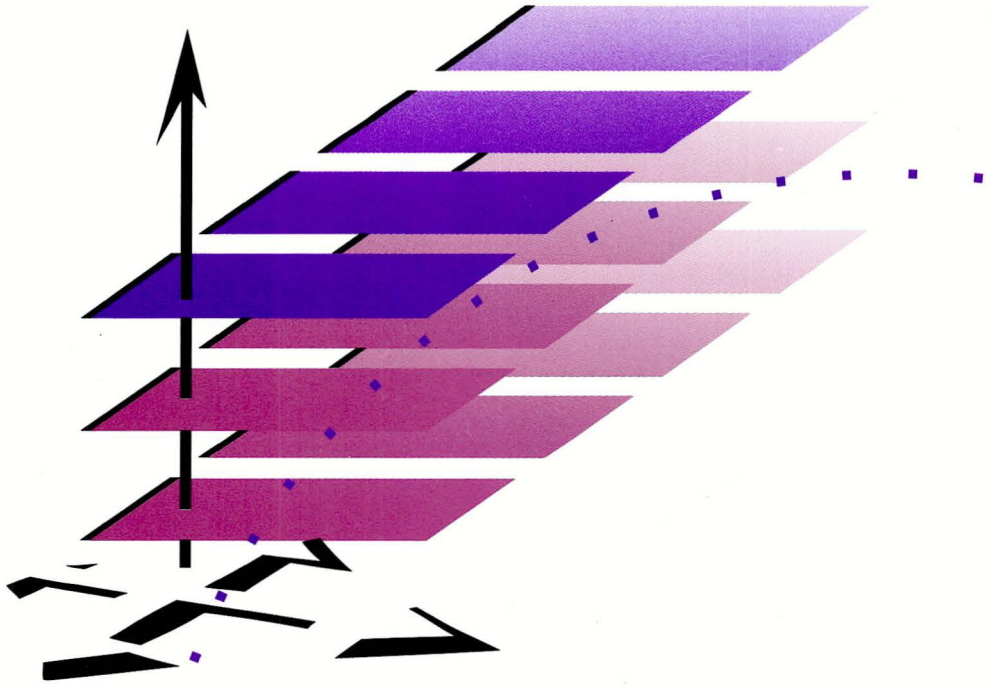


CONVEX

Exemplar Networking Guide

Fourth Edition

Hewlett-Packard Company
Convex Technology Center
3000 Waterview Parkway
P.O. Box 833851
Richardson, TX 75083-3851
United States of America

Exemplar Networking Guide

Order No. DSW-865

Fourth Edition

June 1996

Hewlett-Packard Company
Convex Technology Center
Richardson, Texas
United States of America

Exemplar Networking Guide
Order No. DSW-865

Notice

The information contained in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

This entire book is recyclable.

Printed in the United States of America

Revision Information for Exemplar Networking Guide

Edition	Document No.	Description
Fourth	710-031130-003	Released in June 1996 with SPP-UX V4.0. Includes new chapter on ATM configuration.
Third	710-031130-002	Released in September 1995 with SPP-UX V3.1. Included new chapter on HIPPI configuration.
Second	710-031130-001	Released in June 1995 with SPP-UX V3.1. Included new chapters on automounter, Ethernet configuration, and troubleshooting.
First	710-031130-000	Initial release December 1994, with SPP-UX V2.0.

Contents

Figures

Tables

What this guide contains

Who should read this guide

This guide focuses on the tasks that you as the system manager perform to set up, maintain, and troubleshoot networking on Exemplar systems.

What each chapter contains

If you are not familiar with Exemplar networking, read the chapter entitled "Overview of SPP-UX networking" for an overview of basic networking concepts. This chapter also contains

- An overview of Exemplar systems networking, including:
  - Product descriptions
  - Hardware interfaces
  - Differences between networking on single and multinode Exemplar systems
- Associated documentation

The chapter entitled "Networking concepts" contains general information on the following topics:

- Computer networking components and topology
- Protocols and architecture
- Standards
- Interconnecting networks
- Network services and administration
- Application program interfaces

The chapter entitled "Configuring FDDI" contains instructions on configuring FDDI into your Exemplar system.

The chapter entitled "Configuring Ethernet"contains instructions on configuring Ethernet into your Exemplar system.

The chapter entitled "Configuring HIPPI" contains instructions on configuring HIPPI into your Exemplar system.

The chapter entitled "Configuring ATM" contains instructions on configuring ATM into your Exemplar system.

The chapter entitled "Setting up your network using Internet utilities" presents an overview of tasks required to configure and test your network. It also gives instructions on how to:

- Identify hosts on a network
- Choose an address structure
- Create a host name database
- Regenerate your /etc/hosts and /etc/network files
- Complete your host name database setup

The chapter entitled "NFS utilities" explains how to administer services provided by NFS software, including the following:

- Managing NFS and NIS services
- Setting up an NFS server and client
- Removing temporary files created by NFS
- Configuring portmap
- Obtaining status of portmap and RPC services

Instructions on how you can use NFS's automounter to automatically mount and unmount file systems are located in Chapter 9.

The chapter entitled "Network Information System (NIS)" describes basic NIS concepts including the following:

- NIS master-slave set up
- NIS client-server management
- Modification of NIS maps
- Troubleshooting NIS services
- Security issues

Chapter 11 explains the use of the `ping` and `netstat` commands.

Extensive diagnostic flowcharts to help you deal with networking challenges are located in the chapter entitled "Troubleshooting flowcharts."

You can find instructions on how to track the causes of NFS and NIS problems in "Troubleshooting NFS and NIS."

How to order associated documentation

To order the current edition of these or any other Convex documents, send requests to:

Hewlett-Packard Company
Convex Technology Center
Customer Service
P.O. Box 833851
Richardson TX 75083-3851 USA

Please include the order number (DSW or DHW number) or the exact title of the document.

How to get technical assistance

If you have questions that are not answered in this book, contact the Hewlett-Packard 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 Technology Center office.

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

Introduction

Overview of SPP-UX networking 1

This chapter introduces Exemplar systems networking products and lists prerequisites for configuring networking software. It contains:

- An overview of Exemplar networking:
  - Networking on Exemplar subcomplexes
  - Differences in networking for single and multinode systems
  - Product descriptions
    * Internet utilities
    * Network File System (NFS) utilities
    * Network Information Services (NIS)
  - Hardware interface descriptions
    * Fiber Distributed Data Interface (FDDI)
    * Ethernet interface
    * High Performance Parallel Interface (HIPPI)
    * Asynchronous Transfer Mode (ATM)
- Documentation
  - Test station documentation
  - OpenBoot documentation
  - Related networking documentation
- Where to find what you need in this book
- What is not in this book
- Before you begin

Exemplar networking

Exemplar system computers, with the SPP-UX operating system, offer Internet utilities and Network File System (NFS) utilities.

For SPP-UX, these two networking services are provided over network interfaces.

To make the task of network administration easier, Exemplar systems networking software offers utilities and databases to configure, manage, monitor, and troubleshoot a network. Use of these utilities and their databases are explained throughout this guide.

For information on application program interfaces, refer to the "Application program interfaces" section.

Product descriptions

Hewlett-Packard Convex offers the following networking products:

- Internet utilities
- Network File System (NFS)
- Network Information Services (NIS)
- Network interfaces
  - Fiber Distributed Data Interface (FDDI)
  - Ethernet
  - HIPPI
  - ATM

A summary of each product follows.

Software product descriptions

Internet utilities

Internet utilities 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 Exemplar networking products, as well as other systems. Internet utilities runs over Ethernet, FDDI, HIPPI, and ATM interfaces.

Internet utilities provide transparent access and resource-sharing capabilities among Exemplar systems and a wide variety of other machines, including personal computers, workstations, minicomputers, and supercomputers.

Internet utilities software includes the following:

- Berkeley networking utilities, used to communicate with other SPP-UX or UNIX systems that run Berkeley Software Distribution (BSD) networking.
- DARPA Internet utilities, used to communicate with systems running TCP/IP, but not necessarily BSD networking.
- System management utilities and databases used to configure, maintain, monitor, and troubleshoot a TCP/IP network.
- Interprocess Communication (IPC) system calls and library routines that facilitate communication among application programs.
- The Berkeley Internet Name Domain server (BIND), a network service that enables clients to name resources and share this information with other hosts on the network.

Network File System (NFS) utilities

Network File System (NFS) utilities is a distributed file system that provides transparent access to file systems on remote machines. NFS, developed by Sun Microsystems, Inc., links together heterogeneous systems—including personal computers, workstations, supercomputers, and mainframes—to share resources and files over local area networks. An NFS server allows a remote NFS client to mount and access the server's exported file systems as if they were local to the client.

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

On Exemplar systems, NFS runs over Ethernet, FDDI, HIPPI, and ATM interfaces. 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.

NFS utilities include the following:

Remote Procedure Call (RPC) facilities.

Enable client processes to have another process execute a procedure call as if the caller had executed the procedure call in its own address space.

Network Information Service (NIS).

Distributed lookup system that centralizes administration of certain network configuration files.

Lock manager facilities.

Enable cooperating processes to implement record locking.

System management utilities and databases.

Used to configure, maintain, monitor, and troubleshoot NFS utilities.

Automounter.

Utility that automatically mounts remote file systems when they are accessed and unmounts them when they have not been accessed for a period of time.

Network Information Services (NIS)

Network information services (formerly known as Yellow Pages) is a distributed lookup service that maintains a centralized set of configuration files that would otherwise be maintained as ASCII files on each local system. Networked hosts query a central server for information as they would a database.

Hardware interfaces

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

Fiber Distributed Data Interface (FDDI)

FDDI is a Sbus controller that connects Exemplar systems 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. FDDI supports a dual attached Class A FDDI station 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.

The Ethernet interface supports the TCP/IP protocol to provide the full functionality available with Internet utilities, NIS, and NFS.

Ethernet interface

Exemplar system computers interface to an Ethernet network through Excelan Multibus and Sbus 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 the TCP/IP protocol to provide the full functionality available with Internet utilities, NIS, and NFS.

High Performance Parallel Interface (HIPPI)

HIPPI is a high-performance parallel interface. HIPPI is a copper-based data communications standard capable of transferring data at 800 Mbit/sec over 32 parallel lines or 1.6 Gbit/sec over 64 parallel lines.

The Ethernet interface supports the TCP/IP protocol to provide the full functionality available with Internet utilities, NIS, and NFS.

Asynchronous Transfer Mode interface (ATM)

Asynchronous tranfer mode is the technique for transport, multiplexing, and switching that provides the high degree of flexibility required by B-ISDN. ATM is a connection-oriented protocol that employs fixed-size packets wtih a 5-byte header and 48 bytes of information.

The Ethernet interface supports the TCP/IP protocol to provide the full functionality available with Internet utilities, NIS, and NFS.

Documentation

Several sources of related documentation are available; some are shipped with your Exemplar system.

Test station documentation

The Exemplar test stations (Hewlett-Packard workstations) that are shipped with Exemplar systems have hardware network connections that you must ensure are properly made. For Exemplar/CD systems, refer to the *Convex Exemplar/CD Site Preparation and Installation Guide* for instructions on these connections. For Exemplar/XA systems, refer to the *Convex Exemplar/XA Installation Guide.*

If an error occurs with your test station hardware networking connections, please refer to those guides for troubleshooting information.

OpenBoot documentation

Open Boot Prom (OBP) refers to software used by the Exemplar test station. For basic information on Open Boot commands and instructions, please refer to the*Convex OpenBoot Quick Reference* (DSW-854).

Related networking documentation

This guide focuses on the tasks that you as the system manager perform to set up, maintain, and troubleshoot networking hardware and software on Exemplar systems.

Fundamental networking concepts and theories are described in the "Computer networks" section.

You can find additional information about network services in the technical section of your local book store. If you are responsible for managing Internet utilities or NFS software, consider reading *Managing NFS and NIS*, by Hal Stern (O'Reilly & Associates, 1992. ISBN 0-93175-75-7).

The following lists contain titles of Hewlett-Packard manuals (included with each SPP-UX shipment) that contain useful supplemental information on Exemplar systems networking services.

For Internet utilities:

- *Using ARPA Services* (DHP-175).

- *Installing and Administering ARPA Services* (DHP-176)

For NFS:

- *Installing and Administering NFS Services* (DHP-173)
- *Programming and Protocols for NFS Services* (DHP-174)
- *Using NFS Services* (DHP-172)

At or near the beginning of some chapters in this guide you will find a list of HP documentation, either specific titles or specific chapters, that contains detailed information relevant to the contents of that chapter.

What is not in this book

This book does not provide you with the information you may need to help you decide whether or not to configure a particular service on a particular host.

For information on installing, configuring, and testing network interfaces, refer to the appropriate service guide and diagnostic manual. For information on hardware connections, please refer to the following:

- For Exemplar/CD systems, refer to the *Convex Exemplar/CD Site Preparation and Installation Guide.*
- For Exemplar/XA systems, refer to the *Convex Exemplar/XA Installation Guide.*

Finally, this guide does not cover in detail any task presented in detail in Hewlett-Packard networking documentation (included as part of your SPP-UX documentation set).

Before you begin

This guide provides instructions for configuring and managing NFS and Internet utilities services. Before you perform any configuration procedure, you must successfully complete the following tasks:

1. Install and verify operation of network hardware interfaces.
2. Install SPP-UX and utilities, which includes Internet utilities, NFS utilities, and NIS utilities.

Networking concepts 2

This chapter introduces basic networking concepts employed on Exemplar systems; specifically, 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
- Administering a network

If you are already familiar with the basics of networking, you may want to skip forward to Chapters 3 through 12 for instructions on how to configure network interfaces, use NFS, Internet utilities, and NIS, and troubleshoot your network.

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 shows a LAN that connects an Exemplar system to a file server, a laser printer, and several workstations.

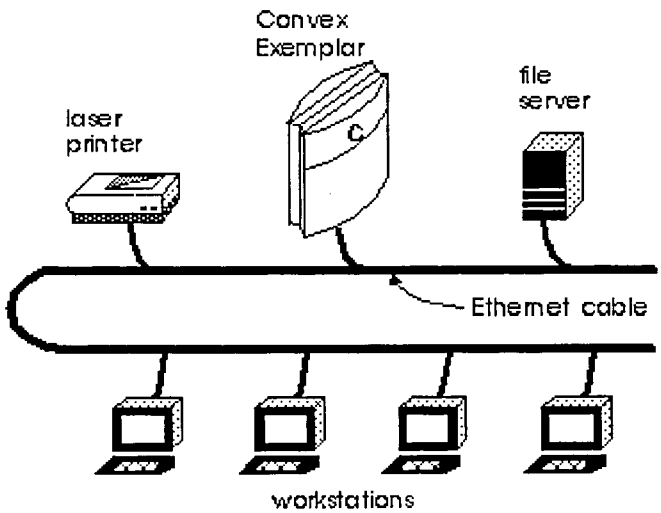


Figure 1 Model of a local area network

Exemplar networking products provide LAN communication over several types of network interfaces, including FDDI, Ethernet, HIPPI and ATM.

Metropolitan Area Networks

The *Metropolitan Area Network* (MAN) provides large corporate customers with the ability to transfer massive amounts of voice and data within a service area 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 discussion groups with other users worldwide.

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. 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.

All hosts and most nodes must be assigned a unique network address so that other machines on the network can direct messages to them.

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.

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 page 10 has a *linear*, or *bus*, topology, in which all network components connect to a common communication line. In Figure 1, that line is 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.

Protocols and network architecture

While 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 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 2 illustrates the concept of protocol layering.

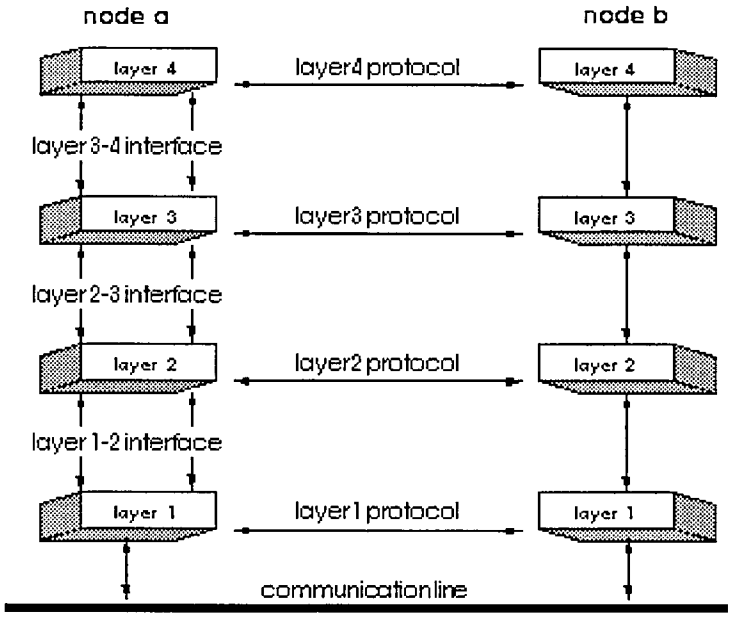


Figure 2 Protocol layering

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.

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 entitled "Obtaining information about standards" for more information.

In this book and many others, you will see the word, "internet," written two ways: all in lowercase, and with an initial capital. 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 entitled "Modes of service" 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 on "Profiles" 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 10) 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 3.

```
#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
#
```

Figure 3 Transferring RFCs from NIC.DDN.MIL

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 4.

```
#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/rfcnnnn.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!
#
```

Figure 4 Transferring RFCs from NISC.SRI.COM

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 5.

```
#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/rfcnnnn.txt (20972 bytes).
226 Transfer complete.
20972 bytes received in 3.7 seconds (5.5 Kbytes/s)
ftp> quit
221 Goodbye.
#
```

Figure 5 Transferring RFCs from NISC.JVNC.NET

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 6 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.

```
#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.
```

Figure 6 Transferring ISO specifications from bruno.cs.colorado.edu

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. Figure 7 shows the seven-layered architecture of the OSI model.

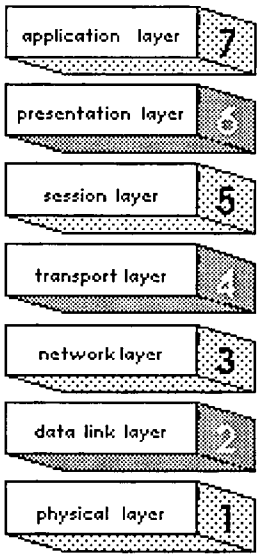


Figure 7 The OSI model

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 Exemplar 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 10) 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
- Backbones

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 necessary.

Gateways are used primarily to interconnect different types of networks. For example, a gateway can be used to interconnect Internet utilities networks and IBM Systems Network Architecture (SNA) networks. 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. However, unlike gateways, 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 8 illustrates the difference between gateways and bridges, and the purpose each serves in connecting networks.

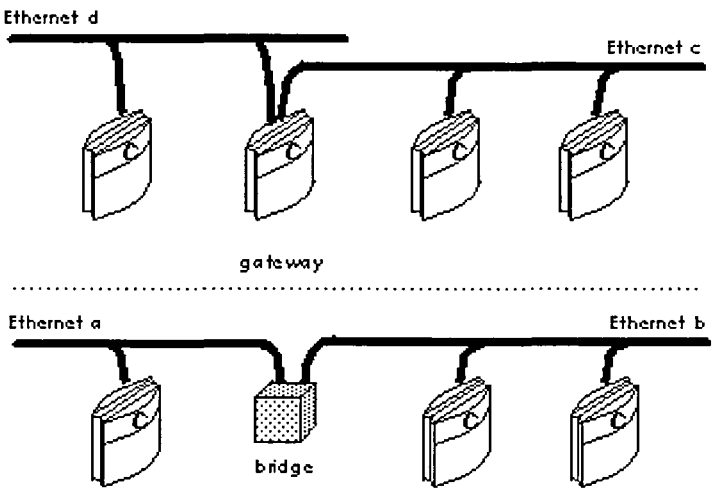


Figure 8 The roles of gateways and bridges

The gateway shown in Figure 8 is an Exemplar 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, however, must use different network numbers to reach hosts on 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 9.

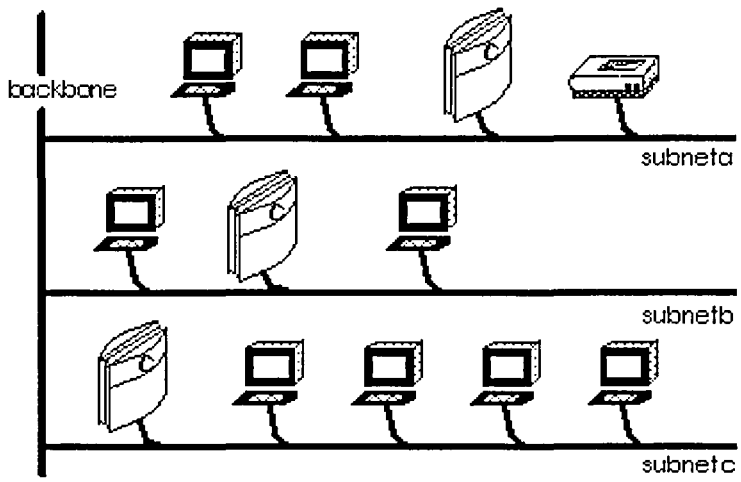


Figure 9 Subnets connected by a backbone network

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.

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, Internet utilities use 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 10) 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. Exemplar 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 Exemplar networking products, refer to the "Application program interfaces" section.

Network administration

Because a computer network is complex, so is the task of administering one. Each Exemplar networking product includes a set of utilities used to configure, manage, and monitor the network.

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 SPP-UX 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.

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.

Exemplar 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.

Application program interfaces

Exemplar 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, you must use IPC facilities for one of two reasons:

- 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`.
- 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.

SPP-UX 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 10, 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.

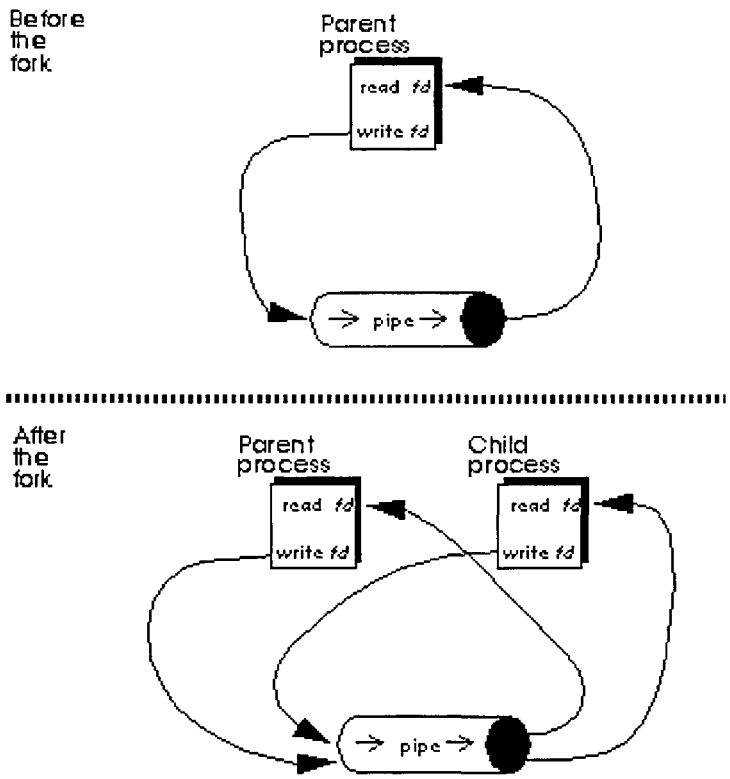


Figure 10 Sharing a pipe between processes

Before the fork, the parent process's 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, and it only allows communication between sockets on the same machine. Therefore, you cannot use socket pairs for network applications. Figure 11 shows a bidirectional flow of data through a socket pair.

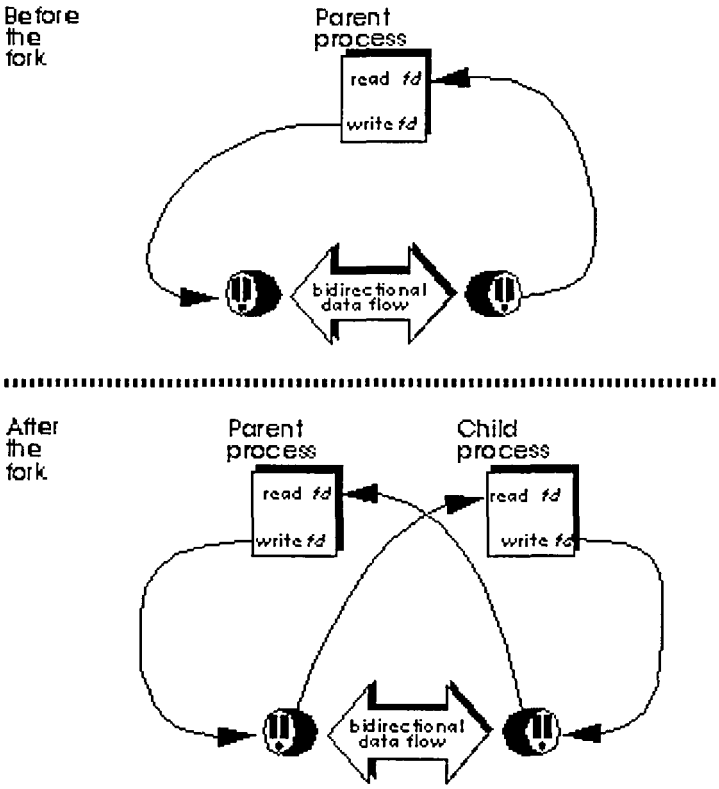


Figure 11 Sharing a socket pair between processes

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. SPP-UX 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 12 illustrates a typical sequence of system calls used for connection-oriented communication between network processes.

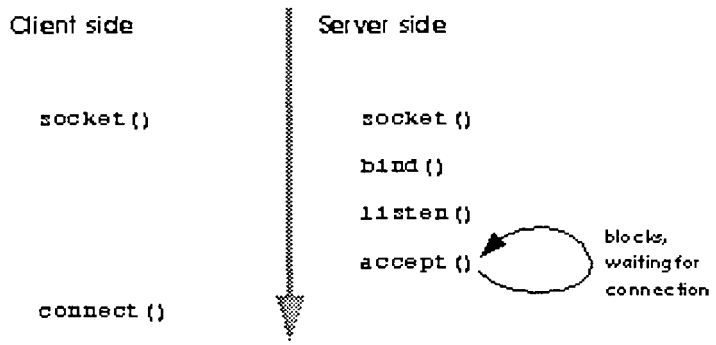


Figure 12 Establishing a connection using sockets

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 13 illustrates a typical sequence of system calls used for this type of network communication.

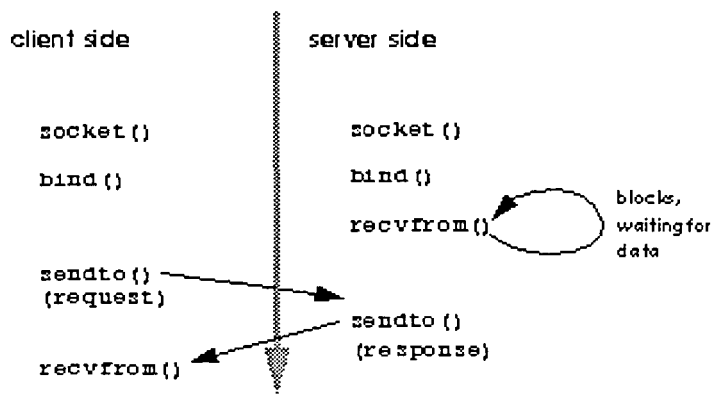


Figure 13 Connectionless communication using sockets

Figure 12 and Figure 13 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. For example, 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.

Exemplar 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`)
- 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.

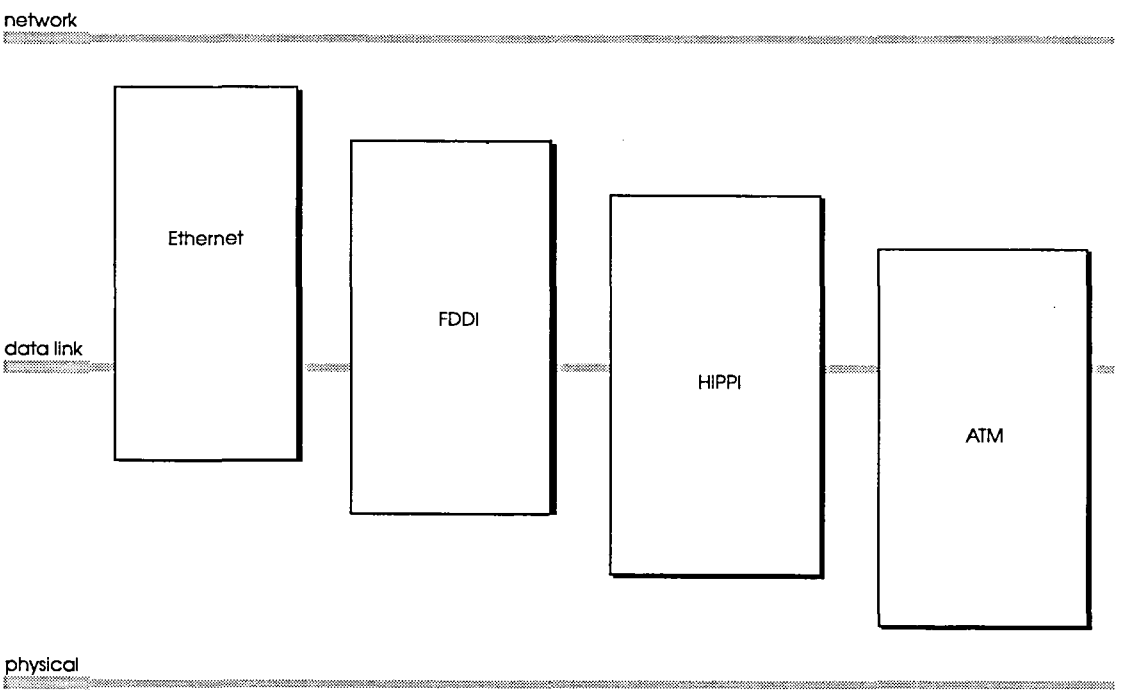
Byte-swapping routines are provided because TCP/IP defines a standard byte order for integral values that may or may not match the natural byte ordering on a particular machine. Because current Hewlett-Packard Convex architecture orders bytes in the way TCP/IP expects them, byte-swapping routines have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

Table 1 lists byte handling routines.

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

Configuring network interfaces



Configuring FDDI 3

Configuring FDDI into your network

When you add a network interface to your system or change characteristics of an existing one, you must integrate the device into SPP-UX and customize the operating system for its particular characteristics.

This section describes how to integrate a Fiber Distributed Data Interface (FDDI) network device into SPP-UX. Specifically, it explains how to:

- Use OpenBoot commands to define the interface at hypernode power on/reset time
- Use the `ifconfig` command to configure your network
- Use the `route` command to define gateways
- Use the `netstat` command to verify network configuration

This section covers configuration and management of network software; it does not explain how to install hardware interfaces. You must install, configure, and verify operation of the hardware before you begin software configuration tasks.

SPP-UX does not support configuration of your FDDI using HP-UX's system administration manager utility, SAM. You must perform the configuration manually.

HP-UX / SPP-UX differences

You perform initial SPP-UX configuration steps on the test station at SPP-UX boot time. Refer to the SPP-UX *Release Notice* specific to your version of the operating system for more information.

SPP-UX does not support the lanscan(1M) command—use netstat(1) instead.

Terms

Table 2 describes OpenBoot terms used in this chapter.

Table 2 OpenBoot terms

OpenBoot term	Analogous SPP-UX term	Description
device path	file	A type of hardware device.
full device name	absolute path name	A series of device paths separated by slashes that represent where a device is located within the hierarchical structure of the device tree.
device tree	file system tree	A hierarchical data structure that OBP creates and maintains.

In addition, the following terms are used in this chapter:

dot notation

Defines a network address as a series of decimal numbers (usually four) separated by periods, as 128.50.10.27.

/etc/networks

Contains logical names, addresses, and aliases for each network with which your LAN communicates.

logical unit

Defines a unique path to physical devices such as disks, tape drives, or networks.

Before you begin

Before you begin FDDI configuration, you must:

- Install and test FDDI hardware.

  For complete instructions on installing and testing FDDI hardware, refer to the appropriate service guide and diagnostic manual.

- Connect the FDDI card to an active ring before booting.

  If you do not initiate this connection, the FDDI card will not be recognized by the kernel.

Summary of steps

The following procedures summarize the steps required for you to define an FDDI interface, configure your network, and verify its configuration.

Defining an FDDI interface

To define your FDDI interface at hypernode power on/reset time, perform the following steps:

Step 1 In each of your Exemplar System Console windows, determine the full device name with the `show-devs` command (see the section "Defining an FDDI with OpenBoot commands").

Step 2 Find the device path that includes the string `cddi` in the output from the `show-devs` command. Enter the entire line of text as a *cddistring* to the `mkmap` command (see Figure 14 on page 45).

Step 3 Add the new logical unit by entering the **`reset`** command at the OpenBoot `ok` prompt (see the "OpenBoot's device tree" section).

Step 4 Boot the nodes of your Exemplar system (see the "OpenBoot's device tree" section).

Configuring your network

To integrate your FDDI system into SPP-UX, perform the following steps:

Step 1 Determine the network address and operating characteristics you need. See "Configuring FDDI with ifconfig" for more information on `ifconfig`.

Step 2 Login as superuser.

Step 3 Test the `ifconfig` options you selected in Step 1 by entering them on the command line.

Step 4 After you test the `ifconfig` command, verify that the system accepted the information.

Step 5 If your network appears to be configured properly, edit your `/etc/netlinkrc` file to add the `ifconfig` command.

Step 6 (Optional) Add a `route` command to the `/etc/netlinkrc` file. See the section "Using route" for more information.

Verifying configuration

To verify your network's configuration, enter

```
netstat -i
```

See the section "Verifying FDDI configuration with netstat" for an explanation of output received from this command.

Defining an FDDI with OpenBoot commands

Before booting the nodes of your Exemplar system, you must specify a logical unit number for each hypernode's FDDI controller to OpenBoot. Steps 1 through 4 detail this process and must be performed for each FDDI adapter on each Exemplar hypernode.

For more information about OpenBoot's device organization, see the section "OpenBoot's device tree."

Step 1 In each Exemplar System Console window, determine the full device name. Enter **`show-devs`** at the OpenBoot `ok` prompt. `show-devs` with no argument displays all devices known on the hypernode; sample output is shown in Figure 14.

```
ok show-devs
    /landmarc@0,ffec0000
    /landmarc@0,ffed0000
    /HP,PA7100@0,ffe70400
    /HP,PA7100@0,ffe60400
    /HP,PA7100@0,ffe50400
    /HP,PA7100@0,ffe40400
    /HP,PA7100@0,ffe30400
    /HP,PA7100@0,ffe20400
    /HP,PA7100@0,ffe10400
    /HP,PA7100@0,ffe00400
    /virtual-memory@0,0
    /memory@0,0
    /dart@0,f0200080
    /console@0,f0200a00
    /openprom
    /chosen
    /aliases
    /options
    /packages
    /landmarc@0,ffec0000/sbus@f,fcfffff00
    /landmarc@0,ffec0000/sbus@f,fdfffff00
    /landmarc@0,ffec0000/sbus@f,fcfffff00/Convex,afws@1,10000
    /landmarc@0,ffec0000/sbus@f,fcfffff00/Convex,afws@1,10000/st
    /landmarc@0,ffec0000/sbus@f,fcfffff00/Convex,afws@1,10000/sd
    /landmarc@0,ffed0000/sbus@f,fcfffff00
    /landmarc@0,ffed0000/sbus@f,fdfffff00
    /landmarc@0,ffed0000/sbus@f,fcfffff00/CRES,cddi@1,400000
    /packages/obp-tftp
    /packages/deblocker
    /packages/tape-label
    /packages/disk-label
```

Figure 14 show-devs sample output

Step 2 Find the device path that includes the string cddi in the output from the show-devs command. Enter the entire line of text as *cddistring* in the following command:

mkmap -n *logical_unit_number cddistring*

where

mkmap

Defines an SPP-UX logical-unit to physical-unit mapping. This mapping is a label for tape and network devices that cannot be labeled like disks.

`-n`

Creates a logical-unit property with no physical unit information; this option should only be used with network adaptors.

logical_unit_number

Designates the logical-unit number of each FDDI device.

You may want to use the *interface* unit number you supply when you enable your network with the `ifconfig` command. For instance, if your interface is `fddi0`, the *logical_unit_number* should be `0`. See the section "Configuring FDDI with ifconfig" for more information.

cddistring

Represents the full FDDI device name.

Your FDDI device's logical unit number is assigned by OpenBoot at hypernode power on/reset time.

Step 3 Add your new logical unit by entering **`reset`** at the OpenBoot `ok` prompt.

Step 4 Boot the nodes of your Exemplar system by entering the proper command at the OpenBoot `ok` prompt in each Exemplar system console window. Refer to the SPP-UX *Release Notice* specific to your version of operating system for more information.

When SPP-UX boots, the FDDI driver asks OpenBoot for its logical unit number and the driver keeps track of the number.

OpenBoot's device tree

You determine the hardware configuration of your system by looking at the device tree, accessed from your system console, using the `show-devs` command. The device tree describes the hardware devices attached to each hypernode; its organization is similar to an SPP-UX file system tree.

Refer to OpenBoot documentation for more detailed information on the device tree.

As the device tree diverges, a route, or device path, to individual devices becomes apparent. The device path represents the type of device and where that device is located within the device tree. A full device name is a series of device paths separated by slashes.

Device path syntax is

name@address:[arguments]

where

name

Represents an address on the main system bus. Valid names for Exemplar systems include:

`sd`
SCSI disk

`st`
SCSI tape

`le`
Ethernet

`CRES, cddi, fddi`
FDDI

`hippi`
HIPPI

`convex, afsw`
SCSI

@address

Represents a physical space unique to the device. *address* consists of two 32-bit numbers, usually in hexadecimal format, separated by a comma. The interpretation of these two numbers depends on the location of the device in the device tree.

arguments

(Optional) Passes additional information to the device's software. Valid options are specific to each device package. *argument* usually shows additional device information, such as disk partition.

For more information on device path syntax, refer to OpenBoot documentation.

