CONVEX Networking Concepts

First Edition

CONVEX COMPUTER CORPORATION

CONVEX Computer Corporation
P.O. Box 833851
Richardson, TX 75083-3851
(214) 497-4000

CONVEX
Networking Concepts

Order No. DSW-128

Second Edition
November 1991

CONVEX Press
Richardson, Texas
United States of America

CONVEX
Networking Concepts

Order No. DSW-128

Printed in the United States of America

Revision Information for

CONVEX
Networking Concepts

Edition	Document No.	Description
Second	710-014230-000	Released with ConvexOS and Utilities V10.0, November 1991.
First	710-000103-201	December 1989. Initial release.

Contents

Figures

Tables

Preface

CONVEX Networking Concepts serves two purposes:

- It introduces general networking concepts and terms, and provides a foundation for discussing CONVEX networking products.
- It describes each CONVEX networking product in terms of services provided, network architecture, application program interfaces, configuration and management utilities, and relevant documentation.

Typographical conventions

This section describes typographical conventions used in this book.

The following typefaces used in this book have special meanings:

`bold courier`	Identifies user input in examples.
`courier`	Identifies input and output, including: • Command and system call names • Data structures
italic	Identifies: • User-supplied variables in a command-line example • New and important terms • Titles of documents

The word "enter" in a phrase such as "enter **`ls`**" means that you type the command and then press **RETURN**.

Accessing associated documentation

Using CONVEX networking products requires more information than is included in this document. This section describes where to find additional information and assistance.

Accessing man pages

For more information on ConvexOS and CONVEX networking products, use the online man pages. To view a man page, enter:

`man` *command*

where *command* is any valid command.

References made to man pages throughout this document are in the form:

cat(1)

where the man page's section number, enclosed in parentheses, follows the command name.

Accessing paper documentation

At the end of each product-specific chapter in this book (Chapter 3 through Chapter 7), you will find a list of related documents and a summary of their contents.

You can order these books from CONVEX Computer Corporation for more in-depth information about ConvexOS or CONVEX networking products.

Ordering additional paper documentation

To order the current edition of this or any other CONVEX document, send requests to:

CONVEX Computer Corporation
Customer Service
P.O. Box 833851
Richardson, TX 75083-3851 USA

If possible, please include the order number (DSW number) or the exact title of the document you are ordering.

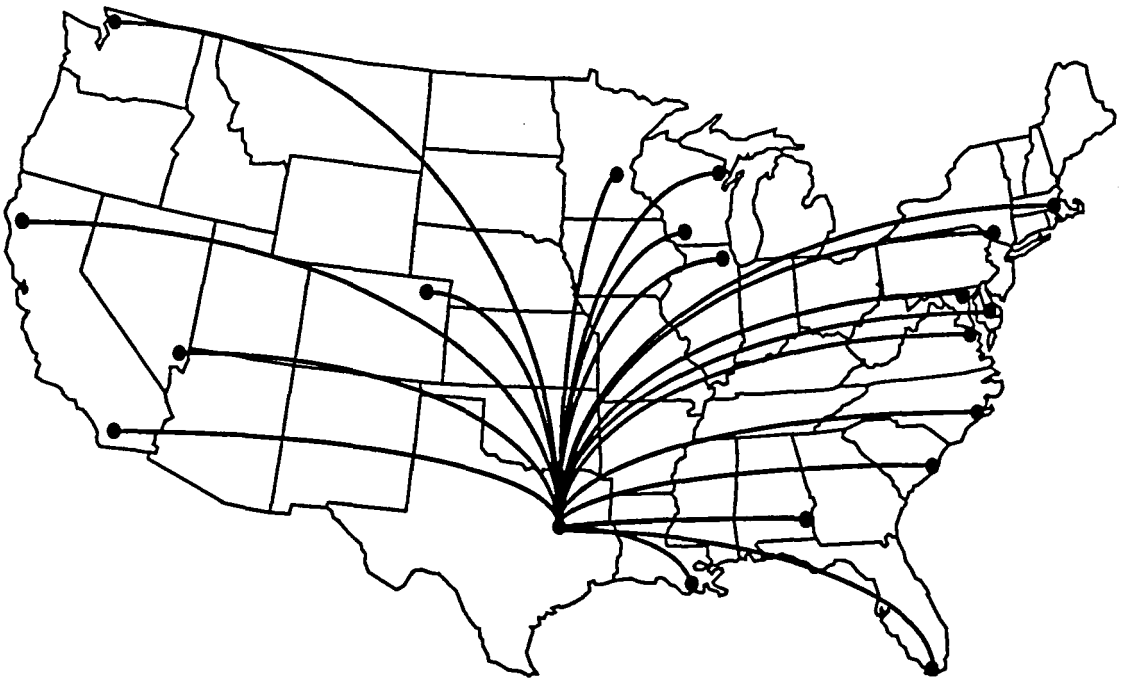
Acquiring technical assistance

If you have questions that are not answered in this book, contact the CONVEX Technical Assistance Center (TAC) at the following locations:

- Within the continental U.S., call: 1 (800) 952-0379.
- From Canada, call: 1 (800) 345-2384.
- All other locations, contact the local CONVEX office.

If you would like to report any problems you may have with ConvexOS or its associated documentation, you can also use the `contact` utility. For more information, refer to the contact(1) man page in *ConvexOS Man Pages for Users*, or "Using `contact`" in the *ConvexOS Primer* or *Managing ConvexOS: Operations Guide*.

Introduction to networking



Introduction to networking 1

This chapter introduces basic networking concepts and terminology. It discusses the following topics:

- Types of computer networks
- Components of a network
- Network topology and architecture
- Protocols
- Networking standards
- Methods used to interconnect networks
- Network services
- Application program interfaces
- Configuration and management issues

If you already know the basics of networking, you may want to skip forward to Chapter 2, "CONVEX networking products," for an overview of networking products offered by CONVEX.

Networking terminology

One of the first problems the novice to networking encounters is lack of consistent terminology. Open any three books on networking and you will find three different sets of terms. For example, one author calls networked machines "hosts," another insists on calling them "nodes," while a third uses the terms interchangeably.

Some variation in terminology simply reflects the author's preference. More often, however, terminology depends on the set of standards on which the network being discussed is based. Each of the two most widely-used sets of nonproprietary networking standards, TCP/IP and OSI, comes with its own vocabulary. The TCP/IP community favors "host" and "packet"; the OSI community prefers "node" and "data unit."

One of the goals of this book is to define a networking vocabulary and use it consistently. To this end, the book

- Uses terms found most often in off-the-shelf books on networking.
- Uses terms appropriate to the type of network being discussed. TCP/IP-based networks are described using terms preferred by the TCP/IP community and OSI-based networks are described using the vocabulary of the OSI community.
- Identifies commonly-used alternate terms. For example, this book refers to the ISO Open Systems Interconnection Model as the "OSI model," but gives "OSI Reference Model" and "ISO model" as commonly-used alternatives for describing the same seven-layer architectural paradigm.

Computer networks

A *computer network* is a system of interconnected computers that enables machines and their users to exchange information and share resources.

Networks are classified according to two characteristics:

- *Service area*—size of the geographic area over which the network provides communication.
- *Data rate*—speed at which data is transferred across the network.

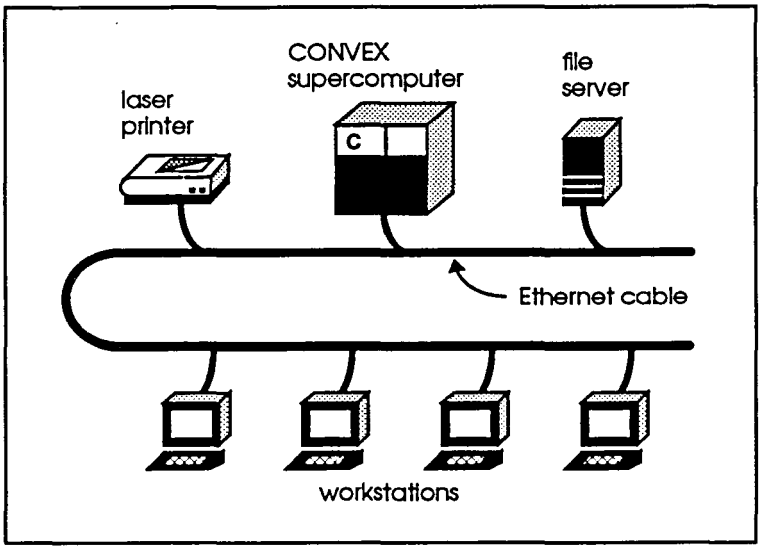
Local Area Networks

Local area networks, or *LANs*, provide high-speed communication between computers located within a small geographic area. LANs operate at data rates of 3 to 100 Mbits per second and have service areas with a radius of a few hundred meters to about 50 kilometers. Primary media used to transmit data over a LAN include twisted pair, coaxial cable, and fiber-optic cable.

Among the services provided by LANs are high-speed data transfer, distributed file systems, and electronic mail.

Figure 1 on page 5 shows a LAN that connects a CONVEX supercomputer to a file server, a laser printer, and several workstations.

Figure 1 Model of a local area network



CONVEX networking products provide LAN communication over several types of network interfaces, including Ethernet, HYPERchannel, FDDI, and UltraNet, as well as over serial lines.

Metropolitan Area Networks

The *Metropolitan Area Network* (MAN) is an emerging technology capable of providing large corporate customers with the ability to transfer massive amounts of voice and data within a service area roughly the size of a city. MANs use technology similar to LANs to provide data rates faster than wide area networks, which rely on telephone switching technology.

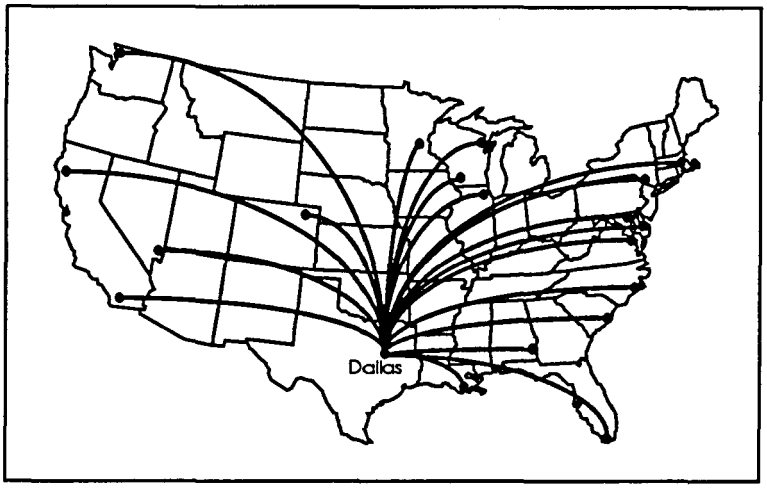
Wide Area Networks

At the most complex level, a computer network might be a worldwide system of computers joined by telephone system trunk lines or by satellite links. Such large-scale networks, called *wide area networks*, *WANs*, or *long haul networks*, are maintained by technical firms, governments, major universities, and commercial time-sharing services. WANs have slower data rates than LANs, typically in the range of 2 Kbits per second to 3 Mbits per second, and they often involve use of special packet-switching computers.

An example of a WAN is USENET, a distributed bulletin board and discussion system started at Duke University and the University of North Carolina. USENET subscribers participate in hundreds of discussion groups with other users worldwide.

Figure 2 shows a hypothetical WAN connecting machines in Dallas, Texas with machines in several major U.S. cities.

Figure 2 Model of a wide area network



Network components

The primary components of a computer network are

- Nodes and hosts
- Communication paths
- Subnets

Nodes and hosts

Nodes are computers that initiate or facilitate the flow of data across a network. Nodes that run user-level network applications are also called *hosts*. By this definition, all hosts are nodes, but not all nodes are hosts—for example, a repeater, used to boost a hardware signal, is a node, but not a host. However, you will often encounter these terms used interchangeably.

Terminals or workstations from which users request network services, and hosts that process those requests are called *end nodes* or *end systems*. *Intermediate nodes* transfer data between end nodes. They include bridges, gateways, routers, and repeaters. (Refer to the section, "Interconnecting networks," on page 19.)

All hosts and most nodes must be assigned a unique network address so that other machines on the network can direct messages to them. For more information about addresses, refer to the section, "Identifying nodes," on page 25.

Communication paths

A *communication link*, also called a *logical link* or *virtual circuit*, is a logical communication path consisting of the hardware and software needed to establish a connection and transfer data between nodes. Communication links comprise physical interfaces, controllers, network adapters, and some form of communication line. A *communication line* is a physical path—coaxial or fiber-optic cables, telephone lines, or satellite links—that connects nodes on a network.

Subnets

A single physical network often cannot meet all the needs of a large organization. For example, an engineering department might require a costly, high-speed network for transferring enormous amounts of data, while a slower, more economical network would suffice for interoffice electronic mail. Therefore, networks often comprise multiple communication lines, each of which is called a *subnet*, or *subnetwork*.

Because this is so often the case, you will find the terms *network* and *subnet* used interchangeably. Each subnet functions autonomously to provide communication between nodes directly connected to it. Refer to the section, "Interconnecting networks," on page 19, for descriptions of methods used to interconnect networks.

Network topology

Network *topology* describes the organization of a network in terms of its components, interconnections, and geography.

The network shown in Figure 1 on page 5 has a *linear*, or *bus*, topology, in which all network components connect to a common communication line—in this case, an Ethernet cable.

Other common network topologies include the *ring*, in which each node is connected to adjacent nodes to complete a circle; and the *star*, in which a central node serves as the point of connection for all other nodes. Figure 2 on page 6 shows a network with a star topology. In the network shown, nodes around the U.S. communicate with each other through the central node in Dallas.

Protocols and network architecture

Whereas network topology describes the physical organization of a network, network architecture describes its logical structure.

Protocols

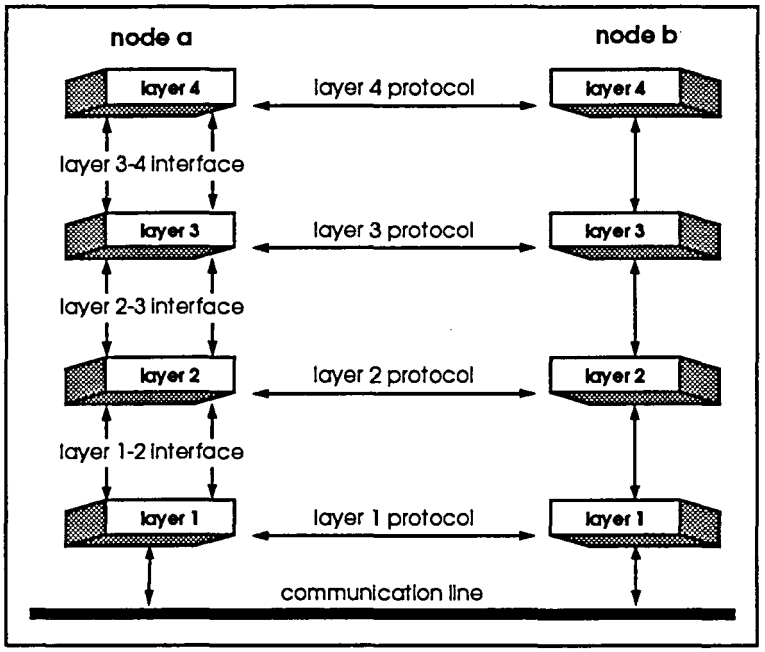
Interoperability is the ability of network components to communicate. To interoperate, components must use a common set of data and message formats, and follow a common set of procedures governing transmission. This set of formats and procedures is called a *protocol*. Protocols govern the way in which network components transmit and interpret information.

Because of the wide range of complex functions required for network communication, protocols are often designed and implemented as logical units arranged in layers. This layered approach to network design offers several advantages:

- It reduces the complexity of the overall system by breaking it into smaller, more manageable parts.
- It allows most networking software to be oblivious to the details of transmitting and receiving data over a specific type of physical medium.
- It facilitates use of standard interfaces between the layers. Therefore, a given layer of one vendor's product can be designed to interoperate with the same layer of other vendors' products.

A layered set of protocols is called a *protocol stack, protocol family,* or *protocol suite*. Each layer provides *services* to the layers above it. Figure 3 on page 9 illustrates the concept of protocol layering.

Figure 3 Protocol layering



Each layer communicates with the next through a standard interface that defines services provided by the lower layer to the one above it. Although it appears that protocols at layer *n* on one node communicate directly with protocols at layer *n* on the other node, data actually flows down one protocol stack, across the communication line, and up the other protocol stack. The data is then processed by the protocol at the corresponding layer. Protocols running at the same level in different machines are called *peer protocols*.

Network architecture

A layered set of protocols and the interfaces between the layers form a logical structure called a *network architecture*. A network architecture includes all hardware, software, protocols and the services they provide, and the structure of and interaction between components. Examples of network architectures are shown on pages 53, 63, 92, and 104.

Networking standards

Early networks were designed around proprietary architectures that could be used to connect only a specific vendor's components. As the need for network interoperability grew, so did the use of standard network architectures. Standard architectures facilitate communication between diverse networks.

Standards organizations

Standards are formalized through the work of committees. Several organizations are responsible for the development and acceptance of networking standards in use today.

American National Standards Institute (ANSI)

ANSI serves as a repository and coordinating agency for standards implemented in the U.S. Its activities include the production of Federal Information Processing Standards (FIPS) for the Department of Defense (DoD).

Comité Consultatif International de Télégraphique et Téléphonique (CCITT)

CCITT is an international organization that makes recommendations about telephone, telegraph, and data communication interfaces. Many CCITT recommendations have become international standards. You will often see CCITT Americanized into the International Telegraph & Telephone Consultative Committee.

CCITT recommendations are usually identified by a letter, followed by a period and a number, as in X.25 and X.400.

Corporation for Open Systems (COS)

COS is an organization composed of major suppliers of data processing and data communications products. It was founded to advance the use of international standards. COS has been instrumental in the development of procedures for certifying communication systems as being compliant with international standards.

Department of Defense (DoD)

The DoD developed the Defense Advanced Research Projects Agency (DARPA) Internet, or simply "the Internet," to serve as a test bed for internetworking technology. From that research came the earliest set of standard protocols used for internetworking. This set of protocols is referred to as the *TCP/IP protocol suite*, after the names of its two major protocols, Transmission Control Protocol (TCP) and Internet Protocol (IP).

TCP/IP protocols are specified in a series of technical papers called Requests for Comments (RFCs). Refer to the section, "Obtaining information about standards," on page 12.

In this book and many others, you will see the word, "internet," written two ways: all in lowercase, and with an initial capital letter. Typically, the lowercase word, "internet," refers to any interconnection of networks; "Internet" refers specifically to the DARPA Internet.

Electronics Industry Association (EIA)

EIA is a U.S. trade association concerned primarily with the development of physical-level standards. Other U.S. organizations make contributions to CCITT through the EIA. EIA standards are identified by the letters EIA, followed by a hyphen and a number, such as EIA-232, a standard used for connecting computers to modems.

European Computer Manufacturers Association (ECMA)

ECMA works closely with ISO and CCITT toward developing standards for data processing and data communication. ECMA includes all European computer manufacturers.

Institute for Electrical and Electronic Engineers (IEEE)

IEEE is an international professional organization and a member of ANSI and ISO. IEEE created Project 802, the committee that developed a set of widely-used LAN standards known as the 802 standard.

IEEE standards have names such as IEEE 802.2, which deals with Logical Link Control (LLC).

International Organization for Standardization (ISO)

Often called the International Standards Organization, ISO is closely affiliated with CCITT. ISO creates standards on a broad range of subjects. Its membership includes such U.S. national organizations as ANSI.

ISO standards are distinguished by the letters ISO, followed by a space and a number, as in ISO 8473, a protocol that provides connectionless-mode network service. (Refer to the section, "Modes of service," on page 23, for a definition of connectionless-mode service.)

ISO is responsible for the OSI model discussed in the next section, and mentioned throughout this book.

National Institute of Standards and Technology (NIST)

Formerly known as the National Bureau of Standards (NBS), NIST is responsible for defining the set of standard protocols required for systems used by U.S. government agencies. NIST activities produced the Government OSI Profile (GOSIP). (Refer to the section, "Profiles," on page 18, for more information about GOSIP.)

Japanese Promoting Conference for OSI (POSI)

POSI is active in promoting OSI standards and standards conformance testing. It membership includes major Japanese computer vendors and the Nippon Telephone and Telegraph Corporation.

Obtaining information about standards

Each product-specific chapter in this book (Chapter 3 through Chapter 7) includes a list of the RFCs, ISO, and other specifications to which the product conforms. Several organizations provide facilities that enable you to obtain copies of these and other specifications over the network. This section provides the network addresses of a few of these organizations and gives instructions for obtaining copies of the standards.

In the following screen examples, enter text shown in boldface type, including the user name and password, exactly as it appears. Because you must have the correct password to access the hosts listed, the password is shown in each example; however, it does not appear on the screen as you type it. Fields shown in italics represent values and file names that you choose.

NIC.DDN.MIL

Transfer copies of RFCs from network address, NIC.DDN.MIL, by using the anonymous `ftp` facility as shown in Figure 4.

Figure 4 Transferring RFCs from NIC.DDN.MIL

```
%ftp NIC.DDN.MIL
Connected to NIC.DDN.MIL.
220 nic FTP server (SunOS 4.1) ready.
Name (NIC.DDN.MIL:yourname): anonymous
331 Guest login ok, send ident as password.
Password:guest
230 Guest login ok, access restrictions apply.
ftp> get rfc/rfcnnnn.txt local.file
200 PORT command successful.
150 ASCII data connection for rfc/rfcnnnn.txt
    (128.150.60.1,3229) (22553 bytes).
22553 bytes received in 9.6e+02 seconds (0.013
    Kbytes/s)
ftp> quit
%
```

In this example, *nnnn* specifies the RFC number. RFCs are available in text format (suffixed by **`.txt`**) or Postscript format (suffixed by **`.ps`**).

FTP.NISC.SRI.COM

Transfer RFCs from network address, NISC.SRI.COM, by using the anonymous `ftp` facility as shown in Figure 5 on page 14.

Figure 5 Transferring RFCs from NISC.SRI.COM

```
%ftp FTP.NISC.SRI.COM
Connected to phoebus.NISC.SRI.COM.
220 phoebus FTP server (SRI Version 1.98 Fri Apr
    19 11:57:54 PDT 1991) ready.
Name (FTP.NISC.SRI.COM:yourname): anonymous
331 Guest login ok, send ident as password.
Password: guest
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> get rfc/rfcnnnn local.file
200 PORT command successful.
150 Opening BINARY mode data connection for rfc/r-
    fcnnnn.txt (20972 bytes).
226 Transfer complete.
20972 bytes received in 3.7 seconds (5.5 Kbytes/s)
ftp> quit
221 Be Excellent to one another!
%
```

In this example, *nnnn* specifies the RFC number. RFCs are available in text format (no suffix) or Postscript format (suffixed by **.ps**). To obtain the RFC index, specify **rfc/rfc-index.txt** as the remote path name.

If your site cannot use `ftp`, NISC.SRI.COM also provides an automatic mail service. To use this service, email your request to **MAIL-SERVER@NISC.SRI.COM**. In your request, specify the RFCs you need by entering **send rfc***nnnn* (text file) or **send rfc***nnnn***.ps** (Postscript file), where *nnnn* is the RFC number. You can request multiple RFCs by listing each `send` command on a separate line.

NISC.JVNC.NET

Transfer RFCs from network address, NISC.JVNC.NET, by using the anonymous `ftp` facility as shown in Figure 6 on page 15.

Figure 6 Transferring RFCs from NISC.JVNC.NET

```
%ftp NISC.JVNC.NET
Connected to NISC.JVNC.NET.
220 nisc.jvnc.net FTP server (SunOS 4.1) ready.
Name (NISC.JVNC.NET:yourname): anonymous
331 Guest login ok, send ident as password.
Password: guest
230 Guest login ok, access restrictions apply.
ftp> get rfc/rfcnnnnTXT.v local.file
200 PORT command successful.
150 Opening BINARY mode data connection for rfc/r-
    fcnnnn.txt (20972 bytes).
226 Transfer complete.
20972 bytes received in 3.7 seconds (5.5 Kbytes/s)
ftp> quit
221 Goodbye.
%
```

In this example, *nnnn* specifies the RFC number and *v* specifies the version number. RFCs are available from NISC.JVNC.NET only in text format.

If your site cannot use `ftp`, NISC.JVNC.NET also provides a mail service. To use this service, email your request to **SENDRFC@JVNC.NET**. On the subject line, specify the RFC number, as in `Subject: RFC`*nnnn*, where *nnnn* is the RFC number. No body text is needed.

bruno.cs.colorado.edu and digital.resource.org

You can obtain copies of recommendations specified in the 1988 and 1992 CCITT Blue Books by using anonymous `ftp` to either bruno.cs.colorado.edu or digital.resource.org. CCITT recommendations are stored in the /pub/standards/ccitt/1988 or /pub/standards/ccitt/1992 directories. The file, blue.book.org, explains the naming scheme used for these recommendations.

