Data General
Data General Corporation, Westboro, Massachusetts 01580

Customer Documentation

Managing TCP/IP on the DG/UX™ System

093–701051–06

AVIION®
PRODUCT LINE

Managing TCP/IP on the DG/UX™ System

093–701051–06

For the latest enhancements, cautions, documentation changes, and other information on this product, please see the Release Notice (085–series) and/or Update Notice (078–series) supplied with the software.

Printed in the United States of America
Rev. 06, December 1993

Ordering No. 093–701051

Notice

Data General Corporation
4400 Computer Drive
Westboro, MA 01580

Managing TCP/IP on the DG/UX™ System

093–701051–06

Revision History:

Original Release – April, 1989
First Revision – June 1989
Second Revision – November, 1989
Third Revision – May, 1990
Fourth Revision – June, 1991
Fifth Revision – August, 1992
Sixth Revision – December 1993

Effective with:

TCP/IP for AViiON Systems, 5.4 Release 3.00

A vertical bar in the margin of a page indicates substantive technical change from the previous revision. (The exceptions are Chapters 2, 3, 7 and Appendix A, which contain entirely new or newly presented material.)

About this manual

This manual describes how to set up and maintain the TCP/IP network on AViiON® systems.

How this manual is organized

Chapter 1	Introduces the basic terms of networking and describes the components of the package.
Chapter 2	Introduces the major administrative areas. Describes network topologies, network and host addressing, subnetting, and routing.
Chapter 3	Explains how to perform TCP/IP administrative procedures from **sysadm**, and describes some additional shell–based administrative procedures.
Chapter 4	Tells how to configure and use **sendmail**.
Chapter 5	Tells how to configure and use the Domain Name System (DNS).
Chapter 6	Describes how to configure and use the Simple Network Management Protocol (SNMP).
Chapter 7	Describes how to configure and use the Serial Line Internet Protocol (SLIP).
Chapter 8	Suggests how to troubleshoot on a network running TCP/IP.
Appendix A	Describes the SNMP Management Information Base (MIB) objects maintained by Data General.
Glossary	Defines some networking terminology used in this manual.

Related Data General manuals

Within this manual, we refer to the following manuals:

- *Managing the DG/UX System* (093–701088). The basic source for DG/UX administrative procedures.
- *Installing the DG/UX System* (093–701087). Explains how to install DG/UX, including TCP/IP, from the release media.
- *Using TCP/IP on the DG/UX System* (093–701023). Explains the standard TCP/IP applications for remote login and file transfer.

- *Programming with TCP/IP on the DG/UX System* (093–701024). Explains how to write TCP/IP client/server applications.
- *Managing Modems and UUCP on the DG/UX System* (069–000698). Explains how to set up modems and unix–to–unix copy (UUCP) software.

For a complete list of DG/UX and networking manuals available from Data General, see *Guide to AViiON and DG/UX System Documentation* (069–701085).

Other related documents

Listed below are some related documents which are not available from Data General Corporation.

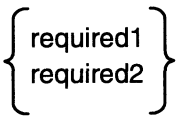
- Birrell, Andrew D., and Nelson, Bruce Jay. *Implementing Remote Procedure Calls*. XEROX CSL–83–7, October 1983.
- *CCITT Recommendation X.25*. You can obtain copies of this document from the National Technical Information Service (NTIS), 5285 Port Royal Road, Springfield, VA, 22161.
- Hunt, Craig. *TCP/IP Network Administration*. O'Reilly & Associates, Inc., 103 Morris Street, Suite A, Sebastopol, CA 95742. This excellent book is available to you at a discount. See *UNIX Books Order Form* (069–100486), which comes with DG/UX.
- Request for Comments (RFCs). You can get copies from InterNIC Information Services, P.O. Box 85608, San Diego, CA, 94186–9784, 1–800–444–4345. Selected titles are listed below.
  - *RFC 768 (User Datagram Protocol)*
  - *RFC 791 (Internet Protocol)*
  - *RFC 792 (Internet Control Message Protocol)*
  - *RFC 793 (Transmission Control Protocol)*
  - *RFC 821 (Simple Mail Transfer Protocol)*
  - *RFC 822 (Standard for the Format of ARPA Internet Text Messages)*
  - *RFC 826 (An Ethernet Address Resolution Protocol)*
  - *RFC 861 (DCN Local–Network Protocols)*
  - *RFC 950 (Internet Standard Subnetting Procedures)*
  - *RFC 1055 (A Nonstandard for Transmission of IP Datagrams over Serial Lines)*

- *RFC 1058 (Routing Information Protocol)*
- *RFC 1144 (Compressing TCP/IP Headers for Low–Speed Serial Links)*
- *RFC 1247 (OSPF Version 2)*

Reader, please note:

Throughout this manual we use the following format conventions:

COMMAND required **required** [*optional*] ...

Where	Means
COMMAND	You must enter the command (or its accepted abbreviation) as shown.
required	You must enter some argument (such as a filename). Sometimes, we use
	which means you must enter one of the arguments. Do not type the braces; they only set off the choices.
required	You must enter the case–sensitive characters as shown.
[*optional*]	You have the option of entering this argument. Do not type the brackets; they only identify the argument as an option.
...	You may repeat the preceding entry.

Additionally, we use certain symbols in command lines.

Symbol	Means
↵	Press the New Line, Carriage Return (CR), or Enter key on your terminal keyboard.
<CTRL–D>	Hold the Control key down and press the D key on your terminal keyboard.
)	The CLI prompt.
%	The UNIX® C shell prompt.
$	The UNIX® Bourne shell prompt.

Finally, in examples we use:

This typeface to show your entry.
This typeface to show system queries and responses.
`This typeface to show file contents.`

Contacting Data General

Data General wants to assist you in any way it can to help you use its products. Please feel free to contact the company as outlined below.

Manuals

If you require additional manuals, please use the enclosed TIPS order form (United States only) or contact your local Data General sales representative.

Telephone assistance

If you are unable to solve a problem using any manual you received with your system, free telephone assistance is available with your hardware warranty and with most Data General software service options. If you are within the United States or Canada, contact the Data General Customer Support Center (CSC) by calling 1–800–DG–HELPS. Lines are open from 8:00 a.m. to 5:00 p.m., your time, Monday through Friday. The center will put you in touch with a member of Data General's telephone assistance staff who can answer your questions.

For telephone assistance outside the United States or Canada, ask your Data General sales representative for the appropriate telephone number.

Joining our users group

Please consider joining the largest independent organization of Data General users, the North American Data General Users Group (NADGUG). In addition to making valuable contacts, members receive *FOCUS* monthly magazine, a conference discount, access to the Software Library and Electronic Bulletin Board, an annual Member Directory, Regional and Special Interest Groups, and much more. For more information about membership in the North American Data General Users Group, call 1–800–253–3902 or 1–508–443–3330.

End of Preface

Contents

Chapter 1 – Introduction to TCP/IP

Chapter 2 – Introduction to Network Administration

Chapter 3 – Administering a DG/UX TCP/IP Network

Chapter 4 – Configuring and Using sendmail

Chapter 5 – Configuring and Using the Domain Name System

Chapter 6 – Configuring and Using the Simple Network Management Protocol

Chapter 7 – Configuring and Using the Serial Line Internet Protocol (SLIP)

Chapter 8 – Troubleshooting

Appendix A – SNMP MIB Objects

Glossary

Index

Tables

Table

Figures

Figure

1 Introduction to TCP/IP

This chapter defines some basic terms of networking and describes TCP/IP's components. For more thorough coverage of the terms used in this manual, refer to the Glossary.

Reviewing Basic Terms

Before covering the components of the TCP/IP package, it may be useful to review some basic terms. If you do not need such a review, skip ahead to the next section.

TCP/IP stands for Transmission Control Protocol and Internet Protocol. These two protocols are described at length later in this chapter.

A *network* enables two or more computer systems to communicate. A network includes the hardware and software that make up the interconnections between computer systems and between a computer and its peripheral devices. Network communications between computers takes place over media such as coaxial cable, fiber optic cable, twisted-pair cable, or microwave.

A *host* is a computer system to which a graphics screen or a number of terminals or other smaller computers are connected, and which provides computation, access to files, and other services. A *local host* is one to which you are directly connected. A *remote host* is one you access through a network.

A *local area network* (LAN) is a network within a small area, such as within a building. A *wide area network* (WAN) is a network of hosts that are far apart. A WAN usually requires connections through public communications facilities (such as the telephone company). The *Internet* network is a collection of local networks and gateways that use TCP/IP to function as a wide area network (WAN).

Networks can be complex. To help simplify them, designers organize networks into layers. Nearly every network system has its layers set up hierarchically. The number of layers and each layer's function varies from network to network. In all networks, though, each layer provides services to the higher layers, without the higher layers knowing the details of how the services are provided. An *interface* consists of the types and forms of messages that each layer uses to communicate with the layer above or below it. A *protocol* specifies how programs on different computers but at the same layer communicate. The set of layers, interfaces, and protocols that govern communication is called the network's architecture.

Applications that run over a network operate on a simple principle: a *process* (program in execution) at one layer on one host converses with a process at the same layer on another host. The client/server model provides a way for two communicating processes to relate to one another. The model describes how connections are initiated and how communicating parties interact. An idea central to the client/server model is that services are desired and available on the network. *Server* programs provide these services and *client* programs use them.

An *OS server* is a host that provides disk space for operating system software over a network. An *OS client* is a host that gets its system files from a disk that is physically connected to an OS server. A *release* is a set of software intended for a specific machine architecture and version of the operating system.

To run a network made up of an OS server and an OS client, first install the DG/UX system, then set up TCP/IP, and then set up DG/UX ONC™/NFS® on the OS server for itself and for its clients. For information about how to install the DG/UX system, which includes TCP/IP for AViiON Systems, see *Installing the DG/UX™ System.*

What Is TCP/IP for AViiON Systems?

TCP/IP for AViiON® Systems is a package of communications software that implements the TCP/IP family of networking protocols on the DG/UX operating system. The package consists of several kernel-level protocols, servers, administrative utilities, user commands, and user-level protocols.

The Defense Advanced Research Projects Agency (DARPA) developed the Internet protocols for the ARPANET network project. The University of California at Berkeley developed the 4.2 Berkeley Software Distribution (BSD) release of UNIX® based on the DARPA work. Other releases followed. Data General developed the TCP/IP for AViiON Systems software package from the BSD 4.3 release, substantially revising it to comply with the Defense Data Network (DDN) specifications. Many BSD 4.3 Tahoe and Reno features have been added to the TCP/IP for AViiON Systems package.

Figure 1–1 shows the TCP/IP network architecture.

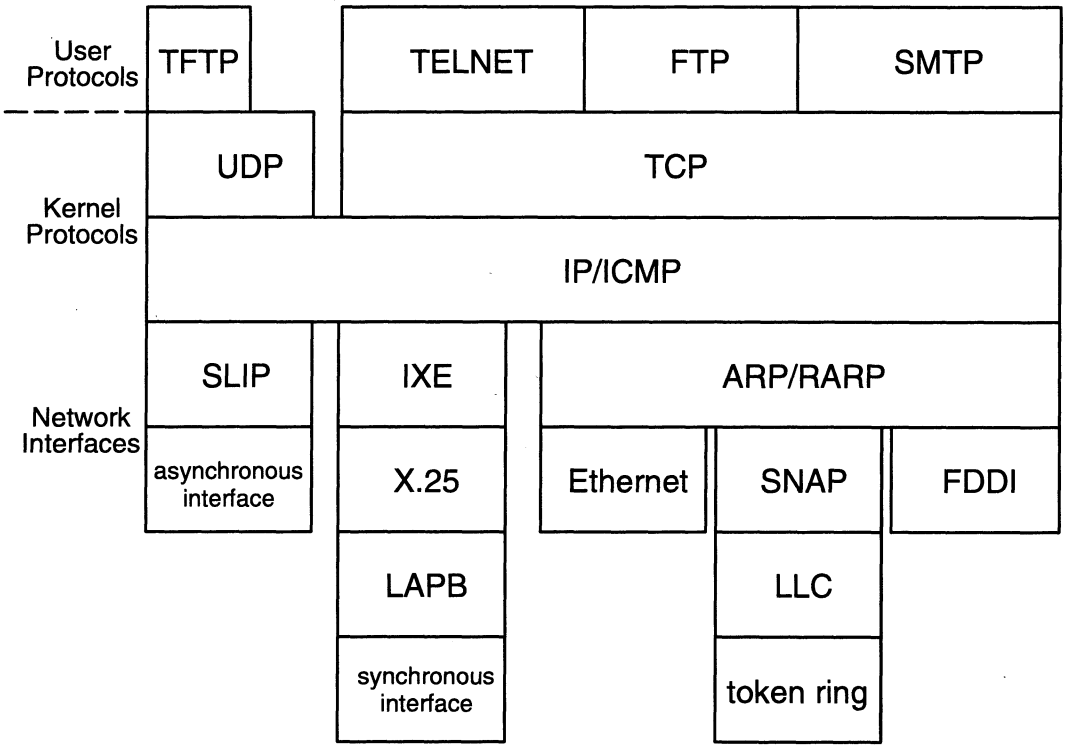


Figure 1–1 TCP/IP for AViiON Systems Network Architecture

At the lowest layer are network interfaces. They prepare IP traffic for transmission onto physical media and receive traffic from the media to deliver it to IP. Network interfaces pass information to a device driver, which is kernel-level software that manages the communications hardware installed in the computer. TCP/IP for AViiON Systems supports network interfaces for these media: IEEE 802.3/Ethernet, IEEE 802.5/token ring, ANSI X3T9.5 Fiber Distributed Data Interface (FDDI), and IXE (IP to X.25 Encapsulation) for use on X.25 networks (synchronous lines). The IXE interface lets you use one or more X.25 networks for TCP/IP traffic.

To prepare traffic for transmission, a network interface translates an IP address into an address that can be used on the underlying physical medium. Typically, the IP datagram is then encapsulated into a medium-specific frame and sent to the communications device for delivery to the physical network.

When receiving traffic, the network interface software accepts frames from the communications device and strips any network or medium-specific information from them. What remains is an IP packet, which is then delivered to IP.

Figure 1-1 shows five scenarios of TCP/IP networking interfaces. The first shows the Serial Line Internet Protocol (SLIP), which allows TCP/IP to be used through an asynchronous (tty) port, either over a direct line or via modem.

The second scenario shows IXE working with X.25 software and a synchronous device driver. For information about the X.25 for AViiON® Systems product, see *Setting Up and Managing X.25 on the DG/UX™ System*.

The third scenario shows the Address Resolution Protocol (ARP) and the Reverse Address Resolution Protocol (RARP) acting in conjunction with an Ethernet device driver (for example, **inen**, **dgen**, or **cien**). ARP and RARP provide a way to map hardware (Ethernet, token ring, or FDDI) addresses to Internet addresses.

The fourth scenario shows IP running over an IEEE 802 network. IEEE 802 defines LAN standards for the physical and data link layers. These layers are specified by the International Organization for Standardization (ISO) Open Systems Interconnection (OSI) model. Specifically, this scenario shows ARP and RARP translating Internet addresses to IEEE 802 addresses. SNAP, which is part of the 802.1 layer, appears below ARP and RARP. It smooths the communication between ARP/RARP and the lower 802 layers. Logical Link Control (LLC) (the 802.2 layer) lets software at higher-level layers of a network architecture use LAN services without regard to the specific LAN implementation (such as 802.3, 802.4, or 802.5). IEEE 802.5 defines a standard for token ring. Here, the token ring is accessed through the VME Token Ring Controller (VTRC), which uses the **vitr**(7) device driver.

The fifth scenario shows ARP and RARP acting in conjunction with a FDDI device driver. ANSI X3T9.5 defines FDDI LAN standards for the physical and data link layers of the OSI model. Here, the FDDI LAN is accessed through the VME FDDI Controller (VFC), which uses the **pefn**(7) device driver.

Next come the kernel-level protocols. First there is the Internet Protocol (IP), and the Internet Control Message Protocol, (ICMP). These protocols provide for connectionless delivery, which means that data is transferred in well defined bundles called packets, and each packet is treated independently of all the others. IP routes packets from one host to another. ICMP handles error and control messages. For more information about these protocols, see *RFC 791 (Internet Protocol)* and *RFC 792 (Internet Control Message Protocol)*.

At the next layer up, still at the kernel level, are two transport protocols, the Transmission Control Protocol, or TCP, and the User Datagram Protocol, or UDP. Transport protocols provide the mechanisms to allow user processes to communicate. TCP lets a process on one system send a stream of data to a process on another. UDP lets a process on one system send a datagram to a process on another. For more information about these protocols, see *RFC 768 (User Datagram Protocol)* and *RFC 793 (Transmission Control Protocol)*.

Next come four user-level protocols: TFTP, TELNET, FTP, and SMTP. They are user-level because their operation executes code in user space. These protocols provide virtual terminal service, file transfer service, and electronic mail service between systems. For more information about these protocols, see *RFC 783 (Trivial File Transfer Protocol)*, *RFC 854 (Telnet Protocol)*, *RFC 959 (File Transfer Protocol)*, and *RFC 821 (Simple Mail Transfer Protocol)*.

You can access TFTP, TELNET, and FTP through network programs of the same name. For more information about these programs, see *Using TCP/IP on the DG/UX™ System*. The **sendmail** program implements the Simple Mail Transfer Protocol (SMTP), which lets mail messages be transmitted. For more information about the **sendmail** program, see Chapter 4.

Programmers can create applications for use on an Internet network using the socket interface to TCP, UDP, or IP or the Transport Layer Interface (TLI) to TCP and UDP. For more information about how to do this, see *Programming with TCP/IP on the DG/UX™ System*.

Kernel-Level Protocols

TCP/IP for AViiON Systems contains the following kernel-level protocols:

ARP — Address Resolution Protocol

Used to associate an Internet address with a hardware (Ethernet, token ring, or FDDI) address. ARP runs only across a single physical network, and runs only over networks that support hardware broadcast. For more information about this protocol, see *RFC 826 (An Ethernet Address Resolution Protocol)*.

RARP — Reverse Address Resolution Protocol

Used by a diskless system at startup to find its Internet address. A diskless client broadcasts a request that contains its Ethernet address, and the server responds by sending the client's Internet address to that Ethernet address. For more information about the protocol, see *RFC 903 (A Reverse Address Resolution Protocol)*.

IP — Internet Protocol

A protocol that provides connectionless delivery of datagrams between hosts. Connectionless service means that the protocol treats each datagram as a separate entity. Each IP datagram contains the addresses of its source and destination, some control information, and the data transmitted. The protocol can deliver packets out of sequence, may drop packets, or may duplicate packets, but IP makes an earnest attempt to deliver packets. IP defines the exact format of data as it travels through a network, but delivery of data is not guaranteed.

ICMP — Internet Control Message Protocol

A partner to IP that handles error and control messages. Gateways and hosts use ICMP to tell the other hosts about problems in delivering the datagrams. ICMP also lets a host test whether a destination can be reached and whether it is responding.

TCP — Transmission Control Protocol

A protocol that defines reliable, stream-oriented, process-to-process communication. TCP is a connection-based protocol; it requires a connection between communicating hosts before it transmits data. After a connection is established, TCP provides a two-way byte stream between communicating processes. Its messages include a protocol port number that lets the sender distinguish between multiple programs on the remote host. TCP provides a checksum mechanism to guarantee that data has arrived intact. TCP uses IP to transmit information across a network.

UDP — User Datagram Protocol

A protocol that defines datagram-based communication between a process on one host and a process on another host. UDP is a connectionless transport protocol. Its messages include a protocol port number that lets the sender distinguish between multiple programs on the remote host. Data General's UDP uses a checksum mechanism to guarantee that data has arrived intact. UDP uses IP to transmit information across a network.

Servers

TCP/IP contains the following server programs (also called daemons or agents). Each server operates at a specified port and provides service for a user protocol. You specify the port and services in **/etc/services** and **/etc/inetd.conf**. You can enable and disable servers through **sysadm**.

gated

The **gated**(1M) program provides dynamic routing services. It supports multiple routing protocols, including Routing Information Protocol (RIP), HELLO, and Open Shortest Path First (OSPF).

inetd

The **inetd**(1M) server invokes network servers on demand. It also provides simple TCP-based services of its own. The following daemons are started by **inetd**.

The **ftpd**(1M) program, which is the File Transfer Protocol (FTP) server, is invoked by **inetd** when an incoming connection is detected on the specified port.

The **telnetd**(1M) program, which is the TELNET server, is invoked by **inetd** when an incoming connection is detected on the specified port. TELNET is described later in this chapter.

The **tftpd**(1M) program, which is the Trivial File Transfer Protocol (TFTP) server, is invoked by **inetd** when an incoming connection is detected on the specified port. TFTP is described later in this chapter.

rshd(1M), **rexecd**(1M), **rlogind**(1M) are servers for **rsh** (**remsh** if you comply with the *System V Interface Definition*), **rexec**, and **rlogin**. They are invoked by **inetd** when an incoming connection is detected on the specified port.

The **bootpd**(1M) server allows a diskless system (OS client) that supports the boot protocol (BOOTP) to find out information about its server (OS server). AViiON servers provide BOOTP service to clents that request it. AViiON clients do not use BOOTP.

named

The **named** server provides DNS services. The process listens on a specified port for queries from a domain name resolver or from another name server. It maintains a database that contains information about specified objects.

pmtd

pmtd(1M) is a server for the tape pseudo device. This server handles local I/O requests to tape devices on a remote host.

routed

The **routed**(1M) program provides dynamic routing using the Routing Information Protocol (RIP). On systems that support **gated**(1M), we recommend it instead of **routed**.

rwhod

rwhod(1M) is the server for **rwho** and **ruptime**.

snmpd

The **snmpd**(1M) server implements the Simple Network Management Protocol (SNMP).

slipd

The **slipd**(1M) program manages the Serial Line Internet Protocol (SLIP), allowing TCP/IP connections over serial lines or over telephone lines via modem.

xntpd

The **xntpd**(1M) program implements the Network Time Protocol (NTP), which synchronizes the time on TCP/IP hosts.

Administrative Utilities

TCP/IP contains the following administrative utilities:

arp

Use the **arp**(1M) command to examine and change kernel ARP tables. (See Address Resolution Protocol (ARP) under the section "Kernel-Level Protocols" earlier in this chapter.)

hostid

The superuser can use the **hostid**(1C) command to set the hostid. Anyone can use the command to display the current hostid in hexadecimal.

hostname

The superuser can use the **hostname**(1C) command to set the hostname. Anyone can use the command to display the current hostname.

ifconfig

The **ifconfig**(1M) command assigns an address to a network interface, configures the network interface parameters, and stops and restarts an interface.

initrarp

The **initrarp**(1M) command initializes an OS server's ARP table. ARP and RARP use the ARP table to maintain Ethernet-to-Internet address translation information for all diskless clients.

netinit

The **netinit**(1M) command builds a protocol stack and logically attaches a network interface to a protocol implementation.

netstat

The **netstat**(1C) command displays the contents of various data structures related to network activity. For example, you could use **netstat** to display the state of all sockets, to show the TCP/IP routing tables, or to display information about communication interfaces.

nslookup

The **nslookup** command lets you query domain name servers directly.

NTP commands

The following commands query or modify the Network Time Protocol (NTP) daemon.

ntpdate(1M) allows the superuser to synchronize the date and time of the local host with that of the specified NTP server.

ntpq(1M) supports a set of more than forty commands for querying specified NTP hosts about the time–setting parameters in effect and the variances allowed.

xntpdc(1M) supports a set of more than fifty commands allowing the superuser to query or modify the **xntpd** process on a specified NTP server.

ping

The **ping**(1M) command reports whether or not a specified network host (or other node) is up and working.

Routing commands

The **ospf_monitor**(1M) command, on networks that use Open Shortest Path First (OSPF), returns information about how this dynamic routing protocol is configured.

The **ripquery**(1M) command, useful for isolating problems on networks that use the Routing Information Protocol (RIP) for dynamic routing, displays all the routes known to a specified gateway.

The **route**(1M) command maintains static routes in a host's routing table, allowing it to reach hosts on different networks.

SNMP commands

The following commands query or set Management Information Base (MIB) objects via the SNMP daemon.

The **snmpgetmany** command retrieves a specified Management Information Base (MIB) object or subtree.

The **snmpgettable** command retrieves the objects comprising a row of a specified MIB table.

The **snmpgetnext** command retrieves a set of MIB objects or object instances.

The **snmpgetone** command retrieves a specified object instance.

The **snmpsetany** command gets the current values of the MIB object instances to be set and then sets the instances to the specified value.

The **snmptraprecv** command allows the superuser to receive SNMP traps from network elements that generate them.

The **snmptrapsend** command allows the superuser to send SNMP traps to stations that monitor for them.

sysconfig

The **sysconfig** command lets you customize your TCP/IP environment dynamically. You can set the maximum hop count an IP packet can travel, control forwarding of IP packets and IP source-routed packets through the local machine, set the time TCP waits before sending keep-alive probes, and determine the ARP cache time-outs.

traceroute

traceroute(1M) prints the route that packets take to their destination. It is useful for isolating problems on the network.

tcpdump and nfc

tcpdump(1M) prints network packets. You can print packets of a certain protocol (for example, only IP packets), arbitrary fields of a packet, packets between specified hosts, or packets longer than a specified length. **nfc**(1M) converts a tcpdump binary file to Network General sniffer binary format, or vice versa.

User Commands and User-Level Protocols

TCP/IP contains the following user commands and user-level protocols:

ftp

The **ftp** command implements the File Transfer Protocol (FTP). FTP lets you transfer files from one host to another. FTP uses TCP as the transport level protocol.

R commands

The R commands let you obtain information from, log in to, and execute commands on a remote host. Some R commands use TCP as the transport level protocol, and some use UDP. TCP/IP includes the following R commands:

rcp, which lets you copy files between systems on the network.

rlogin,which lets you log in to another system over the network.

rsh (**remsh**), which connects to a specified host and executes a specified command. (Optionally, for SVID compatibility, you can modify your system with **sysadm** such that **rsh** runs **restsh**, the restricted Bourne shell.)

rwho, which lists all users logged in to all systems on the local network, as long as the systems are running **rwhod**.

ruptime, which shows the status of each machine that is on the local network and running **rwhod**.

sendmail

The **sendmail** command implements the Simple Mail Transfer Protocol (SMTP), which supports electronic mail (e–mail) over the Transmission Control Protocol (TCP). Chapter 4 discusses how to configure and use **sendmail**.

telnet

The **telnet** command implements the TELNET protocol. TELNET lets a user on one host interact with a remote host as if the terminal is directly connected to the remote host. TELNET uses TCP as the transport level protocol.

tftp

The **tftp** command implements the Trivial File Transfer Protocol (TFTP). TFTP transfers files with minimal capability and overhead. The **tftp** command depends on the UDP protocol, which was discussed earlier in this chapter.

TFTP is also used during a first-stage boot with Data General's AViiON stations. The boot program first determines its Internet address and then uses TFTP to transfer a file that contains the executable image of a second-stage boot program. These topics are covered at length in later chapters of this book.

bftp

The **bftp** command provides the user interface to the Background File Transfer Program, which lets you transfer files in the background. For more information about BFTP, see *Using TCP/IP on the DG/UX™ System*.

