OpenVMS Technical Journal V5

DECnet-Plus Technical Overview

DECnet-Plus Technical Overview

Author: Colin Butcher, Technical Director, XDelta Limited

Overview

This article discusses DECnet-Plus (formerly known as DECnet/OSI), the current implementation of DIGITAL Network Architecture (DNA) Phase V for OpenVMS.

Today’s DECnet networks support remote system communication, resource sharing, and distributed processing. Network users can access resources on any system in the network as well as the resources of other vendors' systems on multivendor networks.

All systems connected to a DECnet network are peers or equals. Systems can communicate with each other without having to traverse a central or master system. Any system can communicate with any other system in the network via specialized devices such as routers, not just to those systems to which it is directly connected.

This article includes:

- A historical perspective of the development of DECnet, from the initial rudimentary networking protocol for small numbers of similar computers to the current DECnet-Plus that embodies the Open Systems Interconnection (OSI) standards and protocols, enabling support for an unlimited number of heterogeneous computers in a multivendor, multiprotocol network
- Descriptions of the main concepts and components of DECnet-Plus, including the OSI layered architecture, Phase V command line interface, node name and address resolution, node addresses and address towers, routing, time synchronization, complementary protocols (such as MOP), and DECnet over TCP/IP
- Brief guidelines and recommendations for choosing implementation and installation options, including advantages and disadvantages where applicable

The main purpose of this article is to provide users and system managers a greater understanding and appreciation of the behavior and capabilities of DECnet-Plus. The author has extensive consulting experience working with DECnet on OpenVMS systems and has written this article in response to questions that arose from users, programmers, and system administrators.

What Is DECnet?

DECnet is the underlying set of rules and software components that enable a wide variety of computer systems to exchange data safely and reliably. The rules describing and enforcing the behavior of the DECnet protocol are carefully constructed and documented in the DIGITAL Network Architecture (DNA) specifications. Each system (or **node**) participating in the DECnet network must rigorously adhere to the rules to ensure consistent and reliable communication with other nodes in the network.

DECnet-Plus is the most recent implementation of DNA Phase V. DNA Phase V incorporates the Open Systems Interconnection (OSI) communications specifications as defined by the International Organization for Standardization (ISO) and maps onto the OSI seven layer reference model.

DNA Phase V also specifies the mechanisms by which a DECnet-Plus (or earlier DECnet/OSI) implementation can use the TCP/IP protocol stacks as a carrier (implemented as HP TCP/IP Services for OpenVMS). This allows existing DECnet applications to operate unchanged in an IP only infrastructure by preserving the end-to-end application programming interfaces (APIs). This functionality is commonly referred to as **DECnet over IP**.

In any network, the protocols for transferring data between systems need to control several major levels:

- The physical level, such as hardware interfaces and cabling
- The interchange level, such as data flow control, integrity checking, and retransmission
- The routing level, such as node addressing and optimal path determination

- The user level, such as the command line interface and application programming interface

Basic data exchange mechanisms such as the Kermit file transfer program implement a simple point-to-point connection. By contrast, a complex heterogeneous data network protocol such as DNA Phase V (where many computers can simultaneously exchange data with many others) requires a far more rigorous approach to design and implementation.

The rules enforced by the Application Programming Interfaces (APIs) in a DECnet network serve to isolate the user of a service from the lower-level details of the network (the physical and interchange levels). If the rules that govern the external (outside the system) application-level programming interface stay consistent, and if all systems run compatible versions of the DECnet protocol, then these systems can exchange data while running any version of the operating system on any hardware platform.

This independence of layers facilitates modifying the network — for example, replacing an OpenVMS VAX V5.5-2 system running DECnet Phase IV with an OpenVMS Alpha V7.3-1 system running DECnet-Plus (in Phase IV compatible addressing mode, which is discussed later in this article). No changes to the application code are needed. The other nodes in the network do not know and do not need to know that the replaced node is now a physically different system. The application software only “sees” the corresponding (unchanged) application on each node through the DECnet end-to-end connection.

A consistent set of APIs allows the design and implementation of “network aware” applications that can be distributed over as many systems as necessary to provide the necessary scalability. One essential concept is that of a network connection to the same node as the originator. This allows network aware applications to be built and tested on physically small networks, then delivered onto larger distributed networks. Realistic testing is essential to ensure that problems of scale and performance do not arise in production use.

What is **DECnet Phase IV**? As described in more detail in the next section, DECnet Phase IV (also known as DNA Phase IV) is one of five major phases in the development of the DECnet protocol. DECnet Phase IV is the DECnet that most people are familiar with. It introduced support for a large number of nodes (up to 64,449 compared to Phase III’s maximum of 255) and added support for area routing. DNA Phase IV was first implemented as DECnet Phase IV, then became DECnet Phase IV-Plus when two significant features were introduced: end node failover and out-of-order packet caching.

> **End node failover** allows a Phase IV End Node to be connected to two entirely separate circuits, but using only one circuit at any one time with automatic and transparent failover from one circuit to the other. The “standby” circuit is entirely operational, actively listening to routing updates and maintaining the node reachability data. However, it is not used for transmitting application data until the “primary” circuit has failed.
>
> **Out-of-order packet caching** allows a Phase IV Routing Node to load balance over multiple equal-cost circuits. Prior to Phase IV-Plus, load balancing over multiple available paths to a destination node was not possible. The out-of-order packet cache feature solved the underlying issue: with multiple available paths between nodes, packet arrival at the destination could not be guaranteed in the same order as the packets were originally transmitted. In contrast, a single path between nodes implicitly guarantees that packets arrive in the same order as they were transmitted.

Host-based routing was part of Phase IV but was not implemented in the release of DECnet/OSI, which required the use of dedicated external routers (such as DECnis and RouteAbout). Host-based routing was re-introduced into Phase V with DECnet-Plus and OpenVMS V7.1. **Host-based routing** allows an OpenVMS system to route data from a local area network (LAN) through a wide area network (WAN) that needs a separate dedicated router. In Phase V terminology, a node running host-based routing is an **OSI Intermediate System (IS)**. Note that it is entirely valid to have a routing node with a single path to provide routing updates about node reachability to other nodes on a LAN.

The Phase V equivalent of a Phase IV End Node is an **OSI End System (ES)**. Phase V nodes with multiple paths are referred to as **Multi-Homed systems**. They each include an out-of-order packet cache. Multi-Homed End Systems can thus load balance over multiple available paths.