You can request these recommendations in one of three formats— troff, ASCII, and Postscript—by using the suffixes, **.troff**, **.txt**, or **.ps**, respectively.

Figure 7 on page 16 shows a sequence of commands used to transfer a copy of an X.25 specification. This example includes a few commands used to search for the desired specification. These steps are unnecessary, of course, if you already know the name of the specification you want.

Figure 7 Transferring ISO specifications from bruno.cs.colorado.edu

```
%ftp bruno.cs.colorado.edu
Connected to bruno.cs.colorado.edu.
220 bruno FTP server (SunOS 4.1) ready.
Name (bruno.cs.colorado.edu:yourname): anonymous
331 Guest login ok, send ident as password.
Password: guest
230-Guest login ok, access restrictions apply.
This server is courtesy of Sun Microsystems, Inc.
Note to European users: src.doc.ic.ac.uk (aka
nic.ja.net) has a copy of the CCITT documents in
the directory doc/ccitt-standards.
ftp> cd /pub/standards/ccitt
250 CWD command successful.
ftp> ls
200 PORT command successful.
150 ASCII data connection for /bin/ls
   (130.168.73.3,3055) (0 bytes).
total 448
drwxrwxr-x 5 2764 rd 512 Oct 8 02:57 1988
drwxrwxr-x 25 2764 rd 512 Sep 15 16:44 1992
226 ASCII Transfer complete.
ftp> cd 1992
250 CWD command successful.
ftp> ls *25*
200 PORT command successful.
150 ASCII data connection for /bin/ls
   (130.168.73.3,3092) (0 bytes).
-rw-rw-r-- 1 2764 rd 118442 Sep 13 15:33 x25_1.doc
-rw-rw-r-- 1 2764 rd 102218 Sep 13 15:33 x25_2.asc
-rw-rw-r-- 1 2764 rd 26776 Sep 13 15:33 x25_i.asc
226 ASCII Transfer complete.
ftp> get x25_i.asc local_file
200 PORT command successful.
150 ASCII data connection for x25_i.asc
   (130.168.73.3,3093) (26776 bytes).
226 ASCII Transfer complete.
27557 bytes received in 5.9 seconds (4.5 Kbytes/s)
ftp> quit
221 Goodbye.
%
```

If your site cannot use `ftp`, email your request to **infoserv@bruno.cs.colorado.edu** or **infoserv@digital.resource.org**. To get help with this service, include the word, **HELP**, in the body of your message. You will receive instructions on how to use the service, as well as information about the documents maintained by the server. To retrieve the index of documents available through the server, include the words, **SEND INDEX**, in your mail.

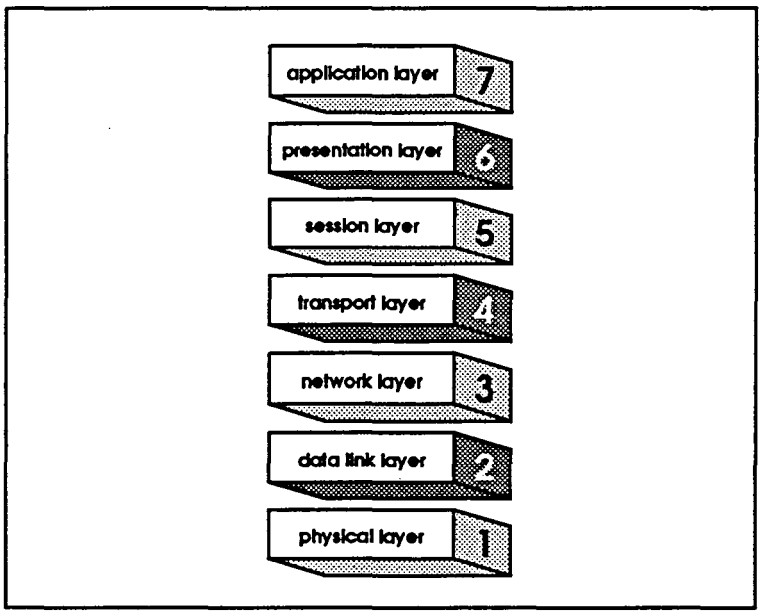
The OSI model

In the late 1970s, several organizations began work on defining a common architectural framework on which to base international standards. In 1984, ISO published the results: a set of specifications for network architecture called the *Open Systems Interconnection (OSI) Reference Model*, or *OSI model*. This model is also known as the ISO Reference Model. The OSI model is widely accepted today as the international standard for network architecture.

OSI standards are specified in two parts: a definition of the service to be provided at a given layer, and a definition of the protocol that provides the service. (Refer to the section, "Network services," on page 22, for more information.)

Figure 8 shows the seven-layered architecture of the OSI model.

Figure 8 The OSI model



The following section describes the functions performed at each layer:

- **Physical layer**—Responsible for transmitting data bits over a specific physical medium. Physical layer protocols include EIA-232 and V.35.
- **Data link layer**—Responsible for transmitting data over a communication link. Error detection, correction, and

recovery are handled at this layer. The data link layer is often divided into two sublayers: Medium Access Control (MAC) and Logical Link Control (LLC).

- **Network layer**—Responsible for routing and relaying data from one node to another on the same network or across multiple networks.
- **Transport layer**—Provides a reliable end-to-end data transfer service that shields upper layers from the details of the underlying network. It is responsible for ordered delivery of data, flow control, and error recovery.
- **Session layer**—Establishes, manages, and terminates a period of communication between two end users. It is also responsible for synchronizing the exchange of data and controlling traffic over the connection.
- **Presentation layer**—Concerned with the syntax of transmitted information. It is responsible for the order and format of data, and for services such as data encryption.
- **Application layer**—Provides system-independent services for end-user applications, such as electronic mail and file transfers.

Systems that conform to any nonproprietary standards are called *open systems*. In recent years, however, the term has come to mean only those systems that use the international standards for network architecture, as specified by the OSI model.

Subsequent chapters in this book use the OSI model as the framework for discussing CONVEX networking products.

Profiles

Systems that conform to ISO standards should be completely interoperable; however, most standards include various options that a particular vendor may choose to implement or not, depending on such factors as how the option affects performance. As a consequence, systems built around the same standards may not interoperate.

To ensure interoperability between systems, groups of organizations agree upon a common subset of standards called a *profile*. A profile identifies the set of services that members of the group must implement to ensure interoperability with other members' systems. The U.S. Government OSI Profile (GOSIP), defined by the National Institute of Standards and Technology (NIST), is one such profile. GOSIP identifies a set of standard OSI protocols with which networking systems used by U.S. government agencies must conform.

Each product-specific chapter in this book (Chapter 3 through Chapter 7) includes a list of profiles to which the product conforms.

Interconnecting networks

Local and wide area networks can operate as self-contained entities or they can be interconnected to form what is called an *internetwork*, or *internet*. Internetworking extends the service area of a network, permitting machines of differing types, as well as greater numbers, to participate. Though composed of diverse physical networks, an internet operates as a single, virtual network.

The benefit of internetworking is that it allows an installation to select the networking equipment and functionality most suited to its own needs. The burden is that it requires each computer on the internet to conform to standard protocol specifications.

Of course, internetworking requires more than conformance with a set of standards; there must also be some physical connection between networks. Several types of components are used to interconnect networks:

- gateways
- bridges
- routers
- repeaters

Gateways

You will sometimes see the terms *gateway* and *bridge* used interchangeably; however, though both of these components interconnect subnets, they do so in different ways. Therefore, a distinction between them is needed.

Gateways are used primarily to interconnect different types of networks. For example, a gateway can be used to interconnect CONVEX Internet Services (TCP/IP) networks and IBM Systems Network Architecture (SNA) networks. CONVEX COVUEnet provides a gateway for transferring electronic mail between VAX and CONVEX nodes. A gateway is shown in the top half of Figure 9 on page 20.

Gateways may be either special-purpose packet-switching computers or nodes that transfer packets across networks in addition to serving as general purpose hosts. Gateways that interconnect networks with incompatible architectures, protocols, and addressing schemes function at all seven layers of the OSI model. Because gateways are nodes on both systems they interconnect, they are required to have two network addresses.

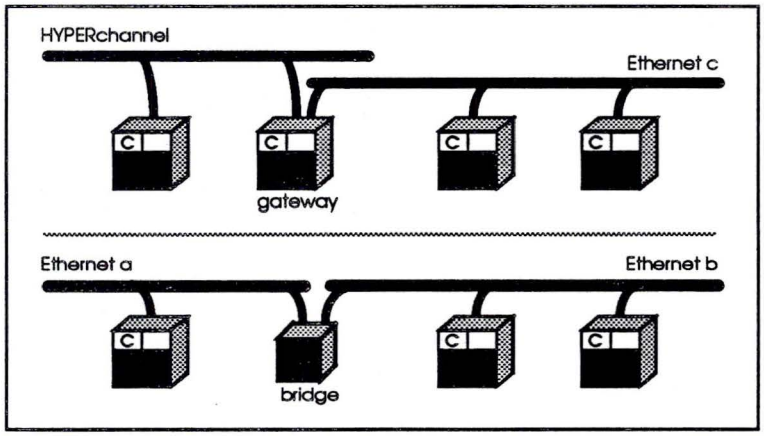
Bridges

Like gateways, bridges interconnect autonomous networks. Unlike gateways, however, bridges usually interconnect networks of the same type. For example, a bridge could be used to connect Ethernet cables so that they appear to network software as a single physical entity.

Bridges operate at the physical and data link layers, and are transparent to network software. A bridge examines the source and destination addresses of each packet that passes by on the network. When it detects that a packet's destination host resides on a different network than its source host, the bridge forwards the packet to the destination host's network.

Figure 9 illustrates the difference between gateways and bridges, and the purpose each serves in connecting networks.

Figure 9 The roles of gateways and bridges



The gateway shown in Figure 9 is a CONVEX host; the bridge is simply a "black box" for transmitting data from Ethernet a to Ethernet b. Because the bridge is transparent to network software, hosts on Ethernet a and Ethernet b view the multicable LAN as a single physical network. Hosts on Ethernet c, however, must use different network numbers to reach hosts on the HYPERchannel or the Ethernet network.

Routers

A router operates as an intermediate node whose purpose is to direct messages to the appropriate network. Routers may be either special-purpose packet-switching computers or nodes that transfer packets across networks in addition to serving as general purpose hosts. Routers use the destination address included in a packet to determine where to send it. If more than one route to the destination exists, the router tries to choose the most efficient one.

Routers operate at the network layer. For packets to be routed from one network to another, the protocols at and above the network layer must be compatible.

Repeaters

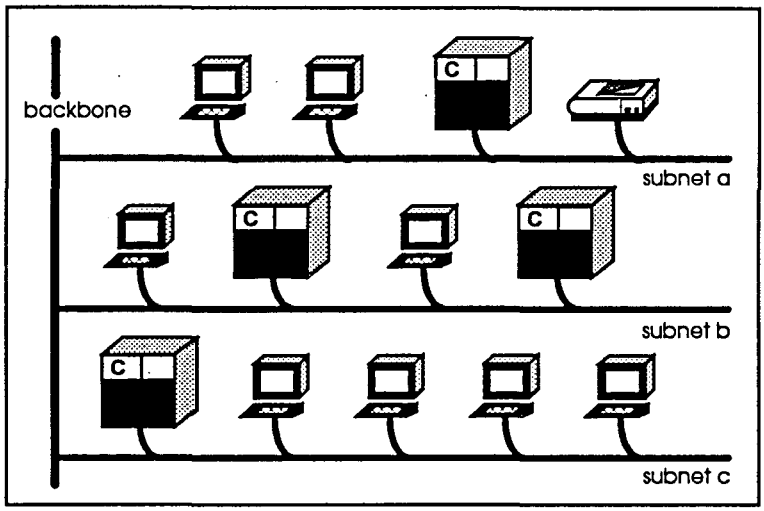
Unlike gateways, bridges, and routers, repeaters are not used to interconnect different networks. Instead, they amplify the electrical signal on a segment of communication line, effectively extending the network beyond the limitations of its physical media. For example, a typical Ethernet cable can maintain a signal over about 500 meters. Repeaters used between Ethernet segments boost the signal so that it can be sent over a series of Ethernets, which extends the service area of the network.

Because repeaters operate at the physical layer, they do not interpret data; they simply amplify it and pass it along. Segments interconnected by repeaters must be of the same type, that is, Ethernet to Ethernet, and they must run identical protocols.

Backbones

Though not technically an internetworking device, another method for interconnecting networks (or subnets) is through a central network, called a *backbone*. Hosts are not usually connected directly to the backbone. Instead, they connect to subnets attached to the backbone, as shown in Figure 10 on page 22.

Figure 10 Subnets connected by a backbone network



The figure shows three subnets (a, b, and c) interconnected through a backbone network. In this example, subnets are directly tapped into the backbone network. Another possibility is to use bridges to attach subnets to the backbone.

Using a backbone network to interconnect subnets extends the total service area of the LAN. Furthermore, because each subnet functions autonomously, using a backbone prevents loss of the entire network if one or more subnets fail.

Network services

In the context of networking, a *service* is a function or set of functions provided by a network *entity*. A network entity is the component that either provides or uses the service offered at a given protocol layer. Software entities are called *processes*. *Peer processes*, also called *peer entities*, or, in OSI terminology, *peer transport users*, are processes running at the same protocol level on different machines. Figure 3 on page 9 illustrates this concept. Peer processes running at layer *n* in one host communicate with peer processes running at layer *n* in the other host through the layer *n* protocol.

A service can be any function performed by a protocol for the benefit of higher layer protocols. The term also refers to the service provided by an application program to the end user, such as a remote login utility.

Client/server model

A commonly-used structure for providing virtual communication between peer processes is the *client/server model*. A pair of processes typically implements a service. A *client* process running on one host makes a request to a *server* process, or *service provider* running on another host. The server fulfills the request and transmits a response back to the client. Any number of clients can request a particular service on a given host. For example, a remote login server can fulfill requests for remote sessions from any number of client machines.

Network servers are often implemented as processes that run continuously, servicing requests as they are received. An example of a network process is `inetd`, the so-called internet superserver. To minimize the number of networking processes running on the system, CONVEX Internet Services uses a single daemon, `inetd`, to service multiple network requests. `inetd` starts automatically at boot time. Once started, it monitors a set of ports associated with each service defined in its configuration file. When `inetd` receives a connection request on one of its ports, it activates the appropriate process to service the request.

Modes of service

Services are provided in one of two communication modes: *connectionless* or *connection-oriented*.

Connection-oriented communication requires that a virtual connection between peer processes be established before data can be transferred.

In connectionless communication, data is exchanged in segments, sometimes called *datagrams*. Each datagram includes the address of the source and destination nodes. Because no continuous connection exists between the source and destination nodes in connectionless-mode communication, there is no guarantee that datagrams will arrive at the destination.

Application-level services

Services that enable people or application programs to take advantage of network facilities are called *end-user* or *application-level services*. Such services include

- Electronic mail
- Remote file access and transfer
- Remote logins
- Distributed file systems

Each product-specific chapter in this book (Chapter 3 through Chapter 7) includes a description of the application-level services the product provides.

Application program interfaces

An *Application Program Interface* (API) is a facility that enables programmers to implement network applications. CONVEX networking software includes application program interfaces that give programmers access to the full functionality of the underlying protocols.

For descriptions of the application program interfaces provided by CONVEX networking products, refer to the section, "Application program interfaces," in Chapter 2, and the section, "Application program interface," in Chapter 3 through Chapter 7.

Network administration

Because computer networks are complex, so is the task of administering one. Each CONVEX networking product includes a set of utilities used to configure, manage, and monitor the network. These utilities are discussed in the section, "Configuration and management" in Chapter 3 through Chapter 7.

Though the details of network administration differ among these products, they all require attention to a few general administration issues:

- Integrating and customizing network interfaces
- Identifying nodes and networks
- Controlling access to the network

Integrating and customizing network interfaces

When you add network hardware to your system, you must integrate it into ConvexOS and customize the operating system for the interface's particular characteristics. Although this is primarily a hardware task, networking utilities and tunable parameters facilitate the addition of network devices.

In the case of CONVEX OSI WAN Transport, tunable parameters are also used to customize and fine-tune network software.

Identifying nodes

For nodes to communicate with one another, each must have a unique identifier and a way to identify other nodes on the network. There are three different ways to identify nodes:

- **Network addresses**—Numeric, universal identifiers assigned to nodes on a network.
- **Names**—Character strings assigned to network addresses. While network software works more efficiently with numeric identifiers, users find such numbers—which are often quite long—cumbersome and hard to remember. When names are associated with network addresses, users need only remember the names.
- **Physical addresses**—A device-dependent numeric address. Identifying nodes by both network addresses and physical addresses allows most network software to be written in a device-independent fashion. Only protocols at the physical layer need to identify nodes in a way that takes the type of physical medium into account.

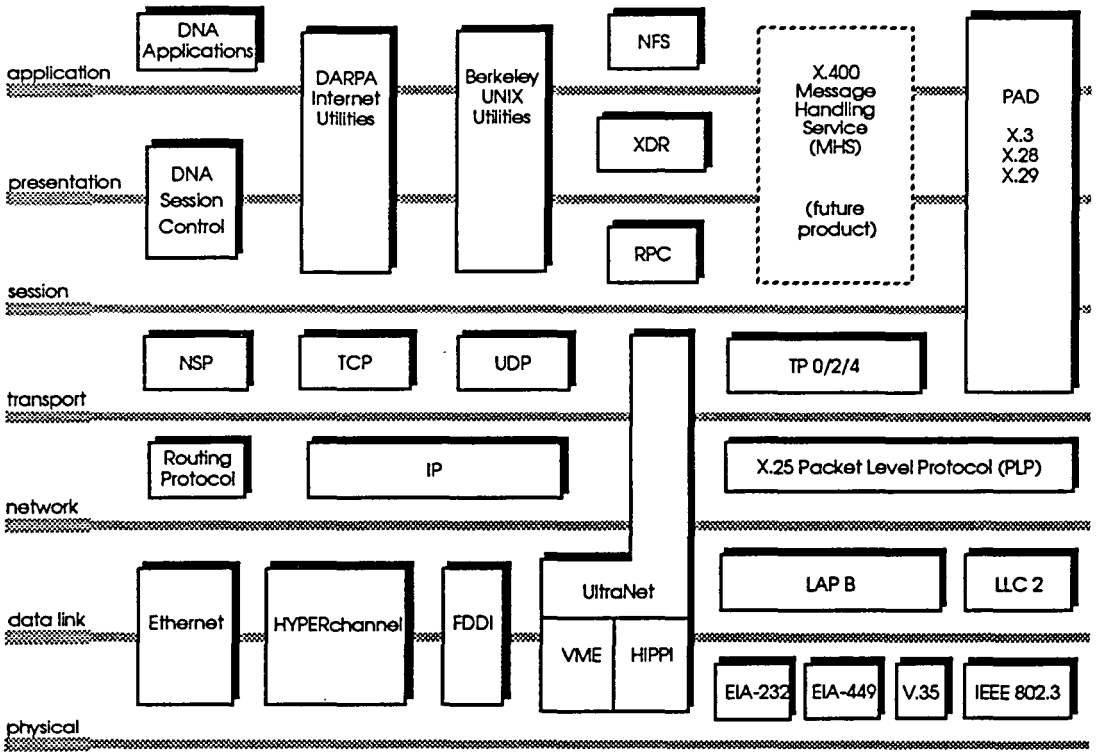
Network software includes methods for converting names to network addresses, and network addresses to physical addresses. Such conversion is called *address resolution*.

Controlling access to the network

Each node on the network must be able to accept or refuse access by remote users. Controlling access to the network requires enabling access by authorized users and preventing access by unauthorized users. For example, the network administrator can specify whether users logging in from other machines must supply a password, or even if logins are permitted on a machine.

CONVEX networking products include various mechanisms for ensuring that only authorized users have access to the network. Access can be permitted or restricted on both a machine and a user basis.

CONVEX
networking products



CONVEX networking products

2

Open supercomputing

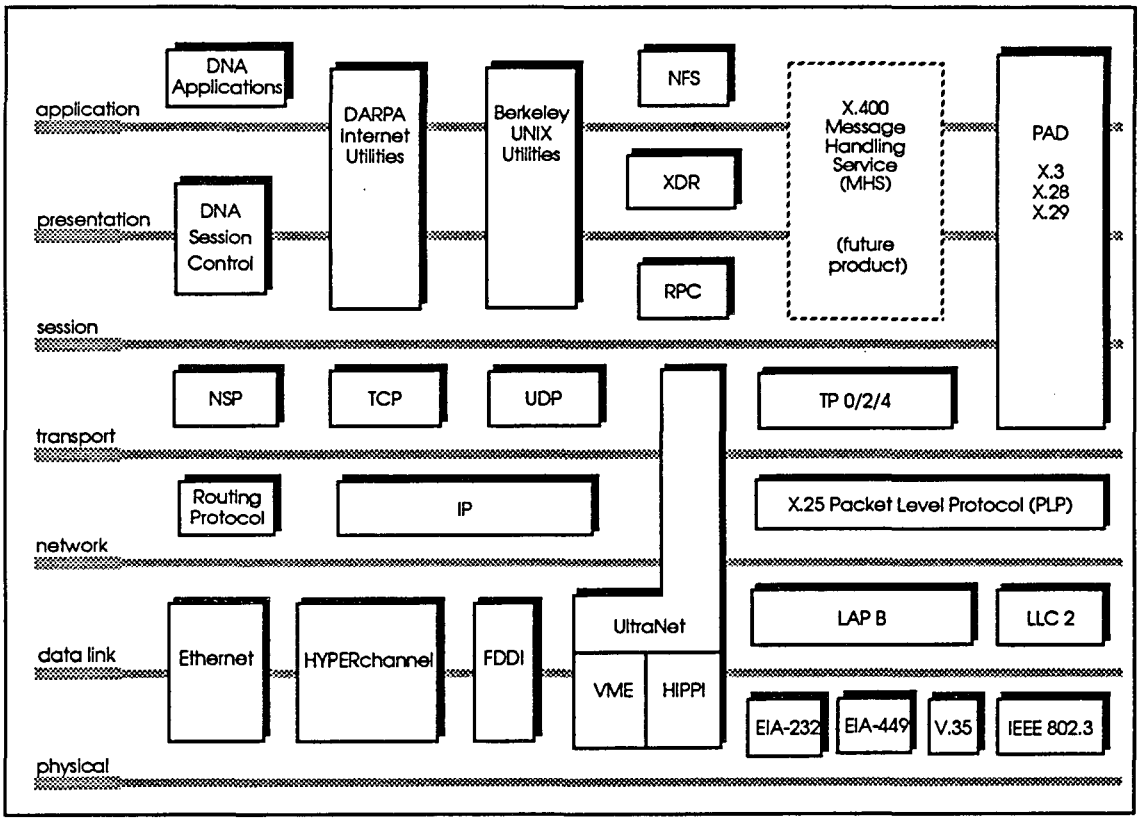
CONVEX introduced *Open Supercomputing* to provide a high-performance environment for scientific computing. By adhering to industry standards and familiar user interfaces, CONVEX provides networking solutions at every level. Here are a few ways in which CONVEX networking products meet a wide range of needs:

- You can connect CONVEX supercomputers to other large-scale systems with high-speed links such as UltraNet, FDDI, and HYPERchannel.
- Through a Systems Network Architecture (SNA) gateway product available from OpenConnect, you can connect CONVEX systems to IBM mainframes.
- You can connect CONVEX systems to minicomputers and workstations over Ethernet.
- You can connect to wide area public data networks through the CONVEX OSI WAN Transport product.
- Open Supercomputing supports both TCP/IP and DECnet protocols over Ethernet. With CONVEX COVUEnet, a CONVEX system can become a transparent addition to a VMS Phase IV DECnet while providing supercomputer performance.
- You can connect virtually all workstations, X terminals, personal computers, and desktop CRTs to CONVEX supercomputers.

In maintaining CONVEX's commitment to open supercomputing, CONVEX networking products are designed around the two most prevalent U.S. and international standards, TCP/IP and OSI, as well as widely-used proprietary architectures, such as Digital Network Architecture (DNA). Because CONVEX networking products adhere to standard architectures, they interoperate with a wide range of networking systems.

Figure 11 shows CONVEX networking products mapped to the OSI model.

Figure 11 CONVEX networking products mapped to the OSI Model



Product descriptions

CONVEX offers the following networking products:

- CONVEX Internet Services
- CONVEX Network File System (NFS)
- CONVEX COVUEnet
- CONVEX UltraNet Interface
- CONVEX OSI WAN Transport
- CONVEX Fiber Distributed Data Interface (FDDI)

A summary of each product follows.

With the exception of CONVEX FDDI, which includes only the FDDI hardware and driver, each product includes a substantial amount of network software, such as application program interfaces and configuration and management utilities. For detailed information about this network software, refer to Chapter 3 through Chapter 7.

