The ULTRIX Implementation of DECnet/OSI

1 Abstract

The DECnet/OSI for ULTRIX software was developed to allow the ULTRIX operating system and ULTRIX workstation software systems to operate in a multivendor, multiprotocol network based on open standards. It operates in a complex networking environment that includes OSI, DECnet Phase IV, X.25, and TCP/IP protocols. BSD sockets and protocol switch tables provide the entry points that define interfaces for protocol modules. The DECnet /OSI for ULTRIX software incorporates Digital's Enterprise Management Architecture, which provides a framework on which to consistently manage the various components of a distributed system. The DECnet/OSI for ULTRIX software provides a set of powerful tools and a system that can be extended to include new functions as they are incorporated in the OSI standard.

DECnet/OSI for ULTRIX is an end system implementation that supports the open systems interconnection (OSI) protocol through the Digital Networking Architecture (DNA) Phase V software. This implementation provides several features and programming environments that are consistent with the UNIX system philosophy of networking. Ease of use, extensibility, and portability were key design goals during product development. Operation of DECnet/OSI for ULTRIX software in a complex networking environment provides coexistence and interaction with the transmission control protocol/internet protocol (TCP/IP), DECnet Phase IV, X.25, and multivendor OSI networks.

The paper "Overview of Digital's Open Networking" (in this issue) provides a suitable introduction to DECnet/OSI concepts.[1] For more details concerning standard Berkeley Software Distribution (BSD) networking concepts, the reader is referred to the general references listed at the end of this paper.

This paper provides an overview of DECnet/OSI for ULTRIX software. It discusses some of the design decisions made during product development, including the use of protocol switch tables. It describes the system's five communication domains, emphasizing the X.25, data link, and OSI domains. The paper continues with a discussion of application programming interfaces, interfaces into kernel modules, and a network management interface established for extensibility. It concludes with a description of network management and network configuration.

2 System Overview

DECnet/OSI for ULTRIX is an end system implementation of the OSI network architecture and Digital's DNA Phase V. The DNA Phase V architecture provides a framework for incorporating OSI protocols as defined by the

International Organization for Standardization (ISO) into DECnet/OSI products. DECnet/OSI for ULTRIX software is integrated into the ULTRIX kernel and layered on existing ULTRIX interfaces. This software allows the

ULTRIX operating system and ULTRIX workstation software (UWS) systems to operate in a multivendor, multiprotocol network based on open standards.

The DECnet/OSI for ULTRIX software provides the following network services:

- Base networking software, which includes transport services, network layer services, X.25, and local area and wide area device driver support as described in the ISO Reference Model and DNA.[2]

- Network management software, incorporating the Digital Enterprise Management Architecture.

- Application programming interfaces to support user development of distributed applications.

- DECnet application software. DNA session control bridges DECnet applications such as file transfer (dcp,dls,drm), remote login (dlogin), and mail to transport layer services.

- DECdns, Digital's distributed name service, which provides a location-independent naming facility. This service is used by DNA session control to provide node name-to-address translations.[3]

- Digital's distributed time service, DECdts. This time synchronization service is required by many distributed applications such as DECdns to maintain a consistent time base for their operations.

- OSI applications software, including file transfer, access, and management (FTAM) and virtual terminal protocol (VTP) support.

3 System Goals and Development

A major goal of DECnet/OSI for ULTRIX was to support large multivendor, multiprotocol networks, including coexistence of OSI and TCP/IP on an ULTRIX UWS system. Coexistence includes the ability to share system resources and to provide a consistent set of services to users of both the OSI and internet protocols. Another goal was to provide connectivity between OSI and TCP/IP networks through the implementations of gateways and hybrid stacks.

Interoperability between DECnet/OSI and DECnet Phase IV products was required to maintain connectivity during network transition to OSI. A framework for the development of new OSI applications such as FTAM was another requirement. As in the DECnet-ULTRIX Phase IV implementation, programming and user interfaces needed to be consistent with the ULTRIX and UNIX systems environment.