The DIGITAL Network Architecture Phase V specification is published and can be purchased if required. Other protocols in the lower layers of the DECnet networking hierarchy also conform to their specific architectural specifications. For example, Ethernet (in all its variants) and the physical cabling such as Category 5 Twisted Pair conform to their own architectural specifications.

The DNA Phase IV specifications are available on the Internet.

Some of these specifications are made available under license, such as LAT (Local Area Transport), which is typically used by Terminal Servers (DECservers) to exchange serial terminal traffic with a host computer, typically over Ethernet.

The RFCs for TCP/IP constitute a set of specifications for individual components of the TCP/IP protocol suite.

A Brief History of DECnet

DECnet was originally designed and developed by Digital Equipment Corporation (DEC), whose outstanding architectural approach and engineering excellence laid the foundation for many modern software and hardware developments. DECnet has evolved over the years from a simple protocol (DNA Phase I) connecting two computers over an RS232 serial line to a sophisticated and complex protocol (DNA Phase V) capable of interconnecting a virtually unlimited number of systems, including those from other manufacturers. The DEC internal network (EasyNet) was probably the largest DECnet network to exist in terms of the number of connected nodes.

Over the past few years, DECnet has been overtaken by TCP/IP as the preferred means of interconnecting systems. To some extent, this has been driven by a simplistic approach to the provision of backbone networks by “managed service” suppliers whose networks are built on equipment from the major network hardware vendors. The majority of those vendors only provide TCP/IP routing, with some providing DECnet routing functionality at additional cost.

A number of vendors provide high bandwidth, low latency “layer 2 services” that give customers a greater choice of protocols to use. These are ideal for applications such as split-site clustering, in which OpenVMS uses the SCS cluster protocol and fibre-channel based storage subsystem interconnects.

DECnet Phase V implements the OSI 7 layer reference model for vendor-independent computer inter-networking. All the Phase V network management entities map clearly onto the OSI 7 layer model (this model is discussed in more detail later). One of the main design issues behind the introduction of Phase V was the need to increase the available address space to accommodate more nodes in the entire network, as well as to manage the routing updates for all the nodes. This need was elicited initially by the growth of the EasyNet DECnet network.

Each new version of the DECnet architectural specification retained backward compatibility with the previous version, thus facilitating migration from one version to another. Initially, DECnet was associated with the RSX and VAX VMS operating systems as the primary network mechanism for interconnecting such systems. (RSX is a PDP-11 operating system and the cultural predecessor to VMS; VMS is the original name of the OpenVMS operating system.) Each version of the network layer specification was linked with a particular version of the OpenVMS operating system; however, recent versions of the DECnet architecture have become independent of the OpenVMS operating system version, to the point where DECnet is no longer a prerequisite for installing and configuring OpenVMS (V7.1 and later).

Incidentally, this separation of DECnet from OpenVMS led to the need for other mechanisms for loading network-booted devices such as cluster satellites and terminal servers. This inspired the introduction of the LANACP and LANCP components, which provide a DECnet-independent method of servicing MOP load requests.

Figure 1 shows the relationships between the versions of OpenVMS and DECnet, from VAX VMS V4.0 with DECnet Phase IV to OpenVMS Alpha and I64 V8.2 with DECnet Phase V:

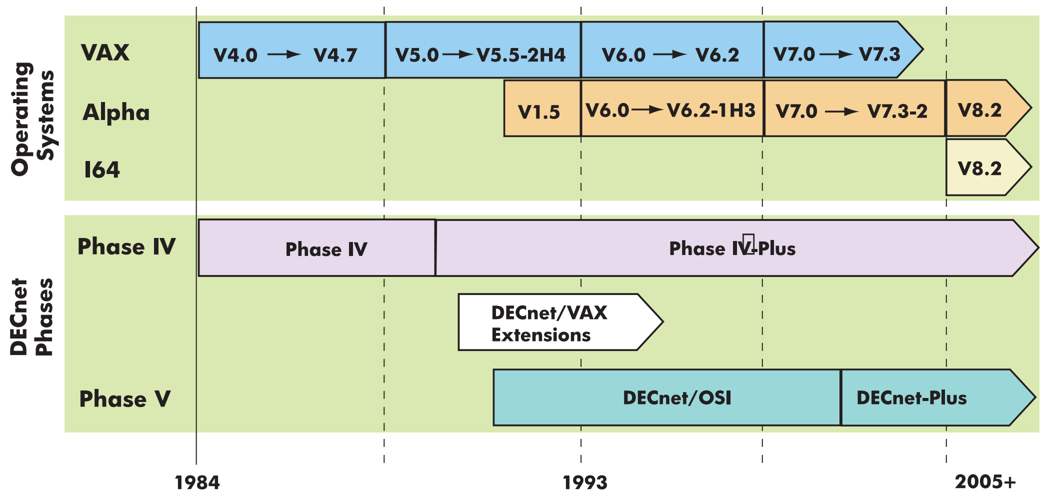

Figure 1 OpenVMS and DECnet Versions

During the evolution of the DECnet protocol, the inter-computer links themselves evolved rapidly in response to the changes undergone by the hardware technologies. One very important aspect of a layered network architecture design is that the end-to-end protocol for the exchange of data need not be changed when the lower layers change (such as the physical medium layer). For example, such a design approach allows you to replace a serial line with Ethernet and to subsequently evolve to gigabit Ethernet – all this without requiring changes to the upper protocol layers.

Another example demonstrating the importance of a sound architectural specification is the ability of the lower layers to operate today at many times the speed of the originally designed lower layers, and yet the old and new can co-exist without creating significant timing issues. Both Phase IV and Phase V follow the principles of a layered design with tightly defined and enforced interfaces between layers.

Table 1 lists some of the major changes and features introduced with each phase of DECnet:

Phase I	Device-specific point-to-point link between computers using low-speed wide area network (WAN) style serial links; limited to two computers.
Phase II	File transfer, remote file access, task-to-task programming interfaces, network management for point-to-point links; up to 32 computers.
Phase III	Adaptive routing (indirect connections between computers), downline loading, record level file access; up to 255 computers.
Phase IV	Ethernet local area network technologies, area routing, host services; up to 64,449 computers.
Phase V	OSI protocol support, transparent transport level links to TCP/IP, multivendor networking, local or distributed name service, distributed network management; virtually unlimited number of computers.

Table 1 The DECnet DNA Phases

Note that Phases I to III predate LAN technologies such as Ethernet.

Phase I was very basic and provided a device specific point to point link between computers using low speed WAN-style links.

Phase II increased the scale of the network, but still required point to point links between the computer systems.

Phase III increased the scale yet again, introducing the concept of routing (indirect connections between systems).