CONVEX Internet Services

CONVEX Internet Services is a set of networking utilities that enables a user to log in to remote systems, execute commands on different machines, transfer files from one machine to another, and perform many other useful network functions. Support for DARPA Internet TCP/IP protocols assures interoperability with other CONVEX networking products as well as with a wide range of other systems.

Internet Services runs over Ethernet, HYPERchannel, FDDI, and UltraNet interfaces, and over serial lines. Internet Services includes drivers for Ethernet and HYPERchannel interfaces, and for serial lines. FDDI and UltraNet drivers are available as separate products.

CONVEX Network File System (NFS)

CONVEX Network File System (NFS) provides transparent access to networked files. NFS links together different types of systems—including personal computers, workstations, supercomputers, and mainframes—to share resources and files over local-area and wide-area networks. Because CONVEX NFS implements standard protocols and runs over TCP/IP, it interoperates with a wide range of machines and operating systems.

NFS runs over Ethernet, HYPERchannel, FDDI, and UltraNet interfaces, and over serial lines. Among the services provided by NFS are the Network Information Service (NIS), a distributed network lookup service that eases the job of administering networked machines, and NETdisk, a file server for diskless workstations.

CONVEX COVUEnet

CONVEX COVUEnet enables CONVEX systems to participate as end nodes on a DECnet-COVUEnet network. All network functions and capabilities, whether standard Digital Network Architecture (DNA) capabilities or user-written applications, are available with another DECnet or COVUEnet node that provides the complementary capabilities. For example, a COVUEnet system can use the network virtual terminal client when communicating with remote nodes that provide a network virtual terminal server.

CONVEX systems running COVUEnet may be attached to Ethernet LANs and share information with systems running DECnet. Information sharing can take the form of remote logins, deletion, printing, renaming, transferring, and batch execution of files between systems, and remote directory listings. COVUEnet also allows you to transfer mail to and from CONVEX systems and any other system running DECnet.

CONVEX UltraNet Interfaces

CONVEX offers two UltraNet interfaces: CONVEX VME/UltraNet Interface and CONVEX HIPPI/UltraNet Interface. Both products include hardware and software to allow CONVEX users to access an UltraNet high-speed network. UltraNet provides the networking performance and speed required by high-performance computing applications. The UltraNet network supports data rates of up to one gigabit per second, making it the highest performance LAN in the industry.

In addition to providing a reliable, high-performance means of transferring data between networked machines, the CONVEX UltraNet Interface includes support for TCP/IP protocols, allowing it to interoperate with existing network application programs and program interfaces.

CONVEX OSI WAN Transport

The CONVEX OSI WAN Transport product allows CONVEX supercomputers to connect to X.25 Packet Switched Data Networks that comply with CCITT and ISO recommended standards. The WAN transport can be used with both wide area and local area networks. By using the OSI WAN Transport, CONVEX users and application programs can communicate with remote

users and remote systems throughout the world. Interactive users of the CONVEX system can also access remote systems using X.3, X.28, and X.29 Packet Assembler/Disassembler (PAD) emulation features, and transfer files using UNIX-to-UNIX Copy (UUCP) facilities.

Hardware interfaces

Hardware interfaces supported by CONVEX software provide a wide range of performance characteristics that meet a wide range of needs. Each interface is described below.

Ethernet interface

CONVEX computers interface to an Ethernet network through Excelan Multibus and VMEbus Ethernet controllers. These controllers support the attachment of both regular and thin-wire Ethernet transceivers. Ethernet networks have a peak data rate of 10 megabits per second over segments of up to 500 meters in length.

The Ethernet interface is compatible with IEEE 802.3 specifications. It supports TCP/IP, DECnet, and OSI WAN protocols to provide the full functionality available with CONVEX Internet Services, CONVEX Network File System (NFS), CONVEX COVUEnet, and CONVEX OSI WAN Transport software.

Fiber Distributed Data Interface (FDDI)

CONVEX FDDI is a VME controller that connects CONVEX computers directly to FDDI networks. FDDI offers the next level of LAN performance beyond Ethernet, with a peak data rate of 100 megabits per second. FDDI also covers a greater service area than Ethernet, allowing up to 2 kilometers distance between nodes.

The FDDI standard specifies a fiber transmission medium and a token ring topology. CONVEX FDDI supports a Class A FDDI station that is dual attached and connects to both the primary and secondary rings of the network. The secondary ring provides redundancy in case of a failure of the primary ring. Dual ring support provides higher reliability and availability of the FDDI network.

CONVEX FDDI complies with ANSI X3T9.5 and ISO 9314 specifications. CONVEX FDDI supports TCP/IP protocols and all functionality available with CONVEX Internet Services and CONVEX NFS.

HYPERchannel interface

The HYPERchannel interface provides high-speed connections between CONVEX computers and other large-scale systems. CONVEX machines can be connected to HYPERchannel networks by using either Network Systems Corporation A400 HYPERchannel Adapters through the IKON 10077-NSC Multibus Interface or Network Systems Corporation NB400 HYPERchannel Adapters through the IKON 10090 VMEbus Interface.

CONVEX HYPERchannel software complies with RFC 1044, "Internet Protocol on Network System's HYPERchannel: Protocol Specification." It supports TCP/IP protocols and all functionality available with CONVEX Internet Services and CONVEX NFS.

UltraNet Interface

The CONVEX UltraNet Interface provides a high-speed connection to an UltraNet hub. A product of Ultra Network Technologies, Inc., UltraNet provides the networking performance and speed required by high-performance computing applications. The UltraNet network supports data rates of up to one gigabit per second, making it the highest performance LAN in the industry.

UltraNet Interface software supports TCP/IP protocols in addition to UltraNet proprietary protocols to enable hosts on the UltraNet to participate in a TCP/IP internet. In addition to providing all functionality available with CONVEX Internet Services, UltraNet interface software include supports high-speed file transfers and CONVEX NFS.

The UltraNet Interface allows the use of fiber optic cable for long-distance connections and coaxial cable for lower-cost short-distance connections. UltraNet Interfaces can be connected to the UltraNet hub via VMEbus or High Performance Parallel Interface (HIPPI).

VME/UltraNet

CONVEX VME/UltraNet hardware includes a VME/UltraNet controller board, the Data Link Interface (DLI), and a shielded ribbon cable that connects the controller to the DLI. The VME/UltraNet Interface complies with IEEE 802.2 LLC and TP4.

HIPPI/UltraNet

The High Performance Parallel Interface (HIPPI) is the fastest industry standard for connecting high-performance computers. HIPPI hardware consists of HIPPI channel control unit (CCU) that supports dual simplex connections (one input, one output) to provide a data rate of 800 megabits per second over distances up to 25 meters. The CONVEX HIPPI/UltraNet Interface complies with the ANSI X3T9.3 specification.

Application program interfaces

CONVEX networking products provide the application program interface through a set of Interprocess Communication (IPC) library routines and system calls. This set of system calls gives application programmers access to the full power and functionality of underlying protocols. IPC facilities enable programs to communicate with other programs running on the same or on different machines.

IPC programming is the development of programs that use IPC facilities to communicate with one another. Generally speaking, IPC facilities are used for one of two reasons:

- You must use IPC facilities to build an application that involves more than one process. Examples of this type of application are distributed systems and multiuser programs such as `talk`.
- You must use IPC facilities to communicate with a system whose interface is an IPC mechanism. Examples of this type of application are piping data through a filter, or connecting to a server on a network.

ConvexOS facilitates and manages communication between processes. IPC programs interact with the operating system in much the same way as other programs—through library routines and system calls.

Basic IPC facilities are provided by pipes, socket pairs, sockets, and STREAMS.

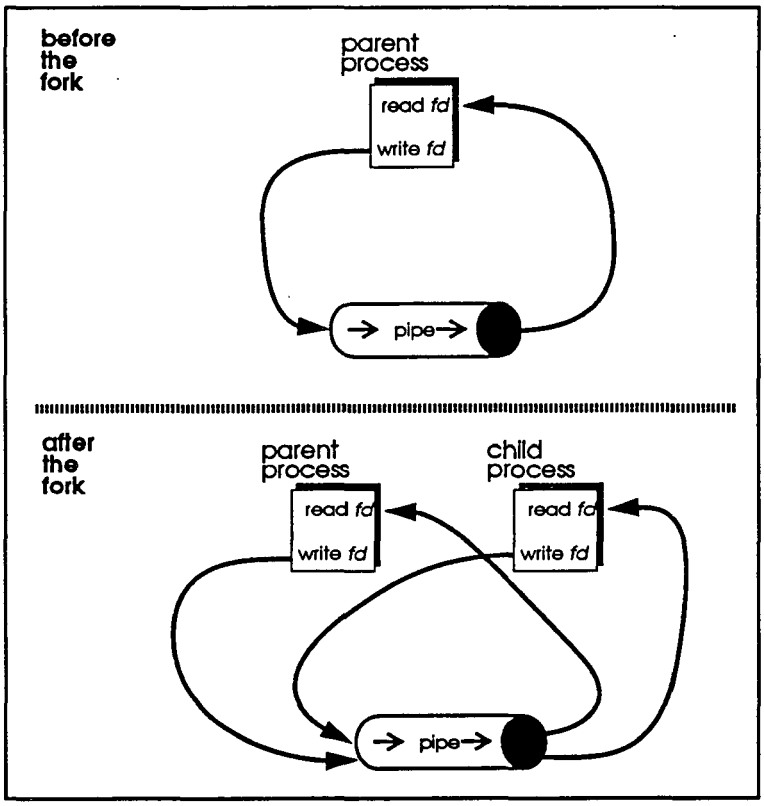
Pipes and socket pairs

A *pipe* is simply a pair of file descriptors that are initially owned by a single process. If this process passes the descriptors to another process, the two processes can communicate through the pipe. Application programs use any of three types of pipes: unidirectional pipes, socket pairs, and named pipes.

Unidirectional pipes

The simplest pipes are unidirectional: data is written at one end and read from the other. Processes that share a unidirectional pipe must have a common ancestor. In Figure 12, the parent process creates a pipe, then forks a child process. When the process forks, the parent's descriptor table is copied into the child's descriptor table.

Figure 12 Sharing a pipe between processes

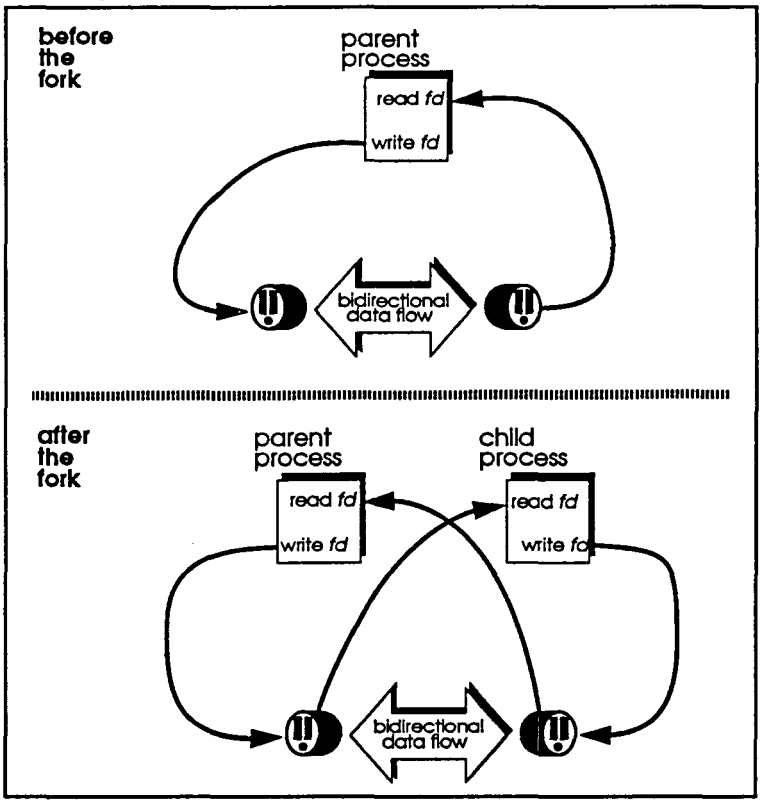


Before the fork, the parent process' descriptor table points to both ends of the pipe. After the fork, both parent's and child's descriptor tables point to the pipe. The child process then uses the pipe to send a message to the parent process. Because a pipe is a one-way communication mechanism, parent and child must agree on whether to turn the pipe from parent to child or vice versa.

Socket pairs

Bidirectional pipes, or *socket pairs*, enable you to set up two-way communication between processes. You can obtain a pair of connected sockets for two-way communication by calling the `socketpair` routine. Socket pairs have only been implemented for the UNIX domain. (A *domain* is an address space shared by sockets. ConvexOS supports two domains: the "UNIX" domain, used for communication within the local system; and the "Internet" domain, used for communication between hosts on a network.) The UNIX domain only allows communication between sockets on the same machine. Therefore, you cannot use socket pairs for network applications. Figure 13 shows a bidirectional flow of data through a socket pair.

Figure 13 Sharing a socket pair between processes



Named pipes

Named pipes exist permanently in the file system with directory entries and path names. Because you can access these pipes by name, you can use them for a variety of applications that you cannot accomplish with ordinary pipes. Typically, named pipes are used to allow a number of processes to communicate with a daemon.

Named pipes can only be used locally. Even if the named pipe exists as an inode on a remote machine, data written to the pipe via NFS is only accessible to processes on the same machine. Therefore, even though multiple clients mount the same file system, they have separate streams associated with any named pipe in that file system.

You create a named pipe using either mknod(2) or mknod(8). Once you have created the pipe, you can use the `open`, `read`, and `write` system calls. For more information, refer to the mknod(2), mknod(8), open(2), read(2), and write(2) man pages.

Sockets

Sockets are communication endpoints supported by the operating system. Each socket has the potential to exchange information with other sockets of an appropriate type within a domain. Unlike pipes and socket pairs, sockets provide communication between processes that have no common ancestor. When communicating between processes on different machines (as when providing or using a network service), you need to have two processes separately create and name sockets and then send messages between them. The socket facility allows you to do this.

Socket types

Sockets are typed according to communication properties. Processes are presumed to communicate only between sockets of the same type, although there is nothing that prevents communication between sockets of different types should underlying communication protocols support this.

There are three basic types of sockets:

- **Stream sockets**—Provide for the bidirectional, reliable, sequenced, and unduplicated flow of data without record boundaries.
- **Datagram sockets**—Support a bidirectional flow of data that is not promised to be sequenced, reliable, or unduplicated.
- **Raw sockets**—Provide users with access to the underlying communication protocols that support sockets.

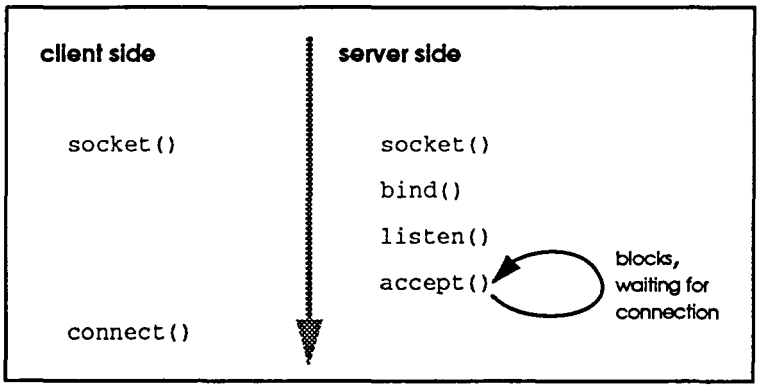
Socket domains

A *domain* is an address space shared by sockets. ConvexOS supports two domains: the *UNIX* domain, used for communication within the local system; and the *Internet* domain, used for communication between hosts on a network. The UNIX domain only allows communication between sockets on the same machine; therefore, it cannot be used to implement network applications.

Socket system calls

You use socket system calls—`accept`, `bind`, `connect`, `listen`, and `socket`—to establish connections between processes. Connection establishment is usually asymmetrical, with one process acting as a client and the other performing the role of a server. When willing to offer its services, the server binds a socket to a well-known address associated with the service and then passively listens on its socket. The client requests services by initiating a connection to the server's socket. Figure 14 illustrates a typical sequence of system calls used for connection-oriented communication between network processes.

Figure 14 Establishing a connection using sockets



After establishing a connection, the client and server processes call `read` and `write` or `recv` and `send` to transmit messages back and forth.

Socket system calls are also used for connectionless-mode communication. Figure 15 on page 40 illustrates a typical sequence of system calls used for this type of network communication.

Figure 15 Connectionless communication using sockets

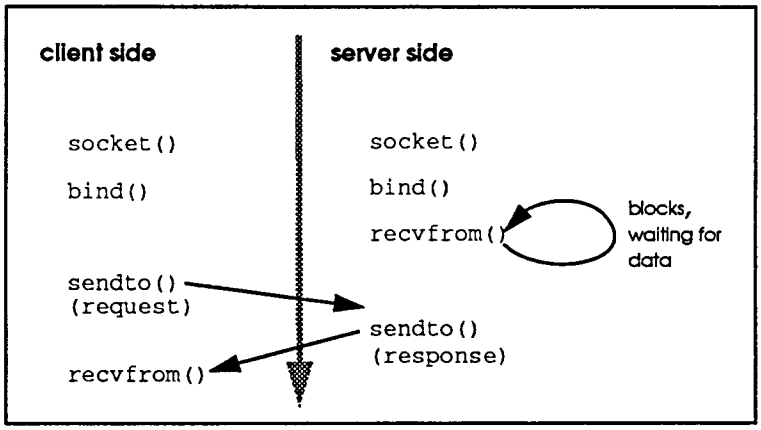


Figure 14 and Figure 15 illustrate only two possible scenarios for network communication. Refer to the *CONVEX Interprocess Communication (IPC) Programming Guide* for more examples, as well as sample code.

Network library routines

Locating a service on a remote host requires many levels of mapping before client and server may communicate. A service is typically assigned a descriptive name; for example, "the `login` server on host `monet`." This name and the name of the peer host must then be translated into network addresses, which are not necessarily as suitable for use by humans. Finally, the address must then be used to locate a physical location and route to the service.

It is better for a network to not require hosts to use names that reveal their physical location to the client host. Instead, underlying services in the network may discover the actual location of the server host at the time a client wishes to communicate. This ability to have hosts named in a location-independent manner may induce overhead in connection establishment, because a discovery process must take place.

CONVEX networking software provides standard routines for mapping host names to network addresses (`gethostbyname`), network names to network numbers (`getnetbyaddr`), protocol names to protocol numbers (`getprotobynumber`), service names to port numbers (`getservbyport`), and appropriate protocols to use in communicating with the server process (`getprotobyname`).

Networking software also provides byte-handling routines that simplify manipulation of names and addresses. Table 1 summarizes routines for manipulating variable-length byte strings and handling byte-swapping of network addresses and values.

Table 1 Byte-handling routines

Call	Synopsis
`bcmp` (*s1, s2, n*)	Compare byte-strings; 0 if same, not 0 otherwise
`bcopy` (*s1, s2, n*)	Copy *n* bytes from *s1* to *s2*
`bzero` (*base, n*)	zero-fill *n* bytes starting at *base*
`htonl` (*val*)	Convert 32-bit quantity from host to network byte order
`htons` (*val*)	Convert 16-bit quantity from host to network byte order
`ntohl` (*val*)	Convert 32-bit quantity from network to host byte order
`ntohs` (*val*)	Convert 16-bit quantity from network to host byte order

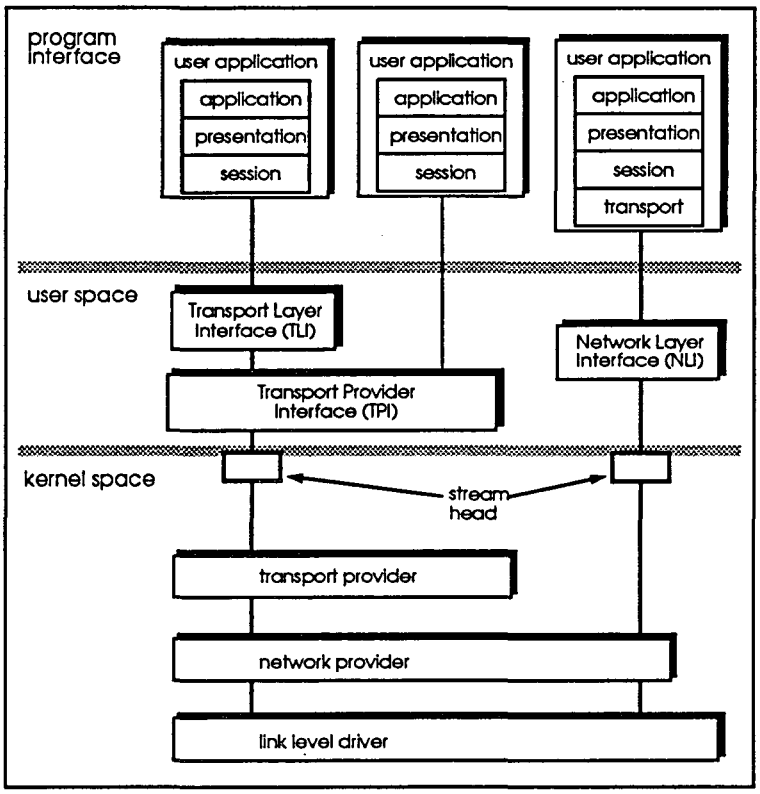
Byte-swapping routines are provided because TCP/IP defines a standard byte order for integral values that may or may not match the natural byte ordering on a particular machine. Because current CONVEX architecture orders bytes in the way TCP/IP expects them, byte-swapping routines have no effect; however, the use of these routines is strongly recommended to ensure portability of source code to other architectures. Application programs normally need to use byte-swapping routines only if they generate or interpret network addresses.

STREAMS

CONVEX's STREAMS implementation supports three application program interfaces: a Transport Layer Interface (TLI), a Transport Provider Interface (TPI), and a Network Layer Interface (NLI). Because TLI and TPI are independent of any specific network or transport layer protocol, using these interfaces improves application portability. NLI is network layer-specific.

Figure 16 on page 42 shows the application program interfaces provided by STREAMS.

Figure 16 WAN transport program interfaces



Transport Layer Interface (TLI)

TLI is implemented as a user-space library. The TLI library provides a high-level interface to transport layer primitives. TLI functions use standard STREAMS I/O system calls to send data to and receive data from the transport provider. TLI handles all communications between STREAMS and user applications, freeing application programmers from having to program STREAMS system calls directly.

The CONVEX TLI Library is based on the X/Open Transport Interface Issue 3 (XTI 3) and the System V Interface Definition Issue 4 (SVID 4).

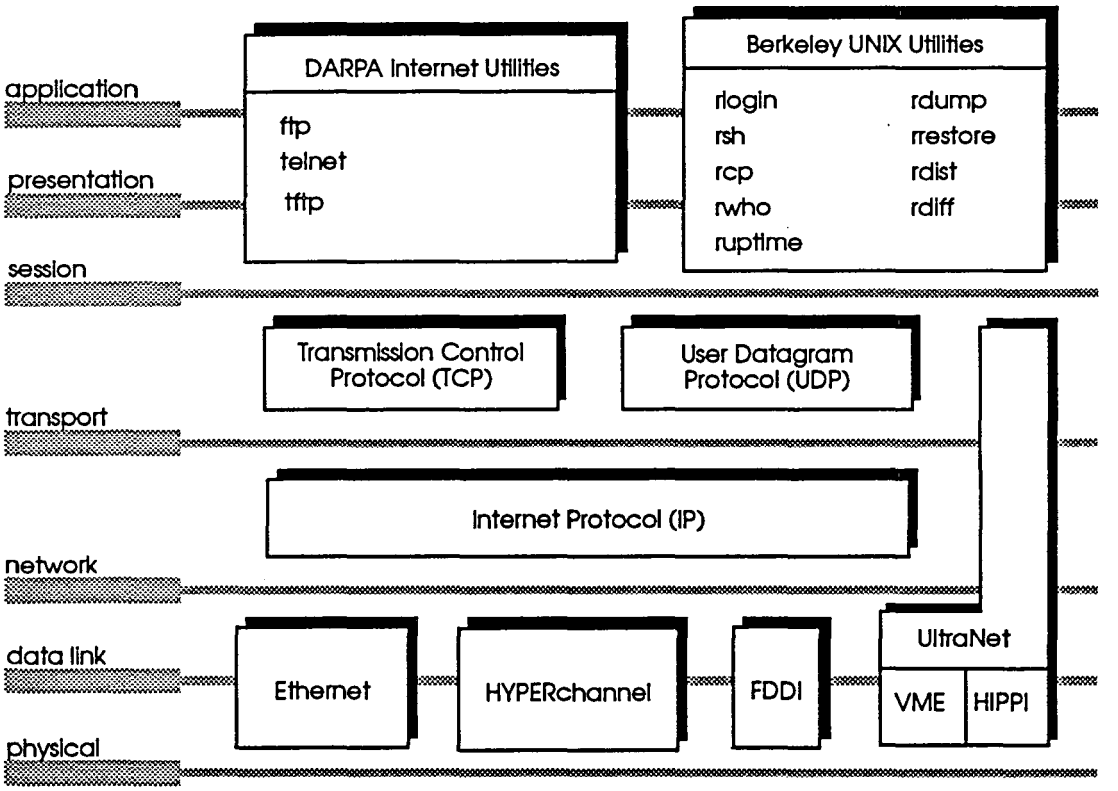
Transport Provider Interface (TPI)

TPI is implemented through standard STREAMS system calls for communicating with the stream head. Application programs access transport services, such as TCP, UDP, and TP0/2/4, by creating a stream to the transport drivers and using `getmsg` and `putmsg` calls to exchange messages with the transport provider. Transport drivers communicate with a user-level program through the stream head.

