Confirms the receipt of an Interrupt message (used for flow control).
No Com	Indicates that a PVC entered a no-communication state.
No Com Seen	Indicates that the user of a PVC acknowledges a No Com message, and thus has been notified that there has been one or more failures of the X.25 network (e.g., Restarts).

Table 8.1
X.25 Gateway Access Protocol

Protocol messages and transmitted over a DECnet logical link to the X.25 Gateway Server process. The X.25 Gateway Server (any node that functions as the DTE to a PSDN) transmits X.25 packets to and receives X.25 packets from the PSDN. These packets flow over an X.25 virtual circuit to the foreign X.25 DTE process.

In a typical Gateway Access Protocol dialogue, the first message is usually an Open Message requesting that a PVC be opened, or an Outgoing Call message indicating that the user process wishes to place a virtual call. In the case of a virtual call, if the foreign DTE accepts the call and issues a Call Accept packet, the user process receives an Incoming Accept message. The user process can then exchange data with the foreign DTE process by sending and receiving Data and Interrupt messages. When the user process wishes to terminate the virtual call, a Clear Request message is sent to the X.25 Gateway Server, which clears the call by issuing a Clear Request packet to the X.25 network. The user process is then informed of the termination of the call through a Clear Confirm message.

Figure 8.2 shows X.25 Gateway Access Protocol exchange for switched virtual circuit operation, where the virtual call is initiated by the user process on the DECnet system.

X.25 Gateway Access Modules

DECnet utilities that provide the X.25 Gateway Access service include:

X.25 Gateway Access Module–This module receives user X.25 requests and communicates with the DECnet DTE that is directly connected to the X.25 PSDN.

X.25 Gateway Server Module–This module resides in the DECnet DTE (also called the gateway system), communicates with the X.25 Gateway Access module over a DECnet logical link, and acts on

User Node Message Description	Messages	Gateway Node Message Description
Request to place a Virtual Call	*Outgoing Call Message* ——→	Gateway Server process issues Call Request packet to X.25 network
	←—— *Incoming Accept Message*	Call Accept Packet received from X.25 network
Request to send a Data Packet	*Data Message* ——→	Data Packet sent on virtual circuit
⋮	⋮	⋮
	Data Message ——→	
User receives Data sent by foreign DTE process	←—— *Data Message*	Data packet received from virtual circuit
User receives Interrupt Data	←—— *Interrupt Message*	Interrupt Packet received from virtual circuit
	Interrupt Confirmation Message ——→	
Request to terminate (clear) Virtual Call	*Clear Request Message* ——→	Clear Packet issued to X.25 network
	←—— *Clear Confirm Message*	Clear Confirmation Packet received from X.25 network
Virtual Call is terminated		

Figure 8.2
X.25 Gateway Access Message Exchange

requests made by the X.25 Gateway Access module and the PSDN. This module uses the X.25 packet-level module to gain access to the PSDN.

X.25 Packet-Level Module–This module resides in the Data Link layer of the gateway system and acts as the DTE on the X.25 PSDN. It uses the X.25 frame-level module for its connection to the X.25 DCE. The DTEs at either end of a virtual circuit access the circuit by exchanging X.25 Level 3 packets with their local DCEs.

Table 8.2 summarizes the X.25 Level 3 packet messages and their functions.

In response to a user request, the X.25 packet-level module initiates the establishment of a virtual circuit to the specified DTE. The module assigns a logical channel number and transmits a Call Request packet, specifying the logical channel number and the address of the remote DTE.

Once the remote DTE has accepted the call and a virtual circuit has been established, data packets can be transferred. Each data packet is numbered sequentially. A Send Sequence Number carried in each data packet specifies the position of the packet in the sequential stream. A Receive Sequence Number from the receiving DTE acknowledges received packets and authorizes the transmission of additional packets. This number is carried in Data packets, Receive Ready packets, and Receive Not Ready packets.

The *flow-control* mechanism (see Chapter 4) operates separately for each direction of data transfer on a logical channel. The mechanism is based on the concept of a *window*, which is a range of packets authorized to be transmitted across the DTE/DCE interface.

Packet Type	Direction	Description
Call Request	DTE to DCE	Assigns a logical channel to the DCE and establishes a virtual circuit to the DTE addressed in the packet.
Incoming Call	DCE to DTE	Indicates that a remote DTE wishes to establish a virtual circuit with the receiving DTE and assigns a logical channel.
Call Accepted	DTE to DCE	Indicates that the DTE accepts the establishment of the virtual circuit.
Call Connected	DCE to DTE	Indicates that the remote DTE has accepted the establishment of the virtual circuit.
Clear Request	DTE to DCE	Indicates that the DTE wishes to deassign the logical channel and destroy the virtual circuit.
DCE Clear Confirmation	DCE to DTE	Informs the DTE that the logical channel is deassigned.
Clear Indication	DCE to DTE	Indicates that the remote DTE destroyed the virtual circuit.
DTE Clear Confirmation	DTE to DCE	Informs the DCE that the virtual circuit has been destroyed and deassigns the logical channel.
DTE Data	DTE to DCE	Carries user data from the DTE over the logical channel.
DCE Data	DCE to DTE	Carries data from the remote DTE over the logical channel.
DCE RR	DCE to DTE	Updates the DTE transmit flow control information.
DCE RNR	DCE to DTE	Indicates a temporary inability of the DCE to accept data packets. This condition is cleared by a DCE RR packet.

Table 8.2
X.25 Level 3 Packet Messages

(Continued)

DTE RR	DTE to DCE	Updates the DCE flow control information, authorizing the transmission of additional DCE Data packets.
DTE Interrupt	DTE to DCE	Carries user interrupt data over the logical channel.
DCE Interrupt Confirmation	DCE to DTE	Acknowledges receipt of a DTE Interrupt packet.
DCE Interrupt	DCE to DTE	Carries Interrupt data from the remote DTE.
DTE Interrupt Confirmation	DTE to DCE	Acknowledges receipt of a DCE Interrupt packet.
Reset Request	DTE to DCE	Requests that the logical channel and virtual circuit be set to an initial state.
DCE Reset Confirmation	DCE to DTE	Acknowledges that the logical channel has been reset.
Reset Indication	DCE to DTE	Indicates that the logical channel has been reset.
DTE Reset Confirmation	DTE to DCE	Acknowledges the resetting of the logical channel.
DTE Restart Request	DTE to DCE	Requests that all logical channels for SVCs be deassigned and that all PVCs be reset.
DCE Restart Cofirmation	DCE to DTE	Acknowledges the Restart Request.
Restart Indication	DCE to DTE	Indicates that all logical channels for SVCs should be deassigned and all logical channels for PVCs should be reset.
DTE Restart Confirmation	DTE to DCE	Acknowledges a Restart Indication.
Diagnostic	DCE to DTE	Indicates to the DTE an error on one of its logical channels.

The window size is selected independently for each direction of data transfer on the logical channel. The range of packets in the window changes as the packet-level module in the destination DTE receives and acknowledges packets.

Flow control can also use Receive Ready and Receive Not Ready packets. DNA uses this method of flow control when transmitting data to a DCE. This method is never used when receiving data from a DCE.

A DTE can construct a logical message from several consecutive packets, by using the *More Data* bit in the data packets. This bit is set in each packet, with the exception of the last packet in a sequence of packets composing a logical message. Data packets also contain another control bit, the Q bit, which allows a transmitting DTE to define two categories of data—normal data and interrupt data.

Each direction of transmission on a virtual circuit can have an outstanding *Interrupt* which allows a DTE to transmit one byte of information not subject to the rules of flow control that apply to normal data packets. Interrupt data is transmitted via an Interrupt packet and acknowledged by an Interrupt Confirmation packet.

The X.25 packet-level module of a DTE can use a *Reset* procedure to initialize a virtual circuit. When a DTE wishes to reset a logical channel, it transmits a Reset Request packet. A reset of one logical channel for a virtual circuit will cause a reset of the corresponding remote logical channel. A DCE confirms a Reset Request packet by transmitting a DCE Reset Confirmation packet.

The PSDN can also initiate a reset. At the time of a reset, the PSDN discards all data and interrupt packets in transit. When the

network wishes to indicate that the virtual circuit is being reset, it transmits a Reset Indication packet containing a reason for resetting. A DTE confirms a Reset Indication packet by transmitting a Reset Confirmation packet.

Either of the DTEs associated with a SVC can destroy the circuit by *Clearing*. The DTE's X.25 packet-level module transmits a Clear Request packet. The DCE returns a DCE Clear Confirmation packet, which deassigns the logical channel. The Clear Request is reported to the remote DTE in a Clear Indication packet. The remote DTE replies with a DTE Clear Confirmation packet. The logical channel at the remote node is then deassigned.

After a catastrophic failure, all SVCs associated with a DTE can be cleared, and all PVCs can be reset by means of the *Restart* procedure. A DTE issues a Restart Indication packet whenever the DTE Level 2 module connects to the DCE. A DCE can transmit a Restart Indication packet to force a DTE to clear all its virtual circuits.

X.25 Frame-Level Module–Like the X.25 Packet-Level Module, this module resides in the Data Link layer of the gateway system. It provides the X.25 frame-level protocol that enables the gateway system to communicate with the PSDN's DCE.

The X.25 frame-level specification defines two alternative link access procedures (LAP and LAPB) for frame-level communication between a DTE and a DCE. DNA supports the LAPB procedure, which defines the control of full-duplex communication over a synchronous line between a DTE and DCE. LAPB provides an error-free link by detecting and recovering from transmission errors. It also detects procedural errors and reports them to the user of the link. In addition, LAPB provides synchronization, to ensure that the frame-level entities in the DTE and DCE are in step.

Frame-level modules in the DTE and DCE communicate by sending and receiving frames to provide link control, data exchange, flow control, and error control. There are three types of frames:

Information Frames (I Frames) are used to transmit packet-level information and are also used to acknowledge previous correctly received information frames.

Supervisory Frames (S Frames) are used to perform channel control functions like acknowledging correctly received information frames, requesting retransmission of information frames, and indicating temporary inability to receive information frames.

Unnumbered Frames (U Frames) are used to provide additional link control functions like connecting and disconnecting the link.

Frames are divided into commands and responses. Table 8.3 offers a summary of the frame types.

Format	Commands	Responses
I	Information Transfer (I)	
S	Receive Ready (RR) Receive NOT Ready (RNR) Reject (REJ)	Receive Ready (RR) Receive NOT Ready (RNR) Reject (REJ)
U	Connect Link (SABM) Disconnect (DISC)	Unnumbered Acknowledgment (UA) Disconnected Mode (DM) Frame Reject (FRMR)

Table 8.3
X.25 Level 2 Frame Types

Figure 8.3 depicts the relationship of the X.25 Gateway Access modules to the PSDN and to the foreign DTE.

X.25 Access Methods

Access methods refer to how a DECnet system establishes communications over a PSDN. For X.25 communications, a DECnet node

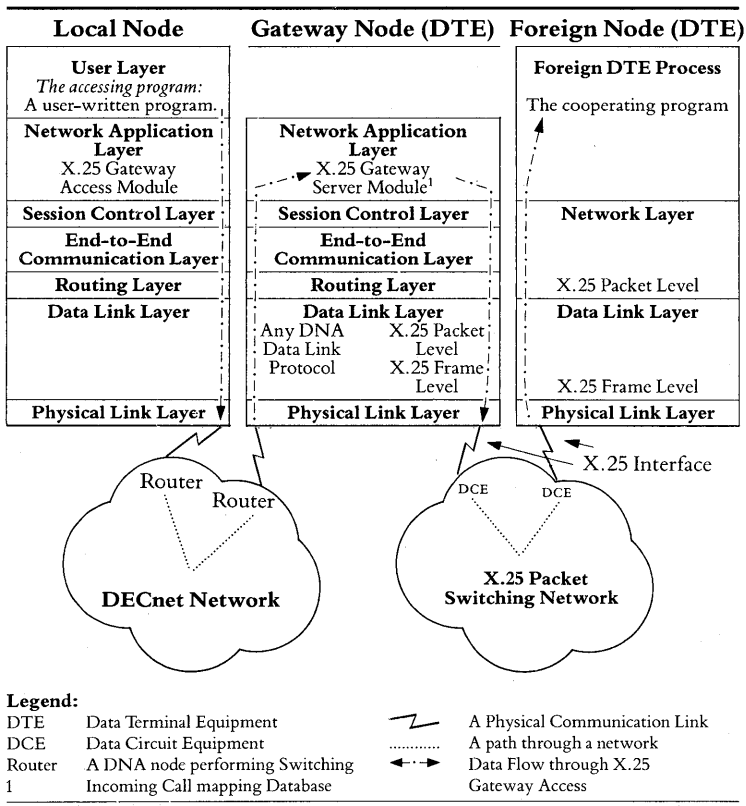


Figure 8.3
X.25 Gateway Access Operation

needs to run specific software and sometimes needs to be able to access dedicated hardware products. What a system needs for communication over a PSDN depends on the type of system (DECnet or non-DECnet) with which it wishes to communicate.

If a DECnet node communicates with DECnet systems only, it may be able to use a DECnet facility called data link mapping (DLM). (Check system-specific documentation to see if a system supports DLM. Some systems, DECnet-10 and DECnet-20, for example, do not.) DLM allows two DECnet nodes to communicate as though the PSDN were not there. The X.25 interface in this situation is completely transparent to users at both nodes. DECnet node A can use DLM if it has access to a DECnet node installed with PSI software. The PSI node must be positioned as a DTE in relation to the PSDN. If node A does have access to a PSI node, it can send data through a DECnet Router/X.25 Gateway node on an Ethernet. To use the DLM interface (either through a PSI node or DECnet Router/X.25 Gateway), node A can be located anywhere on the network and does not require any special software other than DECnet.

If users want the option of being able to communicate with non-DECnet as well as DECnet systems over the PSDN, their DECnet nodes need PSI software. RSX-11, VAX/VMS, and TOPS-20 operating systems support system-specific PSI software. This software allows a DECnet system to communicate in X.25 native mode if it is directly connected to a PSDN. X.25 native mode provides the complete X.25 programming interface which any DECnet or non-DECnet system connected to the PSDN can interpret. Regardless of vendor, all systems connected to a PSDN must support the X.25 protocol.

- **Note:**
  TOPS-20 software is divided into two categories: (1) PSI gateway software installed in the DN20 front-end processor, which provides gateway functions and (2) PSI access routines installed in the

DECSYSTEM-20 main processor. The access routines communicate with the gateway software to set up a connection with the PSDN. Because the DN20 is the DTE in these connections, it must be connected directly to the PSDN.

Native-mode PSI software enables users on the PSI DECnet node to write application programs to access resources on remote DTEs. This process involves issuing calls to create or specify virtual circuits, send and receive data, and terminate communications. Refer to PSI documentation for specific operating systems for a list of the supported calls, functions, and languages.

In DECnet-VAX and DECnet-RSX nodes, PSI software includes the DLM interface, thereby enabling user-transparent communication over the PSDN between these and other DECnet nodes. DECnet nodes containing PSI software do not require the services of intervening DTEs or DECnet Router/X.25 Gateway nodes.

Instead of PSI software, a DECnet-VAX node on an Ethernet can be installed with a VAX-11 X.25/X.29 Extension Package (XEP). The extension package provides subroutines that enable users to employ DECnet communication facilities to create and transmit data packets containing X.25 information. XEP cannot be used without the services of the DECnet Router/X.25 Gateway. The XEP data packets that users send are received by corresponding X.25 software in a DECnet Router/X.25 Gateway. The DECnet Router/X.25 Gateway is connected directly to the PSDN. It translates the XEP messages it receives into X.25 native-mode messages that can be understood by DECnet and non-DECnet systems connected to the PSDN. The combination of XEP software and the gateway facilities provides users with the complete X.25 native-mode programming interface. Multiple DECnet nodes can share a single X.25 connection to a PSDN through the Gateway or by using DLM.

CCITT Recommendations X.3, X.28, and X.29

Thus far, the discussion in this chapter has involved computers or terminals that send "packets" of data to the PSDN. These devices are referred to as packet-mode DTEs. There are, however, start-stop mode devices, such as asynchronous terminals, which can also communicate as DTEs over a PSDN. CCITT has defined three recommendations describing the PSDN interface for start-stop mode communication: X.3, X.28, X.29.

X.3–For terminals operating in start-stop mode in a PSDN, X.3 describes the basic functions and user-selectable capabilities of the packet assembly/disassembly facility (PAD).
X.28–Specifies how to control the user-selectable functions of the PAD from the terminal.
X.29–Specifies procedures for the exchange of control information and user data between a packet-mode DTE and a PAD facility.

A PAD is a facility provided by the PSDN. It acts as an interface between the start-stop mode DTE and the network DCE. The terminal user connects to the PAD and commands it to establish a virtual circuit to a destination DTE. The PAD receives the character-by-character data generated by the user terminal, reformats it into packets, and dispatches the packets through the network to the remote computer. At the remote computer, a pseudodevice called the X.29 Virtual Terminal Driver handles information sent by the terminal user in much the same way that a regular DECnet terminal device would. The X.29 Virtual Terminal Driver sets terminal characteristics, such as typeahead options and line terminators, that the remote terminal specifies. It delivers the information to the appropriate application on the DTE. The PAD and the X.29 Virtual Terminal Driver operate in a manner that is transparent to the user. The terminal appears to the terminal user to be connected directly to the remote computer.

Built-In PADs–Asynchronous terminals connected to a DECnet-VAX node with VAX-11 PSI or VAX-11 X.25/X.29 XEP can initiate a circuit with another DECnet or non-DECnet system over the PSDN *without buying the services of the PSDN PAD facility.* VAX-11 PSI and VAX-11 X.25/X.29 XEP each contain a program that emulates the functions of a PAD on the DECnet system. (The standard DECnet terminal interface is used in these communications.) In addition to the cost savings entailed in having a built-in "PAD," as opposed to leasing the option from a PSDN, VAX-11 PSI and VAX-11 X.25/X.29 XEP software enable a DECnet-VAX node to initiate and to respond to start-stop mode communications. Without the VAX-11 PAD capability, an asynchronous terminal connected to a DECnet-VAX node can only respond to messages initiated by terminals connected to a PAD on a PSDN.

Figure 8.4 shows the relationships between X.3, X.28, X.29, and the PSDN.

The X.3, X.28, and X.29 recommendations are layered on top of X.25. These recommendations define the standards required for start-stop mode systems to communicate over a PSDN *in addition to* the protocols defined by X.25. Digital PSI products include provisions for X.25, X.3, X.28, and X.29 recommendations. Information in this chapter describing X.25 communications applies to these other standards, as well.

- **DECnet/SNA Communications**

DECnet/SNA communications involve sending data from one or more DECnet-VAX or DECnet-RSX node through a Gateway to an IBM system in an SNA network. The protocol differences between the IBM network and the Digital network are resolved by interceding functions in the Gateway. The IBM SNA network environment sees the Gateway as a PU (Physical Unit) Type 2 cluster controller.

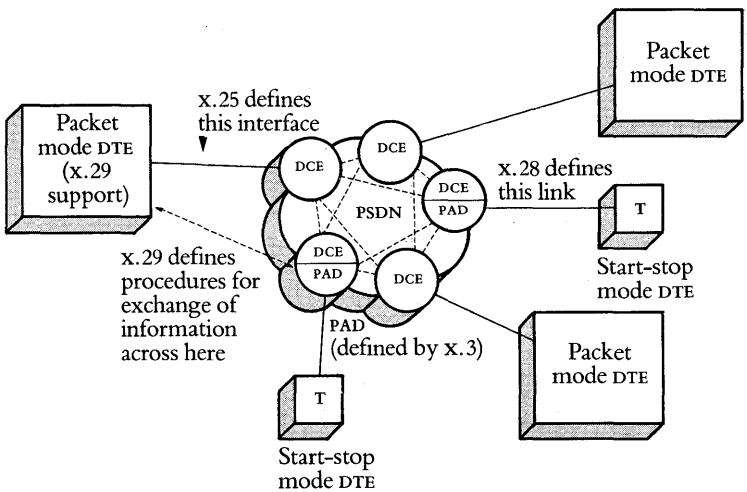


Figure 8.4
Relationships between X.3, X.28, X.29, and the PSDN

The term Gateway in the discussion that follows refers to all implementations of the DECnet/SNA Gateway node.

The functions performed at the Gateway are not visible to the end user. As far as the user is concerned, the Gateway is a black box that receives DECnet data at one end and sends out SNA data at the other. The DECnet node(s) and the Gateway involved in these communications can be located either in a wide area network or on an Ethernet. See Figure 8.5.

DECnet/SNA Gateway Access Functions

DECnet/SNA Gateway Access functions reside in the Network Application layer of a DECnet node. They are designed to make the

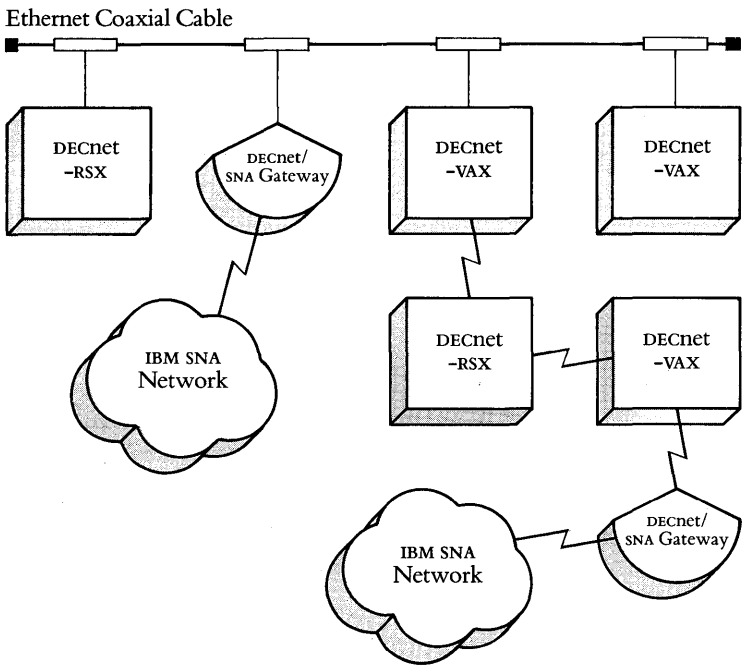


Figure 8.5
DECnet/SNA Environments

full capabilities of the SNA Data Flow Control, Transmission Control, and Path Control layers available to user programs residing anywhere in a DECnet network. The DECnet/SNA Gateway Access functions of a DECnet node communicate over a DECnet logical link with the DECnet system that is the PU2 on the SNA network. The DECnet system and the PU2 system exchange DECnet/SNA Gateway Access Protocol messages over the logical

link. Table 8.4 lists the SNA Gateway Access Protocol messages that the DECnet/SNA Gateway Access modules use to accomplish remote access to the IBM SNA network.