Phase IV increased the scale of the network significantly from the earlier phases, introducing a new address space to enable support of over 64,000 nodes and area routers to reduce the scope of the routing update problem incurred in earlier networks as they grew in size. Phase IV provides areas in the range 1 to 63 and addresses within each area in the range 1 to 1023, thus giving a total of 63x1023 possible node addresses on a single interconnected network.

Phase IV provides both End Node and Routing Node (layer 1 “within area” or layer 2 “between areas”) functionality. Originally, DECnet Phase IV was embedded in the OpenVMS system (Version 4.*x*, 5.*x*, 6.*x*, and 7.0) as the default network mechanism. For many years it formed the backbone of the Digital internal network known as EasyNet. Beginning with OpenVMS V7.1, DECnet Phase IV was no longer embedded as the default network protocol and became available instead as a separate layered product in the OpenVMS distribution kit.

Phase V was introduced primarily to solve addressing scheme limitations brought to light with the rapidly growing internet and also to provide full OSI networking for interoperability with other vendors systems. Phase V also provides full interoperability with Phase IV which is essential for implementing a phased transition from Phase IV to Phase V throughout an organization. Phase V was initially released in a number of stages such as Wave 1 and Wave 2; for example DECnet/VAX extensions for DECnet Phase IV.

Phase V came of age around the time of OpenVMS V6 with the release of DECnet/OSI V6. At this point, DECnet became a much better integrated and packaged product, particularly with the introduction of a local DNS-based local node database.

Phase V was initially implemented with OpenVMS systems acting only as OSI End Systems (End Node equivalent) and with dedicated hardware routers acting as OSI Intermediate Systems (Routing Node equivalent). This was due to both a purist approach to the overall design and to the performance restraints executing the Phase V routing code. With the introduction of a new, unfamiliar (but powerful) network management interface and the hardware expenditure and cabling disruption required for an upgrade from Phase IV to Phase V, most customers stayed with the simpler and more easily understood Phase IV version of DECnet. Phase IV met most of these customers’ requirements.

However, with the advent of the very powerful OpenVMS Alpha systems, host-based routing with OSI Intermediate Systems became feasible and was eventually re-introduced into OpenVMS to provide OSI IS-IS functionality. This eliminated the requirement to have dedicated hardware router devices, although from a network design viewpoint they may be preferable. The re-introduction of host-based routing allows the majority of DECnet users to migrate to DECnet-Plus. It also allows the use of small, inexpensive systems (such as DS10 and RX2600) as dedicated routers for both DECnet and TCP/IP protocols. These systems can also provide other network services, such as MOP downline loading and time servers.

The Layered Approach

The OSI Reference Model, otherwise known as the OSI Seven Layer Model, describes the various layers involved in networks such as DECnet-Plus. The purpose of the OSI Seven Layer Model is to enable dissimilar systems to communicate with each other, the objective being an open architectural specification capable of implementation by many different vendors, thus enabling a high degree of interoperability. Table 2 describes the seven layers of the OSI Reference Model:

Layer	**Name**	**Purpose**
Upper Layers		
7	Application	Supports application-specific and end-user processes. Provides for distributed processing and access, contains application programs and supporting protocols (such as FTAM).
6	Presentation	Maintains independence from differences in data representation by translating from application to network format and vice versa. Coordinates conversion of data and data formats to meet the needs of the individual applications.
5	Session	Organizes and structures the interactions between pairs of communicating applications. Establishes, manages, and terminates communication sessions between the applications.

Lower Layers		
4	Transport	Provides reliable transparent transfer of data between end systems with error recovery and flow control.
3	Network	Enables communication between network entities to provide switching, routing, forwarding, congestion control, error handling, packet sequencing, and so forth.
2	Data Link	Specifies the technique for moving data along network links between defined points on the network and how to detect and correct errors in the Physical layer (layer 1).
1	Physical	Connects systems to the physical communications media and transfers the data (bit stream) at the electrical and mechanical level.

Table 2 The OSI Reference Model Layers

The upper layers (layers five to seven) are often referred to as the application layers. The lower layers (transport layers) are typically implemented by the network infrastructure components (repeaters, bridges, switches, routers, and so forth). DIGITAL Network Architecture Phase V is based on this layered structure.

The layered model splits the elements of a network into interdependent layers, where each layer exchanges information with its corresponding layer on another system by means of using the underlying layers. Each layer provides a service to the layer immediately above it, and each layer is supported by all the layers beneath it.

As shown in Figure 2, the OSI layered model data is passed down the layers on one system from the application layer, across the physical connection at the bottom layer (layer one, the physical layer) and then back up the layers on the other system to its corresponding application layer. The vertical arrows show the actual flow of data between the layers of a node (inter-layer data exchange). The horizontal arrows reflect the resulting communication or understanding between corresponding layers (peer level data exchange) of the two nodes.

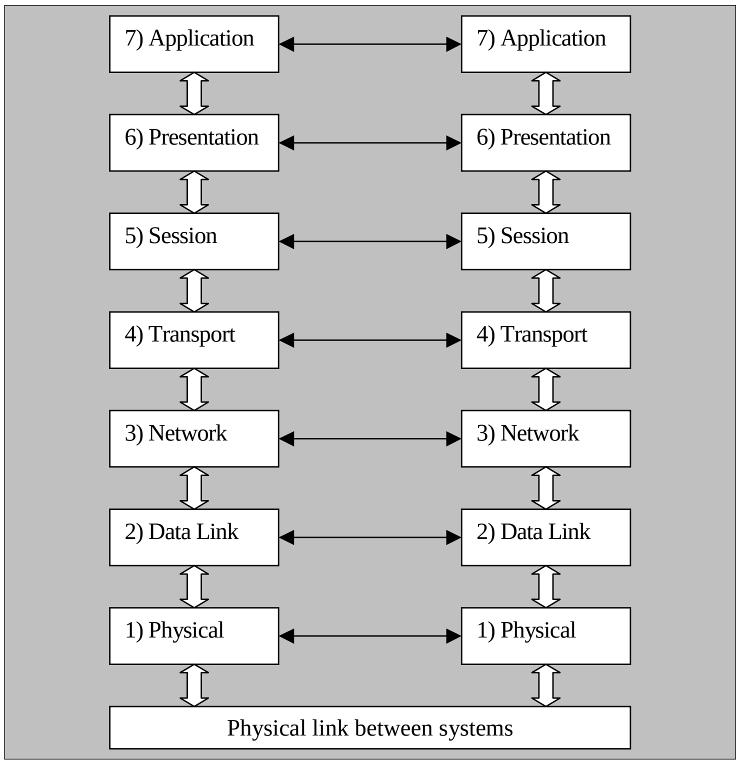

Figure 2 Data Flow in the OSI Model

DIGITAL Network Architecture Phase V

The remaining sections of this article focus on some of the major features and facilities provided by DECnet-Plus.