End of Chapter

2 Introduction to Network Administration

This chapter describes network topologies, addressing, subnetting, and routing. If you are familiar with these topics, you can skim or skip it.

When you purchase a computer system that will function on a network, you purchase also one or more controllers (interface devices) for each type of network the system will be connected to. A number of network layouts and controllers are supported by DG/UX TCP/IP. These are described in the section "Network Layouts and Types."

Networks can be connected to one another in various ways. This topic is discussed in "Connecting Networks."

Every system on a TCP/IP network has an internet address based on the internet address of the network itself. Assignment of internet addresses and names to a system's network interfaces is discussed in the section "Assigning Internet Addresses."

A TCP/IP network can be a single physical and logical unit or it may be a collection of connected units called subnets. Similarly, a network may be isolated (consist entirely of systems at one location), connected to remote branches, or connected to the outside world. These issues are discussed in the section "Using Subnets."

Where a network is made up of interconnected subnets, or is connected to the outside world, access paths between the networks must be defined. This topic is discussed in the section "Routing."

Network Layouts and Types

Networks can have different physical layouts:

- *Star networks* connect each host to a central host.
- *Bus networks* provide a single connective link between hosts; any of the hosts can transmit to any other, but only one host can transmit at a time.
- *Ring networks* connect each system to the next in a closed ring.

Ethernet is a specific type of local area network technology originally developed by the Xerox Corporation. Networks that use Ethernet technology usually use a bus layout.

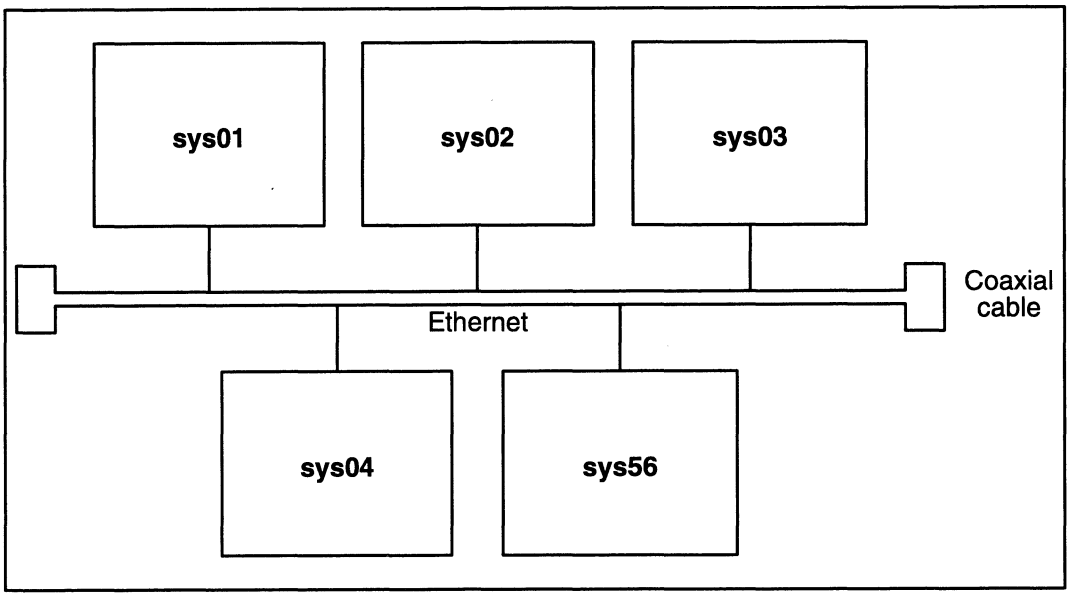


Figure 2–1 A Bus Network

An Ethernet segment is no longer than 500 meters, depending on the medium used. You can extend an Ethernet network with devices called repeaters. A *repeater* is a box that copies each bit of a packet from one segment of a network to another. You can think of a repeater as an amplifier, boosting signals that come through the physical medium. A *bridge* is essentially a repeater, except that typically it contains additional logic to filter traffic that is appropriate for the physical segments that it links. Figure 2–2 shows hosts connected by bridges or repeaters.

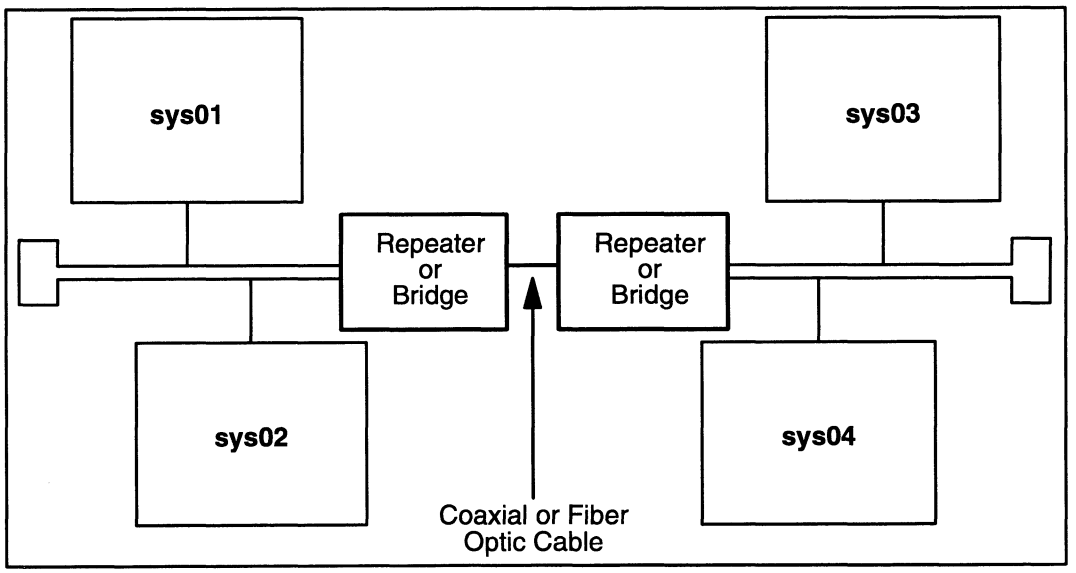


Figure 2–2 Hosts Connected by Bridges or Repeaters

Ethernet networks and controllers for AViiON systems are described in *Ethernet/IEEE 802.3 Local Area Network Installation*

Guide. DG/UX device drivers for currently available Ethernet controllers include **cien** (7), **dgen**(7), and **inen**(7).

Token ring is another type of local area network technology. On token ring networks, a small message called a token circulates around the ring. Put most simply, a host on the network may transmit on the network only when it possesses the token. Figure 2–3 shows an example of a ring network using token ring technology. This ring network uses twisted-pair cable (telephone wire) and supports six hosts.

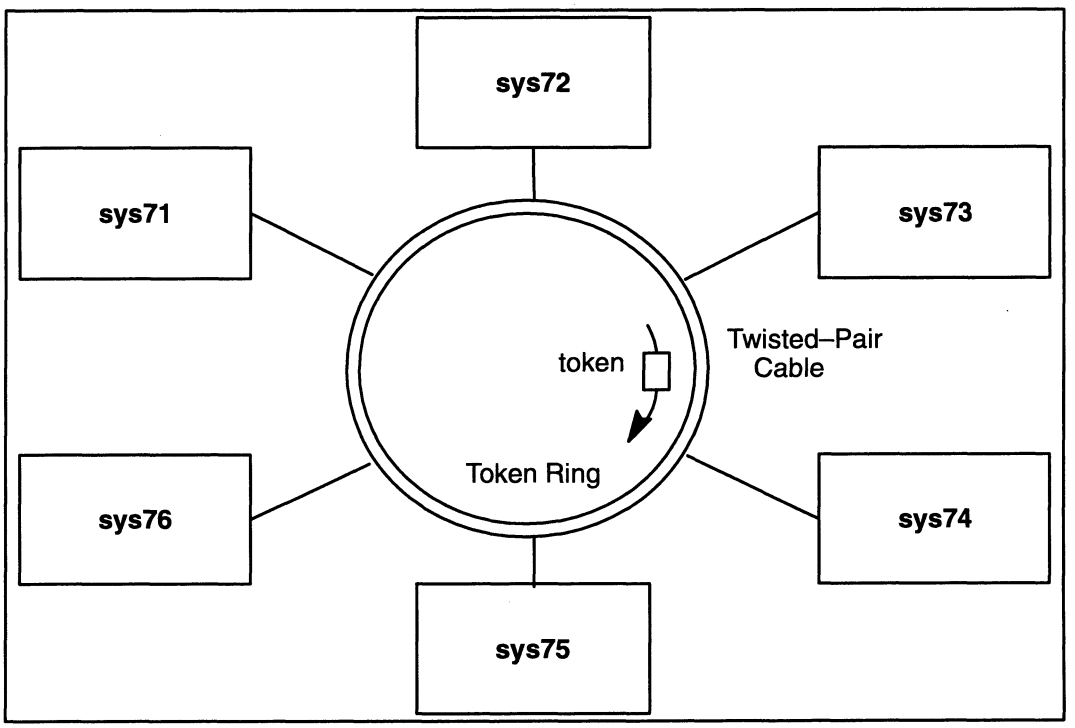


Figure 2–3 A Token Ring Network

Token ring network controllers that use twisted–pair cabling are described in *DG/Token Ring Local Area Network Installation Guide*. DG/UX device drivers for currently available AViiON token ring controllers include **vitr**(7).

Fiber Distributed Data Interface (*FDDI*) is a third type of local area network technology. FDDI-based networks usually use a ring layout. Unlike token ring networks, FDDI networks optimally have two rings operating at the same time. Each ring has its own token with one token moving clockwise and the other moving counterclockwise. This type of LAN set up is called a *dual counter-rotating ring*. Should one of the rings fail, the other ring ensures that the network stays in operation. If both rings fail in one or more places, the network divides into smaller rings that still function, but are isolated from each other. Figure 2–4 shows a FDDI network that uses fiber optic cable and supports six hosts.

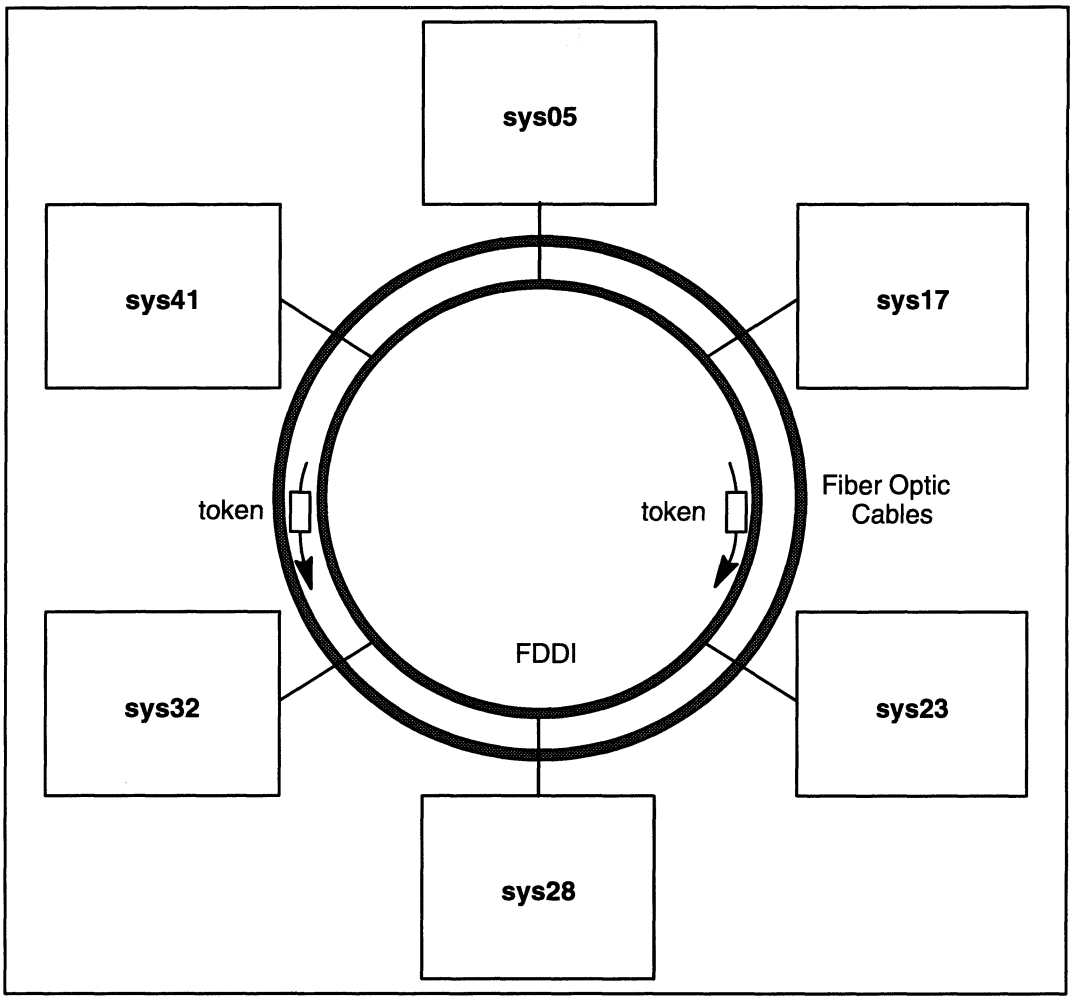


Figure 2–4 A FDDI Network

FDDI networks and controllers are described in *V/FDDI (4211 Peregrine) User's Guide*. DG/UX device drivers for currently available FDDI controllers for AViiON include **pefn**(7).

Connecting Networks

The bus network and the ring network are each examples of a single *physical network*. A *logical network* may consist of one or more physical networks or may be a subdivision of a single physical network. You set up a logical network through the addressing scheme you use, as explained in the following sections.

While a bridge or repeater extends a physical network, a *router* connects two physical networks. Specifically, a router is a computer or special equipment that forwards packets of the *same* protocol (for example, IP) from one network to another. Networks connected by routers may use different transmission media. A computer can act as a router as long as it has more than one communication

tcontroller and contains routing tables and software prepared to forward packets. The section "Routing" discusses the setting up of access paths between networks or subnets.

Figure 2–5 shows how TCP/IP-based networks can communicate with each other through a router.

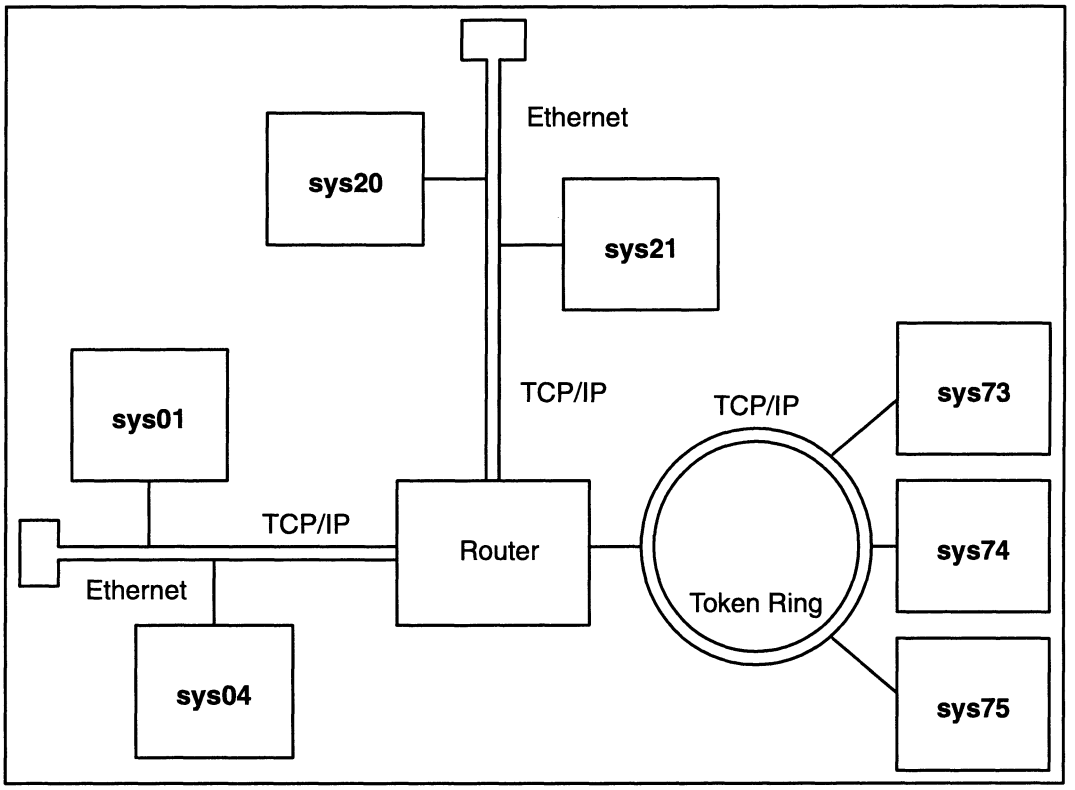


Figure 2–5 TCP/IP Networks Connected by a Router

This raises an important point: TCP/IP software handles several communication functions, such as routing, but it depends on Ethernet, token ring, or FDDI technology to perform other functions, such as the transmission of bits across a physical medium. Specific interfaces prepare TCP/IP traffic for transmission onto Ethernet, token ring, or FDDI media and receive traffic from the media to deliver it to TCP/IP.

To run TCP/IP over X.25 for AViiON® Systems, you must configure an interface that associates a TCP/IP network with the X.25 protocol stack. This interface runs over IXE, the IP to X.25 encapsulation driver. You must configure IXE interface information in both TCP/IP and X.25. For information about how to configure this interface for X.25, see *Setting Up and Managing X.25 on the DG/UX™ System*.

Once you set up an IXE interface on a host on your local TCP/IP-based network, any host on that network can communicate with a host on a distant TCP/IP-based network through an X.25

connection. The X.25 transmission takes place over synchronous communication lines.

Figure 2–6 shows two TCP/IP networks communicating through routers over an X.25 connection.

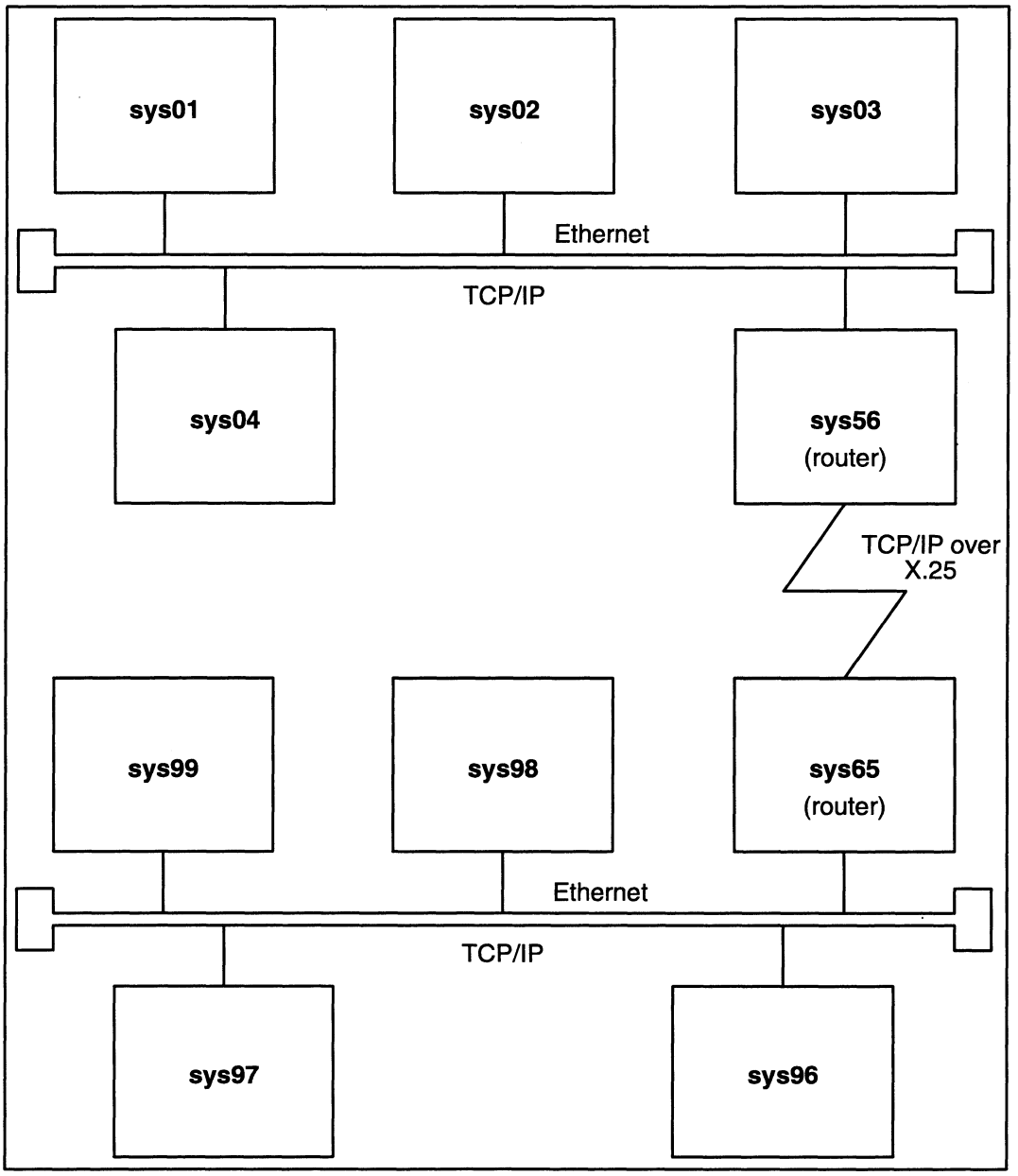


Figure 2–6 TCP/IP Networks Communicating Through an X.25 Connection

TCP/IP-based networks can also communicate with each other through *gateways*. In the most general sense, a gateway, like a router, is a system connected to two different networks. But in contrast to a router, a gateway converts from one protocol to another. For example, a gateway could convert from X.400 to SMTP and vice versa.

Assigning Internet Addresses

TCP/IP networks and the computers (or other devices) on them must have internet (IP) addresses. In function, the network address is analogous to the city/state/zip-code line of a postal address, and the host IP addresses to the name/street lines. You obtain the network address from the Network Information Center, known as the NIC. You then assign IP addresses, and associated names, to each network interface on each host. The host IP addresses are based on the network's address, as explained below.

To obtain a registered Internet network address, contact the NIC at the following address:

InterNIC Registration Services
c/o Network Solutions, Inc.
505 Huntmar Park Drive
Herndon, VA 22070
Phone: 1–800–444–4345 or 1–703–742–4777
FAX: 703–742–4811
email: hostmaster@rs.internic.net

IP addresses are stored as 32–bit integers. They are usually expressed in "dot" notation, having the form **a.b.c.d**. In this format, the dots subdivide the 32–bit integer into four eight–bit segments, or octets. For example, `128.223.1.1`.

A internet address has a network portion and a host portion. The network portion is assigned by the NIC. The host portion (which can designate either a separate physical network or a host) is the part you assign.

When you apply for a network address, the NIC will want to know what *address class* you want. There are three address classes to choose from: Class A, Class B, and Class C, described below.

Table 2–1 Internet Address Classes

Class	First 3 bits of Address	Network Part of Address	Host Part of Address	Maximum Possible Addresses	Range of First Octet
A	0xx	a	b.c.d	16777214	1–127
B	10x	a.b	c.d	65534	128–191
C	110	a.b.c	d	254	192–223

As you can see, the network portion is either 1, 2, or 3 octets long. The host portion is whatever is left over. The criteria you use in deciding which class of network to apply for is the number of

distinct network addresses your organization will need in the foreseeable future.

For example, network `10` is a Class A network. It can have addresses between `10.0.0.1` and `10.255.255.254`. So it allows 254^3, or about 16 million possible addresses.

Network `128.223` is Class B. It can have addresses between `128.223.0.1` and `128.223.255.254`. (Do not use host numbers 0 or 255 because they may be confused with the broadcast address.) Thus, it allows 254^2, or about 65,000 possible addresses.

Network `192.12.88`, a Class C network, has addresses between `192.12.88.1` and `192.12.88.254`. So a Class C network can include only 254 possible addresses.

Where subnetting (described below) is used, network hosts need to know whether a message is destined for a network or for a host (or device) on a network. To do this, they use a value associated with the IP address called a *network mask*. The network mask, when applied to an IP address, masks out (assigns zeroes to) the host portion of the address, leaving the network portion. For example, mask `255.255.0.0` masks out the last two octets of address `128.223.0.1`, leaving 128.223. Network masks are frequently expressed in hexadecimal. Thus, the default network mask for class B networks, `255.255.0.0` in dot–format, may also be expressed as bit pattern `0xffff0000`.

Figure 2–7 shows a part of a Class B logical network. While all of the systems share the same network portion (128.223) obtained from NIC, the addresses were assigned by the local LAN manager. Notice that, where a system is connected to more than one network, it has a unique name and IP address for each network interface. By convention, a system's name (hostname) is the name of its primary network interface. Thus, in Figure 2–7, the names **sales**, **sys04**, and **sys56** refer to these hosts and also to each host's primary network interface. The names **sales–alt**, **sys04–alt**, and **sys56–alt** refer to secondary network interfaces on the hosts.

IP Address Conversion

IP addresses are software addresses. Network devices (hosts) also have hardware addresses (for example, Ethernet addresses), which are assigned at the factory. Application programs always send and receive data based on IP addresses. Before data can be sent to a destination, its IP address must be converted to a hardware address. This is done by Address Resolution Protocol (ARP). Reverse Address Resolution Protocol (RARP) does the reverse: converts a hardware address to an IP address.

In Figure 2–7, hosts **dg1**, **sun1**, and **sales** belong to the same network. (Actually, they belong to the same subnet, as explained in the next section.) As a result, using ARP and RARP, these hosts can reach one another directly.