Wherever possible, code was to be shared with other development projects. For this reason, software development engineers used the C programming language and aimed to produce a portable implementation. This was particularly important for the X.25 implementation, which would be used in other products. The code was structured to minimize system-specific references and dependencies. Code that interfaced directly to the BSD system was isolated in separate modules, and use of system-specific devices such as timers and buffers was hidden behind generic macros or subroutines.

In addition, the software was designed to be extensible so that future OSI protocols could be added. To achieve extensibility, interfaces were established between the various components. These include application programming interfaces, interfaces into each kernel module, and a network management interface. New protocols could be more easily added by supporting these interfaces.

DECnet/OSI for ULTRIX development began with a collection of eight distinct projects, each with its own goals, schedules, and priorities. These projects were developed across engineering organizations, and spanned three continents. They consisted of X.25, wide area device drivers, FTAM, VTP, DECdts, DECdns, OSI applications kernel (OSAK), and the DECnet/OSI base components.

Early in development, it was realized that no individual project could be successful without achieving success at a systems level for the DECnet/OSI for ULTRIX product. This realization caused a change in the way the DECnet /OSI for ULTRIX projects approached engineering development. Our focus switched to providing a common set of goals and one integrated schedule. Priorities for individual projects were reevaluated in the context of the system goals and schedule. It was critical to have a set of well-defined interfaces; any change to these interfaces could have a major system impact. Communication between all projects was essential. A significant amount of time was built into the schedule for system integration, as well as component integration.

4 Kernel Networking Environment

The DECnet/OSI for ULTRIX kernel implementation was designed to be consistent with other ULTRIX networking implementations such as the TCP /IP and Local Area Transport (LAT). The networking structure is based on the BSD networking subsystem.[4]

The ULTRIX networking environment allows protocol components to be insulated from each other. One important aspect of this networking system is the use of protocol switch tables. These tables contain the entry points for various protocol modules in the system, as shown in Figure 1. DECnet/OSI for ULTRIX uses these entry points to define interfaces for each protocol module. This means that there are no direct calls from one protocol component into another, an important consideration when new layers must be integrated. Moreover, one protocol module does not access another's databases. Information is accessed from a module only through the defined interface.

Insulating protocol modules from each other is advantageous for various reasons. As long as a protocol module supports a generic interface, it can act as a service provider for multiple users, which allows a system to support multiple configurations. For example, X.25 or high-level data link control (HDLC) may be configured into the kernel only when those services are needed. New protocol modules can be easily added. If token ring support is added as one of the broadcast devices, using the same interface as the carrier sense multiple access with collision detection (CSMA/CD) and fiber distributed data interface (FDDI) modules, little or no change will be required to the network layer.

Modularity is another advantage. Complexity can be reduced and problems can be isolated more easily when interfaces between each protocol module are carefully defined. For example, defining a network management interface for each protocol removes the requirement for network management to access protocol module databases directly. Network management code does not need to understand the internal organization of a module or the locking strategies that may be required to access the data.

To make use of the protocol switch table entry points, some minor enhancements were required. An extension was made to the control output interface to allow requests from kernel-level protocol modules and network management. The interface was further extended to allow protocol modules to use a port option to identify themselves as clients of the service provider, to acquire information from the service provider, or to modify the service provider's behavior. Network management uses a different option passed through the control output interface to manage kernel entities.

The control input interface was also enhanced. This interface provides two arguments: a request and a pointer to one or more arguments to be interpreted as a function of the request. Originally, this routine was used to notify IP of events, where each event had its own unique request value. To allow DECnet/OSI protocols to use this interface without adding several new request values, a general-purpose request was introduced. This request is used by a service provider to interrupt one or more of its clients to inform them of a change in service. As part of the argument list, the service provider passes a value indicating the exact nature of the event being communicated. As an example, the network layer uses this mechanism to inform the transport layer modules of a change to the set of network addresses. Similarly, X.25 uses this interface to provide status about specific network connections.