The Phase V Command Line Interface (NCL) and How It Differs from the Phase IV Interface (NCP)

One of the most important and noticeable differences between Phase IV and Phase V is in the command-line interfaces. DECnet Phase V introduced the Network Control Language (NCL) command interface, which replaces the DECnet Phase IV Network Control Program (NCP) command interface. The different layers in DECnet-Plus are much better structured for management purposes. The NCL entity hierarchy and syntax reflects the DECnet-Plus internal structure of the network management components, neatly mapped onto the OSI seven layer reference model.

Entities are the manageable components that make up the network; they relate to other entities on the same system. For example, the topmost entity in the management hierarchy is the node, which is identified by a globally unique node name. Next below that are various local entities (or child entities also referred to as module entities) such as Modem Connect Line, OSI Transport, DIGITAL Data Communications Message Protocol (DDCMP), and High-Level Data Link Control (HDLC).

There is only one instance of each of these module entities, so each can be identified uniquely. Each module entity has subentities below it. The HDLC module, for example, maintains HDLC LINK entities for each communications link over which the protocol operates; *hdlc link* is the full class name of the communication link entity. Each instance of the HDLC LINK entity requires further identification to allow it to be distinguished from the others. For example, in HDLC LINK HDLC-1, HDLC-1 is the instance name that further identifies the HDLC LINK entity.

NCL directives, or commands, let you manage DECnet-Plus entities by means of their unique network entity names. Unfortunately, users familiar with NCP might find the NCL interface verbose and difficult to grasp quickly; however, with a little patience and some of the helpful migration tools available, users will find that NCL is far more useful and powerful. Two tools that help users learn to use NCL include DECNET_MIGRATE in SYS$UPDATE, which provides approximate NCP to NCL equivalents, and NET$MGMT in SYS$SYSTEM, which provides a DECwindows interface and has an option to show the generated NCL commands.

Another major network management difference introduced with Phase V in DECnet-Plus is the management of network configuration databases. Phase IV network management involves a volatile and permanent configuration database. The volatile database stores active configuration values reflecting current conditions. They are in effect only while the network is running; they are lost when the network is shut down. The permanent database stores the initial values for configuration parameters, and these values are used when the network is started up. (Default values are embedded into the DECnet software and are overridden by changes to the permanent and volatile databases.) Changes made to the permanent configuration database remain after the network is shut down but do not affect the currently running network. With Phase IV management, volatile configuration values are modified with the NCP SET command, while the permanent configuration database values are set with the NCP DEFINE command.

Phase V performs all configuration actions at network startup by using NCL scripts to configure management entities as the network images are loaded and configured. It is helpful to regard these NCL scripts as constituting the permanent configuration database. The NCL scripts control all network entities except for node naming and addressing. You can edit these scripts with a text editor. Be careful not to make inadvertent changes. It is recommended to set the file version number to ;32767 to prevent inadvertent creation of new versions of NCL script files.

The NCL scripts do not set every single value for every single parameter associated with every single entity. The network software as shipped contains default values for many of the parameters. The actual values can be seen with the NCL SHOW *entity-name* ALL command.

Beware that these defaults may change from version to version of the released software (as indeed happens with OpenVMS itself, such as the recent changes to some parameter defaults in OpenVMS V8.2). Read the manuals and release notes carefully and try not to specify everything in the NCL scripts. As a general rule, it is best to use the default parameter values. Change them only if necessary, and only make the minimum changes necessary once you understand their effects and implications.

Each entity has several attribute groups: characteristics, counters, identifiers, and status attributes. Characteristics are the attributes you can modify, although not all of them are modifiable. Identifiers are set when the entity is created, and can be modified only by deleting the entity and recreating it with a new identifier. Note that changes to certain entity characteristics (static versus dynamic parameters – as with DECnet Phase IV or OpenVMS itself) do not take effect until the entities are restarted or if the system is rebooted.

You can also modify entities "on the fly" using NCL commands interactively (as done with NCP commands). This is useful for temporary changes to the running node, where you modify the in-memory active network software entities (as in the Phase IV volatile database). The changes become effective immediately but last only until the system is rebooted. For example, you might want to monitor a set of counters for a particular entity or you might want to temporarily disable a link.

NCL permits constructs such as WITH, which makes commands much more flexible, powerful, and useful. For example, in the following command the use of the WITH construct allows you to limit the display of session control port parameters to those with a specific value for the process identifier:

SHOW SESSION CONTROL PORT * WITH PROCESS IDENTIFIER = "PID_VALUE"

You can use this type of construct with almost all NCL commands. You can experiment to find commands that are most useful for your own circumstances. For example, try to find all currently available nodes on the LAN by asking the nearest router what active Phase IV-compatible addressing end systems it can see (hint – look for adjacencies on the routing circuits).

For more information on DECnet-Plus network management, refer to the *DECnet-Plus for OpenVMS Network Management* manual. For more information on NCL, refer to the *DECnet-Plus Network Control Language Reference* manual. NCL online help (using the NCL HELP command) is also extremely useful, especially with DECnet-Plus V7.3 and later.

The following are commonly useful entities and other manageable objects in the NCL command hierarchy:

- **Implementation** – The read-only Phase V implementation name and version as embedded in the software.
- **Node** – The local node addressing data. One node entity exists for the node module, and it crowns the hierarchy represented in the entity model described by the DNA specification. All other entities are subordinant to the node entity.
- **Session control application** – A network object (to use Phase IV terminology) that manages the session layer, negotiating and establishing connections.
- **Session control port** – The actual software link to the given application. The session control port stores session control information about the transport connection. One of the values is the corresponding **transport port**, which will be either an NSP port or an OSI transport port (see **NSP port / OSI transport port** below). Another value is the OpenVMS **process identifier** of the active process. This enables you to track the amount of network traffic each process is passing to each node.
- **Routing** – Manages the routing layer. Routes messages in the network and manages the message packet flow.
- **Routing circuit** – A data link or path to another node, available to the routing layer over a CSMA/CD station or DDCMP/HDLC link logical station.
- **NSP** – The Phase IV Network Services Protocol transport layer, it implements one of the protocols in the DNA transport layer.
- **OSI transport** – The OSI transport layer, which implements the OSI Connection-Oriented Transport Protocol specification (International Standard ISO 8073) on DECnet-Plus for OpenVMS.