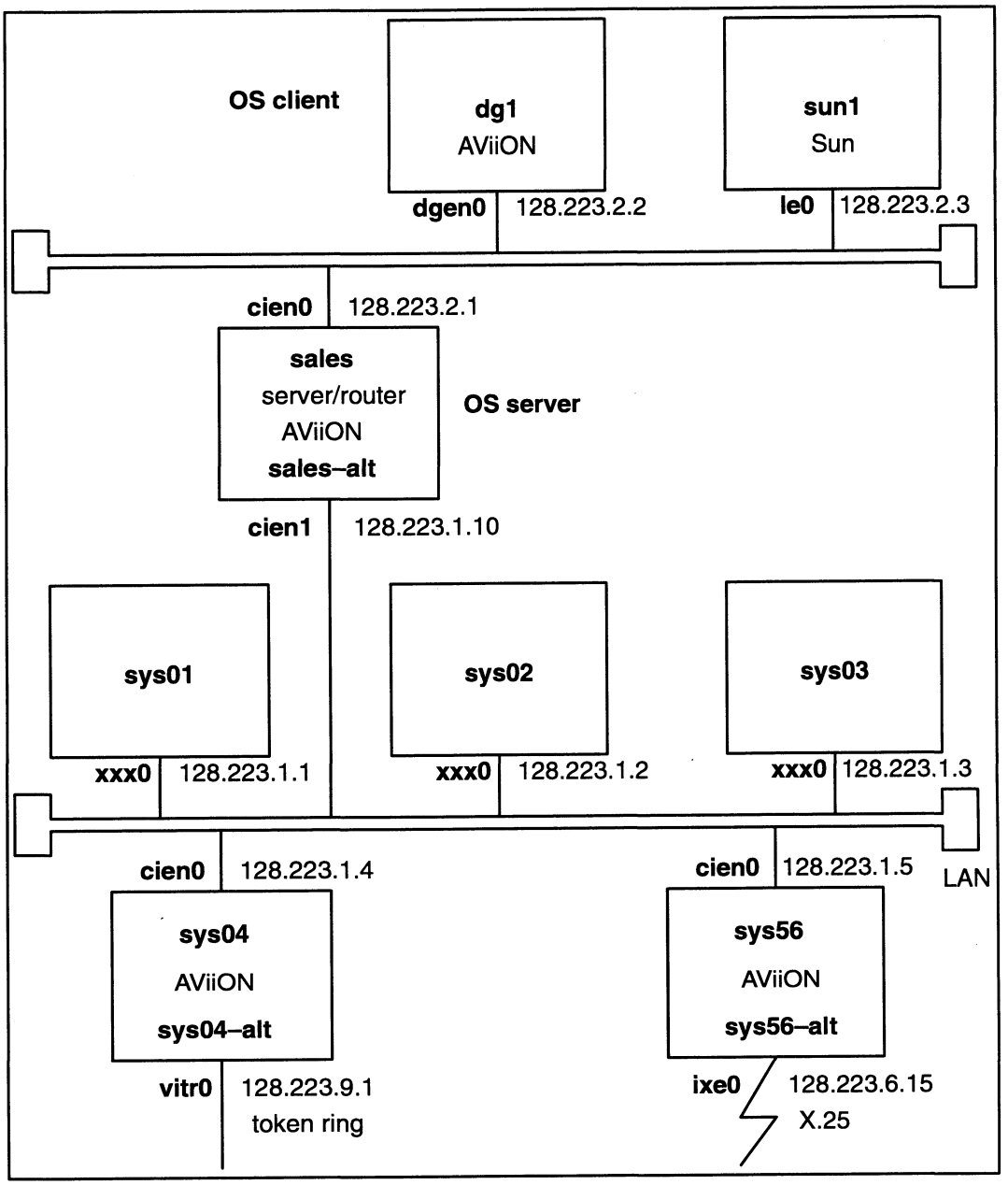


Figure 2–7 Internet Addressing in a Logical Network

Host **dg1** in this figure is an operating system client (OS client): a workstation without a disk. It boots over the network from host **sales**, its operating system server (OS server). This is one network configuration where RARP is used in a special way. During booting, **dg1** uses RARP to request and obtain its IP address from **sales** over the network.

Using Subnets

When the administrator of the system **sales** (see the figure) installed TCP/IP, he or she was prompted for the following information: the controller device name (`cien0`), the interface name (`sales`), the interface address (`128.223.2.1`), whether or not the LAN uses subnetting (`yes`), and the network mask (`0xffffff00`). This mask, which was also supplied by the installer of host **sun1**, indicated that these hosts are connected to a subnet whose address is 128.223.2. Similarly, the installers of **sys01**, **sales–alt**, **sys02**, **sys03**, **sys04**, and **sys56** supplied this same network mask, indicating that these systems share a subnet having the address 128.223.1. And the installers of **sys04–alt** and **sys56–alt** indicated, by supplying this mask, that they belong to subnets 128.223.9 and 128.223.6, respectively. (These two subnets are not shown in Figure 2–7.)

Subnets are optional. Why use them? There are several reasons:

- To accommodate cabling limitations. Ethernet cable can extend up to 500 meters (300 meters for coaxial wire). You can overcome this limitation by using a repeater or bridge as explained earlier (see Figure 2–4), or by using subnets with routing (discussed in the next section).
- To connect dissimilar networks. Figure 2–7 shows a system connected to an Ethernet subnet 128.223.1 through its primary interface (**sys04**), and to a token ring network through a secondary interface (**sys04–alt**).
- To connect parts of a geographically dispersed LAN. The subnets in Figure 2–7 are reachable by their shared Class B address, even though the subnets could be located in different buildings, states, or countries. In the figure, subnet 128.223.6 extends to a remote site. With subnetting, only the LAN administrator need be concerned about physical location. From outside, subnets are invisible.
- To distribute network traffic. Traffic between nodes on a subnet is isolated to that physical segment and does not contribute to the traffic on other subnets.
- To simplify system administration. The subnets shown in Figure 2–7 could be administered by different persons, rather than centrally by a single person or department. Subnets also make it easier to isolate performance or other problems.
- To accommodate the organization. Subnets can be set up so they correspond to different departments in an organization. Because traffic between systems sharing a subnet stays on the subnet, such a correspondence may be advantageous for security or performance reasons.

Where a LAN is geographically dispersed, or made up of different types of network (Ethernet, token ring, FDDI ring), the alternative to subnetting is for each separated branch or network type to apply to the NIC for its own network address. Subnetting is probably the better choice.

Because the original TCP/IP specifications did not allow for subnets, some older TCP/IP software cannot use them. You can tell whether software allows subnets by whether it provides a mechanism to assign or use the network mask. If your network includes software or hosts that cannot support subnets, you can still use subnets as explained in "Using Proxy ARP Routing" in Chapter 3.

For detailed reference information about subnets, see *RFC 950 (Internet Standard Subnetting Procedures)*.

Routing

A route is a path between two networks or subnets, or between hosts on different networks or subnets. If your LAN includes more than one physical network or subnet, or if it is connected to the outside world, you must either define static routes between the networks or select dynamic routing. If you do not, the hosts on each network or subnet will be able to communicate with one another, but will be unable to reach hosts on different networks or subnets.

Figure 2–8, depicting three interconnected subnets, illustrates a number of points about routing. Hosts on a physical network or subnet can communicate directly with hosts connected to the same network or subnet. Thus, in the diagram, the hosts on subnet A can reach one another, as can those on subnets B and C. But for hosts on subnets A, B, and C to reach one another, routes are necessary.

Routing is performed by routers: systems connected to more than one physical network or subnet. In the diagram, **sales** is the router between subnets A and B, and **sys56** between subnets B and C.

Hosts on a network communicate by exchanging pieces of information called datagrams. When a host sends a datagram addressed to a host on a different network, the datagram passes through one or more routers before finally reaching its destination. Thus, if host **dg1** sends a datagram addressed to **sys56**, it goes to the subnet A router, **sales**. Similarly, when a host on a subnet receives a datagram from a host on a different subnet, it arrives through a router. Thus, a datagram destined for **dg1** from **sys56** passes through **sales–alt**.

When a router receives a datagram, it either delivers the datagram directly to the addressee or forwards it to another router for

delivery. Routers make this decision by applying the network mask to the addressee's IP address. Thus, when router **sales** receives from **dg1** a datagram addressed to **sys56** (`128.223.1.5`), it applies the network mask to learn that **sys56** is on subnet B (`128.223.1`). The router then delivers the datagram through its **sales–alt** interface to subnet B. (Section "Static Routing" gives a more detailed example of the role of routing tables.)

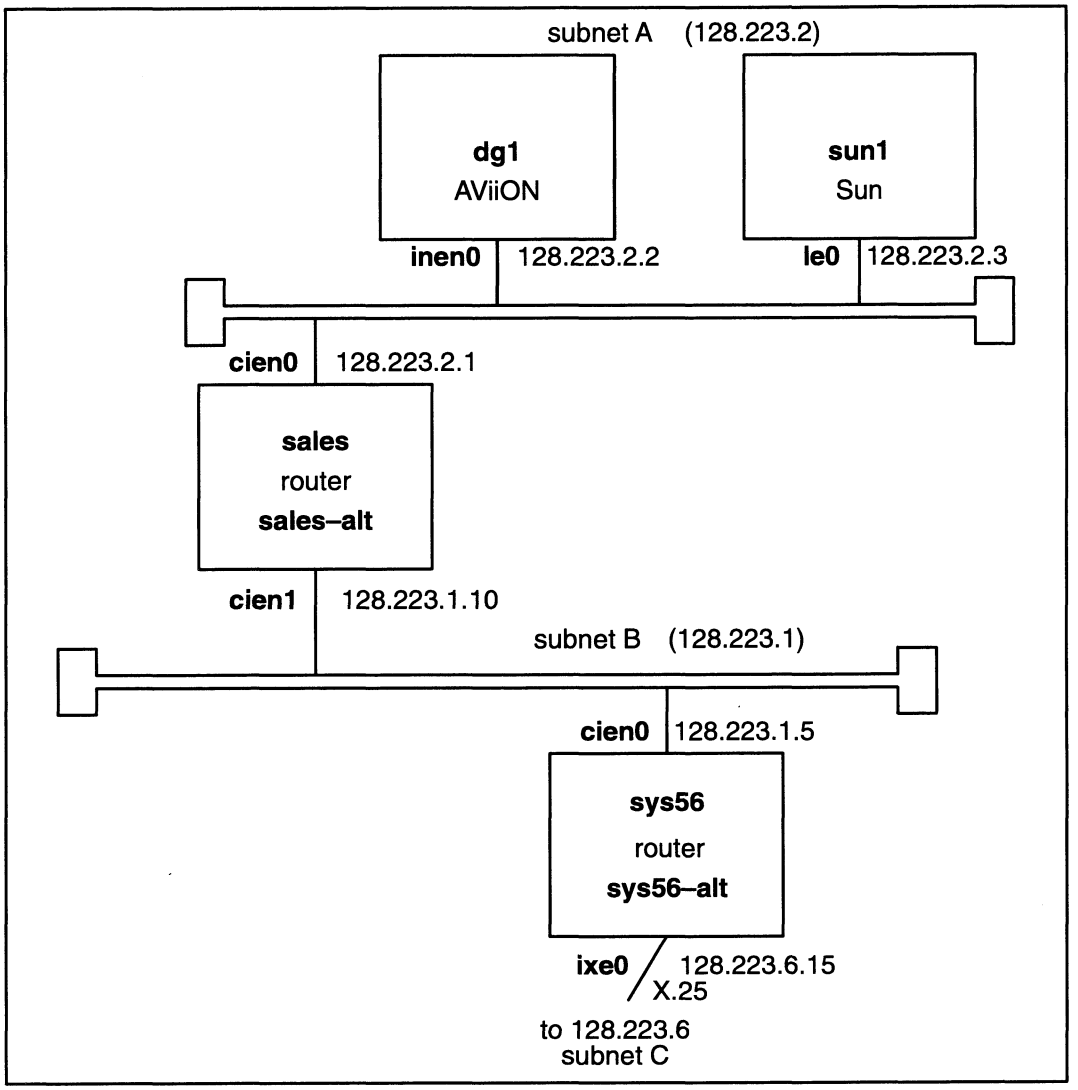


Figure 2–8 Routing Path in the Sample Logical Network

Selecting a Routing Method

What happens when a router receives a datagram it cannot deliver? For example, if router **sales** receives a datagram addressed to a host on subnet C, how does it go about forwarding the datagram to **sys56–alt** for delivery? The answers to these questions depend on

the type of routing you use. DG/UX TCP/IP supports two types, static and dynamic. Both methods are described below.

For performance reasons, you should use static routing on diskless workstations (OS clients). As illustrated in the next section, static routing is also a good choice where only a single path exists between networks. Dynamic routing is recommended where multiple paths exist between networks.

Static Routing

If you choose static routing, someone (the system or network administrator) must manually build routes allowing network hosts to reach one another. If the network topology changes, the administrator must modify the routes. When building static routes, you can use the **route**(1M) command or you can use **sysadm** (explained in the next chapter). Routes built with **route** must be rebuilt each time a system is rebooted, whereas routes built with **sysadm** are permanent (recreated each time a system boots). For this reason, **sysadm** is recommended.

Figure 2–8 illustrates a simple network topology where static routing works well: three subnets (A, B, and C) interconnected by single routers (**sales** between A and B, **sys56** between B and C).

When a host receives a request to connect with another host, it looks up the host's address in its routing table. You can see the routing table by typing **netstat -rn** from a host. Assuming no routes have been added, here's what you see (minus some omitted columns) if you type this command from host **dg1**:

```
Destination     Gateway        Interface
127.0.0.1       127.0.0.1      loop0
128.223.2       128.223.2.2    dgen0
```

Host **sales**'s routing table would have one additional line. If you type **netstat -rn** from **sales** (again, assuming no routes have been added), here's what you see:

```
Destination     Gateway        Interface
127.0.0.1       127.0.0.1      loop0
128.223.2       128.223.2.1    cien0
128.223.1       128.223.1.10   cien1
```

The first entry, the *loopback* entry, is the same in all routing tables. It allows a host to connect with a host named *localhost* – i.e., with itself. The second entry is a route to a network (destination) to which a host is directly connected via its network interface. The Interface column lists the device name of the network controller on

which the network interface is configured, and the Gateway column lists the IP address assigned to this network interface. Since **sales** has two controllers connecting it to two networks, there are two entries.

A minimal routing table such as these is built by default during network startup. The table is sufficient to allow a host to reach other hosts on the same subnet. To illustrate how this works, suppose a user on **dg1** attempts to log in to host **sun1**:

```
dg1% telnet sun1
```

Here's what host **dg1** does:

a. Finds out that **sun1**'s IP address is 128.223.2.3.

b. Searches its routing table for **sun1**'s exact address, which fails.

c. Searches for a route to **sun1**'s network, which succeeds. (For this search, **dg1** applies the network mask to address 128.223.2.3, revealing subnet address 128.223.2.)

d. Connects with **sun1**. (To do this, using ARP, **dg1** maps 128.223.2.3 to **sun1**'s Ethernet address.)

Following this same procedure, because it is connected to both subnets, host **sales** can reach **dg1** and **sun1** on subnet A and **sys56** on subnet B. However, if a user on **dg1** attempts to log in to host **sys56**, here's what happens:

```
dg1% telnet sys56
telnet: connect: Network is unreachable
```

For the simple network topology illustrated in Figure 2–8, there is a simple way to set up static routes allowing the hosts on subnets A, B, and C to communicate. To each routing table, you add a *default* route: a forwarding address to use when no exact host or network address is found. Specifically, on host **dg1**, you add a default route to the subnet A router, **sales**:

```
Destination     Gateway         Interface
127.0.0.1       127.0.0.1       loop0
128.223.2       128.223.2.2     dgen0
default         128.223.2.1     dgen0
```

In effect this line says: send any messages addressed to hosts on other subnets to the subnet router and let it decide what to do. If you add similar routes to each subnet A and subnet B host, the hosts will be able to reach one another. However, if the **dg1** user tries to log in to a host located on subnet C, the same "Network is unreachable" message appears. To solve this problem, on the router

sales, you can add a default route to the router for subnets B and C, **sys56**:

```
Destination     Gateway         Interface
127.0.0.1       127.0.0.1       loop0
128.223.2       128.223.2.1     cien0
128.223.1       128.223.1.10    cien1
default         128.223.1.5     cien1
```

Now, **sales** forwards data addressed to hosts that are not on subnets A and B to **sys56**, which decides what to do with the data. If you add default routes like that shown above for **dg1** to each subnet C host, the hosts on subnets A, B, and C can communicate.

The default route mechanism has its limits. Subnet B (see Figure 2–8) has two routers, **sales–alt** and **sys56**. For subnet B hosts, should you set the default destination to **sales–alt** or **sys56**? You might choose to add specific routes to both routers and also set the default to one of them. Or you might set the default to one subnet router and make sure it has a route to the other subnet. There are other possibilities, which multiply geometrically if you add a third router to subnet B.

As you can see from this discussion, static routing works well in simple network topologies. But where hosts on a LAN have more than one possible way to reach one another, static routing becomes complicated to set up and difficult to maintain. Also, the more possible routes there are between networks, the harder it is to determine the best route.

Dynamic Routing

If you choose dynamic routing, you set up your network to run routing programs that exchange routing information with one another and update the routing tables of network hosts, without manual intervention. Where multiple paths are possible, the programs determine the best path. Where the best or usual path is not available (for example, because a router is down), the programs provide alternative routes.

Routing protocols define the mechanisms used by dynamic routing programs. Protocols run on routers: systems (hosts or special–purpose computers) connected to two or more physical networks or subnets that exchange routing information about local hosts and keep their routing tables current. TCP/IP for AViiON supports Routing Information Protocol (RIP) Version 1 and Version 2 and Open Shortest Path First (OSPF) Version 2. Another protocol, HELLO, is also supported but is seldom used for routing. RIP, the older and more widely used, runs on all versions of DG/UX

and on most other UNIX systems. OSPF is supported on some UNIX versions including current (1994 or later) DG/UX releases.

TCP/IP for AViiON supports two dynamic routing programs, **routed**(1M) and **gated**(1M). The older program, **routed**, implements RIP only. It runs on all versions of DG/UX and on most other UNIX systems. The newer dynamic routing daemon, **gated**, implements OSPF and HELLO as well as RIP. **Gated** runs on current (1994 or later) DG/UX releases. We recommend **gated** in all dynamic routing configurations.

Configuring gated

The **gated** program includes an extensive, C–like language for selecting routing protocols and describing network topologies. By default, **gated** is configured for RIP. If your network uses only RIP, the default configuration may be sufficient. Simply start **gated** on hosts that should participate in dynamic routing (or **routed** on any hosts that do not support **gated**).

This section illustrates how to go about configuring **gated** for OSPF through examples. Compared with RIP, OSPF offers greater flexibility and significant performance gains. See the next chapter, "Configure Dynamic Routing with **gated**" when you're ready to configure **gated**. See **gated**(1M) and **gated–config**(4M) for additional information. For complete OSPF information, see *RFC 1247*.

Figure 2–9 shows a sample LAN consisting of four systems connecting three TCP/IP subnets and an X.25 network. The remainder of this chapter illustrates how you could configure these four systems for dynamic routing with **gated** and OSPF.

The TCP/IP network at the top (128.999.1) runs RIP only. The network on the lower left is an X.25 network. The other TCP/IP subnets run both RIP and OSPF.

Figure 2–9 shows two routers (**systemA** and **systemB**) and two non–routing hosts (**systemC** and **systemD**). While only four systems are shown, assume that the LAN includes multiple additional routers providing additional connections between these (and other) networks, and that each network includes additional non–routing hosts. This is important. If the LAN consisted only of the networks and systems shown in the figure, static rather than dynamic routing would be the better choice.

Why do networks 128.999.8 and 128.999.3 run both RIP and OSPF? The assumption is that these networks include some hosts that support only **routed** and RIP, and other hosts that support **gated** and OSPF. Where all the hosts on a network support **gated** and

OSPF, it is best for performance to configure each host for **gated** and OSPF with RIP disabled.

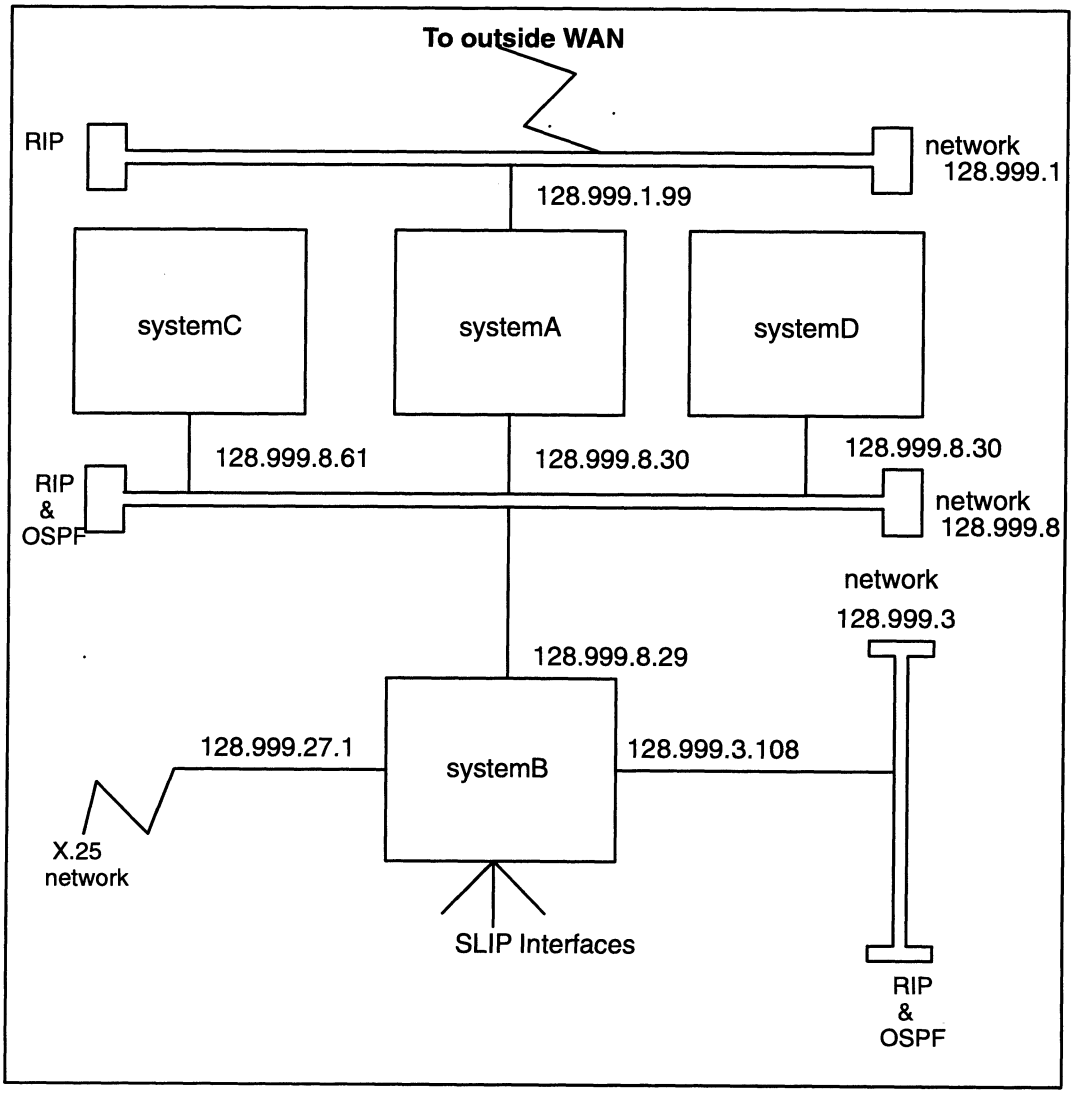


Figure 2–9 Dynamic routing with RIP and OSPF

SystemA

A sample configuration file for the router **systemA** is shown in Figure 2–10. As indicated in the figure, the **gated** configuration file may contain eight classes of statement. The configuration file includes, in each section, comments illustrating **gated** configuration syntax. Figure 2–10 omits these comments, showing only the lines that have been added for **systemA**. Also, the lines shown in the figure are numbered for ease of reference. The numbers are not in the file.

```
100  ############################
101  # DIRECTIVE STATEMENTS
102  ############################
103  #
200  ############################
201  # TRACE STATEMENTS
202  ############################
203  #
300  ############################
301  # OPTIONS STATEMENTS
302  ############################
303  #
400  ############################
401  # INTERFACE STATEMENTS
402  ############################
403  #
500  ############################
501  # DEFINITION STATEMENTS
502  ############################
503  #
504  routerid 128.999.8.30 ; #SystemA
505
506
600  ############################
601  # PROTOCOL STATEMENTS
602  ############################
603  #
604  rip on ;
605
606
607  hello off ;
608
609
610  ospf on  {
611           monitorauthkey "mypasswd" ;
612           backbone {
613                  authtype simple ;
614                  interface 128.999.8.30 {
615                                  enable ;
616                                  priority 100 ;
617                                  authkey "mypasswd" ;
618                  } ;
619           } ;
620  } ;
621
622
700  ############################
701  # STATIC STATEMENTS
702  ############################
703  #
800  ############################
801  # CONTROL STATEMENTS
802  ############################
803  #
804  export proto ospfase {
805          proto direct ;  # Announce *all* direct interfaces via OSPF
806          proto rip ;     # Announce *all* received RIP destinations via OSPF
807  } ;
```

Figure 2–10 Sample **gated** configuration file for **systemA**

Definition Section

By default, where OSPF is enabled, **gated** sets a system's OSPF router ID to the IP address of whichever network interface it first encounters. Line 504 specifically sets the OSPF router ID to the

address of the **systemA** interface (128.999.8.30) to the network that runs OSPF. This statement prevents the other **systemA** interface (128.999.1.99) from possibly becoming the OSPF router ID.

Protocol Section

Line 604 starts RIP in default mode on both of **systemA**'s network interfaces. The RIP default mode is compatible with the **routed** program, whereby RIP updates are supplied only by systems with more than one network interface and RIP version 1 packets are transmitted in IP broadcast mode. Line 607 disables the HELLO protocol. Since RIP is enabled and HELLO is disabled by default, these lines could have been omitted from the example, but are shown for completeness.

Lines 610–620 activate the OSPF protocol. Line 611 specifies that **gated** on this system should look for the named authentication string in all OSPF packets, and ignore any that do not contain the string. The other systems in this network are configured for the same authentication key, to allow unrestricted exchange of packets.

OSPF allows you to subdivide a LAN into *areas*: groups of one or more subnets that you wish to configure independently from the rest of the LAN. Each area of a LAN runs a separate copy of the OSPF routing algorithms. In large or heterogeneous LANs, the area feature of OSPF improves performance by allowing routing to be configured based on particular local characteristics within the LAN.

Our sample network (Figure 2–9) contains only one area. Where this is the case, the area is configured as the *backbone* area. Where there are multiple areas, you configure each in a separate, numbered section, beginning with area number 1. While areas are optional, a backbone section is required. Where area sections other than the backbone are present, they must precede the backbone section.

Since the sample LAN has only one area (the backbone), all the statements in lines 612–619 apply to it. Line 613 enables authorization by simple password within this area. Password generation is on a per–interface basis. Thus, line 617 names the string that will be added to OSPF packets generated on this interface.

While **systemA** has two network interfaces, only one of them (`128.999.8.30`) is to an OSPF network. This is indicated in lines 614–616, which enable OSPF on the interface to the network that runs OSPF.

Note that, when you start RIP (line 604), it starts by default on all of a system's network interfaces. By contrast, there is no default with OSPF. Its startup statement (lines 610–620) specifically identifies each network interface on a system that should run OSPF.

In an OSPF network, the routers within each area exchange routing information only until they reach agreement about the routing topology of the area. This process is called *synchronization*. After reaching synchronization, the routers on each area subnet choose a *designated router* for the subnet, and also a backup designated router. The designated router assumes routing responsibility for its subnet. While an area remains in synchronization, only the designated routers exchange routing information, thus reducing area traffic. If an event occurs that changes the routing topology within an area (for example, an interface or router goes down, or a new interface is added), the routers within that area re–enter synchronization, and at the conclusion re–select one or more designated routers.