Configuring FDDI with ifconfig

Before SPP-UX can communicate over a network interface, it needs to know the interface's network address and operating characteristics. This section explains how to use the `ifconfig` command to assign a network address to your FDDI and set parameters that affect its operation.

Step 1 Determine the network address and operating characteristics you need.

Step 2 Log in as the superuser.

Step 3 Use the `ifconfig` command to test your choices from Step 1 as options on the command line.

The `ifconfig` command is located in your /etc/netlinkrc file. Although the interface is automatically configured at boot time, you might have to modify this entry, for instance to enable `arp`. If so, test your `ifconfig` command first.

The `ifconfig` command has the following syntax:

`/etc/ifconfig` *interface address_family* [*address* [*dest_address*]] [*parameters*]

where

interface

Is a string composed of interface type, or name, and unit number like `fddi0`. Logical unit numbers are assigned at boot time by the OpenBoot command `mkmap`. `netstat` displays the name and unit number of interfaces associated with your Exemplar system.

address_family

Is the name of the protocol on which your naming scheme is based. An interface can receive transmissions in differing protocols, each of which may require separate naming schemes. Therefore, it is necessary to specify the *address_family*, which may affect interpretation of the remaining parameters on the command line. The only address family currently supported is `inet` (DARPA -Internet family).

address

Is either a host name present in the host name database or a DARPA Internet address expressed in Internet standard dot notation.

dest_address

Is the address of the destination system. It consists of either a host name present in the host name database or a DARPA Internet address expressed in Internet standard dot notation.

parameters

Is any of the operating parameters listed in Table 3.

Table 3 `ifconfig` operating parameter options

Option	Description
`up`	Enables interface after an `ifconfig down`. Occurs automatically when setting the address on an interface. Setting this flag has no effect if the hardware is `down`.
`down`	Marks an interface :the system will not attempt to transmit messages through that interface. If possible, the interface will be reset to disable reception as well. This action does not automatically disable routes using the interface.
`broadcast address`	Specifies the address used to represent broadcasts to the network. By default, the broadcast address has a host part of all ones. Refer to "Defining the broadcast address."
`netmask` *mask*	Specifies the portion of the internet address to reserve for the combined network and subnet fields. Refer to "Defining the subnet mask."
`dest_address`	Specifies address of correspondent on the other end of a point-to-point link.
`arp`	Enables use of the Address Resolution Protocol (ARP), a mechanism for dynamically mapping between Internet addresses and physical addresses. Refer to the section "Using the Address Resolution Protocol (ARP)."
`-arp`	Disables use of the Address Resolution Protocol (ARP).
`debug`	Enables driver-dependent debugging code. This usually turns on extra console error logging.
`-debug`	Disables driver-dependent debugging code.
`-trailers`	Disables the use of a trailer link-level encapsulation. This is the default.
`metric` *n*	Sets the routing metric of the interface to *n*. The default is 0. The routing metric is used by the routing protocol. Higher metrics have the effect of making a route less favorable; metrics are counted as additional hops to the destination network or host.

In SPP-UX, the ifconfig default is `-trailers`; setting the `trailers` parameter has no affect.

Figure 15 shows sample `ifconfig` options, to declare the Internet address as 130.150.60.6, the broadcast address as 130.150.60.255, the netmask as 0xffffff00, and marks the interface as up.

```
# ifconfig fddi0 inet 130.150.60.6 broadcast address 130.150.60.255\
netmask 0xffffff00 up
```

Figure 15 Setting up FDDI networking with `ifconfig`

Step 4 Verify that the system accepted the information.

Enter

ifconfig *interface*

In response, the system displays the network address and operating characteristics of the interface. Sample output is shown in Figure 16.

```
# ifconfig fddi0
fddi0: flags=43<UP,BROADCAST,RUNNING>
       inet 130.150.60.6 netmask ffffff00 broadcast 130.150.60.255
```

Figure 16 Using `ifconfig` to verify interface configuration

Step 5 If the network appears to be configured properly, edit your /etc/netlinkrc file to add or modify the `ifconfig` command.

This ensures that the network interface is enabled every time the system boots. Figure 17 shows a partial /etc/netlinkrc file containing `ifconfig` commands.

```
#
# Initialize networking interfaces.
#

case $NODENAME in
  *) /etc/ifconfig fddi0 inet 130.150.60.0 netmask 255.255.255.0 up
     STATUS=$?
     if [!$STATUS -eq 0 ]
     then
        net_init=1
     fi
     ;;
esac
```

Figure 17 Sample /etc/netlinkrc file

Step 6 (Optional) Add a `route` command to the /etc/netlinkrc file. See the section "Using route" for more information.

Defining the broadcast address

All machines that communicate with each other must use the same broadcast address: either all zeros or all ones. The current standard is a broadcast host address of all ones, as in broadcast 128.194.*255.255*.

The default broadcast address contains a host part of all ones. `ifconfig` enables you to change an interface's broadcast address. Exemplar systems can accept broadcasts with a host part of all zeros (for compatibility with systems that use BSD 4.2 broadcasts) because it transmits and receives broadcasts with the address set using the mask in `ifconfig`.

If a machine on your network does not understand the broadcast address you select, some utilities, such as `rwho`, fail. If this happens, use `ifconfig` to set the broadcast address to the same broadcast address for all machines on the network.

Defining the subnet mask

Specify the mask as a single hexadecimal number with a leading 0x, or in dot notation. Ones in the subnet mask indicate bit positions to use for the combined network and subnet fields; zeros mark the positions of bits in the host field. To avoid confusion with broadcast addresses, do not use subnet numbers of all zeros or all ones. For example, specifying

```
netmask 0xffffff00
```

or

```
netmask 255.255.255.0
```

both indicate that you want 24 bits of combined network and subnet fields, and 8 bits of host number. For a class B network, this mask logically partitions your 16 bits of host number into an 8-bit subnet field and an 8-bit host field. If you do not supply a `netmask`, the mask is set according to the network class (A, B, or C with 8, 16, or 24 bits of network part, respectively) or your chosen IP address.

Using the Address Resolution Protocol (ARP)

ARP maps logical internet addresses to physical addresses by caching the logical mappings between dot notation addresses (as in the /etc/hosts file) and physical addresses. If an interface requests mapping for an address not cached, ARP queues the message that requires the mapping, then broadcasts a message on

the associated network to request the address. If a response is received, the new mapping is cached, and the queued message is transmitted.

If you want to communicate with a host on the network that does not use the ARP protocol, you must use the `arp` program to manually add address mapping information to the local ARP table. The `arp` program forces caching of an ARP table entry for a specific host, so that the ARP protocol does not transmit a packet to a host to get its hardware address.

`arp` has the following syntax:

`arp -s -a -f` *hostname address* `[temp] [pub] [trail]`

where

`-s`

Creates an ARP entry for the host called *hostname.*

`-a`

Displays all current ARP entries.

`-f`

Reads file *filename* and set multiple entries in ARP tables.

hostname

Specifies the remote host as listed in the /etc/hosts file. If an ARP entry already exists for hostname, the existing entry is updated with the new information.

address

Specifies the physical hardware stations address. This address is displayed by most systems at boot time and is usually printed on the network controller. You specify it as six hexadecimal numbers separated by colons, as in

08:00:20:06:dd:42

The entry will be permanent unless you also specify `temp`.

`temp`

Specifies that the *address* entry is not permanent.

`pub`

Specifies the entry is published, which means that this system will act as an ARP server responding to requests for *hostname* even though the host address is not its own.

`trail`

Indicates that trailer encapsulations can be sent to this host.

You can check the current status of the ARP table by entering

```
arp -a
```

`arp` has more options than are presented here. For a complete summary, refer to the arp(1M) man page.

Using route

Add a `route` command to /etc/netlinkrc if your hypernode is used as a gateway or will use gateways.

The following shows a common use example of the `route` command where 130.168.85.254 represents the gateway:

```
# route add default 130.150-60.254
```

You may use `netstat -r` to look at the route table contents.

`route` has the following syntax:

`route add net` *destination gateway*

where

`add`

Adds a route.

`net`

Specifies the type of destination address.

destination

Specifies the destination host system where packets will be routed. Can be a host name (official host name or alias), an Internet address in dot notation, or the keyword `default`, which represents the wildcard gateway routing. See the route(7) man page for more information about the keyword, `default`.

gateway

Specifies the gateway through which the destination is reached. Can be a host name (official host name or alias) or an Internet address in dot notation.

`route` has more options than are presented here. For a complete summary, refer to the route(1M) man page.

Verifying FDDI configuration with netstat

Successfully booting your Exemplar in multiuser mode indicates that you have correctly configured network interfaces. In most cases, the machine will not run in multiuser mode if you make a mistake during the configuration process. You should test the network after the system is up and running in multiuser mode.

You can verify network configuration by entering

```
netstat -i
```

If the network is properly configured, the system displays output similar to that shown in Figure 18.

```
# netstat -i

Name    Mtu   Network         Address       Ipkts Ierrs    Opkts Oerrs  Coll
fddi0   4332  130.158.160     doc-f         93187     1    91491     0     0
lo0     4056  127.0.0.1       localhost     324       0      324     0     0
```

Figure 18 Checking FDDI configuration with `netstat -i`

The displayed host name and network name are specific to your installation. The name of the interface you just configured should appear. If it does not, you have a problem with your installation or your configuration of the network. In either case, use the troubleshooting procedures outlined in "Troubleshooting Internet services".

Configuring Ethernet 4

Configuring an Ethernet interface

When you add a network interface to your system or change characteristics of an existing one, you must integrate the device into SPP-UX.

This section explains how to integrate a newly installed or reconfigured Ethernet interface into SPP-UX. Specifically, it explains how to:

- Use OpenBoot commands to define the interface at hypernode power on/reset time
- Use the `ifconfig` command to configure your network
- Use the `route` command to define gateways
- Use the `netstat` command to verify network configuration
- Tune network performance by modifying boot-time parameters

This chapter does not explain how to install network hardware; you must install the hardware before attempting any task described here.

Refer to the *SPP-UX System Administration Guide* for general information about configuring devices into SPP-UX.

SPP-UX does not support configuration of your Ethernet using HP-UX's system administration manager utility, SAM. You must perform the configuration manually.

Before you begin

Before you begin Ethernet configuration, you must:

- Install and test Ethernet hardware.

  For complete instructions on installing and testing Ethernet hardware, refer to the appropriate service guide and diagnostic manual.
- Connect the Ethernet card to an active LAN before booting.

Summary of steps

The following procedures summarize the steps required for you to define an Ethernet interface, configure your network, and verify your configuration.

Defining an Ethernet interface

To define your Ethernet interface at hypernode power on/reset time, perform the following steps:

Step 1 Go to the correct hypernode. In each of your Exemplar System Console windows, determine the full device name with the `show-devs` command.

Step 2 Make maps. Find the device path that includes the string `le` in the output from the `show-devs` command. Enter the entire line of text as a *lestring*.

Step 3 Make sure your current directory is root.

Step 4 Add the new logical unit by entering the following command at the OpenBoot `ok` prompt.

```
reset
```

Step 5 Boot the nodes of your Exemplar system.

Configuring your network

To integrate your Ethernet system into SPP-UX, perform the following steps:

Step 1 Determine the network address and operating characteristics you need. See the section "Configuring Ethernet with ifconfig" for more information on `ifconfig`.

Step 2 Log in as superuser.

Step 3 Test the `ifconfig` options you selected by entering them on the command line.

Step 4 After you test the `ifconfig` command, verify that the system accepted the information.

Step 5 If your network appears to be configured properly, edit your /etc/netlinkrc file to add the `ifconfig` command.

Step 6 (Optional) Add a `route` command to the /etc/netlinkrc file. See the section "Using route" for more information.

If you change the position of the controller from one node to another, you must remove the existing maps with the `vmmap` command, then reconfigure the network.

Verifying configuration

To verify your network's Ethernet configuration, enter

```
netstat -i
```

See "Verifying Ethernet configuration with netstat" for an explanation of output received from this command.

Defining an Ethernet with OpenBoot commands

Before booting the nodes of your Exemplar system, you must specify a logical unit number for each hypernode's Ethernet controller to OpenBoot. Steps 1 through 4 detail this process and must be performed for each Ethernet adapter on each Exemplar hypernode.

For more information about OpenBoot's device organization, see "OpenBoot's device tree" section on page 59.

Step 1 In each Exemplar System Console window, determine the full device name. Enter **`show-devs`** at the OpenBoot `ok` prompt. `show-devs` with no argument displays all devices known on the specified hypernode.

Step 2 Find the device path that includes the string `le` in the output from the `show-devs` command. Enter the entire line of text as *lestring* in the following command:

```
mkmap -n logical_unit_number lestring
```

where

`mkmap`

Defines an SPP-UX logical-unit to physical-unit mapping. This mapping is a label for tape and network devices that cannot be labeled like disks.

`-n`

Creates a logical-unit property with no physical unit information; this option should only be used with network adaptors.

logical_unit_number

Designates the logical-unit number of each Ethernet device.

Use the *interface* unit number you supply when you enable your network with the `ifconfig` command. For instance, if your interface is `ether0`, the *logical_unit_number* should be `0`. See the section "Defining an Ethernet with OpenBoot commands" for more information.

lestring

Represents the full Ethernet device name.

Your Ethernet device's logical unit number is assigned by OpenBoot at hypernode power on/reset time.

Step 3 Add your new logical unit by entering **`reset`** at the OpenBoot `ok` prompt.

Step 4 Boot the nodes of your Exemplar system by entering the proper command at the OpenBoot `ok` prompt in each Exemplar system console window. Refer to the SPP-UX *Release Notice* specific to your version of operating system for more information.

When SPP-UX boots, the Ethernet driver asks OpenBoot for its logical unit number and the driver keeps track of the number.

OpenBoot's device tree

You can determine the hardware configuration of your system by looking at the device tree, accessed from your system console, using the `show-devs` command. The device tree describes the hardware devices attached to each hypernode; its organization is similar to an SPP-UX file system tree.

Refer to OpenBoot documentation for more detailed information on the device tree.

As the device tree diverges, a route (or, device path) to individual devices becomes apparent. The device path represents the type of device and where that device is located within the device tree. A full device name is a series of device paths separated by slashes.

Device path syntax is

name@address : arguments

where

name

Represents an address on the main system bus. Valid names for Exemplar systems include:

`sd`
SCSI disk

`st`
SCSI tape

`le`
Ethernet

`CRES, cddi, fddi`
FDDI

`hippi`
HIPPI

`convex, afsw`
SCSI

@address

Represents a physical space unique to the device. *address* consists of two 32-bit numbers, usually in hexadecimal format, separated by a comma. The interpretation of these two numbers depends on the location of the device in the device tree.

:arguments

(Optional) Passes additional information to the device's software. Valid options are specific to each device package. *argument* usually shows additional device information, such as disk partition.

For more information on device path syntax, refer to OpenBoot documentation.

Configuring Ethernet with ifconfig

Before SPP-UX can communicate over a network interface, it needs to know the interface's network address and operating characteristics. This section explains how to use the `ifconfig` command to assign a network address to an Ethernet and set parameters that affect its operation.

Step 1 Determine the network address and operating characteristics you need.

Step 2 Log in as the superuser.

Step 3 Use the `ifconfig` command to test your choices from step one as options on the command line.

The `ifconfig` command is located in your /etc/netlinkrc file. Although the interface is automatically configured at boot time, you might have to modify this entry, for instance to enable `arp`. If so, test your `ifconfig` command first.

The `ifconfig` command has the following syntax:

`/etc/ifconfig` *interface address_family [address [dest_address]] [parameters]*

where

interface

Is a string composed of interface type, or name, and unit number like `le`. Logical unit numbers are assigned at boot time by the OpenBoot command `mkmap`. `netstat` displays the name and unit number of interfaces associated with your Exemplar system.

address_family

Is the name of the protocol on which your naming scheme is based. An interface can receive transmissions in differing protocols, each of which may require separate naming schemes. Therefore, it is necessary to specify the *address_family*, which may affect interpretation of the remaining parameters

on the command line. The only address family currently supported is `inet` (DARPA -Internet family).

address

Is either a host name present in the host name database or a DARPA Internet address expressed in Internet standard dot notation.

dest_address

Is the address of destination system. It consists of either a host name present in the host name database or a DARPA Internet address expressed in Internet standard dot notation.

parameters

Is any of the operating parameters listed in Table 3 on page 49.

In SPP-UX, the ifconfig default is `-trailers`; setting the `trailers` parameter has no affect.

Step 4 After you test your `ifconfig` options, verify that the system accepted the information.

Enter

`ifconfig` *interface*

In response, the system displays the network address and operating characteristics of the interface. Sample output is shown in Figure 19.

```
hippi0: flags=43<UP,BROADCAST,RUNNING>
        inet 130.150.70.3 netmask ffffff00 broadcast 130.150.60.255
        hardware address aa.00.04.00.2e.28
```

Figure 19 Using `ifconfig` to verify interface configuration

Step 5 If the network appears to be configured properly, edit your netlinkrc file to add the `ifconfig` command. This ensures that the network interface is enabled every time the system boots. Figure 20 shows a partial netlinkrc file containing an `ifconfig` command.

Step 6 (Optional) Add a `route` command to the /etc/netlinkrc file. See the section "Using route" for more information.

```
#
/bin/hostname moose
#
# build the networking streams stack
#
# configure network interface
#
if [ -f /etc/ifconfig ]; then
     /etc/ifconfig eth0 '/bin/hostname' up arp -trailers netmask 0xfffff00
fi
#
```

Figure 20 Sample /etc/netlinkrc file

Defining the broadcast address

The default broadcast address contains a host part of all ones. `ifconfig` enables you to change an interface's broadcast address. An Exemplar system can accept broadcasts with a host part of all zeros (for compatibility with systems that use BSD 4.2 broadcasts), because it transmits and receives broadcasts with the address set using the mask in `ifconfig`.

If a machine on your network does not understand the broadcast address you select, some utilities, such as `rwho`, fail. If this happens, use `ifconfig` to set the broadcast address to the same broadcast address for all machines on the network.

All machines that communicate with each other must use the same broadcast address, either all zeros or all ones. The current standard is a broadcast address of all ones, as in broadcast 128.194.255.255.

Defining the subnet mask

Ones in the subnet mask indicate bit positions to use for the combined network and subnet fields; zeros mark the positions of bits in the host field. You specify the mask as a single hexadecimal number with a leading 0x, or in dot notation. For example, specifying

```
netmask 0xffffff00
```

or

```
netmask 255.255.255.0
```

both indicate that you want 24 bits of combined network and subnet fields, and 8 bits of host number. For a class B network, this mask logically partitions your 16 bits of host number into an 8-bit subnet field and an 8-bit host field. If you do not supply a `netmask`, the mask is set according to the network class (A, B, or

C with 8, 16, or 24 bits of network part, respectively) of your chosen IP address.

To avoid confusion with broadcast addresses, do not use subnet numbers of all zeros or all ones.

Using the Address Resolution Protocol (ARP)

ARP maps logical internet addresses to physical Ethernet addresses by caching the logical mappings between dot notation addresses (as in the /etc/hosts file) and physical Ethernet addresses. If an interface requests mapping for an address not cached, ARP queues the message that requires the mapping, then broadcasts a message on the associated network to request the address. If a response is received, the new mapping is cached, and the queued message is transmitted.

If you want to communicate with a network host that does not use the ARP protocol, you must use the `arp` program to manually add address mapping information to the local ARP table. The `arp` program forces caching of an ARP table entry for a specific host, so that the ARP protocol does not transmit a packet to a host to get its Ethernet address.

`arp` has the following syntax:

`arp -s` *host_name e_address [temp] [pub]*

where

`-s`

Adds an entry to the ARP table.

host_name

Is a remote host as listed in the /etc/hosts file.

e_address

Is a physical Ethernet address of the remote host. This address is displayed by most systems at boot time and is usually printed on the network controller. You specify it as six hexadecimal numbers separated by colons, as in

```
08:00:20:06:dd:42
```

The entry will be permanent unless you also specify *temp*.

temp

Specifies that the ARP table entry is temporary.

pub

Specifies that the entry will be "published;" that is, this system will act as an ARP server, responding to requests for *host_name* even though the host address is not its own.

You can check the current status of the ARP table by entering

```
arp -a
```

The `arp` command has more options than are discussed here. For a complete summary, refer to the arp(8C) man page.

Tuning an Ethernet interface

When the system is booted, the initialization process reads a file containing parameters that affect the way SPP-UX handles CPUs and peripheral devices. Among other uses, these boot-time parameters enable you to optimize network performance.

Verifying Ethernet configuration with netstat

Successfully booting your Exemplar in multiuser mode indicates that you have correctly configured network interfaces. In most cases, the machine simply does not run in multiuser mode if you make a mistake during the configuration process. Of course, you should test the network after the system is up and running in multiuser mode.

To verify network configuration, enter

```
netstat -i
```

If the network is properly configured, the system displays output similar to that shown in Figure 21.

```
# netstat -i

Name    Mtu    Network    Address   Ipkt     Ierrs  Opkt     Oerrs  Collis
hippi0  65280  acme-net   acmes     1520512  0      1327049  0      3
```

Figure 21 Checking Ethernet configuration with `netstat -i`

The displayed host name and network name are specific to your installation. The name of the interface you just configured should appear. If it does not, you have a problem with your installation or your configuration of the network. In either case, use the troubleshooting procedures outlined in the chapters entitled "Troubleshooting Internet services", "Troubleshooting flowcharts", and "Troubleshooting NFS and NIS" to find the problem.

Using route

Add a `route` command to /etc/netlinkrc if your hypernode is used as a gateway or will use gateways.

The following shows a common use example of the `route` command where 130.168.85.254 represents the gateway.

```
# route add default 130.150-60.254
```

You may also use `netstat -r` to look at the route table contents.

`route` has the following syntax:

`route add net` *destination gateway*

where

`add`

Adds a route.

`net`

Specifies the type of destination address.

destination

Specifies the destination host system where packets will be routed. Can be a host name (official host name or alias), an Internet address in dot notation, or the keyword `default`, which represents the wildcard gateway routing. See the route(7) man page for more information about the keyword, `default`.

gateway

Specifies the gateway through which the destination is reached. Can be a host name (official host name or alias) or an Internet address in dot notation.

`route` has more options than are presented here. For a complete summary, refer to the route(1M) man page.

Configuring HIPPI 5

Configuring Convex HIPPI

When you add a network interface to your system or change characteristics of an existing one, you must integrate the device into SPP-UX.

This chapter explains how to integrate a newly installed or reconfigured High Performance Parallel Interface (HIPPI) into SPP-UX. Specifically, it explains how to:

- Use the `ifconfig` command to configure your network
- Use the `route` command to define gateways
- Use the `netstat` command to verify network configuration
- Set up HIPPI routing
- Tune network performance by modifying boot-time parameters

This chapter does not explain how to install network hardware; you must install the hardware before you begin any task described here.

SPP-UX does not support configuration of your HIPPI using HP-UX's system administration manager utility, SAM. You must perform the configuration manually.

Before you begin

Before you begin HIPPI configuration, you must install and test HIPPI hardware. For complete instructions on installing and testing HIPPI hardware, refer to the appropriate service guide and diagnostic manual.

Summary of steps

The following procedures summarize the steps required for you to configure HIPPI into your network and verify your configuration.

Configuring HIPPI into your network

To integrate your HIPPI system into SPP-UX, perform the following steps:

Step 1 Determine the network address and operating chracteristics you need.

Step 2 Log in as superuser.

Step 3 Test the `ifconfig` options you selected by entering them on the command line.

Step 4 After you test the `ifconfig` command, verify that the system accepted the information.

Step 5 If your network appears to be configured properly, edit your /etc/netlinkrc file to add the `ifconfig` command.

Step 6 (Optional) Add a `route` command to the /etc/netlinkrc file.

Verifying configuration

To verify your network's configuration, enter

```
netstat -i
```

See the section "Verifying HIPPI configuration with netstat" for an explanation of output received from this command.

Configuring HIPPI with ifconfig

Before SPP-UX can communicate over a network interface, it needs to know the interface's network address and operating characteristics. This section explains how to use the `ifconfig` command to assign a network address to a HIPPI and set parameters that affect its operation.

Step 1 Determine the network address and operating characteristics you need.

Step 2 Log in as the superuser.

Step 3 Use the `ifconfig` command to test your choices from Step 1 as options on the command line.

The `ifconfig` command is located in your /etc/netlinkrc file. Although the interface is automatically configured at boot time,

you might have to modify this entry, for instance to enable `arp`. If so, test your `ifconfig` command first.

The `ifconfig` command has the following syntax:

`/etc/ifconfig` *interface address_family [address [dest_address]] [parameters]*

where

interface

Is a string composed of interface type, or name, and unit number like `hippi0`. Logical unit numbers are determined by hardware jumpers. `netstat` displays the name and unit number of interfaces associated with your Exemplar system.

address_family

Is the name of the protocol on which your naming scheme is based. An interface can receive transmissions in differing protocols, each of which may require separate naming schemes. Therefore, it is necessary to specify the *address_family*, which may affect interpretation of the remaining parameters on the command line. The only address family currently supported is `inet` (DARPA -Internet family).

address

Is either a host name present in the host name database or a DARPA Internet address expressed in Internet standard dot notation.

dest_address

Is the address of the destination system. It consists of either a host name present in the host name database or a DARPA Internet address expressed in Internet standard dot notation.

parameters

Is any of the operating parameters listed in Table 3 on page page 49. At this time, HIPPI does not support `arp` parameters.

In SPP-UX, the ifconfig default is `-trailers`; setting the `trailers` parameter has no affect.

Step 4 After you test your `ifconfig` options, verify that the system accepted the information.

Enter

```
ifconfig hippi0
```

In response, the system displays the network address and operating characteristics of the interface. Sample output is shown in Figure 22.

```
hippi0: flags=43<UP,BROADCAST NOTRAILERS,RUNNING NOARP>
        inet 130.168.39.1 netmask ffffff00 broadcast 130.168.39.255
```

Figure 22 Using `ifconfig` to verify interface configuration

Step 5 If the network appears to be configured properly, edit your netlinkrc file to add the `ifconfig` command as shown below. This ensures that the network interface is enabled every time the system boots.

```
#
/bin/hostname moose
#
# build the networking streams stack
#
# configure network interface
#
if [ -f /etc/ifconfig ]; then
     /etc/ifconfig hippi inet '/bin/hostname' -h up -arp -trailers netmask 0xffffff00
fi
#
```

Figure 23 Sample /etc/netlinkrc file

Step 6 (Optional) Add a `route` command to the /etc/netlinkrc file.

Defining the subnet mask

Ones in the subnet mask indicate bit positions to use for the combined network and subnet fields; zeros mark the positions of bits in the host field. To avoid confusion with broadcast addresses, do not use subnet numbers of all zeros or all ones. You specify the mask as a single hexadecimal number with a leading 0x, or in dot notation. For example, specifying

```
netmask 0xffffff00
```

or

```
netmask 255.255.255.0
```

both indicate that you want 24 bits of combined network and subnet fields, and 8 bits of host number. For a class B network, this mask logically partitions your 16 bits of host number into an 8-bit subnet field and an 8-bit host field. If you do not supply a `netmask`, the mask is set according to the network class (A, B, or C with 8, 16, or 24 bits of network part, respectively) of your chosen IP address.

Using route

If HIPPI is used for point-to-point connections, you do not need to use the `route` command. When a switch is involved, however, the `route` command indicates how to get to each destination. The format of the `route` command for HIPPI is

`route add hippi host` *dest gateway metric mtu ifield*

where

add
: Adds a route.

dest
: Is the destination host

gateway
: Is the address of the host on which the HIPPI interface is installed

metric
: A count that indicates the number of hops to the destination

mtu
: Is maximum transmission unit (default is 65280)

ifield

Is a 32–bit entity (I-field) that is defined in RFC 1374. Bits relevant to the `route` command are as follows:

Path selection (bits 26, 25) can be 00, 01, or 11 (binary). 00 (source route mode) indicates that the I-field bits 23-00 contain a 24–bit source route; 01 or 11 (logical address mode) indicate that bits 23-00 contain 12–bit source and destination Addresses. The value 11 is meaningful when more than one route exists from a source to a destination; it allows the switch to choose the route. Use of 01 forces the switch to always use the same route for the same source/destination pair.

If logical address mode is used, the source address field (bits 23-12) is not used, and the destination address field (bits 11-0) contains the switch address of the destination. If source route mode is used, the routing control field (bits 23-00) contains the route to the destination.

Source routing examples

If you assume host A and host B are connected to port 0 and port 1 of a switch respectively, the path selection bits should be 00 for source routing. The `route` commands for host A would look similar to the following:

```
route add hippi host addr_A addr_A 0 65280 0
route add hippi host addr_B addr_A 0 65280 1
```

Figure 24 gives an example of the `route` command and shows output from a follow-up `netstat -r` command.

```
# route addhippi host zap-h zap-h 0 65280 0x01000006

# netstat -r
  Routing tables
Destination     Gateway      Flags      Refs      Use      Interface
localhost       localhost    UH         1         10       lo0
zap-h           zap-h        UH         0         0        hippi0
matahari-h      zap-h        UH         0         0        hippi0
130-168.85      zap-h        U          6         98       fddi0
```

Figure 24 Use of `route` and `netstat -r` commands

Logical routing examples

If more than one switch occurs between host A and host B, each port is used for the path selection bits. Assume host A is on port 0 of switch 1, switch 2 is on port 1 of switch 1 and host B is on port 4 of switch 2. The `route` command from host A to host B would look similar to the following:

```
route addhippi host addr_B addr_A 1 65280
0x000041
```

As the I-field is passed from switch 1, the routing control bits are shifted right by 4 bits.

Tuning Convex HIPPI

When the system is booted, the initialization process reads a file containing parameters that affect the way SPP-UX handles CPUs and peripheral devices. Among other uses, these boot-time parameters enable you to optimize network performance.

Most network-related boot-time parameters apply to networking in general, rather than to a specific type of network interface; a few are interface-specific.

When you install Convex HIPPI, related boot-time parameters are set to their default values. By modifying these parameters, you can tune resource requirements to achieve a balance between performance and resource usage.

You tune HIPPI by manipulating the boot-time parameters, `Fifo size` and `Direct Connection`, which are defined as follows:

`Fifo size`

Specifies the number of pages to allocate on the SIOP per I/O channel. Default is 4, minimum is 1, maximum is 16.

`Direct Connection`

Specifies that the HIPPI is not connected to a switch. Some hardware overhead is eliminated when this is On (1). Default is Off (0).

In addition, the SIOP tunable channels and memory are of interest. You must know the node, unit, and slot number of each HIPPI controller.

channels

Numbers of SIOP channels to allocate to the controller. Default is configuration-dependant. Minimum is 3; maximum is 256.

memory

Number of SIOP pages to allocate to this controller. Default is configuration-dependant. Minimum is 13; maximum is 256.

Verifying HIPPI configuration with netstat

Successfully booting your Exemplar in multiuser mode indicates that you have correctly configured network interfaces. In most cases, the machine will not run in multiuser mode if you make a mistake during the configuration process. You should test the network after the system is up and running in multiuser mode.

To verify network configuration, enter

```
netstat -i
```

If the network is properly configured, the system displays output similar to that shown in Figure 25.

```
# netstat -i

Name    Mtu    Network          Address        Ipkts Ierrs     Opkts Oerrs  Coll
hippi0 65280 130.168.39.4     zap-h            538  0         1     0      0
```

Figure 25 Checking HIPPI configuration with `netstat -i`

The displayed host name and network name are specific to your installation. The name of the interface you just configured should appear. If it does not, you have a problem with your installation or the configuration of your network. The troubleshooting chapters contain more complete instructions for using `netstat`.

Configuring ATM

6

Configuring an ATM interface

When you add a network interface to your system or change characteristics of an existing one, you must integrate the device into SPP-UX.

This section explains how to integrate a newly installed or reconfigured Asynchronous Transfer Mode (ATM) interface into SPP-UX. Specifically, it explains how to:

- Use OpenBoot commands to define the interface at hypernode power on/reset time
- Use the `ifconfig` command to configure your network
- Use the `pvcarp` command to assign routes to hosts on the ATM logical subnet
- Use the `netstat` and `ping` commands to verify network configuration

This chapter does not explain how to install network hardware; you must install the hardware before attempting any task described here.

SPP-UX does not support configuration of your ATM using HP-UX's system administration manager utility, SAM. You must perform the configuration manually.

Before you begin

Before you begin ATM configuration, you must:

- Install and test the ATM hardware

  For complete instructions on installing and testing ATM hardware, refer to the appropriate service guide and diagnostic manual
- Connect the ATM card to an active switch

Summary of steps

The following procedures summarize the steps required for you to define an ATM interface, configure your network, and verify your configuration.

Defining an ATM interface

To define your ATM interface at hypernode power on/reset time, perform the following steps:

Step 1 In each of your Exemplar System Console windows, determine the full device name with the `show-devs` command

Step 2 Make maps. Find the device path that includes the string *ia* in the output from the `show-devs` command. Using the `mkmap` command enter the entire line of text as *iastring*.

Step 3 Make sure your current directory is root.

Step 4 Add the new logical unit by entering the following command at the OpenBoot `ok` prompt.

```
reset
```

Configuring your network

To integrate your ATM system into SPP-UX, perform the following steps:

Step 1 Log in as superuser.

Step 2 Determine the network address and operating characteristics you need. See "Configuring ATM with ifconfig" section on page 79 for more information on `ifconfig`. The `-arp` option should always be used for ATM interfaces.

Step 3 Edit the /etc/hosts file and add a unique host name for the ATM interface

Step 4 Test the `ifconfig` options you selected by entering them on the command line.

Step 5 After you test the `ifconfig` command, use `netstat` to verify that the system accepted the information.

Step 6 If your network appears to be configured properly, edit your /etc/netlinkrc file to add the `ifconfig` command.

Step 7 Add a `pvcarp` command to the /etc/netlinkrc file. This command should follow the `ifconfig` line.

If you change the position of the controller from one node to another, you must remove the existing maps with the `rmmap` command, then reconfigure the network.

Step 8 Edit the /etc/atmarps file and add any other hosts, link numbers, and VCI's that should be included in the file.

Verifying configuration

To verify your network's ATM configuration, enter

```
pvcarp -a
```

```
netstat -r
```

and

```
ping
```

See "Verifying ATM configuration with netstat and ping" on page 82 for an explanation of output received from these commands.

Defining an ATM with OpenBoot commands

Before booting the nodes of your Exemplar system, you must specify a logical unit number for each hypernode's ATM controller to OpenBoot. Steps 1 through 3 detail this process and must be performed for each ATM adapter on each Exemplar hypernode.

For more information about OpenBoot's device organization, see "OpenBoot's device tree" section on page 78.

Step 1 In each Exemplar System Console window, determine the full device name with the `show-devs` command. Enter **`show-devs`** at the OpenBoot `ok` prompt. `show-devs` with no argument displays all devices known on the specified hypernode.

Step 2 Find the device path that includes the string `ia` in the output from the `show-devs` command, as shown in the following output example:

```
/mbus@0,ffec0000/sbus@f,fdffff00/ia@1,0
```

Step 3 Enter the entire line of text as *iastring* in the following command:

`mkmap -n` *logical_unit_number iastring*

where

`mkmap`

Defines an SPP-UX logical-unit to physical-unit mapping. This mapping is a label for tape and network devices that cannot be labeled like disks.

`-n`

Creates a logical-unit property with no physical unit information; this option should only be used with network adaptors.

logical_unit_number

Designates the logical-unit number of each ATM device.

Use the *interface* unit number you supply when you enable your network with the `ifconfig` command. For instance, if your interface is `ia0`, the *logical_unit_number* should be `0`. See "Defining an ATM with OpenBoot commands" on page 77 for more information.

iastring

Represents the full ATM device name.

Your ATM device's logical unit number is assigned by OpenBoot at hypernode power on/reset time.

Step 4 Make sure your current working directory is root.

Step 5 And add the new logical unit and reboot the hypernode by entering the following command at the OpenBoot `ok` prompt:

```
reset
```

OpenBoot's device tree

You can determine the hardware configuration of your system by looking at the device tree, accessed from your system console, using the `show-devs` command. The device tree describes the hardware devices attached to each hypernode; its organization is similar to an SPP-UX file system tree.

Refer to OpenBoot documentation for more detailed information on the device tree.

As the device tree diverges, a route (or device path) to individual devices becomes apparent. The device path represents the type of device and where that device is located within the device tree. A full device name is a series of device paths separated by slashes.

Device path syntax is

name@address : arguments

where

name

Represents an address on the main system bus. Valid names for Exemplar systems include:

`sd`

SCSI disk

`st`

SCSI tape

`le`
Ethernet

`ia`
ATM

`CRES, cddi, fddi`
FDDI

`hippi`
HIPPI

`convex, afsw`
SCSI

@address

Represents a physical space unique to the device. *address* consists of two 32-bit numbers, usually in hexadecimal format, separated by a comma. The interpretation of these two numbers depends on the location of the device in the device tree.

:arguments

(Optional) Passes additional information—such as disk partition—to the device's software. Valid options are specific to each device package.

For more information on device path syntax, refer to OpenBoot documentation.

Configuring ATM with ifconfig

Before SPP-UX can communicate over a network interface, it needs to know the interface's network address and operating characteristics. This section explains how to use the `ifconfig` command to assign a network address to an ATM and set parameters that affect its operation.

Step 1 Boot to single user mode by entering

```
boot -s
```

at the OpenBoot `ok` prompt.

Step 2 Determine the network address and operating characteristics you need.

Step 3 Edit the /etc/hosts file:

```
cd /etc
vi hosts
```

and add a unique host name for the ATM interface as shown in the following example:

Internet address atmhostname

Step 4 Test the `ifconfig` command with your choices from Step two as options on the command line.

The `ifconfig` command is located in your /etc/netlinkrc file. Although the interface is automatically configured at boot time, you might have to modify this entry, for instance to enable `arp`. If so, it is good practice to test your `ifconfig` command first.

The `-arp` option should always be used for ATM interfaces.