Network Layer Interface (NLI)

NLI is also provided through standard STREAMS system calls for communicating with the stream head. Applications access network layer services, such as IP and X.25 drivers, by creating a stream to the network provider and using `getmsg` and `putmsg` calls to exchange messages. The network provider communicates with a user-level program through the stream head.

CONVEX Internet Services



CONVEX Internet Services 3

Product description

CONVEX Internet Services provides transparent access and resource sharing capabilities among CONVEX systems and a wide variety of other machines, including personal computers, workstations, minicomputers, and supercomputers.

Internet Services software includes

- DARPA Internet protocols that run over Ethernet, HYPERchannel, FDDI, or UltraNet interfaces, or over serial lines.
- Berkeley networking utilities, used to communicate with other ConvexOS or UNIX systems that run Berkeley Software Distribution (BSD) networking.
- DARPA Internet utilities, used to communicate with systems that use TCP/IP protocols, but not necessarily BSD networking.
- System management utilities and databases used to configure, maintain, and monitor a TCP/IP network.
- Interprocess Communication (IPC) system calls and library routines that facilitate communication among application programs.

Prerequisites

To use CONVEX Internet Services, your system must be running ConvexOS and Utilities.

Internet Services runs over any of the following:

- Excelan VMEbus Ethernet controllers
- Excelan Multibus Ethernet controllers
- Network Systems Corporation A400 HYPERchannel Adapters through the IKON 10077-NSC Multibus Interface
- Network Systems Corporation NB400 HYPERchannel Adapters through the IKON 10090 VMEbus Interface
- CONVEX FDDI (Fiber Distributed Data Interface)
- CONVEX VME/UltraNet Interface
- CONVEX HIPPI/UltraNet Interface
- Serial lines

These interfaces and controllers are available as separate hardware products.

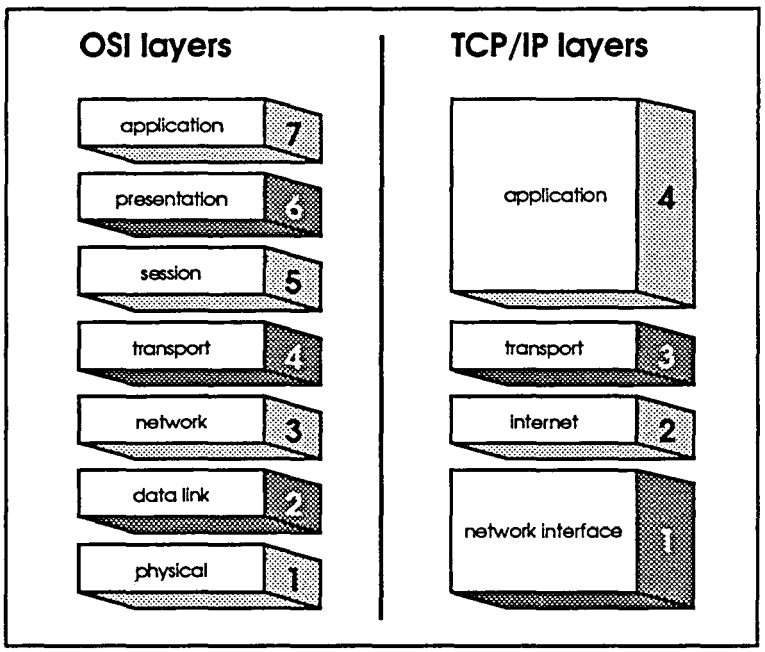
Protocols and standards conformance

The CONVEX Internet Services product is based on Berkeley Networking Software, Release 1 (January 1989), also known as the 4.3BSD Tahoe release. Berkeley Networking implements the DARPA Internet TCP/IP protocol suite.

TCP/IP layers

TCP/IP software is arranged in layers; but, because the TCP/IP protocol suite predates the OSI model, its layers do not correspond directly to OSI layers. Figure 17 on page 49 compares OSI and TCP/IP protocol layers.

Figure 17 OSI *vs* TCP/IP protocol layers

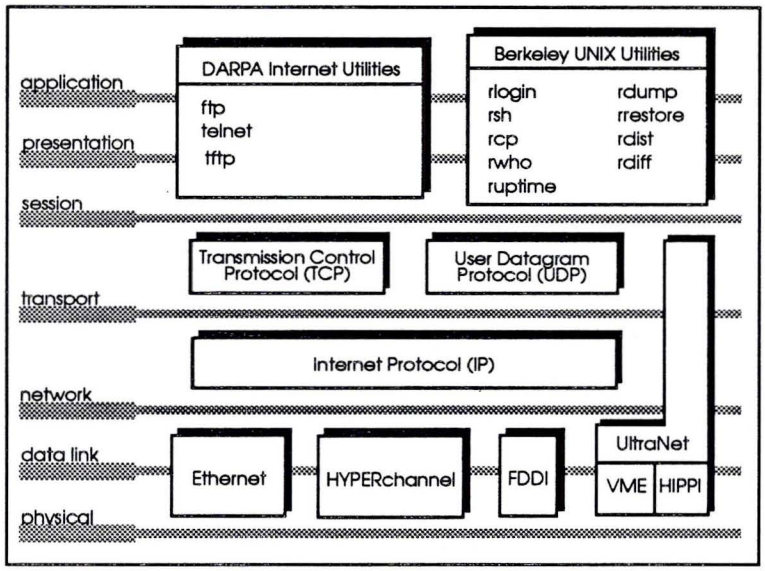


Although TCP/IP protocols are arranged in fewer layers than in the OSI model, the protocol suite provides the same basic services. The following briefly describes the functions performed at each layer of the TCP/IP suite.

- **Network interface layer**—Also called the data link layer, the network interface layer is responsible for transmitting datagrams over a specific network medium, such as Ethernet.
- **Internet layer**—The Internet Protocol (IP) runs at this layer. It encapsulates packets into datagrams and delivers them from one machine to another. This layer includes addressing and routing services.
- **Transport layer**—The Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP) run at this layer to provide error-free communication between application programs.
- **Application layer**—Provides users with access to network services, such as electronic mail and file transfer utilities.Application layer protocols include File Transfer Protocol (FTP) and TELNET protocol.

Figure 18 on page 50 shows TCP/IP services and protocols mapped to the OSI model.

Figure 18 CONVEX Internet Services mapped to the OSI model



DARPA Internet protocols

Internet Services provides a full set of Department of Defense (DoD) compliant TCP/IP protocols, including

- Internet Protocol (IP)
- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)
- Internet Control Message Protocol (ICMP)
- Address Resolution Protocol (ARP)
- Routing Information Protocol (RIP)
- TELNET protocol
- Trivial File Transfer Protocol (TFTP)
- File Transfer Protocol (FTP)
- Simple Mail Transfer Protocol (SMTP)

Internet standards

In keeping with CONVEX's commitment to Open Supercomputing, the Internet Services product conforms to the DoD Requests for Comments (RFCs) and U.S. military standards listed in this section.

General RFCs

RFC 1009, Requirement for Internet Gateways

RFC 1010, Assigned Numbers

Application level RFCs

RFC 742, FINGER Protocol

RFC 783, The TFTP Protocol Revision 2

RFC 821, Simple Mail Transfer Protocol

RFC 822, Standard for the Format of ARPA Internet Text Messages

RFC 854, TELNET Protocol

RFC 920, Domain Requirements

RFC 959, File Transfer Protocol (FTP)

RFC 974, Mail Routing and the Domain System

RFC 1032, Domain Administrator's Guide

RFC 1033, Domain Administrator's Operations Guide

RFC 1034, Domain Names—Concepts and Facilities

RFC 1035, Domain Names—Implementations and Specifications

Transport level RFCs

RFC 793, Transmission Control Protocol

RFC 964, Some Problems with the Specification of the Military Standard Transmission Control Protocol

Internet level RFCs

RFC 791, Internet Protocol

RFC 792, Internet Control Message Protocol

RFC 919, Broadcasting Internet Datagrams

RFC 922, Broadcasting Internet Datagrams in the Presence of Subnets

RFC 950, Internet Standard Subnetting Procedure

RFC 963, Some Problems with the Specification of the Military Standard Internet Protocol

RFC 1058, Routing Information Protocol (RIP)

Network interface level RFCs

RFC 768, User Datagram Protocol

RFC 826, An Ethernet Address Resolution Protocol

RFC 894, Standard for the Transmission of IP Datagrams over Ethernet Networks

RFC 948, Two Methods for Transmission of IP Datagrams over IEEE 802.3 Networks

RFC 1044, Internet Protocol on Network System's HYPERchannel: Protocol Specification

RFC 1055, Nonstandard for the transmission of IP datagrams over serial lines: SLIP

U. S. military standards

MIL-STD 1777, Internet Protocol

MIL-STD 1778, Transmission Control Protocol

MIL-STD 1780, File Transfer Protocol

MIL-STD 1781, Simple Mail Transfer Protocol

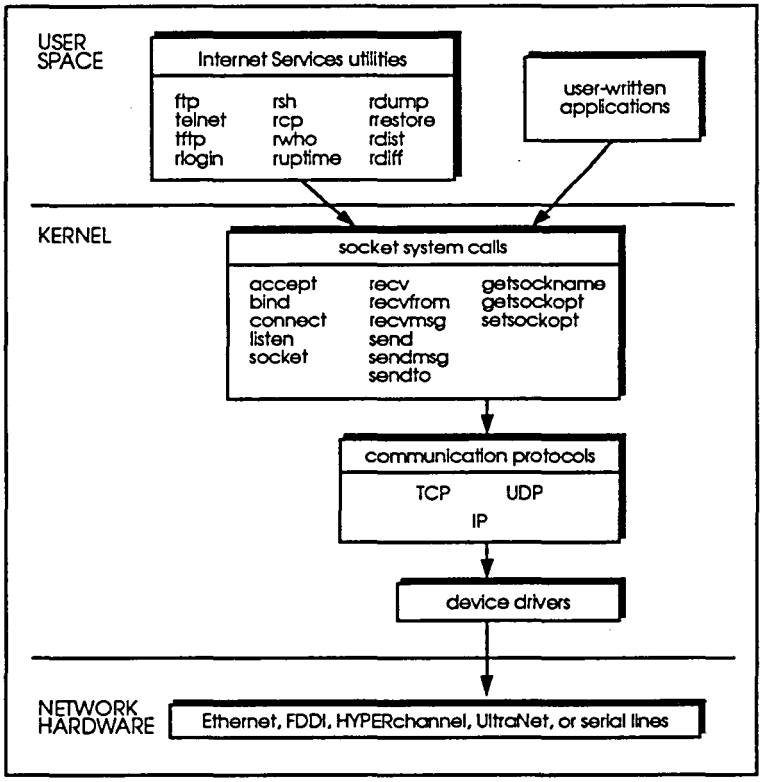
MIL-STD 1782, Telnet Protocol and Options

For information about how to obtain copies of the listed RFCs, refer to the section, "Obtaining information about standards," on page 12.

Network architecture

Internet Services utilities and user-written applications access TCP/IP protocols through the Berkeley socket interface. Sockets are implemented within the ConvexOS kernel, with access provided through a set of system calls, as shown in Figure 19 on page 53.

Figure 19 Internet Services network architecture



For more information about the socket interface, refer to the section, "Application program interface," on page 56.

Services

The CONVEX Internet Services product provides the following services:

- Berkeley networking utilities
- DARPA Internet utilities
- Berkeley Internet Name Domain server (BIND)
- Network interface drivers
- Serial Line Internet Protocol (SLIP)

Each of these services is described in this section.

Berkeley networking utilities

Berkeley utilities—`rlogin`, `rsh`, `rcp`, `rwho`, `ruptime`, `rdump`, `rrestore`, `rdist`, and `rdiff`—are used to communicate with other ConvexOS or UNIX systems that run Berkeley Software Distribution (BSD) networking. These utilities enable users to

- Log in to a remote host.
- Execute shell commands on a remote host.
- Copy files between hosts.
- Determine who is logged in to a remote host, or who is remotely logged in to the local host.
- Check the status of other machines on the network.
- Dump remote files to tape.
- Restore remote files from tape.
- Distribute copies of files to remote hosts.
- Compare files on networked hosts.

DARPA Internet utilities

DARPA Internet utilities—`telnet`, `ftp`, and `tftp`—are used to communicate with systems that use TCP/IP protocols, but do not run BSD networking. They enable users to

- Establish, control, and terminate a connection between the local host and a remote host.
- Log in to a remote host.
- Copy files between hosts.
- Append a local file to a remote file.
- Check the status of remote hosts or files.
- Change the working directory on the remote or the local host.
- Remove or rename files on a remote host.
- Create or remove a directory on a remote host.
- List the contents of remote directories.
- Display the time a remote file was last modified.
- Display the name of the working directory on a remote host.
- Display the type of operating system running on a remote host.

Berkeley Internet Name Domain server (BIND)

BIND enables clients to share information about network resources, such as host names and addresses, with other hosts on the network. In effect, BIND provides a distributed database system for hosts in a network.

BIND is fully integrated into Internet Services software for use in storing and retrieving host names and addresses. The system manager can configure a system to use either BIND, host table lookup routines, or a directory service such as the Network Information Service (NIS). Refer to the section, "Network Information Service (NIS)," on page 65, for a description of NIS.

BIND software consists of two parts: the `resolver`, a set of user interface routines that reside in the C library; and `named`, the name server daemon. Processes use resolver routines to build query packets and exchange them with `named`. `named` runs in the background and services queries on a specific network port.

Depending on whether BIND is running on the system, networking routines, such as `gethostbyname` and `gethostbyaddr`, use either the name server or host table lookup routines to resolve host names and addresses.

The chief advantage to using the name server over host table lookup routines is ease of maintenance. The system manager configures the network resource database on a single system and lets the name server keep data up-to-date on other networked machines. The name server also allows each system to maintain the database for a single domain, but have access to up-to-date information for other domains.

Network interface drivers

Because diverse network needs require different levels of performance, Internet Services supports Ethernet, HYPERchannel, FDDI, and UltraNet interfaces.

To run TCP/IP protocols over UltraNet or FDDI, your system must be licensed to run the CONVEX UltraNet Interface or CONVEX FDDI in addition to Internet Services. CONVEX VME/UltraNet, CONVEX HIPPI/UltraNet, and CONVEX FDDI are available as separate hardware and software products.

Serial Line Internet Protocol (SLIP)

Internet Services also includes the Serial Line Internet Protocol (SLIP). SLIP provides point-to-point communication over asynchronous serial lines at speeds from 1200 bps to 38400 bps.

Application program interface

Internet Services provides an application program interface through a set of Interprocess Communication (IPC) facilities, including pipes, socket pairs, and sockets.

ConvexOS facilitates and manages communication between processes. IPC programs interact with the operating system in much the same way as other programs—through library routines and system calls. These library routines and system calls give application programmers access to the full power and functionality of the TCP/IP protocol suite. IPC facilities enable processes to communicate with other processes running on the same or on different machines.

Pipes, socket pairs, and sockets are described in the section, "Application program interfaces," on page 35.

Configuration and management

Internet Services includes a set of files, utilities, and tunable parameters used to configure, manage, and monitor the operation of a TCP/IP network. These configuration utilities enable the system manager to

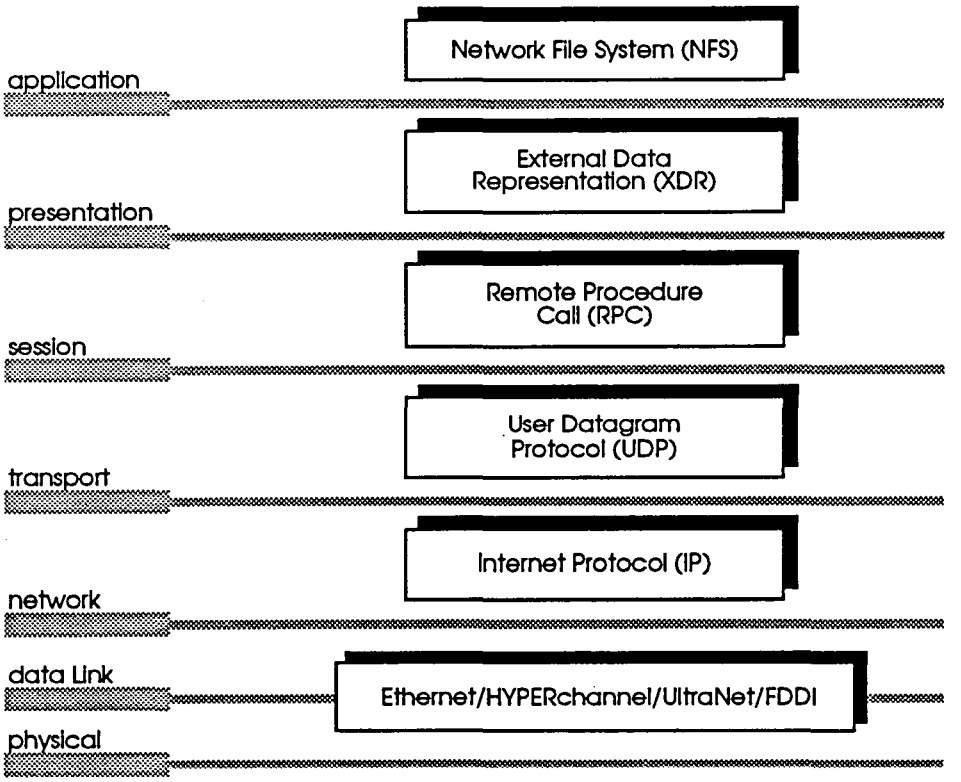
- Configure Ethernet, FDDI, and HYPERchannel interfaces.
- Configure network services.
- Configure and manage anonymous `ftp` accounts.
- Configure and manage BIND.
- Configure SLIP.
- Configure the Address Resolution Protocol (ARP).
- Set up and maintain the host name database.
- Control user access to the network.
- Set up and maintain Internet and HYPERchannel routing.
- Test and troubleshoot the network.
- Perform socket-level debugging.

Documentation

CONVEX Internet Services includes the manuals listed below.

- *CONVEX Internet Services User's Guide* describes how to perform basic networking tasks, such as logging into remote systems, copying files between systems, and executing commands on remote systems.
- *CONVEX Internet Services System Manager's Guide* contains information about administering systems on a CONVEX TCP/IP network. Many of the installation and configurations tasks this book describes are necessary for the operation of NFS and related facilities.
- *CONVEX Interprocess Communication (IPC) Programming Guide* describes additional facilities for implementing network applications.
- *CONVEX UltraNet Programmer's Reference* contains man pages describing CONVEX UltraNet Interface software.
- *ConvexOS Man Pages for Users, ConvexOS Man Pages for Programmers,* and *ConvexOS Man Pages for System Managers* contain man pages for ConvexOS and Internet Services software.

CONVEX Network File System (NFS)



CONVEX Network File System (NFS)

4

Product description

CONVEX Network File System (NFS) provides transparent access to file systems on remote machines. NFS, developed by Sun Microsystems, links together heterogeneous systems—including personal computers, workstations, supercomputers, and mainframes—to share resources and files over local area and wide area networks. File sharing is accomplished by mounting a remote file system, then reading or writing files in place.

NFS allows a variety of machines and operating systems to act as client or server. Because the environment is transparent, users can access remote files as if these files were on the local machine. Individual workstations can have access to information residing anywhere on the network. In addition to providing transparent access to files, NFS provides users with a single network-wide directory structure.

NFS software consists of a modified kernel, a set of library routines, and a collection of utility commands. Where previous networking schemes, such as `rcp` and `ftp`, require additional remote commands to perform routine operations, NFS enables users to operate with the same command set they use for local operations. Thus, NFS is transparent to the user.

Prerequisites

To use NFS, your system must be running ConvexOS and Utilities, and CONVEX Internet Services.

Software architecture

Traditional file systems are composed of directories and files, each of which has a corresponding index node (inode) containing information about the file, such as location, permissions, and access times. inodes are assigned unique numbers within a file system, but a file on one file system could have the same inode number as a file on another file system. This presents a problem in a network environment.

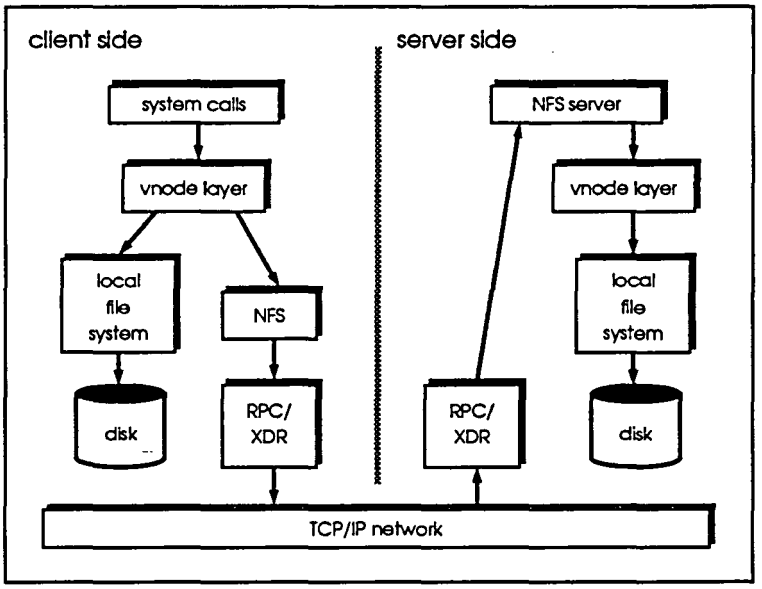
To solve this problem, CONVEX NFS is implemented through the virtual file system (VFS). VFS is based on the vnode, an additional layer in the file system structure that permits multiple types of file systems. VFS architecture divides the file subsystem into architecture-dependent and -independent layers, and provides a generic interface to the file system. This generic interface enables ConvexOS to accommodate local file systems as well as networked file systems.

Because ConvexOS file systems are manipulated through the VFS interface, the underlying file system implementation is transparent. Above the VFS interface, ConvexOS deals in vnodes; below this interface, the file system may or may not implement inodes.

A local VFS connects to file system data on a local device. The remote VFS defines and implements the NFS interface, using the remote procedure call (RPC) mechanism. RPC permits communication with remote services in a manner similar to the procedure calling mechanism available in many programming languages. RPC protocols are described using the eXternal Data Representation (XDR) package. XDR permits machine-independent representation and definition of high-level protocols on the network. (RPC and XDR are discussed in the section, "Application program interface," on page 69.)

Figure 20 on page 63 illustrates NFS network architecture.

Figure 20 NFS network architecture



The arrows in Figure 20 indicate the flow of a request for data from a file system.

On the client side:

- For access through a local virtual file system, the request is directed to file system data on devices connected to the client machine.
- For access through a remote virtual file system, the request passes through the RPC/XDR layers to the network.

On the server side, the request passes through the RPC/XDR layers to the NFS server. vnodes are used to access a local file system on the NFS server and service the request.

The path of the data request is retraced to return results.

Protocols and standards conformance

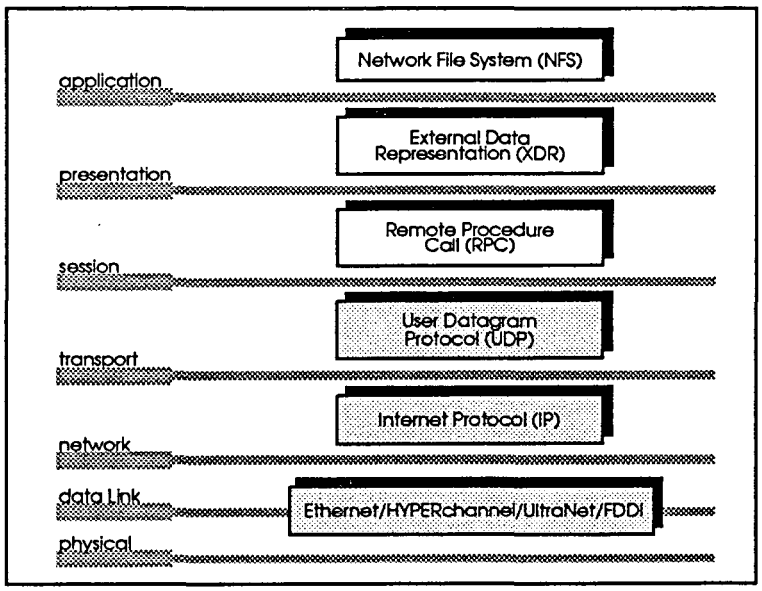
The CONVEX implementation of NFS is based on SUN NFSSRC 4.0, which conforms to the RFCs for NFS, RPC, and XDR listed below.