DECnet/SNA Gateway Access Operation–Three processes are involved in the DECnet/SNA Gateway Access operation: the user process, the DECnet/SNA Gateway Server process, and the IBM host application process (IBM hosts are called PU5 nodes in SNA terminology). The user process issues calls to the DECnet/SNA Gateway Access Routines. These calls are translated into DECnet/SNA

Message	Function
Connect	Requests that a Session be established with a Primary Logical Unit (PLU) in an SNA host.
Listen	Waits for a Session to be established by a PLU.
Bind Data	Indicates that a BIND has been received for a session solicited with Connect or Listen.
Bind Accept	Requests that the session being established with the BIND be accepted and placed in the running state.
Normal Data	Carries data on the SNA session's normal flow to and from the PLU.
Interrupt Data	Carries data on the SNA session's expedited flow to and from the PLU.
Disconnect	Requests orderly session termination.
Disconnect Complete	Indicates normal session termination has been completed successfully.
Abort	Requests abnormal termination of a session, or indicates an UNBIND has been received from the PLU.

Table 8.4
DECnet/SNA Gateway Access Messages

Gateway Access protocol messages and transmitted over a DECnet logical link to the DECnet/SNA Gateway Server process. The DECnet/SNA Gateway Server transmits and receives SNA Basic Information Units by accessing the functions of SNA protocol emulation modules (see below, under *DECnet/SNA Gateway Access Modules*), which connect directly to the SNA network as a PU2 and which contain the functions of the SNA Path Control and Data Link Control (SDLC) layers.

In a typical DECnet/SNA Gateway Access protocol dialogue, the first message exchange consists of Connect and Bind Data messages to establish a session with a Primary Logical Unit (PLU) in the IBM host. The session is completely established when the Bind Accept message is sent successfully. The user process can then exchange data with the PLU by sending and receiving Normal Data and Interrupt Data messages. When the user process wishes to terminate the session, it informs the PLU by using the appropriate Data Flow Control Request Unit. The PLU can then terminate the session by sending an Unbind, which results in session termination accompanied by an appropriate reason code to the user process. Figure 8.6 shows DECnet/SNA Gateway Access Protocol exchange for a session requested by the user process (the Secondary Logical Unit–SLU).

DECnet/SNA Gateway Access Modules

DECnet utilities that provide the DECnet/SNA Gateway Access Service include:

DECnet/SNA Gateway Access Module–This module receives user SNA session requests and communicates with the gateway system (the Physical Unit–PU–connected directly to the IBM SNA network).

User Node Message Description	Messages	Gateway Node Message Description
Request to establish an LU-LU session	*Connect Message* →	Gateway System issues an INIT-SELF message to the SNA network
	← *Bind Data Message*	BIND received from PLU
User accepts session as proposed by BIND	*Bind Accept Message* →	Gateway accepts session to PLU
User Receives Normal flow Data	← *Normal Data Message*	Normal Flow RU received from PLU
⋮	⋮	⋮
	← *Normal Data Message*	
User Sends normal flow RU to PLU	*Normal Data Message* →	
	← *Interrupt Data Message*	Expedited flow RU received from PLU
User sends expedited flow Data to PLU	*Interrupt Data Message* →	
User senses session termination	← *Abort Message*	PLU sends UNBIND to terminate session

Figure 8.6
DECnet/SNA Gateway Access Message Exchange

DECnet/SNA Gateway Server Module–This module resides in the DECnet system that is the PU on the SNA network (the gateway system) and communicates with the DECnet/SNA Gateway Access

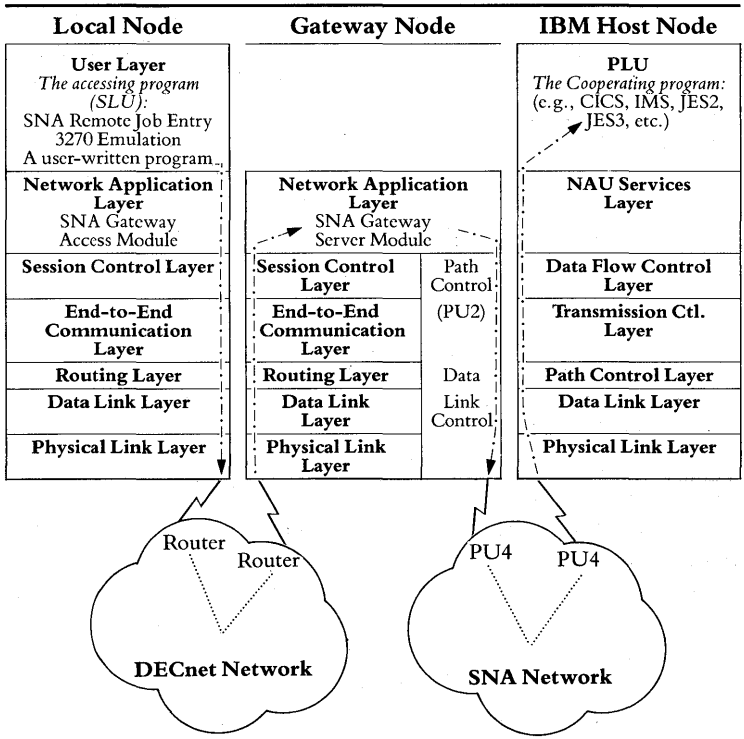
module over a DECnet logical link. It acts on requests made by the DECnet/SNA Gateway Access module and the PLU. The DECnet/SNA Gateway Server module uses the facilities of a SNA protocol emulator to gain access to the IBM SNA network.
DECnet/SNA Protocol Emulation–This set of modules resides in the gateway system and performs the functions of SNA Path Control and Data Link Control. These modules connect to the IBM SNA network as a Physical Unit Type 2 node. They also perform the functions necessary to communicate with the IBM SNA network's System Service Control Point (SSCP).

Figure 8.7 depicts the relationship of the DECnet/SNA Gateway Access modules to the IBM SNA network and to the IBM host. application (the PLU).

DECnet-VAX or DECnet-RSX nodes can communicate with systems in an IBM SNA network if they have been installed with appropriate DECnet/SNA software and have access to a Gateway. (For information on the operating requirements for the IBM system to communicate with the Gateway, such as supported subsystems and component parameters, refer to the *DECnet/SNA Gateway Installation Guide* which is available through a Digital sales representative when a DECnet/SNA Gateway is purchased.)

The DECnet/SNA Gateway implements IBM SNA protocols and works with function-specific gateway access software on a DECnet-VAX or DECnet-RSX node to enable users to interact with the IBM SNA network in the following ways:

Write application programs that communicate with programs in IBM SNA network. DECnet/SNA Application Interface software on the DECnet-VAX or DECnet-RSX node enables this internetwork function.



Legend:

SLU	Secondary Logical Unit	Router	A DNA node performing Switching
PLU	Primary Logical Unit	⚡	A Physical Communication Link
PU2	Physical Unit Type 2	.....	A path through a network
PU4	Physical Unit Type 4	◄···►	Data Flow through SNA Gateway Access

Figure 8.7
DECnet/SNA Gateway Access Operation

- Access IBM applications using a Digital terminal as though it were an IBM 3270 Information Display System (IDS) terminal. DECnet/SNA 3270 Terminal Emulator software on the Digital node makes this possible.
- Submit jobs to an IBM system and receive job output. DECnet/SNA Remote Job Entry (RJE) software enables this function.

Depending on the functions users wish to perform between the DECnet and IBM SNA environments, they can install a DECnet-VAX or DECnet-RSX node with any or all of the software packages described above. There are also gateway management tools that enable an operator to manage the DECnet/SNA Gateway from a DECnet-VAX or DECnet-RSX node.

DECnet/SNA Circuits

Communications between a DECnet node and the Gateway node occur over DDCMP or Ethernet circuits. Communications between the Gateway node and the IBM system occur over SDLC circuits. To perform certain DECnet/SNA functions, users may have to identify particular circuits. Circuit ID specifications are different for DDCMP and SDLC circuits. The DECnet/SNA Gateway management guide for a system contains information on specifying circuit IDs for each type of circuit.

Some system management activities related to SDLC lines and circuits require that the Gateway node be set up as the command executor. The DECnet/SNA Gateway management guide includes information on this process, as well.

Loading the Gateway Node Software

Figure 8.5 illustrates two implementations of the Gateway node: one resides on an Ethernet cable; the other does not. Functionally, both nodes are the same. However, if the Gateway is attached to an Ethernet cable, it must be downline-loaded from a DECnet-VAX or a DECnet-RSX host node on the same Ethernet. If the Gateway node is not located on the Ethernet, it can be directly loaded from floppy disks and bootstrapped either manually or by means of a TRIGGER command from a host node.

To illustrate the DECnet capabilities introduced in the preceding chapters, this chapter describes the user environment of a fictional company called Trident Manufacturing Corporation. Several interacting application subsystems support the company's daily operations. These subsystems include:

- Marketing subsystem, with order-entry, sales analysis, and market research applications.
- Manufacturing subsystem, with production scheduling, inventory, quality control, and purchasing applications.
- Accounting and Financial susbsystems, with billing, accounts-receivable, and short-term cash management applications.

Figure 9.1 illustrates the interaction among these subsystems and applications within the context of the following discussion. The figure does not show application interactions that are irrelevant to a discussion of DECnet capabilities (the short-term cash management application interacts with other applications at Trident Manufacturing, for example).

The order-entry application captures a basic set of order-related information, including customer name, address, products ordered, and delivery requirements. If the products ordered come out of finished-goods inventory, this basic information is then input to order-assembly and dispatch applications. If the ordered goods have to be manufactured, (as in the example to be presented), the basic order-entry information goes to a production scheduling application, which then interacts with raw materials inventory and/or component inventory. If required, the inventory applications interact with purchasing. The production scheduling system updates a corporate database that records historical and scheduled utilization of production facilities.

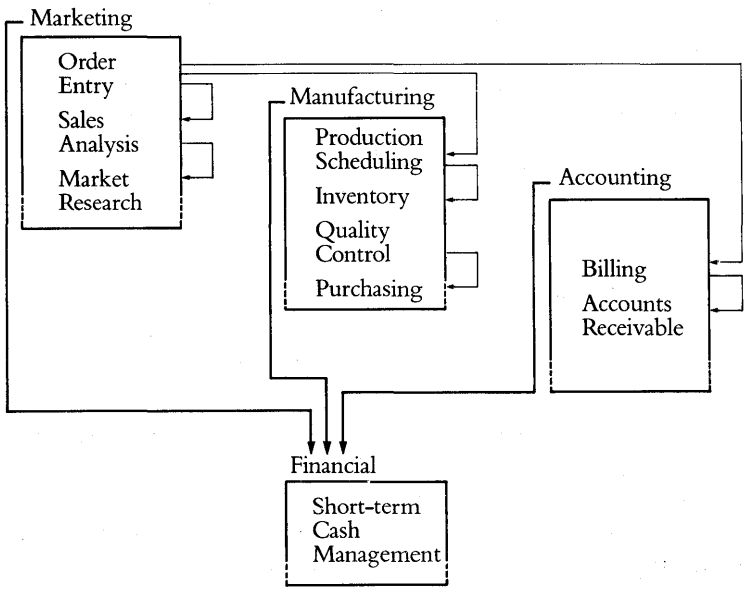


Figure 9.1
Subsystem and Application Interaction at Trident Manufacturing Corporation

The billing and delivery-scheduling applications (not shown in Figure 9.1) also use the basic order-entry information, as well as other information retrieved from various databases that are accessed and updated by the complex of application subsystems. Accounts receivable, which interacts with credit and billing applications, is updated on a regularly scheduled basis.

Since Trident Manufacturing is an international company, its application programs interact both within and over national boundaries. The nodes in Trident Manufacturing's DECnet network communicate over packet-switched data networks (PSDNs) in the United Kingdom, the United States, Canada, France, the Netherlands, and West Germany. These cost-effective communications facilities implement the X.25 recommendation of the CCITT. At Trident Manufacturing, DECnet Router/X.25 Gateway nodes and X.25 Packetnet System Interface (PSI) capabilities resident in general purpose DECnet nodes (see Chapter 8) meet requirements for communicating over PSDNs within and over national boundaries.

The organizational structure of Trident Manufacturing Corporation dictates a network configuration that includes wide area and local area networks. The processors, terminals, and personal computers that make up the network are at functionally defined locations such as sales offices, manufacturing facilities, warehouses, distribution centers, and corporate headquarters. DECnet nodes at these locations are linked by leased lines and X.25 networks. In addition, some applications require interaction between DECnet nodes and IBM/SNA systems.

DECnet capabilities discussed in this chapter include task-to-task communication, file transfer, remote file access, and terminal communications.

• DECnet's Task-To-Task Capability in an Application Environment

DECnet's task-to-task capability handles all the network processing that enables application systems at Trident Manufacturing to interact with one another over the network. (Chapter 5 describes

how the task-to-task communication capability enables interaction between user application programs, between user programs and DECnet modules, and between DECnet modules in different nodes. The example here illustrates only the interaction between user programs.)

The only requirement that network communication imposes on programmers at Trident Manufacturing is that they write applications to issue calls to network services and to cooperate with one another in regard to data formats and function-specific data transfer operations. Logical-link processing by DECnet is completely transparent to terminal operators. End-to-End Communication layer software in the communicating nodes automatically sets up the logical links (see Chapter 4) over which the applications communicate.

Since all DECnet implementations are designed in accordance with the specifications of the Digital Network Architecture, there is a standard application-to-network interface across all implementations. All of Trident Manufacturing's applications, regardless of the Digital operating systems under which they run, access the network by means of the same mechanism. This common network interface encourages an open-ended design of application systems. Thus, future applications that programmers at Trident Manufacturing develop will be compatible with the existing set at the networking level.

Task-to-Task Communication: Order Entry And Production Scheduling

The order entry application has three levels of communication:

- Between local and regional sales offices (order-entry application and sales analysis application)
- Between regional sales offices and corporate headquarters (sales analysis application and market research application)

- Between local sales offices and corporate headquarters (order-entry application and production-scheduling application)

Basic information captured at the local level includes customer name, billing and delivery addresses, products ordered, and delivery requirements.

The production-scheduling application is centralized at corporate headquarters. This application system matches order requirements against available production facilities (as recorded on the production-scheduling database) and against raw-materials-and-component inventories. It then posts an indication back to the order-entering sales office as to whether or not delivery requirements can be met. If they can, the production-scheduling system notifies the appropriate manufacturing locations and updates the production scheduling database.

- **Network and Application Interaction**

Logical-link processing between the applications encompasses three distinct operations: The establishment, control, and destruction of the link.

Establishment of the Link

At the sales offices, data entry operators at VT100 terminals invoke the order-entry transaction system and complete the order-entry forms displayed on the terminal screens. From the operators' point of view, the procedure is straightforward. They enter order information in the displayed forms; the completed forms are then ENTERED; and a new form appears. What happens at the applications level is transparent to the operators. At the applications level, the entered orders are checked locally against customer and product files, and the operators are notified via the screen of errors.

The activities at the network level are also transparent to the operator. At this level, the order-entry system opens communication with DECnet. Then, by issuing connect requests, it tries to establish logical links with a sales analysis system at the regional sales office and with the production-scheduling system at corporate headquarters.

If the other applications are ready to communicate with order entry, the programs are logically linked, and they begin exchanging data in accordance with the rules established by the cooperating applications. The production-scheduling and sales analysis systems could be receiving order-entry information simultaneously from several terminals at one sales office or at different sales offices. DECnet End-to-End Communication layer software sets up unique logical links for each pair of applications bound in a task-to-task session. Any communications from production scheduling back to order entry during a session, such as notification that the delivery requirements specified cannot be met, are transmitted over the logical link that had been established between the two programs at the beginning of the session.

Control of the Link–In the course of the sessions between the programs, DECnet monitors and controls the links, making sure that all transmitted data is received and that all data received at the data link level is turned over to the receiving program in proper sequence. Order data records are of variable length. End-to-End Communication layer software at the sending node segments the record and specifies a sequence number for each segment. End-to-End Communication layer software at the receiving node acknowledges receipt of each segment by sequence number and assembles message segments in the order specified by the sequence numbers.

To make certain that the order-entry and production-scheduling applications interact efficiently, End-to-End Communication software requires that the receiving program issue RECEIVE calls that set up buffers for the receipt of data. End-to-End Communication layer software at the sending node will not transmit data unless it has been notified that a buffer is available at the destination node for the receipt of message segments. DECnet employs this flow-control technique to make sure that buffer overflow conditions do not arise. In this manner, message retransmissions, which increase communications overhead, are almost entirely eliminated.

Destruction of the Link–When all order data has been entered, the application initiating the logical link issues DISCONNECT and CLOSE calls to the network. The logical link binding the programs is destroyed, and the network resources that were employed in maintaining that link are released.

- **DECnet's File Transfer Capability in an Application Environment**

At Trident Manufacturing Corporation, terminal operators and programs initiate file transfer operations over DECnet. Below, are examples of each.

User-Initiated File Transfer: Market Research And Sales Analysis

Trident Manufacturing's market research system runs at corporate headquarters. This system accepts input from the sales analysis applications running at the regional sales offices and from market research analysts located worldwide.

Input from sales analysis provides historical information that is used to plot product acceptance trends. Input from market research analysts quantifies market trends in specific geographical areas and projects the impact of new products on existing and forecasted customer bases.

Market research analysis is supported by a series of charts produced by the sales analysis application. The charts are written to files that are then made available to the research analysts. Terminal operators at the regional sales offices use electronic mail over DECnet to notify the analysts when a new set of charts is available.

The analysts edit the charts, supplying additional input and modifying existing data. Some use DECnet file transfer capabilities to transfer the chart files from the market research application at corporate headquarters to their local nodes. Others use the same DECnet facility to copy the chart files to their accounts on the node running the market research system at corporate headquarters. Others use the editing feature available at their local nodes to update the charts online.

Analysts working at DECnet-VAX nodes, for example, use the command **$ COPY CVAX::CHART.CUR *.*** to obtain the chart files.

With this command, a research analyst copys a file called CHART-CUR that resides on node CVAX (at corporate headquarters). CHART.CUR on node CVAX is written to a file of the same name on the analyst's local node. The analyst can specify a different file name.

As Figure 9.2 shows, the COPY request (solid line) and response (dotted line) can be routed through multiple nodes. To the user

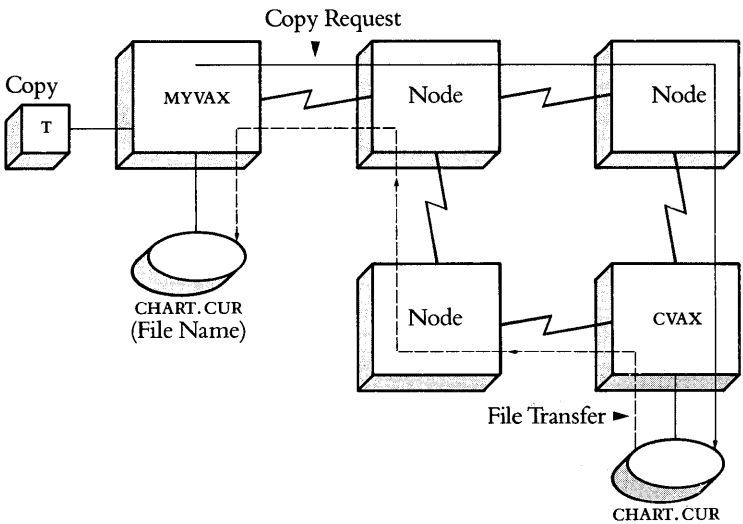


Figure 9.2
COPY Request/Response Is Shown Routed through Multiple Nodes

making the COPY request, this network activity is transparent, however. The preferred path for each transmission is selected by DECnet's routing capability.

The DECnet-VAX node at corporate headquarters runs ALL-IN-1 software (an integrated menu-driven set of office applications that includes word processing, calendar and desk management, graphics, mail, spreadsheet, and other user-specified capabilities). Analysts with accounts on this node frequently use the integrated spreadsheet, graphics, and word processing capabilities to prepare

reports and presentation materials. They can then distribute the reports over the network to specific individuals. The timeliness with which the reports can be prepared and distributed enables management to react to market shifts and adjust production, development, and advertising direction appropriately.

The transfer of the files to or from the research analysts' systems is made possible by the DECnet file transfer mechanisms described in Chapter 6. Operation of the mechanisms is completely transparent to the terminal users.

Program-Initiated File Transfer: Quality Control And Vendor Analysis

At Trident Manufacturing, programmed file transfer operates in the interaction between quality control and purchasing systems. The quality control applications run at the company's manufacturing locations, and the purchasing systems operate at regional purchasing centers.

In one of the quality control operations, incoming components used in product assembly are screened by sensors that analyze dimensional characteristics. The sensors are multidropped from DECnet-RSX nodes that control sensor operation and that translate sensor input to formatted data.

Results of quality control analyses are accumulated by component, shipment, vendor, and quality control location. On a daily basis, a program that runs in a DECnet-VAX node at each of the plants accesses this information and builds files by manufacturing location, date, vendor, shipment, component, and quality control location. The same program also develops information on rejected

items in terms of units and dollar value for each vendor, shipment, component, and quality control location.

During the third work shift, when systems and lines are less busy, a data acquisition program in the purchasing subsystems uses the DECnet file transfer capability to move these files to the regional purchasing centers, at which the data is input to a vendor analysis and evaluation program.

• DECnet's Remote File Access Capability in an Application Environment

The file transfer capability (user- and program-initiated) described above is a subset of DECnet's *remote file access capability.* This capability enables programs and users to transfer files from one node to another, to manipulate files on remote nodes, and to display remote files at local nodes. The DECnet mechanisms used to perform these operations are identical to those used in file transfer.

Network Virtual Terminals

Another DECnet feature that supports extensive remote file access operation at Trident Manufacturing Corporation is the network virtual terminal capability. This capability enables a user at a terminal at one node to log on to a remote node and operate on the remote node as if the terminal were directly connected to it. People using this capability to access remote system resources must, of course, have proper access to the remote nodes and files.