The priority value (line 616) is used during the selection of the designated router. A value of 0 (the default priority) prevents a system from being selected. Routers with non–zero priorities are eligible for selection: the larger a router's priority (maximum 255), the greater its likelihood of being selected.

Control Section

Control statements specify which routes and protocols are received (imported) from other routers or advertised (exported) to them. They are needed on **systemA** because it is a router connecting a network that runs only RIP with one that runs both RIP and OSPF.

Lines 804–807 instruct **gated** on **systemA** to translate the RIP packets it receives from network 128.999.1 to OSPF, before supplying OSPF packets to network 128.999.8. Without these lines, the routing table updates received by **systemC**, which runs only OSPF, would not include the routes being advertised on the RIP network 128.999.1.

SystemB

The configuration file for **systemB** is shown in Figure 2–11. The Definition statement is identical to **systemA**'s except for the address. Because **gated** on this system does not need to translate between protocols, no Control statements are needed. The other differences appear in the Interface and Protocol sections, as explained below.

```
400  ############################
401  # INTERFACE STATEMENTS
402  ############################
403  #
404  interfaces {
405             interface slip passive ;          # Don't time out SLIP interfaces
406             interface 128.999.27.1 passive ;  # Don't time out X.25 interface
407  } ;
408
409
500  ############################
501  # DEFINITION STATEMENTS
502  ############################
503  #
504  routerid 128.999.8.29 ; #SystemB
505
506
600  ############################
601  # PROTOCOL STATEMENTS
602  ############################
603  #
604  rip on   {
605           interface slip noripin noripout ;        # No RIP on SLIP interfaces
606           interface 128.999.27.1 noripin noripout  # or X.25 interface
607   } ;
608
609  hello off ;
610
611
612  ospf on  {
613           monitorauthkey "mypasswd" ;
614           backbone {
615                  authtype simple ;
616                  interface 128.999.8.29  {
617                                 enable ;
618                                 priority 100 ;
619                                 authkey "mypasswd" ;
620                  } ;
621                  interface 128.999.3.108 {
622                                 enable ;
623                                 priority 100 ;
624                                 authkey "mypasswd" ;
625                  } ;
626           } ;
627  } ;
```

Figure 2–11 Sample **gated** configuration file for **systemB**

Interface Section

For each interface route on a system, **gated** maintains a *preference*: a value between 0 (the default for direct interfaces) and 255 indicating the route's likelihood of being used. The lower the preference, the greater the likelihood. Also maintained for each interface is a *down time preference*: a value, 120 by default, that **gated** sets a route's preference to if it believes the interface to be down.

On a router, if **gated** stops receiving updates on an interface, it assumes the interface is down, sets the interface's preference to the down time preference value, and stops advertising that interface to the network. Lines 404–407 prevent this behavior for the SLIP and

the X.25 interfaces. The X.25 network does not run a routing protocol, and the SLIP interfaces are unlikely to. Without these lines, **gated** would probably stop advertising these interfaces on the incorrect assumption that they were down.

SLIP interfaces have the string "slip" in their names. The argument `slip` (lines 405 and 605) is a wildcard name that resolves to any slip interface. The X.25 interface is referenced specifically by its IP address. Interfaces can also be referenced by interface or controller name.

Protocol Section

Host **systemB** is connected to two subnets, each of which runs both RIP and OSPF. The system also has an interface to an X.25 network, and is configured to be a SLIP server for one or more SLIP clients.

Lines 604–607 enable RIP on all of **systemB**'s interfaces except for the SLIP and X.25 interfaces. Because serial or modem lines are slow, it would not be a good idea for **systemB** to supply routing packets to SLIP clients. (Typically, SLIP clients use static default routes to their SLIP server rather than participate in dynamic routing.) The X.25 interface is excluded because it does not run a routing protocol.

Lines 612–627 enable OSPF on **systemB**'s interfaces to the two OSPF networks. Since OSPF is started only on the named interfaces, there is no need, as with RIP, to worry about SLIP clients. OSPF is not enabled on the interface to the X.25 network because that network does not run OSPF. (This is a feature of the example: X.25 networks support the running of routing protocols.)

Hosts systemC and systemD

Hosts **systemC** and **systemD** are typical of clients on a LAN that uses dynamic routing. Each has a single network interface and runs a single network protocol. The hosts get their routing tables updated but do not actively participate in routing decisions.

Host **systemC** is connected to a network that runs both RIP and OSPF, but it runs only OSPF. The configuration file for **systemC** is shown in Figure 2–12.

Because **systemC** has only one interface, a Definition statement would be redundant. Because it runs a single protocol, it requires no Control statements.

In the Protocol section, lines 604 and 607 disable RIP and HELLO. Lines 610–620 enable OSPF on **systemC**'s network interface. To

prevent this system from being selected as designated router, line 616 sets its priority to 0. (Since 0 is the default, this line could be omitted.)

```
600  ############################
601  # PROTOCOL STATEMENTS
602  ############################
603  #
604  rip off ;
605
606
607  hello off ;
608
609
610  ospf on  {
611           monitorauthkey "mypasswd" ;
612           backbone {
613                  authtype simple ;
614                  interface 128.999.8.61 {
615                                 enable ;
616                                 priority 0 ;
617                                 authkey "mypasswd" ;
618                  } ;
619           } ;
620  } ;
```

Figure 2–12 Sample **gated** configuration file for **systemC**

Host **systemD** has a single interface to a network that runs both RIP and OSPF. To run RIP on this system, no configuration is required because the default **gated** configuration file enables RIP and disables HELLO and OSPF. Because **systemD** has only one network interface, **gated** receives broadcasts from area routers but does not send any.

Host **systemD** could be a system that does not support **gated**. For example, it might be an AViiON system running a version of DG/UX released earlier than 1994, or a system from another vendor. In this case, to enable RIP, simply start the program **routed**.

End of Chapter

3 Administering a DG/UX TCP/IP Network

The primary tool for managing your DG/UX TCP/IP network is **sysadm**. To see the TCP/IP maintenance functions, become superuser, start **sysadm**, and select `Networking-> TCP/IP`:

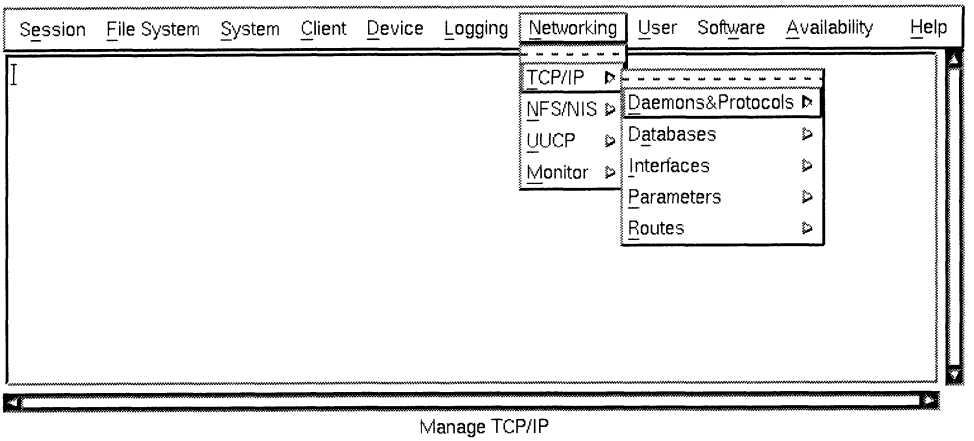


Figure 3–1 The TCP/IP maintenance procedures, graphical interface

The **sysadm** program also has an ASCII interface. If you start the program from an ASCII terminal (or type **asysadm** from a terminal window), a display like the following appears instead:

```
                TCP/IP Network Menu

  1  Daemons&Protocols -> Manage TCP/IP daemons & protocols
  2  Databases ->         Manage TCP/IP databases
  3  Interfaces ->        Manage TCP/IP network interfaces
  4  Parameters ->        Manage TCP/IP parameters
  5  Routes ->            Manage TCP/IP routing

Enter a number, a name, ? or <number>? for help, <NL> to
redisplay menu, ^ to return to previous menu, or q to quit:
```

Figure 3–2 The TCP/IP maintenance procedures, ASCII interface

While the selection methods differ, the graphical and ASCII versions are functionally equivalent. The illustrations in this chapter are taken from the graphical version of **sysadm**.

You may perform the maintenance functions with shell commands, by editing system files, or with **sysadm**. This chapter explains the **sysadm** interface but it names the analogous command or configuration file. If you prefer the command interface, see the appropriate man pages.

This chapter explains the **sysadm** configuration procedures for routing, the Domain Name System (DNS), the Simple Network

Management Protocol (SNMP), and the Serial Line Internet Protocol (SLIP). There are other, non–**sysadm** procedures that you may have to perform to complete configuration in these areas. These subjects are discussed at greater length in other chapters (see the table of contents). There is no **sysadm** interface for sendmail; see Chapter 4 for sendmail configuration information.

The last section of the chapter, "Miscellaneous Administrative Topics," discusses several administrative topics that cannot be performed with **sysadm**.

Maintain Databases

The TCP/IP databases define the pieces that make up your network: host and network names and addresses, network interfaces, and network services. To see the databases, select `Networking-> TCP/IP-> Databases:`

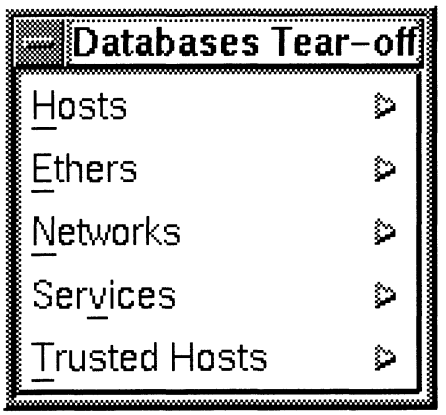


Figure 3–3 Database maintenance procedures

Networks that use the Network Information Service (NIS) administrative database system typically store these databases (except the last) centrally, on the NIS master host, which disseminates the information to the other hosts on the LAN. On networks that do not use NIS, each host on the LAN includes locally the information it needs to communicate with other LAN or Internet hosts.

Maintain Host Names and Addresses

The hosts database matches host names with their IP addresses. Select `Networking-> TCP/IP-> Databases-> Hosts` to see the host maintenance functions:

Figure 3–4 Host maintenance procedures

Every AViiON system has a local host file, `/etc/hosts`. The local hosts file must contain:

- An entry for the loopback pseudo–device. This entry is the same on every system (name `localhost`, IP address `127.0.0.1`) and you don't have to add it. But make sure you do not delete it.
- An entry for each controller installed on the system. For systems having only one controller, you normally do not need to add the name and address, as this is done when you install DG/UX on the system from the release tape or CD–ROM. For systems having more than one controller, you must add a name and IP address for each additional controller as explained in "Add Hosts" below, then add a network interface for the controller as explained in the section "Add a Network Interface."
- On systems that define static routes, an entry for each host named in the route section of `/etc/tcpip.params`.

If your network uses the Network Information Service (NIS), the local hosts file need not (and probably does not) contain any other entries than these. Typically, on networks that use NIS, you add any other host names and addresses needed by the network to the hosts database on the NIS master rather than to the local host files on individual systems.

You can update or list the local file or (if your network uses NIS) list the hosts in the NIS master file. If your host is the NIS master, you can update the NIS hosts database.

The Domain Name System (DNS) provides another method of resolving Internet addresses that supplements the hosts mechanism explained in this section. The DNS software is included with DG/UX TCP/IP but is not configured by default. To set up DNS on a server, see Chapter 5. To set up DNS on a client, see the section "Manage DNS" in this chapter.

List Hosts

Select `Networking-> TCP/IP-> Databases-> Hosts-> List` to see the hosts database:

Hosts database to use: This prompt appears if your network uses NIS. Select **NIS (YP)** to see the NIS hosts database or **Local (/etc)** to see the `/etc/hosts` file on the local system. Once you identify the database, a scrollable and printable report appears.

Add Hosts

Select `Networking-> TCP/IP-> Databases-> Hosts-> Add` to add a host:

Hosts database to use: This prompt appears only if you are on the NIS master host. Select **NIS (YP)** to add to the hosts database on the NIS master or **Local (/etc)** to add a host to `/etc/hosts` file on the local system.

Host Name: Enter the host's name, which must not already exist in the hosts file.

Internet Address: Enter the host's IP address.

Alias List: Optionally, enter one or more alternate names for the host, separated by commas.

IMPORTANT: A host (for example, a gateway or router) may be connected to more than one network or subnet. Such hosts have a different IP address for each LAN interface (controller). Add such hosts once for each interface, each time specifying a different name. We suggest you follow a convention such as **hostname**, **hostname–alt**, and **hostname–slip**. To further differentiate between a host and its LAN interfaces, you can add descriptive aliases.

Modify Hosts

Select `Networking-> TCP/IP-> Databases-> Hosts-> Modify` to change a host name, IP address, or alias. If you are working from any host except the NIS master, this prompt appears:

Host Name: Identify the record in the `/etc/hosts` file you wish to modify.

If you are working from the NIS master, prompts appear allowing you to transfer a record as well as modify it:

Hosts database to search: If you are transferring a record between the local file on the NIS master and NIS master file,

identify the file you wish to transfer from: **Local (/etc)** to transfer from the `/etc/hosts` file, or **NIS (YP)** to transfer from the NIS master file. If you are not transferring a record, these selections identify the file containing the record you wish to modify.

Host Name: Identify the host record to modify or transfer.

Hosts database to modify: Select **NIS (YP)** to modify the NIS master file or **Local (/etc)** to modify the `/etc/hosts` file.

Once you've identified the database and record, you can change the name, address, or alias (see "Add Hosts" above for descriptions).

Delete Hosts

Select `Networking-> TCP/IP-> Databases-> Hosts-> Delete` to remove a host.

Hosts database to use: This prompt appears only if you are on the NIS master host. Select **NIS (YP)** to delete from the hosts database on the NIS master or **Local (/etc)** to delete from the `/etc/hosts` file.

Host Name(s): Enter the name of one or more hosts to delete, separated by commas.

IMPORTANT: The `/etc/hosts` file on each host must contain the loopback entry plus entries for each installed network interface (controller). Do not remove these entries.

Maintain Ethernet Addresses

The ethers database matches host names on a LAN with their Ethernet addresses. On Ethernet networks, diskless clients and X terminals use this database when booting from their servers. You need to maintain this information only on servers that support diskless clients or X terminals.

Select `Networking-> TCP/IP-> Databases-> Ethers` to see the ethernet address maintenance functions:

Figure 3–5 Ethernet address maintenance procedures

Every AViiON system has a local ethernet address file, `/etc/ethers`. You can update or list the local file, or list the entries in the NIS master file. If your host is the NIS master, you can update the NIS ethers database.

A system's Ethernet address is displayed during power up. To find out the Ethernet address of a running system, use one of the following commands. The first works for a remote system named *host* on the same network or subnet. The second method, which returns the Ethernet addresses of all network interfaces on *host*, works for hosts that support SNMP.

```
% ping host; arp host
% snmpgetmany host public ifDescr ifPhysAddress
```

List Ethernet Addresses

Select `Networking-> TCP/IP-> Databases-> Ethers-> List` to see the ethers database:

Ethernet database to use: This prompt appears if your network uses NIS. Select **NIS (YP)** to see the NIS ethernet database or **Local (/etc)** to see to `/etc/ethers` file on the local system. Once you identify the database, a scrollable and printable report appears.

Add Ethernet Addresses

Select `Networking-> TCP/IP-> Databases-> Ethers-> Add` to add an Ethernet address for a host:

Ethernet database to use: This prompt appears only if you are on the NIS master host. Select **NIS (YP)** to add to the ethers database on the NIS master or **Local (/etc)** to add an address to the `/etc/ethers` file on the local system.

Host Name: Enter the host's name, which should already exist in the hosts database.

Ethernet Address: Enter the host's Ethernet address.

Modify Ethernet Addresses

Select `Networking-> TCP/IP-> Databases-> Ethers-> Modify` to change a host's name or its Ethernet address. If you are working from any host except the NIS master, this prompt appears:

Host Name: Identify the record in the `/etc/ethers` file you wish to modify.

If you are working from the NIS master, prompts appear allowing you to transfer a record as well as modify it:

Ethernet database to search: If you are transferring a record between the local file on the NIS master and NIS master file, identify the file you wish to transfer from: **Local (/etc)** to transfer from the `/etc/ethers` file, or **NIS (YP)** to transfer from the NIS master file. If you are not transferring a record, these selections identify the file containing the record you wish to modify.

Host Name: Identify the record to modify or transfer.

Ethernet database to modify: Select **NIS (YP)** to modify the NIS master file or **Local (/etc)** to modify the `/etc/ethers` file.

Once you've identified the database and record, you can change the name, address, or alias (see "Add Ethernet Addresses" above for descriptions).

Delete Ethernet Addresses

Select `Networking-> TCP/IP-> Databases-> Ethers-> Delete` to remove an Ethernet address.

Ethernet database to use: This prompt appears only if you are on the NIS master host. Select **NIS (YP)** to delete from the ethers database on the NIS master or **Local (/etc)** to delete from the `/etc/ethers` file.

Host Name(s): Enter the name of one or more hosts whose Ethernet address you wish to delete. Separate the names by commas. You are prompted to confirm the deletion.

Maintain Network Names and Addresses

The networks database matches network or subnet names with their IP addresses. Select `Networking-> TCP/IP-> Databases-> Networks` to see the network maintenance functions:

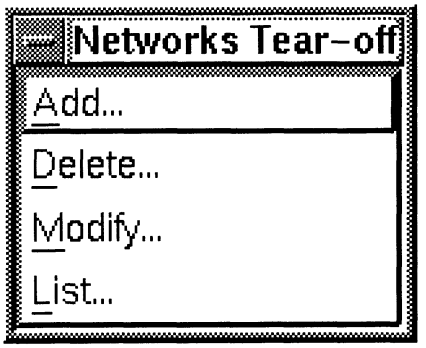


Figure 3–6 Network maintenance procedures

Every AViiON system has a local network file, `/etc/networks`. You can update or list the local file, or (if your network uses NIS) list the networks in the NIS master file. If your host is the NIS master, you can update the NIS networks database.

List Networks

Select `Networking-> TCP/IP-> Databases-> Networks-> List` to see the networks database:

Networks database to use: This prompt appears if your network uses NIS. Select **NIS (YP)** to see the NIS networks database or **Local (/etc)** to see the `/etc/networks` file on the local system. Once you identify the database, a scrollable and printable report appears.

Add Networks

Select `Networking-> TCP/IP-> Databases-> Networks-> Add` to add a network or subnet:

Networks database to use: This prompt appears only if you are on the NIS master host. Select **NIS (YP)** to add to the networks database on the NIS master or **Local (/etc)** to add a network to the `/etc/networks` file on the local system.

Network Name: Enter the network's name, which must not already exist in the networks file.

Internet Address: Enter the IP address of the network. Use a full four octets: for example, 111.222.1.0 rather than 111.222.1.

Alias List: Optionally, enter one or more alternate names for the network, separated by commas.

Modify Networks

Select `Networking-> TCP/IP-> Databases-> Networks-> Modify` to change a network's name, IP address, or alias. If you are working from any host except the NIS master, this prompt appears:

Network Name: Identify the record in the `/etc/networks` file you wish to modify.

If you are working from the NIS master, prompts appear allowing you to transfer a record as well as modify it:

Networks database to search: If you are transferring a record between the local file on the NIS master and NIS master file, identify the file you wish to transfer from: **Local (/etc)** to transfer

from the `/etc/networks` file, or **NIS (YP)** to transfer from the NIS master file. If you are not transferring a record, these selections identify the file containing the record you wish to modify.

Network Name: Identify the record to modify or transfer.

Networks database to modify: Select **NIS (YP)** to modify the NIS master file or **Local (/etc)** to modify the `/etc/networks` file.

Once you've identified the database and record you want to modify, you can change the name, address, or alias list (see "Add Networks" above for descriptions).

Delete Networks

Select `Networking-> TCP/IP-> Databases-> Networks-> Delete` to remove a network.

Networks database to use: This prompt appears only if you are on the NIS master host. Select **NIS (YP)** to delete from the networks database on the NIS master or **Local (/etc)** to delete from the `/etc/networks` file.

Network Name(s): Enter the name of one or more networks to delete, separated by commas.

IMPORTANT: The `/etc/networks` file on each host must contain the loopback–net entry. Do not remove it.

Maintain Service Names and Ports

A *network service* is an application program available via the network. When executing, a service has a unique port number. The services database matches service names and TCP/IP protocols with port numbers. After a packet reaches the correct host destination, the port number ensures that the data reaches the correct process. Select `Networking-> TCP/IP-> Databases-> Services` to see the host maintenance functions:

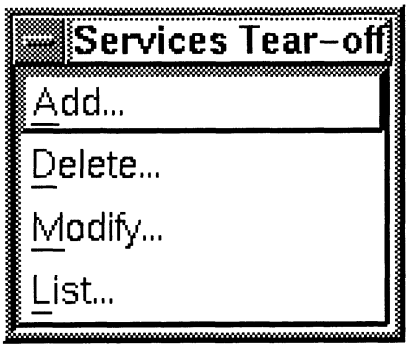


Figure 3–7 Network services maintenance procedures

Every AViiON system has a local services file, `/etc/services`. You can update or list the local file, or (if your network uses NIS) list the networks in the NIS master file. If your host is the NIS master, you can update the NIS services database.

List Network Services

Select `Networking-> TCP/IP-> Databases-> Services-> List` to see the services database:

Services database to use: This prompt appears if your network uses NIS. Select **NIS (YP)** to see the NIS services database or **Local (/etc)** to see the `/etc/services` file on the local system. Once you identify the database, a scrollable and printable report appears.

Add Network Services

Select `Networking-> TCP/IP-> Databases-> Services-> Add` to add a service:

Services database to use: This prompt appears if your network uses NIS. Select **NIS (YP)** to add to the services database on the NIS master or **Local (/etc)** to add to the `/etc/services` file on the local system.

Service Name: Enter a name for the service.

Service Protocol: Enter the protocol used by the service: `udp`, `tcp`, or another protocol supported by your network. Protocols entered here must be present in the file `/etc/protocols`.

Service Port: Enter the port number: an unassigned decimal value > 256 (for UNIX–specific services) or > 1024 (for general services). Numbers below 256 are reserved for system services. (System services are defined in the Assigned Numbers RFC.)

Service Name Aliases: Optionally, enter one or more alternate names for the service, separated by commas.

Modify Network Services

Select `Networking-> TCP/IP-> Databases-> Services-> Modify` to change a service name, protocol, port number, or alias. If you are working from any host except the NIS master, these prompts appears:

Service Name: Identify the record in the `/etc/services` file you wish to modify.

Service Protocol: Enter the service's protocol.

If you are working from the NIS master, prompts appear allowing you to transfer a record as well as modify it:

Services database to search: If you are transferring a record between the local file on the NIS master and NIS master file, identify the file you wish to transfer from: **Local (/etc)** to transfer from the `/etc/services` file, or **NIS (YP)** to transfer from the NIS master file. If you are not transferring a record, these selections identify the file containing the record you wish to modify.

Service Name: Identify the record to modify or transfer.

Service Protocol: Enter the service's protocol.

Services database to modify: Select **NIS (YP)** to modify the NIS master file or **Local (/etc)** to modify the `/etc/services` file.

Once you've identified the database and record you want to modify, you can change the name, protocol, port number or alias list (see "Add Network Services" above for descriptions).

IMPORTANT: A number of services are predefined by Data General. Do not modify these.

Delete a Network Service

Select `Networking-> TCP/IP-> Databases-> Services-> Delete` to delete a service:

Services database to use: This prompt appears only if you are on the NIS master host. Select **NIS (YP)** to delete from the services database on the NIS master or **Local (/etc)** to delete from the `/etc/services` file.

Service Name: Enter the name of the service you want to delete.

Service Protocol: Enter the service's protocol.

IMPORTANT: A number of services are predefined by Data General. Do not delete these.

Maintain Trusted Hosts and Users

By default, users who log in to a network host through **rlogin** or **remsh** must enter passwords. The trusted hosts file (`/etc/hosts.equiv`) may list network hosts whose users should be exempt from this requirement. Select `Networking-> TCP/IP->`

`Databases-> Trusted Hosts` to see the trusted host maintenance functions:

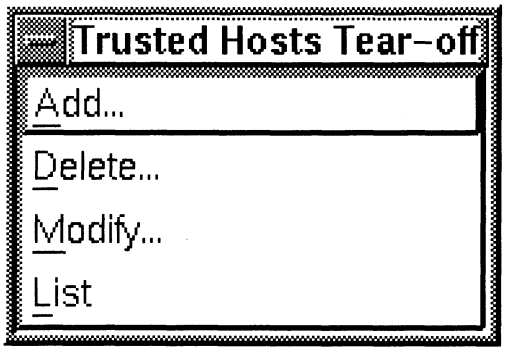


Figure 3–8 Trusted Hosts maintenance procedures

These procedures modify or display the `/etc/hosts.equiv` file on the local host.

List Trusted Hosts

Select `Networking-> TCP/IP-> Databases-> Trusted Hosts-> List` to see the trusted hosts file.

Add Trusted Hosts

Select `Networking-> TCP/IP-> Databases-> Trusted Hosts-> Add` to add a trusted host:

Host Name: Enter a valid hostname expression, as explained below:

+	Match all hosts.
+@ *groupname*	Match all hosts in *groupname*. For more information about defining groups, see **netgroup**(4).
–@*groupname*	Deny access to users from all hosts in *groupname*.
hostname	Allow access to users from *hostname*.
–*hostname*	Deny access to users from *hostname*.

If your network uses the Domain Name System (DNS), you may need to enter fully–qualified DNS host names as well as simple host names: for example, `sys01.tnt.acme.com` as well as `sys01`. Similarly, hosts with multiple network interfaces may need to be added more than once: for example, `sys01-alt` as well as `sys01`.

User Name: Optionally enter a valid username expression. Blank or `all` means all users on **Host Name** can log in to this host without entering passwords. A username expression may include:

+	Match all users.
+@*groupname*	Match all users in *groupname*. For more information about defining groups, see **netgroup**(4).
–@*groupname*	Deny access to all users in the *groupname*.
user	Allow access to specified *user*.
–*user*	Deny access to specified *user*.

IMPORTANT: This procedure enforces correct syntax but does not prevent you from entering an incorrect host or user name.

Modify Trusted Hosts

Select `Networking-> TCP/IP-> Databases-> Trusted Hosts->. Modify` to change a host or user entry in `/etc/hosts.equiv`:

Select trusted host Entry: Select the trusted host entry you wish to modify.

Delete Trusted Hosts

Select `Networking-> TCP/IP-> Databases-> Trusted Hosts-> Delete` to remove a trusted host entry from `/etc/hosts.equiv`.

Select trusted host Entry: Select the trusted host entry you wish to delete.