- RFC 1094, NFS: Network File System Protocol Specification
- RFC 1057, RPC: Remote Procedure Call Protocol Specification Version 2
- RFC 1014, XDR: External Data Representation Standard

For information about obtaining copies of the RFCs listed above, refer to the section, "Obtaining information about standards," on page 12.

CONVEX NFS includes protocols at the upper three layers of the OSI model. The CONVEX Internet Services product provides the lower layers, shown as shaded boxes in Figure 21.

Figure 21 CONVEX NFS protocols mapped to the OSI model



Services

The CONVEX NFS product includes the following services:

- Network File System (NFS)
- Network Information Service (NIS)
- Network Lock Manager
- The Automounter
- Secure NFS
- NETdisk
- Remote EXecution (REX)

Each of these services is described below.

Network File System (NFS)

NFS provides access to remote file systems in a heterogeneous network. Through the use of RPC/XDR protocols, the NFS protocol is machine, operating system, network architecture, and transport protocol independent. In the CONVEX implementation, UDP provides the transport layer.

NFS services are transparent to users; there is no syntactical difference between reading or writing a file on a local disk, and reading or writing a file on a disk attached to a remote machine. The NFS server exports shared files so that client machines can access them. Client machines request access by mounting the exported file systems.

NFS is not a distributed operating system, but rather, an interface that allows a variety of machines and operating systems to play the role of client or server.

Network Information Service (NIS)

NIS, formerly named Yellow Pages (YP), provides a distributed network lookup service that eases the job of administering networked machines. By using NIS, password, group, and host information for an entire network can be maintained in a single database. NIS allows a single point of administration and gives all machines access to current data.

NIS maintains a set of *maps* that machines query as they would a database. All such maps may be fully replicated on several machines known as NIS servers, each of which runs a server process for the maps. The response is the same regardless of which server process answers a client request, because the map set is identical for each server. This allows the network to have multiple servers, and makes NIS service highly reliable and available.

Installing the NIS version of a file does not require changes to existing applications; they are simply relinked with library routines that use the NIS service. The system manager can configure a system to use either NIS, host table lookup routines, or a name server such as BIND. Refer to the section, "Berkeley Internet Name Domain server (BIND)," on page 55, for a description of BIND.

The NIS interface is implemented using RPC and XDR, so a variety of operating systems and machine types can use the service.

Network Lock Manager

The Network Lock Manager supports advisory file and record locking on NFS file systems. Locking prevents multiple processes from modifying the same file at the same time, allows cooperating processes to synchronize access to shared files, and facilitates recovery from system crashes.

The lock manager uses two daemons to provide the locking service. The Network Status Monitor, `statd`, runs on all network machines. It tracks the status of network machines and informs the lock daemon, `lockd`, of relevant machine crashes and recoveries.

To use the locking service, application programs can either call the appropriate daemon directly with RPC/XDR or use the `lockf` system call. In the latter case, the kernel uses RPC to call the daemon. Network daemons use RPC to communicate among themselves.

The Automounter

The `automount` program enables users to access remote file systems on demand, without requiring a prior `mount` operation. When a user on a client machine running `automount` invokes a command that accesses a remote file or directory, such as opening a file with an editor, the automounter mounts the hierarchy to which that file or directory belongs. The automounter leaves

the file or directory mounted as long as the user needs it, but automatically unmounts it if it is not accessed for a certain amount of time. No mounting occurs at boottime, and users no longer must know the superuser password to mount a directory. It is all done automatically and transparently.

Mounting some file hierarchies with the automounter does not exclude the possibility of mounting others with `mount`. For example, a diskless Sun workstation must mount the /export/root, /export/swap, /usr, and /export/exec/kvm directories by using the `mount` program and the /etc/fstab file.

Unlike `mount`, `automount` does not consult /etc/fstab for a list of hierarchies to mount. Instead, it consults a series of direct or indirect maps. The names of the maps are passed to `automount` from the command line or from another map.

Secure NFS

Networking systems are vulnerable to two types of security violations:

1. Access to data by unauthorized users.

   When an NFS server receives a request for a remote file mounted without the secure option, the NFS server authenticates the request by authenticating the machine where the request originated, but not the user making the request. This system poses a threat to network security because a user who has network privileges on one machine can impersonate another user and access files owned by that user on another machine.

2. The injection of arbitrary data into the network.

   Given root access and a good knowledge of network programming, anyone can inject arbitrary data into the network. For example, an unauthorized person could impersonate an authorized user by generating, or collecting and retransmitting, the right packets. These attacks are particularly dangerous, because they affect data integrity.

Secure NFS authentication and data encryption greatly enhance network security by providing protection against both types of violations. Secure NFS performs authentication at the RPC level, resulting in a standard authentication system that covers all RPC-based applications, such as NFS and NIS. This system uses public key cryptography to authenticate both users and machines. Users can log in on any machine by using their password, just as they could log in on any terminal. No knowledge of the underlying authentication system is required.

To access files on an NFS file system mounted with the `-secure` option, each user must have an entry in the key database of the server machine. Keys are encrypted with the user's password to ensure that they remain secret. Because of export restrictions on DES technology, secure NFS is currently unavailable to sites outside the U.S. and Canada.

Secure NFS relies on facilities provided by NIS; therefore, you must run NIS if you wish to use Secure NFS.

NETdisk

NETdisk enables you to use a CONVEX NFS server to boot a diskless workstation. After the workstation has booted, NETdisk provides it with access to the files it needs, such as the /root and /swap directories. Thus NETdisk allows you to use relatively inexpensive diskless workstations without purchasing expensive boot or file servers.

NETdisk currently supports as clients Sun-3 workstations and SPARCstations running SunOS 4.0, 4.0.3, and 4.1.

The NETdisk service consists of a set of administrative programs loaded into the /usr/etc/install directory when NFS is installed.

Remote EXecution (REX)

The `rex` program and its accompanying daemon, `rexd`, enable users to execute commands remotely, in an environment similar to that on the machine where `rex` is invoked. All environment variables are passed, and the current working directory is preserved. To preserve the working directory, the working file system must be already mounted on the host or be exported to it.

To improve load balancing between hosts, `rex` enables users to run `nroff`, `make`, and other jobs onto lightly-loaded machines. Unlike `rsh` and `rlogin`, `rex` neither starts a shell on the remote host nor performs remote logins. Instead, `rex` accomplishes the remote execution task by mounting local file partitions on remote machines via NFS.

Like `rlogin` and `rsh`, `rex` operates quickly, without discernible start-up time. In fact, because `rex` does not perform remote logins or start remote shells, it is faster than either `rsh` or `rlogin`. However, while `rex`'s quick start-up time is certainly an advantage, it limits users to relatively simple command sequences. Because `rex` does not automatically start a remote shell, it cannot interpret the complex command sequences interpreted by `rsh`. You can overcome this limitation by using `rex` to start a shell on the remote host and then using the shell to perform complex commands.

`rex` does not require installation and uses a simple syntactical structure. Because no installation is required, users should be able to use these utilities immediately after installing NFS.

Application program interface

CONVEX NFS includes two application program interfaces: Remote Procedure Call (RPC) and eXternal Data Representation (XDR). The NFS Protocol Specification is implemented using RPC and XDR. These facilities are also available for use in implementing distributed systems. CONVEX NFS also includes `rpcgen`, a compiler for the RPC language.

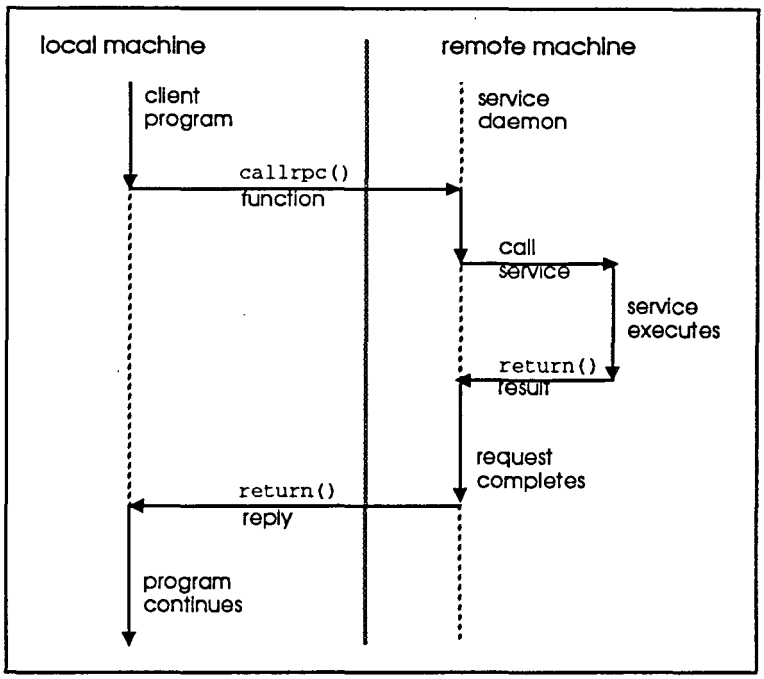
Remote Procedure Call (RPC)

The RPC facility allows a client process to have another process execute a procedure call, as if the caller had executed the procedure call in its own address space. Because caller and server are separate processes, they can reside on different machines.

The RPC mechanism is implemented as a library of procedures and a specification for portable data transmission, known as eXternal Data Representation (XDR). RPC does not depend on services provided by specific protocols, so it can be used with any underlying transport protocol.

RPC allows communication with remote services in a manner similar to the local procedure calling mechanism available in many programming languages. Figure 22 on page 70 shows a model of the remote procedure call mechanism.

Figure 22 RPC model



In the local procedure call model, the caller places arguments to a procedure in a specified location, then transfers control to the procedure. When the procedure completes, the caller regains control. At that point, the caller extracts the results from the specified location and continues execution.

The remote procedure call is similar, in that one thread of control logically winds through two processes—one is the caller's process, the other is the server's process. The caller sends a call message that includes the procedure's parameters to the server and waits for a reply. The reply includes the procedure's results. After the caller receives a reply, it extracts the results and resumes execution.

On the server side, a dormant server awaits the arrival of a call message. When a call message arrives, the server extracts the procedure's parameters, computes the results, sends a reply, and awaits the next call message.

Each RPC server supplies a program made up of a set of procedures described using RPC Language, an extension to the XDR language. The combination of host address, program number, and procedure number uniquely identifies one remote service procedure.

rpcgen

The details of programming applications that use RPC can be overwhelming. Perhaps most daunting is writing XDR routines necessary to convert procedure arguments and results into their network format and vice versa.

The `rpcgen` compiler helps programmers write RPC applications simply and directly. `rpcgen` handles most of the lower-level details, allowing programmers to debug the main features of their applications, instead of spending most of their time debugging network interface code.

`rpcgen` accepts a remote program interface definition written in RPC Language, and produces C language output. Developers compile and link `rpcgen` output files in the usual way. The developer writes server procedures in any language that observes ConvexOS C calling conventions and links them with the server skeleton produced by `rpcgen` to get an executable server program.

To use a remote program, a programmer writes an ordinary main program that makes local procedure calls to the client stubs produced by `rpcgen`. Linking this program with `rpcgen` routines creates an executable program.

`rpcgen` reduces development time needed for coding and debugging low-level routines. All compilers, including `rpcgen`, do this at a small cost in efficiency and flexibility. However, many compilers allow escape hatches for programmers to mix low-level code with high-level code; `rpcgen` is no exception. In speed-critical applications, hand-written routines can easily be linked with `rpcgen` output. Also, programmers may use `rpcgen` output as a starting point, and rewrite it as necessary.

External Data Representation (XDR)

The External Data Representation (XDR) standard consists of a set of library routines that provides a common way of representing data types over a network. XDR allows applications to transfer data between diverse machines, such as Sun Workstations, VAX, IBM-PC, and CONVEX machines.

XDR uses a language to describe intricate data formats in a concise manner. The language can be used only to describe data; it is not a programming language. XDR describes the most commonly used data types of high-level languages such as Pascal or C, so that applications written in these languages will be able to communicate easily over some medium.

The XDR standard makes the following assumption: that bytes (or octets) are portable, where a byte is defined to be 8 bits of data. A given hardware device should encode the bytes onto the various media in such a way that other hardware devices may decode the bytes without loss of meaning.

XDR's approach to standardizing data representations is canonical; it defines a single byte order (Big Endian), a single floating-point representation (IEEE), and so on. Any program running on any machine can use XDR to create portable data by translating its local representation to the XDR standard representations. Similarly, any program running on any machine can read portable data by translating the XDR standard representations to its local equivalents. Using a single standard completely decouples programs that create or send portable data from those that use or receive portable data.

Configuration and management

CONVEX NFS provides network management capabilities that enable the system manager to

- Display statistical information about the NFS and RPC interfaces.
- Display information about RPC services running on networked machines.
- List all clients that are using remotely mounted file systems.
- List exported file systems.
- Test a network connection by using RPC to send a one-way stream of packets from one host to another.

Documentation

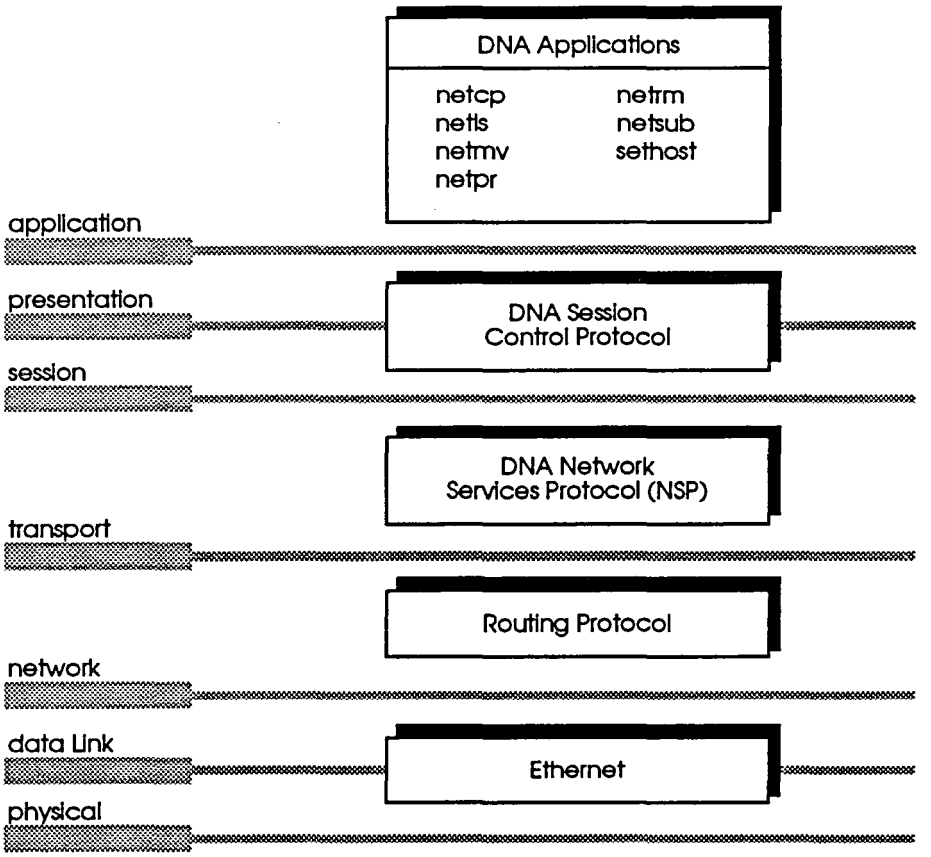
CONVEX NFS includes the manuals listed below.

- *CONVEX NFS Concepts* provides an introduction to NFS and related network facilities, such as the automounter, and Network Information Service (NIS).
- *CONVEX NFS System Manager's Guide* explains how to install, configure, maintain, and troubleshoot NFS and related facilities.
- *CONVEX NFS Reference Set* describes NFS, NIS, RPC and XDR protocol specifications and explains how to use NFS and related facilities to implement distributed programs.
- *CONVEX NFS Programmer's Reference* contains man pages describing CONVEX NFS software.

Other CONVEX manuals that may be useful to the NFS user, programmer, or system manager are listed below.

- *CONVEX Internet Services System Manager's Guide* contains information for administering systems on a CONVEX TCP/IP network. Many of the installation and configurations tasks this book describes are necessary for the operation of NFS and related facilities.
- *CONVEX Interprocess Communication (IPC) Programming Guide* describes additional facilities for implementing network applications.
- *ConvexOS Man Pages for Users, ConvexOS Man Pages for Programmers,* and *ConvexOS Man Pages for System Managers,* contain man pages describing ConvexOS software.

CONVEX COVUEnet



CONVEX COVUEnet 5

Product description

CONVEX COVUEnet enables CONVEX systems to participate as end nodes on a DECnet-COVUEnet network. All network functions and capabilities, whether standard Digital Network Architecture (DNA) capabilities or user-written applications, are available with another DECnet or COVUEnet node that provides the complementary capabilities. For example, a COVUEnet system can use the network virtual terminal client capability when communicating with remote nodes that provide a network virtual terminal server.

CONVEX systems running COVUEnet may be attached to Ethernet LANs and share information with systems running DECnet. This information sharing can take the form of remote logins, deletion, printing, renaming, transferring, and batch execution of files between systems, and remote directory listings. COVUEnet also allows you to transfer mail to and from CONVEX systems and any other system running DECnet.

Prerequisites

To use COVUEnet, your system must be running ConvexOS and Utilities. COVUEnet can communicate only with other COVUEnet or Phase IV DECnet products.

COVUEnet runs over either a VMEbus or Multibus Ethernet controller, available as separate hardware products.

Differences between COVUEnet and DECnet

COVUEnet differs from DECnet in several ways. The significant differences are listed below.

- COVUEnet does not allow transparent remote file access from programs running on a CONVEX computer. In other words, a separate library must be used to perform remote file access because the standard C library I/O functions cannot be used.
- COVUEnet does not support the DECnet ULTRIX system call interface.
- COVUEnet does not support the LAT, DDCMP, MOP, or X.25 protocols.
- COVUEnet provides support for end nodes only; it does not support any DECnet routing features.
- COVUEnet does not support the "%" wildcard character for CONVEX file specifications. It uses "?" instead.
- The COVUEnet version of `NCP` does not support event logging.

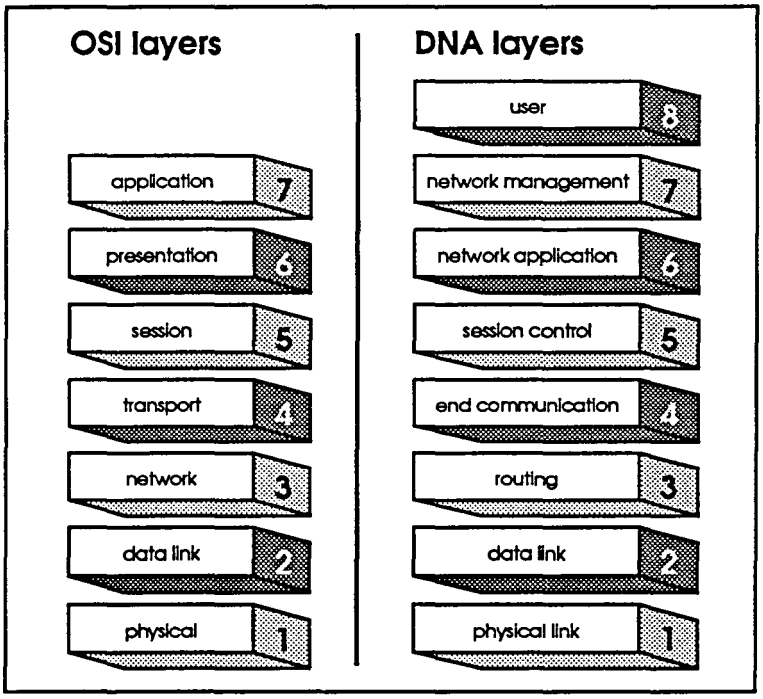
Protocols and network architecture

CONVEX COVUEnet conforms to Digital Network Architecture (DNA) structure and protocols, which makes it compatible with DECnet Phase IV. DNA specifies protocols, interfaces, and functions that enable machines to communicate over a DECnet network.

DNA layers

Like the network architecture described by the OSI model, DNA protocols are arranged in layers. Figure 23 compares OSI and DNA layers.

Figure 23 OSI *vs* DNA layers



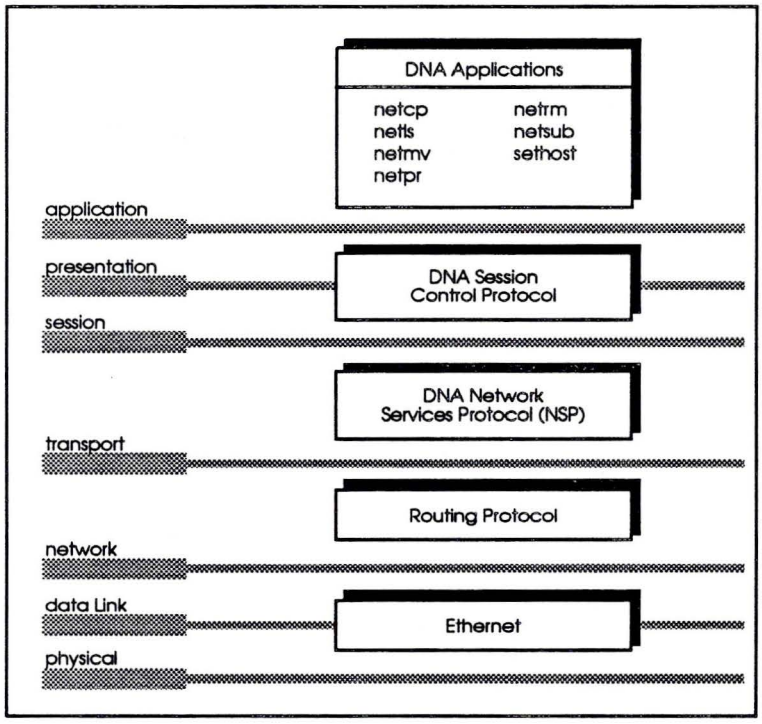
DNA layers are very similar to OSI layers. The following section briefly describes the functions performed at each DNA layer:

- **Physical link layer**—Responsible for transmitting data bits over a particular physical medium.
- **Data link layer**—Responsible for the error-free transmission of data over a communication link.
- **Routing layer**—Analogous to the OSI network layer, this layer is responsible for routing data between nodes.

- **End communication layer**—Responsible for end-to-end data transfer, ordered delivery of data, flow control, and error recovery.
- **Session control layer**—Responsible for establishing, managing, and terminating a period of communication between end-users.
- **Network application layer**—Functions similarly to the application and presentation layers of the OSI model to provide services to the network user, such as electronic mail and file transfers.
- **Network management layer**—Responsible for controlling and monitoring the network.
- **User layer**—Provides network management and user-written programs access to lower layers.

Because of the similarity between OSI and DNA layers, COVUEnet protocols can be easily mapped to the OSI model, as shown in Figure 24.

Figure 24 COVUEnet protocols mapped to the OSI model



DECnet protocols and utilities

COVUEnet implements the following DECnet protocols and utilities:

- Network Services Protocol (NSP)
- Session Control Protocol (SCP)
- Data Access Protocol (DAP)
- Virtual terminal protocol (CTERM)
- VMS/sendmail gateway (cumail)
- Network Control Protocol (NCP)
- Network Information and Control Exchange (NICE) protocol
- Network Manager Listener (NML)

Services

COVUEnet enables CONVEX users to

- Copy files between nodes on a DECnet-COVUEnet network.
- List the contents of a directory on a remote node.
- Rename or move files on the same remote node.
- Submit an existing file on a remote node to the default print queue on the remote node.
- Delete files from any node on the network.
- Submit an existing file on a remote node to the default batch queue on the remote node.
- Establish an interactive connection to a remote node.The user's terminal acts as though it were connected to the remote system, with that system executing commands as they are entered at the terminal.
- Transfer mail to and from CONVEX and VAX nodes on the DECnet-COVUEnet network.

Application program interface

User-written applications have access to the same program interfaces used by COVUEnet network services, virtual terminal, remote file access, and network management programs. COVUEnet provides appropriate mechanisms to ensure reliable, effective communication between tasks, regardless of their location in the network.

Application programs access COVUEnet services through two interfaces: the Network File Access Routine System (NFARS), and task-to-task communication.