Step 5 The `ifconfig` command has the following syntax:

`/etc/ifconfig` *interface address_family [address [dest_address]] [parameters]*

where

interface

Is a string composed of interface type, or name, and unit number such as *ia*. Logical unit numbers are assigned at boot time by the OpenBoot command `mkmap`. `netstat` displays the name and unit number of interfaces associated with your Exemplar system.

address_family

Is the name of the protocol on which your naming scheme is based. An interface can receive transmissions in differing protocols, each of which may require separate naming schemes. Therefore, it is necessary to specify the *address_family*, which may affect interpretation of the remaining parameters on the command line. The only address family currently supported is `inet` (DARPA -Internet family).

address

Is either a host name present in the host name database or a DARPA Internet address expressed in Internet standard dot notation.

dest_address

Is the address of destination system. It consists of either a host name present in the host name database or a DARPA Internet address expressed in Internet standard dot notation.

parameters

Is either `-arp`, `up`, `down`, or `netmask`.

In SPP-UX, the ifconfig default is `-trailers`; setting the `trailers` parameter has no affect.

Step 6 After you test your `ifconfig` options, verify that the system accepted the information by entering the following:

`ifconfig` *interface*

In response, the system displays the network address and operating characteristics of the interface as shown in the following example:

```
ia0: flags=43<UP,BROADCAST,RUNNING>
inet 130.150.70.3 netmask ffffff00 broadcast 130.150.70.255
hardware address aa.00.04.00.2e.28
```

Step 7 If the network appears to be configured properly, edit your /etc/netlinkrc file to add the `ifconfig` command. This will ensure that the network interface is enabled every time the system boots.

```
vi netlinkrc
```

Add an `ifconfig` string after any existing `ifconfig` lines in the /etc/netlinkrc file as shown in the following example:

```
/etc/ifconfig ia0 inet atmhostname netmask
255.255.255.0 -arp up
```

For additional information, see "Defining the subnet mask" on page 82.

Step 8 Edit the /etc/netlinkrc file and add the `pvcarp` command anywhere after the ATM `ifconfig` you added in Step 7. The following example is a common use instance of the `pvcarp` command where /etc/atmarps is the name of the definition file.:

```
if [ -f /etc/atmarps ]; then
        /etc/pvcarp -f /etc/atmarps
fi
```

Set the Internet address to VC mapping used for Permanent Virtual Circuit (PVC) connections for traffic via ATM by adding a `pvcarp` command to /etc/netlinkrc anywhere after the ATM `ifconfig`. `pvcarp` has the following syntax:

`pvcarp -f` *filename*

where

`-f`

Uses the definitions in the file specified by *filename* to define or update an ARP/PVC definition in the driver.

filename

Specifies the name of the definition file where each line is in the format of the /etc/pvcarp -s version of the `pvcarp` command.

`pvcarp` has more options than are presented here, -s for instance. For a complete summary, refer to the pvcarp(1M) man page.

Step 9 Edit or create the /etc/atmarps file with the following command:

```
vi atmarps
```

Add any other hosts, link numbers, and VCI's that should be included to the file:

otheratmhost atm_addr

Defining the subnet mask

Ones in the subnet mask indicate bit positions to use for the combined network and subnet fields; zeros mark the positions of bits in the host field. You specify the mask as a single hexadecimal number with a leading 0x, or in dot notation. For example, specifying

```
netmask 0xffffff00
```

or

```
netmask 255.255.255.0
```

both indicate that you want 24 bits of combined network and subnet fields, and 8 bits of host number. For a class B network, this mask logically partitions your 16 bits of host number into an 8-bit subnet field and an 8-bit host field. If you do not supply a `netmask`, the mask is set according to the network class (A, B, or C with 8, 16, or 24 bits of network part, respectively) of your chosen IP address.

To avoid confusion with broadcast addresses, do not use subnet numbers of all zeros or all ones.

Verifying ATM configuration with netstat and ping

Successfully booting your Exemplar in multiuser mode indicates that you have correctly configured network interfaces. In most cases, the machine will not run in multiuser mode if you make a mistake during the configuration process. To ensure that you have configured the network correctly, test the network in single user

mode. Of course, you should test the network after the system is up and running in multiuser mode as well.

Step 1 To verify network configuration in single user mode, enter

hostname *your_host*

Step 2 Enter

netlinkrc

Step 3 Enter the `pvcarp` command with the `-a` option. In response, the system displays the network address and operating characteristics of the interface as shown in the following example:

```
pvcarp -a
```

```
   Device        IP Address      Link   VC     Rate(mbits)   Encap
-------------   -------------    ----   ---   -----------   -----
otheratmhost    200.200.200.1     0     100        ABR       ipllc
```

Step 4 Enter

netstat -r

If the network is properly configured, the system displays output similar to that shown in following example:

```
Routing tables
Destination      Gateway            Flags     Refs     Use Interface
otheratmhost     atmhostname        UH          0        0  ia0
default          xxxxxxxxx          UG         17        0  fddi0
???.???.???      hostname           U           0        0  fddi0
200.200.200      atmhostname        U         241        0  ia0
```

Step 5 Test the connections to other hosts using the `ping` command.

Enter

ping

If the network is working properly, the system displays output similar to that shown in following example:

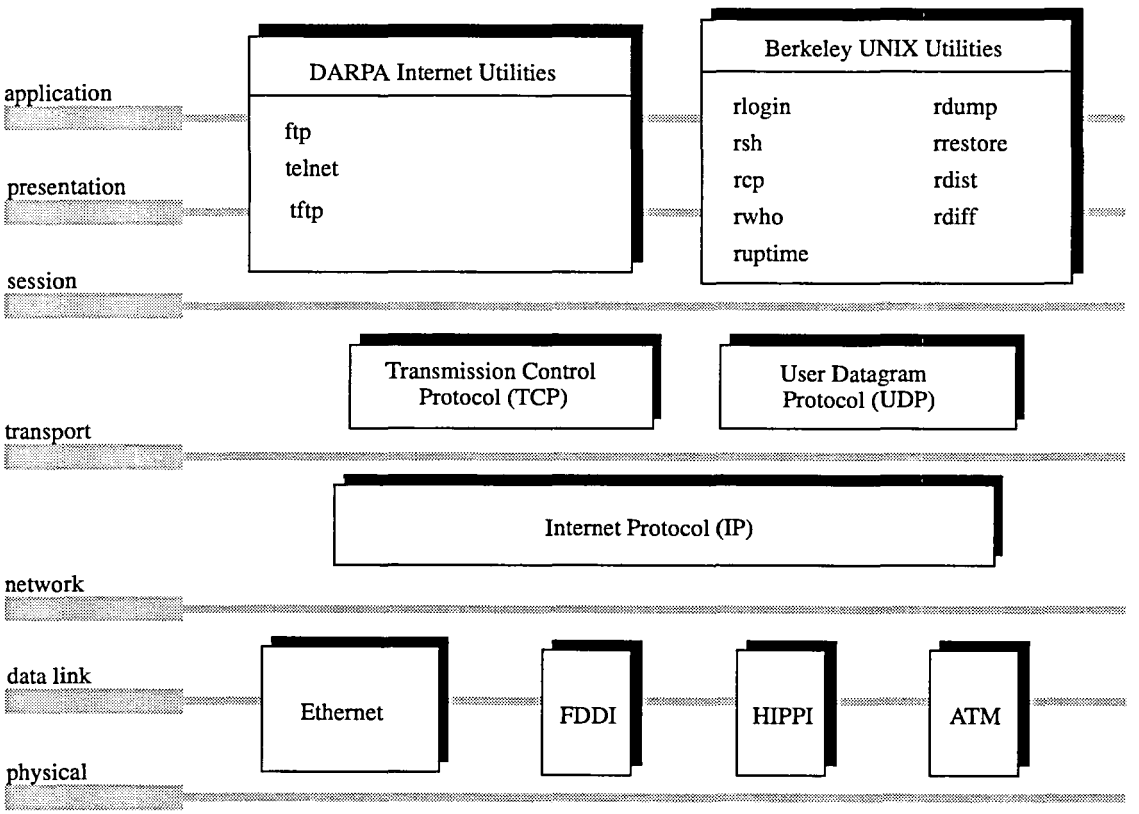
```
PING need-a: 64 byte packets
    64 bytes from 130.168.190.1: icmp_seq=0. time=4. ms
    64 bytes from 130.168.190.1: icmp_seq=1. time=1. ms
    64 bytes from 130.168.190.1: icmp_seq=2. time=1. ms
    64 bytes from 130.168.190.1: icmp_seq=3. time=2. ms

       ----need-a PING Statistics----
```

```
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 1/2/4 ^C
```

`ping` sends one datagram per second, and prints one line of output for every ECHO_RESPONSE returned. No output is produced if the remote host fails to respond. `ping` also reports round-trip times and packet loss statistics. See the "Troubleshooting Internet utilities" section in Chapter 10 for more information on `ping`.

Internet services



Setting up your network using Internet utilities 7

Overview of Internet utilities

This chapter lists prerequisites for configuring Internet utilities and describes how to do the following:

- Identify hosts on a network
- Choose an address structure
- Create the host name database
- Regenerate /etc/hosts and /etc/network
- Complete host name database setup

This chapter provides instructions for configuring and managing Internet utilities; it does not explain how to install hardware or software. Before you begin any procedure described in this chapter, make sure you have completed all the prerequisite hardware and software installation tasks described in the "Before you begin" section.

Terms

The following terms are used frequently in this chapter:

dot notation

Defines a network address as a series of decimal numbers (usually four) separated by periods, as 128.50.10.27.

/etc/hosts

Contains the database of host names and addresses for all hosts on your local network, as well as hosts on networks connected with yours.

/etc/networks

Contains logical names, addresses, and aliases for each network with which your local area network (LAN) communicates

rlogin

Remote login command, connects your terminal on the current host to a designated remote host.

subnetting

Partitioning of the network address space defined by a single network number into multiple virtual networks called *subnets* or *subnetworks*.

Summary of steps

Network management components include commands, daemons, and system files. You use these commands to define network interfaces, assign host names and addresses, test the network, and set up routing. You configure daemons to service network communication requests, and modify system files to reflect network configuration.

To configure Internet utilities, you must complete the following tasks:

Step 1 Integrate your network interface.

Step 2 Install network software according to the installation procedure shipped with the distribution tape.

Step 3 Choose local host names and Internet addresses and set up the host name database.

Step 4 Configure network interfaces with the `ifconfig` command.

Step 5 Configure and start the Internet superserver daemon, `inetd`.

Step 6 Set up and enable routing.

Step 7 (Optional) Configure the Berkeley Internet Name Domain server (BIND).

Step 8 Test the configured network and troubleshoot problems if necessary.

Subsequent sections in this chapter describe these tasks in detail.

Before you begin

Hardware prerequisites

Before you begin network configuration, make sure that you can answer "yes" to the following questions:

1. Has the hardware (cables, controllers, adapters) been completely installed and tested?
2. Have you run diagnostics on the test station?

   Instructions for running diagnostic tests are included in the diagnostic manual for each type of network interface.
3. Have you started the system in multiuser mode without problems?

Software prerequisites

Before you attempt any procedure described in this chapter, you must successfully install SPP-UX and utilities. Refer to the installation procedures that accompany your software distribution tape.

Where to find more information

This chapter provides instructions on how to manage Internet utilities—the tasks that you perform to set up and maintain Internet utilities on an Exemplar system.

If you are responsible for managing Internet utilities, you should also become familiar with the following Hewlett-Packard guides:

- *Using ARPA Services (DHP-175)*
- *Installing and Administering ARPA Services (DHP-176)*

Controlling access to the network

The following sections describe how to control users' access to network services.

Each machine on the network can accept or refuse logins by remote users. The system manager specifies whether users logging in from other machines must supply a password, or even if remote logins are permitted on a machine. Two files control this type of access: */etc/hosts.equiv* and *.rhosts*.

If your system runs NFS, the Network Information System (NIS) uses the /etc/netgroup file to control user access. For information about configuring NIS, refer to the "Network Information System

(NIS)" chapter in this book and Hewlett-Packard's *Installing and Administering NFS Services*, Chapter 7.

The hosts.equiv file

The /etc/hosts.equiv file contains a list of remote machine names that can access the local machine without using a password (using the same login name on both the local and remote machines). When you use `rlogin` to remotely log in to a machine and the local is not listed in the remote /etc/hosts.equiv file, you are prompted for a password unless the local machine is listed in the remote ~/.rhosts file. Figure 26 shows a sample /etc/hosts.equiv file.

```
acme1
dopey
sleepy
sneezy
doc
bashful
grumpy
```

Figure 26 Sample hosts.equiv file

To understand how /etc/hosts.equiv works, assume that a user from a remote host attempts to log in to the local host using `rlogin`. If the remote host is listed in the local /etc/hosts.equiv file and the remote user's login name matches a login name on the local host, the user is not prompted for a password.

If you are logged in as superuser, /etc/hosts.equiv is bypassed.

Security

For security reasons, access to the root account is not controlled by the /etc/hosts.equiv file. You must either have an appropriate entry in root's /.rhosts file, or you must supply the root password when you log in remotely. For security reasons, you should limit read access to the /etc/hosts.equiv file to root.

Creating a hosts.equiv file

Follow these guidelines when you create an /etc/hosts.equiv file:

- Each entry contains a valid official host name or a comment. If you have also installed NFS, an entry may contain a valid network group name.

- Host names can contain any characters except blanks, tabs, comment characters (#s), and newline.
- If you have also installed NFS, network group names must be preceded by the characters +@ or -@. Refer to the explanation below.

An entry in /etc/hosts.equiv takes the following form *host_name*, where *host_name* corresponds to a valid official host name as returned by `gethostbyaddr`.

Table 4 Group entries for /etc/hosts.equiv

Where	Means
group	The valid network group name defined in /etc/netgroup.
+@group	All hosts in that network group are allowed.
-@group	All hosts in that network are not allowed.
+	Everyone is allowed.

If you have also installed NFS, an entry in /etc/hosts.equiv can also take the form

...+*@group*

...-*@group*

...+

Table 4 lists group entries in an /etc/hosts.equiv file

The $HOME/.rhosts file

$HOME/.rhosts files can be created and configured by any user to specify remote login names that are equivalent to the user's login name. `rcp`, `remsh`, and `rlogin` may use $HOME/.rhosts. You must create this file for the home directory of the superuser account if you want to use equivalent login names.

The local host allows a remote user with a login listed in local $HOME/.rhosts file to login to that local user's account without specifying a password.

$HOME/.rhosts must be owned by the local user and must not be a symbolic link.

The /etc/hosts.equiv file is checked before $HOME/.rhosts. If an entry is found in /etc/hosts.equiv, $HOME/.rhosts is not checked. For NFS, -@ entries are not absolute. If a user is excluded

by a minus entry from hosts.equiv but included in .rhosts, then that user is allowed access.

If you are logged in as superuser, the /etc/hosts.equiv file is bypassed and the .rhosts file in your home directory is checked if it exists.

Creating $HOME/.rhosts

Use these guidelines when you create $HOME/.rhosts:

- Each entry contains a valid official host name and a remote login name.
- If you have also installed NFS, an entry may contain a valid network group name followed by a white space and a remote user name or group name.
- The host name and login name are separated by any number of tabs or blanks.
- If you have also installed NFS, network group names must be preceded by the characters +@ or -@.

Table 5 lists group entries for $HOME/.rhosts.

Table 5 Group entries for $HOME/.rhosts

Where	Means
group	The valid network group name defined in /etc/netgroup.
+@group	*remote_user_name* or all users in *remote_group_name* in network group are allowed.
-@group	*remote_user_name* or all users in *remote_group_name* in network are not allowed.

An entry in ~/.r hosts takes the following form:

```
host_name       remote_user_name
```

If you have also installed NFS, an entry in ~/.r hosts can also take the form

```
+@group remote_user_name | remote_group_name
-@group remote_user_name | remote_group_name
```

Each $HOME/.rhosts file must be owned by the user, with 0600 permissions. The user's home directory must be protected such that no one else can read it. Even if users do not want to use an .rhosts file, it is important to write-protect their home directories so no one can create an .rhosts file in another person's directory and then have unauthorized permissions to use that account.

$HOME/.netrc

$HOME/.netrc files can be created and configured by any user. This file is used to specify login names and passwords to remote hosts so that `rexec` and `ftp` can execute without prompting the user for a password.

Creating $HOME/.netrc

Follow these guidelines when you create $HOME/.netrc:

- Entries contain a valid official host name, a remote login name, and a password.
- Fields are separated by tabs, commas, or blanks.
- Host names are followed by only one logical name and password.

An entry in $HOME/.netrc takes the following form:

`machine` *host_name* `login` *remote_user_name* `password` *password*

For example, the $HOME/.netrc entry for the remote account

- on the host springs
- with the login name leanne
- with the password cradle

would look like

```
machine springs login leanne password cradle
```

Each $HOME/.netrc file should be owned by the user, with 0600 permission. The user's home directory should be protected so that no one else can read it or write to it.

The .rhosts file

Each user's home directory may also contain a private equivalence list in a file called .rhosts for use by `rlogin`. Entries in this file contain remote host names (not aliases) and user names specifying which users on which machines are permitted to log in without supplying passwords.

For example, if you use different login names on different machines, add entries to your .rhosts file containing your other login names and names of machines from which you want to access your local account. The example in Figure 27 shows a .rhosts file for a user named `rsmith` on the local machine who uses login names `smith` and `bobsmith` on remote machines.

```
aries smith
```

```
aries bobsmith
```

In the above example, both `smith` and `bobsmith` may access `rsmith's` account on `aries` without entering a password. Your .rhosts file will not be honored if it has either group or world write permissions.

You must use official host names (the first name listed in /etc/hosts file entries), not aliases, in the .rhosts file.

Setting up the hostname database

The following three sections describe how to set up a host name database for your local network:

- An introduction to host naming and addressing concepts, as well as descriptions of ways to represent host names and addresses.
- A discussion on choosing an address structure.
- A step-by-step guide to creating the host name database.

If you are familiar with Internet naming and addressing, you may want to skip to the section, "Creating the host name database," which guides you through steps for implementing an address structure.

Identifying hosts on a network

Internet addresses, also known as *network addresses* or *host addresses,* are numeric, universal identifiers assigned to machines on a TCP/IP network. While software works well with these compact addresses, people find them cumbersome and difficult to remember. *Host names* also identify machines on a network, but in a text form most people find easier to work with. Network software translates host names to Internet addresses before using them internally.

Internal representation of Internet addresses

Internet addresses are expressed as 32-bit values representing two fields: *network number,* which includes *address class,* and *host number.* Network number identifies the interface used by the host to communicate on the network. Address class determines the size (often called *address space* or *name space*) of the network in terms of how many hosts it can support. Host number refers to a specific machine on the network interface.

As Figure 27 illustrates, distribution of the 32 bits of address depends upon the address class defined in the high-order bits.

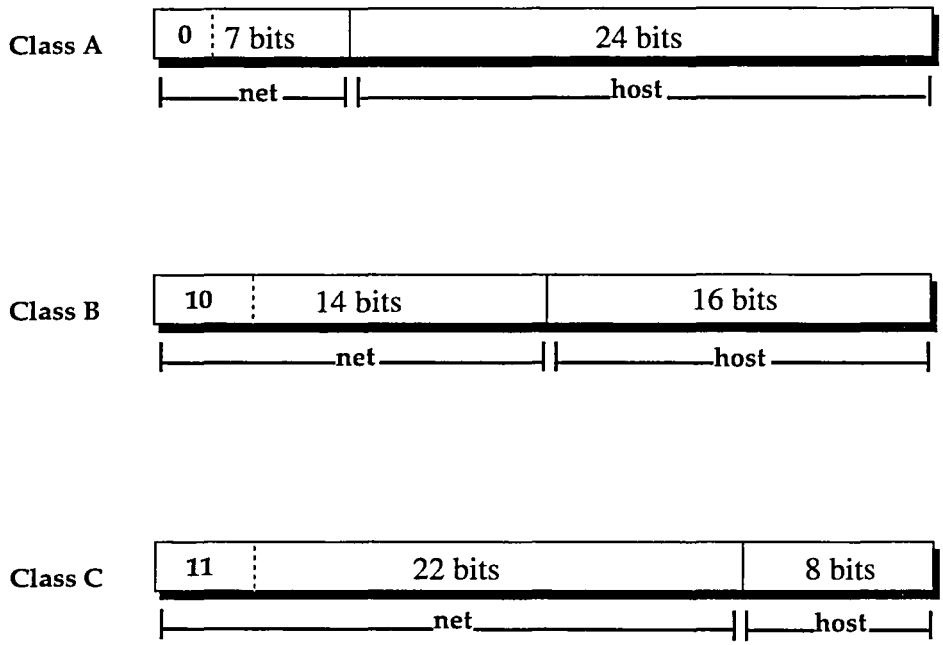


Figure 27 Internet addresses by class

Users typically designate hosts by name, and need not work with addresses. When configuring the network, however, the system manager must work with both names and addresses.

Dot notation

In situations where you identify hosts or networks by address, as in the /etc/hosts and /etc/networks files, you normally use *dot notation*. In dot notation, you specify an address as a series of decimal numbers (usually four) separated by periods, as in

```
128.50.10.1
```

In addition to dot notation, you can specify Internet addresses as decimal, octal, or hexadecimal numbers. Use a leading `0x` or `0X` to signify a hexadecimal address, and a leading `0` to signify an octal address. Numbers without prefixes are interpreted as decimal.

When the address is specified in four parts, each 8-bit value is encoded into the corresponding octet of the Internet address. Figure 28 illustrates this correspondence.

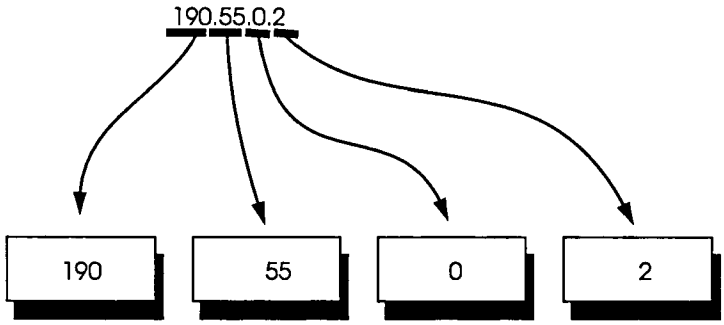


Figure 28 Four-part dot notation address

Address `190.55.0.2` has binary value `10` in its address class to show that it is a class B address. Network software interprets the first two octets of a class B address as the network number and the second two octets as the host number.

Three-part dot notation conveniently represents class B host addresses in the form

```
net.net.host
```

Figure 29 shows how host address, `190.55.2`, is encoded into a four-octet Internet address.

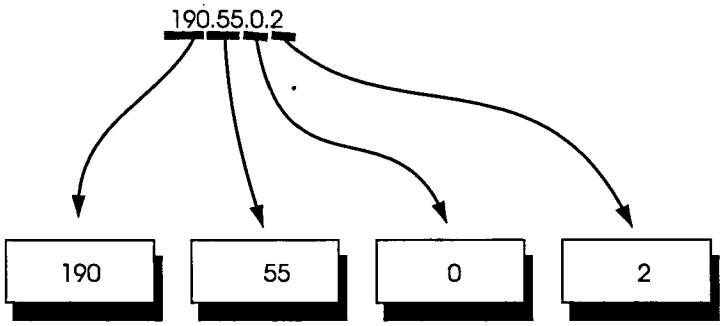


Figure 29 Three-part dot notation address

As you can see from the previous two figures, internal representation of host address `190.55.2` is equivalent to that of host address `190.55.0.2`.

You can also use three-part dot notation to distinguish subnet numbers from network numbers. Subnet addresses are discussed in the section "Dividing your address space into subnets."

Reserved addresses

Certain addresses have special meanings to network software and to other hosts on the network. For instance, the reserved localhost address is `127.0.0.1`.

Broadcast addresses

By default, a host number field of all ones signifies a *broadcast* address. A broadcast is a message sent to all hosts on the network represented by the other octets. For example, networked hosts share routing information with each other by broadcasting their routing tables.

Network addresses

An Internet address with a host number of all zeros refers to the network itself, rather than to a particular host. Class A network addresses are normally expressed in one-part dot notation. Class B network addresses are normally expressed in two-part dot notation, as in `140.55`. Class C network addresses are normally written in three parts, as in `140.55.20`, with each decimal value mapped to an octet of the network field; the host number field is set to zero by the software.

Host naming conventions

To keep matters simple, this discussion of names and addresses has implied that a host address identifies a unique computer. This is only partially true, because a host can communicate over more than one network interface. In such cases, a single address is not enough.

Termed *multihomed hosts*, machines connected to multiple network interfaces have multiple addresses, one per interface. This situation makes it apparent that an Internet address does not actually refer to a host computer, but to a connection—a path name that specifies a particular network interface on a particular machine.

Naming conventions help minimize confusion caused by multihomed hosts. One such convention consists of a root name identifying the host computer (for example, `hamlet`) followed by suffixes for the network connection (for example, `hamlet-f` for accessing it via an FDDI connection). This naming convention is illustrated in Figure 30.

```
#
# Host Database
#
127.0.0.1    localhost
# FDDI
190.56.10.2    ariel-fddi   ariel-f wind
# FDDI
190.56.11.3    ariel-fddi  ariel-h breeze
```

Figure 30 Naming hosts with multiple network connections

Unfortunately, this naming convention sometimes results in names that are not user-friendly. To compensate for this awkwardness, assign aliases that are both meaningful and friendly, as in the example above.

Choosing an address structure

The first step in configuring your network is to select names and addresses for hosts on your LAN.

If you were setting up a LAN that you did not want to connect to any other network, you could pick addresses at random or invent your own numbering scheme. However, if you choose addresses randomly and later, after your small, simple network has grown in size and complexity, you find that you want to connect to other networks, you face the difficult task of changing your address

structure to conform to the expectations of external networks. For this and other reasons, you should consider several factors before assigning addresses to hosts on your LAN. Among these factors are:

- Whether your LAN will connect to external networks
- The number of host addresses you need, with room for expansion
- Whether you will divide your network address space into subnets

Access to other networks

Internet protocols require each host to have a unique address. Unless you use "official" Internet addresses, you cannot guarantee uniqueness. Therefore, it pays to obtain official addresses before you set up your network. To do so costs nothing, and doing so permits you to expand the service area of your network as your needs grow.

After you have determined which address class you need, your site administrator should contact the organization in charge of the network and request the appropriate domain registration form. An organization that connects to multiple networks (such as CSNET, DARPA Internet and BITNET) should register with only one network. Contacts are:

DARPA Internet

Sites that are already on the DARPA Internet and need information about setting up a domain should contact `HOSTMASTER@NIC.DDN.MIL`. You can do so by writing to DDN Network Information Center, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025, or by sending email to bind-request@ucbarpa.Berkeley.EDU.

You may also want to be placed on the BIND mailing list, a mail group for people on the Internet who run BIND. The group discusses future design decisions, operational problems, and other related topics. Contact the same network address to be added to the mailing list.

CSNET

A CSNET member organization that has not registered its domain name should contact the CSNET Coordination and Information Center (CIC) for an application and information about setting up a domain.

An organization that already has a registered domain name should keep the CIC informed about how it would like its mail routed. In general, the CSNET relay prefers to send mail

via CSNET (as opposed to BITNET or the Internet). For an organization on multiple networks, this may not always be the preferred method. The CIC can be reached via electronic mail at `cic@sh.cs.net`, or by phone at (617) 873-2777.

BITNET

If you are on the BITNET and need to set up a domain, contact `INFO@BITNIC`.

Determining address space requirements

The potential size of your network influences how you choose addresses. Because your choice of address class imposes a hard limit on address space, you should choose a class generous enough to meet your needs as your network grows. Few networks require class A addresses (16,777,214 hosts), but many need a larger address space than the 254 hosts available on class C networks (host numbers of all zeros and all ones are reserved, as discussed in section "Reserved addresses"). Choosing an address class is a relatively easy decision to make before the network is set up and a difficult one to change later.

Because the total number of available network numbers in each class is limited (127 class A; 16,383 class B; and 4,194,303 class C), you should choose the largest class that can support your present needs and your expected growth.

Dividing your address space into subnets

Before you assign addresses, you should decide whether you want to use *subnetting*. Subnetting is the partitioning of the network address space defined by a single network number into multiple virtual networks called *subnets* or *subnetworks*. With subnetting, local networks can include multiple physical network interfaces, yet appear as a single entity to remote networks. Figure 31 illustrates a typical way in which a class B network address is partitioned into subnet addresses.

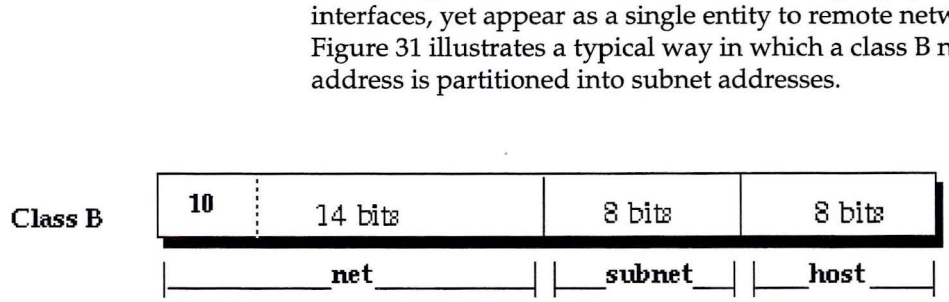


Figure 31 Subnet address partitioning

While this division into 8 bits for subnet number and 8 bits for host number is common for class B networks, you can partition the address any way you choose. The number of bits you give each field depends upon your needs. Will you have just a few large subnets, or is your installation more suited to having many small subnets?

Subnetted class B host addresses are commonly written in four-part dot notation as

```
net.net.subnet.host
```

When setting up subnets, assign addresses to your subnets as well as to the hosts they serve. Three-part dot notation conveniently represents a subnet address as

```
net.net.subnet
```

Figure 32 depicts one possible use of subnets.

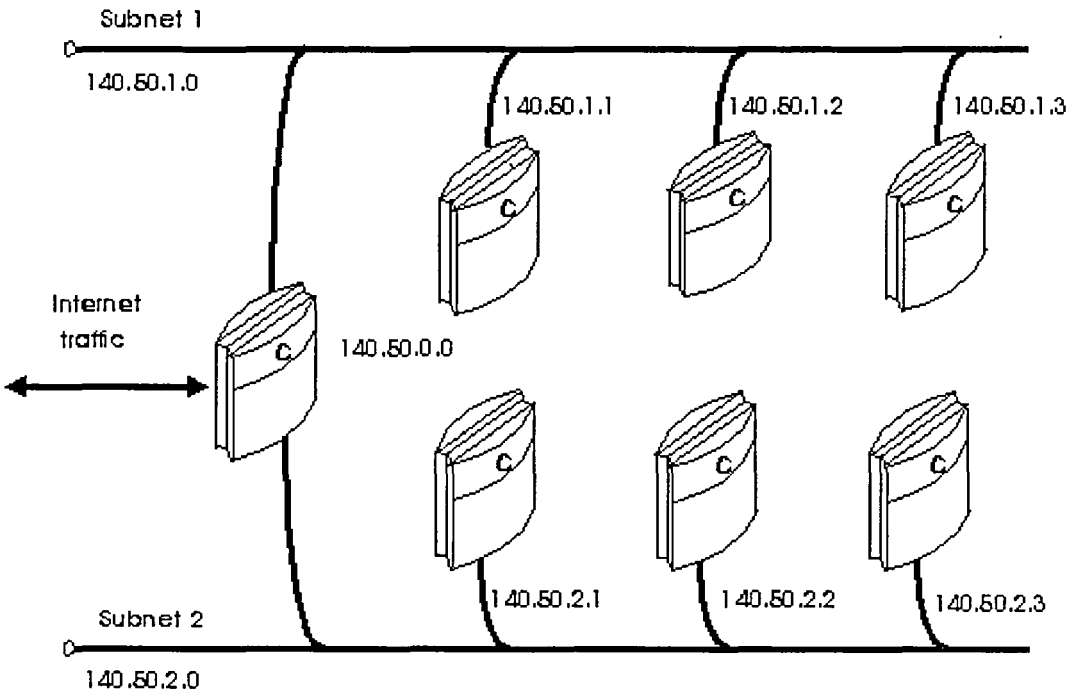


Figure 32 Subnets connected to an Internet by a gateway

You could achieve the same logical partitioning of a large local network by assigning different network numbers to each department's LAN. This method has several disadvantages, however. If a network consists of several physical LANs, you

could obtain one network number and partition it into subnets, or you could request a different network number for each LAN.

Though the decision to use subnets or multiple network numbers is yours to make, be aware that it has an impact on systems beyond your own. Each address class has a fixed number of networks it can support, limited by the number of bits in the network number field. If you use multiple network addresses, you may unnecessarily deplete the pool of network numbers.

Once you have made the decision to divide your address space into subnets, you must tell network software which bits to use as the network and subnet numbers. You do this with the `netmask` parameter when you configure the network interface with `ifconfig`.

Methods for partitioning address space into subnets are discussed later in this chapter.

Other considerations

How you structure your addresses also affects routing, because routing decisions are based on the network number portion of the address. All hosts on the same LAN should have the same network number.

Creating the host name database

Previous sections described ways in which hosts on a TCP/IP network are identified and discussed points to consider before deciding upon an address structure.

After choosing an address structure, create the host name database for your name space and enable a method of associating host names with addresses. Host names are associated with Internet addresses through a mechanism called *name resolution.* Network software uses one of two methods to resolve host names:

- Host table lookup method, whereby network software uses a static host name database containing explicit name-to-address translations. If you choose to use the host table lookup method to resolve names, your primary task is to maintain the /etc/hosts and /etc/networks files. This method requires you to update the /etc/hosts file on every networked host each time you add, remove, or change its address. Consequently, this method is practical only when your network is relatively small and stable.
- Name server method, whereby dynamic host name and address information is shared across the network. Internet

utilities provides the Berkeley Internet Name Domain (BIND) server system, consisting of a name server and an address resolver. When you use the name server method, you need only set up the host name database on a single host and configure other machines on the LAN to send queries to that host.

Regardless of which method you choose, you must create and maintain the /etc/hosts and /etc/networks files on at least one host on your LAN. The host name database must include all machines on your network.

To use the name server, you must also configure hosts on your LAN.

Modifying the /etc/hosts file

The /etc/hosts file contains the database of host names and addresses for all hosts on your local network, as well as hosts on networks connected with yours.

The first few lines of the /etc/hosts file distributed with Internet utilities are similar to those in Figure 33.

```
#
# Host Database
#
127.0.0.1               localhost
#
#
# The format is:
#Internet-address       official-hostname         aliases...
#
# Local Net -- FDDI
#
140.50.0.4              acme1 dopey
140.50.0.5              acme2 sleepy
140.50.0.6              acme3 sneezy
140.50.0.7              acme4 doc
140.50.0.9              acme5 bashful
140.50.0.10             acme6 happy cad
#
```

Figure 33 Sample /etc/hosts file

The /etc/hosts file contains two types of data: dot notation addresses (`140.50.0.9`) and logical host names (`acme5` and `bashful`).

The first number of the addresses in Figure 33 is between 128 and 191, signifying that the hosts belong to a class B network. This means that the 32-bit Internet address is divided into 16 bits of network number and 16 bits of host number.

In addition to dot notation, you can use decimal, octal, or hexadecimal numbers to specify Internet addresses. Use a leading `0x` or `0X` to signify a hexadecimal address, and a leading `0` to signify an octal address. Numbers without prefixes are interpreted as decimal.

Host names in Figure 33 are specified two ways: official host name and aliases. The official host name is listed first; aliases, if any, follow on the same line. For example, in the entry

```
acme6 happy cad
```

`acme6` is the official name, and `happy` and `cad` are aliases. Aliases enable you to reference hosts by symbolic names. For example, the command sequences `rlogin cad` and `rlogin acme6` connect you to the same machine.

The version of /etc/hosts initially installed on your system is large, containing many groups of host names. It is intended to serve as an example and to provide addresses of hosts you may want to access if you connect to the DARPA Internet. Because most of these host names have little to do with names and addresses you choose for hosts on your LAN, delete those you do not need. Then, modify /etc/hosts to reflect your own network configuration.

Add addresses, names, and aliases for each host to which your hosts need access. Internet Requests for Comments (RFCs) require all networks to register with NIC.DDN.MIL. However, if your LAN is not connected to external networks, you are free to choose any network and host numbers; just make sure that each host on any given network has the same network number and a unique host number.

If an official host name appears on more than one entry in the /etc/hosts file, the system uses only the first name. For example, if the /etc/hosts file contains the entries

```
140.50.0.4          acme1
140.50.0.4          cad
```

networking software will send messages destined for **`acme1`** to Internet address `140.50.0.4`.

An example of a customized host database is shown in Figure 34. The file contains names and addresses for machines on two LANs.

```
#
# The form for each entry is:
# <internet address>    <official hostname>    <aliases>
# For example:
# 192.1.2.34    hpfcrm    loghost
#
# See the hosts(4) manual page for more information.
# Note: The entries cannot be preceded by a space.
#       The format described in this file is the correct format.
#       The original Berkeley manual page contains an error in
#       the format description.
 127.0.0.1      localhost          localnet        loopback-net
# local FDDI; network number 140.50
 140.50.0.1     acme-f0            obewan          acmeq-f
 140.51.0.2     acme-f1            greeknet        acmer-f
```

Figure 34 Customized /etc/hosts file

Host names shown in Figure 34 consist of a machine name (for example, `acme-f0`) followed by two aliases (`obewan` and `acmeq-f`). The second alias has an appended interface designator (`-f`). Though you are not required to adhere to this naming convention, you may find it useful if you have multiple network interfaces, because it serves to identify both the machine and the interface.

Read more about the /etc/hosts file in the hosts(4) man page.

The /etc/hosts file will probably be identical for all hosts on the local network. If this is the case on your network, you can create the /etc/hosts file once and transfer it to all the other hosts.

Modifying the /etc/networks file

Just as the /etc/hosts file contains logical names, addresses, and aliases for each host to which your network is connected, the /etc/networks file contains logical names, addresses, and aliases for each network with which your LAN communicates. Several network utilities, including `netstat` and `routed`, use this file for mapping between network numbers and names. A typical /etc/networks file is shown in Figure 35.

```
#
# The form for each entry is:
# <official network name>    <network number>    <aliases>
# For example:
#
 arpanet                10                arpa
# Note: The entries cannot be preceded by a blank space.
#
 loopback-net           127.0.0.1         localnet
#
# Internet networks
#
 arpanet                10                arpa
#
```

Figure 35 Sample /etc/networks file

Network numbers shown in the example correspond to network numbers given in the sample /etc/hosts file in Figure 35.