Maintain Network Interfaces

Each network host has one or more communications controllers connecting it to one or more physical networks. Controllers are hardware devices that, as data passes through them, translate between the different formats required by a physical network and the TCP/IP software. Use these procedures to configure or maintain TCP/IP interfaces for controllers. Select `Networking-> TCP/IP-> Interfaces` to see the interface maintenance functions:

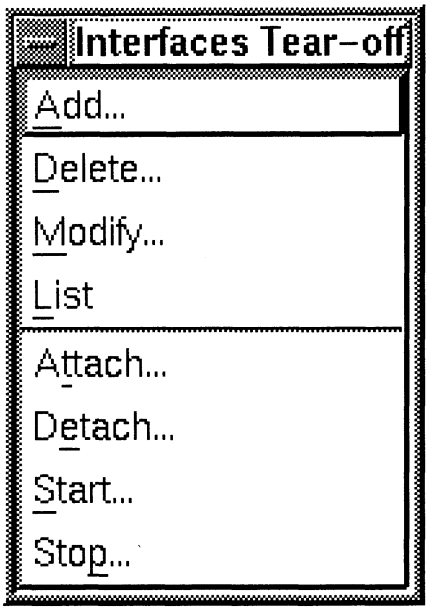


Figure 3–9 Network interface maintenance procedures

If a system includes a controller at the time you install (or update) TCP/IP from a DG/UX release tape or CD–ROM, a TCP/IP interface (the *primary* interface) will be set up as part of the DG/UX installation procedures (explained in *Installing the DG/UX System*). Use the interface maintenance procedures described in this section to:

- Modify an interface.
- Add a TCP/IP interface to an additional controller(s) on a system that includes more than one: the DG/UX installation procedures provide for setup of the primary interface only.
- Add an interface for a controller purchased since the last DG/UX update.

When setting up a new interface on a running system, the recommended order of procedures is: (1) add the interface as explained in this section; (2) rebuild the kernel (select `System-> Kernel-> Build` to add the new controller to the system file); (3) shut down your system (4) install the new controller (5) reboot.

When installing and setting up a controller, consult the applicable hardware documentation. Table 3–1 lists currently available controllers or devices available for AViiON systems.

Table 3–1 Common Network Devices

Name in /dev	Description
loop	Loopback pseudo-device
hken	VME-based Ethernet controller (VLC)
cien	VME-based intelligent Ethernet controller (VLCi)
inen	Integrated LAN controller
dgen	Second generation integrated LAN Controller
ixe	IXE pseudo-device, for X.25 support
vitr	VME-based token ring device (VTRC)
pefn	VME-based FDDI controller (VFC)

For more information about the VLC, see the *V/Ethernet 3207 Hawk Local Area Network Controller for Ethernet User's Guide*. For more information about the VLCi, see *VLCi Ethernet LAN (CMC-130) Controller Reference Guide*. For information about how to jumper the VTRC controller and about how to insert the board in the 2-slot VME chassis, see *Setting Up and Installing VMEbus Options in AViiON Systems*. For more information about how to install the VTRC in AViiON 5000 Series systems, see *Expanding the AViiON® 5000 Series System*. For more information about the about the VTRC controller itself, see *Configuring the VME Token Ring Controller (VTRC) for AViiON Systems*. For more information about the VFC, see *V/FDDI (4211 Peregrine) User's Guide*. Also, see the **hken**(7), **cien**(7), **inen**(7), **dgen**(7), **ixe**(7), **vitr**(7), and **pefn**(7) manual pages.

List Network Interfaces

To see a host's network interfaces, select `Networking-> TCP/IP-> Interfaces-> List`. (See the next section for an explanation of the displayed information.)

Add a Network Interface

IMPORTANT: Before adding an interface, add its IP name and address to the local (`/etc/hosts`) file. See "Add Hosts" above for this procedure.

- After adding a controller, you usually must rebuild the kernel and reboot your system. The new interface does not start until you do.

  To add a network interface, select `Networking-> TCP/IP-> Interfaces-> Add`. Then go to the appropriate section:

  - "Add a Standard Interface" if you're adding a LAN interface over an Ethernet, Token Ring, or FDDI controller
  - "Add a SLIP Interface" to add a serial interface over a TTY port
  - "Add an IXE Interface" if you're adding an interface to an X.25 network over an IXE pseudo–device

Add a Standard Interface

Host Name: Enter the name or associated IP address to be assigned to this network interface. This name must exist in the local hosts (`/etc/hosts`) file.

Device: Enter the name of the controller to which this interface applies. Table 3–1 lists controller device identifiers for currently available controllers.

A device name includes its number, beginning with 0. Thus, if you are adding an interface to a pefn device, enter `pefn0` if the system currently has no pefn device, `pefn1` if you are adding an interface for a second pefn device, `pefn2` for a third, and so on. The device name resolves to an identical name in `/dev`, which the kernel recreates each time the system is booted.

After identifying a controller's internet name or address and device name, you are prompted for these items of information:

Netmask: Enter a netmask or accept the displayed value. You must enter a netmask if your network includes subnets.

Broadcast Host Address: Select ones if the network includes only AViiON and other SVID–compliant systems. Select zeroes if the network must support BSD 4.2–compliant systems.

Link Level Protocol: Select ether (Ethernet) or IEEE 802.3. Based on this entry, the system assigns a name to the interface. For ether protocol, the interface name is the same as the device name. For IEEE 802.3 protocol, the interface name is the device name with a prefix of `snap_`: for example, `snap_inen0`.

Add a SLIP Interface

IMPORTANT: Adding a SLIP interface with this procedure does not complete the procedure. See the section "Manage SLIP."

Host Name: Enter a dialsystem name. This name should be defined in files `/etc/slipdialinfo` and `etc/slipusers`. If it is not, a warning will appear but the interface will still be added.

Device: Enter `ttyn`, where *n* is a 2–digit number (00, 01, etc). This entry causes the system to interpret the **Host Name** as a dialsystem name.

SLIP TTY Baud Rate: Select the baud rate. For a modem connection, select the rate supported by the modem. For a serial cable connection, match the rate selected on the remote system.

Add an IXE Interface

Host Name: Enter the name or associated IP address to be assigned to this network interface. This name must exist in the local hosts (`/etc/hosts`) file.

Device: Enter `ixen`, where *n* is the number of the IXE device: 0 if there is only one, 1 if this is the second of two, and so on.

Netmask: Enter a netmask or accept the displayed value. You must enter a netmask if your network includes subnets.

IXE Template File: Enter the simple filename of the protocol parameter file for configuring IXE. This is the file you created using the **sysadm** procedures explained in Chapter 4 of *Setting Up and Managing X.25 on the DG/UX System* (093–701071).

Modify an Interface

IMPORTANT: Before modifying an interface, first detach it, then re–attach it after making your modification (see "Start or Stop, Attach or Detach" below). If the system uses static routing, you'll have to re–create the routes.

To change a network interface, select `Networking-> TCP/IP-> Interfaces-> Modify`:

Host Name: Select the configured interface you wish to change. You are then given the opportunity to change all the values described in the section "Add a Network Interface."

Delete an Interface

IMPORTANT: Before deleting an interface, first detach it as explained in the next section.

To remove a network interface, select `Networking-> TCP/IP-> Interfaces-> Delete`:

Host Name(s): From the displayed list, select the interface you wish to delete. When prompted, confirm your choice.

Start or Stop, Attach or Detach

Network controllers are started and interfaces are attached when you boot your system. Should you need to disable (suspend) an interface, without affecting the interface software, select `Networking-> TCP/IP-> Interfaces-> Stop`. To enable (resume) a disabled interface, select `Networking-> TCP/IP-> Interfaces-> Start`.

Select `Networking-> TCP/IP-> Interfaces-> Detach` to remove a network interface from memory. Select `Networking-> TCP/IP-> Interfaces-> Attach` to restart (initialize) a previously detached network interface.

IMPORTANT: Stopping a controller or detaching an interface on a system running at init level 3 may cause problems. This depends on how network applications using the interface handle errors, on the type of routing in use, and on whether alternative network interfaces are present.

- Detaching an interface removes any routes associated with the interface, and re–attaching the interface does not re–establish any routes. If the system uses dynamic routing, the routing daemon will re–establish lost routes. However, if the system uses static routes, you must re–create them after Detach and Attach operations. (The Start and Stop operations do not affect static routes.)
- Do not stop the controller or detach the interface on a diskless client. If you do, the system will hang.

Set or Display Parameters

Select `Networking-> TCP/IP-> Parameters` to see the Parameters functions:

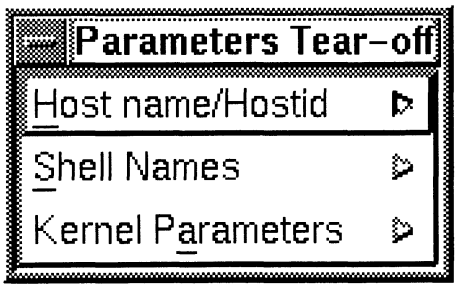


Figure 3–10 Parameters procedures

These procedures display or set items in the `/etc/tcpip.params` file.

Get or Set the Host's Name and ID

IMPORTANT: By convention, a host's name is the same as the name of its primary network interface, but the convention is not enforced. This procedure displays or sets the value returned by the **hostname**(1C) and **uname**(1) commands. Changing a host's name or ID with this procedure does *not* affect the host's internet names or addresses. Internet names and addresses are defined by the procedures explained in "Maintain Host Names and Addresses," and are assigned to a host's network interfaces by the procedures explained in "Maintain Network Interfaces."

Select `Networking-> TCP/IP-> Parameters-> Host name/Hostid-> Get` to get the hostname and hostid. Select `Networking-> TCP/IP-> Parameters-> Host name/Hostid-> Set` to set a new hostname and hostid:

Host Name: Enter the host name.

Hostid: Enter the host's hexadecimal ID. The number you enter sets the value returned by the **hostid** (1C) command. By convention, the hostid is a host's primary internet dot address expressed in hexadecimal. However, the convention is not enforced: any valid hexadecimal number will be accepted.

Get or Set the Shell Name

By default, following the BSD convention, the DG/UX name for the remote shell is either **rsh** or **remsh**, and the name for the restricted shell is **restsh**. Use this procedure if you prefer the System V convention: call the remote shell **remsh** and the restricted shell **rsh** or **restsh**. The set procedure changes the command names and matches them with the correct man pages.

Select `Networking-> TCP/IP-> Parameters-> Shell Names-> Get` to get the current names names of the remote and restricted shells. Select `Networking-> TCP/IP-> Parameters-> Shell Names-> Set` to set the names:

System: Select `BSD` or `System V`.

Get or Set Kernel Parameters

Use these procedures to display the current settings of the TCP/IP tunable parameters or to set new values. See **sysconfig** (1M) for reference.

Select `Networking-> TCP/IP-> Parameters-> Kernel Parameters-> Get` to see the current setting of the tunable parameters, or `Networking-> TCP/IP-> Parameters-> Kernel Parameters-> Reset` to set the parameters to their initial default values:

Select parameter(s) to fetch: Select `all` to inspect or reset all tunable parameters, or select the one you want to see.

To change one or more tunable parameters, Select `Networking-> TCP/IP-> Parameters-> Kernel Parameters-> Set`:

Enable IP Forwarding Enable (the default) if this system should forward IP packets destined for other hosts. Disable to prevent this host from performing this service on behalf of other hosts. This setting does not affect source–routed IP packets (packets that specify the hosts they are to pass through on the way to their destination).

Enable IP Non–Local Source Route Forwarding Disable (the default) to prevent a host that has two or more network interfaces (controllers) from forwarding a source–routed packet between networks. Enable to permit forwarding of source–routed packets between network interfaces. This setting has no effect on hosts having only one controller.

IP Default Time–to–Live: Enter the maximum hop count (number of intermediate hosts) for an IP packet before it should be dropped. Default 255, range 1–255.

TCP Default Keep–Alive Idle Time: Enter the number of seconds that TCP should wait before sending Keep–Alive probes (when the Keep–Alive facility is active). Default 7200, range 0–14400.

ARP Default Cache Timeout: Enter the base value to be used in the calculation of ARP cache time–outs. The actual time–out values are: this value times 20 (complete ARP entries) or times 3 (incomplete ARP entries) seconds. Default 60, range 1–1,000,000.

Maintain Daemons and Protocols

TCP/IP for AViiON includes several daemons and protocols supporting communications applications. Select `Networking-> TCP/IP-> Daemons&Protocols` to see these applications:

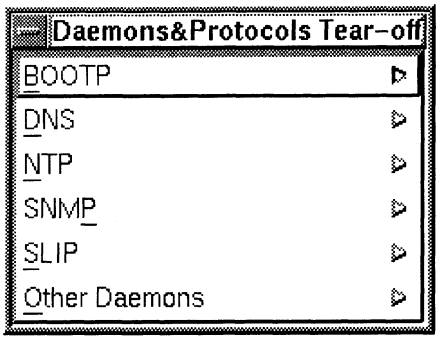


Figure 3–11 TCP/IP Daemon and Protocol Procedures

Manage Daemons

A communications daemon (also called server, server process or agent; optionally spelled demon) is a program that runs on each network host that supports an application. A daemon starts or stops other programs that may be needed by an application, and manages communications with other network hosts.

The main TCP/IP daemon, **inetd**, manages the common internet applications and their associated daemons such as **ftpd**, **telnetd**, and **rshd**: it runs on every TCP/IP host, listening for connections on the ports designated for the services specified in the configuration file `/etc/inetd.conf`, and invoking the appropriate daemon. Other (independent) daemons, such as **smtp**, **rwhod**, **pmtd**, **snmpd** and **named**, run or not depending on whether a host supports an application or interface.

Independent daemons (listed in `/etc/tcpip.params`) are started during system initialization, whenever a system enters run level 3. Daemons managed by **inetd** (listed in `/etc/inetd.conf`) are started when their services are requested. The maintenance procedures explained in this section modify or display the contents of these files.

IMPORTANT: Do not manage the daemons for BOOTP, NTP, DNS, SNMP, dynamic routing, and SLIP with the Daemons procedures. The **sysadm** functions for these applications and protocols include procedures for managing their daemons.

Select `Networking-> TCP/IP-> Daemons&Protocols-> Other Daemons` to see the daemon maintenance procedures:

Figure 3–12 Daemon maintenance procedures

Use the maintenance procedures (add, delete, modify, and list) to view or change the daemons that are started whenever a system enters run level 3. Use start and stop to restart or halt a daemon.

List Daemons

Select `Networking-> TCP/IP-> Daemons&Protocols-> Other Daemons-> List` to see the current daemon setup:

Select the type of daemon: Select `Independent` to see the daemons currently started when your system enters run level 3. Select `inetd Managed` to see the daemons started by **inetd** when service is requested.

The list of independent daemons gives the daemon's name and optional arguments. The list of inetd–managed daemons gives, in addition, the service and its protocol as defined in the services database: see "Maintain Service Names and Ports" in this chapter for service and protocol definitions.

Add a Daemon

Select `Networking-> TCP/IP-> Daemons&Protocols-> Other Daemons-> Add` to arrange for the starting of a daemon during system initialization or on demand.

Select the type of daemon: Select `Independent` for a daemon that should be started during initialization independently, or `inetd Managed` if it should be started on demand by **inetd** (added to `/etc/inetd.conf`).

Daemon Name: Enter the name of the executable program. The program (or a symbolic link to it) must reside in `/usr/bin`.

Daemon Arguments: Optionally, enter arguments to be passed to the daemon when it is invoked. See the daemon's man page for a description of its options and arguments.

Daemon Service: This prompt appears for inetd–managed daemons. Select `Networking-> TCP/IP-> Databases-> Services` to see currently–defined services and their protocols.

Daemon Protocol: This prompt appears for inetd–managed daemons. Select the service's protocol from the displayed choices.

Start the daemon now: Indicate whether the daemon should be started (yes by default).

Modify a Daemon

Select `Networking-> TCP/IP-> Daemons&Protocols-> Other Daemons-> Modify` to change the options or characteristics of a daemon that is started during system initialization:

Select the type of daemon: Select `Independent` for a daemon that is started independently or `inetd Managed` for daemons started by **inetd**.

Daemon Name: Select the daemon to modify from the displayed names.

Once you've identified the daemon to be modified, prompts appear allowing you to change and restart it. See "Add a Daemon" above.

Delete a Daemon

Select `Networking-> TCP/IP-> Daemons&Protocols-> Other Daemons-> Delete` to remove a daemon from the `/etc/tcpip.params` or `/etc/inetd.conf` files:

Select the type of daemon: Select `Independent` for a daemon that is started independently or `inetd Managed` for daemons started by **inetd**.

Daemon Name: Select the daemon to remove from the displayed pathnames. Confirm when prompted.

Stop the daemon now: Indicate whether the daemon should be stopped (yes by default). Confirm when prompted.

Deleting a running daemon stops it.

Start or Stop an Independent Daemon

Select `Networking-> TCP/IP-> Daemons&Protocols-> Other Daemons-> Start` to start a stopped independent daemon or to restart one that is running:

Daemon Name: Daemons in `/etc/tcpip.params` are displayed. Select the one you want to start.

To stop a daemon, select `Networking-> TCP/IP-> Daemons&Protocols-> Other Daemons-> Stop:`

Daemon Name: Daemons that are listed in `/etc/tcpip.params` are displayed. Select the one you want to stop.

Manage DNS

The Domain Name System (DNS) is a client/server application that resolves IP host names and addresses. DNS supplements an older method of resolving names and addresses based on lookup tables (see "Maintain Host Names and Addresses" in this chapter for a description of the local and NIS host databases). With DNS, host name and address information is distributed among domains rather than centralized. Each domain has one or more name servers which supply names and addresses to their clients and query other name domain servers for information they don't have.

Programs that use DNS link themselves with a library of resolver routines. When a program wants a hostname/address pair or some other information, it calls a resolver routine, which in turn queries its name servers, which either supply the information or call other name servers.

DNS is not configured by default. If your site uses DNS, configure it on one or more domain servers as explained in Chapter 5. Then use the procedures described in this section to configure DNS on each client that uses the service. Select `Networking-> TCP/IP-> Daemons&Protocols-> DNS-> Resolver` to see the DNS domain procedures:

Figure 3–13 DNS domain maintenance procedures

These procedures display or update the configuration file `/etc/resolv.conf`.

Get or Set the DNS Domain

To find out the DNS domain name, select `Networking-> TCP/IP-> Daemons&Protocols-> DNS-> Resolver-> Get`.

To set the domain for this client, select `Networking-> TCP/IP-> Daemons&Protocols-> DNS-> Resolver-> Set`:

DNS Domain: Enter the name of the DNS domain to which this system belongs. DNS domain names are provided on request by the Network Information Center (NIC).

List the Domain Servers

A domain has up to three name servers. Select `Networking-> TCP/IP-> Daemons&Protocols-> DNS-> Resolver-> List` to see the IP addresses of the domain's name servers.

Add a Name Server

Select `Networking-> TCP/IP-> Daemons&Protocols-> DNS-> Resolver-> Add` to add a name server for the domain:

Name server: Enter the host name or the IP address of the DNS server.

IMPORTANT: The Add and Modify functions verify that a host exists, but not that it is configured as a DNS server.

Replace a Name Server

Select `Networking-> TCP/IP-> Daemons&Protocols-> DNS-> Resolver-> Modify` to replace a name server:

Name server to Modify: Select the host to replace.

Name server: Enter the replacement name server's host name or the IP address.

Delete a Name Server

Select `Networking-> TCP/IP-> Daemons&Protocols-> DNS-> Resolver-> Delete` to remove a name server:

Name server to Delete: Select the host to remove and confirm your choice when prompted.

Get or Set the Name Resolution Search Order

When a program encounters a host name, it must convert the name to an internet address. Three name resolution methods are available. A program can:

- Search the local (`/etc/hosts`) file. This is the only method available to networks that use neither the Network Information Service (NIS) nor Domain Name System (DNS).
- Search the NIS hosts database. Available to networks that use NIS, which is included with DG/UX and set up during installation by default.
- Use DNS. Available to networks that use DNS, which is included with DG/UX but is not configured by default.

The file `/etc/svcorder` on each host specifies which methods it uses and in which order. When you configure a host to use DNS, add the DNS resolution method to this file in the desired order.

To find out the current resolution search order, select `Networking-> TCP/IP-> Daemons&Protocols-> DNS-> Search Order-> Get`. To change the order, select `Networking-> TCP/IP-> Daemons&Protocols-> DNS-> Search Order-> Set`. As prompted, specify the first, second, and third methods.

These functions display or edit the `/etc/svcorder` file. See **svcorder** (4) for an explanation of the file's syntax.

Start or Stop named

Select `Networking-> TCP/IP-> Daemons&Protocols-> DNS-> Start` to specify startup options for the DNS daemon, **named**:

Start when: Select `start now and on reboots`, `start now`, or `start on reboots`.

Select `Networking-> TCP/IP-> Daemons&Protocols-> DNS-> Stop` to stop **named**:

Stop when: Select `stop now and don't restart on reboots`, `stop now`, or `don't restart on reboots`.

Manage SNMP

The Simple Network Management Protocol (SNMP) allows a network administrator to perform a number of network or system management and inquiry functions from a single location (network management station).

SNMP service is provided by the daemon **snmpd**, which runs on every participating network host and responds to SNMP management stations, or to shell commands: **snmpgetone** (1M), **snmpgetnext**(1M), **snmpgetmany** (1M), **snmpgettab** (1M), and **snmpsetany**(1M). SNMP is included with DG/UX TCP/IP and, by default, **snmpd** is started for inquiry operations whenever a system

enters multiuser mode. (See "Start or Stop snmpd" to disable or modify the daemon.)

Select `Networking-> TCP/IP-> Daemons&Protocols-> SNMP` to see the SNMP management functions:

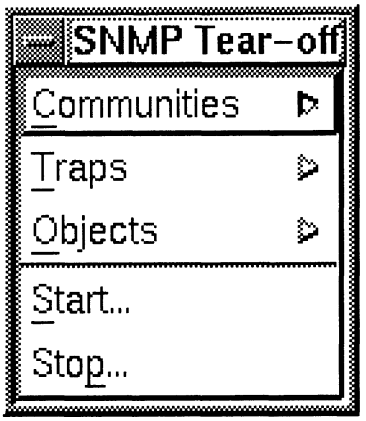


Figure 3–14 SNMP maintenance procedures

These procedures modify or display the local SNMP configuration file, `/etc/snmpd.conf`.

SNMP requests include a host, a community, and one or more object arguments. For example:

```
snmpgetone host community object
```

The `community` argument allows **snmpd** to verify a request before complying with it. One predefined community, `public`, allows SNMP hosts to query one another. The Communities procedures allow you to define additional communities with different access characteristics, which may then be used in requests directed to the hosts on which you've added the communities.

The `object` argument indicates the information you are interested in obtaining or setting. The Objects procedures allow you to set 6 site–specific types of information. Assign appropriate values to these objects on each network host you'll be monitoring from your system.

A trap is a message sent from a host indicating some error (a verification or link failure, for example) or change in state. The Traps procedures specify where a host sends such messages — typically to the host from which you issue SNMP management requests. No traps are provided by default. Unless you define one or more traps on a host, its **snmpd** sends no messages.

Maintain Communities

Select `Networking-> TCP/IP-> Daemons&Protocols-> SNMP-> Communities` to see the Communities procedures:

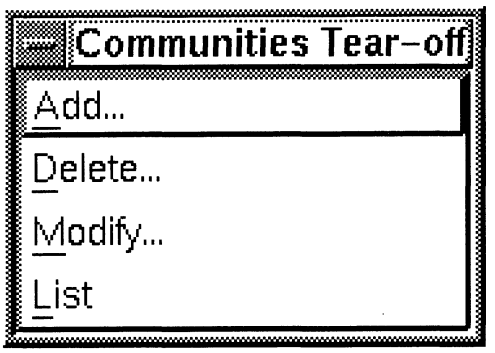


Figure 3–15 Communities maintenance procedures

List Communities

Select `Networking-> TCP/IP-> Daemons&Protocols-> SNMP-> Communities-> List` to see currently defined communities.

Add a Community

Select `Networking-> TCP/IP-> Daemons&Protocols-> SNMP-> Communities-> Add` to define a new community:

Community Name: Enter a name for the new community.

Host Name: Enter a host name or IP address or accept the default `any`. If you enter a particular host name or address, this community can be used only from that host.

Level of access: Select `Read-Only` (disallow updating of objects), `Read-Write` (allow updating), or `No-access` (disable the community argument).

IMPORTANT: Add and Modify do not check an entered host name or address for validity. A community argument associated with an invalid address will be ignored.

Modify a Community

Select `Networking-> TCP/IP-> Daemons&Protocols-> SNMP-> Communities-> Modify` to change a community argument:

Select A Community Entry: Select the community argument you wish to modify. You are then allowed to change its name, associated host, or access as explained above.

Delete a Community

Select `Networking-> TCP/IP-> Daemons&Protocols-> SNMP-> Communities-> Delete` to remove a community argument:

Select A Community Entry: Select the community argument you wish to delete. Confirm your choice when prompted.

Maintain Traps

Select `Networking-> TCP/IP-> Daemons&Protocols-> SNMP-> Traps` to see the Traps procedures:

Figure 3–16 Traps maintenance procedures

List Traps

Select `Networking-> TCP/IP-> Daemons&Protocols-> SNMP-> Traps-> List` to see where this host's **snmpd** currently sends trap messages.

Add a Trap

Select `Networking-> TCP/IP-> Daemons&Protocols-> SNMP-> Trap-> Add` to designate a host to receive a trap:

Community Name: Enter the name of the community for which you are setting a trap destination.

Host Name: Enter the name or IP address of the recipient host. Normally, this is the host from which you manage or monitor the network.

Port Number: Enter a port number or accept the default (161).

IMPORTANT: Add and Modify do not check communities and hosts for validity. Traps associated with nonexistent communities or invalid hosts will be ignored.

Modify a Trap

Select `Networking-> TCP/IP-> Daemons&Protocols-> SNMP-> Traps-> Modify` to change a trap:

Select A Trap Entry: Select the trap you wish to modify. You are then allowed to change the trap community, recipient host, or port as explained above.

Delete a Trap

Select `Networking-> TCP/IP-> Daemons&Protocols-> SNMP-> Traps-> Delete` to remove a trap:

Select A Trap Entry: Select the trap you wish to delete. Confirm your choice when prompted.

Maintain Objects

Select `Networking-> TCP/IP-> Daemons&Protocols-> SNMP-> Objects` to see the Objects procedures:

Figure 3–17 Objects maintenance procedures

List Objects

Select `Networking-> TCP/IP-> Daemons&Protocols-> SNMP-> Objects-> List` to see the current values of six site–specific objects. The Set procedure will let you put anything you like into these objects. Their recommended values are suggested by their names:

`sysDescr` Hardware, operating system, and revision.

`sysObjectID` The Management Information Base (MIB) object ID for this host (MIB is described in Chapter 6)

`sysContact` The system's administrator or assigned user and contact information

`sysLocation` The office location of the system

`sysName` The system's host name. By default, this value is set from **hostname** (1M). Note that, while this value should be the same as a host's internet name, it is not necessarily the same. Setting this object does not change the value of **hostname**(1M).

`snmpEnableAuthenTraps`
Enter 1 (default) to enable or 2 to disable the sending of traps for requests that fail to pass authentication.