The ULTRIX/BSD networking system organizes protocols into communication domains. The purpose of a communication domain is to group together common properties necessary for process-to-process communication. As an example, the X.25 domain was designed to provide a full set of X.25 services that can be selected by client protocols. It includes the socket and protocol switch table interfaces necessary for user-level and kernel-level clients, X.25 accounting, profile loading, and trace utilities.

The components of DECnet/OSI for ULTRIX may be combined in different ways depending on the configuration requirements of individual customers. A multiple domain approach was chosen to allow the various products and their development to be separated from one another. For example, network management software was placed in a separate domain to allow the X.25 and wide area network device driver (WANDD) products to be managed without installing DECnet/OSI for ULTRIX. Similarly, the OSI domain protocols may operate without the X.25 or WANDD products configured into the system.

Five domains were established:

1. The DECnet domain (AF_DECnet) is retained to provide backward compatibility to existing DECnet-ULTRIX Phase IV applications.

2. The data link domain (AF_DLI) contains all the data link protocols, including Logical Link Control (ISO 8802-2), CSMA/CD, FDDI, and HDLC. For DECnet/OSI for ULTRIX, the AF_DLI domain provides access to the drivers for kernel modules as well as user applications.

3. The X.25 domain (AF_X25) contains the protocols necessary to access X.25 networks.

4. The OSI domain (AF_OSI) contains the higher-level DECnet/OSI protocols, i.e., DNA session control, network services protocol (NSP), OSI transport, DNA Phase V routing.

5. The network management domain (AF_NETMAN) contains all the network management functions. These functions can be used to manage any DNA networking product.

Data Link Domain

Under DECnet-ULTRIX Phase IV, the routing protocol module accessed the drivers directly. In the OSI implementation, data link interface (DLI) modules interface to the device drivers and act as service providers to network layer clients such as routing. This decision was made to minimize specific DECnet/OSI support needed in the ULTRIX operating system device drivers. This allows changes to be made more easily, and it provides a central location for common data link protocol code as well as network management code.

The AF_DLI domain provides a common interface to broadcast data links such as CSMA/CD and FDDI. Modules implementing new broadcast data link technologies can be added at any time by conforming to the DLI interface. DLI provides support for ISO 802.2 class I, type 1 functions; these may be used by any broadcast module. Other 802.2 classes are handled by passing frames directly to the client module.

The point-to-point protocols consist of HDLC and the Digital data communications message protocol (DDCMP). ULTRIX relies on the DDCMP support provided by hardware devices. However, a DDCMP software module exists to interface these devices to network management. HDLC, on the other hand, is entirely implemented as a software module operating over a device driver. Similar interfaces are provided by each protocol.

X.25 Domain

To ensure consistency with the goals and requirements of DECnet/OSI for ULTRIX, several design alternatives were considered for integrating X.25 into ULTRIX, including porting a previous Digital implementation of X.25, the VAX Packet Switch Interconnect. These alternatives were rejected because they were not consistent with the DECnet/OSI for ULTRIX implementation and BSD networking in general. A new version of X.25 was implemented in the C language using the protocol switch table infrastructure. This approach provided enough flexibility to allow the ULTRIX X.25 code to be easily ported to other product environments such as the WANrouter 250.

The X.25 components of DECnet/OSI for ULTRIX are provided as part of a wider X.25 strategy that can support multiple protocol suites, such as DECnet/OSI, TCP/IP, and International Business Machine Corporation's Systems Network Architecture (SNA). Under DECnet/OSI for ULTRIX, X.25 is used in two configurations. It provides the connection oriented network services (CONS) support to the OSI transport layer (ISO 8208, ISO 8878), and it can be used as a subnetwork for the connectionless network service (CLNS) layer. When used with TCP/IP networks, X.25 can be used as a subnetwork for the IP (Request for Comment [RFC] 877).

The interface to X.25 services was designed to be accessed by other kernel components. The protocol switch table was used to implement this interface. Components such as OSI connectionless network protocol and OSI transport make direct use of the kernel protocol switch interface with no intervening software layer.