Although there are differences in the details of how the various DECnet implementations require users to exercise the network virtual terminal (NVT) capability, the overall NVT procedures are

similar. A user at a terminal on a DECnet-VAX node would use the SET HOST command to log on to a remote node to use resources located there:

$ SET HOST ATLANTA
Username: RAINER
Password: (not echoed)
Welcome to VAX/VMS Version 3.4 on node ATLANTA
$

In the above example, a user identified as RAINER and located at a terminal connected to a DECnet-VAX node entered a SET HOST command to log on to node ATLANTA. The system then asked for Rainer's name and password. The welcome message that printed out indicates that Rainer has a valid account on node ATLANTA. The prompt ($) that follows, shows that Rainer is logged onto node ATLANTA and can use its files and other resources as if he were at one of ATLANTA's local terminals. If Rainer did not have a valid account on that node, an "Unauthorized User" message would print out and Rainer would be returned to his local system.

Short-Term Cash Management

DECnet's remote file access capability is the foundation for a short-term cash management application currently under development at Trident Manufacturing Corporation. This application is expected to optimize the corporate treasurer's ability to identify and invest idle funds.

Corporate funds are kept in many accounts at foreign and domestic banks. A short-term cash management application system will enable the corporate treasurer to check the status of these funds, identify idle cash, and then transfer these nonworking assets to the

portfolio management group for investment, specify the funds for debt reduction, or retain them in the current account. DECnet's remote file access capability is a principal component in ensuring the success of this application.

The facets of the application development effort include developing appropriate security measures, identifying parameters and processing requirements that will help define "unencumbered funds," cross-referencing regional databases for relevant cash and local currency value data, and developing algorithms that equate investment opportunities with available cash. Through Gateways and through DECnet's remote file access capability over local area, wide area, X.25, and IBM SNA networks, the treasurer's office will be able to isolate and examine cash records maintained in regional databases worldwide. Analysis of this information will then enable the treasurer to make timely transfer and investment decisions.

DECnet's Terminal Facilities in an Application Environment

Interactive terminal communications (see Chapter 7) enable users to communicate with one another over the network in realtime. Network virtual terminal capabilities enable terminal users to access any remote node and to use its resources as if it were the local node. In addition, the ability to include personal computers in a DECnet configuration and to use them in both processing and terminal-emulation modes adds a new and powerful dimension to the terminal facilities available through DECnet.

The following paragraphs describe some of the ways in which Trident Manufacturing Corporation uses these capabilities in its daily operations.

Order Entry And Production Scheduling

Production scheduling frequently involves some online communication between a sales office and a manufacturing site. If, for example, the sales office is notified by the production scheduling system that customer delivery requirements cannot be met, the sales office might consult with the customer and negotiate another acceptable date for delivery of ordered products. A terminal operator or the salesperson will then invoke a DECnet interactive terminal facility (Phone or TLK, described in Chapter 7) at a sales office terminal and communicate revised delivery instructions to the manufacturing location.

When working at a terminal on a DECnet-VAX node, for example, the salesperson would invoke the Phone utility. In the following, a dialogue is established with a user at another DECnet-VAX node at a manufacturing location named MFG7.

- **$ PHONE MFG7::SCHED**

This command places a call to a user named SCHED at the manufacturing site. If SCHED is currently logged on to the DECnet-VAX node, a message will be displayed on the screen indicating that a communication has arrived. If the sender is a user named GREEN at a node called SALES4, the message will read as follows:

SALES4::GREEN is phoning you on MFG7

To receive and respond to the call, SCHED enters:

- **$ Phone Answer**

This reply causes both terminals to display a split screen. The top half of the screen displays the local user's input, and the bottom half displays the remote user's input. Each participant in the dialogue can enter text at the same time.

After a salesperson and a user at the manufacturing site have used this facility to arrive at a mutually acceptable schedule for production and customer delivery, the manufacturing-location user updates the production-scheduling database at corporate headquarters.

Trident Manufacturing finds that such informal interaction between users frequently resolves customer service issues more quickly than do predefined procedures. DECnet interactive terminal communications support efficient user interaction throughout the network.

Market Research

Because effective market research must be responsive to current market trends, market research analysts at Trident Manufacturing develop forecasts and test new product marketing strategies in real time. Field input is played against market models; results are analyzed by statistical processing capabilities; and "what if" input is developed to refine and extend the models.

Research analysts use DECnet's terminal facilities extensively in performing this work. Field analysts provide input via VT100 terminals that are connected to nodes at regional locations. Analysts at corporate headquarters use VT125 terminals connected to a DECnet-VAX node with node-resident ALL-IN-1 software capabilities to develop reports and presentation material.

In developing reports, the analysts use the integrated ALL-IN-1 software capabilities to combine spreadsheet computation, graphics development, and word processing. When required, they use ALL-IN-1's mail capability to distribute reports and analyses over the network.

In addition to the integrated mail capability that ALL-IN-1 provides, several similar capabilities are available to DECnet users who might

not need all of the capabilities of the integrated package. The VMS MAIL capability, for example, enables users at DECnet-VAX nodes to exchange correspondence. To send a message, the user simply enters the MAIL command, specifies the subject of the correspondence, and either specifies a file to be sent or types the message in response to a prompt. For example:

$ MAIL/SUBJECT = ZONE12 UPDATE.MKT LONDON::GRAHAM

This command line delivers the contents of the file, UPDATE.MKT, to a user named Graham at node London. If Graham is logged on to node London, a message such as the following appears on Graham's screen:

New mail from YORK::PEEK

This line indicates that the waiting message is from a user named Peek at node York. to read the mail, Graham enters the MAIL command and presses RET. A message is displayed:

You have 1 new mail message
MAIL

Graham presses RET in response to this prompt, and the mail sent by Peek is displayed on the screen.

Users who are not logged on when mail arrives for them are notified of the messages received when they next log in.

Short-Term Cash Management
The PRO/DECnet node in the treasurer's office is connected with an Ethernet local area network cable that serves the department and

other adjacent groups. This personal computer serves the treasurer in several ways:

- As a stand-alone processor with its own self-contained file and disc services.
- As a node on the DECnet network that is capable of interacting with computing capabilities and other resources located elsewhere on the network.
- As a terminal on the DECnet network that can access other nodes and use resources located at those nodes.

The network mail services available to the treasurer through this node or through another host node on the network will be of significant value when the short-term cash managment application is fully implemented.

Decisions to use available cash must take into account both internal and external factors in a very dynamic environment. Internal factors include payables, operating requirements, and debt constraints. External factors, such as bank policies in regard to minimum balances, for example, are important considerations in determining thresholds that define cash as excess. Such policies, however, are frequently conditioned by the general economic climate, government regulations in regard to fund transfer, and other uncertainties that govern international financial transactions.

Using the mail and dialogue facilities, the treasurer can keep informed, in realtime, of actual or potential changes in such conditions. Fund accumulations can be monitored along with those conditions that can affect movement of those funds. Records

of information that influence investment decisions can be retained online for quick lookup and can also be printed if the treasurer requires hardcopy. This kind of information will enable the treasurer to optimize decisions in regard to existing and potential inventories of investable funds.

- **DECnet as a Consideration in Relocation Planning**
  In planning relocation of the corporate headquarters, Trident Manufacturing Corporation management, along with a group of outside consultants, carefully examined the consequences that the projected move could have on future operations. The company planned to move to an location outside the commuting distance of a major metropolitan area. Real estate costs and tax advantages were attractive; a dependable labor pool was available, as were facilities for employee cultural and recreational activities.

  The expense to be incurred–architectural fees, construction cost, and employee relocation outlay–was substantial, however. In addition, management harbored serious anxieties related to the company's ability to change in the future. Senior managers were concerned whether the location of the planned move would confine their operational structure. Management realized that in moving from its current urban location, Trident Manufacturing would be cutting itself off from the numerous services–like availability of rental space and a market for subleasing–that are common to a metropolitan environment. The primary question facing them was, "Would the new physical plant in its proposed location, and built at great expense, be flexible enough to continue serving a changing organization?" What, for example, would happen if Trident Manufacturing decided to sell a subsidiary operation or to decentralize certain operations that were now centralized or to realign intergroup and reporting relationships?

Together with the consultants, management at Trident Manufacturing determined that recent developments in computer technology, such as those offered by DECnet, facilitate change. Local area networks, for example, permit those changes in information flow that accompany reporting and administrative realignment to be put in place rapidly, without disrupting ongoing business. Computer networking, Trident Manufacturing management learned, provides a measure of protection to capital investment.

Managing a network involves configuring it, maintaining it, monitoring its performance, and detecting and resolving problems before they become crises. Network management also includes planning for the evolution of a network. A network manager is responsible for performing all these tasks and ensuring a smooth-running network. Systems managers, those responsible for the operation of the individual nodes making up a network, can greatly assist a network manager in achieving and maintaining good network performance by closely monitoring the network-related aspects of their specific nodes. System managers are responsible to both local and remote users. They should be aware that, while local applications usually demand the greatest share of system resources, remote users, potentially a very large group, must be assured of each node's response to network applications.

The procedures for performing network management functions can vary from system to system. For instance, the procedure for generating a DECnet/E node is different from that for generating a DECnet-RSX node. RSTS/E is a timesharing system with a software structure quite unlike the structure of the realtime, event-driven RSX-11M. Because each DECnet implementation (DECnet/E, DECnet-VAX, and so on) is an extension of an operating system, the different ways of managing the various nodes reflect differences among Digital's operating systems. This chapter provides an overview of network management functions rather than a description of the actual procedures for performing them. For procedural information, refer to the documentation packaged with a specific implementation.

Network system management functions in DECnet include:

- *Planning for node generation* to tailor DECnet software to suit a specific node's network application. Node-generation planning should be influenced by an overall network plan that establishes

basic network-wide conventions. Implementation of such conventions will assure users of communications and processing capability throughout the network.

- *Generating network software* to build the tailored DECnet software to create an active node.
- *Defining and redefining network parameters* to set the various network parameters whose definitions determine many aspects of a node's role within a specific DECnet configuration.
- *Operating a node*, such as starting up and shutting down a node and the physical lines connected to a node.
- *Monitoring node activity*, including the day-to-day performance of a node by gathering and analyzing logging data that DECnet makes available.
- *Downline loading*, the procedure for loading system and task images from an executor node to satellite nodes such as RSX-11S and communication server nodes.

Network system management responsibilities also include the testing and monitoring functions described in Chapter 11.

- **Network Management Utilities**

Most network management modules reside in the Network Management layer. In addition to these, there are network management modules in the User and Network Application layers. Also, one network management module, the Event Logger, has a queue residing in each lower layer.

The User layer of every DECnet implementation supports *Network Control Program (NCP)*, a utility that interfaces with network management modules in the lower DNA levels. NCP includes a standard set of interactive commands that enables a network or system manager to communicate with lower-level modules by issuing commands from a terminal.

Table 10.1 briefly describes the function of NCP and other network management utilities in each DECnet implementation. NCP accepts commands that activate network management modules in the lower layers to perform specific tasks or to request information about the current state of the local node or the network.

The utilities or modules in the Network Management layer include:

Network Management Listener, which receives network management commands via the Network Information and Control Exchange (NICE) protocol from the Network Management layer of remote nodes. This utility enables a network manager to monitor the status and activity of remote nodes from a node at a central location.

Network Management Access Routines, which provide generic network management functions. These routines communicate across logical links with the Network Management Listener using the NICE protocol.

Local Network Management Functions, the component that takes function requests from the Access Routines, translating the requests into system-dependent calls.

Link Watcher, which senses service requests on a link from an adjacent node and handles state changing for automatic remote load, dump, and trigger functions.

Maintenance Functions, which provide the actual protocol operation to support system maintenance functions like downline loading and data link loopback testing.

Link Service Functions, the module to which the higher-level network management modules interface for services that require a direct data link (bypassing the Session Control, End-to-End Communication, and Routing layers).

Event Logger, which records significant events, like a line going down, occurring in the lower layers. DNA specifies event types in the Network Management interface to each layer. An event processor in the Event Logger takes *raw events* queued in each layer and records events of the types specified by the system and system manager. Using the Event Logger protocol, an event transmitter in one node can inform event receivers at other nodes of event occurrences. Events travel to a node specified by the network manager as the *sink node*, where they are output on console, file, or monitor. This log of events at the sink node helps network managers identify problem areas in the network.

Figure 10.1 shows the relationship among the network management components in a node.

In the Network Application layer, *Loopback Access Routines* and *Loopback Mirror* enable logical-link loopback tests. The Loopback Access Routines in a node use the Loopback Mirror protocol to loop test messages from the Loopback Access Routines of a remote node.

- **Network Management Messages**

The NICE protocol handles most functions of the Network Management layer. Table 10.2 lists and describes NICE messages.

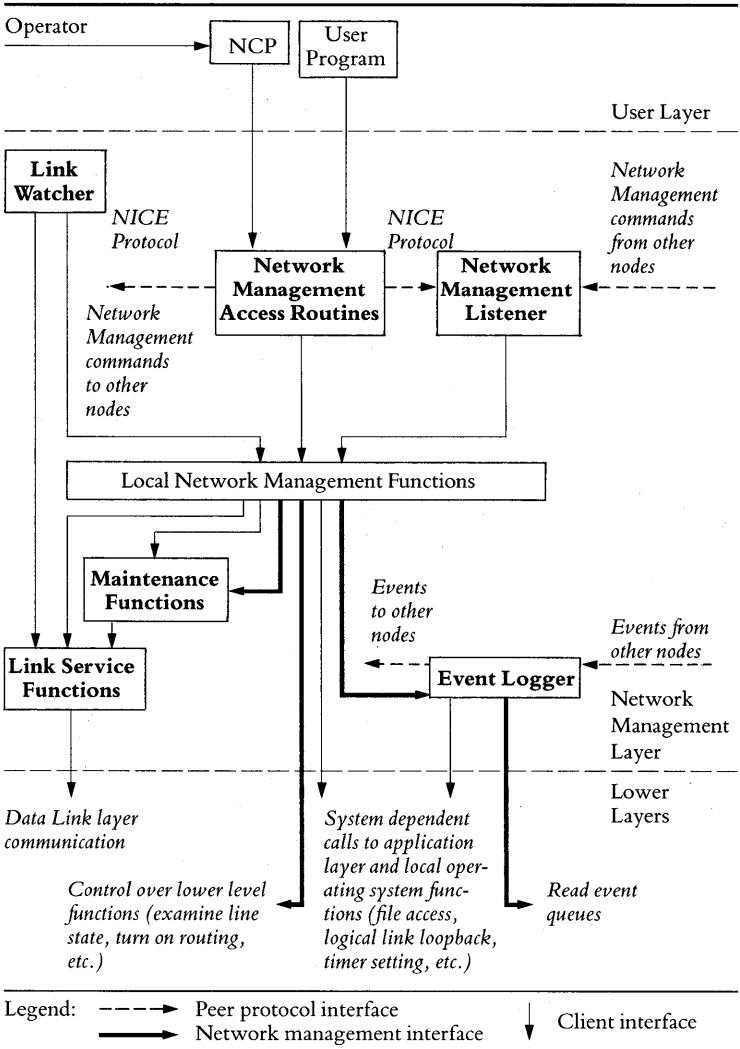


Figure 10.1
Relationship Among Network Management Components at a Single Node

System	Utility	Function
DECnet-VAX	NCP	Loads, controls, monitors, and tests DECnet software; defines configuration database parameters; downline-loads DECnet-11S and communication-server nodes.
DECnet-RSX	NCP	Loads, controls, monitors, and tests DECnet software; down-line loads DECnet-11S and communication-server nodes.
	CFE	Changes parameters in the configuration file CETAB.MAC, which is produced at network generation.
	VNP	Changes the disk image of a DECnet-RSX system (VNP cannot be run from a DECnet-11S node).
DECnet/E	NCP	Loads, controls, monitors, and tests DECnet software; maintains the DECnet/E parameter file; reports the current status of active logical links, known physical links, remote nodes, and programs using local and remote send/receive services.
DECnet-RT	NCP	Loads, controls, monitors, and tests DECnet software; defines and changes configuration database (CETAB.MAC) parameters.
DECnet-20	NCP	Loads, controls, monitors, and tests DECnet software; defines the configuration database; downline-loads system and task images.
DECnet-10	NCP	Loads, controls, monitors, and tests software; defines the configuration database; downline loads system and task images.

Legend:

CFE—Configuration File Editor
NCP—Network Control Program
VNP—Virtual Network Processor

Table 10.1
DECnet Systems and Network Management Utilities

Message	Description
Request Down-line Load	Requests a specified executor node to down-line load a target node.
Request Up-line Dump	Requests a specified executor node to dump the memory of a target node.
Trigger Bootstrap	Requests a specified executor node to trigger the bootstrap loader of a target node.
Test	Requests a specified executor node to perform a node, circuit, or line loopback test.
Change Parameters	Requests a specified executor node to set or clear one or more Network Management parameters.
Read Information	Requests a specified executor node to read a specified group of parameters or counters.
Zero Counters	Requests a specified executor node to either read and zero or zero a specified group of counters.
System Specific	Requests a system-specific Network Management function.
Response	Provides request status and requested information in response to a NICE request.

Table 10.2
NICE Messages

The Event Logger protocol uses only one message–the Event message. This message provides event-related information such as:

- The node at which the event is to be logged (the event sink) and whether the event is to be logged to the sink's console, file, or monitor.
- The event type and class.
- The data and time the event occurred at the source node.
- The name and address of the source node.

Message	Description
Command	Requests a loop test and sends the data to be looped.
Response	Returns status information and the looped back data.

Table 10.3
Loopback Mirror Messages

- Whether the event relates to a DECnet module, node, circuit, or line.
- Specific data concerning the event.

A third protocol in the Network Management layer is the Loopback Mirror protocol, which handles node loopback functions. Table 10.3 describes the messages of this protocol.

The Maintenance Operation Protocol (MOP) is another Network Management protocol. MOP supports special primitive functions between the Network Management and Data Link layers. These functions do not require the services of any of the layers between the Network Management and Data Link layers. Nodes that are being downline-loaded or are upline-dumping cannot support more than a minimal data link; such nodes would not therefore be able to use the services of any layers above the Data Link layer. MOP supports downline loading, upline dumping, loopback testing, and system console control for a remote node and for unattended nodes.

- **Planning for Node Generation**
  This process entails gathering and consolidating information. Digital-supplied DECnet software provides generalized network

capabilities, but users must supply the data and programs that create a complete and working DECnet application. The system managers for the various network nodes accumulate the data and programs relevant to their specific systems so as to incorporate the data and programs eventually into the nodes for which they are responsible.

To ensure a smooth-running network, system managers should exchange information regarding their nodes that is appropriate to network operation. Programmers responsible for network applications should cooperate with system managers. For example, programmers must use correct addresses in calls that generate connect requests (see Chapter 5). System managers must know the correct names and object types that the network programs use to identify themselves.

Configuration Databases–Every DECnet node has some form of configuration database that defines characteristics of that node and determines how it functions within the network. In some cases, Digital-supplied software already includes such a database to provide initial default values for many database entries. For other implementations, the network generation procedure creates the database. Table 10.4 lists the term that each DECnet implementation uses to identify its configuration database.

Depending on the type of DECnet node, the configuration database may need to be updated periodically to reflect changes in the network or to tune the performance of the network. See below under *Defining Configuration and Other Static Parameters* for an explanation of the parameters typically included in a configuration database. Before a node can participate in a network, its system manager must define the configuration database.

System	Term	Comments
DECnet-VAX	Configuration Database	The initial database is provided within the DECnet software supplied by Digital. NCP commands are subsequently used to modify the database.
DECnet-RSX	Configuration File	File CETAB.MAC is created during network generation and can be subsequently modified by the Configuration File Editor (CFE).
DECnet/E	Parameter File	File $NETPRM.SYS is created and subsequently modified by NCP commands.
DECnet-RT	Configuration File	File CETAB.MAC is created during network generation and can be subsequently modified by network management commands.
DECnet-20	CONFIG.CMD	File CONFIG.CMD is shipped with the TOPS-20 monitor and modified by the installer to define the local hardware configuration.
	NETPAR.MAC	The network-configuration file containing system-specific parameters.
	CETAB.MAC	File CETAB.MAC is created during network generation and can be subsequently modified by network management commands.
DECnet-10	Configuration Files	Files mapping.CNF and CETAB.MAC are created during network generation and can be subsequently modified by network management commands.

Table 10.4
Configuration Database Terms

Network Generation Planning Aid–DECnet-RT aids users in planning for node generation. The aid consists of a command file with questions pertaining to node generation options. A system manager runs this command file from a terminal and answers the questions according to the requirements of the local node.

Using the system manager's responses, the command file generates several worksheets. Each tells the system manager how to generate an aspect of DECnet, in accordance with the application requirements of that node. See the network generation manual for a DECnet-RT system for a complete description of the command file and the worksheets it generates.

- **Generating Network Software**

DECnet software arrives from Digital on such distribution media as magnetic tapes or floppy disks. The type of medium depends on the DECnet implementation and, in some cases, on the hardware configuration of a specific system. DECnet-RSX is distributed on several types of media to accommodate a variety of user hardware configurations. The procedures for using the distributed software to generate an active node are different for each implementation of DECnet. To generate an RSX node, a system manager has to regenerate the operating system first and then tailor and build the network application as a second step. Other implementations do not require a system manager to rebuild the distributed software.

The types of DECnet software modules distributed with a system depend on the DECnet implementation and the specific network application that the system supports. In general, distributed DECnet software includes network device controllers, a DDCMP module, an Ethernet protocol module, Routing layer modules, an NSP module, and network utilities like NCP, NFT, and TLK.

Defining Configuration and Other Static Parameters

The configuration database of a node contains parameters which are internal network values that have a significant impact on network operation and performance. These values are of two types: *characteristics* and *status parameters*. Characteristics remain constant until changed by a network or system manager. *Node name* is a characteristic. Status parameters are dynamic and reflect the condition of lines, circuits, modules, or loggers. For example, *line state* is a status parameter. If a failed line comes up or a working line goes down, the change in line state is recorded in the configuration databases of the nodes in the network. System and network managers can examine status parameters.

In addition to defining parameters that are part of a configuration database, a system manager must define non-configuration parameters, such as local node network object descriptions (see Chapter 5).

Parameters can be defined in various ways, depending on the DECnet implementation involved. They can be defined at network generation by means of NCP commands. They may be predefined in Digital-supplied software. The paragraphs that follow offer brief descriptions of the parameters that are typically defined for most types of DECnet nodes.

Node Addresses and Names

For every node in the network, the system manager must assign a numeric address that uniquely identifies that node within the network. See Chapter 2 for a discussion of node addresses and node names.