Get, Set, or Reset an Object

Select:

- `Networking-> TCP/IP-> Daemons&Protocols-> SNMP-> Objects-> Get` to see the current value of one object (similar to List)
- `Networking-> TCP/IP-> Daemons&Protocols-> SNMP-> Objects-> Reset` to set a previously modified object back to the initial default value
- `Networking-> TCP/IP-> Daemons&Protocols-> SNMP-> Objects-> Set` to enter a new value for an object:

Select An Object to Set: Select which of the six objects you wish to modify. Enter its value when prompted. Object values are limited to a single line.

Start or Stop snmpd

Select `Networking-> TCP/IP-> Daemons&Protocols-> SNMP-> Start` to specify startup options for the SNMP daemon, **snmpd**:

Start when: Select `start now and on reboots`, `start now`, or `start on reboots`.

Select `Networking-> TCP/IP-> Daemons&Protocols-> SNMP-> Stop` to stop **snmpd**:

Stop when: Select `stop now and don't restart on reboots`, `stop now`, or `don't restart on reboots`.

Manage NTP

The Network Time Protocol (NTP) keeps the clocks of LAN hosts accurate and synchronized with one another. One or more

designated NTP servers on your LAN update their clocks periodically from hosts on the Internet equipped with very accurate clocks (stratum 1 hosts), or that get their time directly from clock–equipped hosts (stratum 2 hosts). In turn, the NTP clients on your LAN periodically update their clocks from local NTP servers.

NTP synchronization is provided by the **xntpd** daemon, which runs on every host that uses the service. NTP also includes shell commands to synchronize a clock (**ntpdate**), or that query (**ntpq**) or dynamically modify (**xntpdc**) the **xntpd** daemon.

NTP is provided with DG/UX TCP/IP but is not configured by default. To see the NTP setup procedures, select `Networking-> TCP/IP-> Daemons&Protocols-> NTP:`

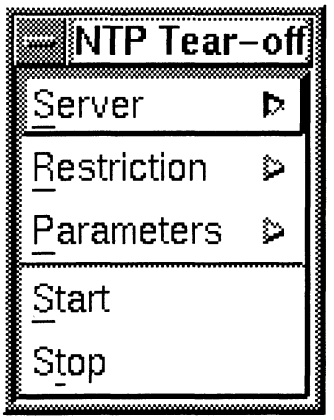


Figure 3–18 NTP setup procedures

The Server, Restriction and Parameters procedures modify the **xntpd** configuration file, `/etc/ntp.conf`. See **xntpd** (1M) for reference.

Setup requires that you add one or more clock servers and then start **xntpd** on each NTP host. These procedures are explained in the next two sections. The optional Restriction and Parameters procedures allow you to modify default **xntpd** behavior.

In a network that uses NTP, most systems are NTP clients: hosts that get their clocks updated by an NTP server, which do not update the time of other hosts. There are two ways to set up NTP clients. On each client, you can specify one or more NTP servers as explained in "Add a Clock Server" below. Or you can do the following:

- On each NTP client, set the **Broadcast Client** parameter to yes as explained in "Get or Set xntpd Parameters."
- On each client's NTP server, set the **Server Type** to `broadcast` and enter the client's broadcast address, as explained in "Add a Clock Server."

Specify Clock Servers

On each system that uses NTP, you must identify one or more network hosts from which the system gets its clock updated. On an NTP server, you identify a host on the Internet that provides the service. On an NTP client, you typically identify a local NTP server.

The file `pub/ntp/doc/clock.txt`, available via anonymous FTP from host `louie.udel.edu`, lists Internet clock servers and explains how to request service from them.

Select `Networking-> TCP/IP-> Daemons&Protocols-> NTP->Server` to see the server maintenance procedures:

Figure 3–19 Clock server maintenance procedures

List Clock Servers

Select `Networking-> TCP/IP-> Daemons&Protocols-> NTP-> Servers-> List` to see clock servers currently used by this system.

Add a Clock Server

Select `Networking-> TCP/IP-> Daemons&Protocols-> NTP-> Servers-> Add` to add a clock server for this system:

Server Type: Select `server`, `peer`, or `broadcast`. Selection server, recommended in most cases, indicates a one–way update path: this system updates its clock from the clock server whose address you enter in the next query, but not the converse. Selection `peer` indicates a two–way update path: the clock server may update its clock from this system should its usual NTP servers be unavailable.

Selection `broadcast` indicates that this system is an NTP server which updates the clocks of the NTP clients whose broadcast address is specified in the next prompt. Note: on each NTP client,

change the **Broadcast Client** prompt to yes, as explained in "Get or Set xntpd Parameters," below. Otherwise, the client will ignore the broadcast.

Address: If **Server Type** is `server` or `peer`, enter the IP address of the clock server — the host from which this host updates its clock. If **Server Type** is `broadcast`, enter the IP broadcast address of NTP clients whose clocks this NTP server should update. Note: your entry is not edited for syntax or checked for validity.

NTP Version: Select the NTP version. This tells **xntpd** which packet version to send to the clock server. The default is probably correct, but if you experience problems, try an earlier version.

Comment: Optionally enter a comment, such as the host name of the clock server. The current date is entered if you leave the field blank.

Modify a Clock Server

Select `Networking-> TCP/IP-> Daemons&Protocols-> NTP-> Servers-> Modify` to change a clock server record:

Enter selection: Select the record to modify. You are then given the opportunity to change any of the items explained above.

Delete a Clock Server

Select `Networking-> TCP/IP-> Daemons&Protocols-> NTP-> Servers-> Delete` to remove a clock server record:

Delete Server: Select the record to remove.

Start or Stop xntpd

Select `Networking-> TCP/IP-> Daemons&Protocols-> NTP-> Start` to specify startup options for the NTP daemon, **xntpd**:

Start when: Select `start now and on reboots`, `start now`, or `start on reboots`.

Select `Networking-> TCP/IP-> Daemons&Protocols-> NTP-> Stop` to stop **xntpd**:

Stop when: Select `stop now and don't restart on reboots`, `stop now`, or `don't restart on reboots`.

Get or Set xntpd Parameters

Select `Networking-> TCP/IP-> Daemons&Protocols-> NTP-> Parameters->Get` to find out which **xntpd** startup options are set.

Select `Networking-> TCP/IP-> Daemons&Protocols-> NTP-> Parameters-> Set` to specify additional or different options. The options, explained below, take effect the next time **xntpd** is stopped and started.

Precision: Enter an integer representing the base 2 logarithm of the local time–keeping precision in seconds, or accept the displayed default value.

Drift File: Enter the name of the file that records drift (frequency errors) or accept the displayed default.

Statistics File: Enter the name of a file for recording cumulative measurement statistics. Leave the field blank (the default) to disable the keeping of statistics. If you enter a filename, **xntpd** creates a new file approximately once a day, by appending a number *.n* to the filename you enter. With each valid clock update, a line is appended to the file showing: the modified Julian date and time (seconds past UTC midnight), peer address and status, offset, delay, and dispersion.

Loop Statistics File: Enter the name of a file for recording loop filter statistics. Leave the field blank (the default) to disable the keeping of loop statistics. If you enter a filename, **xntpd** appends a line to the file with each valid clock update showing the modified Julian date and time (seconds past UTC midnight), the offset, drift compensation, and time constant of the loop filter.

Monitor: Indicate whether **xntpd** should record the IP addresses of received packets and whether the packet originated from a server port. The default is no. Use **xntpdc** to view monitored traffic.

Broadcast Client: Indicate whether this system is an NTP client whose server updates its clock via broadcasts to its subnet address. (See "Manage NTP" and "Add a Clock Server," above.) The default is no.

Broadcast Delay: Estimate the time, in seconds, for a round trip between this system and the clock server. The default is 0.008 seconds.

Key File: Enter the name of the file containing encryption keys used by **xntpd** and **xntpdc**,or accept the displayed default.

Request Key: Enter a number (an unsigned 32–bit integer) for verifying runtime re–configuration changes. Leave the field blank to disallow runtime re–configuration.

Control Key: Enter an encryption key number for verifying mode 6 control messages (for example, setting leap second indications in a server equipped with a radio clock). Leave this field blank if **xntpd** should ignore control messages.

Specify xntpd Restrictions

Select `Networking-> TCP/IP-> Daemons&Protocols-> NTP-> Restrictions` to set behavior ranges for **xntpd**:

Figure 3–20 Restriction maintenance procedures

Restrictions are optional. None are set by default.

List Restrictions

Select `Networking-> TCP/IP-> Daemons&Protocols-> NTP-> Restrictions-> List` to see any current **xntpd** restrictions.

Add Restrictions

Select `Networking-> TCP/IP-> Daemons&Protocols-> NTP-> Restrictions-> Add` to specify one or more **xntpd** restrictions:

Address: If you are adding a restriction that should apply to a particular NTP system, enter its IP address. If the restriction applies to multiple NTP hosts, enter a dot–format address that, when and'ed with **Address Mask**, will resolve to the addresses of those hosts. Note that addresses are not checked for correct syntax or validity.

Address Mask: Enter a dot–format mask that, when and'ed with **Address**, indicates the extent of a restriction. The default mask (255.255.255.255) indicates that **Address** is the IP address of a single host. Masks are not checked for correct syntax.

Restrictions: Select one or more restrictions to add. The restrictions, which apply to matching host(s) are:

`ignore` Provide no services.

`nomodify` Ignore requests to modify the runtime configuration.

`noserve` Ignore clock update service but allow queries.

`noquery` Ignore queries but perform clock update functions.

`notrap` Do not provide control message trap service.

`nopeer` Do not allow matching hosts to perform peer services.

`notrust` Do not allow matching hosts to perform synchronization services.

`lowpriioritytrap`
Assign low priority to trap requests. (Instead of acknowledging traps on a first come first serve basis, allow trap requests from matching hosts to be overridden by subsequent requests from other hosts.)

`ntpport` Apply this restriction only if the source port specified in a packet is the standard NTP UDP port (123).

Modify Restrictions

Select `Networking-> TCP/IP-> Daemons&Protocols-> NTP-> Restrictions-> Modify` to change a restriction:

Enter selection: From the displayed list of restrictions, select the one you want to modify. You are then allowed to enter a new address, mask, or restriction as explained above.

Delete Restrictions

Select `Networking-> TCP/IP-> Daemons&Protocols-> NTP-> Restrictions-> Add` to remove a restriction:

Delete restriction for IP address: From the displayed list of IP addresses, select the restriction you want to remove.

Manage BOOTP

The protocol BOOTP allows a diskless client (a workstation or X terminal, for example) to find out a great deal of information about itself from the system (OS server) it boots from. This service is implemented by the **bootpd** daemon. On OS servers that support diskless clients, **bootpd** is managed by the internet daemon **inetd**, which starts **bootpd** on behalf of clients when requested.

IMPORTANT: While an AViiON OS server can provide BOOTP service for its clients, AViiON diskless clients do not support BOOTP. To find out whether a diskless client on your LAN supports BOOTP, see its documentation.

- BOOTP servers and clients must be on the same network or subnet.

  BOOTP is provided with DG/UX TCP/IP but is not configured by default. To see the BOOTP setup procedures, select `Networking-> TCP/IP-> Daemons&Protocols-> BOOTP-> BOOTP Client:`

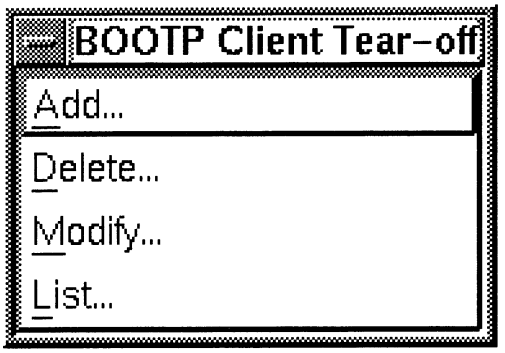


Figure 3–21 BOOTP client maintenance procedures

These procedures display or update the bootp startup file, `/etc/bootptab`. Perform these procedures on the BOOTP server.

List BOOTP Clients

Select `Networking-> TCP/IP-> Daemons&Protocols-> BOOTP-> BOOTP Client-> List` to see the BOOTP clients currently supported by an OS server.

Add a BOOTP Client

Select `Networking-> TCP/IP-> Daemons&Protocols-> BOOTP-> BOOTP Client-> Add` to add a BOOTP client:

Client Name: Enter the hostname of the client to be added. If this client has already been installed, the two fields after the next will be filled in with current information.

Use Defaults for Advanced Features? Indicate yes (the default) or no. If you indicate no, you are prompted, after entering required configuration information, for additional information.

Client's IP Address: Enter the client's IP address or accept the displayed value.

Hardware Type: Select the network interface (controller) type from the displayed values.

Hardware Address: Enter the built–in hardware address of the controller used for booting. (The hardware address is displayed during power up.)

Template Host: Optionally enter the name of an existing client. If you do, the remaining items of information will be set from the template host's configuration values.

IMPORTANT: The template host mechanism sets the following values for this client, but the values do not appear here. Any values you enter here override those specified in a **Template Host** record.

Home Directory: Enter the client's home directory (the directory on the **Boot Server** containing the client's second stage bootstrap file).

Boot File: Enter the client's second stage bootstrap file. If you enter a simple filename, the file is presumed to reside in the **Home Directory** specified above.

Gateway: If the client and its **Boot Server** are on different subnets, enter the IP address of a host connecting the subnets. You can enter multiple addresses separated by commas. Leaving the field blank indicates that the client boots from a server located on the same subnet.

Subnet Mask: If relevant, enter the appropriate subnet mask for this client. A subnet mask is required if the **Boot Server** and client are on different subnets.

Boot Server: Enter the IP address of this client's OS server (the host it boots from). The boot server must be either (1) located on the same subnet as the client or (2) reachable via a **Gateway** host specified above.

If you are entering advanced values (you indicated no at the prompt **Use Defaults for Advanced Features?** above), you are prompted for eleven additional items of information, including:

Time Offset: Optionally enter an offset, in seconds, from UTC time.

Reply Style: Select `rfc1048` (the style defined in RFC 1048), `cmu` (the style defined by Carnegie Mellon University), or `auto` (variable depending on the client's request).

Return hostname to the client? Indicate yes (default) if BOOTP replies should include the client's hostname, no otherwise.

The other advanced items of information, blank by default, allow you to enter the IP addresses of eight servers that may be relevant to BOOTP operation in some LAN environments. See **bootpd**(1M) for descriptions of advanced items.

Modify a BOOTP Client

Select `Networking-> TCP/IP-> Daemons&Protocols-> BOOTP-> BOOTP Client-> Modify` to change a BOOTP client:

Enter selection: Select the record to modify. You are then given the opportunity to change any of the items explained above.

Use Defaults for Advanced Features? Indicate yes (the default) or no. If you indicate yes, you are allowed to change required configuration information, from **Client's IP Address** to **Boot Server**. If you indicate no, you are given the opportunity to changes all items.

Delete a BOOTP Client

Select `Networking-> TCP/IP-> Daemons&Protocols-> BOOTP-> BOOTP Client-> Delete` to remove a BOOTP client:

Enter selection: Select the record to remove.

Start or Stop bootpd

The **bootpd** daemon is managed by the internet daemon, **inetd**. To enable **bootpd**, such that **inetd** starts it whenever requested by a BOOTP client, select `Networking-> TCP/IP-> Daemons&Protocols-> BOOTP-> Start`. This adds the **bootpd** startup command to the **inetd** configuration file, `/etc/inetd.conf`. To disable **bootpd**, select `Networking-> TCP/IP-> Daemons&Protocols-> BOOTP-> Stop`. This removes the **bootpd** startup command from `/etc/inetd.conf`.

Manage SLIP

The Serial Line Internet Protocol (SLIP) allows TCP/IP to be used over serial communication lines. With SLIP, TCP/IP applications such as TELNET and FTP can be used from a system over standard phone lines via modems. Unlike standard serial or modem connections, SLIP links support multiple concurrent sessions. SLIP is included with DG/UX but is not configured by default.

A SLIP session rests on a link connecting two systems:

- A client, which initiates the connection. Also called the local or call–out system.
- A server, which receives the connection request. Also called the remote or call–in system, the server is typically connected to a LAN and provides the client with TCP/IP services to itself or to other LAN hosts.

SLIP is implemented by the daemon **slipd**, which runs on the SLIP client and server during a session. Note that the terms client and server relate entirely to a system's role in a SLIP session. A system may perform both roles on different occasions. If it does, you must set it up as both a client and a server, as indicated below.

The client and server may be connected by a special serial cable or by phone lines via modems. On each SLIP server, you must create a port service and port monitor for the TTY line used for SLIP, whether the line is a direct–connect cable or modem. It is not necessary to create a port service or monitor on a SLIP client.

If a client and server are connected by phone lines, a modem must be installed on each. Modem installation is described in *Managing Modems and UUCP on the DG/UX System* (069–000698). To set up the client, you must know the phone number of the server. Also, you must be familiar with the client modem's dial and connect commands. For these, see the modem's manual.

Each SLIP host must have a unique IP address. Before setting up SLIP:

- On each host that will use SLIP, assign an IP address and name to its SLIP interface. SLIP names and addresses identify each host's serial interface: they are in addition to names and IP addresses of the hosts' LAN interfaces.

  When assigning a name to a SLIP host, choose one which indicates that the name refers to the host's SLIP interface. For example, for a host whose primary LAN interface name is **sys23**, choose a name such as **sys23–slip**. This will simplify network maintenance.

- On each SLIP client, to its `/etc/hosts` file, add its own SLIP name and address and those of all SLIP servers the client calls.

- On each SLIP server, to its `/etc/hosts` file, add its own SLIP name and address and those of all SLIP clients that call the server.

A SLIP server may be equipped with several modems, allowing it to support multiple SLIP links. We recommend that you assign a single name and IP address to a server's SLIP interface regardless of the number of modems. Assigning a different name and address pair for each modem works, but imposes unnecessary restrictions and complicates network administration.

The section "Maintain Hosts" in this chapter explains how to add and maintain host names and addresses.

Select `Networking-> TCP/IP-> Daemons&Protocols-> SLIP` to see the SLIP setup procedures:

Figure 3–22 SLIP functions

The Slipdialinfo and Slipusers procedures display or update the files `/etc/slipdialinfo` and `/etc/slipusers`, respectively.

Maintain SLIP Link Information

Select `Networking-> TCP/IP-> Daemons&Protocols-> SLIP-> Slipusers` to see the SLIP link parameter procedures:

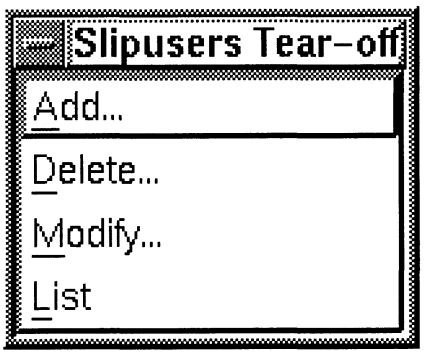


Figure 3–23 SLIP user maintenance procedures

On each SLIP client, add one Slipusers entry for each SLIP server it calls. On each SLIP server, add one Slipusers entry for each SLIP client that calls it. These procedures show or build **slipd** (1M) link initialization parameters. See the man page for reference.

List SLIP Session Information

On a SLIP client, the list procedure shows the SLIP link parameters for each server it can call. On a SLIP server, this procedure lists the link parameters for each client that can call it. Select `Networking-> TCP/IP-> Daemons&Protocols-> SLIP-> Slipusers-> List` to see this information.

Add SLIP Session Information

Select `Networking-> TCP/IP-> Daemons&Protocols-> SLIP-> Slipusers-> Add` to add the information needed to establish a SLIP client/server session. For each link to be established, perform this procedure once on the client and once on the server.

SLIP Parameters Tag: Enter an identifier for the session information. On SLIP clients, this should be the same name as the **SLIP Dialsystem** name in the associated Slipdialinfo record: see "Add a SLIP Dial Command" below.

In choosing this name, follow a convention that is meaningful to you and that reflects the way modem connections are handled in your environment. On a SLIP server, the UNIX login name is a good choice for a server that accepts connections from many client users. On a SLIP client, the host name of the SLIP server (the one you entered in the client's `/etc/hosts` file), or a name containing the server's host name, are good choices for clients that connect to different servers.

IP Packet Header Compression: Indicate whether TCP packet headers should be compressed. Header compression improves the performance of interactive applications. We recommended turning this option ON for SLIP links to servers that support header compression.. In the List report, this option shows up under column Flags as –c (off) or +c (on).

Auto Compression Recognition/Activation: Indicate whether the system should detect and accommodate the header compression preference. ON is recommended for servers. Enter OFF on clients. In the List report, this option shows up under column Flags as –e (off) or +e (on).

Install Default Route via this SLIP Interface: Indicate whether the system should install a default route via the SLIP interface giving it access to all hosts the server can access. YES is recommended for clients. In the List report, this option shows up under column Flags as –r (no) or +r (yes).

Remote Hostname/Address: Enter the IP address or name of the server's SLIP interface, as entered in the local `/etc/hosts` file.

Local Hostname/Address: Enter the IP address or name of the client's SLIP interface, as entered in the local `/etc/hosts` file.

IMPORTANT: Adding SLIP IP address and hostname pairs to the `/etc/hosts` files of SLIP clients and servers is recommended (because doing so simplifies network administration) but not required. If you choose

not to create hostnames for SLIP hosts, you must enter dot–format IP addresses in response to the **Remote Hostname/Address** and **Local Hostname/Address** prompts.

Netmask: Enter the network mask.

Modify SLIP Session Information

Select `Networking-> TCP/IP-> Daemons&Protocols-> SLIP-> Slipusers-> Modify` to change a SLIP session record:

Select SLIP Parameters Tag: Select the record you want to change. You are then given the opportunity to change each of the items explained above.

Delete SLIP Session Information

Select `Networking-> TCP/IP-> Daemons&Protocols-> SLIP-> Slipusers-> Delete` to remove a SLIP session record:

Select SLIP Parameters Tag: Select the record you want to remove.

Maintain SLIP Dial Commands

Select `Networking-> TCP/IP-> Daemons&Protocols-> SLIP-> Slipdialinfo` to see the dial command maintenance procedures:

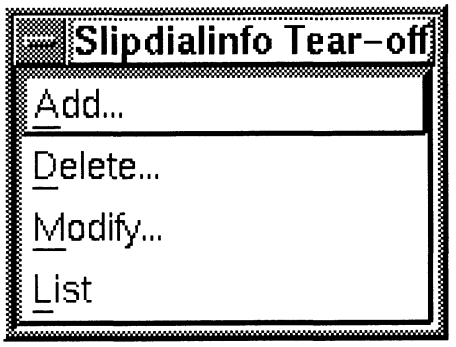


Figure 3–24 SLIP dial command procedures

On each SLIP client, add one Slipdialinfo entry for each modem on each SLIP server you call from the client. These procedures are not relevant for SLIP servers, which do not use dial commands. (However, a host may act as both a client and a server, in which case it must be set up for both roles.)

List SLIP Dial Commands

Select `Networking-> TCP/IP-> Daemons&Protocols-> SLIP-> Slipdialinfo-> List` to see current SLIP dial commands.

Add a SLIP Dial Command

Select `Networking-> TCP/IP-> Daemons&Protocols-> SLIP-> Slipdialinfo-> Add` to add a SLIP dial command:

SLIP Dialsystem: Enter the same name you entered for the **SLIP Parameters Tag** in the associated Slipusers record: see "Add SLIP Session Information" above. When you start a session, you are prompted for this tag. If several tags are defined, you must be able to recognize from this name which server you want to start a session with (which modem for servers that have more than one).

SLIP Dialstring: Enter a command that dials the server's modem, logs in, and starts **slipd**. A command is a series of send and expect strings separated by one or more spaces. For example:

```
atdt5555 CONNECT  \w\w\w\w\w\w\w\w\w\r gin: jones word: 2ulips % exec\sslipd SLIP
```

This command, illustrating the Hayes Smartmodem 2400 commands: dials number 5555; waits for the remote modem to respond with `CONNECT`; delays 5 seconds and sends a carriage return; waits for a login prompt; sends `jones` as login name; waits for a password prompt; sends `2ulips` as the password; waits for the C shell prompt, execs **slipd** from the shell process, and waits for a response of `SLIP` from **slipd** on the server to indicate that it is running.

For the dial command shown in the example (atdt), substitute the command(s) required by your modem (the one installed on the SLIP client). Also, substitute the appropriate phone number, username, and password.

Dialstrings may contain the following escape sequences:

Sequence	Means
\\	\
\n	newline
\r	carriage return
\s	space
\w	1/2 second wait

Modify a SLIP Dial Command

Select `Networking-> TCP/IP-> Daemons&Protocols-> SLIP-> Slipdialinfo-> Modify` to modify a SLIP dial command:

Select SLIP Parameters Tag: Select the record you want to change. You are then given the opportunity to change each of the items explained above.

Delete SLIP Dial Command

Select `Networking-> TCP/IP-> Daemons&Protocols-> SLIP-> Slipusers-> Delete` to remove a SLIP dial command:

Select SLIP Parameters Tag: Select the record you want to remove.

Start and Stop slipd

Start and Stop are for initiating and terminating a SLIP session from the SLIP client. Do not use these procedures from a SLIP server.

To call a SLIP server (or particular modem on a SLIP server), select `Networking-> TCP/IP-> Daemons&Protocols-> SLIP-> Start :`

Select SLIP Dialsystem: From the displayed names, select the server (or modem) you want to connect with.

Select Modem Interface Baud Rate: From the displayed values, select the baud rate appropriate for the modem port.

Select SLIP TTY Port: From the displayed values, select the port used by client's modem or accept the default.

Once started, a SLIP session continues until you stop it (or someone on the server stops it). To stop a SLIP session, from the client, select `Networking-> TCP/IP-> Daemons&Protocols-> SLIP-> Stop`.

Manage Routes

A route is a path between two networks or subnets, and routing is the forwarding of data between different networks or subnets. Routing is performed by routers: hosts (or specialized hardware devices) that are physically connected to multiple networks or subnets that they forward data between.

By default, DG/UX installs routes for the loopback device and for each network interface present on a system. These minimal routes allow hosts to communicate with other hosts that are directly connected to the same physical network or subnet. If this is sufficient, you don't need to build any additional routes. However, if

thosts are to communicate with networks or subnets to which they are not directly connected, you must configure routes.

TCP/IP for AViiON supports two types of routing, static and dynamic. With static routing, you build a routing table on each network host and router, and manually change the tables as network changes require. With dynamic routing, you configure your network to run a program that builds and maintains each host's routing without your intervention. In a few situations, there may be good reasons to use both static and dynamic routing. Usually, however, you choose one or the other.

Static routing would be a good choice for a given host if:

- It is on a physical network or subnet having a single gateway or router connecting it to the outside world.
- You want explicit control over the network(s) a particular system can access.

Where there is more than one possible way for LAN hosts to reach one another, dynamic routing is recommended.

Dynamic routing is performed by routing protocols: programs, which run only on routers (or gateways), that exchange information about routes with one another, and update the routing tables of network hosts. These programs provide the best path to each destination based on network topology. Also, when routers in the network go down, these programs provide alternate routes where possible.