Access by user-level applications to X.25 occurs through the BSD socket interface. The processing requirements of the socket layer and the kernel layer provided by the protocol switch are considerably different. To reduce the complexity of the kernel interface, an X.25 socket converter module was provided. The socket converter module manages issues such as queuing data at the socket interface and converting between protocol switch table routines and socket-layer calls. The converter module is treated as a client of the kernel interface.

Direct access to the X.25 kernel interface from IP was not possible due to TCP/IP development constraints. Instead, an IP device converter was supplied with ULTRIX X.25. This X.25-IP interface module appears as a device driver to IP. Furthermore, IP can be configured to use X.25 without requiring changes to the TCP/IP software. The pseudo-driver establishes an X.25 call when data is sent to the X.25 device. After the IP data has been transmitted, the X.25 connection is maintained to reduce the overhead and cost of X.25 call setup when the next IP data packet is sent. Configuration of the X.25 IP device is performed using standard ifconfig management commands.

OSI Domain

The AF_OSI domain contains the routing module, the transport modules, and DNA session control. The routing module is an end system implementation that adheres to the Digital Network Architecture (Phase V) Network Routing Layer Functional Specification, version 3.0.0. It provides support for the ISO Connectionless Network Service (ISO 8473), End System to Intermediate System Routing Exchange Protocol (ISO 9542), and Phase IV routing. "Ping," a network loopback function specified in {italAmendment X: Addition of an Echo Function to ISO 8473) and in RFC 1139, is provided as a diagnostic tool to test network access to a node.

Routing can be configured to operate over the data link entities previously mentioned as well as X.25. As an end system, DECnet/OSI for ULTRIX does not route protocol data units (PDUs). It can, however, operate over multiple circuits simultaneously, which allows load balancing across circuits and network redundancy. Phase V routing is capable of autoconfiguring to one or more network addresses.[5]

OSI transport (ISO 8072, ISO 8073) and NSP are the two transport modules supported. Both can be configured to operate over CLNS. However, only OSI transport can be configured to operate over CONS/X.25. OSI transport class 4 is supported over CLNS, and classes 0, 2, and 4 are supported over CONS/X.25. OSI transport also provides a connectionless transport service (CLTS) to its users. CLTS is a datagram service that operates over CLNS.

OSI transport supports two client interfaces and NSP supports one. Both support an interface to DNA session control supplied by the protocol switch table entry points. OSI user applications directly access OSI transport through X/Open transport interface (XTI).[6] XTI specifies a transport service interface that is independent of the transport provider. On the ULTRIX implementation, XTI is a library interface implemented using the socket layer. It is discussed in more detail later in the section Application Programming Interfaces.

OSI transport can have multiple clients, and it identifies each client by an address called the transport selector. When OSI transport processes an incoming connect request, it uses the selector to determine which client should receive notification of the request.

The DNA session control protocol engine was implemented as part of NSP for the DECnet-ULTRIX Phase IV release. It is now implemented as a separate entity to allow operation over multiple transports (NSP and OSI transport). This modification created a subtle problem. DNA session control resides between the transport layers and the socket layer. However, both transport modules and DNA session control need access to the socket. DNA session control needs access when performing connection control, and the transport modules need access when appending transmit or receive buffers to the socket queues. Since the socket is actually open to DNA session control, a mechanism was created to relay the socket pointer to the transport modules. This information is passed through the control output interface as part of the port option.

5 Application Programming Interfaces

To ease the transition of applications from Phase IV to DECnet/OSI, the Phase IV socket interface and programming library were retained. Applications using these interfaces will continue to work. This allows programmers time to modify their applications to use the new interfaces and the capabilities provided with DECnet/OSI for ULTRIX.

New application programming interfaces (APIs) were developed. These APIs include a DNA Phase V session control programming library, an X.25 programming library, an X.25 socket interface, and an XTI interface. They allow programmers to write network applications that use DECnet/OSI capabilities.