Node Verification Passwords

Whenever one of its physical lines is turned on, a node exchanges initialization messages with the remote node at the other end of

the line. (See below, under *Controlling Line or Circuit State* for a discussion of starting up a physical line.) The nodes exchange information like the version numbers of their DECnet software modules or their node type–routing or nonrouting.

After exchanging initialization messages, a node can request that the adjacent node supply a password to facilitate verification of its identity. If a local node requires identity verification, adjacent nodes must supply passwords to gain access to the local node. If the local node does not need to verify the identity of adjacent nodes, they do not have to supply passwords, and they automatically gain access to the local node after exchanging initialization messages.

Within a DECnet configuration that enforces node verification, each node maintains a database of passwords that it sends to and expects to receive from its neighbors. *Receive password* is the password that the local node expects to receive from the adjacent node. If the password actually received does not match the receive password expected, the local node denies access to the adjacent node. *Transmit password* is the password that the local node sends to the adjacent node. The transmit password must match the receive password that the adjacent node expects from the local node.

Network Object Parameters

Most nodes maintain a database that describes all the network objects–both user-written programs and DECnet modules–currently residing in the node and capable of engaging in network activity. (For a description of network objects, see Chapter 5.) The manner in which the node's object database is maintained depends on the DECnet implementation on that node. Typically, a node stores object types, names, and addresses; access-control and verification information associated with each object; and the number of copies of a specific object that DECnet can run to satisfy incoming connect requests.

Routing Parameters

Parameters that affect the operation of the Routing module in Phase III and Phase IV full-routing nodes include:

- *Maximum path cost*, a value that limits possible routes between two nodes to paths that are equal to or less than this value.
- *Individual line costs* that figure in routing algorithms used by the Routing module. Each line cost is a number from 1 to the maximum path cost set for the node. Higher line cost can lead to less traffic on the line because the Routing modules dispatch packets on the least costly paths. Lowering the cost of a line does not always increase the traffic it carries. For example, if a line leads to an end node, line cost does not affect that line's traffic (since all traffic arriving at an end node is intended for it). If one of a node's lines is more expensive than others to use, the traffic on that line can be diminished by raising the routing line cost assigned to it. In fact, this can be done whenever the system manager wants to decrease traffic on a line.
- A *maximum number of hops per path* that sets the limit on the number of circuits a packet can traverse to arrive at its destination. A node is unreachable if it cannot be reached within the maximum number of hops.
- *Area maximum cost* is a parameter for an area-based network. It limits possible routes between Level 2 routers in different areas to paths with cost equal to or less than this value.
- *Area maximum hops*, another area parameter that sets the limits on the number of circuits a packet can travel between Level 2 routers in different areas.
- The *maximum area* parameter sets the limit on the number of areas a packet can traverse before arriving at its destination.
- A *routing timer* that determines the interval between automatic updates of the local node's routing database.

- A *buffer size* for the unit of data actually transmitted over physical lines by the Routing module.
- A *buffer count* that determines the size of the Routing module's pool of available buffers.

Line Identification

Each communication line leading from a node has a unique identification that is used in commands. A node's configuration database usually contains the identification of each communication line. This identification has the following general format:

dev-c-u, where *dev* is a mnemonic for the type of device. Examples include UNA (for the DEUNA Ethernet communications controller), DUP (for the DUP11-DA), DMC (for the DMC11-DA/AR, -MA, AL, or FA/AR), DZ (for the DZ11-A or -B), and DMR (for the DMR11).

c is a number (zero or a positive integer) designating the device's hardware controller.

u is a unit/line number (zero or a positive integer) that is included if the device is a multiplexer.

Examples of typical line identifications include:

Identification	Description
UNA-0	DEUNA, controller 0
DMR-0	DMR11, controller 0
DZ-1-0	DZ11, controller 1, unit 0

Circuit Parameters

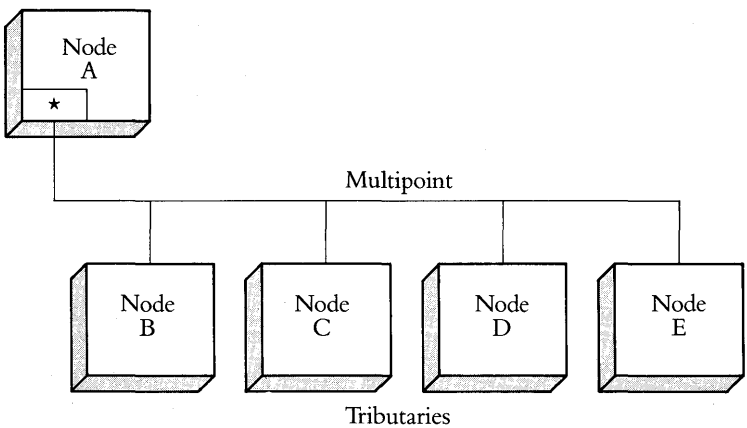
In addition to identifying actual physical lines, most DECnet implementations also identify circuits, which are logical point-to-

point communications paths operating over physical lines (see Chapter 2). At nodes with circuit-identifying implementations, a network manager defines and manipulates circuits, rather than lines, to control the flow of data between nodes. DECnet-VAX, DECnet-RSX, DECnet-20, DECnet-10, and DECnet/E users frequently specify circuits, while users at other DECnet nodes specify lines in equivalent network management commands.

Definitions of circuit parameters are based on the type of link (DDCMP point-to-point, DDCMP multipoint, or PSI virtual circuit) between nodes. When a circuit corresponds to a point-to-point line, circuit IDs and line IDs are identical. For example, the string "DMC-0" in a network management command can identify either the line or the circuit of a point-to-point configuration. In multipoint configurations, circuit and line identifications are not identical. See below for an explanation of how they differ. At a DECnet-RSX/PSI node, Data Link Management (DLM) circuits to be mapped to the PSDN are not associated with specific physical lines. The identification of such a circuit starts with the mnemonic DLM.

Multipoint Line And Circuit Parameters–A multipoint line is a single communications line connected to more than two nodes. (A line connecting two nodes is called a point-to-point line.) The DECnet implementations supporting multipoint are the Phase III and Phase IV versions of DECnet-VAX and DECnet-RSX and the Phase III versions of DECnet/E and DECnet-RT. Figure 10.2 shows a multipoint line with its various components.

The *control station* is the device responsible for overseeing data transmission to and from all the nodes attached to the line. The devices attaching the other nodes to the line are called *tributaries*. A DECnet-RSX, DECnet-VAX, or a DECnet/E node can support either a control station or a tributary device. A DECnet-RT node can support only a tributary device on a multipoint line.



★Device controller = control station for multipoint line:
DMP11 synchronous link
DMV11 synchronous link

Figure 10.2
A Multipoint Line

From the perspective of the control station, the multipoint line and the tributaries connected to it constitute a single line. But the separate paths to each tributary represent individual circuits. A multipoint line, therefore, has more than one circuit associated with it. (Figure 2.1 illustrates the relationship between a multipoint line and the circuits that correspond to its tributaries.)

A tributary supports only one physical link to the control station. From the tributary's perspective, the DDCMP line that links it to the control station is point-to-point. To the tributary, the line and the corresponding circuit are the same.

In the context of full-routing implementations, whether or not a node supports a control station or a tributary on a multipoint line is not significant. The mechanisms for handling data transmission on multipoint lines are transparent to the modules in the Routing layer of the architecture.

At the control station's node, a system manager needs to define several parameters that affect the operation of the multipoint line and its corresponding circuits:

Tributary addresses: The database at the control station's node must contain correct tributary addresses. The system manager must therefore record the unique decimal line address of each tributary on the line.

Polling ratios: The control station delivers data addressed to tributaries under its control. All data transfer between tributary nodes takes place through the control station. In order to transmit data originating from its tributaries, the control station polls each tributary periodically and asks whether it has data waiting to be transmitted. If the polled tributary wishes to transmit, the control station permits it to send data. If a polled tributary does not wish to transmit, the control station goes on to poll the next tributary on the line.

The frequency with which a tributary is polled depends on the frequency of its data transmissions. For efficient network operation, the control station polls active tributaries more often than inactive or dead tributaries. A dead tributary is one that has not responded within a predefined period of time.

Network or system managers can exercise control over the polling of specific tributaries. If system managers decide that a tributary

should not be polled as often as others, they can issue a command to assign an appropriate active polling ratio to that tributary. A command can also be issued to set a dead polling ratio that applies to all inactive tributaries.

A polling ratio is a number from 1 to 255. If a tributary's active polling ratio is 5, the control station passes through the active polling list five times before polling that particular tributary. (DECnet/E does not implement the polling technique discussed here.)

Transmission Mode

The system manager for a node sets the transmission mode for every line or circuit connected to that node. The transmission mode is either half-duplex or full-duplex. See Chapter 2 for a definition of these terms.

- **Operating a Node**

To start up a node, a network or system manager issues commands from a terminal to load and activate required DECnet software and to turn on physical communication lines. In response to the startup procedure, the node's DECnet software initializes the DECnet software in adjacent nodes connected by the activated lines (see below, under *Controlling Line or Circuit State*). To shut down a node, the network or system manager issues commands to halt network activity that involves that node, to shut down physical lines, and to unload DECnet software.

Controlling the State of a Node

For most implementations of DECnet, a system manager turns a node on and off by manipulating the state of that node. For instance, the command NCP>SET EXECUTOR STATE ON activates the DECnet software at the node currently defined to be the executor.

The executor is the node at which the NCP command actually executes. Using an NCP command, the system or network manager can determine whether the executor is the local node or a remote node.

Most DECnet implementations define three distinct node states: ON, SHUT, and OFF.

ON–The local node is enabled for performing network functions.

SHUT–The node maintains all existing logical links but does not permit any new links to be created. When existing links are disconnected, the node's state changes to OFF.

OFF–The local node cannot participate in any network activity, and existing logical links are aborted.

Controlling Line Or Circuit State

A node cannot actively participate in a network until at least one communication line or circuit has been turned on. By issuing an NCP command, a system manager sets the state of a line or circuit to ON, OFF, or SERVICE.

ON–The line or circuit is available for use by the modules in the Routing layer. When a line or circuit is turned on, the local node exchanges initialization messages and, optionally, node passwords with the remote node at the other end of the line or circuit.

OFF–The line or circuit is not available for any kind of network activity.

SERVICE–The line or circuit is available only for special network functions like downline loading, upline dumping, and loopback

testing. Some implementations of DECnet do not recognize SERVICE as a state distinct from ON; such implementations may impose the SERVICE state on a line or circuit by their own internal means.

When a line or circuit is in the ON state, DECnet software exercises the data link protocol that ensures data integrity and sequentiality for normal network transmissions. (The standard DECnet protocol for normal traffic is DDCMP.) In the SERVICE state, a line or circuit embeds data in a protocol provided for special network functions. The standard protocol for these functions is the Maintenance Operation Protocol (MOP).

- **Monitoring Node Activity**
  At each node, DECnet continuously monitors the current state and ongoing performance of the node. From a terminal on a node, a system or network manager can use network management utilities to display the following information:
- The current state of the local node, remote nodes, and physical lines.
- Values currently defined for the node's configuration database and other static parameters.
- The contents of various counters that DECnet software maintains to track network performance.

In addition to displaying information on request, DECnet automatically logs certain events both at the operator's console and in a file. Event logging records operational events such as a line starting up or shutting down. DECnet-RT does not support event logging.

Monitoring Remote Systems or System Console Control–The Maintenance Operation Protocol can be used to monitor and control remote and unattended systems. From an executor node, a

network manager can monitor Phase IV communication servers (the DECnet Router, the DECnet Router/X.25 Gateway, and the Terminal Server). By means of MOP's *Reserve Console* message, an executor node can take control of the console of a remote system and proceed through a complete dialogue with the remote system (the executor sends *Console Command and Poll* messages and the remote target node returns *Console Response and Acknowledge* messages). At the end of the dialogue, the executor sends a *Release Console* message.

Counters–DECnet uses counters to track other kinds of events and errors in addition to line startups and shutdowns. Periodically, a system manager can record these counters in a file or display them at a terminal to obtain detailed statistics on the node's network activity, such as how many connect requests have been sent and received and how many messages have been sent and received over logical links.

In Phase III and Phase IV implementations, counters record Routing layer activity as well. This includes how many errors of different kinds have been found in packet headers and how many line initialization and line verification failures have occurred. Certain Routing counters enable a system manager to determine if too many packets are being discarded due to network congestion. If this is the case, the system manager can adjust the parameters that control the paths over which a packet is routed, thereby improving the distribution of traffic loads over network circuits.

Line counters monitor events on individual communication lines. By periodically checking each line's counters, a system manager can assess its performance and be aware of potential line problems. Line counters record statistics such as the number of data blocks sent and received successfully, the number of blocks received with errors, and the number of times a tributary has passed from an active to a dead state.

After counters have been displayed or recorded in a file, a system manager can issue NCP commands to set the counters to zero. In this way, the system manager can manipulate the time span that the counters monitor. For instance, the system manager could set all node counters to zero as programmers begin to test a network application. At the end of the test, the counters could be examined to see how the application affected the node's performance. Chapter 11 discusses some of the tools that the system manager can use in testing network components and in monitoring network activity.

The Network Management layer has been designed to limit as much as possible the overhead to the network in gathering statistics and errors. For example, there are no Routing counters that detect bugs in the Routing code. Network Management modules assume the code is correct. The Routing layer algorithms themselves have been designed to detect and recover from most failures that would disrupt network operation.

- **Downline Loading**

DECnet-VAX, DECnet-RSX, DECnet-IAS, and DECnet-20 support downline loading, which means loading a memory or system image from a file at one node onto a separate target node. The target node is usually an RSX-11S DECnet node or a communication server node (DECnet Router, DECnet Router/X.25 Gateway, or Terminal Server.) These memory-only systems have no disk-based file storage of their own.

A DECnet-20 node loads a system image downline to a specially adapted RSX-11S node called a DN200, which supports a cardreader and a lineprinter and which serves as a remote batch station. The DN20, a communications front end, is also downline-loaded from a DECnet-20 node.

At a DECnet-VAX, DECnet-RSX-11M, or DECnet-RSX-11M-PLUS node, the system manager generates the target system image. Once this is done, the image can be modified by Virtual Network Processor (VNP) commands at an executor node. The load itself can be initiated in one of two ways. An operator can issue NCP commands to load the image downline to the target node, or an operator at the target can initiate the load by triggering a bootstrap ROM (read-only memory). See below, under *Performing a Downline Load* for an explanation of these procedures.

Upline dumping is a function that complements downline loading. Chapter 11 describes upline dumping.

Downline-Loading Definitions

The downline loading function is distributed among two or more nodes in a DECnet configuration. The *command node* is the node from which the system or network manager issues NCP load commands. The *executor node*, which executes the NCP commands, must be adjacent to the target node. The *target node* receives the system image loaded down the line or dumps a system image up the line. A single node can act as both the command and the executor nodes.

Downline-Loading Database Parameters

For every target node that has to be downline-loaded, the executor has access to a permanent database. Each database contains default parameters, such as node verification passwords and other account information, for downline-loading a specific target node. The system manager can override these defaults by providing parameter values in an NCP LOAD command. The parameters are defined initially at network generation and can be redefined when necessary.

Performing A Downline Load

Whether a remote command node or the target initiates the load, the target must have local access to a cooperating program called a primary loader. This loader is usually contained in a bootstrap ROM (read-only memory) incorporated in the target. During the downline-load procedure, a series of programs can be loaded above the primary loader. Each program calls the next until the entire system image is downline-loaded.

The line or circuit between the executor and the target is in a SERVICE state during the procedure. Either the system manager sets the state to SERVICE, or the DECnet software sets it automatically. How the state is set depends on the DECnet implementation on the executor and on the manner in which a load is initiated.

The NCP LOAD command is the means of initiating a downline load from a remote command node. As soon as the LOAD command has been issued, an operator at the target must trigger the bootstrap ROM manually, unless the service line's device controller is a DMC11 or DMR11 device. These devices can trigger the target's primary loader automatically if the LOAD command passes down the correct password. Figure 10.3 illustrates a downline load initiated by a command node.

In a *target-initiated downline load*, an operator at the target node requests a load by triggering the primary loader manually. In addition, the loader is triggered automatically at the completion of an upline dump from the target. Target-initiated downline loads always use the parameter values defined in the permanent database for the target. Figure 10.4 illustrates a target-initiated downline load.

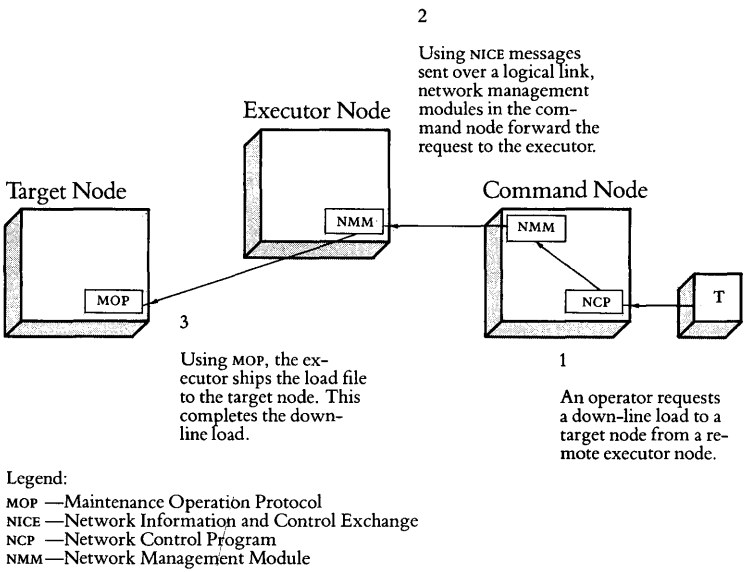


Figure 10.3
Downline Loading Initiated by a Command Node

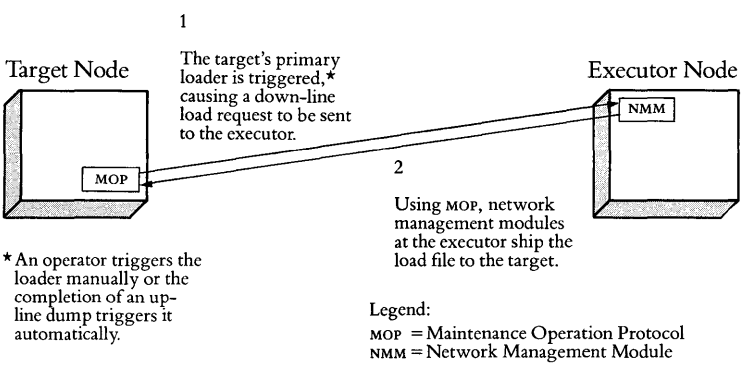


Figure 10.4
Downline Loading Initiated by a Target Node

Downline-Loading And Checkpointing RSX-11S Tasks

DECnet-11M, DECnet-11M-PLUS, and DECnet-VAX support two capabilities relating to a DECnet-11S node. The first is called downline taskloading. RSX-11S tasks can be stored at a DECnet-11M, DECnet-11M-PLUS, or DECnet-VAX node and loaded to the RSX-11S node. The second is called checkpointing, which is a standard RSX-11M capability. An executing RSX-11S task can be interrupted, copied in its interrupted state up the line, and then be replaced by a higher-priority task that is loaded downline from the executor. When the higher-priority task is completed, the RSX-11M executor reloads the interrupted task downline so that the task can continue executing.

At a DECnet-11M or DECnet-11M-PLUS node, the operating system regularly checkpoints tasks to local disk storage. However, RSX-11S nodes are basically memory-only systems, so the only way to checkpoint tasks is to use the executor's disk storage.

Downline loading and checkpointing give great flexibility to an RSX-11S node. To change the set of resident tasks at a stand-alone RSX-11S system, an operator has to reboot with a different system image. See the *DECnet-RSX System Manager's Guide* and the *DECnet-VAX System Manager's Guide* (available through a Digital sales representative) for further information.

Maintenance Operation Protocol–Table 10.5 lists and describes MOP messages that relate to downline-load, upline-dump, and loopback test operations.

Either the target or executor node can initiate a downline load. In either case, the target sends a *Program Request* message to tell the executor what is needed and to indicate its willingness to cooperate. The major message exchange consists of *Memory Load*

messages from the executor and *Request Memory Load* messages from the target. The sequence is completed when the host sends a *Memory Load with Transfer Address* or a *Parameter Load with Transfer Address* message (consult Table 10.5 for explanations of these messages).

On an Ethernet data link, a target can multicast its Program Request message and select an executor from those responding with an *Assistance Volunteer* message.

Upline dumping is very similar to downline loading. The target uses the *Request Dump Service* and *Memory Dump Data* messages. The executor controls the process with the *Request Memory Dump* and *Dump Complete* messages (see Table 10.5 for message descriptions).

Message	Description
Memory Load with Transfer Address	Causes contents of the image data to be loaded into memory at the load address, and the system to be started at the transfer address.
Memory Load without Transfer Address	Causes the contents of the image data to be loaded into memory at the load address.
Request Memory Dump	Requests a dump of a portion of memory to be returned in a Memory Dump Data message.
Request Program	Requests a program to be sent.
Request Memory Load	Requests the next segment of image data in a loading sequence and provides error status on the previous segment.
Request Dump Service	Requests a dump to be taken.
Memory Dump Data	Returns the requested memory image in response to a Request Memory Dump message.
Parameter Load with Transfer Address	Loads system parameters and transfers control to the loaded program.
Dump Complete	Signals completion of a requested dump.
Assistance Volunteer	Indicates willingness to perform dump or load service in response to a Request Program or Request Dump Service message.

Table 10.5
MOP Downline Load and Upline Dump Messages

(Continued)

Loop Data	Contains data to be sent back in a loopback test.
Looped Data	Returns the data from a Loop Data message.
Boot	Causes a system to reload itself if the verification code matches and the system is willing to do so. May result in a down-line load.
Request ID	Causes a system to send a System ID message.
System ID	Identifies the sending system by, for example, maintenance version, maintenance functions supported, processor type, etc.
Request Counters	Causes a system to send a Counters message.
Counters	Returns Data Link counter values in response to a Request Counters message.
Reserve Console	Reserves a system's console for dialogue with the requester.
Release Console	Releases a console reserved with the Reserve Console message.
Console Command and Poll	Sends command data to a reserved console and/or polls for response data.
Console Response and Acknowledge	Returns response data and/or acknowledges receipt of a Console Command and Poll message.