- **OSI transport template** – The template available for use by the OSI transport layer; it supplies default values for certain parameters that influence the operation of a port on a transport connection. The two main templates are RFC1006 and RFC1006Plus (RFC1859). They implement the interface between the OSI transport layer and the TCP/IP ports 102 and 399 using the PATHWORKS Internet Protocol (PWIP) driver. This enables TCP/IP to be used as part of the transport layer for both OSI and DECnet communications between nodes.
- **NSP port / OSI transport port** – The actual software link over the given (NSP or OSI) transport. This is most useful for detecting what traffic (if any) is passing over a specific software connection. One of the values is the corresponding **session control port** (see above).
- **DDCMP / HDLC link** – A link using a port for the given (DDCMP or HDLC) protocol. DDCMP is the Digital Data Communications Message Protocol used over an asynchronous serial line (not supported by DECnet Phase V on OpenVMS VAX). HDLC is the High-level Data Link Control protocol, generally used over a synchronous serial line or frame relay.
- **DDCMP / HDLC link *xxx* logical station** – A specific connection over a DDCMP or HDLC link. It is useful for multi-drop links where a point-to-point link is a special case of a multi-drop link (for example, RS449).
- **Modem connect line** – The physical connection of a port used for DDCMP, HDLC, or a similar protocol. It is not applicable to CSMA/CD LANs.
- **CSMA-CD station** – A LAN adapter. CSMA/CD is the Collision Sense, Multiple Access, Collision Detect LAN protocol mechanism that includes LAN protocols such as Ethernet.
- **DTSS** – The DTSS server and clerk for synchronizing and managing system clocks in the network.
- **MOP** – The MOP (Maintenance Operations Protocol) database for downloading boot images over LANs.

Node Name and Address Resolution

DECnet Phase V implementations support several name services for storing and mapping node names and addressing information: the Local namespace, DECdns (DIGITAL Distributed Naming Service), and DNS/BIND used for DECnet over IP (also referred to as DOMAIN by DECnet-Plus software). Using NET$CONFIGURE, you can configure the naming lookup function to use any or all of the available name services in any specific order (you must select at least one). When you use more than one name service, the search list defines the order in which the existing services are to be accessed.

DECnet node names and address towers are maintained in DECdns and the Local namespace. These are managed using the DECNET_REGISTER utility.

TCP/IP host name and address information is maintained in DNS/BIND or in the local HOSTS database. These are managed using TCP/IP utilities.

The Local namespace (or Local Naming Option) is the most simple DECnet node naming mechanism. The Local namespace is similar to the permanent node database (NETNODE_REMOTE.DAT) used on DECnet Phase IV systems. All values are stored in a single data file on the local node and accessed similarly to how DECdns accesses names. The Local namespace can support up to 100,000 nodes. It works well for relatively small and static networks (10s or 100s of nodes in a well defined network that is not changing or expanding rapidly).

All node names in the Local namespace are prefixed with LOCAL:. to indicate the use of local naming. One main advantage of using the Local namespace is that it provides fast name-to-address lookups (it does not have to interrogate the nearest available name server as do other name services). In addition, no extra software is required for using this name service. The main disadvantage is you have to administer name-to-address mapping information separately on each node, and you must keep all nodes concurrently updated with the local naming database. This is not a significant problem for most networks, especially for stable networks in which the node population rarely changes.

DECdns provides a network-wide distributed database of node names for node name-to-address translation. DECdns is implemented as a global service accessible by any client node on the network. It ensures consistent network-wide name and address information. DECdns requires at least one and preferably a minimum of two nodes configured as DECdns servers. DECdns behaves in a similar manner to local naming except that node population changes can be made centrally at the DECdns servers, which will in turn automatically propagate the changes to all nodes in the network. Note that the use of DECdns can impose additional connection time when first establishing a network link. This is because establishing the connection requires a DECdns lookup if the name resolution data is not cached locally or the data has not been updated and propagated (thus invalidating the local cache).

You should establish at least one DECdns server per LAN in a large WAN interconnected network. This minimizes DECdns lookup times by other nodes on the LANs. One potential problem with DECdns is that having a single master read/write server and several read-only copies of that server can lead to vulnerabilities due to the single point of failure. If the single master read/write server fails, then updating DECdns node-address information might be temporarily impossible — for example, updating DECdfs (the Distributed File Service) entries to add access points.

DNS/BIND is the distributed name service for TCP/IP. It supports the storage of IP addresses and the use of node synonyms. Node synonyms allow for backward compatibility with older applications that cannot use long domain names. (DECnet-Plus also allows for node synonyms to provide backward compatibility with DECnet Phase IV node names.) DNS/BIND is needed if you want DECnet-Plus to run applications over TCP/IP. To use the DNS/BIND name service, DECnet-Plus requires one or more DNS/BIND servers in the network. DNS/BIND must be selected as one of the name services if you plan to use the DECnet over IP or OSI over TCP/IP features.

In general, use the Local namespace where possible, as it forces the network administrator to give more thought to the network node naming and addressing conventions used within an organization. In addition, once appropriate management procedures are in place, the simplicity of the Local namespace (not requiring configuration and management of servers as do DECdns and DNS/BIND) is much more preferable and can make fault finding much easier. All current network-related layered products (such as DECdfs) can operate in either a DECdns environment or in a local naming environment.

As noted, you can also use DECnet over IP, which uses the DNS/BIND name services as used in TCP/IP networks. This can greatly simplify consistent network-wide naming in a mixed-protocol environment. With DECnet over IP, end-to-end connectivity relies entirely on the underlying TCP/IP network configuration and infrastructure — the DECnet features of multiple paths and load balancing are no longer applicable; however, the availability features of TCP/IP (failSAFE IP) and OpenVMS (LAN failover) can be used instead. A DNS/BIND-less implementation is also possible by using the TCP/IP local HOSTS database to provide all the name resolution data.

One final note: the DECnet naming cache is non-volatile, meaning that it will survive reboots. If the naming information has changed, then you should make sure the naming cache is flushed to avoid the risk of stale cache entries and the confusion that follows. Flush the naming cache with the following command:

```
NCL FLUSH SESSION CONTROL NAMING CACHE ENTRY "*"
```

This will flush all cache entries. You can also use NCL to flush individual entries or groups of entries.

Availability and Reliability

DECnet and TCP/IP have very different approaches to re-routing on path failure. Other high-availability features related to LAN communications are now being built into the OpenVMS operating system.