Dynamic routing information is distributed by a daemon, **gated**, running on each host. This daemon replaces an older daemon, **routed**. While **routed** still works, **gated** is recommended because it supports multiple routing protocols, implements superior algorithms for route path fault detection, and provides more flexibility. When you configure dynamic routing with **sysadm**, you edit **gated**'s configuration file.

Currently, **gated** on DG/UX supports three routing protocols: Routing Information Protocol (RIP), Open Shortest Path First (OSPF), and HELLO. The **routed** daemon supports RIP only. Your choice of which protocol(s) to run is based on which are supported by hosts on your network. AViiON hosts running current (1994 or later) revisions of DG/UX support **gated**. Use **routed** only on systems that do not support **gated**.

For additional information about routing, see Chapter 2.

To see the routing functions, select `Networking-> TCP/IP-> Routes:`

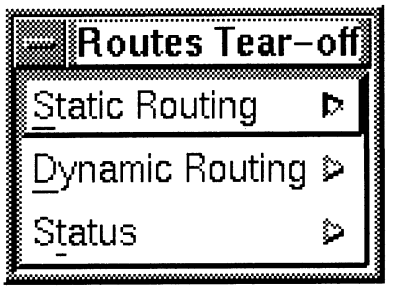


Figure 3–25 Routing procedures

The Static procedures modify or display `/etc/tcpip.params`. The Dynamic procedure modifies the **gated** configuration file, `/etc/gated.conf`.

Display the Routing Configuration

To find out how routing is configured on a host, select `Networking-> TCP/IP-> Routes-> Status-> Configuration`. The resulting report names the type of routing (dynamic or static), which daemon is configured, and whether it is running. If any static routes are configured in `/etc/tcpip.params`, these are listed.

List the Routing Table

To see what is currently in a host's routing table, select `Networking-> TCP/IP-> Routes-> Status-> List`. You can scroll through or print the table, which might be very long. The table contains all current routes regardless of how they were built (whether dynamically or manually).

Each line in the report is a route. The first column indicates whether a route's destination is to a network or to a host on a network. The second column gives the name or IP address of the destination. The third column gives the name or IP address of the network interface on a router or gateway to forward the packet to. The fourth column contains 0 if the destination is directly accessible, or 1 if packets must be forwarded via the gateway to reach the destination..

Maintain Static Routes

To see the static routing functions, select `Networking-> TCP/IP-> Routes-> Static Routing`:

Figure 3–26 Static routing procedures

IMPORTANT: You can add a route to the routing table with the **route** command. In this case, the route becomes effective immediately, but is temporary: whenever the system is rebooted, it must be recreated. You can also add a route by editing the `/etc/tcpip.params` file. In this case, the route is permanent, but does not become effective until you next reboot. Routes built with `Add` take effect immediately (if you indicate they should) and are recreated each time the system boots.

- Building (and maintaining) a static route is a multi–step procedure, depending on the number of physical networks or subnets the route traverses. For example, suppose you manage a LAN with three subnets NET–A, NET–B, and NET–C, and that they are interconnected by two routers: A–B connecting NET–A to NET–B and B–C connecting NET–B to NET–C. To build routes allowing NET–A hosts to reach NET–C hosts, you must build:
  - On each NET–A host, a route to NET–C through router A–B
  - On router A–B, a route to NET–C through router B–C

  To allow NET–C hosts to reach NET–A hosts, you must also build:
  - On each NET–C host, a route to NET–A through router B–C
  - On router B–C, a route to NET–A through router A–B

List Static Routes

To see currently configured static routes, select `Networking-> TCP/IP-> Routes-> Static Routing-> List`. This procedure lists routes specified in `/etc/tcpip.params`. These routes will also appear in the routing table (see above) if:

- You answered yes to the prompt **Apply to currently executing system** when you added or last modified the route

- Regardless of how you responded to this prompt, your system has rebooted since you added or modified the route

Add a Static Route

To add a route to a host's routing table, select `Networking-> TCP/IP-> Routes-> Static Routing-> Add`. Supply the following information.

Host or Network Destination: Indicate whether you are building a route to a network or to a host on a network.

Route Destination: Enter the destination's name or IP address. Your entry is interpreted as either a network or a host, depending on your response to the previous prompt. When specifying a network IP address, you can specify zeroes for the host portion of the address or you can leave off the host portion (for example, either 128.223 or 128.223.0.0 for a class B network; 192.222.1 or 192.222.1.0 for a class C network). If you specify a network name, its name and address must be defined in the file `/etc/networks`.

Route Netmask: If you're building a network route and the destination network does not use a standard byte–aligned network mask, you must enter the hexadecimal network mask here.

Route Gateway: Enter the name or IP address of the router through which the destination is reached. The router must be directly connected to the same physical network or subnet as the host.

Gateway Route Indicate whether packets must be sent to a forwarding host to reach their final destination.

Apply to currently executing system: Indicate whether you want the route to become effective now, or the next time your system is booted.

The items above are the required information for a route, but you are given the opportunity to change the defaults for the following parameters.

Route Trip Time (microseconds):

Route Trip Time Variance (microseconds):

Send Buffer Size (bytes):

Receive Buffer Size (bytes):

Route Maximum Transmission Unit (bytes):

Congestion Threshold:

There are no optimal values for **Route Trip Time** (RTT), **Route Trip Time Variance** (RTTV), and **Congestion Threshold** CT). Rather, these values act as seeds for new TCP connections. RTT and RTTV should only be set in the event of an unusually long network delay. CT should only be set to accommodate unusual circumstances, such as a gateway with few resources.

Send Buffer Size and **Receive Buffer Size** initialize the TCP buffer sizes. If set to 0, DG/UX sizes the buffers, typically to six times the Maximum Segment Size (MSS). Increasing the buffer sizes may improve throughput over high bandwidth/high latency network links (satellite). Reducing the buffer sizes may reduce gateway congestion and system resouce depletion.

Route Maximum Transmission Unit (MTU) is a hint to TCP regarding the connection Path MTU. By default, TCP scales the MSS based on the interface MTU for peers on the same network, and sets the MSS to 512 otherwise. Setting the Route MTU overrides this default behavoir. A smaller Route MTU can help in cases where a network host or bridge is unable to handle a full size packet. A larger MTU can improve throughput to non–local destinations if the actual Path MTU is larger than 512 bytes (as, for example, on Ethernet networks connected by gateways).

Modify a Static Route

To modify a route, select `Networking-> TCP/IP-> Routes-> Static Routing-> Modify`. You are prompted:

Select Route: Select the routing table entry you want to modify.

Apply to currently executing system: Indicate whether you want the modified route to become effective now, or the next time your system is booted.

You are then allowed to change any of the values described above under "Add a Static Route."

Delete a Static Route

To delete a route, select `Networking-> TCP/IP-> Routes-> Static Routing-> Delete`. You are prompted:

Select Route: Select the routing table entry you want to delete.

Apply to currently executing system: Indicate whether you want the route to be deleted now, or the next time your system is booted.

When prompted, confirm your intention to delete the route.

Configure Dynamic Routing with gated

IMPORTANT: This procedure is for systems that support **gated**. If your network includes hosts that should participate in dynamic routing but which do not support **gated**:

- On systems that do not support **gated**, start **routed** as explained in "Maintain Daemons" above. See **routed**(1M) for reference.
- On other systems, enable RIP and start **gated** as explained below.

To configure your network for dynamic routing, on each host, select `Networking-> TCP/IP-> Routes-> Dynamic Routing-> Configure:`

Configuration file: Enter the name of the file to edit or accept the default, `/etc/gated.conf`.

Editor: Enter the pathname of your preferred text editor or accept the default, `/usr/bin/vi`.

Edit the file to indicate how **gated** should behave on this system. When you have finished entering your changes, save and exit from the file. You are prompted:

Install the new configuration file: Indicate whether the configuration file you have modified should replace `/etc/gated.conf` or be discarded.

Send signal to reconfigure gated: Indicate whether **gated**, if currently running, should be reinitialized from the new configuration file.

The released **gated** configuration file (`/etc/gated.conf`) serves as a template. The statements in the file, commented out by the # character, illustrate correct syntax. As indicated, statements must end with a semi–colon and must be enclosed inside nested curly braces. Names enclosed between angle brackets indicate arguments. Square braces indicate optional arguments. All other names are keywords. Most keywords are optional, but when present, they must appear as shown. For an explanation of **gated** configuration language, see **gated–config**(4M).

There are three active lines in the default configuration file:

```
rip on ;
hello off ;
ospf off ;
```

If your LAN runs only RIP, this default configuration will probably be sufficient on most systems. If your LAN runs OSPF, some configuration will be required on each network host.

See the section "Dynamic Routing" is Chapter 2. This section presents a sample LAN and illustrates how you could configure **gated** for RIP or OSPF on several systems shown in the example.

Start or Stop gated

Select `Networking-> TCP/IP-> Routes-> Dynamic Routing-> Start` to start the dynamic routing daemon **gated** and add the startup command to `/etc/tcpip.params`.

Select `Networking-> TCP/IP-> Routes-> Dynamic Routing-> Stop` to stop **gated** and remove its startup command from `/etc/tcpip.params`.

Set Up Monitoring with tcpdump

The **tcpdump** program monitors protocols based on the contents of protocol files present in the file `/usr/etc/tcpdump.d/pf`. To maintain this file, select `Networking-> Monitor-> Protocol Packages:`

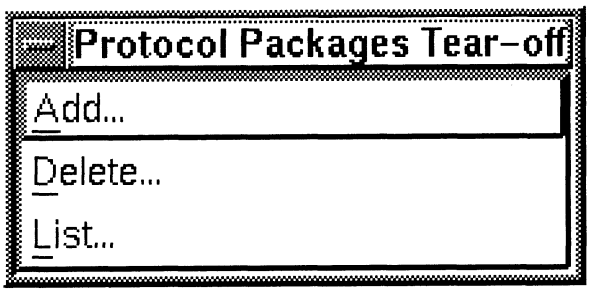


Figure 3–27 Monitor maintenance procedures

As released, the protocol file contains a single protocol package named `dgux`, designating the TCP/IP protocols that are included with DG/UX. The Protocol Packages procedures allow you to find out which protocol packages tcpdump monitors, add new ones, or remove packages.

List Protocol Files for Monitoring

To see **tcpdump** protocol packages and protocols, select `Networking-> Monitor-> Protocol Packages-> List:`

Package report type: Select:

`packages currently contributing to monitor` to see the names of protocol packages present in `/usr/etc/tcpdump.d` and also being monitored. Unless you have added protocol packages to your system that include **tcpdump** protocol files, the report resulting from this selection shows the single name `dgux`.

`packages eligible but not currently contributing to monitor` to see the names of protocol packages present in `/usr/etc/tcpdump.d` but which have not been added to the **tcpdump** monitor list. Unless you have added protocol packages to your system, an empty report appears indicating that no protocol files other than `dgux` are present.

`protocols which can currently be monitored` to see a list of protocols that **tcpdump** currently monitors. You can choose to see `all` protocols or the protocols in a particular package.

Add a Protocol Package

To add a protocol package to the **tcpdump** monitor list, select `Networking-> Monitor-> Protocol Packages-> Add`:

Package: From the displayed list, select the package to add.

IMPORTANT: The displayed list shows the names of subdirectories in `/usr/etc/tcpdump.d` that are not currently being monitored. If there are none, an error message appears. Any subdirectory in `/usr/etc/tcpdump.d` is presumed to contain valid **tcpdump** information. For any that does not, the Add operation fails.

Delete a Protocol Package

To remove a protocol package from the **tcpdump** monitor list, select `Networking-> Monitor-> Protocol Packages-> Delete`:

Package: From the displayed list, select the package to delete.

IMPORTANT: If **tcpdump** is currently monitoring only one package (the default case), you cannot delete it.

- If **tcpdump** exits with a fatal error, after you've added a protocol package, delete that package. Otherwise, **tcpdump** may be unable to monitor any packages.

Miscellaneous Administrative Topics

The remaining sections in this chapter describe some administrative topics that cannot be managed through **sysadm**.

Editing /etc/protocols

The `/etc/protocols` file contains information about the known protocols on the network. Each protocol has a one-line entry with the following information:

protocol-name `protocol-number` [aliases] [#comments]

Separate each item with blanks or tabs. The protocol number must be in decimal. The pound sign (#) character indicates a comment. Figure 3–28 shows a sample `/etc/protocols` file.

```
ip      0     IP      # internet protocol, pseudo protocol number
icmp    1     ICMP    # internet control message protocol
ggp     3     GGP     # gateway-gateway protocol
tcp     6     TCP     # transmission control protocol
pup     12    PUP     # PARC universal packet protocol
udp     17    UDP     # user datagram protocol
```

Figure 3–28 A Sample /etc/protocols File

Editing /etc/bftp.conf

The `/etc/bftp.conf` file contains an ASCII value that specifies the maximum number of simultaneous Background File Transfer Program (BFTP) transfers allowed from a host. The absence of the file or a value of **0** specifies no limit. If you have a slow link between hosts, edit `/etc/bftp.conf` to limit the number of simultaneous transfers.

Setting BFTP's Working Directory

The Background File Transfer Program (BFTP) maintains a working directory for control files and request files. Usually, the working directory is **$HOME** (as set by **sh** or **csh**), but you can use another by setting the environment variable **$BFTPDIR** to the desired directory. Using a separate directory for BFTP files avoids clutter and name confusion. Also, keeping BFTP files in a directory that is not normally mounted by another host helps avoid problems caused by simultaneous access to the same directory through NFS. You can avoid some of these problems if you run **lockd** on all

systems that access the directory through NFS, but you should also set up **$BFTPDIR**. For more information about **lockd**, see *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System*.

You can set **$BFTPDIR** globally in `/etc/profile` or in `/etc/login.csh`. A suggested value for **$BFTPDIR** is `/var/spool/bftp/`*user*. Permissions should allow *user* at least write and execute access.

BFTP Files

BFTP creates temporary files associated with your request (files that end with **.atjob**, **.msg**, **.cmd**, **.list**, and **.req**). Should something go wrong during the execution of your BFTP request, you may have to remove these files manually.

BFTP lets you create request files that contain BFTP transfer parameters used by request commands. BFTP maintains other files such as **bftp_saved_info** and **bftp-save.***, which you can leave alone. For more information about creating request files and about BFTP in general, see *Using TCP/IP on the DG/UX™ System*.

Configuring ftpd

You can configure the File Transfer Protocol (FTP) daemon, **ftpd**, with command arguments or with customizing files. The optional arguments and files are described below. See **ftpd**(1M) for additional information.

ftpd Command Options

The characteristics you can configure with command switches are:

–d Enable debugging. This option specifies that detailed debug statements should be logged to a per–session log file located in the `/tmp` directory.

–l Log each **ftp** session to the system log. Each connection, disconnect, login, get, put, mkdir, rmdir, delete, and rename operation will be logged via **syslog**(1M) at level LOG_INFO.

–s *nses* Limit the number of concurrent **ftp** connections for any given username to *nses* sessions. The default is no limit. Username **root** is excluded. If *nsec* is exceeded at the time a user executes **ftp**, this message appears:

```
530 Session limit exceeded...try again later.
Login failed.
```

–t *tsec* Set the inactivity time–out period (default 900) to *tsec* seconds. If no activity is detected within *tsec* seconds, **ftpd** terminates. After timeout, any subsequent attempt to use **ftp** results in a message like the following:

```
421 Timeout (900 seconds): closing control connection.
```

The **ftpd** daemon is started on request by **inetd**. To add or modify an **ftpd** command switch, use `Networking-> TCP/IP-> Daemons&Protocols-> Other Daemons-> Modify`: see "Modify a Daemon" for instructions.

ftpd Customizing Files

The optional **ftpd** customizing files are described below.

/etc/ftpwelcome This file may contain a global welcome banner for the system, presented to a user of **ftp** prior to login.

.ftpbanner If this file exists in the user's **$HOME** directory, its contents are displayed after successful login. Additionally, where the file is present in a directory, its contents are displayed as the user **cd**'s into that directory.

/etc/ftpbanner If this file exists and **$HOME**/.ftpbanner does not exist, this file is displayed as the global system banner. Note: access restrictions prevent this file from being displayed for restricted and anonymous ftp users.

/etc/ftppswd This file may contain customized password prompts for any user accounts, except for the anonymous FTP account.

/etc/ftpanonpswd This file may contain a customized password prompt for the anonymous ftp account.

/etc/ftpd.rest This file may list the *usernames*, one per line, of users who should have restricted FTP access. Upon successful login, a **chroot**(1M) is performed on the $HOME directory of a restricted user.

Usernames listed in this file must have valid username/password accounts. Also, the **$HOME** directory of each restricted user must contain a directory structure to support restricted access, such as $HOME/bin/ls, $HOME/bin/pwd, $HOME/etc/group. Otherwise, upon executing **ftp**,

a restricted user will receive the error message `550 Bad account.`

/etc/ftpd.allow — This file may list the names of users, hosts or networks allowed to access this host's FTP server. If this file does not exist, all users from all hosts and networks, except those listed in `/etc/ftpd.deny`, are allowed access.

The format of lines in this and the deny file (see below) is:

username [network [netmask]]

For example, in the following lines, the first allows anonymous FTP access from anywhere, and the second restricts FTP access to the network 128.223.0.0 for all other accounts:

```
ftp
+ 128.223.0.0 255.255.0.0
```

/etc/ftpd.deny — This file may list the names of users, hosts or networks that are denied access to this host's FTP server.

Customizing files are clear–text files, such as those created by **vi** or other text editors.

IMPORTANT: To ensure security, restrict the access of customizing files.

Using Proxy ARP Routing

The Address Resolution Protocol (ARP) translates between software (IP) addresses and hardware (Ethernet, token ring, or FDDI) addresses. Since all LANs support ARP, you can use the ARP table:

- To hide the existence of subnets from hosts running software that does not support subnetting
- To allow a diskless workstation (OS client) to boot from a server (OS server) that is located on a different subnet

When you use ARP in this way, it is called proxy ARP.

When sending a datagram to a destination that is located on the same network or subnet, a host looks in its ARP table for the destination's hardware address. If the destination's address is present, the sending host puts the datagram directly on the Ethernet, token ring, or FDDI network along with its address. If the address is not present in the table, the sender broadcasts an

ARP request asking the host having the specified IP address to return its hardware address. Based on the destination's reply, the sender puts the addressed message on the network. Also, the sender updates its ARP table so that in the future it can send datagrams to this destination without delay.

The ARP mechanism works only with hosts located on the same physical network or subnet. Should a host issue an ARP request to a host located on a different subnet, the destination host will not receive it. This in fact is what happens when a host that does not support subnets attempts to send a datagram to a host located on a different subnet. Whereas a host that supports subnetting uses the network mask to distinguish between local and distant addresses, and forwards datagrams with distant addresses to a router based on its routing table, a host that does not support subnetting uses the ARP mechanism in both cases. With distant addresses, the mechanism fails.

The proxy ARP work around for this problem is to set up a subnet router to act as a proxy for distant hosts. To illustrate, suppose that, in Figure 3–29:

- Host **sys03**, located on subnet 128.223.1, does not support subnets
- Host **sys03** wishes to reach host **dg1** located on subnet 128.223.2

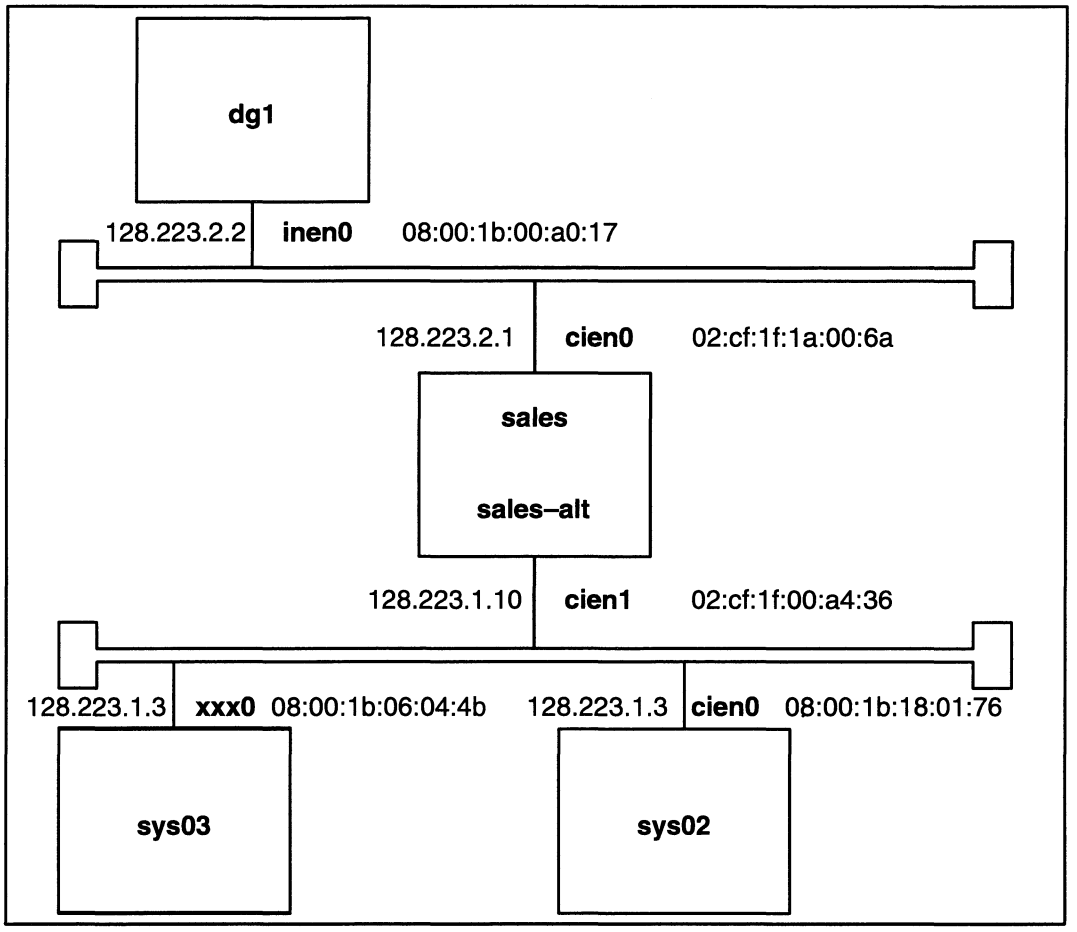


Figure 3–29 Proxy ARP illustration

If you do nothing, **sys03** will broadcast for the Ethernet address corresponding to **dg1**'s IP address (128.223.2.2) and receive no reply. So you set up **sales** to reply to **sys03**'s announcement with its own Ethernet address. To do this, on host **sales**, issue this command:

```
# arp -i cien1 -s dg1 02:cf:1f:00:a4:36 pub ↵
```

This command adds what is called a *published entry* to **sale**'s cien1 interface. Now, when **sys03** asks "will the host with Internet address 128.223.2.2 please tell me what its Ethernet address is," **sales** replies "here I am; IP address 128.223.2.2 is Ethernet address 02:cf:1f:0:a4:36." Based on this reply, **sys03** sends **dg1**'s datagram to **sales**. Using its routing table, **sales** then delivers the datagram to **dg1**. Also, **sys03** updates its ARP table. After the update, if you display **sys03**'s ARP table (type **arp –a**), it contains the following entries:

```
Current ARP table entries for device xxx0

Hostname  IP Address    Hardware address  Status
sales     128.223.1.10  02:cf:1f:00:a4:36 temporary
dg1       128.223.2.2   02:cf:1f:00:a4:36 temporary
```

This section has described a case where the routing tables are insufficient to accomplish your routing requirements. Generally, you should use the routing tables rather than proxy ARP.

Administering Diskless Systems

This section covers some additional administration topics specific to diskless systems.

Running Software That Listens to Broadcasts

There is a special situation that occurs with any application that runs on diskless workstations and that listens to broadcast messages. By its nature, a diskless workstation depends upon the network and file servers to load programs and to perform swapping. Running software, such as routing software, that listens to network broadcasts may cause excessive work for diskless workstations for reasons described below.

For example, if there is dynamic routing through **routed** or **gated**, each server on the network might broadcast its routing tables every 30 seconds. The problem with having diskless workstations on a network is that the software to listen to these broadcasts must be loaded over the network. On a busy computer, programs that are not used for a few seconds are swapped or paged out. When they are activated again, they must be swapped or paged in.

Whenever a broadcast is sent, every computer on the network activates the routing software to process the broadcast. This means that many diskless workstations are swapping or paging at the same time, causing additional network overhead. Thus, it is unwise for diskless workstations to run software that requires them to listen to broadcasts (for example, **routed** or **rwhod**). We recommend you make the server the default router for those diskless workstations.

Route and Interface Management

Diskless systems rely on the network to support NFS access for their **root**, **usr**, or **swap** file systems. This access is created initially at boot time, using information from **bootparamd**, and must always remain open.

To support this requirement, network shutdown takes a conservative approach and does not stop a network interface being used for diskless access. A shutdown also does not flush the routing tables, because that could remove a route being used to access a file server on a different physical network.

This design produces the following results:

- When the network is started by a run level change (such as **init 3**), an `already configured` message from the network interface is normal. Do not stop this interface with **sysadm**, **admipinterface**, or **ifconfig**.
- When the network is stopped and restarted, the addition of permanent routes from **/etc/tcpip.params** produces `invalid argument` error messages from the **route add** operations. These messages appear because a conflicting (and usually identical) route is already present.

The best way to do manual route management on diskless systems is to first start the network, and then use **sysadm** or **admroute** on permanent routes only. Be careful not to accidently delete any route carrying your **root**, **usr**, or **swap** file system to a different network.

Setting RARP Information on the Server

You do not have to set RARP information manually. The DG/UX startup scripts automatically invoke the **initrarp**(1M) command. This command searches the **/tftpboot** directory, and for every entry in **/tftpboot**, it finds the hostname that corresponds to the Internet address and the Ethernet address of that host. **initrarp** then places the Ethernet-Internet address pairs in the server's ARP table.

RARP uses the ARP table to maintain Ethernet-to-Internet address translation information for all diskless clients.

Restarting the Network

TCP/IP for AViiON Systems lets you restart the network without rebooting the system. To do this, become the superuser and the following commands:

```
%su↵
Password: (enter the superuser password)
# /usr/sbin/init.d/rc.nfsfs stop↵ (only if running NFS)
# /usr/sbin/init.d/rc.tcpipserv stop ↵
# /usr/sbin/init.d/rc.nfsserv stop ↵ (only if running NFS)
# /usr/sbin/init.d/rc.nfslockd stop ↵ (only if running NFS)
# /usr/sbin/init.d/rc.ypserv stop ↵ (only if running NIS)
# /usr/sbin/init.d/rc.tcpipport stop
# /usr/sbin/init.d/rc.tcpipport start ↵
# /usr/sbin/init.d/rc.ypserv start ↵ (only if running NIS)
# /usr/sbin/init.d/rc.nfslockd start ↵ (only if running NFS)
# /usr/sbin/init.d/rc.nfsserv start ↵ (only if running NFS)
# /usr/sbin/init.d/rc.tcpipserv start ↵
# /usr/sbin/init.d/rc.nfsfs start ↵ (only if running NFS)
```

These command lines invoke several **rc** scripts. An **rc** script is a file that sets parameters and default values for a command, and establishes an environment or a configuration in which a command can run. The benefit of using an **rc** script is that you do not have to remember command formats. The disadvantages are that you have to remember the order in which to run the scripts, run the scripts from the system console, and restart all the network devices, rather than the one that appears hung. If you decide to run the **rc** scripts, run them in the order specified above.