DNA Session Control Library

Through the use of the DNA Phase V session control library and DECdns, applications can provide location-independent services to the network. DNA session control stores information about an application and its services in an object in the DECdns namespace. Client applications can access these services by referencing the object name without knowing the current location of the service.

DNA Phase V session control applications also have the option of operating over various transport services and network services. The library gives the application programmer the flexibility of specifying the particular combination of services to be used. As an alternative, the library can determine the possible combinations of protocols that are supported on both the local and remote systems. This is done by accessing the addressing information stored in DECdns for each of these systems. If any combinations of protocols exist, DNA session control tries each of them in succession until a connection is established.

The DNA Phase V session control programming library is designed to be extensible. Instead of using a calling sequence with numerous parameters, one parameter is passed on all calls. This parameter is an extensible data structure that consists of both input and output arguments. It allows new arguments to be added by appending fields to the end of the data structure.

The library is designed to support multithreaded application development. If a threads programming interface is supported on the ULTRIX operating system, programmers are able to write applications that have multiple control paths executing in parallel. This is useful in writing a network server application that frequently needs to handle requests from multiple clients. A single server application can process requests in parallel instead of creating additional processes to service each request. Multithreaded support in the library was accomplished by removing the use of static and global data by the library. Information is returned in dynamically allocated memory, which the applications are responsible for freeing.

X.25 Interfaces

Two programming interfaces are provided for the X.25 component. A socket interface is provided for full access to X.25 features in a manner compatible with BSD UNIX. This allows applications to make use of a direct socket interface to both TCP/IP and X.25.

An X.25 programming library was created to provide a portable programming interface that could be used for access to X.25 across current and future Digital implementations. The format of calls to the X.25 library was constructed on lines more compatible with the interface defined in the DNA X.25 access architecture than that available through the socket interface.

XTI Library

The XTI library has been extended to provide a framework for developing OSI applications. XTI provides a transport-independent programming interface that is standard across UNIX operating systems. On ULTRIX, XTI was implemented to provide a portable interface for writing TCP/IP applications. In DECnet/OSI for ULTRIX, the implementation was extended to provide support for OSI transport, including both connection oriented transport service (COTS) and CLTS. In addition to supporting the standard XTI calls, service routines were implemented. These routines provide a mechanism to build and access addressing information needed within XTI. The addressing information consists of transport selectors, network addresses, and internet ports.

Support for the Internet RFC 1006 specification was also added to the XTI library.[7] This specification allows OSI applications to run over the TCP /IP protocol suite. RFC 1006 defines a mechanism for OSI transport class 0 (TP0) messages to be mapped across a TCP connection. OSI applications can be written to communicate over either TCP/IP networks or OSI networks, using the same API.

An RFC 1006 daemon was implemented to work in conjunction with the XTI library to handle incoming connection establishment. To allow multiple OSI applications to bind to the same RFC 1006 TCP port, a simple protocol exchanges file descriptors and a few basic messages between the XTI library and the daemon, using UNIX domain sockets. RFC 1006 specifies that a TCP connection be completed and a TP0 connect request be received before an OSI application server can be selected to process the incoming connect. The daemon hides the TCP connection and effectively blocks the OSI application server until the TP0 connect request occurs.

6 Network Management

DECnet/OSI network management is completely different from the management provided for DECnet Phase IV. It is based on the Enterprise Management Architecture (EMA), which provides a framework to consistently manage the various components making up a distributed system.[8] DECnet/OSI for ULTRIX network management consists of a director, an event logger, an agent access module, and an agent for each manageable protocol entity. Figure 2 shows the network management environment.

The director, network control language (NCL), provides the user interface that allows network management commands to be entered. NCL encodes the network management commands using the common management information protocol (CMIP). The encoded directives are passed to the common management listener (CML). CML, in turn, passes the directives to the appropriate agent in a form the agent can understand. On the ULTRIX implementation, when the connection between NCL and CML is local, a pipe is used. When NCL needs to connect to a remote CML, an OSI network connection is established.