DECnet node addressing is on a per-node basis and can thus provide both load balancing over all available data paths and automatic re-routing of data. These are handled by the end system with no external intervention and with minimal packet loss. With the appropriate changes to DECnet parameters, path failure can be detected quickly and failover can be achieved within a few seconds without disruption to the higher layer applications. This is best suited to a fully duplicated network infrastructure with separate multiple LANs.

In contrast to DECnet, TCP/IP addressing is on a per-interface basis. TCP/IP Services now provides failSAFE IP, which enables the IP address to be moved to a different physical interface when the primary interface fails or is no longer connected to the network infrastructure.

In addition, OpenVMS now provides mechanisms such as LAN failover so that all LAN protocols on a specific physical interface can move to an alternate physical interface when the primary interface fails or is no longer connected to the network infrastructure.

Node Names, Addresses, and Address Towers

Node names and synonyms help simplify the command line interface. For example, using node synonym XDVMS1, it is much easier to type the SET HOST command as SET HOST XDVMS1 rather than with the address as in SET HOST 10.240, or SET HOST 49::00-0A:AA-00-04-00-F0-28:21, or SET HOST IP$10.255.255.123. In this example, node XDVMS1 is running DECnet-Plus V7.3-1-ECO02 with Phase IV-compatible addressing enabled using OSI transport and the Local namespace. The true full node name is LOCAL:.XDVMS1.

The OpenVMS operating system uses the logical names SYS$NODE and SYS$NODE_FULLNAME for the Phase IV synonym and the full node name, respectively. The remote node (such as the target of a SET HOST command) implements the job logical names SYS$REM_NODE and SYS$REM_NODE_FULLNAME to provide the necessary information about the originating node for the incoming network connection.

The target node for a network request will perform a **back translation** of the inbound source address (from DECdns or the local namespace) to look up the corresponding name and will then cache it locally on the target node. This information is used to populate the SYS$REM_NODE and SYS$REM_NODE_FULLNAME job logical names on the target node so that software on the target node can easily determine where the inbound request originated. If the back translation fails, then the data provided in these logical names is simply the originator node address in DECnet, TCP/IP, or DECnet over IP format.

Nodes are registered in the DECdns or Local name databases using the DECNET_REGISTER tool. When a node is registered, the domain part (the part that precedes the ":.") is filled in with the appropriate domain name used by the relevant name service. The network administrator provides the name portion, the synonym (which defaults to the final part of the name portion after the "."), and the address tower information. If DECdns is used, then a node can **autoregister** its own address tower data based on its network adapter addresses, provided that DECdns access control has been configured correctly.

An **address tower set** stored in the namespace describes the protocols needed to establish a connection with a node. The address tower indicates which transport(s) to use to reach the destination node. The transports are either NSP or TP4, where NSP is the Phase IV compatible transport and TP4 is the OSI Class 4 transport. (Both transports can be used simultaneously but it is best to declare one only.) In addition, the address tower indicates which session control implementation is to be used. By default, SC3 is used for DECnet Phase V and SC2 for Phase IV. Finally, the address tower data contains the address field for the CLNS connection.

For example, the tower data for node XDVMS1 could appear as follows (using DECNET_REGISTER):

```
Address prefix = 49::
Fullname = LOCAL:.XDVMS1
Synonym = XDVMS1
Tower = SC2/NSP/CLNS=10.240,  Tower = SC3/TP4/CLNS=10.240
```

Notice that the tower data here contains two transport related entries for remote node XDVMS1 — one entry for OSI transport (SC3/TP4) and one for NSP (SC2/NSP). To connect to node XDVMS1 initiating only OSI transport connections (only available between Phase V nodes), simply delete the NSP (SC2/NSP) tower data entry for remote node XDVMS1 on the local node. To force initiation of only NSP connections (the only transport available on a Phase IV node), simply restrict the tower data to the NSP (SC2/NSP) entry by deleting the OSI transport (SC3/TP4) entry.

In general, do not attempt to use the OSI transport (SC3/TP4) for connecting your Phase V node to a Phase IV node (even with Phase IV-compatible address format). Otherwise, on attempting to connect to a Phase IV node, the initial connection attempt using the OSI transport will fail, and the connection attempt will have to be retried using the NSP transport.

For Phase V-to-Phase V communication, either transport can be used. Note that DECnet Phase V nodes will accept either transport for incoming connections using either SC2 or SC3. The address tower data is only used by the local node when initiating a connection to the remote node. In general, it is recommended to specify a single transport rather than both transports. This minimizes the timeout period if connection problems occur. If both transports are specified in the tower data, then both transports will be tried in succession, thus leading to a "double timeout," one per transport.

For detailed information on addresses for DECnet-Plus systems, including network service access points (NSAPs), service access points (SAPs), and OSI-style address formation, see the *DECnet-Plus Planning Guide*.

Phase IV Compatible Addressing

Phase V provides for a Phase IV-compatible address format. This is particularly relevant on CSMA/CD (Ethernet) LANs. The Phase IV address format uses the AA-00-04-00-*xx-xx* Ethernet MAC address, where the "*xx-xx*" portion is derived from the Phase IV area and node numbers. In contrast, the Phase V address format uses the hardware MAC address of the Ethernet adapter. This means that a pure Phase V node can use multiple adapters to connect to the same physical LAN while a Phase IV node cannot. DECnet Phase IV assumes one connection (station or network interface) per network segment, and multiple controllers are permitted only when there is no connection between the network segments, or when a DECnet router exists between the network segments. If more than one adapter attempts to start the DECnet protocols with the same (duplicate) MAC address, then subsequent adapters are prevented from coming on-line, although there can be timing issues with checking for duplicate MAC addresses, thus leading to unpredictable results.

This feature of Phase V provides the ability to load balance traffic over different adapters, which can be extremely useful especially for network reliability and throughput. It also permits the construction of bridged network configurations to provide increased availability of network paths. One major inconvenience is that existing LAN analyzers have to be reprogrammed to recognize a completely different set of Ethernet MAC addresses. Phase IV-compatible addressing still provides the ability to select the appropriate transport (NSP or OSI) for communication with other nodes, although only NSP is valid for communication with a pure Phase IV node.

Routing

With a mixed network of Phase IV and Phase V nodes and both NSP and OSI transports in use, the recommended routing mechanism is the Routing Vector Routing (RVR) algorithm as used by Phase IV Level 1 and Level 2 routers. DECnet-Plus host-based routing also uses the Routing Vector routing (RVR) algorithm. If the network has no Phase IV nodes and uses dedicated routers, then the recommended routing mechanism is Link State Routing (LSR) introduced with Phase V, which is faster to converge when the link topology changes. The main differences between the routing algorithms only become apparent in large networks with a large number of WAN links.