Modify the /etc/networks file to include a logical name and network number for each network to be referenced from your machine. Be sure to use the same network numbers that you used in the /etc/hosts file. If your LAN has connections to other large networks such as the DARPA Internet, you must use officially assigned network numbers.

You may read more about the /etc/networks file in the networks(4) man page.

Creating subnets

Subnetting allows interconnected local networks to share a single network number, thus reducing the demand on the available pool of network numbers and simplifying routing for external hosts and gateways. With subnets, local networks can include multiple physical network interfaces, yet appear as a single entity to remote networks.

To implement subnets, divide the host number field of your LAN's Internet address into subnet and host portions by specifying a *subnet mask* with the `ifconfig` command used to configure the network interface. The subnet mask determines which part of the 32-bit Internet address is combined with the network number to identify the subnet. By default, the mask is set

to the size of the normal network number field for the particular network class (A, B, or C with 8, 16, or 24 bits of network part, respectively).

Within a subnetted local network, the subnet field is considered part of the network number. Outside the local network, the subnet field is not included in the network number.

Set the subnet mask at boot time in /etc/netlink with `ifconfig`. The section, "Configuring FDDI with ifconfig," explains how to use `ifconfig` for this purpose.

Figure 36 shows subnet partitioning in the /etc/networks and /etc/hosts files.


```
#
# Network Database (/etc/networks)
#
# The format is:
# network-name    Internet-address    aliases. . .
# Class B Network - 140.50.
# subnet mask - 255.255.255.0 (0xffffff00)
# 255 subnets with 255 hosts available
#
 acme-ex0              140.50.100              fishnet
 acme-ex1              140.50.200              astronet
 acme-ex2              140.50.300              faulknet
#
```

Figure 36 Subnet partitioning in the /etc/networks file

Figure 37 shows subnet partitioning in the /etc/hosts file.

```
# Host Database (/etc/hosts)
#
# The format is:
#Internet-address    official-hostname    aliases. . .
#
 127.1                 localhost
#
# Engineering Dept.  (fishnet 140.50.100.hh)
#
 140.50.100.10          eng1a          carp
 140.50.100.20          eng1b          tuna
 140.50.100.30          eng1c          sushi              squishy
 140.50.100.40          eng1d          squid              inky
#
# Marketing Dept.  (astronet 140.50.200.hh)
#
 140.50.200.10         eng2a          altair
 140.50.200.20         eng2b          sirius
 140.50.200.30         eng2c          betelgeuse         beetle
```

Figure 37 Subnet partitioning in the /etc/hosts file

Regenerating /etc/hosts and /etc/networks files

Internet host name databases are normally derived from a file retrieved from the Internet Network Information Center at SRI. The `gettable` program retrieves the NIC host database, and the `htable` program converts the data to the format that library routines use to resolve addresses. Both programs are located in the /usr/etc directory.

To retrieve and reformat the Internet host database table, change to the directory where you maintain local additions to the host table and execute the commands shown in Figure 38.

```
# /usr/etc/gettable nic.ddn.mil
Connection to sri-nic.arpa opened.
Host table received.
Connection to sri-nic.arpa closed.
# /usr/etc/htable hosts.txt
Warning, no localgateways file.
#
```

Figure 38 Using `gettable` and `htable` commands

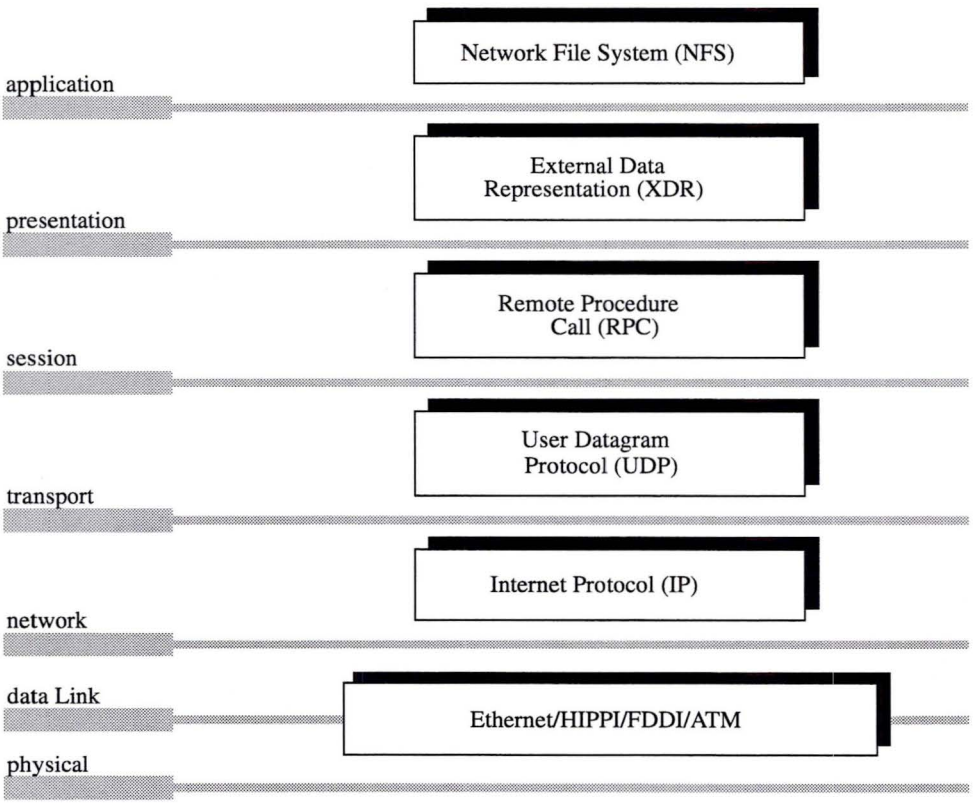
The `htable` program generates three data files: /etc/hosts, /etc/networks, and /etc/gateways. If a file named localhosts is present in the working directory, it is copied to the output file preceding the /etc/hosts file created by the `htable` program. Similarly, a file named localnetworks is copied to the /etc/networks file preceding the network data created by `htable`, and the localgateways file is copied to the front of the /etc/gateways data. Before installing new host and network databases in the /etc directory, run the `diff` file comparator on the old and new host databases to ensure that the files are complete.

If you use the name server for address resolution, you need only install the /etc/networks file and a small /etc/hosts file describing your local hosts. You may want to place the full host table elsewhere for reference by users. If you are connected to the DARPA Internet, we recommend that you use the name server, because it provides access to a much larger set of hosts than are in the host table. Many large organizations on the network that run the name server currently have only a small percentage of their hosts listed in the host table retrieved from the NIC.

Completing host name database setup

If your system uses the host table lookup method of name resolution, your host name database is configured and ready to use upon completion of the procedures outlined in this chapter. Otherwise, refer to to chapter entitled "Network Information System (NIS)".

Network file system



NFS utilities

8

Introduction to NFS

Network File System (NFS) utilities enable users to access remote file systems as if they were local. As the administrator of a system running NFS on Exemplar systems, your goal is to set up a distributed file system that is completely transparent to users. Users should be able to directly access files without knowing where data actually resides.

With the network services provided by NFS, network file administration is no more difficult than administration of local files on a time-sharing system.

This chapter explains how to administer services provided by NFS software. The following sections guide you through step-by-step tasks for setting up and running NFS on a SPP-UX system. These tasks include:

- Setting up a host to function as an NFS server
- Setting up a host to function as an NFS client
- Removing temporary files created by NFS
- Configuring portmap
- Obtaining status of portmap and RPC services

Services provided by NFS

NFS is a distributed file system that enables users to access remote file systems as if they were local and includes software to support the services listed below. You can configure hosts on your network to provide or to use some or all of these services, depending on the specific needs of your site.

Remote Procedure Call (RPC) facilities

Facilities that enable client processes to have another process, often running on a different host, execute a procedure call as

if the caller had executed the procedure call in its own address space.

portmap

A daemon that maps port numbers to RPC program numbers, providing a standard way for clients to look up the port number of RPC-based programs supported by a server.

NIS

A distributed lookup system that centralizes access to information in certain network configuration files otherwise maintained on each host.

REX

A utility used to execute commands on a remote system.

Terms

The following terms are used frequently in this chapter:

dot notation

Defines a network address as a series of decimal numbers (usually four) separated by periods, as 128.50.10.27.

/etc/biod

Starts NFS asynchronous block I/O daemons.

/etc/hosts

Contains the database of host names and addresses for all hosts on your local network, as well as hosts on networks connected with yours.

/etc/netnfsrc

Contains commands to automatically start NFS networking.

/etc/netnfsrc2

Contains commands to automatically mount remote NFS file systems and the automounter utility (if `NIS` is running). Do not change this file.

/etc/networks

Contains logical names, addresses, and aliases for each network with which your LAN communicates.

/etc/nfsd

Starts NFS filesystem requests daemons.

`rlogin`

Connects your terminal on the current host to a designated remote host.

subnetting

Partitions network address space defined by a single network number into multiple virtual networks called *subnets* or *subnetworks*.

Before you begin

This chapter does not discuss installing NFS software. Before you attempt any procedure described in this chapter, you must successfully complete the following tasks, in the order in which they are listed:

- Install SPP-UX and Utilities, which installs
  - Internet utilities
  - NFS utilities
  - NIS
- Configure Internet utilities

For information about installing the products listed above, refer to the installation procedures that accompany your software distribution tape.

For information about configuring Internet utilities, refer to the chapter entitled "Setting up your network using Internet utilities".

Summary of steps

Managing NFS requires you to perform the following tasks:

Step 1 Set up NFS server configuration and export local file systems.

Step 2 Set up NFS client configuration and mount remote file systems.

Step 3 Manage the RPC portmap daemon.

Step 4 Set up and manage NIS (optional).

Where to find more information

This chapter provides instructions on the tasks that you perform to set up and maintain NFS on an Exemplar system. If you are responsible for managing NFS, you should also become familiar with the following books:

- *Managing NFS and NIS*, by Hal Stern (O'Reilly & Associates, 1992. ISBN 0-937175-75-7)
- Hewlett-Packard guides:

- *Installing and Administering NFS Services* (DHP-173)
- *Programming and Protocols for NFS Services* (DHP-174)
- *Using NFS Services* (DHP-172)

Setting up an NFS server

An NFS server is a host that exports directories in its local file system, making them available to client hosts on the network. NFS servers control who may mount a directory by limiting named file systems to approved clients. Options can be used to further limit clients to particular types of access, such as read–only.

Any host that has a local file system can be configured as both an NFS server and an NFS client. An NFS server can be a client of another NFS server, but no server can act as an intermediary between a client and another server. That is, a server cannot export directories that have been remote-mounted from another server.

This section contains instructions for performing the following tasks:

- Configuring an NFS server for the first time
- Identifying file systems to be exported at boot time
- Exporting and unexporting file systems on demand

Before attempting to configure a host as an NFS server, you must successfully complete the tasks listed in the section "Before you begin."

Configuring an NFS server for the first time

This section describes daemons that provide NFS server functionality and lists steps to make sure the necessary daemons will be started at boot time. Information in this section is of interest only if you are configuring an NFS server for the first time.

NFS server daemons

Five daemons provide NFS server functionality.

`nfsd`

Receives file access requests from clients, performs the actual file system operation, and sends a response back to the client. If the client request is for a read operation, requested data is also returned.

An `nfsd` processes client requests one at a time. Completion of a request often involves delays, such as waiting for data to be retrieved from disk. While a server processes a request, it

ignores all others, causing other clients to retransmit requests until the server becomes free.

You can avoid this potential bottleneck by running multiple `nfsd` daemons. The number of `nfsd` daemons running on the server should correspond to the number of `biod` daemons running on the clients. The default number of `nfsds` is four; if the clients experience slow NFS response, you may want to increase this number.

`mountd`

Receives client mount requests and reads the server's xtab file to determine which clients are permitted to mount which file systems. If the client has permission to mount the requested file system, `mountd` returns a file handle to the client.

`portmap`

Maps the RPC program number of a service to a port number and provides port numbers in response to queries by clients.This port number then becomes part of the IP address that clients use to request a particular service.

Because NFS servers communicate with clients through RPC, you must ensure that the `portmap` daemon is running before you start NFS server daemons. For more detailed information about the service provided by `portmap` and instructions for running it, refer to the section "Managing portmap."

`rpc.lockd`

Provides file and record locking services for file systems exported to NFS clients. Receives locking RPC requests and acquires and releases local file system locks on the client's behalf. The Network Lock Manager must be informed whenever one of the NFS clients fails or reboots.The Lock Manager communicates with the Network Status Monitor (`rpc.statd`) to monitor client failure information.

`rpc.statd`

The Network Status Monitor uses a simple protocol to monitor the status of other hosts. Stateful services such as network file locking use the status monitor to determine when another host has failed so that they can perform appropriate recovery.

NFS server daemon startup verification procedure

Your system should be ready to function as an NFS server—all you need to do is identify the file systems to be exported; however, verify that necessary daemons are started at boot time by completing the following steps:

Step 1 Log in as superuser to the host you intend to use as an NFS server.

Step 2 Confirm that the `portmap`, `nfsd`, `rpc.mountd`, `rpc.statd`, and `rpc.lockd` daemons were started at boot time by entering the command shown in Figure 39.

```
# ps -fe|grep -e portmap -e nfsd -e rpc.mountd -e rpc.statd -e
rpc.lockd
root     1     0     17:44:03?     0:00   /etc/portmap
root     1     0     17:44:06?     0:00   /usr/etc/rpc.mountd
root     66    0     17:44:06?     0:00   /etc/nfsd 4
root     1     0     17:44:06?     0:00   /etc/nfsd 4
root     66    0     17:44:06?     0:00   /etc/nfsd 4
root     66    0     17:44:06?     0:00   /etc/nfsd 4
root     1     0     17:44:09?     0:00   /usr/etc/rpc.statd
root     1     0     17:44:10?     0:00   /usr/etc/rpc.lockd
root     805   0     10:46:30      ttys7  0:00 grep -e portmap -e b
```

Figure 39 Verifying daemon startup

If you did not initiate any of the NFS daemons at boot time, perform Step 3 to start the daemons manually. Also, perform Steps 4 and 5 to ensure that the daemons will start up automatically the next time your server reboots. If the daemons did start up properly, skip Step 3 through Step 5.

Step 3 Start up any daemons missing from the process listing you obtained in step 2. If no NFS server daemons started, start all daemons up manually:

```
/etc/portmap
/etc/nfsd 4
/usr/etc/rpc.mountd
/usr/etc/rpc.statd
/usr/etc/rpc.lockd
```

Step 4 Ensure that the `NFS_SERVER` variable is set to 1 in the `/etc/netnfsrc` script so that server daemons are started automatically when the host is booted, as shown in Figure 40.

```
##
#       NFS_CLIENT      --   1 if this node is an NFS client, 0 if not
#       NFS_SERVER      --   1 if this node is an NFS server, 0 if not
#       Note: it is possible for one host to be a client, a server, both or
#       neither!  This system is an NFS client if you will be NFS mounting
#       remote file systems; this system is a server if you will be
#       exporting file systems to remote hosts.
#       See Also:  nfsd(1M), mount(1M).
#
##
NFS_CLIENT=1
NFS_SERVER=1
```

Figure 40 Declaring client/server status in /etc/netnfsrc

The machine illustrated in Figure 40 is both a NFS client and server. Comments give information about editing the file and describe additional functions which are started in /etc/netnfsrc.

See Hewlett-Packard's *Installing and Administering NFS Services* (DHP-173) for more information about setting up an NFS client. Any changes you make to the netnfsrc script will take effect when the client is rebooted, causing `netnfsrc` to run.

Step 5 Verify that the server daemons start automatically when the host boots, by examining the /etc/netnfsrc file for lines similar to those shown in Figure 41, Figure 42, and Figure 43 and correcting any differences.

```
##
#       Start the portmapper -- should be started first
#       Note: portmap must be started before inetd!
#       Needs to be started on ARPA clients (only dependent upon inetd)
#       (See portmap(1M).)
##
if [ -f /etc/portmap ] ; then
    echo "\t/etc/portmap"
    /etc/portmap
    if [ $? -ne 0 ] ; then
        echo    "Error:  NFS portmapper NOT powered up" >&2
        exit 1
    fi
fi
(starting rpc.statd & rpc.lockd in /etc/netnfsrc)
```

Figure 41 Starting portmap in/etc/netnfsrc

```
##
#       Start mountd
##
if [ $NFS_SERVER -ne 0 -a $START_MOUNTD -ne 0 -a -f /usr/etc/rpc.mountd ] ;
then
    /usr/etc/rpc.mountd && echo "    starting up the mountd" && echo
"\t/usr/etc
/rpc.mountd"
    set_return
fi

##
#       Start the NFS daemons -- four was determined to be the optimal
#       number to start for performance reasons.
#       Note:   nfsd should only be started on NFS servers; only systems
#               with local file systems may be NFS servers.
#       (See nfsd(1M).)
##
if [ $LFS -eq 0 -a $NFS_SERVER -ne 0 -a -f /etc/nfsd ] ; then
    /etc/nfsd 4 && echo "    starting up the NFS daemons" && echo
"\t/etc/nfsd 4
"
    set_return
fi
```

Figure 42 Starting `mountd` and `nfsd` in /etc/netnfsrc

```
##
#       If this system is an NFS client or an NFS server, start the Lock
#       Manager daemon and the Status Monitor daemon.
##
if [ $NFS_CLIENT -ne 0 -o $NFS_SERVER -ne 0 ] ; then
    if [ -f /usr/etc/rpc.statd ] ; then
        /usr/etc/rpc.statd && echo "    starting up the Status Monitor
daemon" &
& echo "\t/usr/etc/rpc.statd"
        set_return
    fi
    if [ -f /usr/etc/rpc.lockd ] ; then
        /usr/etc/rpc.lockd && echo "    starting up the Lock Manager daemon"
&&
echo "\t/usr/etc/rpc.lockd"
        set_return
    fi
fi
```

Figure 43 Starting `rpc.statd` and `rpc.lockd` in /etc/netnfsrc

If your server does not include an entry for `portmap`, `rpc.mountd`, `nfsd`, `rpc.statd`, or `rpc.lockd`, use an editor to add the lines shown in Figure 43.

Any changes you make to the netnfsrc file will take effect at boot time, when `inetd` restarts.

Once you have verified that the server's netnfsrc is set up correctly, you need to identify the file systems the server will export. The sections that follow present instructions for exporting file systems.

Exporting file systems

You can use either or both of the following methods to cause an NFS server to export file systems:

- Identify file systems to be exported at boot time by including entries in the /etc/exports file.
- Export and unexport file systems explicitly with the `exportfs` command.

Export options

Regardless of which exporting method you use, a set of *export options* controls the way in which NFS clients access exported directories. You can specify export options within entries in the

exports file and on the `exportfs` command line. If you do not provide export options, exported directories are available for unrestricted use by any NFS client on the network. Export options are summarized as follows:

`ro`

Export directory read-only. (Default is read/write.)

`async`

Export directory asynchronously. (Default is synchronously.) Refer to "Operating NFS asynchronously," for more information.

`rw=hostname[:`*hostname*`]...`

Export the directory read-mostly. Read-mostly means exported read-only to most machines, but read-write to those specified. If not specified, the directory is exported read-write to all. Up to 256 *hostnames* can be specified.

`access=client[:`*client*`]...`

Limit access to named client(s). The default value allows any machine to mount the given directory.

`root=hostname[:`*hostname*`]...`

Grant root access to named client(s). You can specify a maximum of 256. Refer to "Allowing over-the-net root access." (Default is for no clients to be granted root access.)

`anon=`*uid*

Use *uid* as the effective user ID, if a request comes from an unknown user. Refer to "Allowing over-the-net root access."

In the above syntax, *client* can be either a host name or a *netgroup*. The `exportfs` utility first checks for the named client in the hosts file, then in the /etc/netgroup file. To use netgroups, the server must run `NIS`.

Options that require a more complete explanation are discussed in the following sections.

Operating NFS asynchronously

This section describes NFS operation on file systems exported with the `async` option specified.

NFS V2.0 servers normally operate synchronously—data is written to permanent storage before an NFS transaction is acknowledged. Synchronous operation enables the server to maintain file integrity despite crashes. In synchronous operation, clients do not distinguish between a server crash and slow server response; they simply retry until the transaction completes.

The benefit of synchronous operation is increased reliability; however, reliability is gained at a cost in performance. Each time a client makes a write request, the server writes not only the data buffer, but "in-core" copies of the inode and indirect blocks as well. When NFS operates asynchronously, making full use of the SPP-UX buffer cache, large files can be written several times faster.

Note

Before you begin asynchronous NFS operation, carefully consider the NFS environment and its ability to recover from inconsistent data if a server fails. Weigh carefully your need for higher performance against inconvenience and administrative effort required to recover when the server crashes. Be aware that clients may not detect the server's crash and may not be aware that data has been lost.

You enable asynchronous operation on a file system basis by specifying the `async` option in the exports file. For example, adding the following lines to the exports file enables asynchronous operation for /tmp and /usr/spool:

```
/tmp            -asynch
/usr/spool    -access=genghis:attila:leona,asynch
```

File systems served by asynchronous servers should be soft mounted, so that NFS returns an error if the server malfunctions. By using `mount` command options, you can set the time-out and retry count values so that the operation times out and returns an error before the server can be rebooted; default values for the time-out and retry `mount` options ensure this.

Refer to "Mounting remote file systems" and to the mount(1M) and checklist(4) man pages for more information about the use of `mount` options.

Allowing over-the-net root access

Under NFS, a server exports file systems it owns so clients may remotely mount them. When a client becomes superuser, it is by default denied permission on remote-mounted file systems, as shown in Figure 44.

```
# cd
# touch test1 test2
# chmod 777 test1
# chmod 700 test2
# ls -l test*
-rwxrwxrwx    1    jsbach        0    Mar 24 16:12          test1
-rwx------    1    jsbach        0    Mar 24 16:12          test2
# su
Password:
# touch test1
# touch test2
# touch: test2: Permission denied
# ls -l test*
-rwxrwxrwx      1    jsbach        0    Mar 21 16:16          test1
-rwx------      1    jsbach        0    Mar 21 16:12          test2
```

Figure 44 Over-the-net access—superuser failure

The problem appears during execution of a `set-uid root` program. Programs that run as root cannot access files or directories unless "others" have permission to do so.

Also, you cannot change ownership of remote-mounted files. Because users cannot do a `chown` command and root is treated as a normal user on remote access, only root on the server can change the ownership of remote files. For example, as yourself, you attempt to change ownership of a new program, `a.out`, that must be `set-uid root`. The `chown` command fails, as demonstrated in Figure 45.

```
# chmod 4777 a.out
# su
Password:
# chown root a.out
a.out: Not owner
```

Figure 45 Failure of `chown` command

To change ownership, you must log in to the server as root, then make the change. Or, you can move the file to a file system owned by your machine (for example /usr/tmp is always owned by the local machine) and make the change there.

If you are prepared to accept serious security risks, you may also solve this problem by enabling over-the-net root access. Enabling over-the-net root access allows client accounts superuser privileges on remotely-mounted file systems. You can use export options to enable over-the-net root access on a file-system-by-file-system basis. Before you consider using this method, however, carefully consider the security risks.

Caution

Enabling over-the-net root access, as discussed in the following paragraphs, poses serious security risks.

You can enable over-the-net root access for particular file systems by using the export options, `-anon=0` or `-root=` (refer to "Export options"). Export options can be specified either in the exports file or on the `exportfs` command line. Use export options as follows:

- To enable over-the-net root access to particular hosts on the access list, use the `-root=`*hostname* option. In the following exports file entry, root access is given to `convex2`, but not to `convex3`:

```
/usr -root=convex2,access=convex2:convex3
```

- To enable over-the-net root access for all hosts, specify the `-anon=0` option for each file system to be accessed by root. In the following example, the /usr file system has been modified to allow over-the-net root access by `convex2` and `convex3`:

```
/usr -anon=0,access=convex2:convex3
/fonts -access=convex4
/usr/spool -access=convex2:convex3:convex4
```

Identifying file systems to be exported at boot time

The start-up script, `netnfsrc`, runs the `exportfs` utility at boot time. `exportfs` reads the exports file, makes each listed directory available for remote mounting by approved NFS clients, and stores a list of currently exported file systems in the xtab file.

To cause the server to export file systems at boot time, you must create or modify the exports file, which identifies file systems to be exported, clients that may access exported file systems, and any restrictions on that access.

Follow these steps to export file systems[1] from an NFS server at boot time:

Step 1 Log in to the host you intend to use as an NFS server and become superuser.

Step 2 Create an entry in your /etc/exports file for each directory you want the server to export. (Because the exports file is not installed with NFS software, you must create it if you are setting up an NFS server for the first time on a given host.) The format of the exports file is

`directory` [*-export_option,export_option,...*]

where

`directory`

Specifies the mount point path name of the directory or file you want to export. *directory* must be local to the NFS server. You cannot export either a parent directory or a subdirectory of an exported directory within the same file system. For example, it would be illegal to export /usr and /usr/local if both directories reside on the same disk partition.

export_option

Controls the way in which NFS clients may access exported directories. If you do not provide options in the exports file, exported directories are available for unrestricted use by any NFS client on the network. If present, the list of options is preceded by a dash; options are separated by commas. See the exports(4) man page for supported options.

Figure 46 shows a few typical exports file entries.

```
/usr/local -ro                          # export read-only to all clients
                                        #
/usr2 -access=saki:sushi:boris          # limit access to named hosts
                                        #
/home -root=daisey:odie                 # allow read/write access by any
                                        # client grant root access to named
                                        # hosts
/share/suns -rw=admin:ops,-secure       #
                                        # export to all clients, but limit
                                        # write access to named hosts
                                        # and require use of Secure RPC
                                        #
```

Figure 46 Typical exports file entries

1. You cannot export a child directory of an already-exported directory.

In the example in Figure 46:

- Any NFS client can read /usr/local, but none can write to it.
- Only `saki`, `sushi`, and `boris` can access /usr2.
- Any NFS client can read and write /home, but only `daisey` and `odie` have root access.
- Any NFS client can read /share/suns, but only `admin` and `ops` can write to it. All clients must use Secure RPC when accessing this directory.

Refer to the exports(4) man page for more information about this file.

Changes made to the exports file will not take effect until you either reboot the server or run `exportfs` from the command line, as described in the following section.

Exporting and unexporting files on demand

This section describes the `exportfs` command and provides examples for using `exportfs` to export or unexport file systems on demand.

Syntax of the `exportfs` command is
`exportfs [-avuif] [-o` *export-options*`] [`*dir*`]`

where

`a`

Exports all directories listed in the /etc/exports file. If you also specify `-u`, unexport all currently exported files.

`i`

Ignores options found in the /etc/exports file. Used to override exports file options with options specified on the `exportfs` command line.

`u`

Unexports the indicated directories.

`v`

Runs in verbose mode, printing the name of each directory as it is exported or unexported.

`o` *export-options*

Overrides options in the exports file, or applies to the *dir* specified on the command line. Options are separated by commas. Identical to the exports file options (refer to the section "Export options").

dir

Specifies the directory you wish to make available to NFS clients.

You can run `exportfs` at any time to export or unexport file systems on demand. For example, if you have just added entries to the exports file and you want to give clients access to the additional file systems without waiting for the server to be rebooted, you can cause the server to immediately export all directories listed in the updated exports file by entering

```
exportfs -a
```

In addition to forcing a modified exports file to take effect, there are other situations in which you may want to run `exportfs` from the command line. For example,

- To make a directory not listed in the exports file available immediately and without rebooting, enter

  `exportfs -o` *export_options dir*
- To change the options of a currently exported directory, enter

  `exportfs -i -o` *overridden.options dir*
- To unexport a directory, enter

  `exportfs -u` *dir*
- To unexport all directories, enter

  `exportfs -au`

Because `exportfs` stores the list of directories available to NFS clients in the xtab file, after running `exportfs -a`, your xtab file is identical to your exports file. Later, if you run `exportfs` from the command line and specify a directory, the xtab file will reflect the changes you made. For example, you reboot your system, causing `exportfs` to make available the directories listed in the example exports file shown in Figure 47. Your xtab file is now identical to that in Figure 47.

Later, a user on an NFS client requests read-only access to /usr/bin/games, so you run `exportfs` from the command line by entering

```
exportfs -o ro /usr/bin/games
```

Your xtab file now contains all entries found in the exports file, plus an entry for /usr/bin/games, as shown in Figure 47.

```
/usr/local -ro                          # export read-only to all clients
                                        #
/usr2 -access=saki:sushi:boris          # limit access to named hosts
                                        #
/home -root=daisey:odie                 # allow read/write access by any
                                        # client grant root access to named
                                        # hosts
/share/suns -rw=admin:ops,-secure       #
                                        # export to all clients, but limit
                                        # write access to named hosts
                                        # require use of Secure RPC
                                        #
/usr/bin/games -ro
```

Figure 47 Sample /etc/xtab file

Any time you run `exportfs -a`, the contents of the exports file replace the contents of the xtab file, overwriting any xtab file entries you created by running `exportfs` from the command line. Also, any xtab entries you create by running `exportfs` from the command line are overwritten when the system is rebooted.

Refer to the exportfs(1M), exports(4) and xtab(4) man pages for more information about the use of the `exportfs` command.

Setting up an NFS client

An NFS client is a host that accesses remote directories exported by one or more NFS servers.

Any host that has a local file system can be configured as both an NFS server and an NFS client. An NFS server can be a client of another NFS server, but no server can act as an intermediary between a client and another server. That is, a server cannot export directories that it has remote mounted from another server.

This section contains instructions for performing the following tasks:

1. Configuring an NFS client for the first time
2. Creating local mount points for remote directories
3. Mounting remote directories

Before you begin to configure a host as an NFS server, you must successfully complete the tasks listed in the section "Before you begin."

Configuring an NFS client for the first time

This section describes daemons that provide NFS client functionality and lists steps make sure the necessary daemons will be started at boot time. Information in this section is of interest only if you are configuring an NFS client for the first time.

NFS client daemons

These daemons provide NFS client functionality.

`portmap`

Maps the RPC program number of a service to a port number and provides port numbers in response to queries by clients. This port number then becomes part of the IP address that clients use to request a particular service.

Because NFS servers communicate with clients through RPC, you must ensure that the `portmap` daemon is running before you start NFS server daemons. For more detailed information about the service provided by `portmap` and instructions for running it, refer to the section "Managing portmap."

`biod`

Assists in processing large client I/O requests by breaking the requests up into NFS blocks and performing the block requests in parallel. Because each `biod` can handle only a single bock at a time, it is possible for all `biods` to be occupied while they wait for data from the server. To ensure that biods are always available to assist I/O requests, run multiple `biods`. The default number is four; if you experience slow NFS client performance, increase that number.

`rpc.lockd`

Provides file and record locking services for file systems exported to NFS clients. Receives locking RPC requests and acquires and releases local file system locks on the client's behalf. Since locking is an inherently stateful service, the Network Lock Manager must be informed whenever one of the NFS clients fails or reboots.The Lock Manager communicates with the Network Status Monitor (`rpc.statd`) to client monitor failure information.

`rpc.statd`

The Network Status Monitor uses a simple protocol to monitor the status of other hosts. Stateful services such as network file locking use the status monitor to determine when another host has failed so that they can perform appropriate recovery.

NFS client daemon startup verification procedure

Follow these steps to verify that NFS client daemons are started at boot time:

Step 1 Log in to the host you intend to use as an NFS client and become superuser.

Step 2 Confirm that the `biod`, `portmap`, `rpc.statd`, and `rpc.lockd` daemons were started at boot time by entering the commands shown in Figure 48.

```
# ps -fe|grep -e portmap -e biod -e rpc.statd -e rpc.lockd
  root     60        1  0 17:44:03?        0:00     /etc/portmap
  root     71        1  0 17:44:07?        0:00     /etc/biod              4
  root     72        1  0 17:44:07?        0:00     /etc/biod              4
  root     73        1  0 17:44:07?        0:00     /etc/biod              4
  root     74        1  0 17:44:07?        0:00     /etc/biod              4
  root     77        1  0 17:44:09?        0:00     /usr/etc/rpc.statd
  root     79        1  0 17:44:10?        0:00     /usr/etc/rpc.lockd
  root     5481805   0 10:46:30            ttyp0:00 grep -e portmap -e bio
```

Figure 48 Confirming startup of NFS client daemons

If your system did not initiate any of the NFS daemons at boot time, perform Step 3 to start up the daemons manually. Also, perform Step 3 and Step 5 to ensure that the daemons will start up automatically the next time your client reboots.

If the daemons did start up properly, you may skip Step 3 through Step 5.

Step 3 Start up any daemons missing from the process listing shown in step two. If no NFS client daemons started, start up all daemons manually, using the following commands:

```
/etc/biod 4
/etc/portmap
/usr/etc/rpc.statd
/usr/etc/rpc.lockd
```

Step 4 Ensure that the NFS_CLIENT variable is set to 1 in the /etc/netnfsrc script so that the client daemons are started up automatically when the host is rebooted, as shown in Figure 49.

```
##
#       NFS_CLIENT          --    1 if this node is an NFS client, 0 if not
#       NFS_SERVER          --    1 if this node is an NFS server, 0 if not
#       Note: it is possible for one host to be a client, a server, both or
#       neither!  This system is an NFS client if you will be NFS mounting
#       remote file systems; this system is a server if you will be
#       exporting file systems to remote hosts.
#       See Also:  nfsd(1M), mount(1M).
#
##
NFS_CLIENT=1
NFS_SERVER=1
##
```

Figure 49 Declaring client/server status in /etc/netnfsrc

The machine illustrated in Figure 49 is both an NFS client and server. Comments (lines which begin with a # sign) give information about editing the file and describe additional functions which are started in /etc/netnfsrc.

See Hewlett-Packard's *Installing and Administering NFS Services* (DHP-173) for more information about setting up an NFS client. Any changes you make to the netnfsrc script will take effect when the client is rebooted, causing `netnfsrc` to run.

Step 5 Verify that the client daemons start up automatically when the hosts boots next by examining the /etc/nfsrc file for line similar to those shown below and correcting any differences.

Figure 50, Figure 51, and Figure 52 illustrate the commands that initiate client daemons. These utilities initiate every daemon needed to enable NFS utilities at boot time.

```
##
#       If this system is an NFS client, start the BIO daemons -- four
#       was determined to be the optimal number to start for performance
#       reasons.
#
#       (See mount(1M) and nfsd(1M).)
##
if [ $NFS_CLIENT -ne 0 ] ; then
        if [ -f /etc/biod ] ; then
                /etc/biod 4 && echo "     starting up the BIO daemons" &&
echo "\
t/etc/biod 4"
                set_return
fi
        /bin/cat /dev/null > /etc/nfs.up
fi
```

Figure 50 Starting `biod` from /etc/netnfsrc

```
##
#       Start the portmapper -- should be started first
#       Note: portmap must be started before inetd!
#       Needs to be started on ARPA clients (only dependent upon inetd)
#       (See portmap(1M).)
##
if [ -f /etc/portmap ] ; then
    echo "\t/etc/portmap"
    /etc/portmap
    if [ $? -ne 0 ] ; then
        echo    "Error:  NFS portmapper NOT powered up" >&2
        exit 1
    fi
fi
```

Figure 51 Starting `portmap` from /etc/netnfsrc

```
##
#       If this system is an NFS client or an NFS server, start the Lock
#       Manager daemon and the Status Monitor daemon.
##
if [ $NFS_CLIENT -ne 0 -o $NFS_SERVER -ne 0 ] ; then
    if [ -f /usr/etc/rpc.statd ] ; then
        /usr/etc/rpc.statd && echo "     starting up the Status Monitor
daemon" &
& echo "\t/usr/etc/rpc.statd"
        set_return
    fi
    if [ -f /usr/etc/rpc.lockd ] ; then
        /usr/etc/rpc.lockd && echo "     starting up the Lock Manager
daemon" &&
echo "\t/usr/etc/rpc.lockd"
        set_return
    fi
fi
```

Figure 52 Starting rpc.statd and rpc.lockd in /etc/netnfsrc

Once you have verified that the client's /etc/netnfsrc file is set up correctly, identify the file systems the client will remotely mount. The sections that follow present instructions for mounting remote file systems.

Creating mount points

Before an NFS client can mount a remote directory, a *mount point* for that directory must exist in the local file system. A mount point is the location in the local directory structure at which the remote directory is accessed once it is mounted. For example, in Figure 53, /usr is the mount point at which remote directories /usr/bin, /usr/lib, and /usr/man are accessed.

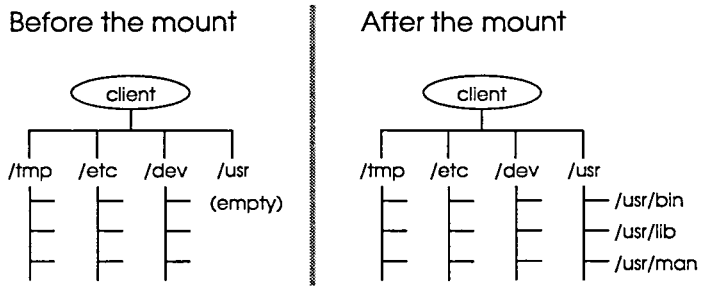


Figure 53 Effect of remote mounts on the client file system

Follow these steps to create NFS mount points:

Step 1 Log in to the host you intend to use as an NFS client and become superuser.

Step 2 Use the `mkdir` command to create mount points for all directories to be remotely mounted from NFS servers. For example, to create the mount point for the remote directory structure shown in Figure 53, enter

```
cd /
mkdir usr
```

Step 3 Use the `chmod` command on the mount point to enable read and execute permissions for others (drwxrwxr-x). If you do not, attempts to determine the current working directory with the `pwd` command in the mounted file system may fail.

Note

Do not set up remote mounts and soft links to root; SPP-UX attempts to resolve links constantly, which could result in a hung process if a server is down.

Mounting remote file systems

An NFS client can mount any exported file system if:

- The client can communicate with its server over the network.
- The client is granted access to the file system in the server's /etc/exports file.

NFS clients access remote directories through any combination of the following methods:

1. Automatic mounting at boot time

At boot time, a client mounts remote directories to which it needs access by including specific entries in its checklist file.

2. Explicit mounting on demand

   A superuser on the client can explicitly mount and unmount remote directories at any time with the `mount` command.

You can configure an NFS client to use any or all of these methods, depending on factors such as the frequency with which a directory is used by remote systems. In all likelihood, you will mount some remote directories at boot time and others with the `mount` command. Because users' needs for access to remote directories change over time, you will need to periodically reassess your mounting strategy.