Network File Access Routine System (NFARS)

Programs interface to the Remote File Access System through a set of routines called the Network File Access Routine System (NFARS). NFARS routines are contained in an object-code library that can be accessed by application programs.

NFARS library routines use the DNA Data Access Protocol (DAP) for file transfers. For more information on DAP, refer to the *VAX/VMS Guide to DECnet-VAX Networking* or the *VAX/VMS Networking Reference Manual*.

NFARS library routines enable application programs to perform the following operations on a remote COVUEnet-DECnet node:

- Initiate a logical link to a remote node and open or create a file on that node.
- Close a remote file.
- Delete a remote file.
- Read data from an open remote file.
- Write data to an open remote file.
- Rename an open file on a remote node.
- Retrieve file name specifications by using wildcard expansion.
- Display the text of an error message returned from an NFARS library routine call.

Task-to-task communication

A logical link is a temporary software data path established between two unrelated processes in a COVUEnet-DECnet network. The logical link between two processes is comparable to an I/O channel over which both processes can send and receive data. A logical link must be established between the two processes before data can be exchanged. This exchange of data between two processes over a logical link is called *task-to-task communication.*

To establish a logical link between two processes, one process informs the other of the attempt to communicate with it. After the logical link has been established, the two processes exchange data through system calls, such as `open`, `read`, and `write`. At any time, either process can terminate the logical link and transmit data explaining the reason for termination.

COVUEnet task-to-task communication system calls enable application programs to perform the following operations:

- Request a connection to a task.
- Accept or reject a logical link request.
- Terminate or disconnect a logical link.
- Transmit and receive data over a logical link.
- Return the status of a logical link.
- Receive access control information from a client.
- Obtain the name of the local node.
- Select I/O data format and mode.
- Determine the maximum number of bytes allowed in a read/write request.
- Register a program as a server by object name or number.
- Inform a program whenever interrupt data is received.

Configuration and management

COVUEnet provides client and server network management capabilities through Network Control Program (NCP) commands. NCP enables the system manager to

- Set, display, modify, and clear network control parameters and statistics.
- Specify the local node or a remote node as the executor for a specific NCP command or all subsequent NCP commands.
- Disconnect logical links.
- Display information about an NCP command or topic.
- Test logical links on a single node or between the local and a remote node.
- Manipulate volatile and permanent node databases.

Documentation

The CONVEX COVUEnet product includes the manuals listed below.

- *CONVEX COVUEnet User's Guide* explains how to log in to remote nodes and how to access remote files using COVUEnet software.
- *CONVEX COVUEnet System Manager's Guide* explains how to install, start, and stop the network; set up user accounts; and use NCP to configure, control, monitor, and test the network.
- *CONVEX COVUEnet Programmer's Guide* explains how to use COVUEnet from within source programs.
- *CONVEX COVUEnet Reference* describes user commands, the remote file access library, task-to-task function calls, and NCP commands.
- *CONVEX COVUEnet Programmer's Reference* contains man pages describing COVUEnet software.

CONVEX manuals that may help you use COVUEnet and other CONVEX COVUE products are listed below.

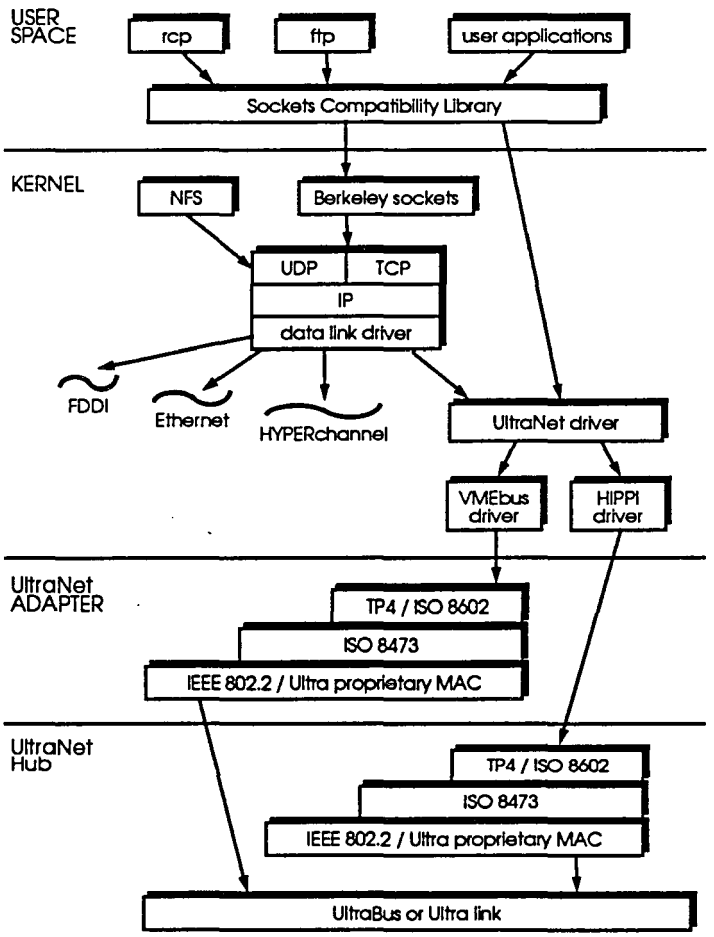
- *CONVEX COVUEbatch Guide* describes how to use CONVEX COVUEbatch software. It explains how to submit jobs; how to display and remove jobs from a CONVEX batch queue; and how to initialize, start, stop, and delete COVUE batch queues.

- *CONVEX COVUEbatch Programmer's Reference* contains man pages describing COVUEbatch software.
- *CONVEX COVUEbinary User's Guide* describes the COVUEbinary utility that allows VAX data files to be read and written by FORTRAN programs running on a CONVEX.
- *CONVEX COVUEbinary Programmer's Reference* contains man pages describing COVUEbinary software.
- *CONVEX COVUElib User's Guide* contains information about CONVEX's version of DEC's runtime library routines. It explains how COVUElib translates file names under COVUEshell, how to use arguments and string descriptors in CONVEX FORTRAN, and the differences between VAX/VMS and COVUElib runtime library functionality.
- *CONVEX COVUElib Programmer's Reference* contains man pages describing COVUElib software.
- *CONVEX COVUEshell Reference Manual* describes the CONVEX version of DEC's Digital Command Language (DCL). The manual describes in detail DCL-like commands supported on CONVEX systems.
- *CONVEX COVUEshell System Manager's Guide* describes how to configure and maintain COVUEshell.
- *CONVEX COVUEshell Programmer's Reference* contains man pages describing COVUEshell software.
- *ConvexOS Man Pages for Users, ConvexOS Man Pages for Programmers,* and *ConvexOS Man Pages for System Managers* contain man pages describing ConvexOS software.

The following Digital Equipment Corporation (DEC) manuals may help you with VAX/VMS products:

- *Guide to Using VAX/VMS* provides information on general tasks performed by any user utilizing the VAX/VMS operating system.
- *VAX/VMS Guide to DECnet-VAX Networking* gives an overview to networking on a VAX/VMS system using DECnet, describes network operations, and discusses how to bring up the network and keep it operating.
- *VAX/VMS Networking Reference Manual* provides detailed information on concepts, procedures, and tasks involving the network and on programming operations.
- *VMS Network Control Program Manual* provides a summary of the operation, command prompts, and syntax of NCP commands.

CONVEX UltraNet Interfaces



CONVEX UltraNet Interfaces 6

Product description

CONVEX UltraNet Interfaces are hardware and software products that allow CONVEX users to access an UltraNet high-speed network. A product of Ultra Network Technologies, Inc., UltraNet provides the networking performance and speed required by high-performance computing applications. The UltraNet network supports data rates of up to one gigabit per second, making it the highest performance LAN in the industry.

An UltraNet network consists of one or more UltraNet hubs that interconnect CONVEX and other supercomputers, mainframes such as IBM, and high-performance workstations such as Sun and Silicon Graphics. UltraNet Interfaces can be connected to the UltraNet hub via VMEbus or High Performance Parallel Interface (HIPPI).

CONVEX VME/UltraNet Interface hardware consists of a VME/UltraNet controller board, the Data Link Interface (DLI), and a shielded ribbon cable that connects the controller to the DLI. HIPPI/UltraNet Interface hardware consists of a HIPPI Channel Control Unit (CCU) that supports distances up to 25 meters. The interface supports both fiber optic and coaxial cable links to meet the distance and speed requirements of any host connection.

UltraNet Interface software includes the Sockets Compatibility Library, support for TCP/IP protocols, high-speed file transfer facilities, and network management utilities.

In addition to providing a reliable, high-performance means of transferring data between networked machines, the CONVEX UltraNet Interface is designed to interoperate with existing network application programs and program interfaces.

Prerequisites

To use the CONVEX UltraNet Interface, your system must be running ConvexOS and Utilities and CONVEX Internet Services.

The CONVEX VME/UltraNet Interface requires a VMEbus input/output processor, available as a separate hardware product.

Protocols and standards conformance

CONVEX UltraNet Interface software supports DoD TCP/IP protocols, and the ISO standards listed below.

Transport layer

- ISO 8073 Information Processing Systems—Open Systems Interconnection—*Transport Protocol Specification*, July 1986 (TP4)
- ISO 8602 Information Processing Systems—Data Communications—*Protocol for Providing the Connectionless-mode Transport Service*, July 1987

Network layer

- ISO 8473 Information Processing Systems—Data Communications—*Protocol for Providing the Connectionless-mode Network Service*, January 1988

Data link layer

- IEEE 802.2 Logical Link Control

In addition to standard protocols, the CONVEX UltraNet Interface uses proprietary, high-speed MAC level bridges to connect serial links, hubs, and hosts.

Network architecture

UltraNet achieves its high throughput by using large packets and proprietary protocols that run on the UltraNet adapter rather than on the CONVEX host. While UltraNet's use of proprietary protocols allows it to operate at high speeds, it has two disadvantages in terms of interoperability. UltraNet's protocols

1. Do not support TCP/IP facilities such as internet routing, subnetting, and broadcasting.
2. Can be used only between hosts directly connected to the UltraNet; that is, UltraNet's internal protocols cannot be used to communicate with hosts on other types of networks, such as Ethernet.

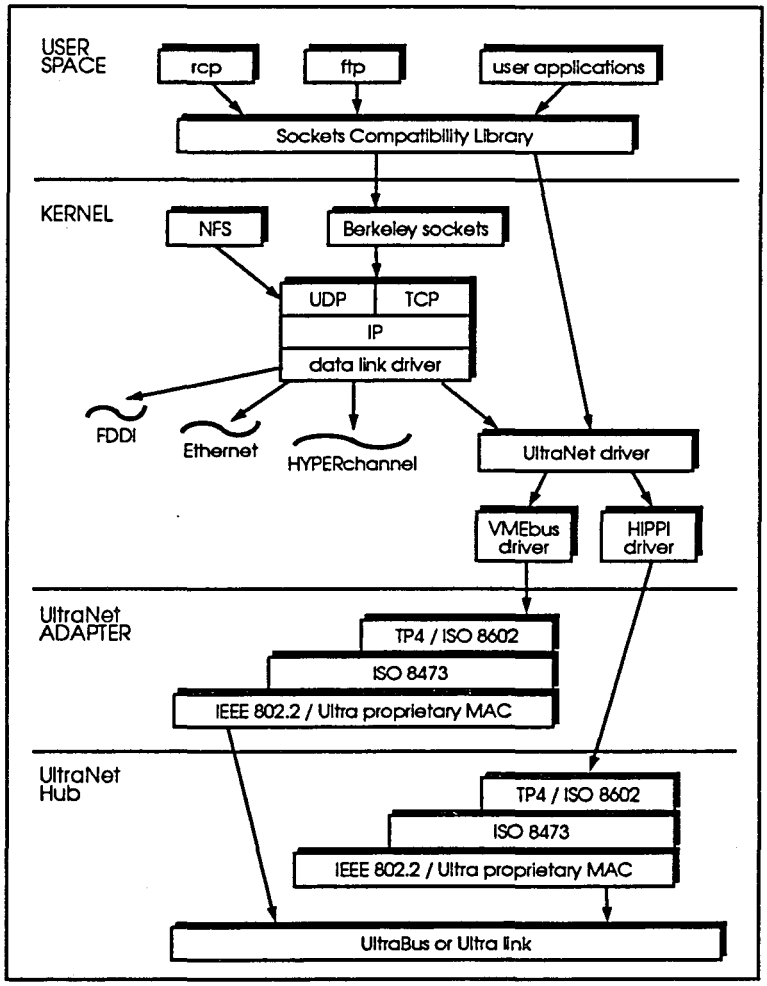
Because network installations often connect to multiple networks and need the full capabilities provided by TCP/IP protocols, UltraNet Interface software includes an internet interface module. An UltraNet network, therefore, has two network interfaces: a *native* UltraNet interface that uses proprietary protocols and is completely isolated from the internet, and a TCP/IP interface that is a full member of the internet.

The CONVEX UltraNet Interface can also be used with host-resident protocols to link hosts not directly connected to the UltraNet. This feature allows you to access the TCP/IP protocol stack using Ethernet, FDDI, HYPERchannel, or UltraNet as a data link.

The ability to run TCP/IP protocols over the CONVEX UltraNet Interface and to use UltraNet as a data link is provided by the Sockets Compatibility Library, discussed in the section, "Application program interface," on page 93.

A model of CONVEX UltraNet Interface network architecture is shown in Figure 25 on page 92.

Figure 25 UltraNet network architecture



As Figure 25 illustrates, the UltraNet Adapter provides the lower four OSI layers. The Adapter, rather than the CONVEX host, handles protocol processing, data flow, error processing, and network management. This design helps the UltraNet Interface achieve its high performance by completely eliminating packet processing within the CONVEX host.

Services

The CONVEX UltraNet Interface includes two high-speed transfer facilities:

- `rcp`—Used to transfers files between ConvexOS or UNIX systems that run BSD networking.
- `ftp`—Used primarily to transfer files to and from systems that run TCP/IP protocols, but not BSD networking.

CONVEX UltraNet also supports standard TCP/IP protocols for virtual terminals and provides users with access to file systems via NFS.

Application program interface

CONVEX UltraNet Interface software includes the *Sockets Compatibility Library* to allow execution of both BSD applications and socket-based user programs. By emulating the socket interface, UltraNet software allows network applications to take full advantage of UltraNet's high performance through a standard application program interface, Berkeley sockets.

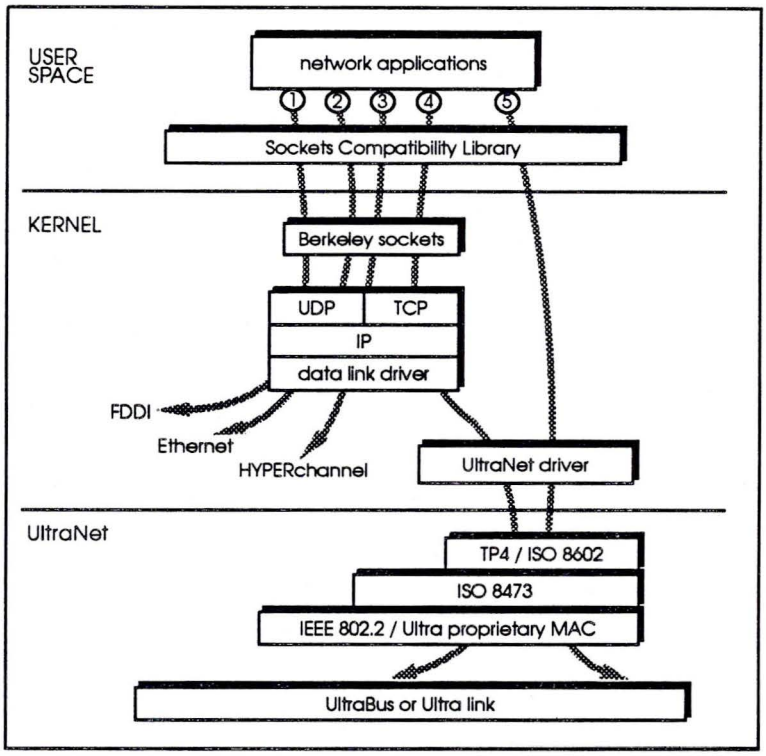
The Sockets Compatibility Library is a C language library that enables existing applications to run over either DARPA Internet TCP/IP or UltraNet TP4 protocols without requiring changes to the software. It also provides programmers with a facility for writing network-independent programs.

In addition to emulating Berkeley sockets, the Sockets Compatibility Library includes a multiplexing facility that provides socket-based processes with transparent access to standard sockets, UltraNet sockets, or both.

UltraNet software uses a packet's destination network address to determine which set of protocols to use and which path to take through the networking software. If the destination address is the one assigned to the native UltraNet interface, the software uses UltraNet proprietary protocols and takes the path over the UltraNet Interface. If the destination address is assigned to an Ethernet, FDDI, or HYPERchannel interface, the software uses TCP/IP protocols, complete with full Internet functionality, over the Ethernet, FDDI, or HYPERchannel interface.

The use of UltraNet as a data link creates yet another path through the UltraNet software. The arrows in Figure 25 on page 92 trace each possible path.

Figure 26 Multiple data paths over UltraNet



In the above diagram

- Path 1 uses TCP/IP protocols over FDDI as a data link.
- Path 2 uses TCP/IP protocols over Ethernet as a data link.
- Path 3 uses TCP/IP protocols over HYPERchannel as a data link.
- Path 4 uses TCP/IP protocols over UltraNet as a data link.
- Path 5 uses UltraNet proprietary protocols over UltraNet, bypassing the TCP/IP protocol stack entirely.

To use the Sockets Compatibility Library, users link network programs with `libulsock.a`. By default, programs are linked with standard sockets in `libc.a`; for a program to use UltraNet sockets, you must specify `libulsock.a` with the `-l` option when you link it.

Configuration and management

CONVEX UltraNet Interface software provides network configuration and management tools that enable the system manager to

- Load VME/UltraNet Adapter firmware.
- Assign names and addresses to hosts on the UltraNet network.
- Define UltraNet adapters.
- Create and maintain mappings between internet addresses and UltraNet station addresses.
- Define and maintain UltraNet routes.
- Test and exercise network connections.
- Gracefully shut down network connections for repairs and maintenance.
- Gather and display statistics on network performance and stability.

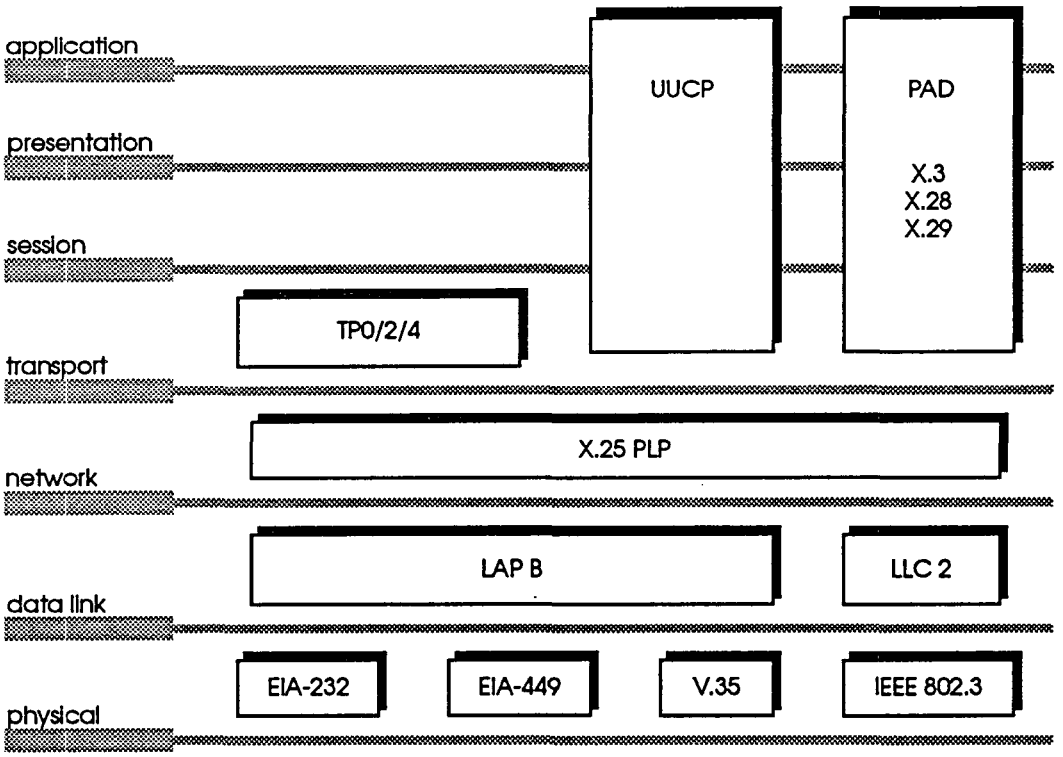
The TCP/IP interface provided by UltraNet Interface software is configured, maintained, and monitored by using the network management tools included with CONVEX Internet Services. Installation of CONVEX Internet Services is prerequisite to using the UltraNet Interface.

Documentation

The CONVEX UltraNet Interface product is documented in the manuals listed below.

- *CONVEX Internet Services User's Guide* describes how to perform basic networking tasks, such as logging into remote systems, copying files between systems, and executing commands on remote systems.
- *CONVEX Internet Services System Manager's Guide* contains information for administering systems on a CONVEX UltraNet or TCP/IP network. Many of the installation and configurations tasks this book describes are necessary for the operation of NFS and related facilities.
- *CONVEX Interprocess Communication (IPC) Programming Guide* describes additional facilities for implementing network applications.
- *ConvexOS Man Pages for Users, ConvexOS Man Pages for Programmers,* and *ConvexOS Man Pages for System Managers,* contain man pages describing ConvexOS software.
- *CONVEX UltraNet Programmer's Reference* contains man pages describing CONVEX UltraNet Interface software.

CONVEX OSI WAN Transport



CONVEX OSI WAN Transport 7

Product description

The CONVEX OSI WAN Transport product allows CONVEX supercomputers to connect to X.25 Packet Switched Data Networks that comply with CCITT and ISO recommended standards. The OSI WAN Transport can be used with both wide area and local area networks. By using the OSI WAN Transport, CONVEX users and application programs can communicate with remote users and remote systems. Interactive users of the CONVEX system can also access remote systems using X.3, X.28, and X.29 Packet Assembler/Disassembler (PAD) emulation features.

CONVEX OSI WAN software uses one or more serial communication controllers to connect to the packet data network or Ethernet controllers to connect to IEEE 802.3 networks.

Prerequisites

To use the CONVEX OSI WAN Transport, your system must be running ConvexOS and Utilities.

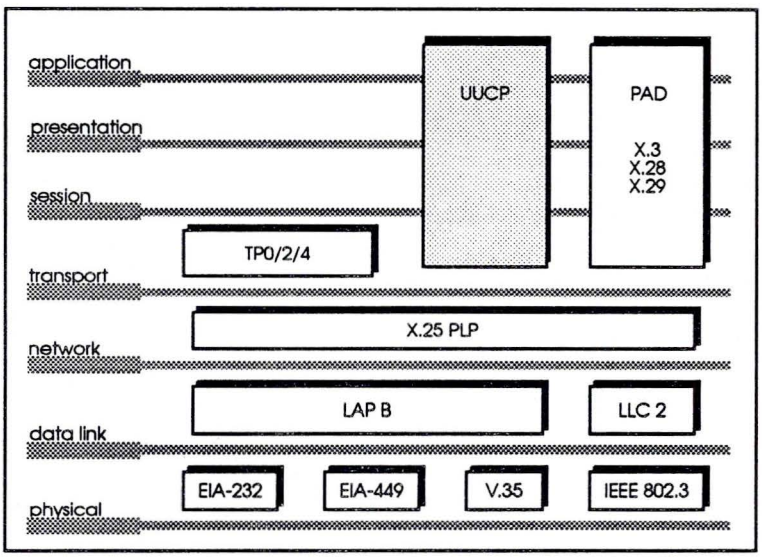
The CONVEX OSI WAN Transport runs over any of the following communication controllers, available as separate hardware products:

- For operation of X.25 over a synchronous network, the OSI WAN Transport requires a Multibus SBE/COM-4 Synchronous Communication Controller and an SBE/SCI-232, SBE/SCI-449T, or SBE/SCI-V35P Serial Communication Interface Module.
- For operation over an Ethernet-based network, either an Excelan Multibus Ethernet Controller or an Excelan VMEbus Ethernet Controller must be installed.

Protocols, profiles, and standards conformance

In keeping with CONVEX's commitment to Open Supercomputing, the CONVEX OSI WAN Transport product complies with Open Systems Interconnection (OSI) architecture, and conforms to several government profiles and international standards and recommendations. Figure 27 shows CONVEX OSI WAN Transport protocols mapped to the OSI model. (UUCP, shown in the shaded box, is included with ConvexOS and Utilities.)