Prior to the release of DECnet-Plus V7.1 for OpenVMS V7.1, all routing functionality was provided by external dedicated routers (or OSI Intermediate Systems) such as DECnis routers. All Phase V implementations for OpenVMS were End Systems, analogous to the Phase IV End Node but with the ability to be Multi-Homed and thus they could drive several network paths in parallel simultaneously (giving features such as load balancing on dual-rail Ethernet).

DECnet-Plus V7.1 introduced host-based routing using the RVR mechanism. Host-based routing allows an OpenVMS system to operate as a DECnet-Plus intermediate system (IS). Host-based routing is analogous to the Phase IV Level 1 and Level 2 router functionality for OpenVMS systems that required a DECnet Extended Function (or Full-Function) license.

As already mentioned, host-based routing is useful for network configurations where data must be routed from a LAN to a wide area network (WAN) using an existing system rather than investing in a dedicated router. This will greatly benefit many network administrators who could not justify the purchase of additional hardware to provide routing functionality, although this was alleviated somewhat by Phase V with the ability to create multi-homed End Systems. Multi-homed End Systems can have multiple simultaneously active network paths to different locations but cannot route between these paths. This feature can be extremely useful from a network security viewpoint when isolating systems and network segments from each other.

It is strongly recommended that all networks have at least one router node, even on a single LAN. Routing provides considerable additional functionality and the presence of a router assists with node unreachability detection and notification. Without a router node, unreachable status can only be determined by End System timeouts, which by default are set to expire after several minutes. The addition of routing functionality detects the loss of end system hello messages and then propagates routing updates, thus informing non-routers of node reachability changes. Routers also provide end systems with the necessary information to initiate a connection on the correct path without having to try paths and wait for timeouts.

DTSS – Time Synchronization Service

DTSS is the Distributed Time Synchronization Service. It is the topmost entity of the DECdts (DIGITAL Distributed Time Service) module. It provides a network-wide time synchronization mechanism that can keep clocks on all nodes accurately synchronized to within a small number of milliseconds. Not only can nodes be synchronized relative to each other but also relative to an external time source such as an IRIG-B signal or some other externally available clock. The basic concept behind DTSS is the introduction of a time specification that includes an inaccuracy value. The time stamp consists of the current time value with an associated inaccuracy. Without an external time provider, this inaccuracy value is infinite, indicating that all nodes are synchronized relative to each other and not with an external time reference.

An example time provider exists in SYS$EXAMPLES:. You can modify it to provide a time provider that simply reads the local clock and returns that time stamp with a small inaccuracy value, thus forcing the inaccuracy value to be non-infinite. Alternatively, code can be written to interface to an external dial-up or locally connected time reference source. Such devices are easily obtained, and many of them are simple radio clocks receiving a broadcast signal, for example from the Rugby clock in the UK. Coding a time provider requires use of the UTC time/date routines to generate the correct timestamp data structure with appropriate time values and inaccuracy values.

DTSS synchronizes the local node's clock with local (LAN connected with a non-DECnet layer 2 protocol) and global (DECnet connected over WAN or LAN) time servers. To synchronize, DTSS requires at least one time server to provide a time stamp with low inaccuracy. By default, DTSS is configured to communicate with a minimum of three servers (either local or global, or a mixture of both) to make an accurate synchronization. DTSS uses time stamp data from all three servers to make a balanced estimate of the actual time. If you have insufficient time servers available, then you will receive OPCOM messages indicating that too few servers were detected by the DTSS clerk.

To avoid the OPCOM messages, you can change the value of the SERVERS REQUIRED value to 1 in the DTSS startup NCL script (NET$DTSS_CLERK_STARTUP.NCL or NET$DTSS_SERVER_STARTUP.NCL), as shown in the example that follows, and have a single node as a DTSS global server, where *x* is the number of actual DTSS servers available on the local LAN or over WAN links and that are registered as Global Timeservers:

```
NCL SET DTSS SERVERS REQUIRED x
```

Register a node as a global server either by making an entry in the DECdns .DTSS_GlobalTimeServers directory (if DECdns is in use) or by making an entry in the SYS$MANAGER:DTSS$CONFIG.DAT file (if not using DECdns).

The selection of DTSS server or clerk software is made at the time of installation of the DECnet-Plus software. The DTSS startup procedure sets up the relevant time zone information and starts the appropriate DTSS server or clerk software when the network is started at system boot. Time zone information is configured using NET$CONFIGURE in the usual manner.

It is also possible (again see SYS$EXAMPLES:) to use an NTP time server as a time source for a DTSS time provider process.

The simplest method of initiating time synchronization is to set the DTSS server's time manually from a watch, using an inaccuracy value of one second or so. When one time server has a low (non-infinite) inaccuracy component, the clocks on all other nodes will converge toward that server's time.

DTSS can also be disabled and prevented from starting on more recent versions of OpenVMS (V7.3-2 or later). You can prevent DTSS from starting at system boot by defining the NET$DISABLE_DTSS logical name in SYLOGICALS.COM. DTSS can be stopped on a running system by using the NCL DISABLE DTSS and NCL DELETE DTSS commands.

Associated Protocols – MOP and Remote Console

Maintenance Operation Protocol (MOP) is used to load executable code images into devices that boot from the local area network, such as DECserver terminal servers and OpenVMS cluster satellite nodes. As mentioned previously, as of OpenVMS V7.1 these load requests can be serviced by LANACP, DECnet Phase IV, or DECnet Phase V. The load host needs to have (1) the MOP service enabled and (2) a list of known devices and their corresponding MAC addresses, together with the load image file name and other associated data.

The Remote Console protocol is the mechanism used for establishing a remote console session on a network device, typically a DECserver.

With Phase V, use the following DCL SET HOST/MOP command to establish a remote console session. This command invokes the console carrier requester program and connects to the target device (*mop-client-name*) remote console interface.

SET HOST/MOP *mop-client-name*

Note that in contrast, Phase IV uses the NCP CONNECT NODE command to perform this function. In Phase IV, the target nodes need to be defined in the DECnet volatile database (usually loaded from the permanent database). This enables the MOP load request service mechanism to find the necessary load image and other node-specific data based on the Ethernet hardware MAC address information received from the device requesting a MOP load from the network.