As a general rule,

- Use the /etc/checklist file to automatically mount directories to which networked hosts need more or less continuous access.

For example, if hosts on your network share utilities located in /usr/local/bin, it would be a good candidate for automatic mounting at boot time.

- Use the `mount` command to explicitly mount and unmount sporadically used directories.

The following sections outline procedures for setting up an NFS client to mount remote directories with the `mount` command, whether automatically at boot time or explicitly on demand.

Setting up the checklist file

Follow these steps to mount remote file systems:

Step 1 Log in to the host you intend to use as an NFS client and become superuser.

Step 2 Create an entry in the /etc/checklist file for each remote directory you want the server to mount at boot time. The format of the checklist file is

fsname dir `nfs` *mount_options*

where

fsname

Specifies the name of remote file system to be mounted at mount point *dir*. For NFS file systems, *fsname* takes the form *server:path*.

server

Is the name of the NFS server on which file system *path* resides.

dir

Specifies the full path name of the local directory on which the remote file system will be mounted.

`nfs`

Specifies a remote NFS-type file system is to be mounted.

mount_options

Specifies a list of comma-separated options. Most checklist file options are identical to `mount` command *mount_options*. Refer to the list of mount options in Table 6, or to the `mount`(1M) man page for information about *mount_options*.

Figure 54 shows typical checklist entries for NFS-served file systems.

```
michael:/usr2 /usr2 nfs rw,hard 0 0
bart:/usr/man /usr/man nfs rw,soft 0 0
bonnie:/usr/docs /mnt nfs rw,intr,bg 0 0
zainab:/usr/man /usr/man nfs rw,soft 0 0
```

Figure 54 Checklist entries for NFS file system mounts

Step 3 Examine the /etc/netnfsrc2 startup file to confirm the existence of `mount` commands.

For all file systems listed in the checklist file to be mounted at boot time, `netnfsrc2` must run the `mount -a` command. Confirm the existence of both of the following commands:

```
/etc/umount -at nfs
/etc/mount -at nfs
```

The `umount` command ensures that all NFS file systems are unmounted prior to attempting to mount them.

Step 4 (Optional) Mount the file systems listed in the checklist file without rebooting the client (for example, after modifying the checklist file) by entering the following commands at your system prompt:

```
/etc/umount -at nfs
/etc/mount -at nfs
```

Step 5 Verify that you have successfully mounted a file system where you expected to, by entering either the `df` or `mount` commands without arguments. Both display information about currently mounted file systems, but in different formats.

Using the mount command

Use the `mount` command to mount a file system on demand. File systems mounted explicitly with the `mount` command remain mounted until you explicitly unmount them or reboot the client.

Step 1 Log in to the host you intend to use as an NFS client and become superuser.

Step 2 Mount remote file systems by entering `mount` commands according to the following syntax:

`mount [-afvp] [-o` *mount_options*`] [`*fsname*`] [`*dir*`]`

where

`a`

Mounts all directories listed in the checklist file. If you also specify `-u`, unexport all currently exported files.

`f`

Forces the file system to be mounted.

`v`

Runs in verbose mode, printing the name of each directory as it is exported or unexported.

`p`

Prints a tabular list of mounted file systems suitable for use in /etc/checklist.

o *mount_options*

Overrides options in the checklist file, or applies to the *dir* specified on the command line. Identical to checklist file *mount_options*. Several options are listed in Table 6.

Table 6 Common `mount` options

Option	Function
`rw`	Allow read/write access to file system. This is the default.
`ro`	Mount file system read-only.
`suid`	Set-uid execution allowed.This is the default.
`nosuid`	Set-uid execution not allowed.
`hard`	Retry request until server responds.
`soft`	Return error if server does not respond.

Table 6 Common `mount` options

Option	Function
`bg`	Retry mount in the background.

`mount` has more options than are presented here. For a complete summary, refer to the mount(1M) man page.

fsname

Specifies the name of the remote file system. If *fsname* is of the form, *host:path*, the file system type is assumed to be `nfs`.

dir

Represents the local mount point through which the client accesses the remote file system specified as *fsname*.

Step 3 Verify that you have successfully mounted a file system where you expected to, by entering either the `df` or `mount` commands without arguments. Both display information about currently mounted file systems, but in different formats.

Removing temporary files created by NFS

To create a temporary file in a local environment, many applications create or open a file and then immediately unlink (remove) the file from the directory in which it was created. The file handle returned by the kernel is how the application process continues with reads and writes, while the file-open connection state is maintained by the kernel. When the application closes the file, the operating system (kernel) detects that no more references are being made to this temporary file, and so it deletes the temporary file.

In the case of NFS, servers do not maintain connection state information. Hence, temporary files created by client calls to a server are renamed in the format `.nfs`*xxxx*, where *xxxx* is a number related to the time the file was renamed. It is the NFS client's responsibility to detect when a temporary file actually closes, that is, when the file's reference count goes to zero. With the reference count zero, the client calls the server to remove the temporary `.nfs`*xxxx* file. Figure 55 shows the format of `.nfs` file entries in a directory.

```
-rw-r--r--1        jones       1024       Apr 22 13:46        .nfs0DAA
-rw-r--r--1        jones       3072       May 11 12:52        .nfsDCAA
-rw-r--r--1        jones       1024       May 18 10:16        .nfsE637
-rw-r--r--1        jones       3072       Jun 11 12:12        .nfsECAA
-rw-r--r--1        jones       0          Jun 18 10:16        .nfsF637
```

Figure 55 NFS temporary files

NFS temporary files may never get deleted if an NFS client (NFS code in kernel on the client system) does not call the server to remove them. Conditions such as a workstation reboot or program error conditions can occur before a client has a chance to request the removal of its temporary files.

You can manually remove `.nfs`*xxxx* files, or you can set `cron` to remove them for you. For example, to have `cron` change the working directory to a specified project directory, and remove all the `.nfs`*xxxx* files at 7 a.m. every Sunday, create a `.crontab` entry that contains

```
0 7* *7 cd ~/project_dir ".nfs????" -atime +7 rm {}\;
```

Managing portmap

The portmapper (`portmap`) is a network service that maps port numbers to RPC program numbers. It provides a standard way for a client to look up the port number of RPC-based programs supported by a server. NFS and the SPP-UX tape system use `portmap`.

When an RPC server starts up, it tells `portmap` what port number it is listening to, and what RPC program numbers it is prepared to service. When a client wants to make an RPC call to a given program, it first contacts `portmap` on the server to determine which port receives RPC packets.

Any system that provides RPC services must have `portmap` running before starting any other RPC daemons. Because the SPP-UX tape system relies on `portmap`, you must ensure that `portmap` is running, even if your system does not run `NFS` or `NIS`.

This section explains how to start `portmap`, and obtain status information about `RPC` services.

Starting portmap

To ensure that `portmap` is started at boot time complete the following steps:

Step 1 Log in as superuser.

Step 2 Verify that the /etc/netnfsrc file installed with SPP-UX includes lines to start `portmap`. Figure 56 shows a partial netnfsrc startup file.

```
##
#   Start the portmapper -- should be started first
#   Note: portmap must be started before inetd!
#   Needs to be started on ARPA clients (only dependent upon inetd)
#   (See portmap(1M).)
##
if [ -f /etc/portmap ] ; then
   echo "\t/etc/portmap"
   /etc/portmap
   if [ $? -ne 0 ] ; then
        echo "Error:  NFS portmapper NOT powered up" >&2
        exit 1
   fi
fi

##
```

Figure 56 Starting portmap from netnfsrc

If the lines shown in Figure 56 are not in your netnfsrc file, use your editor to add them, then reboot the system.

Step 3 Verify that `portmap` is running after you reboot the system, by entering

```
ps -e | grep portmap
```

The system should respond with a line similar to

```
56 ? S 0:56 /etc/portmap
```

Obtaining status of portmap and RPC services

Use the `rpcinfo` utility to:

- Query a server about the programs registered with its `portmap`
- Verify that a remote machine is accepting and responding to RPC requests

You can use `rpcinfo` to get status information about the local host or a remote host by entering

`/usr/etc/rpcinfo -p` ***hostname***

Supply *hostname* for information about a remote host; omit it for information about the local host. `rpcinfo` displays information about all registered RPC services.

Information reported on each service includes:

- RPC number assigned to the service
- Version number of the service
- Protocol supported
- IP port used by the service
- RPC service name

Figure 57 shows sample `rpcinfo` output for the local host as requested by the `-p` option.

```
# /usr/etc/rpcinfo -p
   program   vers    proto     port
    100005      1      udp      665  mountd
    100005      1      tcp      667  mountd
    100003      2      udp     2049  nfs
    100017      1      tcp     1027  rexd
    100001      1      udp     1028  rstatd
    100001      2      udp     1028  rstatd
    100001      3      udp     1028  rstatd
    100002      1      udp     1029  rusersd
    100002      2      udp     1029  rusersd
    100012      1      udp     1030  sprayd
```

Figure 57 Checking RPC services with `rpcinfo`

When a remote `portmap` session has died or is not accepting connections, `rpcinfo` times out while attempting to reach it and reports an error.

Using automounter

9

The `automounter` utility enables users to mount and unmount remote directories on an as-needed basis. When a user on a client machine running `automounter` invokes a command that accesses a remote file or directory, such as opening a file with an editor, `automounter` mounts the hierarchy to which that file or directory belongs. `automounter` leaves the file or directory mounted as long as it is needed, but automatically unmounts it if it is not accessed for a certain amount of time. No mounting occurs at boot-time, and users need not know the superuser password to mount a directory. It is all done automatically and transparently.

Mounting some file hierarchies with `automount` does not exclude the possibility of mounting others with mount. For example, a diskless machine must mount /export/root, /export/swap, /usr and /export/exec/kvm through the `mount` command and /etc/fstab file.

This chapter discusses the following topics:

- Unmounting file systems
- Preparing `automounter`
- Invoking and tuning `automounter`
- Understanding error messages

Unmounting file systems

Unless suppressed, `automounter`'s start-up definition in /etc/netnfsrc directs `automounter` daemon to read the master map from the NIS auto.master map, as directed by the following shell script lines:

```
if [-f /usr/etc/automount]; then
   echo "    Starting up the Automount daemon" && \
   echo "\t/usr/etc/automount"
   /usr/etc/automount -f /etc/auto.master
   save=$?
   if [$save = 3]; then
      echo "\t No Maps were specified "
   else
      if [$returnstatus = 0]; then
          returnstatus=$save
      fi
   fi
 fi
```

To suppress `automount` from reading the NIS auto.master map and instead direct it to read an `automounter` auto.master file on a local host, the following script lines would replace the above script lines in /etc/netnfsrc:

```
automount -m -f /etc/auto.master
```

The `-m` option instructs `automounter` to not consult the master file distributed by NIS. If you do not run NIS, you do not have to specify the `-m` option. Automounter remains silent when it does not find a distributed master file.

Without NIS, a system manager must distribute copies of `automounter` maps and keep them consistent on every host. This is not much of an improvement over hard-coding /etc/fstab files. A network benefits by using NIS and `automounter` together.

A master map lists the mount points of `automounter`'s direct, indirect maps and usually a built-in map. The behavior of `automounter` is based on the type of map in which a remote file system's mount-point information resides—in a direct map, an indirect map, or a built-in map. The difference between a direct and indirect map is the key; for a direct map, the key is a full pathname; for an indirect map, it is a simple name. Built-in maps, explained further on, use existing files to give the appearance of being an extension of the local file system.

Referencing direct maps

`automounter` acts in part as a standalone NFS server, running as a utility independently of the operating system's NFS server. Having parsed its maps, and having created, on the local host, any directories needed as mount points, it does the equivalent of an NFS mount on the mount points specified by its maps.

When a user tries to access anything in the target filesystem, the operating system finds that the mount point is of filesystem type NFS, and that the mount point is a symlink. It attempts to resolve the symlink by sending an ordinary NFS READLINK RPC to the local `automounter`, which is acting as the NFS server for the mount. `automounter` then does an actual remote NFS mount of the target filesystem to the host entry in /tmp_dir. If it succeeds, it returns a string pointing to the new mount point in its response to the READLINK RPC. Thereafter, file system accesses across the mount point proceed as would any other NFS access.

Referencing indirect maps

Indirect maps are relative to some mount point, which is given either at the command line or in a master map. When a mount-point of the remote file system is in an indirect map, `automounter` emulates a directory of symbolic links.

In response to a `readlink`, it returns a path to the mount point in /tmp_mnt. A `readdir` of `automounter`'s mount point returns a list of the entries that are currently mounted.

Whether the map is direct or indirect, if the file system is already mounted and the symbolic link has been read recently, the cached symbolic link is returned immediately. Because `automounter` is on the same host, the response is much faster than a `READLINK` to a remote NFS server. On the other hand, if the file system is not mounted, a short delay occurs while the mounting takes place.

Referencing built-in maps

The `-hosts` map is a built-in map. This map uses the hosts database, which consists of the /etc/hosts file and the hosts.byname NIS map, for a list of every hostname on the network. How this map works is built into `automounter`. Its entry in the auto.master map would be

```
/net -hosts
```

which tells `automounter` when a user changes to a named host, to mount all accessible file systems exported from that host.

For example, this command line

```
cd /net/belmont
```

causes `automounter` to mount at /net all of system belmont's exportable file systems.

Preparing automounter maps

A server is indifferent as to whether its shared files are accessed through `mount` or `automount`. For servers, nothing needs to be done differently for `automount` than what is done for `mount`; however, `mount` is restricted in that it cannot handle replicated file systems.

A client, however, does need special files for `automounter`. As mentioned previously, `automount` does not consult /etc/fstab. Rather, when it starts up, it is passed the path name of a master map that lists where to find the file systems that can be accessed from the master map's mount point. All `automounter` maps are located in /etc, identified by the prefix, auto.

Locating maps through the master map

A master map, usually found in /etc/auto.master, tells `automounter` where to find the other maps. These other maps are either direct or indirect maps, and their location definition within auto.master are shown in looks like this:

```
# mount-point map mount-options
/net -hosts
/- /etc/auto.direct-soft, rw, intr, timeo=20
/home /etc/auto.hom-soft, ro, intr, timeo=20
```

Each definition contains three fields to describe the mounting of each map. These fields are

mount-point

Full pathname of a directory.

map

Name of the map that `automount` should use to find the mount points and locations.

mount-options

An optional, comma-separated list of options that regulate the mounting of the entries contained in the map, unless the map entries list other options. Options listed in the map entries override options specified in the mount-options list.

A line whose first character is # is treated as a comment. Note that /etc/auto.home is the name of an indirect map, /etc/auto.direct is the name of a direct map, and `-hosts` is a built-in map.

Referencing a built-in map: mount point /net

In this example, `automounter` mounts under the mount point, /net, all the entries under the special map `-hosts`. This is a

built-in map that does not use any external files except the hosts database, /etc/hosts. When a user changes to

/net/*hostname*

automounter will mount the exportable file systems for the host named in *hostname*.

Referencing an indirect map: mount point /home

The mount point /home is the directory under which the entries listed in /etc/auto.home (an indirect map) are to be mounted. Those entries are mounted under /tmp_mnt/home and a symbolic link is provided between /home/directory and /tmp_mnt/home/directory.

Referencing a direct map: mount point /-

The mount point /- is a filler that tells automounter to reference /etc/auto.direct, but not to associate the entries in /etc/auto.direct with any directory. Automounter then uses the mount points listed in the map; for a direct map, the key is a full path name.

Looking inside a direct map

Direct maps contain direct mount points for file systems with nonuniform naming schemes. The key for a direct map is a full path name of some directory. A direct map entry follows this syntax:

key mount-option [. . .] *location*

where each element provides the following information:

key

Path name of the mount point.

mount-option

One or more options that you want to apply to this particular mount.

location

Server:path name of the resource.

Writing an entry

Of all the maps, direct map entries most closely resemble their corresponding entries in /etc/fstab. Compare these two entries, and note their similarities.

- /etc/fstab entry

```
dancer:/usr/local /usr/local/tmp nfs ro 0 0
```

- direct map entry

```
/usr/local/tmp -ro dancer:/usr/local
```

The following lines show the setup of a direct map:

```
/usr/local\
                /bin     -ro,soft        ivy:/export/local/sun3   \
                /share   -ro,soft        ivy:/export/local/share  \
                /src     -ro,soft        ivy:/export/local/src
/usr/man                 -ro,soft        oak:/usr/man  \
                                         rose:/usr/man  \
                                         willow:/usr/man
/usr/games               -ro,soft        peach:/usr/games
/usr/spool/news          -ro,soft        pine:/usr/spool/news
/usr/frame               -ro,soft        redwood:/usr/frame1.3  \
                                         balsa:/export/frame
```

Describing multiple mounts in a map entry

A map entry can describe any number of mounts, where the mounts can be from different locations and with different mount options. Consider the first entry in the previous example. It is, in fact, one long entry whose readability has been improved by splitting it into three lines by using the backslash and indenting the continuation lines. This entry mounts /usr/local/bin, /usr/local/share, and /usr/local/src from the server ivy with the options read-only and soft. The following example shows the full path name of these mount points.

```
/usr/local/bin           -ro,soft        ivy:/export/local/sun3
/usr/local/share         -ro,soft        willow:/usr/local/share
/usr/local/src           -ro,soft        oak:/home/jones/src
```

The next example shows another version of how this entry could read, but the options are different and more than one server is used.

```
/usr/local\
                /bin             -ro,soft        ivy:/export/local/sun3
                /share           -ro,secure      willow:/usr/local/share \
                /src             -ro,intr        oak:/home/jones/src
```

Multiple mounts can be hierarchical. When file systems are mounted hierarchically, each file system is mounted on a subdirectory within another file system. When the root of the hierarchy is referenced, `automounter` mounts the whole hierarchy. The concept of root here is very important. The symbolic link returned by `automounter` to the kernel request is a path to the mount root. This is the root of the hierarchy that is

mounted under /tmp_mnt. This mount point should theoretically be specified:

```
parsley    /     -ro,intr veg:/usr/greens
```

In practice, it is not specified because in a case of a single mount as above, it is assumed that the location of the mount point is at the mount root or '.' Therefore, instead of the above, it is preferable, to enter

```
parsley          -ro,intr veg:/usr/greens
```

However, the mount point specification becomes important when mounting a hierarchy: here `automounter` must have a mount point for each mount within the hierarchy. The example above is a good illustration of multiple, non-hierarchical mounts under /usr/local when the latter is already mounted. The following example shows a true hierarchical mounting:

```
/usr/local\
                /              -rw, inrt              peach:/export/local\
                /bin           -ro,soft               ivy:/export/local/sun3\
                /share         -ro,secure             willow:/usr/local/share\
                /src           -ro,intr               oak:/home/jones/src
```

A true hierarchical mount can be problematic if the server for the root of the hierarchy goes down. Any attempt to unmount the lower branches fail, because the unmounting has to proceed through the mount root, which also cannot be unmounted while its server is down.

The mount points used here for the hierarchy are /, /bin, /share, and /src. Note that these mount-point paths are relative to the mount root, not the host's file system root. The first entry in the example above has / as its mount point. It is mounted at the mount root. There is no requirement that the first mount of a hierarchy be at the mount root. `automounter` issues `mkdir` to build a path to the first mount point if it is not at the mount root.

Describing multiple locations

In the following example for a direct map, the mount points /usr/man and /usr/frame list more than one location (three for the first, two for the second). This means that the mounting can be done from any of the replicated locations. This procedure makes sense only when you are mounting a hierarchy read-only, since you have some control over the locations of files you write or modify.

```
/usr/local      /bin           -ro,soft        ivy:/export/local/sun3
/usr/local      /share         -ro,soft        ivy:/export/local/share
```

```
/usr/local      /src            -ro,soft        ivy:/export/local/src
/usr/man                        -ro,soft        oak:/usr/man  \
                                                rose:/usr/man  \
                                                willow:/usr/man
/usr/games                      -ro,soft        peach:/usr/games
/usr/spool/news                 -ro,soft        pine:/usr/spool/news
/usr/frame                      -ro,soft        redwood:/usr/frame1.3  \
                                                balsa:/export/frame
```

A good example is man pages. In a large network, more than one server may export the current set of manual pages. It does not matter which server you mount them from, so long as the server is up and running and sharing its file systems. In the example above, multiple mount locations are expressed as a list of mount locations in the map entry

```
/usr/man  -ro,soft  oak:/usr/man  rose:/usr/man willow:/usr/man
```

This could also be expressed as a list of servers, separated by commas and followed by a colon and the pathname (as long as the pathname is the same for all the replicated servers)

```
/usr/man  -ro,soft  oak,rose,willow:/usr/man
```

Here you can mount the man pages from the servers oak, rose, or willow. From this list, `automounter` selects servers on the local network and verifies these servers. This launches a series of RPC requests to the null procedure of the mount service in each server. (Note that the list does not imply any ordering.) The first server to respond is selected, and an attempt is made to mount from it.

This redundancy, which is useful in an environment where individual servers may or may not be sharing their file systems, exists only at mount time. There is no status checking of the mounted-from server by `automounter` once the mount occurs. If the server goes down while the mount is in effect, the file system becomes unavailable. An option here is to wait five minutes until the automount takes place and try again. The next time, `automounter` will choose one of the available servers. Another option is to use the `umount` command, inform `automounter` of the change in the mount table, and retry the mount.

Looking inside an indirect map

Indirect maps show their association to regularly named file systems such as home, tools, and system utilities. Inconsistent naming within a directory can be written to appear consistent on a client machine. Additionally, this map allows you to edit the mount point for a single indirect map, rather than all the files within that map, you reorganize your mount tree.

The syntax for an indirect map (like that for a direct map) is

key [mount-option...] location

where *key* is the name of the directory to be used as the mount point and *location* is the server:path name pair that identifies the source of the file system. When `automounter` obtains the key, it appends the key to its associated mount point. If the mount point is not provided on the command line, `automounter` takes the mount point from the master map that invoked the indirect map.

Writing an entry

An entry in an indirect map reads as

```
parsley   -ro, intr      veggies:/usr/greens
```

The key here needs more information: where is the mount point parsley really located? That is why you must provide that information either at the command line or through another map. The solution here is: let us say that this entry above is part of a map called /etc/auto.veggies. The command line entry would be:

```
automount   /veggies   /etc/auto.veggies
```

or listed in the master map as:

```
/veggies  /etc/auto.veggies   -ro,soft,nosuid
```

In either case, you are associating a mount directory veggies with the entries (parsley in this case) mentioned in the indirect map /etc/auto.veggies. The end result is that the file system /usr/greens from the machine veggies is mounted on /veggies/parsley when needed.

Another sample entry in the master map reads as

```
/home/etc/auto.home rw,intr,secure
```

In the following example, /etc/auto.home is the name of the indirect map that contains the entries to be mounted under /home.

```
# key          [mount-option...]          location
willow                                    willow:/home/willow
cypress                                   cypress:/home/cypress
```

```
poplar                                          poplar:/home/poplar
pine                                            pine:/export/pine
apple                                           apple:/export/home
ivy                                             ivy:/home/ivy
peach           -rw,nosuid                      peach:/export/home
```

Assume that the sample map is on host oak. If user laura has an entry in the password database specifying her home directory as /home/willow/laura, then, whenever she logs into machine oak, `automounter` mounts (as /tmp_mnt/home/willow) the directory /home/willow residing in machine willow. If one of the subdirectories is indeed laura, user laura will be in her home directory, which is mounted read/write, interruptible, and secure.

Suppose, however, that user laura's home directory is specified as /home/peach/laura. When she logs into oak, `automounter` mounts the directory /export/home from peach under /tmp_mnt/home/peach. Her home directory will be mounted read/write, nosuid. Any option in the file entry overrides all options in the master map or the command line.

Now, assume the following conditions occur:

- User laura's home directory is listed in the password database as /home/willow/laura.
- Machine willow shares its home hierarchy with the machines mentioned in auto.home.
- All those machines have a copy of the same auto.home and the same password database.

Under these conditions, user laura can run `login` or `rlogin` on any of these machines and have her home directory mounted in place for her.

Furthermore, now laura can also enter the command

```
cd ~brent
```

and `automounter` will mount brent's home directory for her (if permissions allow it).

On a network without NIS, you must change all relevant databases (such as /etc/passwd) on all systems on the network in order to accomplish this. On a network running NIS, propagate all the relevant databases throughout the network to ensure this.

Adding a subdirectory field within an indirect map

In the indirect file auto.home access to a home directory in /home/willow requires that all the directories under it be mounted. Frequently used mount points such as home can become large and have multiple references to the same parent directory.

Because automounter treats each indirect map entry as a separate entity, it has to request multiple mounts from the same parent directory—one mount request for each of its subdirectories listed in the indirect map. Using a subdirectory field specifies only to mount the parent directory if needed. If it is already mounted, automounter establishes a link to the new subdirectory of the common parent.

When using subdirectory fields, an indirect map has a slightly different appearance

```
# key          [mount-option...]          location
john                                      willow:/home/willow:john
mary                                      willow:/home/willow:mary
joe                                       willow:/home/willow:joe
```

This example assumes that home directories are of the form /home/user rather than /home/server/user. In general, it is a good idea to provide a subdirectory entry in the location when multiple map entries refer to the same mounted file system from the same server. This approach means less mounts and better response time.

In response to this request

```
ls ~john ~mary
```

automounter performs the equivalent of the following actions:

```
mkdir /tmp_mnt/home/john
mountwillow:/home/willow/john
/tmp_mnt/home/john
ln -s /tmp_mnt/home/john /home/john
mkdir/tmp_mnt/home/mary
mount willow:/home/willow/mary
/tmp_mnt/home/mary
ln -s /tmp_mnt/home/mary/home/mary
```

The complete syntax of a line in a direct or indirect map is actually

key [*mount-option*] server:pathname[:subdirectory]

Now, when a user refers to john's home directory, automounter mounts willow:/home/willow. It then places a symbolic link between /tmp_mnt/home/willow/john and /home/john.

If the user then requests access to mary's home directory, automounter sees that willow:/home/willow is already mounted, so it only returns the link between /tmp_mnt/home/willow/mary and /home/mary. In other words, automounter now only does

```
mkdir  /tmp_mnt/home/john
mount willow:/home/willow  /tmp_mnt/home
```

```
ln -s  /tmp_mnt/home/john  /home/john
ln -s  /tmp_mnt/home/mary  /home/mary
```

Simplifying map entries

Two conventions reduce the size of map entries. One is the ampersand stating that the key should be substituted into the map. The following example is an indirect map with subdirectories specified:

```
# key          [mount-option...]          location
john                                      willow:/home/willow:john
mary                                      willow:/home/willow:mary
joe                                       willow:/home/willow:joe
able                                      pine:/export/home:able
baker                                     peach:/export/home:baker
```

The next example shows the same entries but substitutes the key (the ampersand) for the subdirectory field. Wherever the ampersand (&) appears, it translates into the key value.

```
# key          [mount-option...]          location
john                                      willow:/home/willow:&
mary                                      willow:/home/willow:&
joe                                       willow:/home/willow:&
able                                      pine:/export/home:&
baker                                     peach:/export/home:&
```

Another example of using the & is when the key and server have the same name. Inserting the ampersand says substitute the key value here. The following example shows how to use an ampersand:

```
# key          [mount-option...]          location
willow                                    &:/home/&
peach                                     &:/home/&
pine                                      &:/home/&
oak                                       &:/home/&
poplar                                    &:/home/&
```

You could also use key substitutions in a direct map, in situations like

```
/usr/man willow,cedar,poplar:/usr/man
```

which is a good candidate to be written as

```
/usr/man willow,cedar,poplar:&
```

Notice that the ampersand substitution uses the whole key string, so if the key in a direct map starts with a / (as it should), that slash is carried over, and you could not do something like

```
/progs &1,&2,&3:/export/src/progs
```

because automounter would interpret it as

```
/progs /progs1,/progs2,/progs3:/export/src/progs
```

The second convention is the asterisk—the wildcard—that matches all keys and provides a value for the & specified in the remainder of the map information. It should be the last or only entry in a map because automounter does not read any further once it reads a wild card entry.

All the entries in have the same format, a good case for the use of the wildcard. The asterisk reduces one line entry to

```
*&:/home/&
```

where each ampersand is replaced by the value of any given key.

Once automounter reads the wildcard, it does not continue reading the map, so the map is viable, as shown in the following example:

```
# key           [mount-option...]          location
oak                                        &:/export/&
poplar                                     &:/export/&
*                                          &:/home/&
```

Whereas, in the map in the next example, the last two entries are ignored.

```
# key           [mount-option...]          location
*                                          &:/home/&
oak                                        &:/export/&
poplar                                     &:/export/&
```

Using the -hosts built-in map

The -hosts map provides a convenient way for users to access directories in many difference hosts without having to use rlogin or rsh. Both of these remote commands have to establish communication through the network every time they are invoked.

automounter has built into it the semantics for the -hosts map. This map only references the hosts databases and the mount is done on /net, indicating that it contains exported filesystems from all over the network. Its entry in a master map would read

```
/net -hosts
```

Calling it from the command line would read

```
automount /net -hosts
```

automounter can also do a mount of all exported filesystems from a named server (host), in response to an entry

/net *servername*

giving users a way to force mounts of all file systems by giving the server's name as a subdirectory of /net

Modifying NIS managed maps

You can modify automounter maps at any time, but these changes only appear after automounter is restarted. If you want a map modification to take effect immediately, refer to the section titled "Invoking and tuning automounter." automounter has each direct mount point as an entry in a mount table, so changes to this table cannot be done without automounter. Attempts by NIS to reference a mount point removed from this table returns an error message, even though the mount point is still listed in the mtab file.

Changes to an indirect map must be done at a master server so the map is rebuilt and pushed to the slave servers. Hence, the next time automounter reads the map for a mount request—and automounter has to read indirect maps for each mount request it handles—it will read an updated map.

To change parameters for a currently mounted file system, you must manually unmount it in /tmp_mnt and send automounter a SIGHUP (kill -1). When automounter receives this signal, it parses the mtab file and identifies mounted file systems that were unmounted by someone else.

Invoking and tuning automounter

The automount(8) man page contains a complete description of all options. The suboptions are the same as those for a standard NFS mount, except for `bg` (background) and `fg` (foreground), which do not apply.

It is preferable to invoke `automounter` from /etc/netnfsrc, but it can be invoked from the command line. The syntax to invoke `automounter` is:

```
automount [-m -n -T -v] [-Dname=vvalue]\
[-fmaster-file][-Mmount-directory] \
[-tsub-options][directory map][-mount-options]
```

Sample automounter map files

If you define your auto.master, auto.home, and auto.direct maps like the following samples, `automounter` can be invoked from the command line—or preferably from /etc/netnfsrc.

- Sample auto.master map:

```
#Mount-point        Map              Mount options
/net                -hosts
/home               /etc/auto.home   -rw,intr,secure
/-                  /etc/auto.home   -ro,intr
```

- Sample auto.home map:

```
# key               [mount-option...]      location
willow                                     willow:/home/willow
cypress                                    cypress:/home/cypress
poplar                                     poplar:/home/poplar
pine                                       pine:/export/pine
apple                                      apple:/export/home
ivy                                        ivy:/home/ivy
peach               -rw,nosuid             peach:/export/home
```

- Sample auto.direct map:

```
# key               [mount-option...]      location
/usr/local/bin      -ro,soft               ivy:/export/local/sun3
/usr/local/share    -ro,soft               ivy:/export/local/share
/usr/local/src      -ro,soft               ivy:/export/local/src
/usr/man            -ro,soft               oak:/usr/man
/usr/man            -ro,soft               rose:/usr/man
/usr/man            -ro,soft               willow:/usr/man
/usr/games          -ro,soft               peach:/usr/games
/usr/spool/news     -ro,soft               pine:/usr/spool/news
/usr/frame          -ro,soft               redwood:/usr/frame3.0
/usr/frame          -ro,soft               balsa:/export/frame
```

Invoking automount

You can log in as superuser and type any of the following commands at shell level to start `automounter`. Ideally, edit /etc/netnfsrc and include your preferred command there.

- Specify all arguments to `automounter` without reference t the master map, as in

```
automount /net -hosts /home /etc/auto.home -rw,intr,secure /-
/etc/auto.direct -ro,intr
```

- Specify the same in the *auto.master* file and instruct `automounter` to look there for instructions
  `automount -f /etc/auto.master`
- Specify mount points and maps in addition to those mentioned in the master map as follows

```
automount -f /etc/auto.master /src /etc/auto.src -ro,soft
```

- Nullify one of the entries in the master map. (This is particularly useful if you use a map you cannot modify tha does not meet the needs of your machine.)

```
automount -f /usr/lib/auto.master /home -null
```

- Replace one of the entries with your own

```
automount -f /usr/lib/auto.master /home /myown/auto.home -rw,intr
```

In the example above, `automounter` first mounts all items i the map /myown/auto.home under the directory /home. Then, when it consults the master file /usr/lib/auto.maste it ignores the line corresponding to /home, because it has already mounted on /home.

Given the auto.master file of the previous example, the first tw commands are equivalent as long as your network does not hav a distributed auto.master file. This file is available only on networks running NIS. If your network includes a distributed auto.master file, the second example must be modified in the following way to be equivalent to the first example:

```
automount -m -f /etc/auto.master
```

The `-m` option instructs `automounter` not to consult the maste file distributed by the NIS. If you do not run NIS, you do not hav to specify the `-m` option. `automounter` is completely silent whe it does not find a distributed master file.

Using environment variables

You can use the value of an environment variable by prefixing a dollar sign ($) to its name. You can also use braces to delimit the name of the variable from appended letters or digits.

Environmental variables can be inherited from the environment or can be defined explicitly with the command line option -D. For example, if you want each client to mount client-specific files in the network in a replicated format, create a specific map for each client according to its name. The relevant line for host oak would be

```
/mystuff cypress,ivy,balsa:/export/hostfiles/oak
```

and in willow it would be

```
/mystuff cypress,ivy,balsa:/export/hostfiles/willow
```

This scheme is viable within a small network, but maintaining this kind of host-specific map across a large network soon becomes unmanageable. The solution for large networks is to invoke `automounter` with a command line similar to

```
automount -D HOST='hostname' ......
```

and have the entry in the direct map read

```
/mystuff cypress,ivy,balsa:/export/hostfiles/$HOST
```

Now each host finds its own files in the mystuff directory, and the task of centrally administering and distributing the maps becomes easier.

After you complete the maps, make sure that there are no equivalent entries in /etc/fstab and that all entries in the maps refer to NFS shared files.

Logging automount activities

The trace option `-T` tells automount to log its actions to the standard error stream. This option enables you to see the expansion of NFS calls handled by `automounter`. To use this option as a debugging tool, as part of the `automounter` start-up command in /etc/netnfsrc, redirect the output of standard output and standard error to a file. Something like this:

```
automounter -T /-auto.direct >& /tmp/log.nfscalls
```

Naming another directory as mount point

The default name for all mounts is /tmp_mnt. As with other names, this one is arbitrary. It can be changed at invocation by using the `-M` *tmpdir* option. For example:

```
automount -M /auto ...
```

causes all mounts to occur under the directory /auto, which `automounter` creates if it does not exist. Do not designate a directory in a read-only file system, as this would prohibit `automounter` from modifying anything.

A working example

Here is an example that outlines what `automounter` does when passing a command. With the `automount` daemon in place, issuing

```
cd /net/gumbo
```

signals `automounter` to start executing these steps:

1. Changes directory to the top of the hierarchy of files (that is, the root file system) of the machine gumbo, provided the machine is in the hosts database.

   Not all files and directories may display under /net/gumbo, because `automounter` can only mount the shared file systems of host gumbo in accordance with the restrictions placed on the sharing.
2. Verifies the null procedure of the server's mount service to see if it is active.
3. Requests the list of shared hierarchies from the server
4. Sorts the shared list according to length of path name (to ensure proper mounting order):

   ```
   /usr/src
   /export/home
   /usr/src/sccs
   /export/root/blah
   ```
5. Proceeds down the list, mounting all the file systems at mount points in /tmp_mnt (creating the mount points as needed).
6. Returns a symbolic link that points to the top of the recently mounted hierarchy.

For the -hosts map, automounter must mount all the file systems that the server in question advertises for sharing. Even if the request is as follows:

```
ls /net/gumbo/usr/include
```

automounter mounts all gumbo shared systems, not just /usr.

In addition, unmounting occurs after a certain amount of time has passed, working from the bottom up. This means if one of the directories at the top is busy, automounter must remount the hierarchy and try again later.

Nevertheless, the -hosts special map provides a convenient way for users to access directories in many different hosts without having to use rlogin or rsh. (These remote commands have to establish communication through the network every time they are invoked.)

Notice that both /net and /home are arbitrary names dictated by convention. automounter creates them if they do not already exist.

Mount table

Every time automounter mounts or unmounts a file hierarchy, it modifies /etc/mtab to reflect the current situation. automounter keeps an image of /etc/mtab in memory and refreshes this image every time it performs a mount or an automatic unmount. If you use umount to unmount one of the automounted hierarchies (a directory under /tmp_mnt), force automounter to reread the /etc/mtab file. To do so, enter

```
ps -e | grep automount | egrep -v grep
```

The second field gives you the process ID of automounter. automounter is designed to reread /etc/mtab when it receives a SIGHUP signal. To send it that signal, enter

```
# kill -1 PID
```

where *PID* stands for the process ID you obtained from the ps command.

Error messages

Table 7 lists error messages displayed if `automounter` fails and explains what each message indicates.

There are conditions and actions that need to be defined.