Chapter 11 · Monitoring and Testing DECnet Performance

A network is a highly dynamic environment. Applications and other processes are exchanging information simultaneously over thousands of circuits; functions at various levels are being enabled and disabled; the status of network components is changing rapidly; and internetwork communications could be making multiple demands on DECnet facilities for protocol handling. Warning conditions can arise and fatal events occur in different regions of the network and at various levels of network function. As the number of nodes increases, so does the volume and complexity of this activity.

Maintaining a high level of network availability under such conditions is a challenge to network managers. They must have tools that report on, diagnose, and correct conditions that could cause performance degradation or failures. Facilities must also be available for monitoring network operation so as to detect conditions that could lead to major communications problems.

Several software tools, integrated in DECnet and available separately, are designed to help the network manager in these assignments. The functions of the integrated testing tools include:

- Ensuring that user-specified nodes communicate properly and that messages move over specified circuits and lines without being corrupted.
- Testing operation of X.25 and IBM SNA Gateways to determine whether or not protocols are being translated accurately.
- Triggering memory dumps to help system and network managers establish the cause of node crashes.

Digital has developed *Observer*™, a monitoring tool that is not integrated in DECnet and that is available as a separate product.

Running in a DECnet-RSX node, Observer alerts network managers to incipient network problems and gathers performance statistics that can help them reconfigure and fine-tune the network for optimal performance.

This chapter discusses the integrated testing tools and Observer.

Integrated Testing Tools

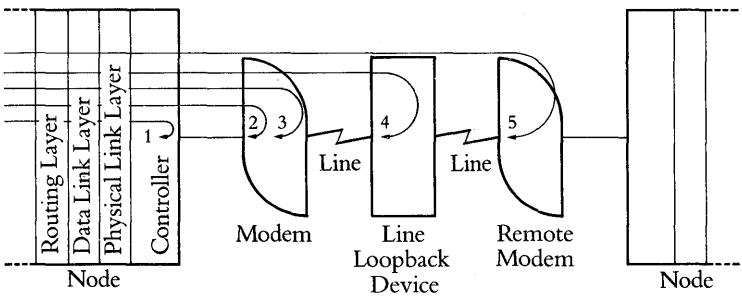
Integrated into most DECnet implementations are tools that enable loopback testing, tracing, and upline dumping.

Loopback Testing

This is a procedure for isolating faults in a connection between two nodes. Loopback tests exercise software and hardware by repeatedly sending data through a number of network components and then returning the data to its source. By changing the loopback point, a network manager can locate problems precisely. Figure 11.1 illustrates the various hardware points from which test data can be looped back to the point of origin. If a test completes successfully, the data loops back to its source uncorrupted. If a test fails, the data either does not return to its source or returns in a corrupted state.

A system manager can run variations of the loopback tests to help isolate the precise network component responsible for losing or corrupting data. Digital software services personnel run loopback tests routinely after installing DECnet software. Successful tests verify that both the software modules and the hardware equipment on the tested path are operating correctly.

The loopback-testing capability is exercised through the NCP LOOPBACK command in combination with other NCP commands.



1 Loopback at controller
2 Loopback at modem on controller side
3 Loopback at modem on line side
4 Loopback at hardware loopback device inserted in line
5 Loopback at a remote modem

Figure 11.1
Hardware Loopback Devices

Loopback enables network and system managers to check whether or not logical links have been established properly between nodes, whether or not DECnet protocols operate correctly within and between nodes, and whether or not messages are being transmitted accurately over the circuits and lines that the managers specify.

Figures 11.2 and 11.3 show the NCP commands used in conjunction with command- and program-initiated node-level loopback tests, respectively. Figure 11.4 shows the NCP commands used in conjunction with line- and circuit-level loopback testing.

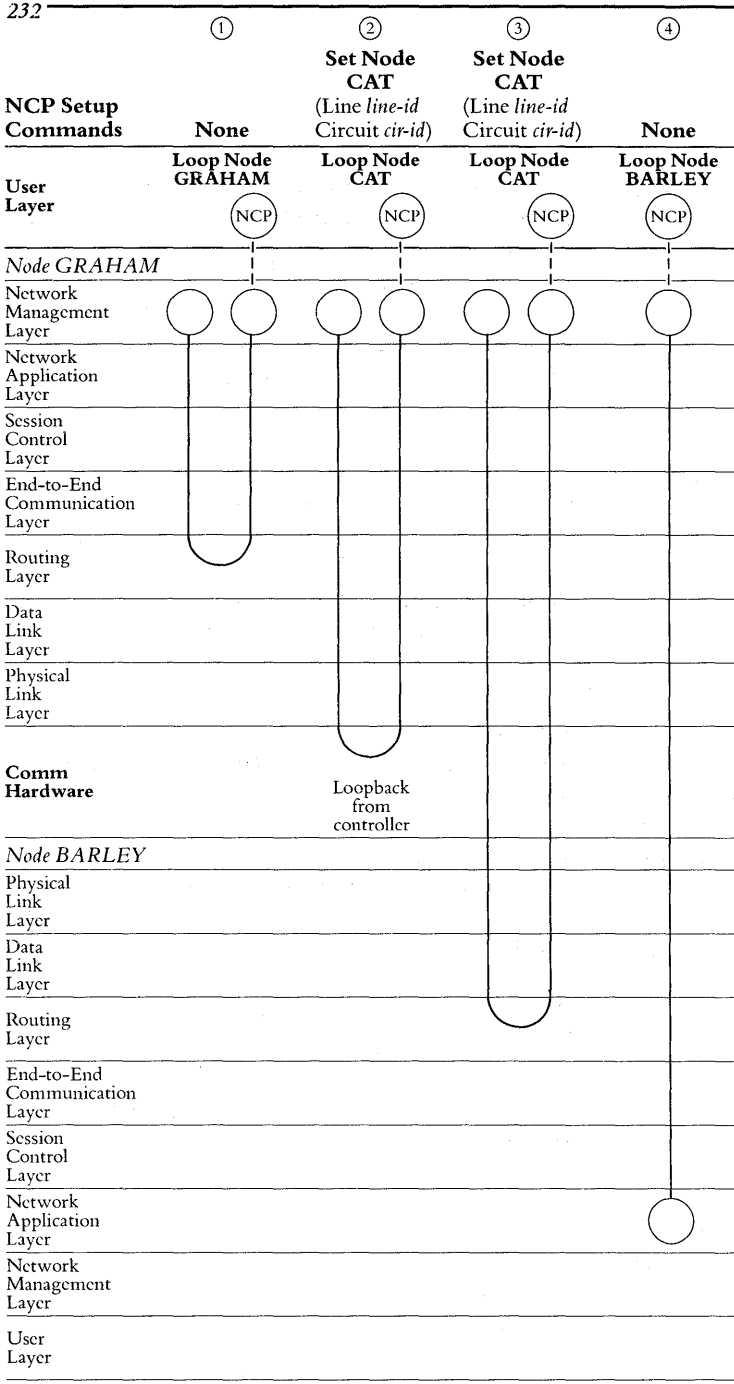
Test data is transmitted from a source node to one of several points–a local modem, a point on the line connecting two nodes, a remote modem, or a remote node–and then looped back from any of these points to the source node. Test results that are printed out indicate whether or not the specified target network component (line, circuit, or node) passed the test. Test messages specify the nature of an error condition that may have caused the test to complete unsuccessfully.

Loopback testing can be performed for general purpose nodes and communications servers over Ethernet, DDCMP, SNA, or X.25 circuits.

Used together, loopback testing of nodes and circuits can help network managers isolate problem areas in the network.

Node Loopback Testing–Node loopback testing consists of sending test messages over a logical link to be looped back at a specific point. The network manager can set a hardware loopback device on a line, line device, or modem between the executor and adjacent target. Alternatively, software components can loop back test data. Different software components can be used. For example, FAL, a user program, or the Loopback Mirror can loop data back to their associated access routines. Figures 11.5 and 11.6 illustrate some types of node loopback tests.

One type of node loopback test uses the Network Management Loopback Access Routines and Loopback Mirror residing in the Network Application layer (Figure 11.5 B and Figure 11.6 C). These modules communicate over logical links using the Loopback Mirror protocol.


①
②
③
④
NCP Setup Commands
None
Set Node CAT (Line line-id Circuit cir-id)
Set Node CAT (Line line-id Circuit cir-id)
None
User Layer
Loop Node GRAHAM
Loop Node CAT
Loop Node CAT
Loop Node BARLEY
NCP
NCP
NCP
NCP
Node GRAHAM
Network Management Layer
Network Application Layer
Session Control Layer
End-to-End Communication Layer
Routing Layer
Data Link Layer
Physical Link Layer
Comm Hardware
Loopback from controller
Node BARLEY
Physical Link Layer
Data Link Layer
Routing Layer
End-to-End Communication Layer
Session Control Layer
Network Application Layer
Network Management Layer
User Layer

Comments	The command loops data within the local node.	The command loops data via loopback node CAT associated with a line whose controller has been set to loopback mode.	The command loops data via loopback node CAT. The line associated with CAT leads to adjacent node BARLEY where data is looped back at the transport layer.*	The command loops data through node BARLEY where network management software in application layer loops data back.

* The routing layer uses its routing algorithm to determine which path the test data will be looped back on.

Legend:

(NCP)	=	the network management software that accepts command input
( )	=	network management software that sends, loops, or receives test data.
——	=	path travelled by test data
- - -	=	interface between modules of network management software

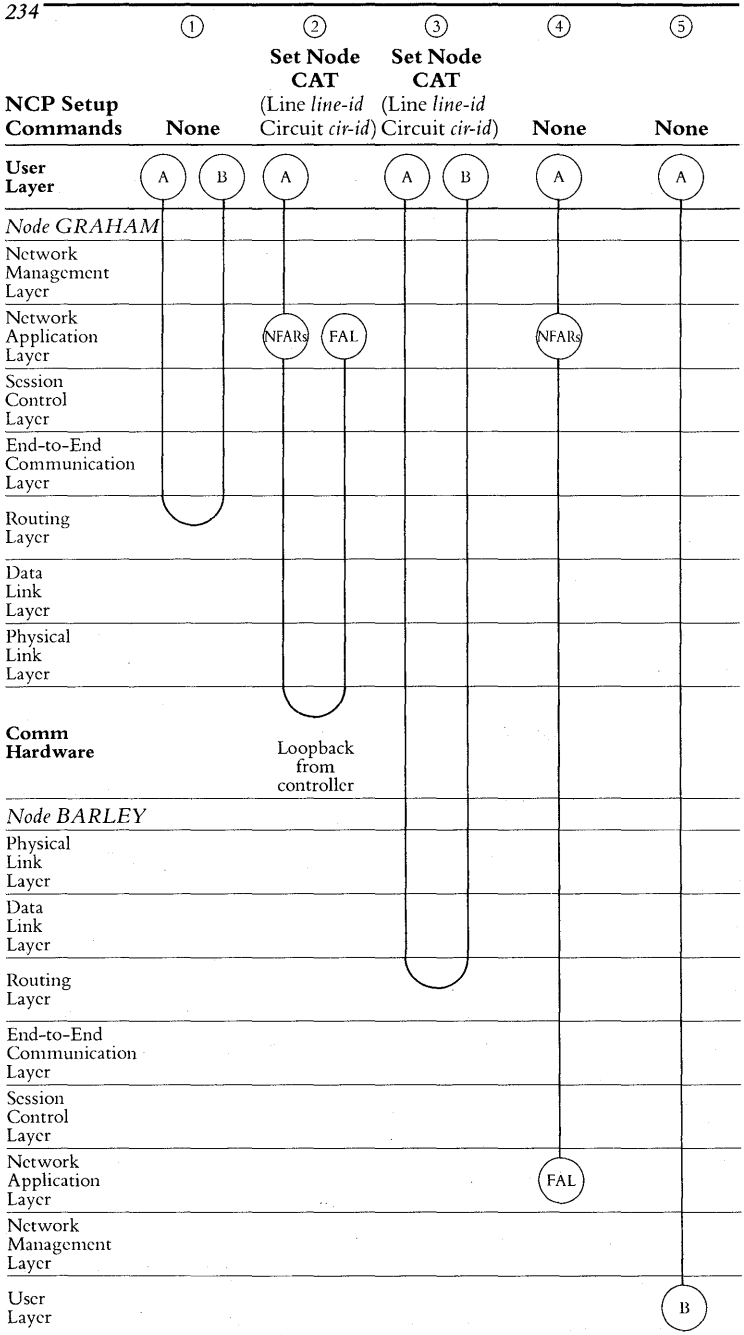
Figure 11.2
Command-Initiated Loopback Tests

Other loop tests use logical links, but they do not use Network Management software. For example, Figure 11.5 B shows a test message traveling from one user task to another. Figure 11.6 D shows a file transfer being used as a logical link test.

By using the NCP SET NODE LINE command, a network manager can set up a special *loop node* path (Figure 11.5). In this case, if the network manager sets no hardware device, the adjacent node's Routing layer loops back the test.

The several methods of performing loopback tests–both hardware- and software-based–give the network manager great flexibility in testing network operation. In addition, the variety of tests enable network managers to identify the exact component causing a problem.

Trace Capability–The trace capability is useful in identifying protocol problems that can arise in the gateway products that handle internetwork communication and in the PSI (Packetnet System Interface) products that enable DECnet and non-DECnet Digital systems to communicate over PSDNs. For the DECnet Router/X.25


1
2
3
4
5
NCP Setup Commands
None
Set Node CAT (Line line-id Circuit cir-id)
Set Node CAT (Line line-id Circuit cir-id)
None
None
User Layer
A
B
A
A
B
A
A
Node GRAHAM
Network Management Layer
Network Application Layer
NFARs
FAL
NFARs
Session Control Layer
End-to-End Communication Layer
Routing Layer
Data Link Layer
Physical Link Layer
Comm Hardware
Loopback from controller
Node BARLEY
Physical Link Layer
Data Link Layer
Routing Layer
End-to-End Communication Layer
Session Control Layer
Network Application Layer
FAL
Network Management Layer
User Layer
B

Comments	Program A sends data to program B, which returns data back to program A. Target node is local node GRAHAM.	Program A uses network file access routines (NFARs) and File Access Listner (FAL) to store and then retrieve file data. CAT, the target node, is a loopback node name associated with a controller set to loopback.	Same as test 1 except target node is a loopback node named CAT; data therefore loops back from an adjacent node.*	Same as test 2 except that target node is BARLEY.	Same as test 1 except Program A and Program B reside in separate nodes.

*The routing layer uses its routing algorithm to determine which path the test data will be looped back on.

Legend:

= user program or DECnet module

= path travelled by test data

Figure 11.3
Program-Initiated Loopback Tests

Gateway, the trace capability is exercised through the NCP TRACE command. For the DECnet/SNA Gateway, the capability is exercised through the SNATRACE utility. NCP TRACE and SNATRACE enable network managers to capture and analyze messages that define how protocols are being handled.

Protocol processing by the DECnet Router/X.25 Gateway can be traced at Levels 2 and 3 of the protocol-handling software. Level 2 defines the link access procedure for data exchange between computers and the data communications equipment (DCE) that establishes, maintains, and terminates communications between computers over a PSDN. The Level 3 protocol defines the procedures for formatting and exchanging packets transmitted over a PSDN.

SNATRACE, used for identifying protocol problems that can arise in using the DECnet/SNA Gateway, enables network managers to examine protocol handling at the SDLC circuit level, the physical-unit level, and the session level. At the SDLC circuit level, network managers retrieve messages on a specified circuit as the messages pass between the SDLC module and the device driver in the Gateway

NCP Setup Commands	① **Set** (Line *line-id* Circuit *cir-id*) **State Service/On***	② **Set** (Line *line-id* Circuit *cir-id*) **State Service/On***	③ **Set** (Line *line-id* Circuit *cir-id*) **State Service/On***
User Layer	**Loop Line** *line-id* (NCP)	**Loop Line** *line-id* (NCP)	**Loop Line** *line-id* (NCP)
Node GRAHAM			
Network Management Layer			
Data Link Layer			
Physical Link Layer			
Comm Hardware	Loopback from controller	Loopback from modem	or from line loopback device
Node BARLEY			
Physical Link Layer			
Data Link Layer			
Network Management Layer			
User Layer			
Comments	The controller must be set manually to loopback mode.	Both modem loopback and line loopback require manual setting or insertion of loopback device.	The line must also be in SERVICE or ON* state at node BARLEY

*SERVICE or ON—depends on the implementation

Legend:

(NCP)	=	the network management software that accepts command input
◯	=	network management software that sends, loops, or receives test data.
——	=	path travelled by test data
– – –	=	interface between modules of network management software

Figure 11.4
Line/Circuit-Level Loopback Tests

node. At the physical-unit level, network managers retrieve messages handled by a specified physical unit as they pass between the SNA module and the SDLC module of the Gateway. At the session level, network managers retrieve messages being exchanged within a specified session as they pass between the SNA and SDLC modules of the Gateway.

Documentation for each of the gateway products describes how to use these facilities.

Upline Dump

In the context of upline dumping, a target node is either a DECnet RSX-11S node or a communications server node. The executor node in both cases is usually the node from which the target node's system was downline-loaded or a backup node of similar capability, in the event that the original executor is not available.

The upline-dump capability enables a target node to initiate automatically a memory dump upline to the executor node. This process occurs when the target node senses an impending system crash. By analyzing the memory dump, software specialists can determine the cause of the system failure and take steps to ensure that similar conditions do not arise again.

In order to keep the target node available to the network, the executor node will automatically downline-load the target node's system image when the dump is completed. For detailed descriptions of the DECnet facilities that operate in upline dumping and of the NCP commands to be used to ensure that lines and circuits specified in the procedure can accommodate the dump, see the documentation of the systems that support upline dumping.

A system manager can trigger an upline dump from an executor node at any time.

A. Local to Loop Node Test, Single Node, using files as test data with hardware loopback.

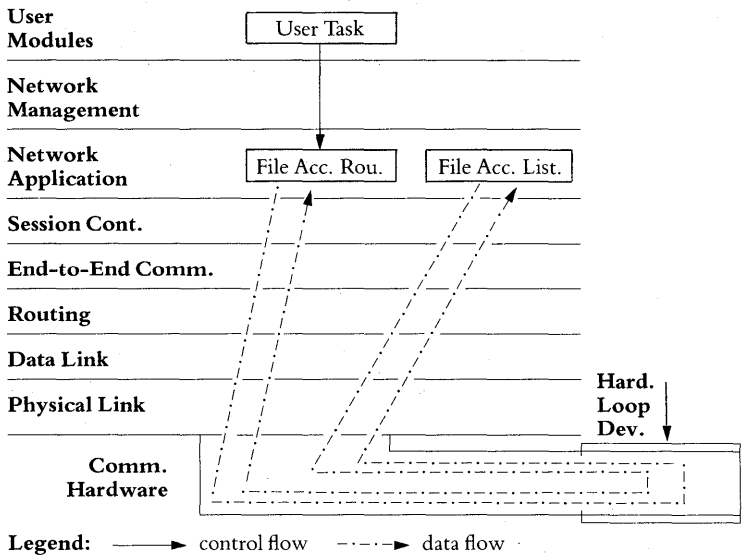
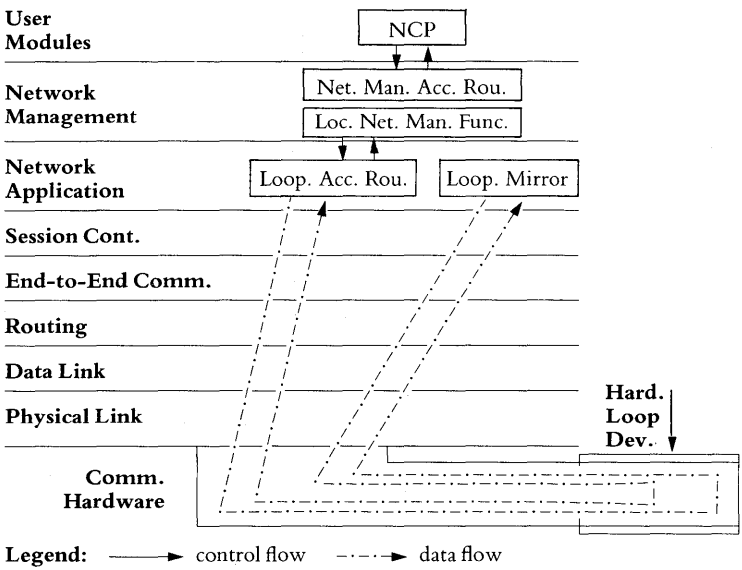


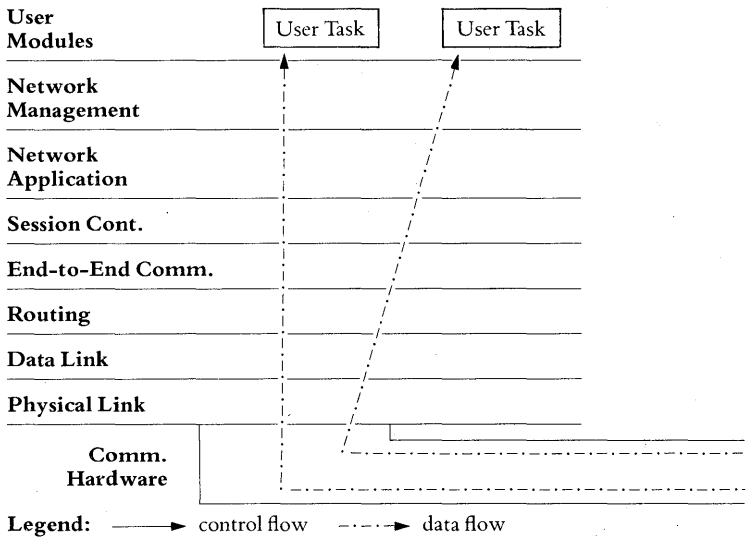
Figure 11.5
Examples of Node-Level Testing Using a Loopback Node

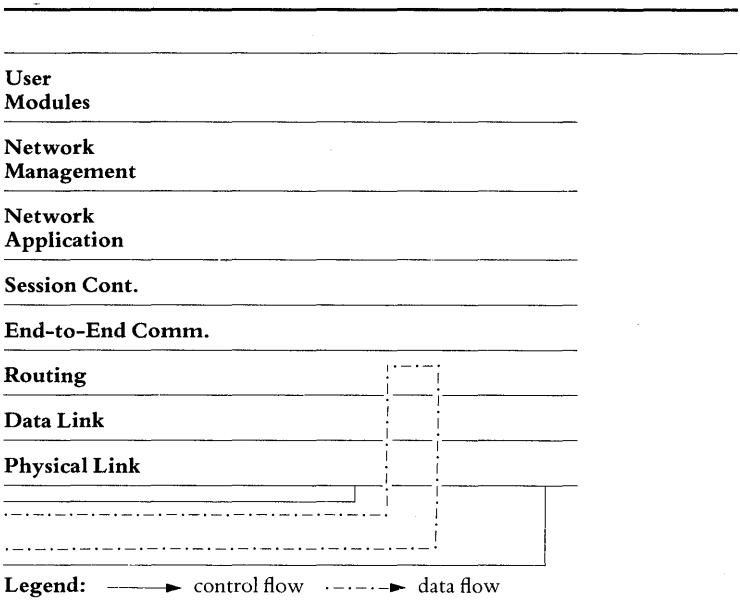
B. Node Test, Single Node, using loopback mirror and test messages with hardware loopback.