The event logger (EVL) takes event messages generated by agents and sends them to either a local sink or a remote event sink. A local sink is a process that is executing locally, but a remote event sink is executing on a system elsewhere in the network. In the latter case, the CMIP protocol is used to convey the event message. Events are typically displayed on the console or in a file.

The DECnet/OSI for ULTRIX network management implementation is designed to be modular and extensible. The data dictionary, a key component, describes all the management attributes of each entity. The data dictionary is a dynamically extensible database and is used by all network management applications. NCL uses the data dictionary to parse command lines and display output. CML uses the data dictionary to decode/encode CMIP protocol messages from/to NCL, and EVL uses it to display an event locally. Information about new attributes or entire entities can be added to the data dictionary without modifying the network management applications. Thus layered products can easily add support for new manageable objects.

The network management environment in DECnet/OSI for ULTRIX is essentially a message passing scheme, as shown in Figure 2. Like the data dictionary, it was designed to be extensible and generic. All manageable, DNA-architected entities use this environment. At the core is a switch, kernel CML. Kernel CML passes messages between user CML and any DNA entity. User CML and kernel CML communicate through the socket layer. User-level agents, in turn, communicate with CML using the socket-layer interface, and kernel-level agents communicate with CML through the control output routine for the entity.

User-level agents can send multiple responses to a single request, but kernel-level agents can send only one response per request. Because user-level agents reside in process space and are separated by the socket layer, their transactions can be asynchronous. Transactions of kernel-level agents, on the other hand, must be synchronous. When called, they must process the request and return a single response. Whenever multiple responses are to be returned, as in a wild-card operation, the agent relies on being invoked again by kernel CML for each of the response messages. This programming precludes the possibility of exhausting system buffers while conveying information about a large number of subentities. Kernel CML stops requesting additional responses from a kernel entity when it detects that the socket receive queue is full. Once there is more room on the queue, it resumes the wild-card operation.

The network management environment provides a core set of routines as an aid to processing and building the syntax for each message. It also provides routines that assist in wild-card processing. Agents that make use of these routines need not be aware of the physical structure of each message. This has several benefits. It provides a common set of code that is not duplicated from entity to entity. If there is a problem, it is corrected in one location instead of several. Also, it makes the implementation more portable. The message passing scheme uses the local operating system's network buffers. When changing from one operating system to another, the buffering needs to change only in the common code and not in each of the agents.

Entities may need to originate event messages bound for the EVL. The mechanism providing this support is basically the same as the message passing scheme previously described. A kernel EVL switch receives event messages from either a user-level or kernel-level agent and passes the event up to its counterpart through the socket layer. With this mechanism, however, messages flow in only one direction, from the entity to the event logger.

In DECnet/OSI, some significant architectural changes were made to the maintenance operations protocol (MOP). As in Phase IV, the current implementation supports down-line loading and up-line dumping over FDDI and CSMA/CD devices. These functions are now performed by using the MOP version 4.0 protocol over ISO 8802-2 or MOP version 3.0 over Ethernet. As part of implementing the new protocol, support for down-line loading CMIP scripts was added. These are used by remote systems such as DECnet/OSI routers to perform network management initialization. Client information is kept in a MOP-specific database. By keeping entity-specific information modular and distinct, the DECnet/OSI for ULTRIX MOP implementation is consistent with EMA. This contrasts with the DECnet-ULTRIX Phase IV implementation, which stores MOP client information in the DECnet nodes database.

7 Applications Supported

The DECnet Phase IV applications continue to be provided with the DECnet/OSI for ULTRIX product. These include the file transfer utility, dcp, the remote terminal utility, dlogin, and the mail utility. These DECnet applications have been modified to use the DECnet/OSI for ULTRIX programming interface and to take advantage of the new DNA Phase V capabilities. They can accept DECdns full names for node names and run over both the NSP and OSI transport. The DECnet-internet gateway is also provided as part of the product. The gateway provides bidirectional network access between DECnet and internet systems. It allows DECnet and TCP/IP users to communicate through their respective file transfer, remote login, and mail facilities.