Figure 27 CONVEX OSI WAN Transport protocols mapped to the OSI model



Government profiles and specifications

The CONVEX OSI WAN Transport conforms to the following profiles and specifications:

- Version 1 of the U.S. Government OSI Profile (GOSIP) as specified in the Federal Information Processing Standard, FIPS-146
- U.K. GOSIP
- U.K. Joint Academic Network (JANET) Coloured Books
- X.25 certified with FIPS-100 and NET 2 CEPT
- Transport layer classes 0/2/4
- Link Access Procedure-Balanced (LAPB)
- Logical Link Control type 2 (LLC2)
- X.3, X.28, and X.29 Packet Assembler/Disassembler (PAD)

ISO standards and CCITT recommendations

The CONVEX OSI WAN Transport conforms to standards and recommendations specified in the documents listed below.

Adopted ISO standards have the prefix "ISO." The "DIS" prefix stands for "Draft International Standard" and indicates that the standard is in the final stage of approval.

General ISO references

ISO 7498, Information Processing Systems—Open Systems Interconnection—*Basic Reference Model*, October 1984

NBS-SP 500-162, *Stable Implementation Agreements for OSI Protocols*, Version 2, Edition 1, December 1988

MAP 3.0, *Manufacturing Automation Protocol Specification*, Version 3.0

TOP 3.0, *Technical and Office Protocol Specification*, Version 3.0

GOSIP, U.S. Government Open Systems Interconnection Profile, Version 1, June 1988

Transport layer

ISO 8072, Information Processing Systems—Open Systems Interconnection—*Transport Service Definition*, June 1986

ISO 8073, Information Processing Systems—Open Systems Interconnection—*Transport Protocol Specification*, July 1986

ISO 8072/AD1, *Addendum 1 to the Transport Service Definition Covering Connectionless-mode Transmission*, July 1986

ISO 8073/DAD2, *Addendum 2 to the Transport Protocol Definition Covering Class Four Operation over Connectionless Network Service*, July 1987

ISO 8602, Information Processing Systems Data Communications—*Protocol for Providing the Connectionless-mode Transport Service*, July 1987

CCITT X.224, 1988 Version, Recommendation X.224, Transport Service Definition for Open Systems Interconnection (OSI) for CCITT Applications

Network layer

ISO 8348, Information Processing Systems—Data Communications—*Network Service Definition*, April 1987

ISO 8348/AD1, Information Processing Systems—Data Communications—*Network Service Definition Addendum 1: Connectionless-mode Transmission*, April 1987

ISO 8348/AD2, Information Processing Systems—Data Communications—*Addendum to the Network Service Definition Covering Network Layer Addressing*, March 1988

ISO 8648, Information Processing Systems—Data Communications—*Internal Organization of the Network Layer*, February 1988

ISO 8208, Information Processing Systems—Data Communications—*X.25 Packet Level Protocol*, September 1987

ISO 8878, Information Processing Systems—Data Communications— *Use of X.25 to Provide the Connection-mode Network Service*, September 1987

CCITT Red Book, Volume VIII, Fascicle VIII.3, Recommendation X.25, *Interface Between DTE and DCE for Terminals Operating in the Packet Mode and Connected to Public Data Networks by Dedicated Circuit*

ISO 8473, Information Processing Systems—Data Communications—*Protocol for Providing the Connectionless-mode Network Service*, January 1988

Data link layer

ISO 7776, Information Processing Systems—Data Communications—2nd DP 7776 Revised—*Description of the 1984 X.25 LAPB—Compatible DTE Data Link Procedures*, December 1986

ISO/DIS 8886.3, Information Processing Systems—Data Communications—*Data Link Service Definition for Open Systems Interconnection*, June 1988

ISO/DIS 8802-2.2, Information Processing Systems—Local Area Networks Part 2: *Logical Link Control*, October 1987

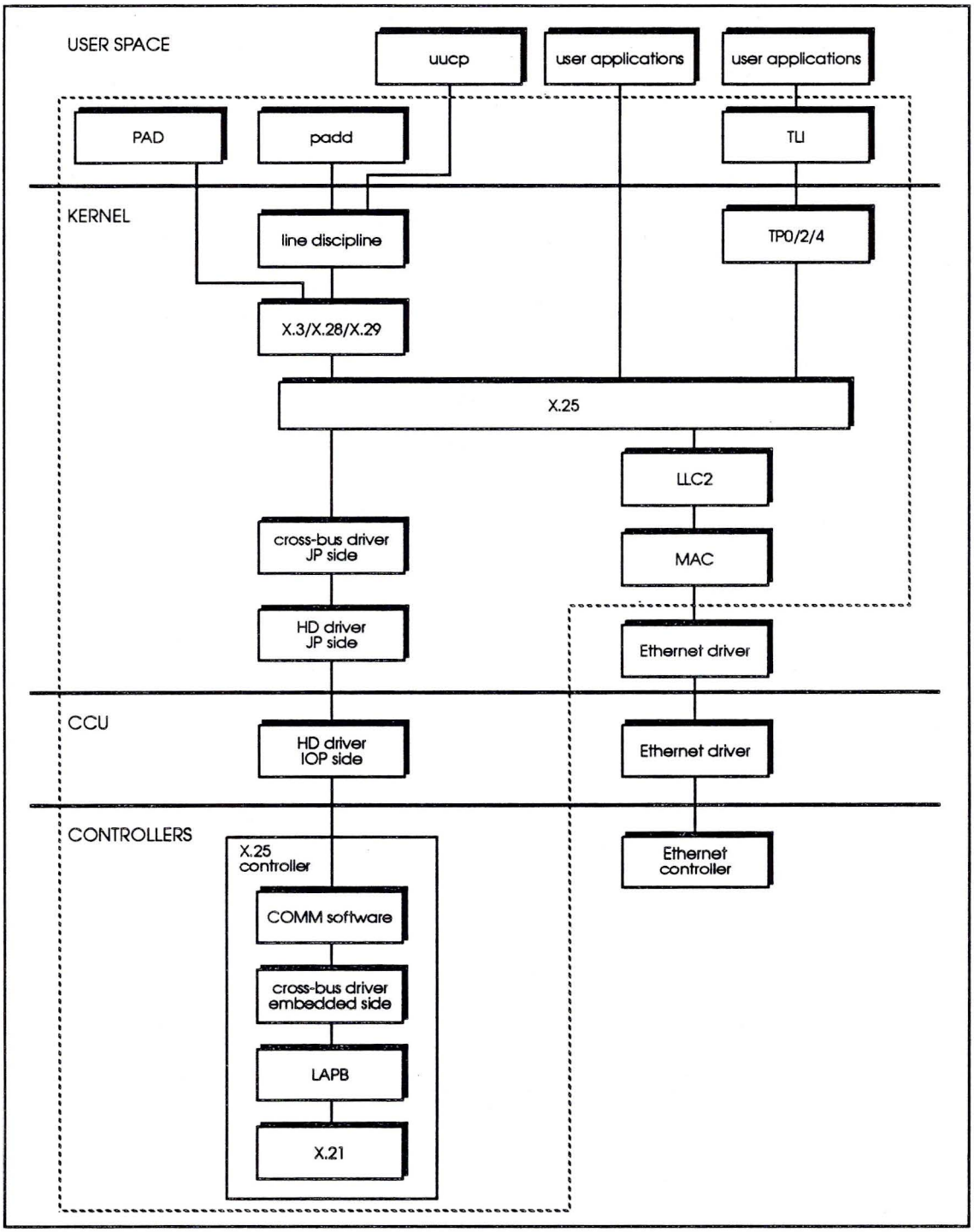
Network architecture

CONVEX OSI WAN Transport protocol modules are structured using the standard UNIX STREAMS mechanism. STREAMS defines a set of routines, data structures, and processing rules for data flow and messaging systems. Streams are well suited for implementation of multi-layer network protocols because of the modularity and structure they impose. Protocol layers are grouped into processing components (modules) of a stream.

A stream consists of a series of stream modules linked between the stream head and a stream pseudo or physical device driver. The sequence of modules in a stream can be dynamically configured. The stream head provides the interface between a user space application and the kernel space protocol modules by converting streams system calls into messages for the downstream modules and by relaying messages from the protocol modules upstream to the application.

Figure 28 on page 104 illustrates the network architecture of the OSI WAN Transport product in terms of streams modules and their distribution across the various processors in a CONVEX system. OSI WAN Transport components are shown within the dashed lines.

Figure 28 OSI WAN Transport network architecture



As Figure 28 shows, the OSI WAN Transport protocol stack is split into three components:

1. Application programs execute in user space.
2. Transport, network, and logical link layers are implemented as Stream modules in the ConvexOS kernel.
3. Data link and physical layers reside on communication controllers.

Services

The CONVEX OSI WAN Transport product supports two application-level services: PAD and UUCP.

Packet Assembler/Disassembler (PAD)

PAD provides an interface between a remote terminal and an X.25 network. PAD commands enable users to log in to hosts on an X.25 network and use the remote host as if it were local.

The `pad` program functions as the client side of a virtual terminal connection. Its primary function is to transfer data between a user terminal and the X.25 network. Its corresponding server, `padd`, receives incoming call requests from remote PADs and establishes a connection between the remote PAD and the local host.

UNIX-to-UNIX-Copy (UUCP)

PAD software supports the ability to run UUCP utilities over X.25 networks. UUCP utilities are provided by ConvexOS to enable users to copy files between systems and execute commands on remote systems using dial-up or hardwired communication lines.

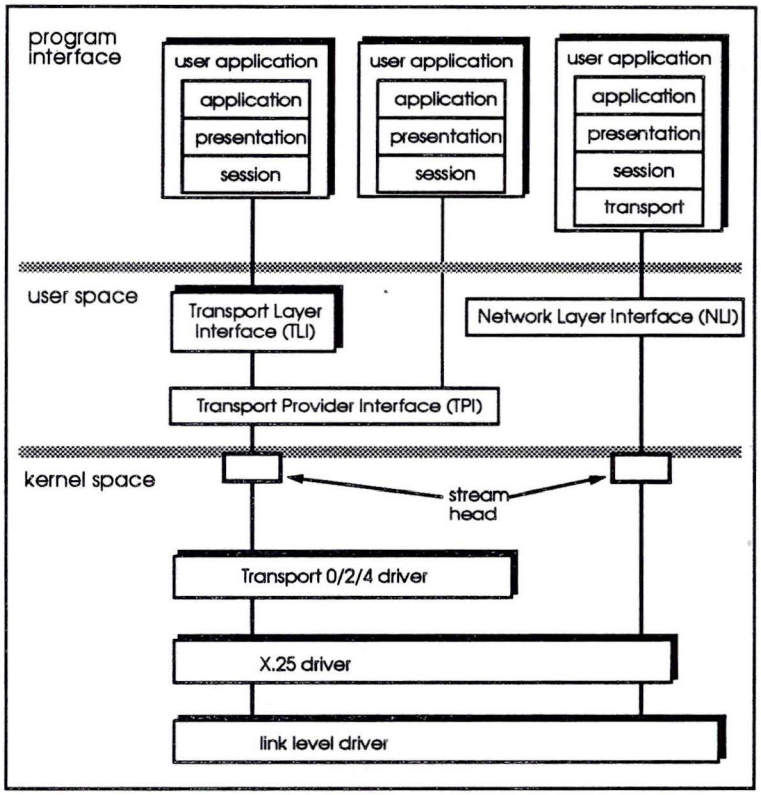
UUCP serves primarily as a transport mechanism for electronic mail and news. Normally, users do not take advantage of UUCP facilities directly; most UUCP services are transparent to the user.

Application program interface

CONVEX OSI WAN Transport supports the STREAMS application program interface described in the section, "STREAMS," on page 41. TLI and TPI are independent of any specific network or transport layer protocol; NLI is specific to X.25.

Figure 29 shows the STREAMS program interface to the OSI WAN Transport.

Figure 29 OSI WAN Transport application program interfaces



In the figure above, TPI and NLI are shown as flat boxes because they define the interfaces to the transport and network layers; they do not include library routines.

Transport Layer Interface (TLI)

TLI is implemented as a user-space library. The TLI library provides a high-level interface to the OSI transport layer primitives. TLI functions use standard STREAMS I/O system calls to send data to and receive data from the WAN Transport layer. TLI handles all communications between STREAMS and user applications, freeing application programmers from having to program STREAMS system calls directly.

The CONVEX TLI Library is based on the X/Open Transport Interface Issue 3 (XTI 3) and the System V Interface Definition Issue 4 (SVID 4).

Transport Provider Interface (TPI)

TPI is implemented through standard STREAMS system calls for communicating with the stream head. Application programs access transport services by creating a stream to the transport drivers and using `getmsg` and `putmsg` calls to exchange messages with the transport provider. Transport drivers communicate with a user-level program through the stream head.

Network Layer Interface (NLI)

NLI is also provided through the standard STREAMS system calls for communicating with the stream head. Applications access X.25 services by creating a stream to the X.25 drivers, and using `getmsg` and `putmsg` calls to exchange messages with the X.25 module. X.25 drivers communicate with a user-level program through the stream head.

Configuration and management

CONVEX OSI WAN Transport includes a set of configuration files, utilities, and tunable parameters used to customize the OSI WAN Transport for a particular network configuration or application and to isolate network problems.

OSI WAN Transport configuration and management utilities enable the system manager to

- Download controller firmware and reset controller boards from the SPU.
- Build streams stacks and configure protocol modules.
- Manipulate configuration parameters for the `lapb` streams driver, `llc2` streams multiplexor-driver, `tp024` streams multiplexor-driver, `x21` streams driver, and `x25` streams driver.
- Display statistics maintained by the streams modules and drivers present in the protocol stack.
- Test the installation and configuration of the OSI transport stack by sending and receiving data.
- Create and maintain information about known hosts and subnets.
- Allocate streams resources and start the streams scheduler.
- Display detailed statistics about streams resource utilization.
- Configure PAD.
- Configure UUCP to run over X.3, X.28, and X.29 protocols.
- Customize memory utilization of the protocol modules and the transport module, and define memory requirements such as the number of data blocks, event cells, and timeout cells.

Documentation

The CONVEX OSI WAN Transport product includes the manuals listed below.

- *CONVEX OSI WAN Transport Programmer's Guide* describes how to develop applications that interface with the CONVEX OSI WAN Transport.
- *CONVEX OSI WAN Transport Administrator's Guide* describes how to configure and run the CONVEX OSI WAN Transport.
- *CONVEX OSI PAD User's Guide* explains how to use the `pad` program to communicate with remote hosts.
- *CONVEX TLI Library Programmer's Guide* describes how to use TLI to provide transport layer services for network applications.

Each of the above manuals includes an appendix containing man pages for the software product.

For more information about transport-level protocols and interfaces, refer to the following reference documents:

- *Information Processing Systems, Open Systems Interconnection, Basic Reference Model* (IS:1984). ISO 7498.
- *ISO Transport Connection-mode Protocol Definition.* IS 8073-1986.
- *ISO Transport Connection-mode Service Definition.* IS 8072-1986.
- *ISO Transport Connectionless-mode Protocol Definition.* IS 8602.
- *ISO Transport Connectionless-mode Service Definition.* IS 8072/Add.1-1986.
- X/Open Company Limited. *Networking Services.* Vol. 7 of the *X/Open Portability Guide.* ISBN 0-13-685892-9
- AT&T. *Base System and Kernel Extensions.* V Interface Definition, Third Edition, Vol.1 of System V Interface Definition. 320-136.

Glossary

a

abortive release
Occurs when a transport user issues a disconnect request to terminate a connection, possibly resulting in loss of data.

active user
A user that initiates a connection-mode service.

address
A unique number or character string that identifies a particular network node. Also called a *network address, host address,* and *internet address.*

address class
An attribute of a DARPA Internet network that indicates the size of the network and limits the network's name space. For example, class C networks are limited to a maximum of 254 hosts.

address resolution
Mechanism for mapping host names to network addresses, or network addresses to physical addresses.

Address Resolution Protocol (ARP)
TCP/IP protocol that maps internet addresses to physical addresses.

American National Standards Institute (ANSI)
A repository and coordinating agency for standards implemented in the U.S. Its activities include the production of Federal Information Processing (FIPS) standards for the Department of Defense (DoD).

ANSI
See *American National Standards Institute.*

application layer
Protocol layer that provides system-independent services for end-user applications, such as electronic mail and file transfers.

application-level services
Service that enable application programs to take advantage of network facilities. Also called *end-user services*.

Application Program Interface (API)
A set of system calls and library routines that provide programmers with access to network services.

ARP
See *Address Resolution Protocol*.

ARPANET
The Advanced Research Project Agency Network, now called the Defense Advanced Research Projects Agency (DARPA) Internet. ARPANET was funded by the U.S. government to serve as a testbed for internetworking technology. TCP/IP protocols were developed as part of this project.

asynchronous mode
Execution mode in which control returns to a user immediately following a library function call even if the corresponding asynchronous event has not occurred.

b

backbone
A central network used to interconnect LANs.

BCUG
See *Bilateral Closed User Group*.

Berkeley Internet Name Domain (BIND)
A service that enables clients to name objects and services on the network and share those names with other clients. In effect, BIND is a distributed database system for objects in a network.

Berkeley sockets
Interprocess Communication (IPC) facilities provided by a set of system calls and library routines for use in writing applications involving more than one task.

Bilateral Closed User Group (BCUG)
Similar to a CUG (Closed User Group), but restricts access between pairs of DTEs.

BIND
See *Berkeley Internet Name Domain*.

bridge
A node used to transparently connect networks, usually of the same type, at the physical layer.

C

call request
An X.25 term that is analogous to initiating a connection request in general OSI terminology.

called address
Address of the destination host.

calling address
Address of the source host.

Call User Data (CUD)
Network service parameter carried with a call request. This parameter allows data to be sent with the call request.

CCITT
See *International Telegraph & Telephone Consultative Committee.*

circuit
A virtual communication path between nodes. Sometimes used to refer to a physical communication path.

client
The user of a service.

client/server model
The structure by which services are implemented. A client process on one host makes a request that a server process on another hosts fulfills.

client stub
Used in implementing remote procedure call (RPC) facilities, the client stub takes the place of the actual called procedure in the client to abstract the details of passing messages over the network.

Closed User Group (CUG)
An X.25 facility that allows a DTE to belong to one or more groups in which grouped DTEs can communicate with each other but not with other DTEs.

Comité Consultatif International de Télégraphique et Téléphonique (CCITT)
See *International Telegraph & Telephone Consultative Committee.*

communication line
A physical communication path, such as coaxial or fiber-optic cable.

communication link
A logical communication path consisting of the hardware and software needed to establish a connection and transfer data between nodes. Also called a *logical link* or *virtual circuit.*

computer network
A system of interconnected computers that enables machines and their users to exchange information and share resources.

confirmation
An event that usually results when a peer entity issues a response, such as the acknowledgment of a connection request.

connection establishment
The connection-mode service phase in which two peer processes set up a communication link.

connectionless mode
A transport service mode in which independent units of data are transferred between peer entities.

connectionless service
A service that does not require a continuous communication link between end nodes. Instead, data is sent in independent units addressed to the destination node.

connection-oriented mode
A service mode in which two peer entities establish a communication link prior to exchanging data. Also called *connection mode.*

Connection-Oriented Network Service (CONS)
A network-layer service that uses connection-oriented mode to provide services to NSUs. 1984 X.25 PLP is an example of a CONS provider.

connection-oriented service
A service that requires two peer entities to establish a communication link prior to exchanging data.

connection release
The connection-mode service phase in which two processes terminate their connection.

CONS
See *Connection-Oriented Network Service.*

Corporation for Open Systems (COS)
COS is an organization composed of major suppliers of data processing and data communications products founded to advance the use of international standards. COS has been instrumental in the development of procedures for certifying compliance of communication systems with international standards.

COS
See *Corporation for Open Systems.*

COVUEnet
A CONVEX networking product that enables CONVEX systems to participate as end nodes on a DECnet-COVUEnet network.

CUD
See *Call User Data.*

CUG
See *Closed User Group.*

d

D-bit
Used to guarantee delivery of X.25 packets. If the D-bit is set in a received packet, packet acknowledgment is delayed until the packet is delivered to the remote DTE. The D-bit is a location in the packet type identifier field of an X.25 header.

daemon
A process that executes continuously to provide a service on an as-requested basis.

DARPA
See *Defense Advanced Research Projects Agency.*

DARPA Internet
A group of networks interconnected through the use of TCP/IP protocols.

DARPA Internet protocols
TCP/IP protocol suite developed during research on internetworking funded by the U.S. government.

Data Circuit-Terminating Equipment (DCE)
One of two systems that form a connection (the other is DTE). The DCE, or network interface, conveys the data received from one DTE to another.

datagram
In connectionless-mode service, the independent unit of data exchanged between two peer transport users.

datagram socket
IPC facility that provides a bidirectional flow of data that is not promised to be sequenced, reliable, or unduplicated. Applications send and receive data from many different sockets without establishing connections first.

data link layer
Layer of the OSI model responsible for transmitting data over a communication link, including error detection, correction, and recovery functions. Often divided into two sublayers: medium access control (MAC) and logical link control (LLC).

Data Network Identification Code (DNIC)
Part of a DTE address that identifies a specific data network.

Data Terminal Equipment (DTE)
One of two systems that comprise an connection (the other is DCE). The DTE, or host, establishes and terminates a network connection.

data transfer
A phase in both connection-mode and connectionless-mode services in which two peer transport users exchange data.

DCE
See *Data Circuit-Terminating Equipment.*

DECnet
A network used to interconnect computers that run DNA protocols.

Defense Advanced Research Projects Agency (DARPA)
The U.S. government agency that funded research on internetworking that led to the development of the TCP/IP protocol suite. See *ARPANET, DARPA Internet.*

Destination Service Access Point (DSAP)
The link service access point of the destination link service user.

Digital Network Architecture (DNA)
Digital Equipment Corporation's proprietary network architecture.

DNA
See *Digital Network Architecture.*

DNIC
See *Data Network Identification Code.*

DNS
See *Domain Name System.*

DoD
U.S. Department of Defense.

domain
A portion of name space that can be identified by a label, such as .edu in the host name, foobar.edu.

Domain Name System (DNS)
A distributed database service that enables clients to name objects and services on the network and share those names with other clients. BIND is an implementation of DNS. See *Berkeley Internet Name Domain.*

dot notation
The common practice of representing internet addresses as a group of four 8-bit decimal numbers separated by dots, as in 128.50.10.1.

downstream
Refers to the direction from a stream head toward a streams driver. This is the direction of data flow that results from a `write` or `putmsg` operation.

driver
In general, a driver is the software that controls a physical device, such as a network interface. In OSI terminology, a driver is an entity in a streams protocol stack used to multiplex data between protocols. Streams protocol stacks have at least one device driver, which is located at the stream's end and is the interface to the hardware. Drivers are accessed through a node (or nodes) in the file system.

DSAP
See *Destination Service Access Point.*

DTE
See *Data Terminal Equipment.*

---

e

ECMA
See *European Computer Manufacturers Association.*

EIA
See *Electronics Industry Association.*

EIA-232
EIA specification for a widely-used physical interface. It describes the functions for a 25-pin connector.

EIA-422A
EIA specification for a balanced electrical interface.

EIA-449
EIA specification for a physical interface having a greater range and higher data rate than EIA RS-232. It describes the functions for a 37-pin connector.

EIA-423A
EIA specification for a unbalanced electrical interface.

Electronics Industry Association (EIA)
A national trade association concerned primarily with the development of hardware-level standards. EIA standards are identified by the letters EIA, followed by a hyphen and a number, such as EIA-232, a standard used for connecting terminals to computers.

electronic mail
A facility that allows users to communicate information across a network in a way similar to the traditional exchange of memos and correspondence through interoffice mail or the postal system.

email
See *electronic mail*.

end node
Terminals or workstations from which users request network services and hosts that process those requests.

endpoint
The point of communication between a service user and a connection. Each connection has two endpoints.

end system
Terminals or workstations from which users request network services and hosts that process those requests.

end user
The person or program that requests a service.

end-user services
Services that enable people or application programs to take advantage of network facilities.

entity
An addressable unit of software or hardware that provides a service to or makes use of a service provided by another addressable unit.

Ethernet
A local area network that interconnects nodes via coaxial cable. It uses the CSMA/CD (Carrier Sense Multiple Access/Collision Detection) access method and transmits at 10 megabits per second. Ethernet has evolved into the IEEE 802.3 standard.

European Computer Manufacturers' Association (ECMA)
ECMA works closely with ISO and CCITT toward developing standards for data processing and data communication. ECMA includes all European computer manufacturers.

event
In OSI terminology, the transfer of data from a service provider to a service user.

executor
Node at which Network Control Program (NCP) commands are executed. The executor can be the local node or a remote node.

expedited data
Data considered significant and not subject to normal flow control for the connection. Specific handling is defined by individual transport providers.

eXternal Data Representation (XDR)
A facility consisting of a set of library routines that provides a common way of representing data types over a network. XDR allows applications to transfer data between diverse machines, such as Sun Workstations, VAX, IBM-PC, and CONVEX machines.