Table 7 Error messages related to `automount`

Error message	Condition cause with recommended action
No mount maps specified	`automounter` was invoked with no maps to serve, and it cannot find the NIS auto.master map. This message is produced only when the `-v` option is given. Recheck the command, or restart NIS if that was the intention.
Mapname: Not found	The required map cannot be located. This message is produced only when the `-v` option is given. Check the spelling and pathname of the map name.
Dir mountpoint must start with '/'	`automounter` mount point must be given as full pathname. Check the spelling and pathname of the mount point.
Mountpoint: Not a directory	The mount point exists but it is not a directory. Check the spelling and pathname of the mount point.
Hierarchical mountpoint: mountpoint	`automounter` does not allow itself to be mounted within an automounted directory. You will have to use another strategy.
WARNING: mountpoint not empty!	The mount point is not an empty directory. This message is produced only when the `-v` option is given and it is only a warning. It means that the previous contents of the mount point will not be accessible.
Can't mount mountpoint: reason	`automounter` cannot mount itself at mount point. The reason given with the message should be self-explanatory.
Hostname:file system already mounted on mountpoint	`automounter` has been mounted on an already mounted-on mount point and is attempting to mount the same file system there. This will happen if an entry in /etc/fstab also appears in an `automounter` map (either by accident or because the output of mount `-p` was redirected to fstab). Delete one of the redundant entries.
WARNING: hostname:file system already mounted on mountpoint	`automounter` is mounting itself on top of an existing mount point.
Couldn't create directory: reason	Could not create a directory. The reason should be self-explanatory.
Bad entry in map mapname "map entry"	

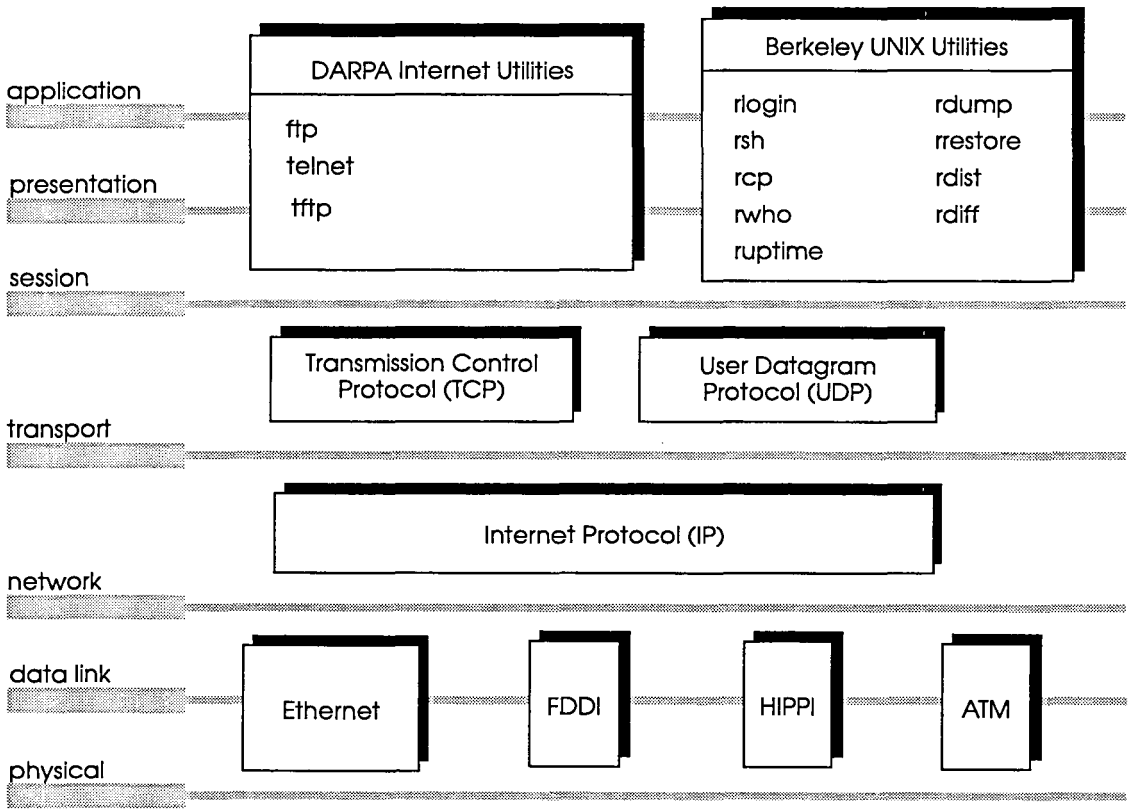
Table 7 Error messages related to `automount -(continued)`

Error message	Condition cause with recommended action
Map mapname, key map key: bad	The map entry is malformed, and `automounter` cannot interpret it. Recheck the entry; perhaps there are characters in it that need escaping.
Mapname: yp_err	Error looking up an entry in a NIS map.
Hostname: exports: rpc_err	Error getting share list from hostname. This indicates a server or network problem.
Host hostname not responding	
Hostname:filesystem server not responding	
Mount of hostname:filesystem on mountpoint: reason	You will see these error messages after `automounter` attempted to mount from hostname but either got no response or failed. This may indicate a server or network problem.
Mountpoint - pathname from hostname: absolute symbolic link	When mounting a hierarchy, `automounter` has detected that mountpoint is an absolute symbolic link (begins with "/"). The content of the link is pathname. This may have undesired consequences on the client.
Cannot create socket for broadcast rpc: rpc_err	
Many_cast select problem: rpc_err	
Cannot send broadcast packet: rpc_err	
Cannot receive reply to many_cast: rpc_err	All these error messages indicate problems attempting to ping servers for a replicated file system. This may indicate a network problem.
Try many: servers not responding: reason	No server in a replicated list is responding. This may indicate a network problem.
Remount hostname:filesystem on mountpoint: server not responding	Attempted remount after unmount failed. Indicates a server problem.

Table 7 Error messages related to `automount` **`-(continued)`**

Error message	Condition cause with recommended action
NFS server (pidn@mountpoint) not responding still trying	An NFS request made to the `automount` daemon with PID *n* serving mountpoint has timed out. `automounter` may be temporarily overloaded or dead. Wait a few minutes; if the condition persists, the easiest solution (for a diskless client) is to reboot the client. If this is not possible, exit all processes that use automounted directories (or, change to a non-automounted directory in the case of a shell), kill the current automount process and restart it again from the command line. If this does not work, you must reboot.
Path: <*pathname*> does not exist on host: <*hostname*>	Where <pathname> is the pathname that was mounted from the remote system (including subdirectories from indirect maps using ':') and <hostname> is the name of the NFS server. To correct the problem, create the specified directory on the NFS server system.
Domain name not set. Cannot use NIS.	
NIS bind failed: can't communicate with ypbind.	
Mount: serverhost:/exported/fs server no responding: RPC: Authentication error; why = Invalid client credential mount: giving up on: /exports/fs	

Network information system



Network Information System (NIS) 10

This chapter explains procedures for setting up and troubleshooting the Network Information System (NIS). These procedures include:

- Setting up NIS servers and clients
- Accessing, modifying, and propagating NIS maps
- Making new NIS maps after installing NIS
- Adding an additional NIS server
- Changing the master server to a different host
- Disabling NIS
- Troubleshooting servers, clients, and daemons
- Managing NIS security

Introduction to NIS

The Network Information System (NIS), formerly known as Yellow Pages (YP), is a distributed lookup service that maintains a centralized set of configuration files that would otherwise be maintained as ASCII files on each local system. Networked hosts query a central server for this information as they would a database.

NIS greatly simplifies management of such configuration files as /etc/hosts, /etc/passwd, /etc/ethers, and /etc/group because it creates a single point of administration and gives all hosts access to a recent version of the data. NIS clients and servers retrieve needed information from maps—database tables derived from these centralized configuration files.

Terms

The following terms are used in this chapter:

NIS client

Requests data from maps on NIS servers.

NIS map

Contains Non-ASCII administrative files, usually derived from ASCII files found in the /etc directory. Each NIS map has a map name used by programs to access it.

NIS server

Fulfills requests made by NIS clients. Also called a *service provider*.

master server

Contains the master set of NIS maps. When the master server's maps are updated, it distributes the updates to each NIS slave server.

slave server

Contains complete copies of the master server's set of NIS maps.

Before you begin

NIS uses network services provided by Internet utilities. Therefore, before beginning any task described in this chapter, you must successfully complete the following tasks:

- Install SPP-UX and Utilities, which includes:
  - Internet utilities
  - NFS utilities
- Configure Internet utilities

For information about installing the products listed above, refer to the installation procedures that accompanied your software distribution tape.

Summary of steps

The following procedures are summaries of the steps required for you to set up and manage NIS.

Preparing to set up a master NIS server

Step 1 Log in as superuser.

Step 2 Select an NIS domain name. Make sure your /etc/netnfsrc script contains a `domainname` command.

Step 3 Verify that your host name is defined in /etc/netnfsrc.

Step 4 Ensure the following files, located in /etc, are complete and up to date:

- passwd
- hosts
- ethers
- group
- netgroup
- networks
- protocols
- rpc
- services

Step 5 Set up /etc/netgroup.

Step 6 Edit /usr/lib/aliases, if needed.

Step 7 If you wish to restrict access to the master NIS server, perform the following procedure:

- Make a copy of the complete /etc/passwd file.
- Edit out undesired users from the remaining /etc/passwd using `vipw`.
- Edit out the NIS escape entry.

Setting up a master NIS server

Step 1 Log in as superuser and change directory to /usr/etc/yp.

Step 2 Run `ypinit`.

Step 3 Run `ypserv` and `ypbind`.

Step 4 Check /etc/netnfsrc for the following entry:

```
NIS_MASTER_SERVER=1
```

Setting up a slave NIS server

Step 1 Become superuser and change directory to /usr/etc/yp.

Step 2 Select a host already set up as a master NIS server.

Step 3 Check that an entry for `daemon` exists in the /etc/passwd files of both slave and master. The `daemon` entry must precede any other entries that have the same user ID.

Step 4 Set the default domain name on the host intended to be the slave NIS server to be the same as the default domain name on the host named as the master.

Step 5 Run `ypinit` with the `-s` option.

Step 6 Make copies of the following files:

- /etc/passwd
- /etc/hosts
- /etc/group
- /etc/networks
- /etc/protocols
- /etc/netgroup
- /etc/services

Step 7 Edit the original files according to the instructions in the section "Altering NIS client files to use NIS services."

Step 8 Make backup copies of the edited files.

Step 9 Run `ypbind` from the slave, to give access to domain maps.

Step 10 Run `ypserv`.

When the system is booted, `ypserv` starts automatically from /etc/netnfsrc.

Setting up an NIS client

Step 1 Edit the following local files to use NIS services according to the instructions in the section "Altering NIS client files to use NIS services:"

- /etc/passwd
- /etc/hosts
- /etc/group
- /etc/networks
- /etc/protocols
- /etc/services
- /etc/netgroup
- /etc/rpc

- /.rhosts

Step 2 Start `ypbind`, if it is not already running.

Step 3 Verify that /etc/netnfsrc contains the proper line to set the domain name.

Modifying existing maps

Step 1 Become superuser on the master NIS server.

Step 2 Edit the ASCII database file.

Step 3 Change directory to /usr/etc/yp.

Step 4 Run `ypmake` (`makedbm`).

Step 5 Modify nonstandard maps manually.

Adding a new NIS slave server

Step 1 Become the superuser on the master NIS server.

Step 2 Add the host's name to the ypservers map in the default domain.

Step 3 Change directory to /usr/etc/yp and run `ypmake` (`makedbm`).

Step 4 Login to the new NIS slave server and change directory to /usr/etc/yp.

Step 5 Enter **`ypinit -s ypmaster`**.

Step 6 Make copies of the following files:

- /etc/passwd
- /etc/hosts
- /etc/group
- /etc/networks
- /etc/protocols
- /etc/netgroup
- /etc/services

Step 7 Edit the original files according to the instructions in the section "Altering NIS client files to use NIS services."

Changing a map's master server

Step 1 Become superuser on the new master NIS server.

Step 2 Build the map at the new master.

Step 3 Change directory to /usr/etc/yp and edit the Makefile to add an entry for the new map.

Step 4 Remake the NIS map locally with `ypmake`.

Step 5 Edit the old master's /usr/etc/yp/Makefile—if it is to remain an NIS server—to comment out the section that made the map.

Step 6 Remake maps which only exist as dbm databases on the new master by disassembling an existing copy from any NIS server, then running the disassembled version back through `makedbm`.

Step 7 Send a copy of the new master's map to the other (slave) NIS servers with `ypxfr`.

Step 8 Run `yppush`.

Where to find more information

This chapter provides instructions on the tasks that you perform to set up and maintain NIS on an Exemplar system. If you are responsible for managing NIS, you should also become familiar with the following books:

- *Managing NFS and NIS*, by Hal Stern
  (O'Reilly & Associates, 1992. ISBN 0-937175-75-7)
- Hewlett-Packard guides:
  - *Installing and Administering NFS Services* (DHP-173)
  - *Programming and Protocols for NFS Services* (DHP-174)
  - *Using NFS Services* (DHP-172)

Setting up an NIS server

This section explains how to:

- Prepare a host to become a master NIS server
- Set up a master NIS server
- Set up a slave NIS server

Preparing to set up a master NIS server

Complete the following procedure before you begin setting up a master NIS server:

Step 1 Login as superuser.

Step 2 Select an NIS domain name, if it differs from the name selected for your network domain during installation. Make sure your /etc/netnfsrc script contains a `domainname` command that defines your NIS domain. The domain name is defined in /etc/netnfsrc as follows:

`/bin/domainname` *your-domain-name*

Step 3 Verify that your host name is defined in /etc/netnfsrc. If you have configured Internet utilities, you have already defined the host name. The host name is defined in /etc/netnfsrc as follows:

`/bin/hostname` *local-host-name*

Step 4 Ensure that the following files in /etc are complete and up to date:

- passwd
- hosts
- ethers
- group
- netgroup
- networks
- protocols
- rpc
- services

Step 5 If you know how /etc/netgroup is going to be set up, set it up now. If you do not set up /etc/netgroup at this time, `ypinit` makes an empty netgroup map.

Step 6 Edit /usr/lib/aliases if it needs to be updated.

Step 7 If you wish to restrict access to the master NIS server, perform the following steps:

- Make a copy of the complete /etc/passwd file.
- Edit out undesired users from the remaining /etc/passwd using `vipw`.
- Edit out the NIS escape entry.

You are ready to create a new master server.

Setting up a master NIS server

Step 1 Log in as superuser and change directory to /usr/etc/yp.

Step 2 Run `ypinit`.

`ypinit` prompts you for the following information:

- A list of other hosts that will also be NIS servers.

  Initially, this is the set of NIS slave servers, but in the future, any of them might become the NIS master server. You need not add any other hosts at this time, but if you know that you will be setting up more NIS servers, add them now. You will save yourself some work later, and there is little runtime penalty for doing it.
- Whether you want to exit at the first nonfatal error or to continue despite nonfatal errors.

  If `ypinit` exits, you can fix the problem and restart `ypinit`. Use this option if you have never done this procedure. You may elect to continue despite nonfatal errors. In this case, you can try to fix all the problems by hand, or fix some, then restart `ypinit`.

The following sample shows a run of `ypinit` on a master server.

```
host1# cd /usr/etc/yp
host1# ypinit
usage:
     ypinit -m [DOM=domain_name]
     ypinit -s master_server [DOM=domain_name]
     -m: make this machine a master NIS server
     -s: make this machine a slave NIS server
"master_server" is the existing master NIS server from which the databases
are copied.
"domain_name" is the NIS domain name to initialize; if not supplied, DOM
defaults to the name returned by domainname(1).
host1# ypinit -m
The host1's domain name hasn't been set.
Please set it using domainname.
host1# domainname host1
```

```
host1# ypinit -m
Installing the NIS data base will require that you answer a few questions.
Questions will all be  asked at the beginning of the procedure.
Do you want this procedure to quit on non-fatal errors? [y/n: n]  y
Can we destroy the existing /usr/etc/yp/host1 and its contents? [y/n: n]  y
At this point, we have to construct a list of the hosts which will run NIS
servers.  host1 is in the list of NIS server hosts.  Please continue to add
the names for the other hosts, one per line.  When you are done with the
list, type a <ctl D>.
     next host to add:  host2
     next host to add:  CONTROL-D
The current list of NIS servers looks like this:
host1
host2
Is this correct?  [y/n: y]  y
There will be no further questions. The remainder of the procedure should take 5
 to 10 minutes.
Building /etc/yp/host1/ypservers...
Running /etc/yp/Makefile...
updated ypservers
updated passwd
updated group
updated hosts
updated netgroup
updated networks
updated rpc
updated services
updated protocols
updated pwrestrict
updated newsgroup
updated netgroup
update bootparams
10 aliases, longest 51bytes, 310 bytes total
updated aliases
updated publickey
updated netid
host 1 has been set up as a NIS master server without any errors.
If there are runing slave NIS servers, run yppush now for any databases
that have been changed. If there are no running slaves, run ypinit
on hosts that are to be slave servers.
```

Step 3 Run ypserv and ypbind.

Step 4 Check the NIS_MASTER_SERVER entry in /etc/netnfsrc is set to 1

The NIS_MASTER_SERVER should be set to 1, as shown in the following example:

```
##
#       NFS_CLIENT      --      1 if this node is an NFS client, 0 if not
#       NFS_SERVER      --      1 if this node is an NFS server, 0 if not
#       Note:   it is possible for one host to be a client, a server, both
```

```
#               or neither!  This system is an NFS client if you will be
#               NFS mounting remote file systems; this system is a server
#               if you will be exporting file systems to remote hosts.
#       See Also:  nfsd(1M), mount(1M).
##
NFS_CLIENT=1
NFS_SERVER=1
```

Setting up a slave NIS server

The network must be operational before you can set up a slave NIS server. In particular, you must be able to copy files from the master NIS server to NIS slaves. To create a new slave server, complete the following steps:

Step 1 Become superuser and change directory to /usr/etc/yp.

Step 2 Select a host already set up as a master NIS server.

Ideally, the host you select is really the master server, but it can be any host that has its NIS database set up and is reachable over the network. To determine whether the host is reachable, enter

/usr/etc/rpcinfo -b ypserv 2 | grep *hostname*

The server in question is reachable if it appears in the output list as *hostname*.

Step 3 Check that an entry for `daemon` exists in the /etc/passwd files of both slave and master. The `daemon` entry must precede any other entries that have the same user ID.

Step 4 Set the default domain name on the host intended to be the slave NIS server to be the same as the default domain name on the host named as the master.

Step 5 Run `ypinit` with the `-s` option.

The following sample shows a run of `ypinit` on a slave server:

```
goofy# cd /usr/etc/yp
goofy# ypinit -s mickey
Installing the yp data base will require that you answer a few questions.
Questions will all be asked at the beginning of the procedure.
Do you want this procedure to quit on non-fatal errors? [y/n: n]  y
There will be no further questions. The remainder of the procedure should take a
few minutes, to copy the data bases from mickey.
Transferring group.bygid...
Transferring group.byname...
Transferring hosts.byaddr...
Transferring hosts.byname...
Transferring mail.aliases...
Transferring netgroup...
Transferring netgroup.byuser...
```

```
Transferring netgroup.byhost...
Transferring networks.byaddr...
Transferring networks.byname...
Transferring passwd.byname...
Transferring passwd.byuid...
Transferring protocols.byname...
Transferring protocols.bynumber...
Transferring services.byname...
Transferring ypservers...
Transferring services...
Transferring rpc.byname...
Transferring rpc.bynumber...
Transferring pwrestrict.byname...
Transferring pwrestrict.byuid...
Transferring bootparams...
Transferring publickey.byname...
Transferring netid.byname
```

At this point, make sure that /etc/passwd, /etc/hosts, /etc/networks, /etc/group, /etc/protocols, /etc/services, /etc/rpc, and /etc/netgroup have been edited so when NIS is activated, the data bases you have just created will be used, instead of the /etc ASCII files.

As stated in the section "Setting up a master NIS server," you are not prompted for a list of other servers, but you do have the opportunity to choose whether or not the procedure gives up at the first nonfatal error.

Step 6 Make copies of the following files:

- /etc/passwd
- /etc/hosts
- /etc/group
- /etc/networks
- /etc/protocols
- /etc/netgroup
- /etc/services

For example, to make a copy of /etc/passwd, enter

```
cp /etc/passwd /etc/passwd-
```

Step 7 Edit the original files according to the instructions in the section "Altering NIS client files to use NIS services," to ensure that processes on the slave NIS server actually use the NIS services, rather than local ASCII files. (That is, make sure the NIS slave server is also a NIS client.)

Step 8 Make backup copies of the edited files. For example,

```
cp /etc/passwd /etc/passwd+
```

Step 9 Run `ypbind` from the slave to give the slave access to domain maps, by entering

```
/etc/ypbind
```

Step 10 After the NIS database is set up by `ypinit`, to begin providing NIS services, enter

```
/usr/etc/ypserv
```

When the system is booted, `ypserv` starts automatically from /etc/netnfsrc.

Setting up NIS clients

This section explains how to:

- Set up NIS clients
- Alter an NIS client's database to use NIS services

Setting up an NIS client

To set up an NIS client, perform these steps:

Step 1 Edit the following local files to use NIS services as described in the section "Altering NIS client files to use NIS services:"

- /etc/passwd
- /etc/hosts
- /etc/ethers
- /etc/group
- /etc/networks
- /etc/protocols
- /etc/services
- /etc/netgroup
- /etc/rpc
- /.rhosts

Step 2 Start `ypbind`, if it is not already running.

Step 3 Verify that /etc/netnfsrc contains the proper line to set the domainname. This line is of the format

`/bin/domainname` *your-domain-name*

With the ASCII databases of /etc abbreviated and `ypbind` running, processes on the host will be clients of the NIS services.

At this point, there must be an NIS server available. Processes will hang if no NIS server is available while `ypbind` is running.

Note the possible alterations to the client's /etc database as discussed in the section "Altering NIS client files to use NIS services." Because some files may not be there, or some may be specially altered, it is not always obvious how the ASCII databases are being used. The escape conventions used within those files to force data to be included or excluded from the NIS databases are found in the following man pages:

- passwd(4)
- hosts(4)
- netgroup(4)
- hosts.equiv(4)
- group(4)

In particular, notice that changing passwords in /etc/passwd by editing the file, or by running the `passwd` command affects only the local client's environment. Change the client's NIS password database by running `yppasswd`.

Altering NIS client files to use NIS services

Once you decide to run NIS at your site, you should have all hosts on the network access the NIS maps, instead of potentially out-of-date information in their local ASCII configuration files. To enforce the use of NIS maps, run a `ypbind` process on each client, including clients that may also be NIS servers, and abbreviate or eliminate the files traditionally implementing NIS maps, namely:

- /etc/passwd
- /etc/hosts
- /etc/ethers
- /etc/group
- /etc/networks
- /etc/protocols
- /etc/services
- /etc/netgroup
- /etc/rpc
- /.rhosts

The treatment of each file is discussed in this section.

- /etc/networks, /etc/protocols, /etc/ethers, /etc/rpc, /etc/services, and /etc/netgroup need not exist on any NIS clients. These files are never consulted.
- /etc/hosts.equiv is never served by NIS. However, you can add escape sequences to reference NIS. This reduces problems with `rlogin` that are sometimes caused by different /etc/hosts.equiv files on the two hosts.

  To let anyone log in to a host, edit /etc/hosts.equiv to contain a single line, with only the character, + (plus) on it. A line containing only a + means that all further entries are retrieved from NIS rather than the local file.

  Alternatively, you can exercise more control over logins by using lines of the form:

  +@trusted_group1

  +@trusted_group2

  -@distrusted_group

where the name to the right of an at sign (@) is interpreted as a netgroup name defined in the global netgroup database. The netgroup database is served by NIS.

If neither the + or - is used, only entries in /etc/hosts.equiv are used; NIS is not used.

- /.rhosts is never served by NIS. Its format is identical to that of /etc/hosts.equiv; however, because this file controls remote superuser access to the local host, allowing unrestricted access to it is not recommended. Make the list of trusted hosts explicit, or use netgroup names for the same purpose. You can not use secondary hostnames in your .rhosts, hosts.equiv, or netgroup files. You can, however, use secondary hostnames in /etc/hosts. All of the above files enable local hosts to access remote hosts in some fashion.
- /etc/hosts must contain entries for the local host's name, and the local loopback name. These are accessed at boot time when the NIS service is not yet available. After the system is running, and after the `ypbind` process is up, the /etc/hosts file is not accessed at all.
- /etc/passwd should contain entries for the user name, `root`, and the primary users of the host, and the + escape entry to force the use of the NIS service. A few additional entries are recommended: `daemon`, to allow file-transfer utilities to work; and `operator`, to let a dump operator log in. An NIS client's /etc/passwd file might look like.

```
root:9wxntql2tHT.k:0:1:Operator:/:/bin/csh
nobody:*:-2:-2::/:
daemon:*:1:1::/:
sys:*:2:2::/:/bin/csh
bin:*:3:3::/bin:
uucp:*:4:4::/var/spool/uucppublic:
news:*:6:6::/var/spool/news:/bin/csh
sync::1:1::/:/bin/sync
raks:7kjDXZD/Hug2s:624:20:Stefania:/home\
/dancer/raks:/bin/csh
+::0:0:::
```

The last line informs the library routines to use the NIS service. If you remove the last line in the passwd file, you will disable NIS password access.

A program that uses `getpwent` (refer to the getpwnet(3) man page) to access /etc/passwd first looks in the password file

on your host; it then looks in the NIS password file (only if your host's password file contains + (plus sign) entries), as shown in the above example.

Earlier entries in the file take precedence over, or mask, later ones with the same user name, or the same user ID. Therefore, please note the order of the entries for `daemon` and for `sync` (which have the same user ID) and duplicate this order in your own file.

- /etc/group may be reduced to a single line

  `+:`

  which forces all translations of group names and group IDs to be made via the NIS service. This is the recommended procedure.

Administering NIS changes

Administering NIS takes place after the set up procedures have been completed. Data accessibility requirements over networks never remain stagnant; changes will always be needed. Creating and updating maps, adding new servers, changing the master server are the type of tasks you will need to do. This section explains how to make these changes. It includes instructions for:

- Modifying existing maps
- Propagating NIS maps
- Creating new NIS maps after installing NIS
- Adding an additional NIS server
- Changing the master server to a different host

Modifying existing maps

Always make changes to the databases served by NIS on the master server. To modify the databases you expect to change most frequently, such as /etc/passwd, follow these steps:

Step 1 Become superuser.

Step 2 Edit the ASCII database file.

Step 3 Change directory to /usr/etc/yp.

Step 4 Run `ypmake` (`makedbm`).

Step 5 Modify nonstandard maps manually.

These maps include:

- Maps specific to applications of a particular vendor or site
- Maps that change infrequently

- Maps for which no ASCII form exists; for example, maps created when you installed NIS.

Use `makedbm` with the `-u` option to disassemble them into a form that can be modified using standard tools, such as `awk`, `sed`, or `vi`.

Build a new version using `makedbm`, which can be done manually using either of two methods.

- Redirect the output of `makedbm` to a temporary file that can be modified and fed back into `makedbm`.
- Operate on the output of `makedbm` within a pipeline that feeds into `makedbm` again directly. This is appropriate if the disassembled map can be updated by modifying it with `awk`, `sed`, or a `cat` append.

Suppose you want to create a nonstandard NIS map, called mymap. You want it to consist of key-value pairs in which the keys are strings like al, bl, cl, etc. (l is for left), and the values are ar, br, cr (r is for right).

There are two possible procedures to follow when creating new maps: one uses an existing ASCII file as input; the other uses standard input.

Creating new maps from existing ASCII files

For example, an existing ASCII file named /usr/etc/yp/mymap.asc was created with an editor or a shell script. The map is located in the subdirectory, home_domain. Create the NIS map for this file by entering:

```
cd /usr/etc/yp

makedbm mymap.asc home_domain/mymap
```

At this point, you notice the map really should have included another 2-tuple: (dl, dr). You can make the modification simply. In all situations like this, remember to modify the map by first modifying the ASCII file. Modifications made only to the dbm files, and not also in the ASCII file, will be lost. Make the modification as follows:

```
cd /usr/etc/yp

makedbm mymap.asc home_domain/mymap
```

Creating new maps from standard input

When there is no original ASCII file, create the NIS map from the keyboard by entering input as follows:

```
cd /usr/etc/yp
makedbm - home_domain/mymap
al ar
bl br
cl cr
CTRL-D
```

When you need to modify that map, use `makedbm` to create a temporary ASCII intermediate file that you can edit using standard tools. For instance

```
cd /usr/etc/yp
makedbm -u home_domain/mymap > mymap.temp
```

At this point, you can edit mymap.temp to contain the correct information. Create a new version of the database using the commands

```
makedbm mymap.temp home_domain/mymap
rm mymap.temp
```

The preceding paragraphs explained how to use a few tools. Almost everything you must do can be done by `ypinit` and /usr/etc/yp/Makefile. If you add nonstandard maps to the database or change the set of NIS servers after the system is already running, other procedures such as those explained above are required.

Whether you use the Makefile in /usr/etc/yp or some other procedure, the goal is the same: a new pair of well-formed dbm files must end up in the domain directory on the master NIS server.

Propagating map changes from master to slaves

Maps must be changed on the master, then propagated from the master NIS server to all slave NIS servers. Initially, `ypinit` moves changes, as described in the section "Setting up a slave NIS server." After a slave NIS server has been initialized, updated maps are transferred from the master server by `ypxfr`. You can force the propagation of changes by running `ypxfr` using any one of three different methods: `cron`, `ypserv`, or interactive use of `ypxfr`. An explanation of each method follows.

Scheduling map updates using cron

Maps have differing rates of change; for instance, protocols.byname may not change for months at a time, but

passwd.byname may change several times a day in a large organization. You can set up entries in a crontab file to periodically run `ypxfr` at a rate appropriate for any map in your NIS database. `ypxfr` contacts the master server and transfers the map only if the master's copy is more recent than the local copy.

To avoid needing a crontab entry for each map, several maps with approximately the same change characteristics can be grouped in a shell script, and the shell script can be run from a crontab file. Suggested groupings, mnemonically named, can be found in /usr/etc/yp:

- ypxfr_1perhour
- ypxfr_1perday
- ypxfr_2perday

If the rates of change used by these scripts are inappropriate for your environment, you can easily modify or replace these shell scripts. For your site, look to see which maps are to be updated by each of these timed shell scrips.

These same shell scripts should be run on each NIS slave server in the domain. Alter the exact time of execution from one server to another, to prevent the checking from bogging down the master. If you want the map transferred from some particular server, not the master, you can specify that server within the shell script by using the `ypxfr -h` option. You can use the `-s` option to transfer maps from another domain. Finally, maps having unique change characteristics can be checked and transferred by explicit invocations of `ypxfr` within the system crontab file, /.crontab.

Starting update cycle from the master server with ypserv

You may also send files out from the master server, rather than asking for them on the slave server. `yppush` sends a Transfer Map RPC request to `ypserv`, which invokes `ypxfr` to send the map. `yppush` enumerates the NIS map ypservers, generating a list of NIS servers in your domain, and sends a Transfer Map request to each of the listed NIS servers. `ypserv` forks off a copy of `ypxfr`, invokes it with the `-C` flag, and passes it the information it needs to identify the map and call back the initiating `yppush` process with a summary status.

Entering ypxfr interactively

You can run `ypxfr` as a command. Typically, you do this only in exceptional situations, such as when you set up a temporary NIS server to create a test environment, or when you try quickly to get a NIS server that has been out of service consistent with the other servers.

Logging ypxfr activities

In most situations, except when started interactively, `ypxfr` transfer attempts and results can be captured in a log file. If /usr/etc/yp/ypxfr.log exists, results are appended to it. No attempt to limit the log file is made. To turn off logging, remove the log file.

Adding new maps

Adding a new NIS map entails getting copies of the map's dbm files into the domain directory on each of the NIS servers in a domain. This section describes how to get the proper files in place so the propagation works correctly. Both master and slaves must be set up correctly.

After deciding which NIS server is to be the master of the map, modify /usr/etc/yp/Makefile on the master server so that the map can be conveniently rebuilt. Filter an ASCII file through `awk`, `sed`, and/or `grep` to make it suitable for input to `makedbm`.

Consult the existing Makefile as a source for programming examples. Use the mechanisms already in place in /usr/etc/yp/Makefile when deciding how to create dependencies that `ypmake` will recognize; specifically, the use of .time files allows you to see when the Makefile was last run for the map.

To get an initial copy of the map, run `yppush` on the NIS master server. The map must be globally available before clients begin to access it. If the map is available from some, but not all, NIS servers you will see unpredictable behavior from client programs.

Adding a new NIS server

To add a host that has never been an NIS slave server, make the following modifications on the master NIS server:

Step 1 Become superuser on the master NIS server.

Step 2 Add the host's name to the ypservers map in the default domain. If a host name is not in ypservers, it will not be notified of updates to the NIS map files.

To illustrate, the command sequence for adding a server to domain_name is

```
vi /etc/ypservers
cd /usr/etc/yp
ypmake
```

Step 3 Change directory to /usr/etc/yp and run `ypmake`.

Step 4 Log in to the new NIS slave server and change directory to /usr/etc/yp.

Step 5 Set up the new slave NIS server's databases by copying the databases from NIS master server, `ypmaster`.

To do this, remote log in to the new NIS slave, and run the `ypinit` command. To continue on with the example, the commands are

```
cd /usr/etc/yp
ypinit -s ypmaster
```

Step 6 Make copies of the following files:

- /etc/passwd
- /etc/hosts
- /etc/group
- /etc/networks
- /etc/protocols
- /etc/netgroup
- /etc/services

Step 7 Edit the original files according to the instructions in the section "Altering NIS client files to use NIS services."

Changing a map's master server

To change a map's master, you first have to build it on the new NIS master. The old master's name occurs as a key-value pair in the existing map (this pair is inserted automatically by `makedbm`). Using an existing copy at the new master or transferring a copy to the new master with `ypxfr` is insufficient. You have to reassociate the key with the new master's name. If the map has an ASCII source file, you should copy it in its current version to the new master. The following instructions illustrate remaking a sample NIS map called `belmont.mass`.

Step 1 Become superuser on the new master NIS server.

Step 2 Build the map at the new master.

Step 3 Change directory to /usr/etc/yp and edit /usr/etc/yp/Makefile to add an entry for the new map.

Step 4 Remake the NIS map locally with the following command:

```
ypmake belmont.mass
```

Step 5 Edit the old master's /usr/etc/yp/Makefile—if it is to remain an NIS server—to comment out the section that made `belmont.mass` so it is no longer made on the old master.

Step 6 Remake maps which only exist as dbm databases on the new master by disassembling an existing copy from any NIS server, then running the disassembled version back through `makedbm`.

For example:

```
cd /usr/etc/yp
ypcat -k belmont.mass | /usr/etc/yp/makedbm -mydomain/belmont.mass
```

`belmont.mass` should be in the alias file too.

Step 7 Send a new copy of the new master's map to the other (slave) NIS servers. However, do not use `yppush`; the other slaves will try to get new copies from the old master instead of from the new one. Rather, become superuser on the old master server and enter

```
/usr/etc/yp/ypxfr -h newmaster belmont.mass
```

newmaster is the new master server's hostname.

Step 8 Run `yppush`.

Now that you have a new copy on the old master, the remaining slave servers still believe that the old master is the current master, and will attempt to get the current version of the map from the old master. When they do so, they will get the new map, which names the new master as the current master.

If the method above fails, the following option is cumbersome but effective. Login as superuser to each NIS server machine and enter

```
/usr/etc/yp/ypxfr -h newmaster belmont.mass
```

newmaster is the new master server's hostname.

This will work, but should be considered the worst case solution.

Troubleshooting an NIS client

This section provides suggestions to help you solve network problems encountered on NIS clients.

Hanging commands

Commands may be in a hang state even though the system as a whole seems okay and you are able to run new commands. The most common problem at an NIS client node is for a command to hang and generate the console message:

```
NIS: server not responding for domain <wigwam>. Still trying
```

The message above indicates that `ypbind` on the local host is unable to communicate with `ypserv` in the domain wigwam. This often happens when hosts running `ypserv` crash. It may also display if the network or the NIS server is so overloaded that `ypserv` cannot get a response back to your `ypbind` within the time-out period. Under these circumstances, all the other NIS client nodes on your net show the same or similar problems. This condition is temporary in most cases. The message usually goes away when the NIS server reboots and `ypserv` gets back in business, or when the load on the NIS server nodes or the Ethernet decreases.

On the other hand, in the circumstances described below, this condition does not resolve itself.

- The NIS client has not set, or has incorrectly set, domain name on the host. Clients must use a domain name that the NIS servers know. Use `domainname` to see the client domain name. Compare that with the domain name set on the NIS servers. The domain name should be set in /etc/netnfsrc. When /etc/netnfsrc fails to set or incorrectly sets domainname, do the following: become superuser on the host in question, edit /etc/netnfsrc to fix the domainname line with a proper domain name (this assures domain name will be correct every time the host boots), and set domainname manually so it is fixed immediately. To do this, enter:

  `domainname` *good_domain_name*

- If your domain name is correct, make sure your local network has at least one NIS server. You can automatically bind only to a `ypserv` process on your local network, not on another accessible network. There must be at least one NIS server for your host's domain running on your local network. Two or more NIS servers will improve availability and response characteristics for NIS services.
- If your local network has an NIS server, make sure it is up and running. Check other hosts on your local net. If several clients

have problems simultaneously, suspect a server problem. Find a client behaving normally, and try the `ypwhich` command. If `ypwhich` never returns an answer, kill it and go to a terminal on the NIS server by entering

```
ps -ef | grep yp
```

and look for `ypserv` and `ypbind` processes. If the server's `ypbind` daemon is not running, start it up by entering

```
/etc/ypbind
```

If there is a `ypserv` process running, do a `ypwhich` on the NIS server. If `ypwhich` returns no answer, `ypserv` has probably hung and should be restarted. Kill the existing `ypserv` process (you must be logged on as superuser), and start `/usr/etc/ypserv`, as follows:

```
kill -9 pid
```

```
/usr/etc/ypserv
```

If `ps` shows no `ypserv` process running, start one.

NIS service unavailable

When other hosts on the network appear to be okay, but NIS service becomes unavailable on your host, different types of symptoms may show up. Symptoms include

- Some commands appear to operate correctly while others terminate.
- Some commands limp along in a backup-strategy mode particular to the program involved.
- Some commands or daemons crash with obscure messages or no message at all.
- An error message prints about the unavailability of NIS.

Here are some examples of what you might see when one of these conditions occurs.

1. `my_machine#` **`ypcat myfile`**

   ```
   ypcat: can't bind to NIS server for domain
   <wigwam>.
   ```

   Reason: cannot communicate with `ypbind`.

2. `my_machine#` **`/usr/etc/yp/yppoll myfile`**

   ```
   Sorry, I can't make use of the yellow pages.
    I give up.
   ```

When symptoms like those shown in items 1 and 2 occur, try entering

```
ls -l
```

on a directory containing files owned by many users, including users not in the local host's /etc/passwd file; for example /usr. When the `ls -l` reports file owners, not in the local host's /etc/passwd file, as numbers rather than names, this is one more symptom that NIS service is not working.