(Continued)

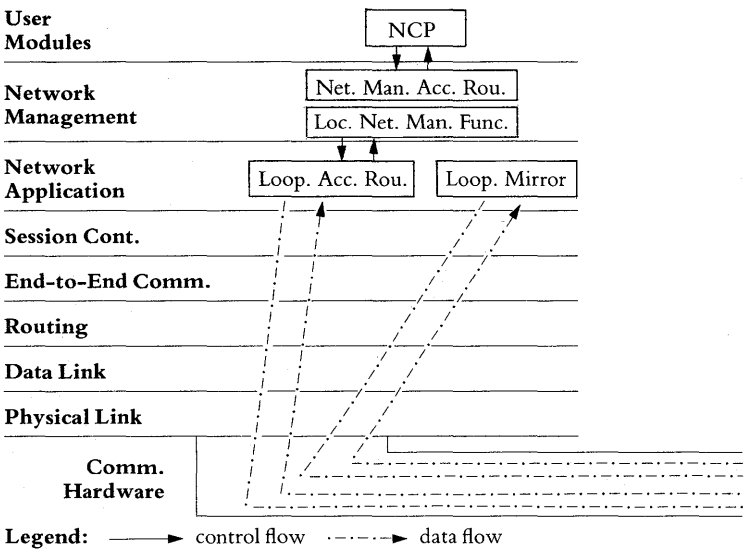
C. Local to Loop Node Test, Two Nodes, using user tasks.

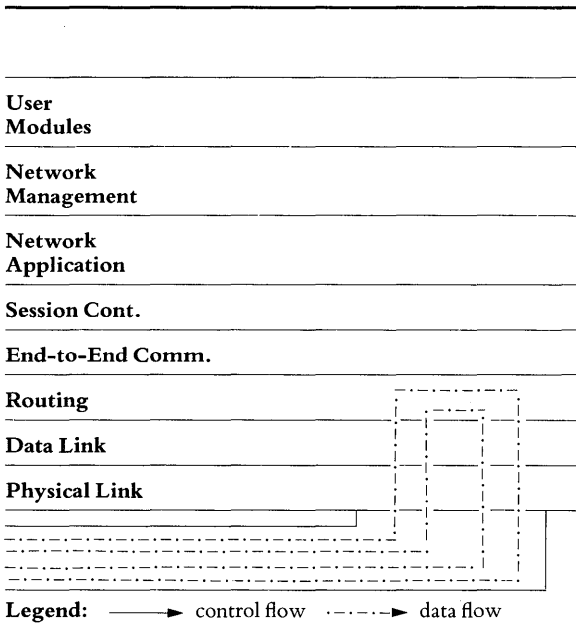



User Modules
Network Management
Network Application
Session Cont.
End-to-End Comm.
Routing
Data Link
Physical Link
Legend: control flow data flow

(Continued)

D. Local to Loop Node Test, Two Nodes, using loopback mirror and test messages.




User
Modules
Network
Management
Network
Application
Session Cont.
End-to-End Comm.
Routing
Data Link
Physical Link
Legend: control flow data flow

A. Normal Local to Local, using loopback mirror.

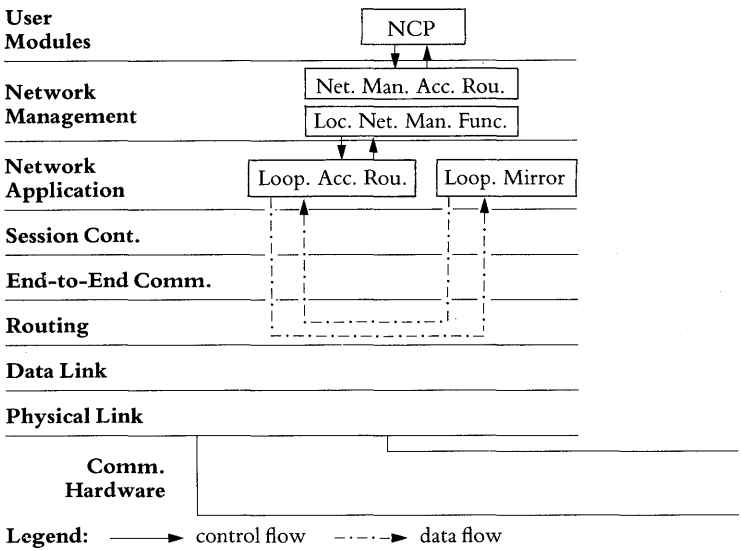
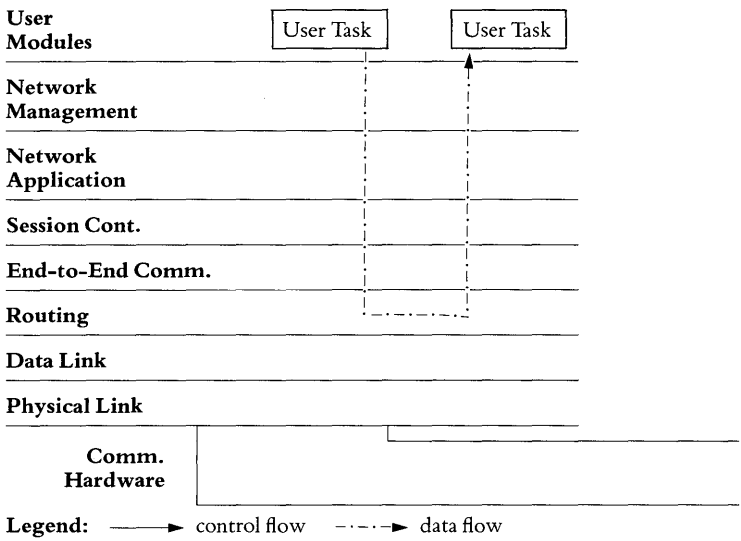


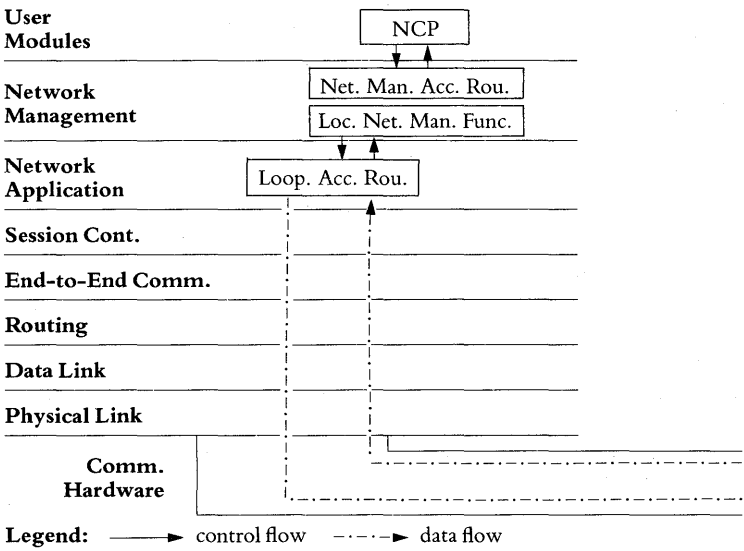
Figure 11.6
Examples of Node-Level Logical-Link Loopback Tests

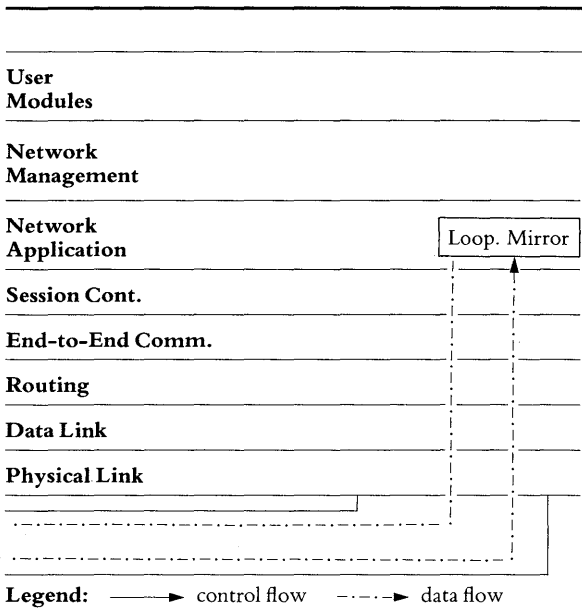
B. Normal Local to Local, using user tasks.



(Continued)

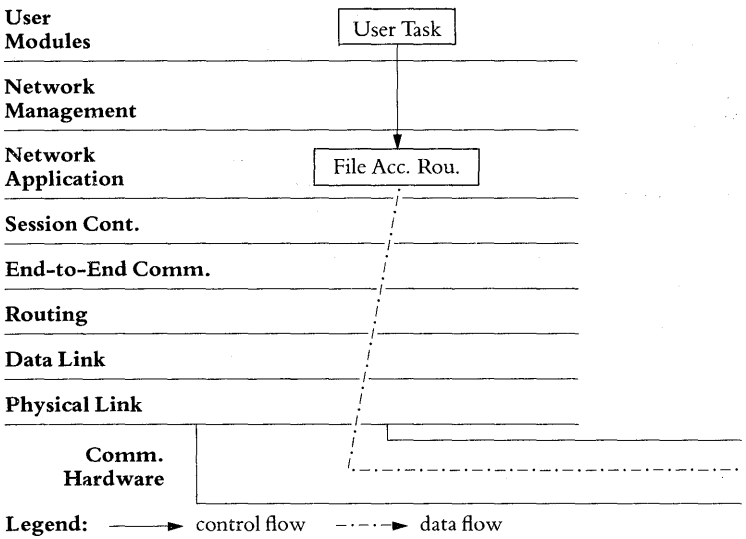
C. Normal Local to Remote, using loopback mirror.

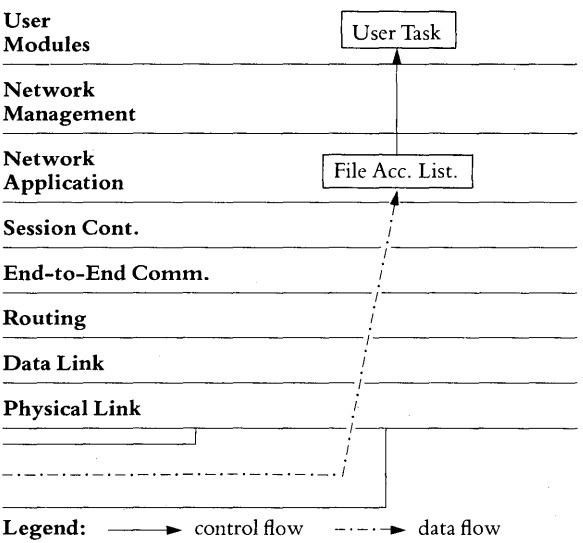



User Modules
Network Management
Network Application
Loop. Mirror
Session Cont.
End-to-End Comm.
Routing
Data Link
Physical Link
Legend: control flow data flow

(Continued)

D. Normal Local to Remote, using files as test data




User Modules
User Task
Network Management
Network Application
File Acc. List.
Session Cont.
End-to-End Comm.
Routing
Data Link
Physical Link
Legend: control flow data flow

- **Monitoring Network Operation with Observer**

Observer is a Digital-developed software product that monitors the performance of DECnet networks. Installed in a Phase III or Phase IV DECnet-RSX node, it gathers, processes, and generates information on network performance. Network events such as changes in line, link, and node states are displayed as they occur on a VT100 terminal dedicated to this function. Other information is logged to files that can be processed periodically to produce network performance reports. A command terminal that offers a menu-based approach to network surveillance enables a network manager to display a broad range of information concerning the state of designated network components. Network managers can, for example, retrieve statistical information regarding network operation and also enter commands affecting the relationship between Observer and specified network components or regions.

The information that appears on the event-display terminal can alert network managers to conditions that could cause problems in network operation. Network managers can then take steps to correct such conditions before major communications or performance problems develop.

Consider the following example. A message from Observer notifies a network manager of high data traffic over a specific link. This might mean that certain nodes are processing and transmitting unexpected data loads or that the network manager may have originally specified too low a data rate for that link. The network manager could take appropriate corrective measures like changing link and node parameters and thereby improve performance in that part of the network.

The menu-based approach of the command terminal (shown in Figures 11.7, 11.8, and 11.9) enables network managers to monitor closely specific regions and components of the network and to control Observer operation.

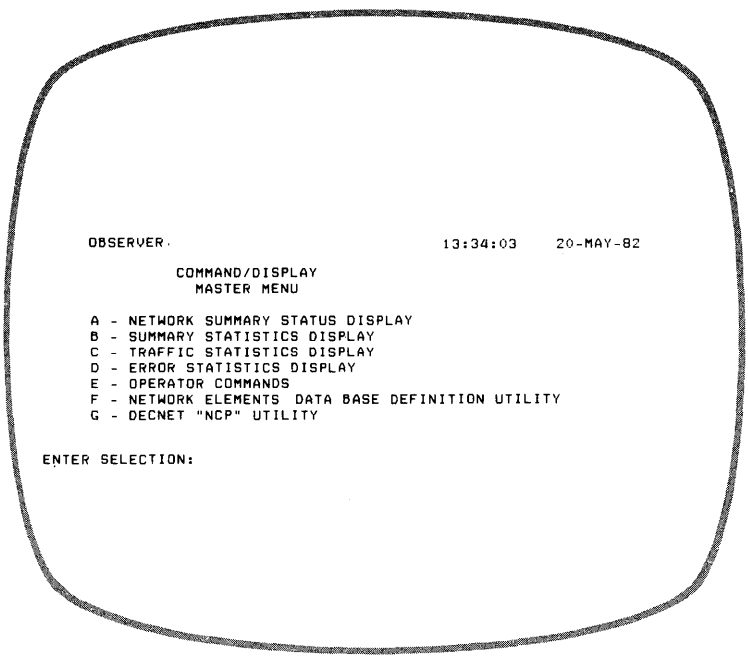


Figure 11.7
Observer Master Menu

Figures 11.7, 11.8, and 11.9 illustrate just one example of how network managers can use Observer to focus on various network areas and components. Figure 11.7 illustrates the Master Menu that appears on the command terminal. If the network manager selects "A" from this menu, the Network Summary Status Display (Figure 11.8) appears. This screen, which is updated approximately every 30 seconds, displays summary information concerning the network being monitored. In addition to present-

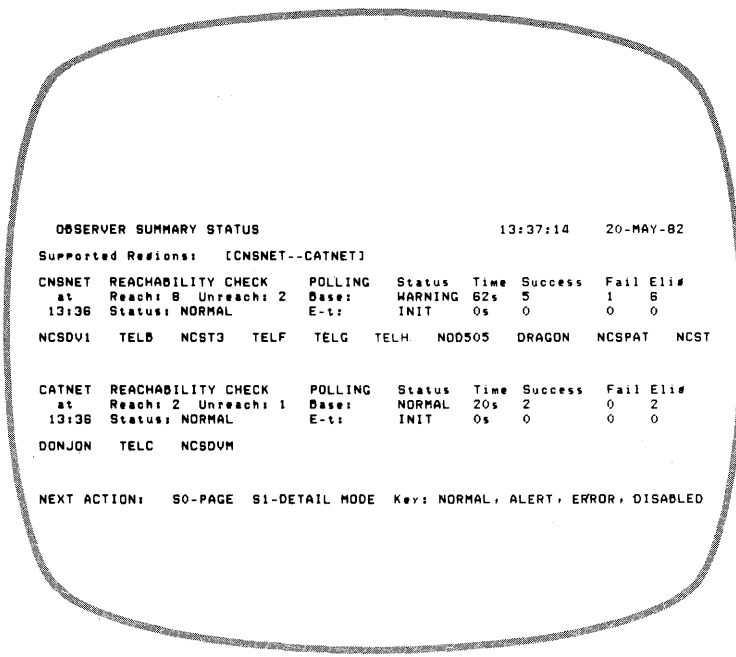


Figure 11.8
Observer Summary Status Display

ing identification information, the display specifies the number of nodes reachable and unreachable from the Observer node, the results Observer obtained, and other statistics that Observer gleaned in the most recent polling cycles. The names of the nodes in the network or network regions (which are user-defined) are also displayed. Analysis of the information logged by Observer can serve as a basis for reconfiguring the network and tuning it for optimum performance.

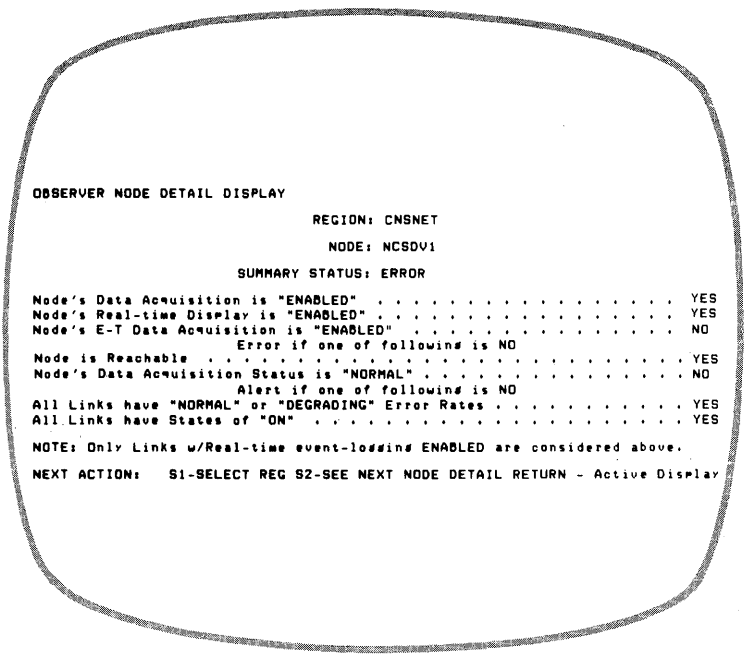


Figure 11.9
Observer Node Detail Display

To focus-in more closely on the network, the network manager can summon up a Node Detail Display (Figure 11.9) for a particular node. As the figure illustrates, this screen displays status information concerning the selected node. If network managers need further information in order to diagnose a problem, they can select another display directly or go back to the Summary Status display. The *Observer User's Guide* (available through a Digital sales representative) contains detailed information on the ease with which Observer enables users to move from one screen to another.

Element	State/Status	Possible Conditions
Link	Link Error-Rate Status	Normal/Degrading/Degraded/ Unusable
Link	Traffic-Rate Status	Normal/High/Low
Link	Realtime Event-Logging Status	Enabled/Disabled
Link	State	Initialization/Online/Unknown/ Removed
Node	Base-System and End Traffic-Enable Status	Enabled/Disabled
Node	Data Acquisition Status	Unreachable/Normal/Unknown/ Error
Node	Traffic-Rate Status	Normal/High/Low
Node	Realtime Event-Logging Status	Enabled/Disabled
Node	State	Initialization/Normal/Warning/ Fatal Error
Region	Base-System and End Traffic-Enable Status	Enabled/Disabled
Node	Reachability Check-Completion Status	Initialization/Normal/Warning/ Fatal Error
Node	Base-System and End Traffic-Polling Completion Status	Initialization/Normal/Warning/ Fatal Error

Table 11.1
Observer-maintained State and Status Information

Observer monitors nodes, links, end-to-end traffic, and network regions. It provides three kinds of information reports: state information, status information, and statistics.

Table 11.1 identifies state and status information. Table 11.2 lists the kinds of statistics that Observer gathers and maintains. See Observer documentation for details on function, installation, and operation of the product.

Type of Statistic	Element	Statistics
Time-related	Node	Time base in seconds, Time last data received, Time of last state
	Link	Time base in seconds, Time last data received, Time of last state change to on
Error-related	Node	Buffer-resource errors, Packet-format errors, Partial routing update loss, Aged packet loss, Node-unreachable packet loss, Out-of-range packet loss, Oversize packet loss, Network errors, Node errors
	Link	Data blocks sent, Data blocks received, Collision-detect-check failure (Ethernet only), Data errors outbound, Local-reply timeouts, Data errors inbound, Local-buffer errors, Unknown frame destination (Ethernet only), Congestion loss, Selection timeouts, Resets, Link error level, Link down
Traffic-related	Node	Bytes received, Bytes sent, Node traffic level (bytes), Packets received, Packets sent, Node traffic level (packets)
	Link	Link traffic level (bytes), Link traffic level (packets), Bytes sent, Data blocks sent, Data blocks received, Transit packets sent, Originating packets sent, Blocks deferred (Ethernet only), Blocks sent with collision (Ethernet only), Bytes received, Transit packets received, Terminating packets received, Multicast blocks received (Ethernet only)

Table 11.2
Statistics Maintained by Observer

Access Control: The mechanism in a node for screening inbound connect requests and verifying them against a local system account file. Access control is an optional Session Control function.

Active Side: In the context of MOP loopback tests, the node that controls a test.

Adjacency: A circuit and node pair. An Ethernet with *n* attached nodes represents *n-1* adjacencies to a router on that Ethernet. On DDCMP and X.25 circuits, adjacencies are in one-to-one correspondence with circuits, since these circuits interconnect pairs of nodes.

Adjacent Node: A node removed from another node by a single physical line.

Aged Packet: A packet that has exceeded the maximum number of visits.

ASCII: American Standard Code for Information Interchange. ASCII is a seven-bit-plus-parity code established by the American National Standards Institute to achieve compatibility among different computers and terminals.

Asynchronous Line: A physical line on which the time intervals between transmitted characters may be of varying and unequal length. On an asynchronous line, the Physical Link layer provides bit and byte framing, and the Data Link layer, message framing.

Back-off: The procedure on an Ethernet Data Link channel whereby nodes attempting to retransmit after a collision (see below, under *collision*) delay retransmissions in order to reduce the load on the channel.

Bandwidth: The range of frequencies assigned to a channel; the difference, expressed in Hertz, between the highest and lowest frequencies of a band. The higher the bandwidth, the more the data throughput.