New OSI applications were written to provide similar capabilities to the DECnet applications. They allow users to access files and terminal emulation in a multivendor environment. These OSI applications include FTAM, VTP, and X.29 terminal support. Just as the DECnet-internet gateway is provided, OSI applications provide their own gateways to link OSI and internet.[9]

ULTRIX X.25 includes X.29 terminal support. A packet assembler/disassembler (PAD) provides outgoing access. Thus PAD allows terminal emulation for X.25 connections to remote hosts in much the same way that the VTP does in a full OSI stack. For incoming X.29 calls, a UNIX daemon creates an X.29 login process or activates an application based on X.29.

8 Installation and Configuration

DECnet/OSI for ULTRIX networking software allows the use of OSI addressing and access to global naming services. It provides new network management utilities and the ability to configure a network stack in many different ways. For example, in configuring X.25, many attributes can be set to allow conformance to many public and private packet-switched data networks. The new capabilities add a degree of complexity to the process of configuring the networking software. To simplify this process, configuration was separated from installation. Installation occurs when files are moved from the distribution media to the target system. Configuration is the process of providing information to make the networking subsystem operational.

The ULTRIX DECnet/OSI and X.25 setup utilities provide two modes of configuration, basic and advanced. The DECnet/OSI for ULTRIX setup basic configuration process asks a limited number of questions and is designed for the user who will be installing DECnet/OSI for ULTRIX on a workstation connected to a local area network. The advanced configuration process and X.25 setup utility provide more configuration choices for the network manager who will be installing DECnet/OSI for ULTRIX in a server configuration, or who will require more detailed network configurations.

X.25 and wide area network device driver setup utilities supply a mechanism for configuring TCP/IP or DECnet/OSI for ULTRIX to run over X.25 or synchronous data links. For a more unified approach to configuring an OSI stack, these setup utilities are integrated with the DECnet/OSI for ULTRIX setup advanced process. These setup utilities add a logical abstraction above the EMA, which helps to reduce complexity. For each manageable entity on the system, NCL scripts are generated through default assumptions and responses to configuration questions.

Network configuration is accomplished with shell scripts and network management scripts. These mechanisms initialize manageable entities. At system start-up, the decnetstartup script is executed from within rc.local. This invokes the various NCL scripts to configure the networking software. One or more NCL scripts can be modified independently of the configuration utilities to change attributes of the manageable entities. As an alternative, the setup utilities can be rerun to modify the scripts.

In addition, responses to configuration questions are stored in a file to provide default answers to simplify subsequent reconfiguration.

9 Summary

The design of DECnet/OSI for ULTRIX was a challenging endeavor that resulted in a rich set of capabilities and a system on which to build new functions. It operates in a complex networking environment that includes OSI, DECnet Phase IV, X.25, and TCP/IP protocols. DECnet/OSI for ULTRIX software allows OSI applications to function in TCP/IP networks. RFC 1006 supports the operation of OSI applications using TCP/IP connections, and RFC 877 allows TCP/IP to be configured over X.25. In addition, a set of gateways allows intercommunication between DECnet/OSI and TCP/IP networks.

The DECnet/OSI for ULTRIX system was also designed to be extended to include new functions as they are incorporated into the OSI standards. New protocol components can be added and used without changing existing components or network management. In addition, the software was designed to be portable. The DECnet/OSI for ULTRIX software has been ported to the DEC OSF/1 AXP operating system, and DECnet/OSI version 1.0 for DEC OSF/1 AXP was released in March 1993.

DECnet/OSI for ULTRIX demonstrates Digital's continuing commitment to provide the OSI protocol on platforms based on open systems. The ULTRIX system was the first end system to include products that followed the DNA OSI strategy. These systems can interoperate with either DECnet Phase IV systems or other OSI systems. As with DECnet Phase IV, DECnet/OSI for ULTRIX continues to provide a set of components consistent with the UNIX philosophy of networking.