The nodes booted in this manner do not necessarily run the DECnet protocol and thus do not really exist as DECnet nodes on the network. This has been a source of confusion for many system and network administrators. The term pseudo-node is more appropriate and the best practice is to allocate the DECnet Phase IV addresses to these pseudo-nodes so that they are easily distinguishable from the operational DECnet node addresses. For example, in a network where the various sites are split into areas 10, 11 and 12, the DECnet pseudo-node entries for MOP loaded devices could be in area 51, which does not physically exist in the chosen DECnet Phase IV addressing scheme. All MOP-loaded devices could safely be so configured across the entire network, assuming that there were less than 1023 such devices within the unused area 51. This would provide a clear separation of real DECnet nodes from MOP loaded devices not running DECnet, such as DECserver terminal servers.

Phase V introduced the more accurate term **MOP clients**. MOP clients do not need DECnet address information defined for them unless they are actually going to be running DECnet with the address information sent to the MOP client as part of the network boot process. This neatly distinguishes the terminal server type devices that need MOP to boot but do not run DECnet, from the cluster satellite type devices that run DECnet.

DECnet-Plus Installation Notes

Two DECnet license types are available: DVNETEND or DVNETEXT. Some of the installation options require the DVNETEXT license. The following are the license requirements for options discussed in this article:

- The DTSS server component does not require the DVNETEXT license.

- The DECdns server (re-introduced on OpenVMS Alpha V7.3-1) requires the DVNETEXT license.
- Host-based routing requires the DVNETEXT license. However, note that dedicated routers will always provide better functionality and greater performance for high traffic networks than host-based routers.

The OSI applications (VOTS, OSAK, FTAM) are installed separately from their kits located in subdirectories within the main kit directory structure.

The X.25 kit and WAN device drivers kit are separately licensed and installed products. The current version of X.25 is V1.6-ECO02. Note that X.25 V1.6-ECO02 is required for use on OpenVMS Alpha V7.3-2 (see the OpenVMS V7.3-2 release notes). X.25 V2.0 will support OpenVMS V8.2 Alpha and I64.

Changing the DECnet-Plus installation options (DTSS server, DNS server) after the initial installation requires any patch kits to be re-applied.

DECnet over IP Notes

Running DECnet over IP and OSI Applications over IP requires DECnet-Plus (or DECnet/OSI) and TCP/IP Services for OpenVMS V5.0 or later (or UCX V4.2-ECO04 or later).

Both OSI transport templates RFC1006 and RFC1006Plus must be configured in DECnet-Plus, and the PWIP driver must be configured in TCP/IP Services for OpenVMS. These provide the connection between DECnet and TCP/IP, allowing DECnet to use TCP/IP as a transport layer. This is implemented by the OSI Transport Templates for RFC1006 and RFC1006Plus (RFC1859), which map the data stream to TCP/IP port numbers 102 and 399. Note that it may be necessary to enable these TCP/IP port numbers on your firewall.

In addition, you must configure the DECnet naming service to use both the DNS/BIND (select "DOMAIN") and the Local or DECdns namespace options. This allows the TCP/IP naming to be resolved from the local TCP/IP HOSTS database or a TCP/IP DNS/BIND server. When using local naming for TCP/IP and DECnet, the order of name services is Local followed by DNS/BIND, and in either case the local node name (LOCAL:.*node-name* for DECnet, or the IP fully-qualified host name for TCP/IP) is the name of the relevant naming service. This allows DECnet to use the local naming file and TCP/IP to use the local HOSTS file. If you are using the local TCP/IP HOSTS database, then specify the TCP/IP DNS/BIND server as the local TCP/IP node.

DECnet will first attempt to establish a connection by looking up the name in the Local or DECdns namespace. It then resolves that name into the DECnet address. If DECnet over IP is enabled (RFC1006 & RFC1006Plus OSI transport templates, PWIP driver enabled and the DOMAIN name added to the local name path search list), then DECnet queries the local TCP/IP BIND resolver, which in turn looks up the local TCP/IP HOSTS database or queries the local TCP/IP BIND server to resolve the name into a TCP/IP address. This is configured by (re)running NET$CONFIGURE in ADVANCED mode to change the naming services and the OSI transport NCL scripts once TCP/IP Services for OpenVMS has been installed, configured (using TCPIP$CONFIG), and started with the PWIP driver enabled. When configuring the OSI transport, answer "Yes" to the "Do you want to run OSI applications over IP" and "Do you want to run DECnet over IP" questions.

Useful Commands:

- To list the BG devices, corresponding ports, and so forth:

  ```
  TCPIP SHOW DEVICE
  ```

  This command should display the RFC1006 and RFC1006Plus ports (102 and 399, respectively). DECnet over IP needs the RFC1006Plus port for allowing the IP data stream to be handed to the RFC1006Plus OSI transport template in DECnet. For full details of the extensions to RFC1006, see RFC1859.

- List the active OSI transport templates:

  ```
  NCL SHOW OSI TRANSPORT TEMPLATE * ALL
  ```

The display should confirm the presence of the RFC1006 and RFC1006Plus OSI transport templates. The NCL commands to create and start those OSI transport templates are in the NET$OSI_TRANSPORT_STARTUP.NCL, which is created by NET$CONFIGURE.

Current Versions of DNA Phase V Implementations

As of the time of writing (February 2005), several Phase V implementations exist, depending on the version of OpenVMS and the type of hardware in use:

- DECnet/OSI V6.3 with ECO16 on OpenVMS VAX V6.2
- DECnet/OSI V6.3 with ECO16 on OpenVMS Alpha V6.2-1H3
- DECnet-Plus V7.3 with MUP01 and ECO04 on OpenVMS VAX V7.3
- DECnet-Plus V7.3-1 with MUP01 and ECO03 on OpenVMS Alpha V7.3-1
- DECnet-Plus V7.3-2 with MUP01 and ECO01 on OpenVMS Alpha V7.3-2
- DECnet-Plus V8.2 with ECO01 on OpenVMS Alpha V8.2
- DECnet-Plus V8.2 with ECO01 on OpenVMS I64 V8.2

Make sure all relevant patches have been applied to the base operating system, especially the LAN driver patches for V7.3, V7.3-1 and V7.3-2.

For more information

Further information is available at:

- http://www.hp.com/go/openvms/doc - OpenVMS Documentation
- http://www.hp.com/go/openvms/wizard - Ask the Wizard
- http://www.research.digital.com/wrl/DECarchives/DTJ
- http://ftp.digital.com/pub/DEC/DECnet/PhaseIV/ — DNA Phase IV specifications
- http://rfc.net then searching for “decnet”
- http://standards.ieee.org/getieee802/802.3.html — IEEE802.3 (Ethernet) specifications
- http://linux-decnet.sourceforge.net

You can contact the author directly by e-mail at vmstj@xdelta.co.uk.