Binary Synchronous Protocol: A data link protocol that uses a defined set of control characters and control character sequences for synchronized transmission of binary coded data between stations in a data communications system.

Carrier Sense Multiple Access With Collision Detection (CSMA/CD): A distributed channel-allocation procedure in which every station can receive the transmissions of all other stations. Each station awaits an idle channel before transmitting, and each station can detect overlapping transmissions by other stations.

CCITT: The International Telegraph and Telephone Consultative Committee, the technical committee of the International Telecommunications Union (ITU), which is responsible for the development of recommendations regarding telecommunications, including data communications.

Channel: The data path joining two or more stations, including the communications control capability of the associated stations.

Characteristics: Node parameters (values) that are generally static. These reside in a permanent database on the node. Characteristics are a Network Management information type and can be altered by a network manager. They remain unaffected by network activity. Node name is an example of a characteristic.

Circuit: A logical connection between protocol modules at the Data Link layer. There is one circuit for each DDCMP point-to-point line and one circuit for each tributary station on a DDCMP multipoint line. There is one circuit for each X.25 permanent or switched virtual circuit. There is one circuit for each protocol type on an Ethernet.

Circuit Cost: A positive integer value, assigned by a network manager, associated with using a circuit. When travelling from node to node, messages are routed along the path with the smallest total cost.

Client: A module that requests service of another module.

Client Layer: A layer, typically the *n + 1* layer, which acts as the client of the *n* layer. For example, the End-to-End Communication layer is the client of the Routing layer.

Collision: Overlapping transmissions by two or more stations on an Ethernet. The Etherent Physical Link layer detects collisions and the Ethernet Data Link layer retransmits affected data after a random interval.

Command Node: The node where an NCP (the Network Control Program, used in network management) command originates.

Congestion: The condition that arises when there are too many packets in the queue waiting to be transmitted.

Congestion Control: The Routing component that manages buffers by limiting the maximum number of packets on a queue for a physical line. This component is also called *transmit management*.

Control Station: From the perspective of the Data Link layer protocol, a station that has overall responsibility for the orderly operation of a circuit.

Controller: The hardware that controls a line. A multiple-line controller device is responsible for one or more lines. The controller identification is part of a line identification.

Cost: See *circuit cost*.

CSMA/CD: See *carrier sense multiple access with collision detection*.

Data Flow: The movement of data from a Session Control module in a source node to a Session Control module in a destination node. NSP (Network Services Protocol, operating in the End-to-End Communication layer) transforms data from

Session Control transmit buffers into a network form before transmitting it across a logical link. NSP retransforms the data at the destination from its network form to its receive-buffer form. Data flows in both directions (full-duplex) on a logical link.

Data Communications Equipment (DCE): The packet-switched data network (PSDN) equipment that provides the functions required to establish, maintain, and terminate a connection between DTEs (see below, under *data terminal equipment*) using a physical or virtual circuit.

Data Link: A logical connection between two stations on the same circuit. In the case of a multipoint link there can be multiple circuits.

Data Terminal Equipment (DTE): The equipment, typically a computer system or terminal, that consists of a data source and sink connected to common carrier communication facilities.

Data Transparency: The capability of receiving, without misinterpretation, data containing bit patterns that resemble protocol control characters.

Datagram: In the context of DECnet, a unit of data passed between the Routing Layer and the End-to-End Communication layer. When the Routing layer adds a route header to a datagram, it becomes a packet.

DCE: See above, under *data communications equipment*.

Designated Router: A routing node (see below) on an Ethernet chosen to perform additional duties, such as informing each end node on the Ethernet of the existence of other end nodes and of the identity of the Ethernet routing nodes. The router chosen to be the designated router is the one with highest priority, with highest station address-breaking ties.

Downline Load: The process of sending a copy of a system-image or other file over a line to the memory of a target node.

DTE: See above, under *data terminal equipment*.

EIA: Electronic Industries Association. A standards organization specializing in the electrical and functional characteristics of interface equipment.

End Node: A topological description of a non-routing node (one that cannot forward packets intended for other nodes). An end node supports only a single active line.

End-User Module: A module or program that runs in a user process of a network node and communicates with Session Control software to obtain logical link service.

Error control: The protocol function that ensures the reliable delivery of data. It typically consists of sequencing, acknowledgment, and retransmission mechanisms.

Ethernet: A local area network protocol using a carrier sense multiple access with collision detection (CSMA/CD) scheme to arbitrate the use of a 10 megabits-per-second baseband coaxial cable.

Events: Occurrences that the Network Management layer logs for recording.

Executor Node: The node in which the active local network management function is running (that is, the node actually executing a Network Management command).

Flow Control: The protocol function that coordinates the flow of data between two protocol modules to ensure that data is not lost, buffer deadlocks do not occur, and communication overhead is kept to a minimum.

Frame: A Data Link layer message as sent or received by an Ethernet Data Link module or an X.25 frame-level module.

Frame Level: Level 2 of the CCITT X.25 recommendation, which defines the link-access procedure for data exchange over the link between the DTE and the DCE.

Framing: The Physical or Data Link layer component that synchronizes data at the bit, byte, and message level.

Full-Duplex Channel: A channel that provides concurrent communication in both directions (to and from a station).

Full-routing Node: An implementation of the DNA Routing layer which contains the full complement of Routing components. A full-routing node performs route-through functions by forwarding packets intended for other nodes.

Gateway: A module, or set of modules, that transforms the conventions of one network into the conventions of another.

Half-duplex Channel: A channel that permits two-way communication, but only in one direction at any given time.

Hierarchical Network: A computer network in which computers perform process-control functions at several levels. The computers at each level are specially suited to perform the functions of that level. In a laboratory application, for example, computers at the lowest level collect data, which they transmit to the computers at the next highest level. Here, the collected data is reduced for analysis, and sent up to the next level of the hierarchical configuration.

Hop: To the Routing layer, the logical distance between two adjacent nodes in a network.

Host Node: A node that provides services for another node.

Interface: The relationship between different software modules that are usually in the same node.

Intra-Ethernet Bit: A special bit, set by the designated router on an Ethernet, in a packet forwarded from a source end node to a destination end node that indicates to the destination end node that the source end node is on the same Ethernet.

ISO Reference Model: The International Standards Organization Reference Model for Open System Interconnection, ISO draft proposal DP7498. A proposed international standard for network architectures that defines a seven-layer model, specifying services and protocols for each layer.

Jam: A bit sequence that an Ethernet Data Link module transmits when it experiences a collision to ensure that all affected stations detect the collision.

Leased Line: A non-switched circuit that a user (company or other institution) leases from a public utility company (common carrier) for exclusive use.

Line: A physical path that provides direct communication among a number of stations.

Link-Level Loopback: The network management process of testing a specific data link by sending a message directly from the Network Management layer of a node to its Data Link layer (bypassing the intervening layers) and over a circuit or line to a device that returns the message to the source.

Link Management: The DDCMP component that controls transmission and reception on links connected to two or more transmitters and/or receivers in a given direction. Also, the Ethernet data link component responsible for channel allocation (collision avoidance) and contention resolution (collision handling).

Logging: The network management process of recording information from an occurrence that has potential significance in the operation and/or maintenance of a network. Logging maintains a permanent record of the occurrence so that persons or programs responsible for making realtime or long-term decisions about network operation can access the record.

Logging Sink Node: A node to which logging information is directed.

Logical Channel: An association between an X.25 DTE and its DCE for a given virtual circuit.

Logical Link: A virtual circuit between two end-user processes in the same node or in separate nodes. The Session Control layer mediates between an end-user process requiring logical link service and the End-to-End Communication layer, which actually creates, maintains, and destroys logical links.

Loop Node: A special name for a node that is associated with an adjacency for loop-testing purposes. The NCP SET NODECIRCUIT command sets the name of the loopback node.

Master Station: A station that has control of a channel at a given instant for the purpose of sending data messages to a slave station.

Maximum Cost: An operator-controlled Routing layer parameter that defines the value at which the routing decision algorithm in a node declares another node unreachable because the cost of the least costly path to the other node is excessively high (more than the defined value). For correct network operation, this parameter must not be less than the maximum path cost of the network.

Maximum Hops: An operator-controlled Routing layer parameter that defines the value at which the routing decision algorithm in a node declares another node unreachable because the length of the shortest path between the two nodes is too long (more than the defined value). For correct network operation, this parameter must not be less than the network diameter.

Maximum Path Cost: The value of the path between the two nodes of the network with the greatest routing cost, where routing cost is the cost of the least costly path between a given pair of nodes. In Figure 2.11, the maximum path cost is 9.

Maximum Path Length: The routing distance between the two nodes of the network with the greatest routing distance, where routing distance is the length of the least costly path between a given pair of nodes. In Figure 2.11, the maximum path length is 9.

Maximum Visits: An operator-controlled Routing layer parameter that defines the value at which the packet-lifetime-control algorithm discards a packet which has traversed too many nodes. For correct network operation, this parameter must not be less than twice the maximum path length of the network.

Message Exchange: The DDCMP component that transfers data correctly and in sequence over a link.

Monitor: A logging sink that is to receive a machine-readable record of events for possible realtime decision-making.

Multidrop Line: See *multipoint line*.

Multiple Line Controller: A controller that can manage more than one line unit. (Digital's multiple line controllers are also called multiplexers.)

Multiplex: Simultaneously transmit or receive two or more datastreams on a single channel.

Multipoint Line: A line linking more than two stations, where one station is responsible for channel control. Also referred to as multidrop.

Network: A collection of nodes interconnected by lines.

Network Diameter: The distance between the two nodes of the network having the greatest reachability distance, where reachability distance is the length of the shortest path between a given pair of nodes. In Figure 2.11, the network diameter is 3.

Node: An implementation of a computer system that supports the Routing layer, the End-to-End Communication layer and the Session Control layer. Each node has a unique node address (see below, under *node address*).

Node Address: The unique numeric identification of a specific node.

Node-Level Loopback: The network management process of testing a logical link by using messages that flow, along with normal data traffic, through the Session Control, End-to-End Communication, and Routing layers within one node or from one node to another and back. In some cases, node-level loopback involves using a loopback-node name associated with a particular circuit.

Node Name: An operator-assigned optional alphanumeric identification associated with a node address. There is a strict one-to-one mapping between node name and node address. A single node can be known by different names, but no name maybe used more than once. The node name must contain at least one alphabetic character.

Node-Name Mapping Table: A table that defines the correspondence between node names and node addresses or adjacencies. The Session Control layer of a a node uses the table to identify destination nodes for outgoing connect requests and source nodes for incoming connect requests.

Non-Routing Node: A DECnet node that contains a subset of Routing modules and can send and receive packets but not perform route-through functions by forwarding packets intended for other nodes. It is connected to the network by a single active circuit.

Object Type: A numeric value that may be used, instead of a process name, for process or task addressing by DECnet modules. Object types are normally used to identify generic services, like file access.

OSI: See ISO reference model.

Other Data: The NSP Data Request, Interrupt Request, and Interrupt messages. These are NSP messages other than the Data Segment message. All Other Data messages move in one data subchannel.

Packet: A unit of data to be routed from a source node to a destination node. When stripped of its route header and passed to the End-to-End Communication Layer, a packet becomes a datagram.

Packet Level: Level 3 of the CCITT X.25 recommendation, which defines the packet format and control procedures for the exchange of packets.

Packet Lifetime Control: The Routing layer component that monitors adjacencies to detect if they have gone down, and prevents excessive looping of packets by discarding those that have exceeded the maximum-visit limit.

Packet Switching: See *route through*.

Parallel Data Transmission: A data communication technique in which more than one code element (a bit, for example) of each byte is sent or received simultaneously.

Parameters: DNA values to which the Network Management layer has access for controlling and monitoring purposes.

Passive Side: In the context of MOP loopback tests, the node that loops back the test messages.

Path: A possible route for a packet from source node to destination node. The path can comprise a sequence of connected nodes between the source and destination nodes.

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.

Peer Protocol: A protocol for communication between modules in the same DNA layer.

Permanent Virtual Circuit (PVC): A virtual circuit between two X.25 DTEs that is always established. A logical channel is permanently allocated at each DTE/DCE interface to a PVC.

Physical Link: A hardware-addressable communication path.

Piggybacking: The mechanism by which the Data Link layer of a node of sends an acknowledgment (of data received) to another node within a returned data or control message.

Pipelining: The mechanism by which the Data Link layer of a transmitting node sends messages without waiting for individual acknowledgment of each successive message.

Point-To-Point Circuit: A link connecting only two stations.

Protocol: A set of messages with specific formats and rules for exchanging the messages.

Raw Event: A logging event as recorded by the source process, incomplete in terms of total information required

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

Reassembly: The mechanism by which NSP places multiple received data segments into a single Session Control receive buffer.

Remote Node: To a node, any other network node.

Request Count: This term has two definitions in the context of NSP functions: (1) Variables that NSP uses to determine when to send data and (2) values sent in link-service (Data Request and Interrupt Request) messages.
To determine when to send data, the flow-control mechanism adds the request counts received in Data Request and Interrupt Request messages to the request counts it maintains.

Retransmission: The process of resending messages that have not been acknowledged within a certain period of time. This is part of a protocol's error-control mechanism.

RMS: Record Management Services. This file system is used on all major systems from Digital, except where space is limited (for example, RT-11). In addition to access modes provided by previous file systems, RMS provides random access for direct and indexed files, and offers ISAM.

Router: See *full-routing node*.

Route Through: The process by which one or more intermediary node in the path between a source node and a destination node directs packets from the source node to the destination node. Routing nodes permit route-through. Also called packet switching.

Routing: Directing data message packets from source nodes to destination nodes.

Routing Node: See *full-routing node*.

Segment: The data carried in a Data Segment message. NSP divides the data from Session Control transmit buffers into numbered segments for transmission over logical links.

Segmentation: The mechanism by which NSP divides normal data in Session Control transmit buffers into numbered segments for transmission over logical links.

Server: A module or set of modules in a layer that performs a well-defined service, like remote file access or gateway communication, on behalf of another module.

Server System: A node that contains one or more server. A server system is often dedicated to server functions.

Sink Node: A node that receives and records Network Management events.

Slave Station: A tributary station that can send data only when a master control station polls it or requests it to transmit.

Star Topology: A network configuration in which one central node is connected to more than one adjacent end node. A star can be a subset of a larger network.

Station: With regard to the Data Link layer protocol, a termination on a data link. A station is a combination of the physical link (communication hardware) and the data link protocol implementation.

Station Address: An address assigned at the data link level to a station.

Status: Dynamic information, such as line state, relating to network operation. Status is a Network Management information type.

Subchannel: A logical communication path within a logical link that carries a defined category of NSP data messages. Because Data Segment messages are handled differently from Other Data messages, these two types of messages can be thought of as travelling in two different subchannels.

Switched Virtual Circuit (SVC): A temporary association between two X.25 DTEs.

Synchronous Line: A physical line on which data characters and bits are transmitted at a fixed rate, with transmitter and receiver synchronized. On a synchronous line, the Physical Link layer provides bit framing, and the Data Link layer provides message framing.

Target Node: The node that receives a memory image during a downline load, generates an upline dump, or loops back a test message.

Topology: The physical arrangement and relationship of interconnected nodes and lines in a network. A legal topology satisfies the requirements of the Routing layer specification.

Transparent Data: Binary data transmitted with the recognition of most control characters suppressed. DDCMP provides data transparency because it can receive, without misinterpretation, data containing bit patterns that resemble DDCMP control characters.

Tributary Station: A station on a multipoint line that is not a control station.

Unit: The hardware controlling one circuit on a multiple line controller. A unit, a controller, and associated Data Link modules form a station.

Unreachable Node: A node to which the cost of the least costly path exceeds the maximum cost of the network, or the length of the least costly path exceeds the maximum hops of the network.

Upline Dump: The process by which a target node sends a copy of its memory image up a line to a file at a host node.

User: A person or module that requests service from a DNA layer. *Client* is the preferred term when referring to modules.

Virtual Call: See *switched virtual circuit*.

Virtual Circuit: A temporary connection between a data source and a sink in a network. Virtual circuits typically guarantee delivery and sequentiality of client data.

Wildcard: With regard to DAP, an asterisk (*) that replaces an element in a file specification. The asterisk specifies all known items or values in the range that its position indicates. For example, FILE.*;* specifies all known types and versions of all files named FILE.

Window: A range of packets authorized for transmission across an X.25 DTE/DCE interface.

A

aborting links, 121
access
  on DECnet/SNA Gateway, 170-73
  remote, to files and records, 122-35
access complete messages, 124
access control
  in DECnet, 109-11
  in remote file access, 134
  in task-to-task communications, 115
access-control information, 131-32
access methods, 132
  in X.25 Gateway Access protocol, 165-66
access request messages, 124
acknowledgements, l06
Acknowledge Message (ACK), 32
adaptive routing, 17
addresses
  node, 21-22, 209
  node, mapping of names to, 99
  tributary, 215
addressing
  in Ethernet, 41-43
  in task-to-task communications, 113-14
adjacencies, 59
algorithms, routing, 52-53
applications
  DECnet file transfer capability for, 185-89
  DECnet remote file access capability for, 189-91
  DECnet task-to-task capability for, 181-83
  DECnet terminal facilities for, 191-96
  network interaction with, 183-85
  supported by DECnet, 179-81
  see also programs; software
architecture, network, 13-14
area-based networks
  Level 1 and 2 routers in, 53
  node addresses in, 21
  parameters for, 211
  routing in, 47-48
area routing, 17
areas (groups of nodes), 3
  routing between, 17
ASCII data, 133
asynchronous terminals, 168-69

B

  bandwidth, in Phase IV Ethernet networks, 76
baseband local area networks, 76
  in Ethernet, 17, 18
binding, 143, 146
bisync (binary synchronous) protocol, 28

bit-oriented protocols, 28
blocking and deblocking Routing layer messages, 64
buffers
  for flow control, 108
  management of, 54
  parameters for, 212
  Session Control, 106
byte-oriented protocols, 28
byte synchronization, 35

C

calls
  DECnet task-to-task, 112-13
  to open files, 130-31
  for remote file access, 122, 126
Carrier Sense Multiple Access with Collision Detect (CSMA/CD) protocol, 27
  in Ethernet, 39
CCITT (Comite Consultatif International Telephonique et Telegraphique), 18, 151
  X.3, X.28, and X.29 recommendations by, 168-69
channels
  collision handling on, in Ethernet, 46-47
  logical, 153
  in Phase IV Ethernet networks, 76
  subchannels of, 105
characteristics (node), 209
character-oriented protocols, 28
checkpointing, 224-25
checksums of X.25 packets, 64
circuits, 24-26
  adapting to different kinds of, 54
  controlling state of, 217-18
  costs (measurement) of, 49-52
  DECnet, characteristics of, 88-89
  DECnet/SNA, 177
  parameters for, 212-16
  in X.25 protocol, 153-55
circuit timers, 148
client layers (Ethernet), 44
coaxial cable, 78
collisions, Ethernet handling of, 46-47
collision windows, 46
Comite Consultatif International Telephonique et Telegraphique (CCITT), 18, 151
  X.3, X.28, and X.29 recommendations by, 168-69
command files
  remote submission of, 135
  TLK, 138
command nodes, 221
commands
  in Phone utility, 136-37
  in TLK utility, 137-38
  Virtual Network Processor (VNP), 221

command terminal module, 140
command terminal protocol, 143-44
command terminals, 250
communications
  Digital Network Architecture (DNA) for, 5-6
  inter-area, 48-49
  logical links for, 92
  network architecture for, 13-14
  between networks, DECnet/SNA for, 169-78
  between networks, X.3, X.28, and X.29 protocols for, 168-69
  between networks, X.25 Gateway Access protocol for, 155-67
  between networks, X.25 protocol for, 151-55
  task-to-task, 112-21
  terminal-to-terminal, 136-38
  transmission modes for, 26-27
  see also networks
communications servers, 18, 82, 86
  loopback testing of, 231
  in Phase IV, 20
configuration databases, 206, 209
configurations
  of DECnet, 71-72
  defining, network management and, 209-16
  general purpose and dedicated nodes in, 84-88
  line and circuit characteristics of, 88-89
  local area networks, 74-81
  multipoint, 19
  Phase III and Phase IV nodes in, 82
  topology alternatives in, 16-19
  wide area networks, 72-74
congestion control, 61-62
connect blocks, 113, 114
contention, Ethernet handling of, 44-47
control messages
  in Digital Data Communications Message Protocol (DDCMP), 31, 32
  routing, 56, 58
control stations, 89, 213-15
copying files to remote nodes, 122, 133, 186-87
costs (of path lengths), 49-52
  parameters for, 211
counters, 219-20
CTERM protocol, 138, 149
cyclic redundancy checks (CRCs), 68

D

data
  segmentation and reassembly of, in DECnet, 105-6
  in task-to-task communications, transmitting and receiving, 119-21
Data Access Protocol (DAP) Interface, 123-26
  handshaking by, 131
databases
  configuration, 206, 209
  downline-loading, 221
  routing, 52-53
  in Routing Initialization sublayer, 63
data circuit terminating equipment (DCEs), 153
data encapsulation/decapsulation, 40, 41
data flow, 64-66
  across network to destination node, 68-69
  at destination node, 69
  at source node, 66-68
datagrams, 68
Data Link layer (DNA), 9
  data flow through, 68
  in Ethernet, 39, 44, 45
  frame and packet levels (X.25) in, 152
  protocols supported by, 13, 27-28
data link mapping interface (DLM), 167
data link mapping mode (DLM), 155, 166
Data Link module, 68
data messages, in Digital Data Communications Message Protocol (DDCMP), 29, 31
data packets, 56, 162
data request messages, 105
Data Segment messages, 106
data terminal equipment (DTEs), 153-55
in DECSYSTEM-20s, 167
start-stop mode, X.3, X.28, and X.29 recommendations for, 168-69
  X.25 Gateway Access protocol used by, 155-65
data types, 133
  within logical links, 105
DCEs (data circuit terminating equipment), 153
dead tributaries, 215
decentralization, in DECnet, 4
decision process, in routing, 61
DECNA adapters, 76-77
DECnet, 1-4
  access control in, 109-11
  applications supported on, 179-81
  capabilities of, 14-16
  configurations of, 71-72
  data flow within, 64-70