These symptoms usually indicate that your `ypbind` process is not running. Run `ps ef` to check. If it is not running, start it by entering

```
/etc/ypbind
```

NIS problems should disappear.

ypbind crashes

If `ypbind` crashes almost immediately each time it is started, you should look for a problem in some other part of the system. Check for the presence of the `portmap` daemon by entering:

```
ps -ef | grep portmap
```

If you do not find it running, take the host down to single-user mode, then return it to multiuser mode. Because so many other daemons must be started after `portmap`, this is the least disruptive means of correcting the problem.

If `portmap` itself will not stay up or behaves strangely, look for more fundamental problems. Check the network software.

You may be able to talk to the `portmap` on your host from a host operating normally. From such a host, enter

```
/usr/etc/rpcinfo -p client
```

Where `client` is the name of your host. If your `portmap` is okay, the output should look like the following sample:

```
program vers proto   port
 100007    2   tcp   1024  ypbind
 100007    2   udp   1028  ypbind
 100007    1   tcp   1024  ypbind
 100007    1   udp   1028  ypbind
 100003    2   udp   2049  nfs
 100024    1   udp    686  status
 100024    1   tcp    688  status
 100021    1   tcp    689  nlockmgr
 100021    1   udp   1030  nlockmgr
 100021    3   tcp    693  nlockmgr
 100021    3   udp   1031  nlockmgr
 100020    1   udp   1032  llockmgr
 100020    1   tcp    698  llockmgr
```

On your host, the port numbers will be different. The following four entries represent the `ypbind` process:

```
100007    2    tcp    1024  ypbind
100007    2    udp    1028  ypbind
100007    1    tcp    1024  ypbind
100007    1    udp    1028  ypbind
```

If they are not there, `ypbind` has been unable to register its services. Reboot the host. If they are there and they change each time you try to restart /etc/ypbind, reboot the system, even if the `portmap` is up. If the situation persists after reboot, call the Hewlett-Packard Convex Technical Assistance Center.

ypwhich inconsistent

When you use `ypwhich` several times at the same client node, the answer you get back varies, whereas, the NIS server changes. This is normal. The binding of NIS client to NIS server changes over time on a busy net and when the NIS servers are busy. Whenever possible, the system stabilizes at a point where all clients get acceptable response time from the NIS servers. As long as your client gets NIS service, it does not matter where the service comes from. Often an NIS server gets its own NIS services from another NIS server on the net.

Troubleshooting an NIS server

This section provides suggestions to help you solve network problems encountered on NIS servers.

Multiple versions of an NIS map

Because NIS works by propagating maps among servers, you will sometimes find different versions of a map at servers on the network. This version skew is normal if transient, and abnormal otherwise.

Normal update is prevented when some NIS server or some router between NIS servers is down during a map transfer attempt. Normal updating of maps is described in the section "Propagating map changes from master to slaves." When all the NIS servers, and all the routers between them, are up and running, `ypxfr` should succeed.

If a particular slave server has problems updating, log in to that server and run `ypxfr` interactively. If `ypxfr` fails, it will tell you why it failed, then you can fix the problem. If `ypxfr` succeeds, but

you think it has been failing sometimes, create a log file to enable logging of messages by entering

```
cd /usr/etc/yp

touch ypxfr.log
```

This saves all output from `ypxfr`. The output looks much like what `ypxfr` creates when run interactively, except each line in the log file is time-stamped. You may see unusual ordering in the time stamps, which is okay. The time stamp tells you when `ypxfr` began its work. If copies of `ypxfr` ran simultaneously, but their work took differing amounts of time, they may actually write their summary status line to the log files in an order different from the order in which they were invoked.) Any pattern of intermittent failure shows up in the log. When you have fixed the problem, turn off logging by removing the log file. If you forget to remove it, it will grow without limit.

While still logged in to the problem NIS slave server, inspect the system crontab file, /usr/spool/cron and the `ypxfr*` shell scripts it invokes. Typos in these files cause propagation problems, as do failures to refer to a map within any shell script.

Also, make sure that the NIS slave server is in the map ypservers within the domain. If omitted, it will still work as a server, but `yppush` will not tell it when a new copy of a map exists.

If the problem is not obvious, you can work around it while you debug using `rcp`, `ftp`, or `tftp` to copy a recent version from any healthy NIS server. You may not be able to do this as superuser, but you can probably do it as daemon. For instance, the following example illustrates transfer of map `busted`:

```
# chmod go+w /usr/etc/yp/mydomain
# su daemon
# rcp ypmaster:/etc/yp/mydomain/busted.\* /usr/etc/yp/mydomain
# CTRL-D
# chown root /usr/etc/yp/mydomain/busted.*
# chmod go-w /usr/etc/yp/mydomain
```

Notice that the * character has been escaped in the command line, so it will be expanded on ypmaster, instead of locally on ypslave. Also, notice that the map files should be owned by root, so you must change ownership of them after the transfer. Obviously, if you can do the `rcp` as superuser, it makes the whole thing easier.

ypserv crashes

When the `ypserv` process crashes almost immediately, and will not stay up even with repeated activations, the debug process is

virtually identical to that described above in the section "ypbind crashes." Check for the `portmap` daemon:

```
ps -ef | grep portmap
```

Reboot the server if you do not find the daemon. If it is there, enter

```
/usr/etc/rpcinfo -p hostname
```

and look for screen output similar to the following:

```
program vers proto   port
 100004    2   tcp   1027  ypserv
 100004    2   udp   1024  ypserv
 100004    1   tcp   1027  ypserv
 100004    1   udp   1024  ypserv
 100007    2   tcp   1025  ypbind
 100007    2   udp   1035  ypbind
 100007    1   tcp   1025  ypbind
 100007    1   udp   1035  ypbind
 100003    2   udp   2049  nfs
 100024    1   udp    686  status
 100024    1   tcp    688  status
```

On your host, the port numbers will be different. The four entries that represent the `ypserv` process are

```
100004    2   tcp   1027  ypserv
100004    2   udp   1024  ypserv
100004    1   tcp   1027  ypserv
100004    1   udp   1024  ypserv
```

If they are not there, `ypserv` has been unable to register its services. Reboot the machine. If they are there, and they change each time you try to restart `ypserv`, reboot the machine. If the situation persists after reboot, call the Hewlett-Packard Convex Technical Assistance Center (TAC).

Other NIS error conditions

The following tables list NIS utilities responses when one or more daemons are killed or aborted. Network problems, host server network problems, or a missing host server `portmap` can also cause these responses.

ypserv killed or aborted

Table 8 lists `ypserv` responses.

Table 8 `ypserv` responses when killed or aborted

`ypserver# ypcat passwd`	Reason: cannot bind to a server that serves domain.
`ypserver# ypmatch <username> passwd ypmatch: Can't match key <username> in map passwd.byname.`	Reason: cannot bind to a server that serves domain.
`ypserver# ypwhich ypwhich: Domain <mydomain> not bound on <mydomain>.`	
`ypserver# ypwhich ypwhich: can't call ypbind on <ypclient>: RPC: Timed out`	Reason: Two ypbind processes have spawned during the ypwhich command, causing a time out.
`ypserver# /usr/etc/yp/yppoll passwd`	Never returns
`ypserver# /usr/etc/yp/ypset <mydomain>`	Never returns

ypbind killed or aborted

Table 9 lists `ypbind` responses.

Table 9 `ypbind` responses when killed or aborted

`ypclient# ypcat passwd ypcat: can't bind to yp server for domain <mydomain>.`	Reason: cannot communicate with `ypbind`.
`ypclient# ypmatch <username> passwd ypmatch: Can't match key username in map passwd.byname.`	Reason: cannot communicate with `ypbind`.
`ypclient# ypwhich ypwhich: can't call ypbind on <ypclient>: RPC: Timed out`	Insufficient information to determine specific cause.

Table 9 `ypbind` responses when killed or aborted **(continued)**

`ypclient# /usr/etc/yp/yppoll passwd RPC: Timed out`	Insufficient information to determine specific cause.
`ypclient# /usr/etc/yp/ypset <ypserver>Sorry, I can't make use of the yellow pages. I give up.`	Insufficient information to determine specific cause.

ypbind and ypserv not running

`ypbind` and `ypserv` response when not running:

```
ypclient# /usr/etc/yp/ypset -d <mydomain> ypserver
Sorry, I couldn't send my rpc message to ypbind on host <ypclient>.
```

Security with NIS

Security on a system running NIS is dependent upon how NIS consults the administrative files on which its maps are based. Local files on the host are consulted first, then the system consults maps in the NIS domain. For example, the system checks the NIS/etc/aliases file in the NIS domain for mail aliases, then checks the mail.aliases NIS map.

The remaining files on which NIS maps are based are global files:

- /etc/hosts
- /etc/networks
- /etc/ethers
- /etc/rpc
- /etc/services
- /etc/protocols
- /etc/netgroup

The information in these files is network-wide data and accessed only from NIS. However, when booting, each host needs an entry in /etc/hosts for itself. In summary, if NIS is running, global files are only checked in the NIS maps; a file on your local host is not consulted.

Special NIS password change

When you change your password with the `passwd` command, you change the entry explicitly given in your host's local /etc/passwd file. If your password is not given explicitly, but rather pulled in from NIS with a + entry, then the `passwd` command will print the error message

```
Not in passwd file.
```

To change your password in the NIS password file, you must use the `yppasswd` command. To enable this service, you must start up the daemon `yppasswd` server on the machine serving as the master for the NIS password file.

Network-wide groups: hosts and users

NIS uses the /etc/netgroup file on the master NIS server for permission checking during remote mount, login, remote login, and remote shell. It uses /etc/netgroup to generate three NIS maps in the /usr/etc/yp/domainname directory: netgroup, netgroup.byuser and netgroup.byhost. The NIS map netgroup contains the basic information in /etc/netgroup. The two other NIS maps contain a more specific form of the information to speed the lookup of netgroups given the host or user.

The programs that consult these NIS maps are `login`, `mountd`, `rlogin`, and `rsh`. `login` consults them for user classifications if it encounters netgroup names in /etc/passwd. `mountd` consults them for machine classifications if it encounters netgroup names in /etc/exports. `rlogin` and `rsh` consult the netgroup map for both machine and user classifications if they encounter netgroup names in the /etc/hosts.equiv or /.rhosts file.

Disabling NIS

If `ypserv` on the master is disabled, you can no longer update NIS maps.

If you choose not to use NIS, you can disable it by simply renaming the /etc/ypbind to /etc/ypbind.orig. (This is what the install script does automatically if you tell it you do not want to run NIS.)

```
ypclient# ps -ef | grep yp
ypclient# kill pid#
ypclient# mv etc/ypbind /etc/ypbind.orig
```

where *pid#* is the `yp pid` found using the previous command.

To disable NIS on a particular NIS slave or master, enter

```
ypclient# mv /usr/etc/ypserv
/usr/etc/ypserv.orig
```

Accessing and modifying NIS maps

This section describes how to use four interface programs designed to help you access or modify data stored in NIS maps. Actually, these maps are stored as dbm databases— indexed files with fast access provided through a hash table. Specifically, these programs—`ypcat`, `ypmatch`, `yppasswd`, and `ypwhich`—enable you to access selected data from various maps, modify your NIS password file, and determine which host is acting as a database server. None of these programs demand more than a casual knowledge of NIS.

`ypclnt`, also covered in this section, is a library of functions that enables you to create your own interface to NIS. Although `ypclnt` is more complicated to use than the other programs, individuals familiar with C language and the SPP-UX operating system should have no problems with it.

Printing values stored in NIS maps: ypcat

`ypcat` enables you to view on standard output the values from the NIS maps stored on your host or on other hosts across the network. `ypcat`, the cat equivalent for NIS, is typically used with a map name as an argument. The command sequence

```
# ypcat hosts.byaddr
```

writes the contents of the map hosts.byaddr to standard output, which is a listing of the names and addresses of network hosts. You can also invoke a map by using a "nickname." For example:

```
# ypcat hosts
```

produces the same output as hosts.byaddr. To list the nicknames for maps in your domain, enter `ypcat` with its `-x` argument as shown here

```
ypcat -x

Use "passwd" for map "passwd.byname"
Use "group" for map "group.byname"
Use "networks" for map "networks.byaddr
Use "hosts" for map "hosts.byaddr"
Use "protocols" for map "protocols.bynumber"
Use "services" for map "services.byname"
Use "aliases" for map "mail.aliases"
```

If you invoke `ypcat -x` on your host, you get a good look at the maps that are available to you.

To display maps from different domains, use the `-d` option. If, for example, you wanted to list networks.byname from the venus

domain, you would enter `ypcat` with the `-d` options, as shown in the following example:

```
ypcat -d venus networks
sun-ether       192.9.200       sunether ethernet
sun-oldether    125             sunoldether
mktg-net        105
ucb-ether       46              ucbether
arpanet         10              arpa
dragon-net      102
eth-net         101             eth
convex-net      100             convex
```

Printing selected data within NIS maps: ypmatch

`ypmatch`, like `ypcat`, enables you to view data stored in databases not only on your host but on other hosts connected to the network. Unlike `ypcat`, `ypmatch` enables you to get at *selected* parts of the database by printing information that matches *keys* that you provide. You enter a key, NIS searches an NIS map for a match, then writes the data associated with the matching key on standard output.

Suppose, for example, you want to check the Internet address of the host *venus*. You could log in to *venus* remotely, or you could use `ypcat` to print the contents of the *hosts* database on your host. A simpler method is to use `ypmatch` to search the *hosts* database for values associated with the *venus* key, as in

```
# ypmatch venus hosts
100.0.5.1 venus
```

The key you supply `ypmatch` must exactly match the key in the map. If you supply a key for which no values can be associated, you receive an error message:

```
# ypmatch loopback networks
ypmatch: Can't match loopback.
Reason: no such key in map
```

As with `ypcat`, you can use the *-x* option to display the map nickname table

```
# ypmatch -x
Use "passwd" for map "passwd.byname"
Use "group" for map "group.byname"
Use "networks" for map "networks.byaddr
Use "hosts" for map "hosts.byaddr"
Use "protocols" for map "protocols.bynumber"
```

```
Use "services" for map "services.byname"
Use "aliases" for map "mail.aliases"
```

You can also use the `-d` flag to access data in a remote database, just as you did with `ypcat`

```
# ypmatch -d eclipse docstaff aliases
```

```
moe, curly, larry
```

When you specify the `-k` option, the key is printed (with a colon) before the value of the key. This option is useful primarily for improving the clarity of the output you receive when you specify more than one key. For example

```
# ypmatch -k moe curly larry passwd
moe:
moe:VPF75JjbUCrvw:107:51:Moe,88/487,295,3415009:/doc/moe:/bin/csh
curly:
curly:WTqWgH2mHeFRc:172:51:Curly,89/488,296,5307495:/doc/curly:/bin/csh
larry:
larry:VL.fEiCaWwb/w:133:51:Larry,90/49,297,5273452:/doc/larry:/bin/csh
```

Specifying your NIS password: yppasswd

You can use `yppasswd` to change or install your NIS password. As you remember from the previous discussion of maps, user passwords are usually one of the first things networked when NIS is installed. The NIS password is much like a typical host password, but it can be used to sign onto many hosts across the network. Your NIS password may be different from the one on your own host.

Adding (or changing) a NIS password is just like adding or changing your password on a Convex Exemplar system. You enter `yppasswd` at a command prompt to invoke the program; then you enter your old password, followed by the new one. As with `passwd` on a Convex supercomputer, you must enter the new password twice. The following example shows a typical `yppasswd` terminal session: (You may receive some messages not listed here, depending on how you use the command.)

```
# yppasswd
```

```
Old yp passwd: enter_old_password_here
New yp passwd: enter_new_password_here
Retype new passwd: enter_new_password_here
```

Your new password must be at least six characters long. If you are not the password "owner," you must be a superuser to change a password. In either case, you must prove you know the old password. The update protocol passes all the information to the server in one RPC call, without ever looking at it. Thus if you enter

your old password incorrectly, you are not notified until after you have entered your new password.

If you try to change your password on a host that is not running the `yppasswd` daemon (`/usr/etc/rpc.yppasswdd`), you receive a message similar to the following:

```
# yppasswd
convex1 is not running yppasswd daemon
```

If you receive this message, log in to the `passwd` master server and try again.

Querying about maps and NIS services: ypwhich

`ypwhich` provides a general information about which server is supplying NIS services and which host is acting as master for a particular map.

In the simplest case, you can use `ypwhich` just as you did `ypcat` and `ypmatch` to display a list of maps and their nicknames. You do this by using the `-x` option, as follows:

```
# ypwhich -x
Use "passwd" for map "passwd.byname"
Use "group" for map "group.byname"
Use "networks" for map "networks.byaddr"
Use "hosts" for map "hosts.byaddr"
Use "protocols" for map "protocols.bynumber"
Use "services" for map "services.byname"
Use "aliases" for map "mail.aliases"
```

The `-m` flag enables you to find out which host is the master server for a particular map. Suppose, for example, that you want to find out which host is the master server for the RPC map you are using. Simply invoke `ypwhich` with the `-m` option and the name of the map:

```
# ypwhich -m rpc
convex1
```

In this example, *convex1* is the master server for the RPC map.

To locate master servers for maps in other domains, use the `-d` switch in conjunction with the `-m` switch:

```
# ypwhich -d eclipse -m networks
convex2
```

The `-V1` and `-V2` flags allow you to determine, respectively, which server is supplying Version 1.0 and Version 2.0 of the

Yellow Pages. (Version 1.0 is provided for backward compatibility with older systems from other vendors. Version 2.0 is used as the default Convex version of NIS.) For example:

```
# ypwhich -V1
convex1
# ypwhich -V2
convex1
```

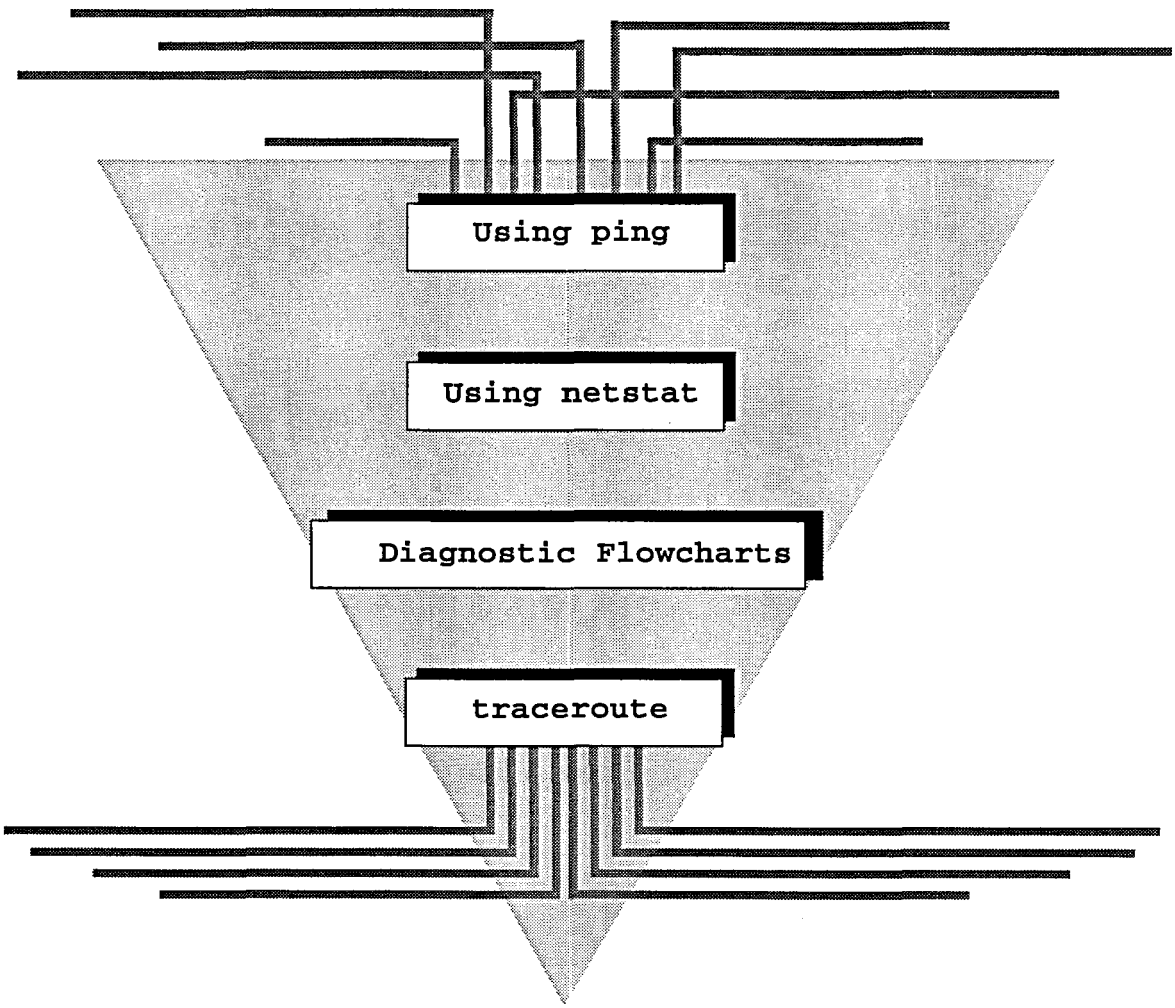
You can print a list of every map in a domain, with its associated master server, by invoking the `-m` option without arguments. A sample output follows

```
# ypwhich -m
bootparams convex1
mail.aliases convex1
pwrestrict.byuid convex1
pwrestrict.byname convex1
netgroup.byhost convex1
netgroup.byuser convex1
netgroup convex1
protocols.byname convex1
services convex1
services.byname convex1
rpc.byname convex1
rpc.bynumber convex1
networks.byaddr convex1
networks.byname convex1
ethers.byname convex1
ethers.byaddr convex1
hosts.byaddr convex1
hosts.byname convex1
group.bygid convex1
group.byname convex1
passwd.byuid convex1
protocols.bynumber convex1
ypservers convex1
passwd.byname convex1
```

To list maps in other domains, use the `-d` option in conjunction with the `-m` switch, as follows:

```
# ypwhich -d eclipse -m
```

Troubleshooting your network



Troubleshooting Internet services

11

Troubleshooting network interface problems can be difficult because a variety of hardware and software components may be involved and because the problem affecting your network may originate in another part of the network interface.

This chapter illustrates how to use the `ping` and `netstat` commands.

Using ping

The simplest way to test the network is to use `ping`. `ping` tests connections with other hosts by sending them a particular type of datagram, called an ICMP (Internet Control Message Protocol) ECHO. These datagrams work well for testing connections, because they are simply bounced back by the other host once they are received.

To test a connection to another host, start the transmission stream by invoking `ping`, and then check to make sure that packets are being returned. If at least 95 percent of the packets are being returned, you know that the network is functioning properly.

Use `ping` as follows:

`ping -r -v -o` *remote_host_name [packetsize] [count]*

where

`-r`

Bypasses the normal routing tables and sends datagrams directly to a host on an attached network.

`-v`

Displays verbose output (ICMP packets other than ECHO are listed).

`-o`

Inserts a "record route" IP option in outgoing packets, which summarizes routes taken when the program exits.

remote_host_name

Specifies the name of the remote machine with which you want to communicate. Can be a hostname (from /etc/hosts) or an Internet address in "dot" notation.

packetsize

Specifies the size of transmitted packets. The default is 64 bytes; the minimum value of packetsize is 8 bytes, the maximum is 4096 bytes.

count

Specifies the number of packets `ping` transmits before terminating. The default sends packets until interrupted.

`ping` has more options than are presented here. For a complete summary, refer to the ping(1M) man page.

The following example illustrates using `ping` to check whether the connection between `veeger` and `jupiter` is viable:

```
# ping jupiter
PING jupiter: 64data bytes
64 bytes from 140.50.0.1: icmp_seq=0. time=40. ms
64 bytes from 140.50.0.1: icmp_seq=1. time=40. ms
64 bytes from 140.50.0.1: icmp_seq=2. time=20. ms
64 bytes from 140.50.0.1: icmp_seq=3. time=20. ms
64 bytes from 140.50.0.1: icmp_seq=4. time=20. ms
64 bytes from 140.50.0.1: icmp_seq=5. time=30. ms
64 bytes from 140.50.0.1: icmp_seq=6. time=30. ms
64 bytes from 140.50.0.1: icmp_seq=7. time=30. ms
----veeger-il PING Statistics----
8 packets transmitted, 8 packets received, 0% packet
loss round-trip (ms) min/avg/max = 20/29/40
```

In this example, eight packets were received from network address 140.50.0.1 with transmission times ranging from 20 to 40 milliseconds. No packets were lost; therefore, you can assume that the link between the hosts is viable. If no packets had been returned, or if there had been a high rate of packet loss, there was most likely a problem in the line between hosts. Return-trip-time statistics included in the output allow you to determine whether transmission times are prohibitive.

You can specify the size and number of packets sent over the network. If you specify the number of packets, you must also specify the size. In the following example `ping` sends 3 packets with a data byte size of 24 to the remote host `jupiter`:

```
# ping -v jupiter 24 3
PING jupiter.convex.com [140.50.0.1]: 24 data bytes
24 bytes from 140.50.0.1: icmp_seq=0. time=7. ms
32 bytes from 140.51.1.2: icmp_type=3 (Dest Unreachable)
x 0: x45000024
x 4: xa4630000
x 8: xfe010000
x c: x82a84601
x10: x82a84701
x14: x 3032d16
x18: x 0
x1c: x4500004c
x20: xda070000
x24: x1d110000
x28: x82a84701
x2c: x82a84601
icmp_code=3
24 bytes from 140.50.0.1: icmp_seq=1. time=7. ms
24 bytes from 140.50.0.1: icmp_seq=2. time=8. ms
----jupiter.convex.com PING Statistics----
3 packets transmitted, 3 packets received, 0% packet loss
```

Using netstat

`netstat` displays statistical "snapshots" of various aspects of network operation. These snapshots enable you to monitor network use and efficiency during periods of normal operation, and to quickly track down problems when the network is malfunctioning. A few of the most useful `netstat` options are discussed here. For information about other `netstat` options, refer to the netstat(1C) man pages.

When invoked with no options, `netstat` displays the status of current network transmissions. Network addresses are shown as host names. `netstat` has the following command syntax:

`netstat -i[`*`interval`*`] -n -r -s`

where

`-i`

Shows the state of auto-configured interfaces (interfaces statically configured into a system, but not located at boot time are not shown). The *interval* option is a number which causes the statistics to update and redisplay every *n* seconds. See the section "Displaying status of autoconfigured interfaces" for more information.

`-n`

Shows network address as numbers. See the section "Displaying addresses numerically" for more information.

`-r`

Shows routing tables. If used with `-s` option show routing statistics. See the section "Displaying routing information" for more information.

`-s`

Shows statistics for all protocols. See the section "Displaying protocol statistics" for more information. The two most common protocols used by Convex Internet Services are `tcp` and `udp`.

interval

Specifies how often to redisplay a running count of statistics related to network interfaces. *interval* is counted as seconds.

`netstat` has more options than are presented here. For a complete summary, see netstat(1) man page.

Displaying status of autoconfigured interfaces

To make quick checks on network status and to check the fate of outgoing `ping` packets use `netstat -i`. The command displays information for all network interfaces autoconfigured at boot time in the following format:

```
# netstat -i
Name    Mtu   Network       Address          Ipkts    Ierrs    Opkts Oerrs  Coll
eth0    1500  syssw-net     lynx             63986        0    63986     0     0
fddi0   4352  fddi2-net     lynx-f           255918       5  6760314     0     0
```

The output shows interfaces that are up (`syssw-net` and `fddi2-net`), the name and addresses of the interfaces, and the maximum size of packets that can be transmitted across each network (listed under the column headed by "Mtu").

`netstat -i`displays statistics on the number of incoming packets (*Ipkts*), incoming packet errors (*Ierrs*), outgoing packets (*Opkts*), outgoing packet errors (*Oerrs*), and "collisions" on the network (*Collis*). (Collisions occur when two computers try to transmit packets across the network at exactly the same instant.)

If you suspect trouble with a particular remote host, you can use `netstat -i` to watch the network in real time. Use the following procedure:

Step 1 Run `ping` to start a packet stream directed toward the host in question. For example:

```
ping obewan > & /dev/null &
```

Step 2 Start `netstat -i` with an appended update, or interval, argument.

The following sample output from the appended argument, `3`, causes `netstat` to update the statistics and redisplay them every *n* seconds. In this example, the display is updated every three seconds. Statistics are displayed only for the most active interface.

```
# netstat -i 3
    input   (lo0      output              input  (Total)     output
 packets  errs  packets  errs colls      packets  errs  packets  errs colls
  717561    30   657802     0     0   717831    30   658072     0     0
       4     0        3     0     0        4     0        3     0     0
       6     0        7     0     0        6     0        7     0     0
      12     0        3     0     0       12     0        3     0     0
       4     0        4     0     0        4     0        4     0     0
      12     0       13     0     0       12     0       13     0     0
      14     0       10     0     0       14     0       10     0     0
      14     0       13     0     0       14     0       13     0     0
      11     0        9     0     0       11     0        9     0     0
```

Step 3 Check to see if packets are being sent and received by the local host.

Displaying addresses numerically

`netstat -n` displays the status of current network transmissions. Internet addresses are displayed instead of host names. The `-n` option may be used with any display format.

The following output enables you to determine the status of current network connections. The `Proto` field shows you which protocols are being used, and the `state` codes displayed show the state of the protocol when `netstat` was run.

```
# netstat -n
Active Internet connections
Proto Recv-Q Send-Q Local Address          Foreign Address        (state)
tcp        0      0  100.0.28.2.1324       100.0.1.13.25          TIME_WAIT
tcp        0      0  100.0.28.2.1021       100.0.1.13.513         ESTABLISHED
tcp        0      0  100.0.28.2.513        100.0.1.13.1022        FIN_WAIT_2
tcp        0      0  100.0.28.2.513        100.0.1.13.1023        FIN_WAIT_2
tcp        0      0  100.0.28.2.1022       100.0.1.13.513         ESTABLISHED
tcp        0      0  100.0.28.2.1014       100.0.1.13.1022        ESTABLISHED
```

Displaying routing information

This section describes how to display routing information with `netstat`.

Displaying routing tables

Use `netstat -r` to display available routes and their status. Each route consists of a destination host or network and a gateway to use in forwarding packets as shown in the following output:

```
# netstat -r

Routing tables
Destination      Gateway           Flags     Refs     Use  Interface
localhost        localhost         UH           3     246  lo0
fddi2-net        lynx-f            U          229       0  fddi0
```

Direct routes are created for each interface attached to the local host; the `Gateway` field for such entries shows the address of the outgoing interface.

The flags field shows the state of the route:

`U`
: Route is up

`G`
: Route is to a gateway

`H`
: Route is for a particular host

`D`
: Route was created dynamically by a redirect

`M`
: Route was modified by a redirect

The `Refs` field gives the current number of active uses of the route.

Connection oriented protocols normally hold on to a single route for the duration of a connection while connectionless protocols obtain a route while sending to the same destination. The `Use` field provides a count of the number of packets sent using that route.

The `Interface` entry indicates the network interface used for the route.

Displaying routing statistics

When you use the `-r` option with the `-s` option, `netstat` displays routing statistics, similar to that shown in the following example:

```
# netstat -rs

routing:
        0 bad routing redirects
        0 dynamically created routes
        0 new gateways due to redirects
        8 destinations found unreachable
        21242 uses of a wildcard route
        0 routes marked doubtful
        0 routes cleared of being doubtful
        3 redirects deleted
```

Displaying protocol statistics

`netstat -s` displays statistics about all protocols in use, as shown in the following example:

```
# netstat -s

tcp:
        376489 packets sent
                300155 data packets (65645447 bytes)
                2670 data packets (1372635 bytes) retransmitted
                67679 ack-only packets (16951 delayed)
                20 URG only packets
                106 window probe packets
                4379 window update packets
                2500 control packets
        377019 packets received
                251408 acks (for 65558061 bytes)
                9037 duplicate acks
                0 acks for unsent data
                233240 packets (36613597 bytes) received in-sequence
                645 completely duplicate packets (231932 bytes)
                645 packets with some dup. data (231932 bytes duped)
                31 out-of-order packets (338 bytes)
                1 packet (536 bytes) of data after window
                1 window probe
                3836 window update packets
                53 packets received after close
                7 discarded for bad checksums
                0 discarded for bad header offset fields
                0 discarded because packet too short
        967 connection requests
        0 connection accepts
        1394 connections established (including accepts)
        249 connections closed (including 129 drops)
        29 embryonic connections dropped
        239360 segments updated rtt (of 235928 attempts)
        1360 retransmit timeouts
                9 connections dropped by rexmit timeout
        133 persist timeouts
```

```
        177 keepalive timeouts
                115 keepalive probes sent
                13 connections dropped by keepalive
udp:
        0 incomplete headers
        0 bad data length fields
        3 bad checksums
        0 socket overflows
        3 data discards
ip:
        1369846 total packets received
        0 bad header checksums
        0 with size smaller than minimum
        0 with data size < data length
        0 with header length < data size
        0 with data length < header length
        142517 fragments received
        0 fragments dropped (dup or out of space)
        1 fragment dropped after timeout
        1 packets forwarded
        0 packets not forwardable
        0 redirects sent
icmp:
        17 calls to icmp_error
        0 errors not generated 'cuz old message was icmp
        Output histogram:
                echo reply: 23
                destination unreachable: 17
        0 messages with bad code fields
        0 messages < minimum length
        0 bad checksums
        Input histogram:
                echo reply: 2
                source quench: 1
                echo: 23
        23 message responses generated
arp:
        0 Bad packet lengths
        0 Bad protocol requests
        0 Bad headers
```

A typical LAN uses many different protocols. Transmission Control Protocol (TCP) is used by the standard utilities for packet transmission. User Datagram Protocol (UDP) is used by `rwhod` for the transmission of user datagrams. Internet Control Message Protocol (ICMP) is the protocol used by `ping`. Internet Protocol (IP) is the underlying protocol layer associated with TCP, UDP, ICMP, and other protocols.

As you look over the output from `netstat -s`, check the checksums to see how many transmissions have had errors. (Checksums are validity checks on the packets transmitted.) If the

percentage of bad checksums is high, you more than likely have problems with the LAN cabling, transceiver, or connections. Unless the percentage is high, though, do not worry—networks are designed to handle errors.

The `udpcksum` parameter, enables checksumming of User Datagram Protocol (UDP) datagrams. UDP checksumming results in additional overhead, because each transmitted and received UDP datagram is checksummed when the parameter is turned on.

Troubleshooting flowcharts 12

Once you have properly configured NFS and NIS into your network, you should experience few serious problems. Often, you will find that problems reported are the result of user error. A user trying to connect to a nonexistent host, for example, may reach the conclusion that `ftp` is broken when the connection times out. Or, a user may decide the network is down when trying to connect to a host that is not running.

You can address most other problems—runaway jobs, missing daemons, lack of kernel memory, bad connections, unplugged transceivers—by following these flowcharts.

Diagnostic flowcharts

Table 10 summarizes the types of network tests contained in the following flowcharts. Follow the flowcharts in sequence, beginning with Flowchart 1, to diagnose a network problem.

Table 10 Flowchart summary

Flowchart	Description
1	Network interface connections; checks that all the hardware connections between your system and the network interface are connected and operational.
2 & 3	Configuration test; verifies the configuration of the network interface on a host using `ifconfig`.
4 & 5	Network level loopback test; checks roundtrip communication between network layers on the source and target host using `ping`.
6	Transport level loopback test (using ARPA); checks roundtrip communication between transport layers on the source and target host using `telnet` and `ftp`.

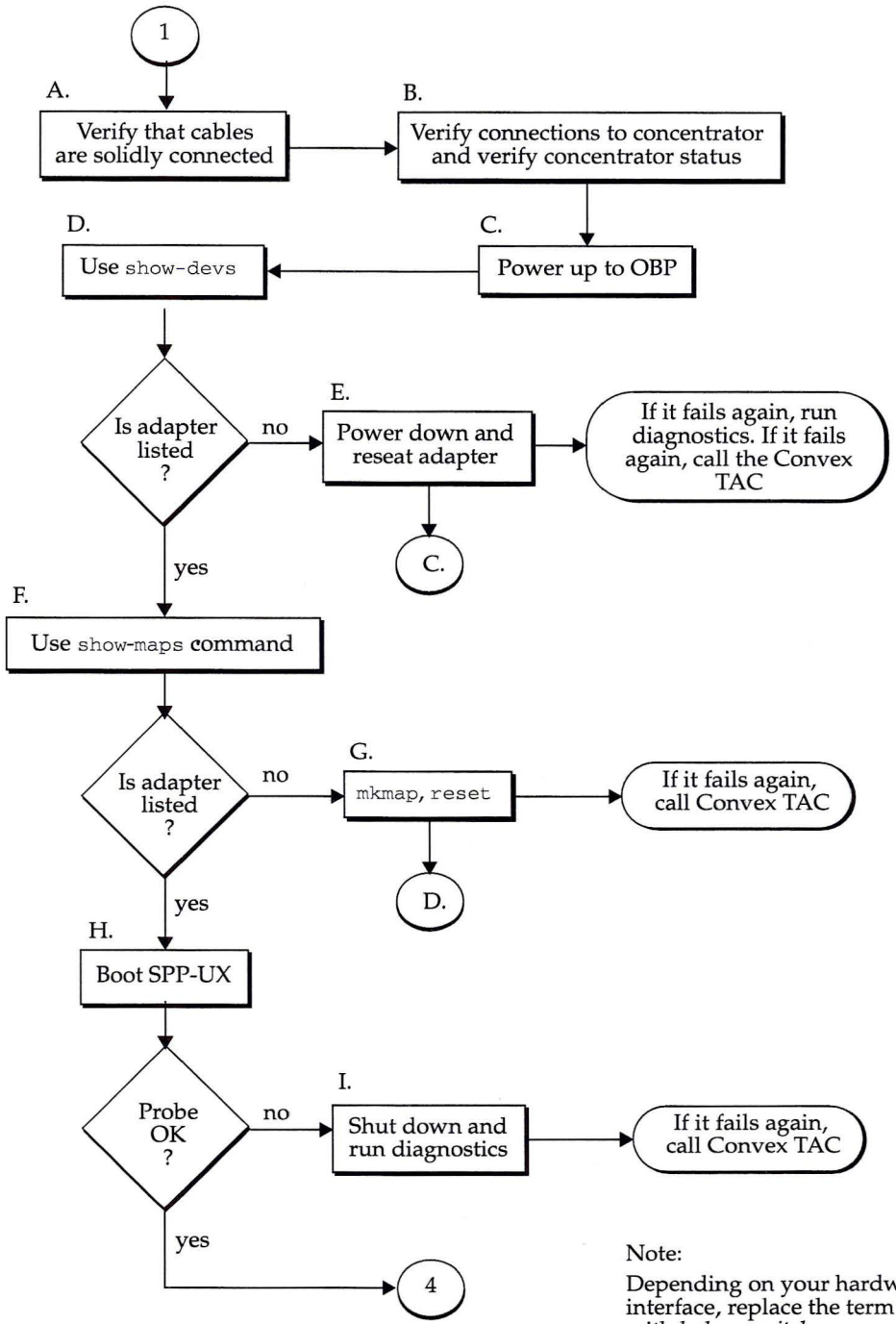


Figure 58 Flowchart 1: Network interface connections

Flowchart 1: Network interface connections

Figure 58 contains the network interface connections flowchart.