---

f

FDDI
See *Fiber Distributed Data Interface.*

Fiber Distributed Data Interface (FDDI)
A network interface that connects CONVEX computers directly to FDDI networks. FDDI offers the next level of LAN performance beyond Ethernet, with a peak data rate of 100 megabits per second. The FDDI standard specifies a fiber transmission medium and a token ring topology.

file server
Software that makes local file systems available to remote clients.

File Transfer, Access, and Management (FTAM)
An ISO application-level service that allows users to transfer and manipulate files between hosts.

File Transfer Protocol (FTP)
TCP/IP protocol that allows users to transfer files between hosts.

frame
Data and link-level control information that is transmitted over a network. A packet is carried within the data portion of the frame.

FTAM
See *File Transfer, Access, and Management.*

FTP
See *File Transfer Protocol.*

g

gateway
Device used to interconnect networks with incompatible architectures, protocols, and addressing schemes.

GOSIP
See *Government OSI Profile.*

Government OSI Profile (GOSIP)
Identifies a set of standard OSI protocols with which networking systems used by U.S. government agencies must conform.

h

hardware address
The device-dependent physical address of a node attached to a communication line.

High Performance Parallel Interface (HIPPI)
Currently the fastest industry standard for connecting high-performance computers. HIPPI hardware consists of HIPPI channel control unit (CCU) that supports dual simplex connections (one input, one output) to provide a data rate of 800 megabits per second over distances of to 25 meters.

HIPPI
See *High Performance Parallel Interface.*

host
A computer system that supports network applications. See *node.*

HYPERchannel
High-speed network interface that supports TCP/IP protocols. Software support for HYPERchannel is provided by CONVEX Internet Services.

i

ICMP
See *Internet Control Message Protocol.*

IEEE
See *Institute for Electrical and Electronic Engineers*

IEEE 802.2
IEEE LAN standard that specifies the data link layer for the CSMA/CD (Carrier Sense Multiple Access/Collision Detection), token passing bus, and token passing ring access methods.

IEEE 802.3
IEEE LAN standard that specifies the physical layer for the CSMA/CD (Carrier Sense Multiple Access/Collision Detection) access method. See *Ethernet.*

I-frame
LAPB information frame; X.25 packets are transferred as the information field of an I-frame.

initiator
In connection-mode service, a transport user that requests a connection to a peer user. See *active user, client.*

inode
A data structure containing information about a file, such as ownership, permissions, and the file's location on disk. An inode exists for every file accessible to the CONVEX system.

Institute for Electrical and Electronic Engineers (IEEE)
An international professional organization and a member of ANSI and ISO. IEEE created Project 802, the committee that developed a set of widely-used LAN standards known as the 802 standard.

interface
(1) A logical data path between adjacent layers of the OSI model.
(2) A logical path between any two modules or systems. (See *network interface.*)
(3) As a verb, interface means to interconnect to or interoperate with.

intermediate node
A networked machine responsible for transferring data along the path between end nodes. See *end node.*

International Organization for Standardization (ISO)
An international regulatory body for information processing and communication systems. Among other achievements, ISO is responsible for the design of the OSI model. This organization is often referred to as the International Standards Organization.

International Standards Organization
See *International Organization for Standardization.*

International Telegraph & Telephone Consultative Committee (CCITT)
An international organization that sponsors standards for data networks, telephone switching, digital systems, and terminals. You will often see this organization referred to as the Consultative Committee for International Telephony and Telegraphy, to match the acronym for its French name, Comité Consultatif International de Télégraphique et Téléphonique.

internet
When shown in lowercase, any interconnection of autonomous networks. When written with an initial capital letter, usually refers specifically to the DARPA Internet. See *DARPA Internet.*

internet address
A 32-bit number that identifies a specific node on a TCP/IP network. Usually written in dot notation, as in 128.40.10.1.

Internet Control Message Protocol (ICMP)
TCP/IP protocol responsible for relaying error messages detected by gateways back to the source node.

Internet Protocol (IP)
TCP/IP protocol that encapsulates packets into datagrams and delivers them from one machine to another. IP also provides addressing and routing services.

Internet Services
CONVEX networking product that provides TCP/IP-based services over Ethernet, FDDI, and HYPERchannel interfaces and over serial lines.

internetwork
See *internet.*

internetworking
The art and science of interconnecting diverse networks.

interoperability
The ability of network components to communicate.

Interprocess Communication (IPC)
A set of system calls and library routines that gives application programmers access to the full power and functionality of the TCP/IP protocol suite. Through UNIX domain sockets and shared memory, IPC also enables programs to communicate with other programs running on the same machine.

IP
See *Internet Protocol.*

IPC
See *Interprocess Communication.*

ISO
See *International Organization for Standardization.*

ISO reference model
See *OSI model.*

j

JANET
See *Joint Networking Team.*

Joint Networking Team (JANET)
United Kingdom regulatory body for data communication standards.

k

kernel
The core of the operating system where basic system facilities, such as file access and memory management functions, are performed.

l

LAN
See *Local Area Network.*

LAPB
See *Link Access Procedure Balanced.*

line
A physical communication path between nodes, such as that provided by an Ethernet cable. Also called a *communication line.*

link
A virtual communication path between processes running on different nodes. Also called a *communication link* or *logical link.*

Link Access Procedure Balanced (LAPB)
Data link layer protocol used over a physical link between DTE and DCE in an X.25 packet-switched network.

Link Service Access Point (LSAP)
An address or identifier through which a service user accesses data link layer services.

listener
Generally, a user application that manages multiple transport endpoints by waiting on (or listening to) each port for incoming connection requests.

listening endpoint
A transport endpoint that has been bound for listening.

LLC
See *Logical Link Control.*

Local Area Network (LAN)
A network that serves several users at relatively high speeds within a small geographic area.

logical channel number
In packet-switched networks, a number assigned to a virtual connection.

logical link
See *link.*

Logical Link Control (LLC)
Data link layer protocols for operation over a LAN.

long haul network
See *Wide Area Network.*

LSAP
See *Link Service Access Point.*

m

MAC
See *Media Access Control.*

MAN
See *Metropolitan Area Network.*

M-bit
More-data bit, used to implement fragmentation and reassembly of X.25 packets. The M-bit is a location in the packet type identifier field of an X.25 header.

Media Access Control (MAC)
Component of the data link layer that controls access to the physical medium.

message
An arbitrarily long unit of data, such as the contents of a file, that can be transmitted over a network. The size of a message is not limited by the size of buffers used within the protocol stack to transmit the message.

Message-Handling System (MHS)
See *X.400 Message-Handling System.*

Message Transfer Agent (MTA)
A component of the X.400 Message-Handling System, the MTA is responsible for relaying messages initiated by the user agent (UA).

Metropolitan Area Network (MAN)
A medium-scale network capable of providing large corporate customers with the ability to transfer massive amounts of data within a service area roughly the size of a city. MANs use LAN technology to provide data rates faster than wide area networks, which rely on regular telephone switching technology.

MHS
See *X.400 Message-Handling System.*

modem eliminator
A device used to connect a local terminal and a computer port without the pair of modems normally used with synchronous links.

module
A STREAMS component that performs intermediate transformations on messages flowing between the Stream head and the driver.

MTA
See *Message Transfer Agent.*

multi-homed host
A host attached to multiple network interfaces. Such hosts are assigned one network address per interface. Multi-homed hosts are often used as gateways through which networks are interconnected.

named pipe
Pipes that exist permanently in the file system with directory entries and path names. Because you can access these pipes by name, you can use them for a variety of applications that you cannot accomplish with ordinary pipes. Typically, named pipes are used to allow a number of processes to communicate with a daemon.

name server
A system that provides distributed database facilities for hosts in a network. See *Berkeley Internet Name Domain, Domain Name System,* and *Network Information Service.*

name space
The set of possible host names within a domain or within an address class.

National Institute of Standards and Technology (NIST)
Formerly known as the National Bureau of Standards (NBS), NIST is responsible for defining the set of standard protocols required for systems used by U.S. government agencies. NIST activities produced the Government OSI Profile (GOSIP).

NCP
See *Network Control Program.*

NETdisk
Allows a CONVEX NFS server to boot a diskless workstation. After the workstation has booted, NETdisk provides it with access to the files it needs, such as the /root and /swap directories.

network
A system of interconnected computers that enables machines and their users to exchange information and share resources.

network architecture
The logical organization of a networking system.

Network Control Program (NCP)
A facility provided with COVUEnet that enables the system manager to control, test, and monitor a DECnet-COVUEnet network.

Network File Access Routine System (NFARS)
An application program interface provided with the COVUEnet product that consists of a set of routines for accessing network file systems.

Network File System (NFS)
A system that provides transparent access to networked files over a TCP/IP-based network. NFS links together heterogeneous systems to share resources and files over local area and wide area networks.

Network Information Center (NIC)
The central authority responsible for assigning and maintaining addresses of networks and hosts on the DARPA Internet.

Network Information Service (NIS)
Formerly called the Yellow Pages (YP), NIS is a distributed network lookup service that eases the job of administering networked machines. By using NIS, password, group, and host information for an entire network may be maintained in a single database.

network interface
A logical path between an application and a physical network. A device driver and network interface board provide the network interface for a CONVEX host.

network layer
The OSI layer responsible for routing and relaying data from one node to another on the same network or across multiple networks.

Network Layer Interface (NLI)
Interface used by programmers to access the X.25 Packet Layer Protocol (PLP) driver. NLI describes a common format for messages transmitted over the network.

Network Service Access Point (NSAP)
An address or identifier through which a Network Service User (NSU) may access network layer services.

Network Service User (NSU)
A software entity that uses the services of the network layer.

network topology
A description of the organization of a network in terms of its components, interconnections, and geography.

Network User Identification (NUI)
An X.25 facility that enables the transmitting DTE to provide billing, security, or management information to the DCE on a per-call basis.

NFARS
See *Network File Access Routine System.*

NFS
See *Network File System.*

NIC
See *Network Information Center.*

NIS
See *Network Information Service.*

NIST
See *National Institute of Standards and Technology.*

NLI
See *Network Layer Interface.*

node
A computer attached to a network that initiates or facilitates the flow of data across the network.

NSAP
See *Network Service Access Point.*

NSAP address
An address used to identify a specific Network Service Access Point (NSAP) and a particular Network Service User (NSU).

NSU
See *Network Service User.*

NUI
See *Network User Identification.*

O

object
In COVUEnet-DECnet terminology, a process that receives requests for a logical link.

OpenConnect
A gateway product that allows CONVEX computers to communicate with nodes in an SNA network.

open systems
Systems that conform to any non-proprietary, publicly-available standards. In recent years, however, the term has come to mean only those systems that use the international standards for network architecture, as specified by the OSI model.

Open Systems Interconnection (OSI)
The ISO definition of a system that provides reliable, data-transparent, host-independent services.

orderly release
A transport service feature in which two cooperating transport users gracefully terminate a connection to avoid data loss.

OSI
See *Open Systems Interconnection.*

OSI model
Also known as the *OSI Reference Model* or the *ISO reference model,* the OSI model is an architectural framework for the development of standardized networking systems.

OSI reference model
See *OSI model.*

outstanding connect indication
A connect indication that a transport user has received but has not yet responded to.

p

packet
Data carried within the frame as the information field. In DARPA Internet terminology, packets are also called *datagrams.* See *datagram.*

Packet Assembler/Disassembler (PAD)
A user-level program that provides remote login capability over an X.25 network.

Packet Level Protocol (PLP)
A network layer protocol used for connection-oriented operation.

PAD
See *Packet Assembler/Disassembler.*

passive user
In connection-mode service, a transport user that waits for another user to initiate establishment of a connection.

PDN
See *Public Data Network.*

peer entities
Entities running in different nodes at the same layer of the OSI model between which virtual communication takes place. Also called *peer processes* or *peer transport users.*

peer processes
See *peer entities.*

peer protocols
Protocols running at the same OSI layer on different machines.

peer transport users
See *peer entities.*

permanent database
A COVUEnet term referring to data stored on disk that contains initial values for the volatile database.

Permanent Virtual Circuit (PVC)
A logical association between two physically separate X.25 DTEs. A call request/call connect exchange is not required.

physical address
The device-dependent physical address of a node attached to a communication line.

physical layer
OSI layer responsible for transmitting data bits over a specific physical medium. Physical layer protocols include EIA-232 and V.35.

pipe
A pair of file descriptors that provide the mechanism for a one-way flow of data.

PLP
See *Packet Level Protocol.*

port
The logical point through which data flows between a node and a network. A given port is often associated with a particular service.

presentation layer
The OSI layer concerned with the syntax of transmitted information. The presentation layer is responsible for the order and format of data, and for services such as data encryption.

profile
(1) An agreed-upon subset of standards that identifies services members of a group must implement to ensure interoperability with other members' systems. (2) A set of default settings for PAD parameters.

protocol
A set of data and message formats, and a set of procedures governing transmission that when complied with enable network components to interoperate. Protocols define a common way in which network components must transmit and interpret information.

protocol address
Within the context of an entire network, the identifier of a specific transport endpoint.

protocol stack
A layered set of protocols. Entities at each layer provide services to entities at the next higher layer. Also called a *protocol suite.*

protocol suite
See *protocol stack.*

proxy
COVUEnet term for a mechanism that allows authorized users to log in to a node without having an account on that node.

Public Data Network (PDN)
A network established by a Post, Telephone, and Telegraph (PTT) authority, common carrier, or private operating company for the specific purpose of providing data communication services to the public.

PVC
See *Permanent Virtual Circuit.*

q

Q-bit
Data-qualified bit, contained in the header of an X.25 data packet, used to indicate special control packets.

QOS
See *Quality of Service.*

Quality of Service (QOS)
Parameters that determine the quality of service provided by a connection over X.25.

r

raw socket
IPC facility that provides users with access to the underlying network protocols.

Recognized Private Operating Agency (RPOA)
A packet network carrier. RPOA selection is an X.25 facility that provides transit networks with additional routing information about calls to a particular system.

Remote Procedure Call (RPC)
A facility provided by the CONVEX NFS product that allows a client process to have another process execute a procedure call, as if the caller had executed the procedure call in its own address space.

repeater
A node that amplifies the electrical signal on a segment of communication line without interpreting the data, effectively extending the network beyond the limitations of its physical media.

request
An action that initiates the transfer of data between peer entities. For example, an initiator may request that a connection be established. Requests are made by a service user, or *client*, to a service provider, or *server*.

Requests for Comments (RFCs)
A set of notes and technical papers that discuss various subjects related to TCP/IP protocols, including proposed and accepted standards.

responder
A system that responds to a request from an initiator. See *server, passive user.*

response
An action that an entity may issue to respond to a request. For example, when a peer entity issues a connection request, the responding entity issues a connection response that either accepts or rejects the connection.

reverse charging
An X.25 facility that allows network charges for a particular connection to accrue to a receiving DTE. Reverse charging is analogous to a collect call.

RFC
See *Requests for Comments.*

ring
A network topology in which each node is connected to adjacent nodes to complete a circle.

RIP
See *Routing Information Protocol.*

router
Operates as an intermediate node whose purpose is to direct messages to the appropriate network. Routers use the destination address included in a packet to determine where to send it. If more than one route to the destination exists, the router tries to choose the most efficient one.

routing
The task of finding the most efficient path over which to send packets to their destination.

Routing Information Protocol (RIP)
TCP/IP protocol implemented by the `routed` program to provide routing services for the local network or subnet.

RPC
See *Remote Procedure Call.*

RPOA
See *Recognized Private Operating Agency.*

S

SAP
See *Service Access Point.*

server
A process that fulfills a request issued by a client process, and transmits a response back to the client. Also called a *service provider.*

server stub
Used in implementing remote procedure call (RPC) facilities, the server stub waits for an RPC request from a client, executes a local procedure call on behalf of the client, and returns results to the client. Server stubs are used to abstract the details of passing messages over the network.

service
A function or set of functions provided by a network *entity.*

Service Access Point (SAP)
An address or identifier through which a service user can communicate with a service provider.

service provider
An entity that provides service for the next higher entity on a protocol stack. For example, the physical layer is a service provider to the data link layer. Also called a *server*.

service user
An entity that uses the service of the next lower entity in a protocol stack. For example, a transport entity is the service user of a network entity. Also called a *client*.

session layer
The OSI layer that establishes, manages, and terminates a period of communication between two end users. It is also responsible for synchronizing the exchange of data and controlling traffic over the connection.

Serial Line Internet Protocol (SLIP)
TCP/IP protocol that provides point-to-point communication over asynchronous serial lines at speeds from 1200 bps to 38400 bps. SLIP is included with CONVEX Internet Services.

Simple Mail Transfer Protocol (SMTP)
TCP/IP protocol used to relay electronic mail messages between networked machines.

SLIP
See *Serial Line Internet Protocol*.

SMTP
See *Simple Mail Transfer Protocol*.

SNA
See *System Network Architecture*.

SNPA
See *Subnetwork Point of Attachment*.

socket
Endpoint used for interprocess communication. See *Berkeley sockets*.

socket pair
Bidirectional pipes that enable application programs to set up two-way communication between processes that share a common ancestor.

Sockets Compatibility Library
See *UltraNet Sockets Compatibility Library*.

Source Service Access Point (SSAP)
Link service access point of the source link service user.

SSAP
See *Source Service Access Point*.

star
A network topology in which a central node serves as the point of connection for all other nodes.

Stream
Data flow path between a Stream head and driver in a STREAMS protocol stack.

Stream head
The entity in a STREAMS protocol stack that interfaces with a user process, providing the interface between the Stream in kernel space and the user application in user space. The Stream head processes STREAMS system calls from the user application and allows data to be transferred between the user application and the Stream in both directions.

STREAMS
A combination of system calls, kernel routines, and kernel utilities that provide an efficient means of implementing a dynamic network stack. STREAMS defines a standard interface between a STREAMS-based user application and the STREAMS protocol stack with which it communicates.

stream socket
IPC facility that provides for the bidirectional, reliable, sequenced, and unduplicated flow of data without record boundaries.

subnet
A network accessed via a single physical link or a single Ethernet interface.

subnetting
The partitioning of network address space defined by a single network address into multiple networks called *subnets* or *subnetworks*.

subnetwork
See *subnet*.

Subnetwork Point of Attachment (SNPA)
An address that identifies the physical attachment of a system to a network. For a WAN, the SNPA is the DTE address of the system; for an Ethernet LAN, the SNPA is the combined Ethernet address and Link SAP. See *hardware address*.

synchronous mode
Execution mode in which a user calls a library function and control does not return until after the corresponding asynchronous event occurs.

System Network Architecture (SNA)
Proprietary network architecture developed by IBM to connect IBM computers. CONVEX computers can be interconnected with SNA networks through the gateway product, *OpenConnect*. See *OpenConnect*.

---

t

task-to-task communication
An application program interface provided by COVUEnet that facilitates the exchange of data between two processes over a logical link.

TCP
See *Transmission Control Protocol*.

TCP/IP
A layered set of internetworking protocols named after its two major protocols, Transmission Control Protocol (TCP) and Internet Protocol (IP).

TCP/IP protocols
See *DARPA Internet protocols*.

TELNET Protocol
TCP/IP protocol that enables a user to log in to a remote host that runs basic TCP/IP protocols, but not necessarily BSD networking protocols.

TFTP
See *Trivial File Transfer Protocol*.

throughput class
QOS parameter that selects throughput rate in bits per second. Throughput describes the maximum amount of data that can be sent through the network when the network is operating at saturation.

TLI
See *Transport Layer Interface*.

topology
A description of the organization of a network in terms of its components, interconnections, and geography.

TP0/2/4
Transport Protocol (TP), Classes 0, 2, and 4. Each class provides a different set of transport layer services, such as packet sequencing and retransmission.

TPI
See *Transport Provider Interface.*

Transmission Control Protocol (TCP)
A connection-oriented TCP/IP protocol used to transfer a stream of data from a process running in one machine to its peer process running in a remote machine.

Transport address
An address consisting of a TSAP selector and a network address that uniquely identifies a TSU.

transport connection
In connection-mode service, the communication circuit between two peer transport users.

transport endpoint
Within the context of a single system, the path of communication between a transport user and an underlying transport provider. The path is identified by a local file descriptor.

transport layer
The OSI layer that provides a reliable end-to-end or host-to-host data transfer service that shields upper layers from the details of the underlying network. It is responsible for ordered delivery of data, flow control, and error recovery.

Transport Layer Interface (TLI)
The collection of operations, states, and events that define the interface between transport users and transport providers.

Transport Layer Interface library
The collection of user-level functions that implement the CONVEX Transport Layer Interface.

transport provider
A protocol that provides transport-level services.

transport provider identifier
When initializing a transport endpoint, the character string that identifies the device file that corresponds to a transport provider.

Transport Provider Interface (TPI)
Interface used by application programs to access transport services, such as TCP, UDP, and TP0/2/4. Transport drivers communicate with application programs through a stream head.

Transport Service Access Point (TSAP)
On a specific end system, uniquely identifies the context of a transport provider for a transport user. In connection-mode service, a single TSAP may be connected to multiple remote TSAPs with each connection uniquely identified by the peer transport endpoints.

Transport Service User (TSU)
An OSI software entity that uses the services of one or more transport providers.

transport user
See *Transport Service User.*

Trivial File Transfer Protocol (TFTP)
TCP/IP protocol that enables users to transfer files to and from remote hosts. It is normally used to transfer files, such as boot files, from the CONVEX system to remote workstations. Unlike File Transfer Protocol (FTP), TFTP does not require a user account or password on the remote host.

TSAP
See *Transport Service Access Point.*

TSAP selector
The name of a TSAP relative to an NSAP. When combined with a network address, it provides a globally unique name for a transport address.

TSU
See *Transport Service User.*

U

UA
See *User Agent.*

UDP
See *User Datagram Protocol.*

UltraNet
Highest-performance LAN product offered by CONVEX. UltraNet achieves its high data rate through the use of proprietary protocols. UltraNet software also includes an Internet interface that runs TCP/IP protocols.

UltraNet Sockets Compatibility Library
By emulating the socket interface, UltraNet software allows network applications to take full advantage of UltraNet's high performance through a standard application program interface, Berkeley sockets. The Sockets Compatibility Library allow execution of both BSD applications and socket-based user programs.

UNIX-to-UNIX Copy (UUCP)
A communication system that can run over direct serial lines, network connections, or ordinary telephone lines. UUCP is used for file copying and remote command execution.

unreliable data delivery service
Connectionless service that does not guarantee delivery of a packet to its destination. Other entities must perform error detection and correction functions.

upstream
Refers to the direction from a STREAMS driver to the Stream head. This is the direction of data flow that results from a `read` or `getmsg` operation.

User Agent (UA)
Entity that provides the interface between an end user and a Message Transfer Agent (MTA) in an X.400 Message-Handling System (MHS).

user data
Data that originates from a service user.

User Datagram Protocol (UDP)
Connectionless TCP/IP protocol used to transfer datagrams from a process running in one machine to its peer process running in a remote machine.

UUCP
See *UNIX-to-UNIX Copy.*

V

virtual circuit
A communication stream between two X.25 processes.

volatile database
A COVUEnet term referring to memory-resident data that allows you to control the operation of the network without modifying the permanent database.

W

WAN
See *Wide Area Network.*

Wide Area Network (WAN)
A network whose size and service area is larger than a single site. WANs usually include telephone system trunk lines or satellite links. Also called a *long haul network.*

X

X.3
CCITT recommendation for PAD operation that defines a set of terminal control parameters.

X.21
CCITT recommendation for the physical interface in an X.25 network.

X.25
CCITT recommendations for network access protocols for ISO layers 1, 2, and 3.

X.28
CCITT recommendation for PAD operation that defines the control procedures, user commands, and service signals `pad` sends to the terminal.

X.29
CCITT recommendation for PAD operation that defines the control messages sent between `pad` and the remote host. These messages are sent as X.25 qualified data packets.

X.400 Message-Handling System (MHS)
Service that enables the exchange of messages, such as electronic mail, between users on networked machines.

XDR
See *eXternal Data Representation.*

XXX
Refers to the three protocols—X.3, X.28, and X.29—that implement PAD functions.

Y

Yellow Pages (YP)
See *Network Information Service.*

YP
See *Network Information Service.*

Index

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

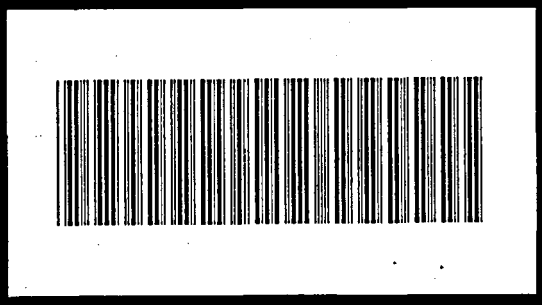
U

V

W

X

Y



Order Number
DSW-128

Document Number
710-014230-000