DDCMP supported by, 28-37
environments for, 72
error control in, 106-8
Ethernet supported by, 37-49
file transfer capability of, 185-89
flow control in, 108-9
general purpose and dedicated nodes in, 84-88
internetwork communications on, 151
line and circuit characteristics of, 88-89
lines and circuits in, 24-26
local area networks, 74-81
logical links in, 92-105
monitoring performance of, 228-55
network and application interaction in, 183-85
nodes in, 21-23
Phase III, 19-20
Phase III and Phase IV nodes in, 82
Phase IV, 20
Phone utility used with, 136-37
protocols supported by, 27-28
relocation planning and, 196-97
remote file access in, 122, 189-91
routing by, 47-64, 82-83
segmentation and reassembly of data in, 105-6
task-to-task communications in, 112-13, 181-83
terminal facilities of, 191-96
topology alternatives in, 16-19
transmission modes for, 26-27
user transparency in, 111
uses of, 4-5
virtual terminals on, 138
wide area networks, 72-74
X.25 access methods for, 165-66
DECnet-20, 20, 220
DECnet Router servers, 86
DECnet Router/X.25 Gateway, 5, 87, 167
trace capability for, 233-37
DECnet-RSX, 20
PSI software on, 167
TLK utility on, 137-38
DECnet-RSX-11M, 221
DECnet-RSX-11M-PLUS, 221
DECnet/SNA Gateway, 5, 87-88, 169-78
loading of, 86
SNATRACE for, 235-37
DECnet/SNA Gateway Access module, 173
DECnet/SNA Protocol Emulation, 175
DECnet-VAX, 20
downline loading of, 221
PSI software on, 167
Record Management Services in, 126

DECsystem-10s, 3
DECSYSTEM-20s, 3, 167
dedicated nodes, 84-88
DELNI, 77-78
DEQNA, 76-77
DEUNA, 14, 76-77
Digital Data Communications Message Protocol (DDCMP), 19, 27-30
  DECnet/SNA communications using, 177
  initialization and monitoring functions in, 62
  messages in, 31-34
  operation of, 34-37
Digital Network Architecture (DNA), 5-6
  architecture of, 13-14
  Digital Data Communications Message Protocol (DDCMP) designed for, 28
  interfaces in, 6-10
  protocols in, 10-11
directory files, 133
disconnects, 121
DLM (data link mapping) interface, 167
DN20 front-end processors, 166, 167
DNA dump/load protocol, 24
DNA remote console protocol, 24
DNA Routing Functional Specification, 50
DNA routing protocol, 24
downline loading, 16, 220-23
  checkpointing and, 224-25
DTEs (data terminal equipment), 153-55
  in DECSYSTEM-20s, 167
  start-stop mode, X.3, X.28, and X.29 recommendations for, 168-69
  X.25 Gateway Access protocol used by, 155-65
dumping, 237

E

electronic mail, 194
encapsulation/decapsulation, 40, 41
end nodes, 82, 83
End-to-End Communication layer (DNA), 9, 49
  accepting and rejecting logical link requests in, 117-19
  data flow through, 66-69
  logical links created by, 95-99
  object types and names declared in, 116
Session Control layer connections with, 101-3
End-to-End Communication modules
  DECnet calls sent by, 112
environments
  application, DECnet capabilities in, 181-97
  DECnet, 72
  network, 71

error detection
in Digital Data Communications Message Protocol (DDCMP), 36
in Ethernet, 43
error handling
by Data Link modules, 68
by DECnet, 106-8
Ethernet, 37-40
communications servers supported on, 18
configurations of, 79
DECnet/SNA communications using, 177, 178
initialization and monitoring functions in, 62
local area networks using, 76
Local Area Transport protocol on, 148
messages in, 41-43
operation of, 43-47
Phase III and Phase IV nodes connected to, 82
protocols supported by, 24-26
Ethernet coaxial cable, 77
Ethernet End node Data Packet format, 56
Ethernet End node Hello message, 58-59, 62
Ethernet local area networks, 17
Local Area Transport (LAT) protocol for, 136
network virtual terminals in, 140
routing on, 55-56
Ethernet protocol, 27
Ethernet Router Hello message, 58, 62
Event Logger, 199, 201, 204-5
executor nodes, 218-19, 221
downline loading by, 222
upline dumping from, 237
extended handshakes, 130-31

F

File Access Listener (FAL), 123, 126, 131, 134
File attributes messages, 124
file characteristics, 132-33
file organization, 132
file protection, 134
files
DECnet transfer capabilities for, 185-89
in networks, 4
remote access to, 14-15, 122-35, 189-91
file specifications, 131, 135
file systems, 128-30
fixed-length record format, 133
flow control
in DECnet, 108-9
in X.25 Gateway Access protocol, 159-62
forwarding process, in routing, 61
Foundation layer (NVT), 141
fragmenting and reassembling Routing layer Messages, 64

Frame level (X.25), 152
frames, in X.25 Gateway Access protocol, 164-65
framing, 34-35, 40, 41
full-duplex mode, 26, 105
full-function nodes, 83

G

Gateways
  DECnet Router/X.25, 87
  DECnet/SNA, 87-88, 169-78
  in Phase IV, 20
  PSI software for, 166-67
  SNATRACE for, 235-37
  supported by DECnet, 5
  X.25 Gateway Access protocol, 155-65
general purpose nodes, 84-88
  loopback testing of, 231

H

H4000 transceivers, 77, 79
half-duplex mode, 26, 35
handshake procedures, 94, 130-31
hardware
  adapters, 76-77
  loopback testing of, 229-37
HDLC (high-level data link control), 28
headers, in packets, 152
Hello and Test message, 58, 59, 62
hops, 49
  parameters for, 211
host nodes
  communications servers downline-loaded from, 86
  on Ethernet, 80-81
  in Local Area Transport protocol, 148
  in network virtual terminal function, 141

I

IBM systems, communications with, DECnet/SNA Gateway for, 5, 87-88, 169-78
identification
  of lines, 212
  node names, 22-23, 99, 209
  of target nodes, 114
  of users, 131, 132, 134
image data, 133
indexed file organization, 132
Information Frames (I Frames; X.25), 164
initialization and circuit monitor, 62
Initialization messages, 58, 59, 62
Intel Corporation, 39
interfaces
  Data Access Protocol (DAP), 123-26
  in Digital Network Architecture (DNA), 6-10

International Standards Organization (ISO), 5, 28
International Telephone and Telegraph Consultative Committee (CCITT), 18, 151, 168-69
interrupt data, 119, 121
interrupt messages, 105, 106
  in X.25 Gateway Access protocol, 162
interrupt request messages, 105

J

jams, 46

K

Level 1 routers, 17, 48-49, 52, 53
  as full function nodes, 83
Level 2 routers, 17, 48-49, 52, 53
  as full-function nodes, 83
levels, in X.25, 152
lines, 24-26
  controlling state of, 217-18
  cost parameters for, 211
  DECnet, characteristics of, 88-89
  identification of, 212
  in wide area networks, 72-74
link access procedures (LAP and LAPB), 163
link identifiers, 114
link management, 35, 40
links, 183-85
Link Service Functions, 201
Link Watcher, 200
local area networks (LANs), 3, 74-81
  Ethernet, 37-40
  Ethernet, routing on, 55-56
  network virtual terminals in, 140
Local Area Transport (LAT) protocol, 86, 136, 148-49
Local Network Management Functions, 200
local repeaters, 78-79
logging on to remote nodes, 139
logical channels, 153-55
logical links
  access control for, 111
  creation of, 94-95
  data types within, 105
  in DECnet, 92-95
  identified by programs, 103-4
  loopback testing of, 233
  multiple, within programs, 104-5
  Network Services Protocol and, 95-99
  for remote file access, 130
  Session Control layer and, 100
  for task-to-task communications, 113-15, 117-19
  terminating, 121

Loopback Access Routines, 201
Loopback Mirror, 201, 205, 231
loopback protocol, 24
loopback testing, 229-37

M

MAIL command, 194
Maintenance Functions, 200
maintenance messages, in Digital Data Communications Message Protocol (DDCMP), 31, 34
maintenance mode, 37
Maintenance Operation Protocol (MOP), 34, 205
  messages for, 224-27
  to monitor node operation, 218-19
management, *see* network management
mapping
  data link mapping (DLM), 166
  of node names to node addresses, 99
maximum path cost parameter, 211
memory, downline loading of, 224
message exchange, 35-37
messages
  in command terminal protocol, 144-45
  Data Access Protocol (DAP), 123-26
  data flow of, 66-69
  DECnet/SNA Gateway Access, 172-74
  in Digital Data Communications Message Protocol (DDCMP), 31-34
  in Ethernet, 41-43
  Ethernet, routing of, 55-56
  Event, 204-5
  MAIL command for, 194
  Maintenance Operation Protocol (MOP), 224-27
  network management, 201-5
  Network Services Protocol (NSP), 96-98
  polling, 89
  routing, 56-59
  Session Control, 103
  in X.25 Gateway Access protocol, 155-57
MicroVAX I systems, 3
MicroVMS, 3
mode management, 143
modems, 74
modes, in network virtual terminal service, 143
monitoring
  DECnet performance, 228-37
  network operation with Observer (software), 250-55
  node activity, 218-20

multipoint configurations, 19
multipoint lines and circuits, 88-89
  parameters for, 213-16

N

names
  node, 22-23, 209
  node, mapping of addresses to, 99
  of objects (network programs), 116-17
NCP (Network Control Program), 199-200
  loopback testing in, 230-35
negative acknowledgements, 106
Negative Acknowledge Message (NAK), 32
NET utility, 138
Network Application layer (DNA), 7
  Data Access Protocol (DAP) in, 123
  DECnet/SNA Gateway Access functions in, 170
  network virtual terminal services in, 139-40
  X.25 Gateway Access protocol in, 155
Network Control Program (NCP), 199-200
  loopback testing in, 230-35
Network File Access Routines (NFARs), 128
Network File Transfer (NFT) utility, 126, 133
Network Generation Planning Aid, 208
Network Information and Control Exchange (NICE), 200
  messages handled by, 201
network interfaces, 153
network management, 15-16, 198-99
  defining configuration and parameters, 209-18
  downline loading and, 220-25
  generating network software, 208
  messages for, 201-5
  monitoring node activity, 218-20
  node generation planning in, 205-8
  in Phase IV, 20
  utilities for, 199-201
Network Management Access Routines, 200
Network Management layer (DNA), 7
  node activity monitored by, 220
  Routing initialization sublayer database maintained by, 63
  utilities in, 199-201

Network Management Listener, 200
network management modules, 128
network managers, 2, 198
  diagnostics and maintenance by, 228
  Observer (software) used by, 250-53
  protocols implemented by, 13
  routing parameters defined by, 50
network object parameters, 210
networks
  architecture of, 13-14
  data flow within, 64-69
  DECnet for, 3-5
  DECnet/SNA Gateway for communications between, 170-78
  definition of, 1
  environments for, 71
  generating software for, 208
  interaction between applications and, 183-85
  interactive terminal-to-terminal communications on, 136-38
  local area, 74-81
  Local Area Transport (LAT) protocol on, 148-49
  monitoring and testing performance of, 228-37
  monitoring and testing performance of, with Observer (software), 250-55
  nodes in, 21-23
  planning node generation for, 205-8
  remote file access in, 122-23
  topology alternatives in, 16-19
  virtual terminals on, 138-48
  wide area, 72-74
  X.3, X.28, and X.29 protocols for communications between, 168-70
  X.25 Gateway Access protocol for communications between, 155-67
  X.25 protocol for communications between, 151-55
  see also communications
Network Services Protocol (NSP), 95-99
  flow control in, 108-9
  segmentation and reassembly of data by, 106
network specifications, 113, 115
network terminal communications, 15
network virtual terminals (NVTs), 15, 138-47, 189-90
nodes, 1, 21-23
  addresses and names for, 41-43, 99, 209
  characteristics of, 71
  collision handling by, in Ethernet, 46-47
  data flow between, 64-69

DECnet, 3, 81-88
downline loading of, 220-21
lines and circuits connecting, 24-26
loopback testing of, 229-37
monitoring activity of, 218-20
in multipoint configurations, 19, 89
network virtual terminals as, 138-39
operation of, network management and, 216-18
in packet-switched data networks (PSDNs), 153
passwords for, 209-10
in Phase IV Ethernet networks, 76
planning generation of, 205-8
protocols for communications between, 10-11
remote file access by, 122-23
routing between, 47-49
RSX-11S, downline-loading and checkpointing tasks on, 224-25
server and host, 79-81
X.25 access methods for, 165-66
normal data, 105, 119-20

O

objects (network programs), 114-17
Observer (software), 228-29, 250-54
open systems interconnection (OSI), 5
operating systems, 3
DECnet Phases implemented on, 82
file systems dependent on, 130
logical connections between nodes under, 138
Phase IV supported by, 20

P

packet assembly/disassembly facility (PAD), 168-69
Packet level (X.25), 152
packet lifetime control, 62
packet-mode DTEs, 153
Packetnet System Interface (PSI) software, 166-67
PAD emulation in, 169
trace capability in, 233
packet route headers, 56-57
packets, 21-22
forwarding of, 53-54
routing of, 52
types of, 160-61
packet-switched data networks (PSDNs)
physical lines and virtual circuits in, 24
routing in, 56
start-stop mode communications with, X.3, X.28, and X.29 recommendations for, 168-69

X.25 access methods for, 165-66
X.25 protocol for, 18, 151-53
parameters
circuit, 212-16
defining, network management and, 209
downline-loading database, 221
network object, 210
routing, 50, 211-12
passwords, 109, 131, 134
node verification, 209-10
path length, 49
paths, parameters for, 211
PDP-11 systems, 3
peer relationships, 3
performance of DECnet, monitoring and testing, 228-37
with Observer (software), 250-55
peripherals, in networks, 4
permanent virtual circuits (PVCs), 154
Phase III, 19-20
counters used in, 219
node characteristics in, 82
nodes logically connected by, 138
Observer (software) on, 250
Phase IV, 20
communications servers implemented on, 86
counters used in, 219
Digital Data Communications Message Protocol supported by, 19
Ethernet in, 17, 37-39, 76
logical terminal to remote terminal connections under, 138
network virtual terminals in, 15
node characteristics in, 82
Observer (software) on, 250
X.25 protocol supported by, 18
Phase IV Data Packet format, 56
Phone (utility), 136-37
Physical layer modules, 68
Physical level (X.25), 152
Physical Link layer (DNA), 10
point-to-point lines and circuits, 88
polling, 89
polling ratios, 215-16
portals, 143, 146
positive-acknowledgement-with-retransmission protocols, 35-36
P/OS operating system, 3
Postal Telephone and Telegraph Authorities (PTTs), 151
primary loaders, 222
Primary Logical Unit (PLU), 173
process-to-process communications Session Control layer for, 99-101

PRO/DECnet, 20
DELNI, 78
in Ethernet LANs, 81
Professional 350 personal computers, 3, 20
programming
network terminal facilities used for, 139
of remote file access, 126-28
programs
DECnet calls in, 112-13
file transfers by, 188-89
handshake procedures between, 94
links terminated by, 121
logical links between, 92
logical links identified by, 103-4
multiple logical links within, 104-5
object types and names for, 116-17
remote file access by, 122-23
see also applications; software
protocols
command terminal, 143-44
CTERM, 149
Data Access Protocol (DAP), 123-26
Digital Data Communications Message Protocol, 19
in Digital Network Architecture (DNA), 10-11
Event Logger, 204-5
Local Area Transport (LAT), 86, 136, 148-49
Loopback Mirror, 205
Maintenance Operation Protocol (MOP), 205
Network Information and Control Exchange (NICE), 200, 201
Network Services Protocol (NSP), 95-99
supported on Data Link layer, 27-28
supported on Ethernet, 24-26
terminal communication, 141-43
X.25, 18, 151-55
X.25 Gateway Access, 155-65
PSI (Packetnet System Interface) software, 166-67
PAD emulation in, 169
trace capability in, 233

Q

Q-bus structures, DEQNA for, 76

R

random access files, 132
Receive Data Decapsulation (Ethernet), 45
Receive Link Management (Ethernet), 45
receive process, in routing, 61
receiving
without contention, in Ethernet, 45-46

data, in task-to-task communications, 119-21
record attributes, 132
record formats, 133
Record Management Services (RMS), 126
records, remote access to, 14-15, 122-23
relative file organization, 132
relocation planning, 196-97
remote file and record access, 122-23
access control in, 111, 134
accessing remote files from terminals for, 133-34
in application environment, 189-91
Data Access Protocol (DAP) for, 123-26
file protection in, 134
file specifications for, 135
programming, 126-33
remote command file submission for, 135
remote repeaters, 79
remote systems, monitoring, 218-19
repeaters, 78-79
Reply to Message Number (REP), 32
reserved object types, 117
resets, in X.25 Gateway Access protocol, 162-63
routers, 48-49
routes, 47
routing, 17, 47-55
in DECnet, 82-83
in Ethernet LANs, 55-56
messages for, 56-59
operation of, 59-64
parameters for, 211-12
in Phase IV, 20
routing control messages, 58
Routing Control sublayer, 59
Routing Initialization sublayer, 59, 63
Routing layer (DNA), 9, 49
data flow through, 69
messages in, 64
segmentation and reassembly of data by, 106
routing messages, 58
Routing modules, 52-54, 68
routing timer, 211
RSTS/E operating system, 3
RSX-11M operating system, 3
RSX-11M-PLUS operating system, 3
RSX-11S operating system, 3, 224-25
RT-11 operating system, 3

S

SDLC, DECnet/SNA communications using, 177
segmentation and reassembly of data, 105-6

sequential access files, 132
sequential file organization, 132
server nodes
  on Ethernet, 79-80
  in network virtual terminal function, 141
servers, 86-87
Session Control buffers, 106
Session Control layer (DNA), 9, 99-101
  data flow through, 69
  extended by terminal communication protocol, 141
  nodes names in, 22-23
Session Control modules, 66, 69
SNA (Systems Network Architecture; IBM), 5, 88
  DECnet/SNA Gateway for, 169-78
SNATRACE, 235-37
software
  DECnet, logical links and, 95
  DECnet calls in, 112-13
  DNA interfaces, 6-10
  network, generation of, 208
  for network diagnostics and maintenance, 228-29
  PSI, 166-67
  see also applications; programs
source programs, 122-23
Start Acknowledge Message (STACK), 32
  Start Message (STRT), 32
start-stop mode DTEs, 153
  X.3, X.28, and X.29 recommendations for, 168-69
states
  of lines and circuits, 217-18
  of nodes, 216-17
status paramenters, 209
stream record format, 133
subchannels, 105
subroutines, for remote file access, 126-28
Supervisory Frames (S Frames; X.25), 164
switched virtual circuits (SVCs), 154-55, 163
synchronous disconnects, 121
system commands, 133
system images, downline loading of, 224
system managers, 2, 198
  loopback testing by, 229, 230
  node names assigned by, 22, 23
  upline dumping by, 237
Systems Network Architecture (SNA; IBM), 5, 88
  DECnet/SNA Gateway for, 169-78

T

target-initiated downline loads, 222
target nodes, 221
  downline loading of, 222, 224
  identifiers for, 114
target programs, 123
target system images, 221
task-to-task communications, 14, 112
  accepting and rejecting logical link requests in, 117-19
  access control in, 109-11
  in applications environment, 181-83
  in DECnet, 112-13
  logical links for, 113-15
  object types and names for, 116-17
  terminating links in, 121
  transmitting and receiving data in, 119-21
terminal communication module, 140
terminal communication protocol, 141-43
terminal communication services module, 146
terminal management module, 146
terminals
  access control for, 111
  accessing remote files from, 133-34
  communications between, 15, 136-38
  DECnet facilities for, in application environment, 191-96
  Local Area Transport (LAT) protocol for, 148-49
  remote file access from, 122
  virtual, 138-47
Terminal Servers, 86-87
  Local Area Transport (LAT) protocol for, 136, 149
terminating links, 121
  for network virtual terminals, 147
  in X.25 Gateway Access protocol, 163
testing DECnet performance, 228-55
throughput, in Phase IV Ethernet networks, 76
TLK (utility), 137-38
topology of networks
  adapting to changes in, 54
  DECnet, 16-19
TOPS-10 operating system, 3
TOPS-20 operating system, 3, 166-67
trace capability, 233-35
transceivers, 77
transmission
  without contention, in Ethernet, 44-45

of data, in task-to-task communications, 119-21
in Ethernet, 44-45
modes of, 216
over wide area networks, 74
transmission modes, 26-27
Transmit Link Management (Ethernet), 44, 46-47
transmit management process, in routing, 61-62
tributaries, 89, 214-16

U

UNIBUS structures, DEUNA for, 76
Unnumbered Frames (U Frames; X.25), 164
update process, in routing, 61
upline dumping, 16, 237
User layer (DNA), 7
Network Control Program (NCP) in, 199
Network File Transfer utility in, 126
users, 1-2
access control of, 109
identification of, 131, 132, 134
user transparency, 111
utilities
NET, 138
network management, 199-201
Phone, 136-37
for remote file access, 126-28
TLK, 137-38

V

variable-length record format, 133
variable-with-fixed length control record format (VFC), 133
VAX-11 PSI software, PAD emulation in, 169
VAX-11 X.25/X.29 Extension Package (XEP), 167, 169
VAXELN, 3
VAX systems, 3
VAX/VMS Command Lanaguage Interpreter, 128
VAX/VMS operating system, 3, 136-37
verification, node passwords for, 209-10
Verification messages, 58, 59, 62
virtual circuits, 24
in X.25 Gateway Access protocol, 162
in X.25 protocol, 153-55
Virtual Network Processor (VNP) commands, 221
virtual terminals (NVTs), 15, 138-47, 189-90

W

wide area networks, 3, 16-17, 72-74
network virtual terminals in, 140
windows (packets), 159-62

X

X.3 recommendation (CCITT), 168-69
X.25 Frame-Level module, 163-64
X.25 Gateway Access module, 157
X.25 Gateway Access protocol, 155-65
X.25 Gateway Server module, 157-59
X.25 Packet-Level module, 159
X.25 protocol, 5, 18, 27, 151-55
  checksums in, 64
  initialization and monitoring functions in, 62-64
  PSI software and, 166-67
  routing in, 56
  X.3, X.28, and X.29 recommendations and, 169
X.28 recommendation (CCITT), 168-69
X.29 recommendation (CCITT), 168-69
X.29 Virtual Terminal Driver, 168
XEP (VAX-11 X.25/X.29 Extension Package), 167
  PAD emulation in, 169
Xerox Corporation, 39

Notes

Notes

digital

EB 26013-42

PRINTED IN USA EB-26013-42/84 11 23 125.0 COPYRIGHT © 1984 DIGITAL EQUIPMENT CORPORATION ALL RIGHTS RESERVED