A.

Verify that all hardware cables are solidly connected.

B.

Verify that all connections to the concentrator are solid, and verify the concentrator's status.

C.

Power up to OBP.

D.

Use the `show-devs` command to verify that the adapter is listed.

E.

Power down the system and reset the adapter. If the test fails a second time, run diagnostics and Flowchart 1 begin again. If it fails a third time, call the Hewlett-Packard Convex TAC.

F.

Enter the `show-maps` command to verify that the adapter is listed.

G.

Enter `mkmap`, then `reset`. If the test fails again, call the Hewlett-Packard Convex TAC.

H.

Boot SPP-UX. Watch the probe in the boot sequence to verify that the device is initialized.

I.

If the probe is not OK, shut down the system and run diagnostics.

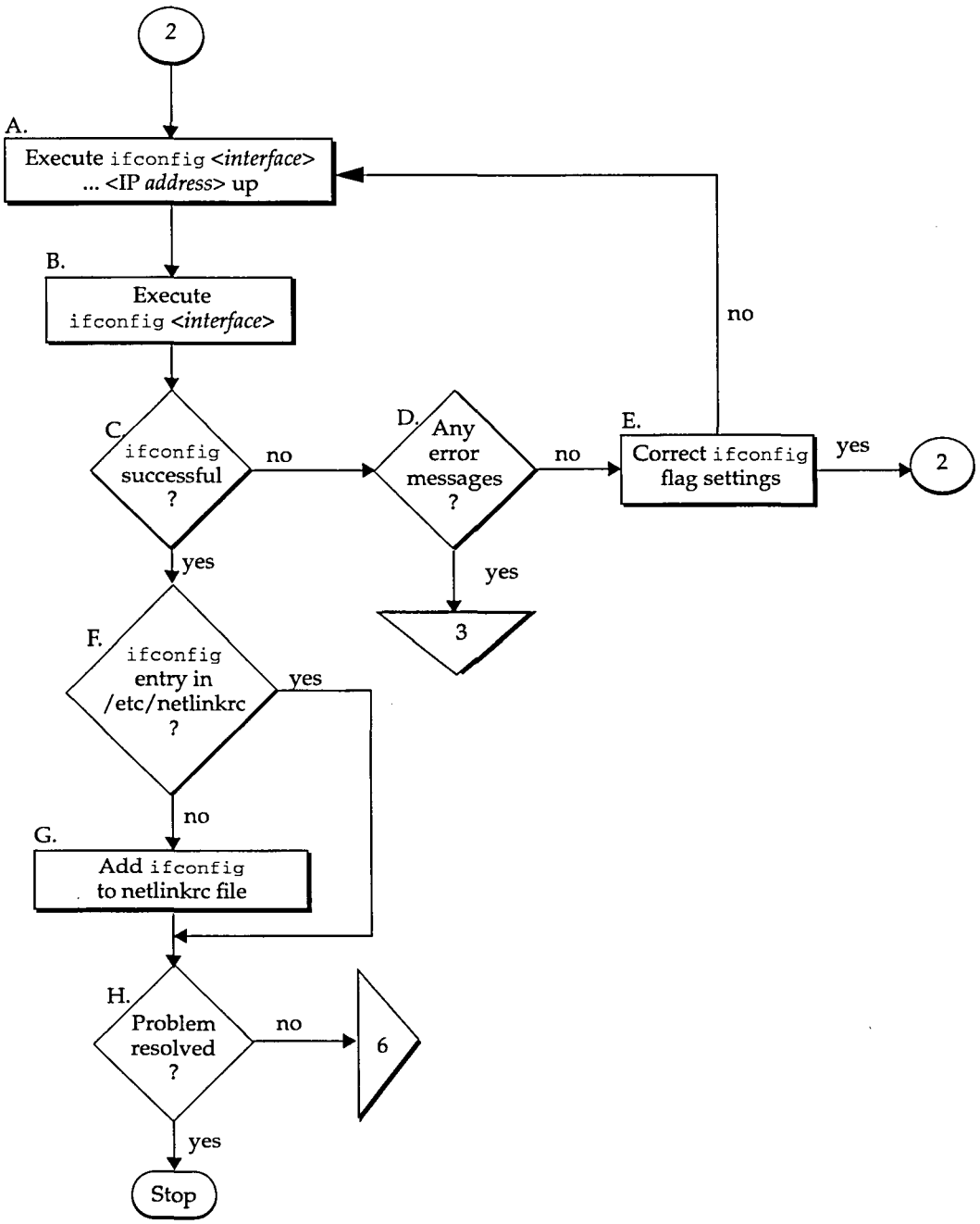


Figure 59 Flowchart 2: Configuration test

Flowchart 2: Configuration test

Figure 59 contains more of the configuration test.

A.

Execute `ifconfig` <interface<>IP address> up to be sure that the interface is enabled.

B.

Execute `ifconfig` <interface> again without the up parameter to check the flag setting for the up parameter.

C.

`ifconfig` successful? `ifconfig` is successful if the output shows the correct Internet address and the flags up, broadcast, route, notrailers, and running.

D.

Any error message returned? If `ifconfig` is not successful, and an error message appears, go to Flowchart 5, which shows error messages and what you should do in response to each. If no error message appears, go to E.

E.

Correct `ifconfig` flag settings. If `ifconfig` returns an incorrect flag setting, reexecute the command with the proper setting.

F.

`ifconfig` entry in /etc/netlinkrc? Check that there is an `ifconfig` entry in the netlinkrc file for your hardware adapter.

G.

Add `ifconfig` command to netlinkrc, using the format in the appropriate configuration chapter, and reboot.

H.

Problem resolved? If you have found and corrected the adapter problem, stop. If not, recheck your connections to the wall plug and concentrator. If the problem is still unresolved, it appears to be outside of your domain. Contact the Hewlett-Packard Convex TAC.

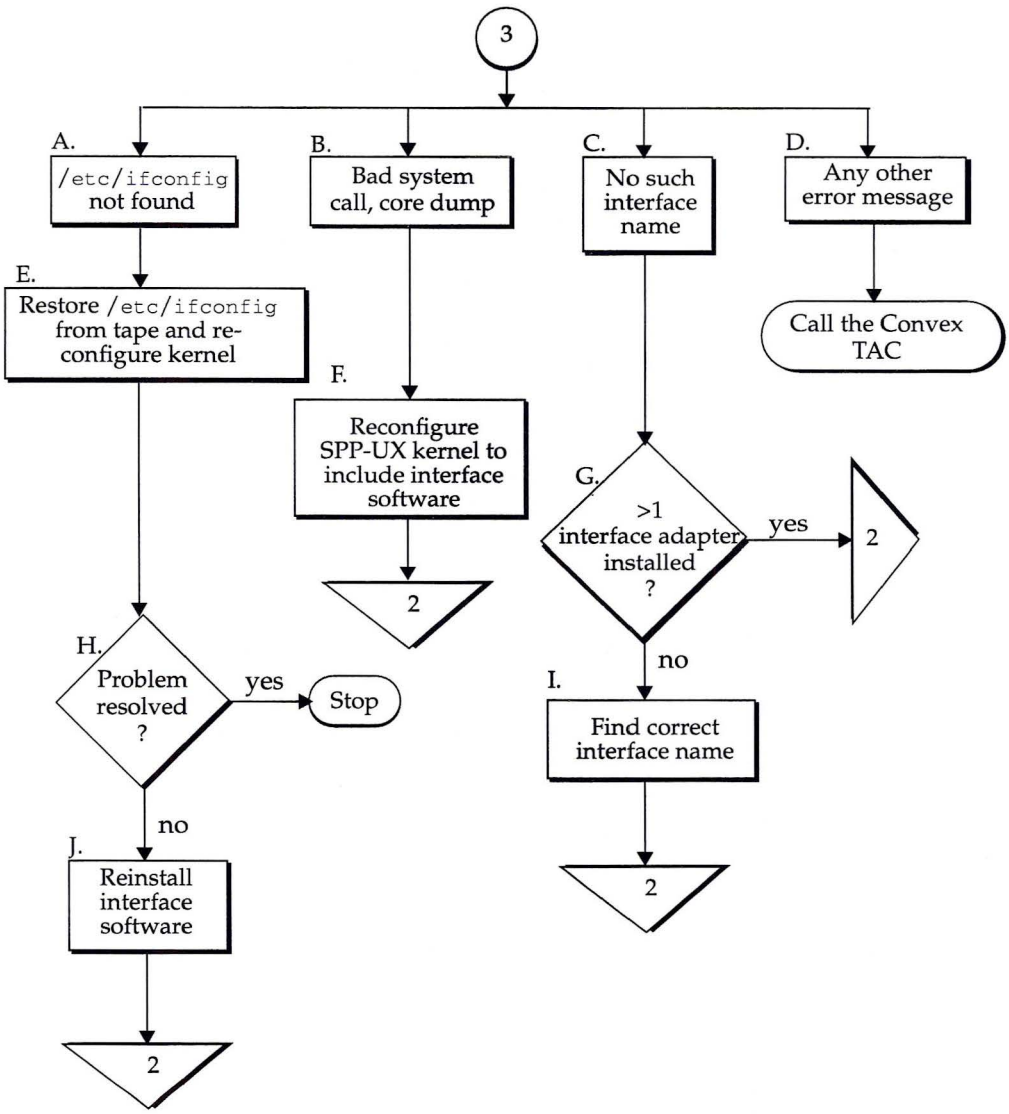


Figure 60 Flowchart 3: Configuration test **(continued)**

Flowchart 3: Configuration test (continued)

Figure 60 contains the final steps in the configuration test.

A.

/etc/ifconfig not found. The command may have been relocated on the system or deleted.

B.

Bad system call; core dumped. Networking is not configured into the kernel.

C.

No such interface name. The interface name passed to `ifconfig` does not exist on the system. Check spelling and names of interfaces on the system using `netstat -i`. Then verify that the interface was probed and attached during the boot sequence.

D.

Any other error message. If you received an error message not listed on the flowchart, interpret the message and take the appropriate action. If you need assistance, call the Hewlett-Packard Convex TAC.

E.

Restore /etc/ifconfig from tape and reconfigure the kernel.

F.

Reconfigure SPP-UX kernel to include network interface software.

G.

>1 adapter installed? If you have more than one adapter installed, go to Flowchart 10.

H.

Problem resolved? If so, stop. If not, reinstall the entire software product and start again with Flowchart 2.

I.

Find correct interface name. Using the correct interface name, start again with Flowchart 2.

J.

Reinstall software and start again with Flowchart 2.

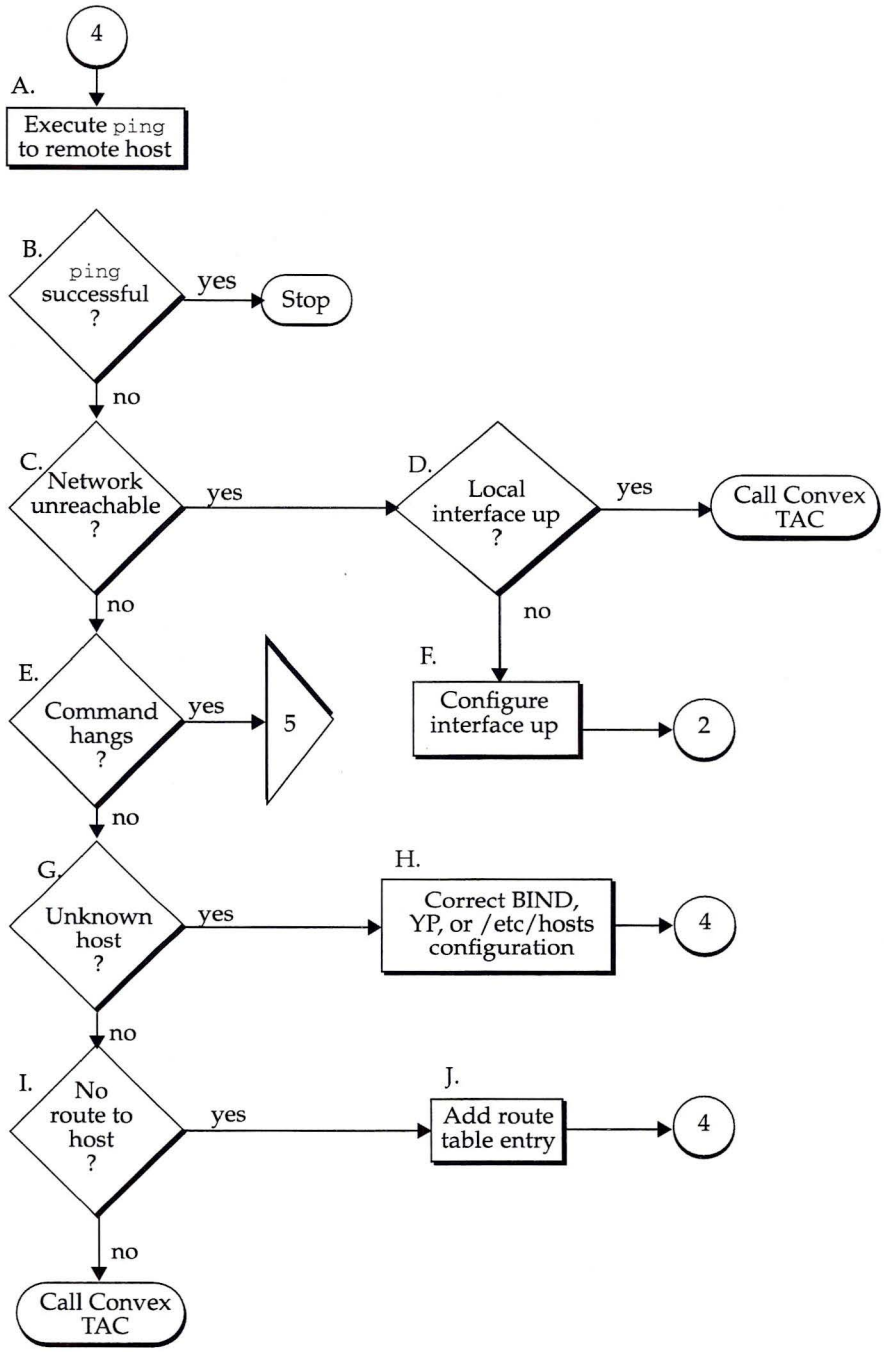


Figure 61 Flowchart 4: Network level loopback test

Flowchart 4: Network level loopback test

This test, shown in Figure 61, verifies that all hardware and software connections are operational.

A.

Execute `ping` to remote host. Using `ping`, send a message to the remote host you are having problems connecting to.

B.

`ping` successful? A message is printed on stdout for each `ping` packet returned by the remote host. If packets are being returned, your system has network connectivity to the remote host.

C.

Network unreachable? If so, check the status of the local interface first.

D.

Local interface up? Execute `ifconfig` on the local interface to be sure it is configured up.

E.

Command hangs? If a message is not returned after executing `ping`, go to Flowchart 7.

F.

Configure interface up. If you find the local interface is not up, execute `ifconfig` with the appropriate flags set. Refer to Flowchart 4, then start again with Flowchart 6.

G.

Unknown host? Error = Unknown host hostname?

H.

Correct BIND, YP, or /etc/hosts configuration. Add the missing host name and start again with Flowchart 6.

I.

No route to host? Error = Sendto: No route to host? If so, go to J. Otherwise, call the Hewlett-Packard Convex TAC for help.

J.

Add route table entry. Using /etc/route, add a route table entry for that host.

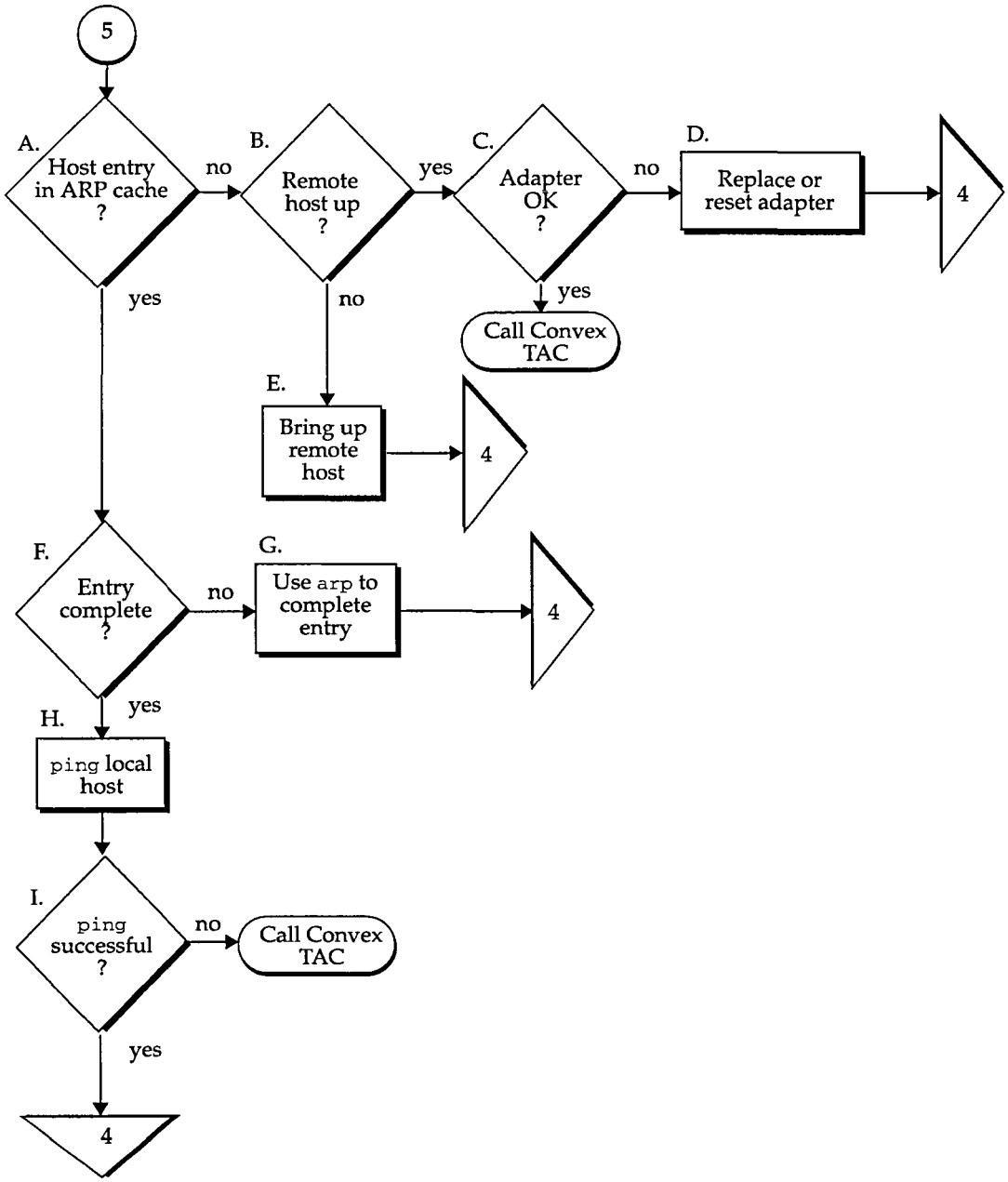


Figure 62 Flowchart 5: Network level loopback test (continued)

Flowchart 5: Network level loopback test (continued)

Figure 62 contains the final steps of this test.

A.

Host entry in ARP cache? Using `arp`, check that an entry exists for the remote host in your system's ARP cache. Note that some interfaces do not support `arp`. Go to Steps B and H.

B.

Remote host up? If there is no ARP cache entry for the remote host, check that the remote host is up. If not, the remote host has not broadcast an ARP message, and that likely is why there is no entry in the ARP cache.

C.

Adapter OK? Use hardware diagnostics to ensure that the adapter is operational.

D.

Replace or reset adapter. When the adapter is operational, use **`ifconfig`** *[interface]* **`down`** to reset.

E.

Bring up remote host. Have the node manager of the remote host bring that system up.

F.

Entry complete? ARP cache entry is wrong or not complete?

G.

Use arp to complete the entry. Using `arp`, enter the correct Station Address. Refer to the arp(1) man page for details.

H.

`ping` local host. Using `ping`, do an interval loopback on your own system. In other words, `ping` your own system.

I.

`ping` successful? If the internal loopback is successful, your system is operating properly to the Network Layer (OSI Layer 3). In addition, you know an ARP cache entry for the remote host exists on your own system. If this is true, the network interface or software on the remote host is suspect. Start again with Flowchart 6, but this time `ping` from the remote host to your system.

J.

`netisr` running? Use `ps` to check that `netisr` is an active process on your host.

K

Start `netisr`. If the `netisr` daemon is not running, make sure that it is configured as an interrupt.

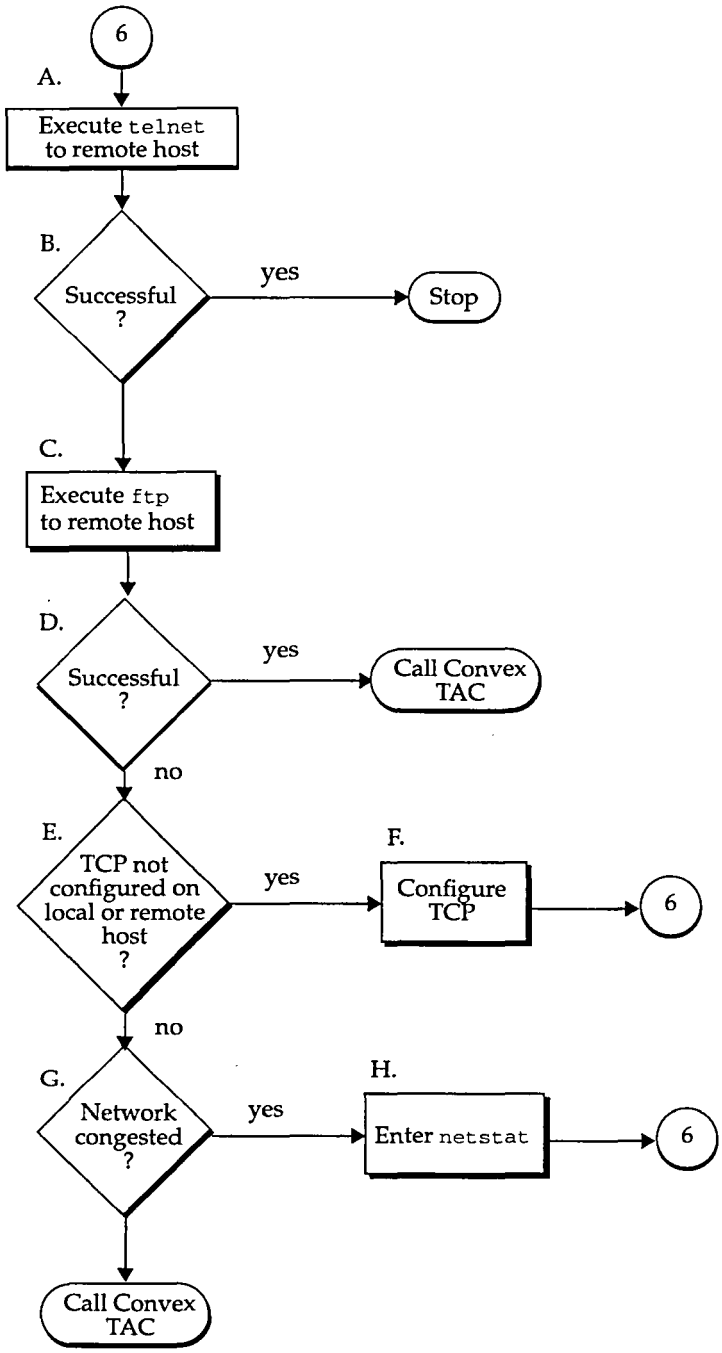


Figure 63 Flowchart 6: Transport level loopback test

Flowchart 6: Transport level loopback test

This test, shown in Figure 63, checks roundtrip communication between Transport Layers on the source and target host using ARPA services `telnet` and `ftp` commands.

A.

Execute `telnet` to remote host. Try to establish a telnet connection to the remote host.

B.

Successful? If your `telnet` attempt was successful, stop. The connection has been made through the Transport Layer.

C.

Execute `ftp` to remote host. Unlike `telnet`, `ftp` does not go through a pseudoterminal driver (pty) on your system. This step tests to see if the pty is why `telnet` failed.

D.

Successful? If `ftp` is successful, you likely have a problem with a pty on your system. Contact Hewlett-Packard Convex TAC.

E.

TCP not configured on local or remote host? Neither telnet nor ftp will work if TCP is not configured on either side of the connection.

F.

Configure TCP. If necessary, install TCP on either or both hosts.

G.

Network congested? If TCP is installed on both hosts, do a file transfer to another remote host on the network. Use netstat to check for lost packets. If 10 percent or more packets are lost, the network is extremely busy. If you cannot determine the cause, contact the Hewlett-Packard Convex TAC. If network congestion is not the cause, more detailed diagnostics are required. Again, contact the Hewlett-Packard Convex TAC.

Troubleshooting NFS and NIS 13

Before you investigate any NFS-related problem, become familiar with the names and functions of various NFS daemons and database files. Suggested readings are man pages on:

- mount(8)
- nfsd(8)
- biod(8)
- showmount(8)
- rpcinfo(8)
- mountd(8c)
- inetd(8c)
- fstab(5)
- mtab(5)
- exports(5).

When you track an NFS problem, keep in mind the three main points of failure:

- Server
- Client
- Network

You must isolate each individual component to find the one that is not working properly. The following sections provides general pointers and describe common problems.

Tracking causes

If a client is having NFS trouble, make sure the server is up and running. From a client, to see if the server is up, enter

```
# /etc/ping server_name
```

If the server responds, enter

```
# /usr/etc/rpcinfo -p server_name
```

The server should print a list of program, version, protocol, and port numbers that resembles the following output:

```
program vers proto   port
 100000    2   tcp    111  portmapper
 100000    2   udp    111  portmapper
 100005    1   udp    665  mountd
 100005    1   tcp    667  mountd
 100003    2   udp   2049  nfs
 100024    1   udp    686  status
 100024    1   tcp    688  status
 100021    1   tcp    689  nlockmgr
 100021    1   udp   1030  nlockmgr
 100021    3   tcp    693  nlockmgr
 100021    3   udp   1031  nlockmgr
 100020    1   udp   1032  llockmgr
 100020    1   tcp    698  llockmgr
 100021    2   tcp    701  nlockmgr
 100001    1   udp   1034  rstatd
 100001    2   udp   1034  rstatd
 100001    3   udp   1034  rstatd
```

Use `rpcinfo` to determine whether the *`mountd`* server is running by entering

```
# /usr/etc/rpcinfo -u server_name 100005 1
```

where `100005 1` comes from the `mountd` entry in the `rpcinfo` output above. The response should return as:

```
proc 100005 version 1 ready and waiting
```

If attempts to get the server to respond fail, log into the server's console and see if it is running.

- If the server is alive but your machine cannot reach it, check the hardware connections between your machine and the server.
- If the server is fine and the network is fine, enter `ps` to check your client daemons. `portmap` and several `biod` daemons should be running. If NIS is running, `ypbind` must also be running.

Entering

```
# ps -ef
```

at a client running NIS, results in output similar to that shown in the following example:

```
    32     ?     I     1:07    /etc/portmap
    38     ?     I     0:42    /etc/ypbind
    61     ?     S     0:45    (biod)
    62     ?     S     0:36    (biod)
    63     ?     S     0:30    (biod)
    64     ?     S     0:27    (biod)
```

The four subsections ahead discuss the most common types of failures. The first tells what to do if your remote mount fails; the next three describe servers not responding after you have file systems mounted.

Never mount or unmount a file system while one of its directories is exported. The proper sequence is to:

1. Unexport a directory, unmount the filesystem
2. Re-mount the same or a different filesystem
3. Re-export the directory

If you do not follow this procedure, it is possible to have a directory that `exportfs` claims is exported, but that clients cannot mount. Reexporting the directory corrects this inconsistency.

Handling failed remote mount operations

`mount` can get its parameters from either the command line or from the *fstab* file (refer to the mount(8) man page). The example below assumes command line arguments, but the same debugging techniques work if *fstab* is used in a `mount -a` command. The following `mount` command, entered on an NFS client,

```
# mount krypton:/usr/src /krypton.src
```

asks the server machine krypton to return a file handle for the directory /usr/src. This file handle is then passed to the kernel in the `mount` system call. The kernel looks up the directory /krypton.src and, if conditions are right, it ties the file handle to the directory in a mount record. From now on, all file system requests to that directory and below go through the file handle to the server krypton.

Mounting: failure types and their messages

The mount failure types listed here are arranged according to where they occur in the mounting sequence and are labeled with the accompanying error message.

- `mount: ... already mounted`

  The file system that you are trying to mount is already mounted or it has a bogus entry in /etc/mtab.

- `mount: ... Block device require`

  You probably omitted `krypton:` from the command line

  ```
  # mount krypton:/usr/src /krypton.src
  ```

  `mount` assumes you are doing a local mount unless it sees a colon in the file system name or the file system type is NFS in /etc/fstab.

- `mount: ... not found in /etc/fstab`

  If you see this message, it means the argument you gave `mount` was not in any of the entries in /etc/fstab. If `mount` is called with a directory or file-system name but not both, it looks in /etc/fstab for an entry whose file system or directory field matched the argument. For example:

  ```
  # mount /krypton.src
  ```

  searches /etc/fstab for a line that has a directory name field of /krypton.src. If it finds an entry, such as

  ```
  krypton:/usr/src /krypton.src nfs rw,hard 0 0
  ```

  it performs the mount as if you had typed

  ```
  # mount krypton:/usr/src /krypton.src
  ```

- `/etc/fstab: No such file or directory`

  `mount` tried to look up the name in /etc/fstab but there was no /etc/fstab.

  `... not in hosts database`

  This means NIS could not find the hostname you gave it in the /etc/hosts database or that `ypbind` is dead on your machine. (Note that if you are not running NIS, the hostname entry is in the local /etc/hosts file.) First check the spelling and the placement of the colon in your mount call. If those look right, make sure that `ypbind` is running by entering

  ```
  # ps -ef| grep ypbind
  ```

  Try `rlogin` or `rpc` to some other machine. If this also fails, your `ypbind` is probably dead or hung. If you get this message only for one host name, it means that the /etc/hosts entry on NIS server needs to be checked. See the section on debugging NIS later in this chapter.

- `mount: directory path must begin with '/'`

  The second argument to `mount` is the path of the directory to be covered. This must be an absolute path starting at "/".

- `mount: ... server not responding: RPC: Port mapper failure - RPC: Timed out`

  Either the server you are trying to `mount` from is down, or its `portmapper` is dead or hung. Try logging in to that machine. If you can log in, enter

  ```
  # rpcinfo -p
  ```

  to get a list of the registered programs. If you do not get a listing, restart `portmapper` on the server. Restarting `portmapper` requires that you kill and restart both `inetd` and `ypbind`.

  If you cannot `rlogin` to the server but the server is up, check your hardware connection by trying to `rlogin` to some other machine. Also check the server's hardware connection.

- `mount: ... server not responding: RPC: Program not registered`

  This means that mount got through to the `portmapper` but the NFS mount daemon (rpc.mountd) was not registered. Go to the server and be sure that /usr/etc/rpc.mountd exists and that there is an entry in /etc/inetd.conf exactly like the following example:

```
mount    dgrm    udp     wait    root    1       /usr/etc/mount  mountd
```

Finally, use `ps` to be sure that `inetd` is running. If you had to change /etc/inetd.conf, you must kill `inetd` and restart it.

- `mount: ...: No such file or directory`

  Either the remote directory or the local directory does not exist. Check spelling. Try to `ls` both directories.

- `mount: not in export list for ...`

  Your machine name is not in the export list for the file system you want to mount from the server. You can get a list of the server's exported file systems by entering

  `# showmount -e` *hostname*

  If the file system you want is not in the list, or your machine name or netgroup name is not in the user list for the file system, log in to the server and check the /etc/exports file for the correct file system entry. A file system name that appears in the /etc/exports file, but not in the output from `showmount`, indicates a failure in `mountd`. Either it could not parse that line in the file, or it could not find the file system, or the file system name was not a locally-mounted file system. See the exports(5) man page for more information. If exports seems okay, check the server's `ypbind` daemon: it may be dead or hung.

- `mount: ...: Permission denied`

  This message is a generic indication that some authentication failed on the server. It could simply be that you are not in the export list (see above), or the server could not figure out who you are (`ypbind` dead). Or, it could be that the server does not believe that you are who you say you are. Check the server's /etc/exports and ypbind. Here, you can change your hostname with hostname and retry the mount.

- `mount: ...: Not a directory`

  Either the remote path or the local path is not a directory. Check spelling and try to `ls` both directories.

- `mount: ...: Not owner`

  You have to do the mount as root on your machine because it affects the file system for the whole machine, not just you.

- `mount: ...: Cannot mount a directory on top of itself`

  Self-explanatory.

When the system hangs at startup

If your machine comes part way up after a boot, but hangs where it would normally be doing remote mounts, probably one or more servers is down or your network connection is bad. Refer to the two previous subsections.

To avoid a hung system, add the `bg` flag to the NFS partitions listed in /etc/fstab. For example, an entry would look like this:

```
krypton:/usr/man /usr/man nfs rw,soft,bg 0 0
```

When remote file access seems slow

If access to remote files seems unusually slow, enter

```
# ps -ef
```

on the server to be sure it is not being clobbered by a runaway daemon, bad tty line, etc. If the server seems okay and other people are getting good response, make sure your block I/O (`biod`) daemons are running; enter

```
# ps -ef
```

on the client and look for `biod`. If they are not running or are hung, you can find the process ID's by entering

```
# ps -ef| grep biod
```

and kill them with

```
# kill -9 pid1 pid2 pid3 pid4
```

Restart them with

```
# /etc/biod 4
```

To determine whether they are hung, enter `ps` as above, copy a large remote file, then enter `ps` again. If the `biod`s do not accumulate CPU time, they are probably hung.

If `biod` is okay, check your hardware connection. The command `netstat -i` tells you if you are dropping packets. Also, you can use `nfsstat -c` and `nfsstat -s` to lookup retransmission rates for the client or server. A retransmission rate of 5% is considered high. Excessive retransmission usually means a bad hardware board, a bad hardware tap, a mismatch between board and tap, or a mismatch between your hardware board and the server's board.

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
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).

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
TCP/IP protocol that maps internet addresses to physical addresses.

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.

Asynchronous Transfer Mode (ATM)
Asynchronous Transfer Mode (ATM) is the technique for transport, multiplexing, and switching that provides a high degree of flexibility required by B-ISDN. ATM is a connection-oriented protocol that employs fixed-size packets with a 5-byte header and 48 bytes of information.

B

backbone
A central network used to interconnect LANs.

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

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
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.

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

C

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.

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.

(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.

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.*

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 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.

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.

CUD
Network service parameter carried with a call request. This parameter allows data to be sent with the call request.

D

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

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.

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
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.

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
Digital Equipment Corporation's proprietary network architecture.

DNIC
Part of a DTE address that identifies a specific data network.

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 (BIND).

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 (BIND).

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
The link service access point of the destination link service user.

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

E

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

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.

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
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.

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
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.

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
An ISO application-level service that allows users to transfer and manipulate files between hosts.

FTP
TCP/IP protocol that allows users to transfer files between hosts.

G

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

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

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
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.

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
TCP/IP protocol responsible for relaying error messages detected by gateways back to the source node.

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.

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.

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 (ISO).*

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

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.

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.

ISO reference 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.

J

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
A network that serves several users at relatively high speeds within a small geographic area.

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 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
Data link layer protocols for operation over a LAN.

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
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.*

LSAP
An address or identifier through which a service user accesses data link layer services.

M

MAC
Component of the data link layer that controls access to the physical medium.

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.

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 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.

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

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.

N

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 (BIND), Domain Name System (DNS), and Network Information Service (NIS).

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).

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 File System (NFS)
A system that provides transparent access to networked files over a TCP/IP-based network. NFS links together heterogeneous sys-

tems 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 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.

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.

NIC
The central authority responsible for assigning and maintaining addresses of networks and hosts on the DARPA Internet.

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.

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

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

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

NSU
A software entity that uses the services of the network layer.

O

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
The ISO definition of a system that provides reliable, data-transparent, host-independent services.

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.

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

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

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

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

peer transport users
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*.

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
A network layer protocol used for connection-oriented operation.

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.

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.

R

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

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.

RFC
A set of notes and technical papers that discuss various subjects related to TCP/IP protocols, including proposed and accepted standards.

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

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

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
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.

S

SAP
An address or identifier through which a service user can communicate with a service provider.

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
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.

SMTP
TCP/IP protocol used to relay electronic mail messages between networked machines.

SNA
Proprietary network architecture developed by IBM to connect IBM computers. Convex computers can be interconnected with SNA networks through the gateway product, *OpenConnect*.

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.

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.

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

SSAP
Link service access point of the source link service user.

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

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.

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*.

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.

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

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.

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.

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.

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
An OSI software entity that uses the services of one or more transport providers.

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
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.

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.

U

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

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.

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
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.

W

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.*

Y

Yellow Pages (YP)
The Yellow Pages (YP), or 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.

Index

Symbols

A

B

C

D

E

F

G

H

L

M

N

O

P

R

S

T

U

V

W

X

Y

ORDER NUMBER
DSW-865

DOCUMENT NUMBER
710-031130-003