10 Acknowledgments

The authors would like to thank the people, past and present, who contributed to the design and development of the DECnet/OSI for ULTRIX product. Special thanks go to members of the following teams for their dedication and hard work: DECnet-ULTRIX, ULTRIX FTAM, ULTRIX VT, OSAK, DECdns, DECdts, ULTRIX X.25, and ULTRIX Wide Area Device Drivers.

11 References

1. J. Harper, "Overview of Digital's Open Networking," Digital Technical Journal, vol. 5, no. 1 (Winter 1993, this issue).

2. Information Processing Systems-Open Systems Interconnection-Basic Reference Model, ISO 7498 (New York: American National Standards Institute, 1984).

3. S. Martin, J. McCann, and D. Oran, "Development of the VAX Distributed Name Service," Digital Technical Journal, vol. 1, no. 9 (June 1989): 9-15.

4. S. Leffler, W. Joy, and R. Fabry, "4.2BSD Networking Implementation Notes," (Berkeley, CA: University of California Technical Report, 1983).

5. R. Perlman, R. Callon, and M. Shand, "Routing Architecture," Digital Technical Journal, vol. 5, no. 1 (Winter 1993, this issue).

6. X/Open Company, Ltd., X/Open Portability Guide, Networking Services (Englewood Cliffs, NJ: Prentice Hall, 1988).

7. M. Rose and D. Cass, "Request for Comments: RFC 1006, ISO Transport Services on Top of the TCP, Version 3," May 1987.

8. M. Sylor, F. Dolan, and D. Shurtleff, "Network Management in Phase V," Digital Technical Journal, vol. 5, no. 1 (Winter 1993, this issue).

9. D. Robinson, L. Friedman, and S. Wattum, "An Implementation of the OSI Upper Layers and Applications," Digital Technical Journal, vol. 5, no. 1 (Winter 1993, this issue).

12 General References

D. Comer, Internetworking with TCP/IP: Principles, Protocols and Architecture (Englewood Cliffs, NJ: Prentice Hall, 1988).

S. Leffler, M. McKusick, M. Karels, and J. Quarterman, The Design and Implementation of the 4.3 BSD UNIX Operating System (Reading, MA: Addison-Wesley Publishing Company, May 1989).

S. Leffler, W. Joy, and R. Fabry, "A 4.2BSD Interprocess Communication Primer," (Berkeley, CA: University of California Technical Report, 1983).

13 Biographies

Kim A. Buxton Kim Buxton is a principal software engineer in the Networks and Communications Group. During the past seven years, Kim has been working on DECnet and OSI for UNIX operating systems. She is currently the project leader of the DECnet/OSI DEC OSF/1 (AXP) release. Prior to assuming the role of project leader, Kim worked on network management, session control, and transport protocols for DECnet-ULTRIX products. She has worked in the area of networks and communications since joining Digital in 1980. She earned her B.S. degree in mathematics and secondary education from the University of Lowell.

Edward J. Ferris Ed Ferris is a principal engineer in the Networks and Communications Group. During the past seven years, Ed has been working on DECnet-ULTRIX. He is currently one of the technical leaders of the DECnet /OSI DEC OSF/1 (AXP) release. Ed has primarily worked at the data link and network protocol layers. He has worked on networks and communication products since joining Digital in 1982. Ed earned a B.A. in English from the University of Massachusetts, and a B.S. in computer engineering from Boston University.

Andrew K. Nash Andrew Nash is a principal software engineer with NaC Australia and was the project leader for the ULTRIX Phase V X.25 products. He is currently technical leader for NaC Australia and has been with the group since 1988. Since joining Digital in 1980, he has worked for Educational Services and the Customer Support Centre and has been a consultant for Software Services. Andrew received a B. Sc. (Ma Sc) from the University of Adelaide and a graduate diploma in software engineering from the University of Technology, Sydney.