IMPORTANT: As the lines above suggest, if you are running NIS, you must stop it before restarting the network. If you do not stop NIS, the host lookup for the **ifconfig** command that runs in **rc.tcpipport** may fail.

It is meaningless to run **/usr/sbin/init.d/rc.tcpipport stop** on a diskless client, where you must reboot the system to restart the network.

Using NIS with TCP/IP

The Network Information Service (NIS) is a distributed lookup service that provides access to a set of replicated databases. You set up the NIS when you set up DG/UX ONC/NFS. The primary purpose of NIS is to simplify administration of the following TCP/IP network databases:

/etc/aliases
/etc/ethers
/etc/group
/etc/hosts
/etc/networks
/etc/passwd
/etc/protocols
/etc/services

NIS databases in **/etc/yp** contain all of the above information for an entire network. For more information about the NIS, see *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System*.

When you set up TCP/IP, you set up the TCP/IP network databases on the host. If you intend to run the NIS, these files should minimally contain the information shown in Table 3–2.

Table 3–2 Minimal Contents of Essential Network Files

File	OS Server	OS Client
/etc/aliases	Alias for mailer daemon (**MAILER_DAEMON: root**), alias required by *RFC 822* (**postmaster:root**), and aliases to handle mail to msgs and news (**nobody:/dev/null**)	Alias for mailer daemon (**MAILER_DAEMON: root**), alias required by *RFC 822* (**postmaster:root**), and aliases to handle mail to msgs and news (**nobody:/dev/null**)
/etc/ethers	Ethernet addresses of all OS clients	none
/etc/group	**root**, **daemon**, **mail**, **lp**, and so on; groups unique to server	**root**, **daemon**, **mail**, **lp**, and so on; groups unique to client
/etc/hosts	Address of *localhost*, address of *server*, address of *clients*, address of some remote *host*, addresses of hosts that are routers	Address of *localhost*, address of *server*, addresses of hosts that are routers

Continued

Table 3–2 Minimal Contents of Essential Network Files

File	OS Server	OS Client
/etc/networks	Network number of *loopback_net,* network number of local network, network number of some remote network, network numbers used for route commands from **rc** scripts	Network number of *loopback_net*, network number of *server*, network numbers used for route commands from **rc** scripts
/etc/passwd	Logins unique to *server*	Logins unique to *client*
/etc/protocols	Protocols as they appear in the prototype file, **protocols.proto**	Protocols as they appear in the prototype file, **protocols.proto**
/etc/services	Services as they appear in the prototype file, **services.proto**	Services as they appear in the prototype file, **services.proto**
/etc/inetd.conf	All daemons specified in the prototype file, **inetd.conf.proto**	All daemons specified in the prototype file, **inetd.conf.proto**

Monitoring Network Services and Traffic

When several physical networks combine to form a single logical network, you may want to establish, whenever practical, a central point where you can connect to each physical network. At this central point, you can put a LAN analyzer on every cable and collect extensive network performance statistics. With these statistics, you can observe how the network is used, whether hosts are using the wrong netmask or wrong broadcast address, and whether bad packets are being delivered. A central point also is more convenient when you run utilities such as **netstat**; you do not have to go to different places to check every network.

Regardless of whether you establish such a central point, there are three things you can do to monitor network services and traffic. These tasks involve maintaining the consistency of network files, periodically checking network status, and periodically checking mail delivery.

Maintaining the Consistency of Network Files

If your network does not run the NIS, you must maintain a consistent copy of important network files such as **/etc/hosts**, **/etc/protocols**, and **/etc/services** on each host. Without the NIS, this is the only way to maintain consistency among the network names, protocols, and services. You could create a shell script that updates files on remote hosts each time the files are changed on the host where they are normally kept. This implies, of course, that you maintain a master copy of network files on a single host. Keeping master files on one host requires that other system administrators make changes on the master host, or ask the administrator of that host to make the change.

You could use **rcp** to copy each of these files on each host. Alternatively, you could use **ftp** or **tftp**, which frees you from the security restrictions of the R commands. These restrictions involve read/write privileges.

Moving network files from host to host can provide an early warning of network troubles. Does the file reach every host you send it to? Does it get the file from some hosts, but not from others? If some hosts do not receive it, begin troubleshooting as described in Chapter 7.

Checking Network Status

Another way to monitor the network is to run the command **netstat -i** each night on each host on the network. Figure 3–30 shows sample output from **netstat -i**:

```
Name    Mtu   Network        Address     Ipkts   Ierrs  Opkts  Oerrs   Collis
cien1   1500  128.223.1.10   sales       6171    0      3081   0       91
cien0   1500  128.223.2.1    sales-alt   44901   0      50287  0       9309
loop0   4136  127.0.0.0      localhost   76      0      76     0       0
```

Figure 3–30 Sample Output from netstat -i

Over time, you would collect statistics regarding input errors (`Ierrs`), output errors (`Oerrs`), and collisions (`Collis`). You could analyze these statistics to detect whether a troublesome trend is developing. If a traffic problem is developing over the entire network (every host shows collisions), consider subdividing an existing subnet. If only a single host has a problem, check that host's transceiver, communication board, or both.

Alternatively, you could run **netstat -i** on each router in the network to obtain information about the use of subnets. For more details about how to use the **netstat**(1M) command, see the manual page and Chapter 7.

Checking Mail Delivery

A third way to monitor the network is to check mail delivery on each host regularly. Are mail messages to a particular address consistently misdirected? How long do messages take to arrive? Do the mail queues clog? You can check the current contents of the mail queue with the **mailq** command. It may take several days (by default, three) for mail to be rejected. If the network has a gateway to outside networks, that gateway may be the best place to change the interval for the mail queue, because outside connections are often prone to problems.

End of Chapter

4 Configuring and Using sendmail

This chapter tells you how to configure and use the **sendmail** program on DG/UX systems.

The UNIX mail system has a layered design, and different programs implement the different layers. At one layer, **mailx** provides a friendly user interface for sending and receiving mail, and at another, **sendmail** provides a sophisticated mail transport service. A layered design lets programmers develop different user interfaces while still making use of the common transport mechanism provided by **sendmail**. Because it has been written as a transport mechanism for any compatible user interface, **sendmail** is easy for programs, but not necessarily users, to use to send mail.

Understanding sendmail

Normally, you do not use **sendmail** to compose messages. Instead, you compose mail messages with user programs such as **mailx**. After initially processing a message, such a program gives it to **sendmail**. The **sendmail** program routes and forwards the message using the Simple Mail Transfer Protocol (SMTP). By examining the recipient address, **sendmail** determines whether the intended recipient or recipients receive mail on the local host or on another host. If necessary, **sendmail** rewrites the address of the message using special rules. These rewriting rules are described in detail later in the chapter.

This address interpretation and rewriting is quite flexible and is controlled by the configuration file, which is named **sendmail.cf**. The configuration file also specifies what programs should be used to deliver mail to various destinations.

For **sendmail** to receive mail from other hosts, you must start the **smtp** mail server program (daemon) on your system. You only need to start it once, at boot time. This is provided for in the startup scripts when you set up TCP/IP on your DG/UX system. Once started, the **smtp** daemon listens for messages coming in from other systems.

If the mail recipient is local, **sendmail** invokes the local mailer to place the message in the recipient's mailbox. Usually, the local mailer is **/bin/mail**. If the recipient resides on another host, **sendmail** forwards the message to the **smtp** daemon on the other host. (The **sendmail** program and the **smtp** daemon are the same executable file.) The **smtp** daemon on the other host then examines the recipient address and attempts delivery.

Figure 4–1 shows the path that a message intended for local and remote recipients takes as it flows through the mail system. This figure simplifies the path for delivery of remote mail; **sendmail** may use *UUCP* (UNIX-to-UNIX copy) rather than SMTP to reach the remote host.

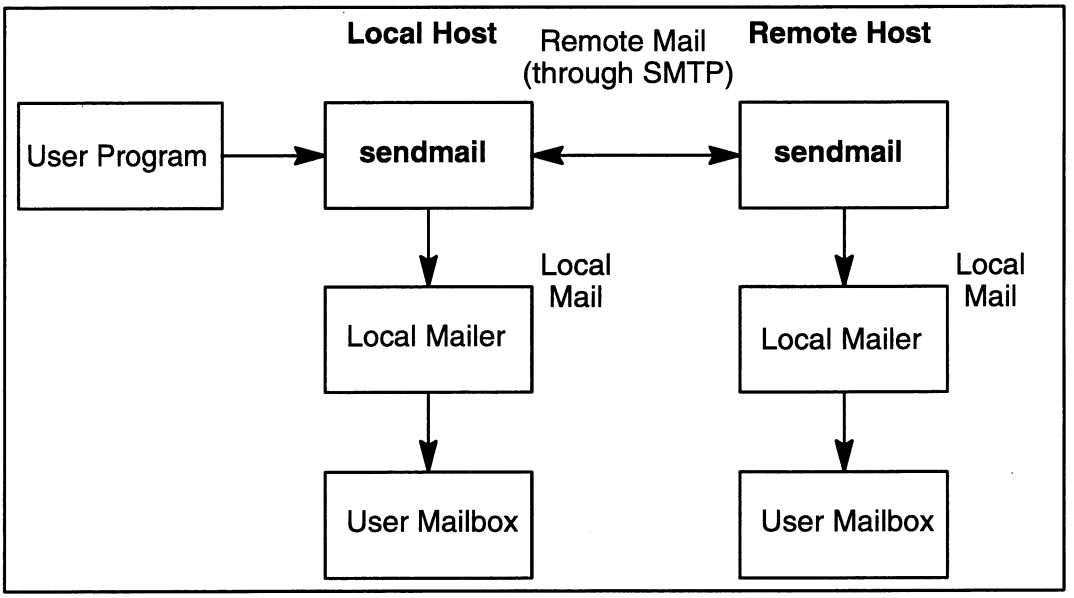


Figure 4–1 Path for a Local and a Remote Message

In addition to mail routing and transport, **sendmail** also supports an alias mechanism. The alias information can either be in a local (per host) file or stored in a global NIS database accessible to all hosts within an organization.

The **sendmail** program guarantees that every mail message is delivered correctly, rejected to the originator, or at least brought to the attention of a "postmaster" who can correctly attend to it.

Configuring sendmail to Use the Domain Name System

You can configure **sendmail** to use the domain name system (DNS) to determine where to route mail on the Internet. To do this, you must edit **sendmail**'s configuration file so that it uses **MX** records. An **MX** record specifies the name of a mail gateway, which is a host that knows how to deliver mail to hosts that are not directly connected to the same physical network as the sending host. When it uses **MX** records, **sendmail** first resolves the recipient address of the receiving host, for example, **tnt.acme.com**. The **sendmail** program then uses the DNS to determine whether there is an **MX** record for **tnt.acme.com**. If an **MX** record for that host exists, **sendmail** sends the mail message to the mail gateway specified on

the record. Otherwise, **sendmail** sends the mail message to the receiving host.

If your network is connected to the Internet, you should run the DNS and configure **sendmail** on your mail gateway to request **MX** records. To configure **sendmail** in this way, set the **I** option in the configuration file (for details, see "Defining Options" later in the chapter). See RFC 1123, *Requirements for Internet Hosts — Application and Support*, for a discussion of the requirements for Internet host software. For more information about the DNS in general, see Chapter 5 of this manual.

Files Loaded with sendmail

When you set up TCP/IP for AViiON Systems, the system loads the files described in Table 4–1.

Table 4–1 Files Loaded with sendmail

Filename	Description
/etc/aliases	Text file for alias database
/etc/aliases.dir	Compiled alias database—contains database index
/etc/aliases.pag	Compiled alias database—contains database data
/etc/sendmail.cf.proto	Prototype **sendmail** configuration file.
/etc/sendmail.cf	**sendmail** configuration file
/etc/sendmail.fc	Frozen version of **sendmail** configuration file
/etc/sendmail.hf	Help file for **sendmail** (SMTP)
/etc/sendmail.pid	File that contains the process ID of **smtp**. You can use this file to kill **smtp**; for example: **# kill 'cat /etc/sendmail.pid'**
/etc/mailstats.st	Repository for mail statistics
/usr/bin/dbm	Program to build **dbm** files for use with **sendmail**
/usr/bin/mailq	Executable file to print contents of mail queue—a hard link to the same executable file as **/usr/bin/sendmail**
/usr/bin/mailstats	Program to print accumulated mail statistics
/usr/bin/newaliases	Executable file to rebuild the alias database—a hard link to the same executable file as **/usr/bin/sendmail**. Turns **/etc/aliases** entries into the **dbm** databases **/etc/aliases.pag** and **/etc/aliases.dir**
/usr/bin/smtp	Executable file to run **smtp** server program (daemon)—a hard link to the same executable file as **/usr/bin/sendmail**
/usr/bin/sendmail	Executable file to provide a mail routing facility

Understanding the Configuration File

The **sendmail** program requires a configuration file, which it reads at startup. The configuration file is named **/etc/sendmail.cf** unless you use a frozen form of the file named **/etc/sendmail.fc**. The frozen configuration file is discussed later in this chapter.

The configuration file contains information, tailored for your environment, that **sendmail** uses to handle mail appropriately. You can build a configuration file before or after you set up TCP/IP on the server system.

The **sendmail** configuration file contains information about the following:

- Macros used by **sendmail**
- Options specified for **sendmail**
- Classes of parameters that **sendmail** uses in the rewriting rules
- Procedures **sendmail** uses to assign a priority order to a message
- Trusted users, who are permitted to override the sender address in a message
- Headers that **sendmail** changes or adds to a message
- Mailers, which are services that **sendmail** uses to deliver mail
- Rewriting rules, which **sendmail** uses to parse and change addresses

The configuration file consists of a series of lines, each beginning with a single capital letter that defines how **sendmail** should interpret the rest of the line. Blank lines and comments (beginning with #) are allowed.

Table 4–2 summarizes what you can define in a **sendmail** configuration file and the corresponding letters you use to begin the line of the configuration file. Writing a configuration file from scratch can be difficult. Most often, you build a configuration file by adjusting an existing configuration, or by editing a prototype configuration file.

Table 4–2 Components of a sendmail Configuration File

Component	Letter	Description
Macros	D	This line defines a macro and its values.
Options	O	This line specifies certain options in the configuration file, such as the interval for queue timeouts or the alias file name.
Classes	C, F	This line defines classes of parameters to match tokens in the left-hand side of the rewriting rules. **C** classes require a list as an argument, and **F** classes require a filename as an argument.
Procedures	P	This line defines precedence values which order the priority of messages.
Trusted Users	T	This line list the names of trained users, who are permitted to override the sender address using the **–f** option to the **sendmail** command.
Headers	H	This line defines the text and the format of a header line.
Mailers	M	This line defines a mailer **sendmail** may use, such as TCP or UUCP.
Rewriting Rules	R, S	Rewriting rules translate and route mail. Each line that begins with an **S** begins a new ruleset. Each line that begins with an **R** defines a rule within the current ruleset.

Editing the Configuration File

TCP/IP for AViiON Systems provides a prototype configuration file named **/etc/sendmail.cf.proto**. To use this prototype file as a template, copy it to **/etc/sendmail.cf**, which is the file that **sendmail** reads when it processes messages. Then edit **/etc/sendmail.cf**.

After you have copied the prototype file to **/etc/sendmail.cf**, read the comments in the file to find out what changes you need to make. You typically do not need to make many changes.

In a small number of cases, you may have to edit **/etc/sendmail.cf** more extensively. This more extensive editing involves five steps.

Step 1 Define **sendmail** operating characteristics. This means defining macros, options, classes, precedence, trusted users, and headers.

Step 2 Define the mailers you want and define rewriting rules for each mailer.

Step 3 Modify ruleset 0 to call the new mailers.

Step 4 Test your configuration.

Step 5 After you fashion a configuration that works well for your host, freeze the **sendmail.cf** file for better performance.

Each of these steps is described in detail in the sections that follow.

Take care when you edit the configuration file; **sendmail** may behave unpredictably if the file contains errors. After editing the file, fix all errors, such as typographic mistakes or omitted keywords. Test any rulesets that you change before using them (see "Step 4: Testing Your Configuration" below). You must stop and restart the **smtp** daemon to make it recognize changes to the **sendmail.cf** file.

Step 1: Defining sendmail Operating Characteristics

The first step in editing the **sendmail** configuration file is to define its operating characteristics. This means defining macros, options, classes, precedence, trusted users, and headers.

Defining Macros

A macro is a single instruction that represents a sequence of instructions. When you use a macro name, the system substitutes the macro's contents.

A **sendmail** macro is a single character that, when processed, is replaced by a longer piece of text. Once defined, you can use a macro throughout the configuration file. To change a macro's value throughout the file, you need only edit the definition itself.

The syntax for a **sendmail** macro is as follows:

D*x val*

where **D** tells **sendmail** that the line contains a macro definition, *x* is the name of the macro defined, and *val* is the macro's value.

Suppose you want to use the **D** macro to specify the name of your Internet domain. The following line defines macro **D** to be replaced by **tnt.acme.com**.

DDtnt.acme.com

Once defined, you can reference the macro by putting a dollar sign ($) before the macro name. So **$D**, as defined above, expands to **tnt.acme.com**.

You also can assign a macro the current value of another macro. For example, you can define a macro such as **D***x***$***y***.$***z*; if the macro *y* expands to the value **sys13** and the macro *z* expands to the value **com**, *x* assumes the value **sys13.com**.

When you assign a macro the value of another macro, the assigned macro does not assume that value until the other macro is referenced. For example, the following set of statements cause messages to have the header **Subject:foo** even though **$y** is set after **$x**:

```
Dx$y
HSubject:$x
Dyfoo
```

You can specify conditionals within a macro definition with the following syntax:

D*x***$?***y text***$.**

Here, *x* expands to *text* only if the macro *y* is set.

Optionally, conditional statements can include an else ($ |) clause.

D*x***$?***y texta* **$** | *textb* **$.**

Here, *x* expands to *texta* if the macro *y* is set and expands to *textb* otherwise.

You can use conditionals only in macro definitions. You cannot use them to define options, classes, precedence, and so on.

To ensure that an address conforms to *RFC 822* standards, you can use a special construct, **$!x**, within a macro. When **sendmail** expands **$!x**, it puts any unpermitted characters in double quotation marks.

Here is an example of how you would use the **$!x** construct with the **q** macro (which is described below):

Dq$?***x***$!x$.<$*g***>**

Here, **sendmail** would rewrite the first address, which contains unpermitted blanks, to the second.

John Smith @sys56 <jsmith@sys56.tnt.acme.com>

to

"John Smith@sys56" <jsmith@sys56.tnt.acme.com>

Required Macros

Table 4–3 shows macros that must be defined in your **sendmail** configuration file. These macros are defined in **/etc/sendmail.cf.proto**.

Table 4–3 Required sendmail Macros

Macro	Definition
e	The SMTP entry message
j	The official domain name for your site
l	The format of the DG/UX `"from"` line
n	The name of the mailer daemon (for error messages
o	The set of separators in addresses
q	The default format of the sender address

The text of the **e** macro is sent by **smtp** to any process to which it connects, the first part of the definition of **e** must be **$j**. The **o** macro contains a list of characters that separate tokens in addresses during parsing (for details, see "Defining Rewriting Rules" later in this chapter). The **q** macro specifies how a default address should appear in a message.

Predefined Macros

Table 4–4 shows macros defined by **sendmail**. Use them to define other parameters, such as header lines and options.

Table 4–4 Required sendmail Macros

Macro	Definition
a	The origination date, in RFC 822 format; the time as extracted from the Date: line of the message (if there was one). If no Date: line is found in the incoming message, **$a** is set to the current date. For example: `Tue, 4 Jun 1992 18:43:57 -0400`
b	The current date, in RFC 822 format
c	The current hop count for the message
d	The origination date, in DG/UX (ctime) format. For example: `Tue Jun 4 18:43:57 1992`
f	The sender (from) address
g	The sender address relative to the recipient
h	The recipient host
i	The queue ID
k	Your host's UUCP hostname. Defaults to the value you set for the **$w** macro without full qualification. For example, if **$w** is **tnt.acme.com**, then **$k** becomes **tnt**.
p	The **sendmail** process ID
r	The protocol used
s	The sender's hostname
t	The current time, in seconds, since 1/1/1970
u	The recipient's username (for example, **vance**)
v	The version number of **sendmail**
w	The hostname of your site
x	The full name of the sender (for example, **Jack Vance**)
y	The ID of the sender's tty
z	The home directory of the recipient

Examples of sendmail Macros

The following examples show typical **sendmail** macros and briefly explain each macro's function. These macro lines come from a modified **/etc/sendmail.cf.proto** file. Remember, you must change these macros to meet the needs of you site, and you must define the **e**, **j**, **l**, **n**,**o**, and **q** macros.

`DA$w` — Defines macro `A` to expand to `$w`. In turn, `$w` expands to the hostname of your site. Thus, whenever `$A` appears, the hostname that `$w` expands to replaces `$A`.

`Dj$w.$D`

Defines the macro `j` to expand to `$w.$D`, which in turn expands to *hostname.domain* (refer to the value of the macro `D` above).

`DnMAILER-DAEMON`

Defines the macro `n` to expand to `MAILER-DAEMON`. Macro `n` names the agent that rejects mail. The mailer-daemon is the return address for messages sent by the **smtp** daemon.

`DlFrom $g  $d`

Defines the macro `l` to expand to the format of the `"From"` line. Here, the format is the sender address (`$g`) and the current date (`$d`).

`Do.:%@!^=/[]`

Defines the macro `o` to expand to the separators used in addresses.

`Dq$g$?x  ($x)$.`

Defines the macro `q` to expand to the default format of the sender address. In this case, if the macro `x` is defined, the symbol `$q` is replaced by the string `$g ($x)`. If `x` is not defined, `$q` is replaced by the string `$g`. For example, if `g` expands to `vance`, and `x` is undefined, then the sender address appears as follows:

```
From: vance
```

However, if `x` is defined as `Jack Vance`, the sender address appears as follows:

```
From: vance (Jack Vance)
```

`De$j  Sendmail  $v/$V ready at $b`

Defines the macro `e` to expand to the SMTP entry message. In this case, the message might read

```
sys99.tnt.acme.com. Sendmail
5.64+/5.64-1.1 ready at Tue, 4 Jun 91
00:00:00.
```

Defining Options

After you define **sendmail** macros, you must define specific options in the configuration file. The syntax of the options line is as follows:

O*oval*

where **O** tells **sendmail** that the line defines an option, *o* is the option, and *val* is the option's value. The *val* can be a string, an integer, a Boolean value (with legal values **t**, **T**, **f**, or **F**; the default is **T**), or a *time* in the following format:

numunit[...]

where *num* is an integer and *unit* is either **s** for seconds, **m** for minutes, **h** for hours, **d** for days, or **w** for weeks. For example, you could specify **30m** for 30 minutes, **2h30m** for 2 hours, 30 minutes, and so on.

You set options one of two ways: by inserting an **O** line in the configuration file, or by using **-o** on the **sendmail** command line. For example, to set the **T** (timeout) option to 2 minutes, insert the following line in the configuration file:

OT2m

Thus set in the configuration file, the timeout would be two minutes every time the **sendmail** program runs. To set the timeout on the command line, you would enter the following:

sendmail -oT2m -q ↵

However, setting the timeout on the command line sets it for only that instance of **sendmail**, not every time **sendmail** runs.

The configuration options are as follows.

Option	Description
A*file*	Use *file* as the alias database. The **sendmail** program uses the aliasing mechanism to redirect mail to a different address or set of addresses. For more information about aliasing, see "Using the Aliases Database" later in the chapter. If no file is specified, **sendmail** uses the file **aliases** in the queue directory (See option **Q**, later in this section, for details about the queue directory). If *file* does not begin with a slash (/), **sendmail** interprets the filename relative to the queue directory.
a[*n*]	Specify this option to tell **sendmail** to wait for an `@:@` entry to exist in the alias database before rebuilding the database. If the entry does not appear in five minutes, rebuild the database. Optionally, you can tell **sendmail** to wait for *n* minutes rather than five minutes. For more information about `@:@` entries, see

"Potential Problems with the Aliases Database" later in this chapter.

B*c* Substitute the character *c* for any blank encountered in the address.

c If an outgoing mailer is marked as being expensive, wait for the next time the mail queue is cleared rather than attempt immediate delivery. Mark mailers as expensive with the **e** flag. See "Using Mailer Flags" later in this chapter for details about the **e** flag.

d*x* Deliver in mode *x*. Legal modes are as follows:

- **b** Deliver in background (asynchronously).
- **i** Deliver interactively (synchronously).
- **q** Just queue the message (deliver when clearing the mail queue).

D Rebuild the alias database if necessary and possible. If this option is omitted, **sendmail** never rebuilds the alias database unless explicitly requested through **sendmail -bi** or the **newaliases** command. If a revised alias database is not rebuilt, the changes made to **/etc/aliases** do not take effect.

e*x* Dispose of errors using mode *x*. Legal values for *x* are as follows:

- **e** Mail errors back and give zero exit status.
- **m** Mail back errors to the sender (default).
- **p** Place messages in **~/deadletter** if local sender, otherwise mail back to sender.
- **q** No messages, just give exit status. Undeliverable mail is lost.
- **w** Write errors back to the sender (mail errors back if sender not logged in).

The error mode is **m** unless the **e** option is set otherwise and you are sending mail.

F*mode* The temporary file *mode*, in octal. For example, mode 0644 (owner has read/write permissions, group and others have read permissions) or mode 0600 (owner has read/write permissions, group and others have no permission). The **sendmail** program creates temporary files, located in **/usr/spool/mqueue** by default, that are affected by this option.

f Save DG/UX-style `From` lines at the front of headers. Normally they are assumed redundant and are discarded.

g*id* Set the default group ID for mailers to run to *id*. The **sendmail** program sets the group ID to *id* before running the mailer.

H*file* Specify the file used by SMTP's **help** command. For details about SMTP, see *RFC 821 (Simple Mail Transfer Protocol)*.

I If set, **sendmail** uses the domain name system (DNS) to determine where to route mail on the Internet. If the DNS is not available, **sendmail** queues the mail. If this option is not set, **sendmail** does not attempt to get **MX** records.

Set the **I** option only if you are using the DNS; otherwise, mail will never get delivered.

i Do not interpret a dot character that appears by itself on a line as the end of the message.

K*xname*

or **K***x%map* Declare the keyed database *x* to be associated with the **dbm**(3) file named *name*. Always specify *x* as a single letter. If you follow *x* with a percent sign (%), specify the name of a Network Information Services (NIS) *map* after the percent sign. See "The Right-hand Side of a Rewriting Rule" for an instance of how **sendmail** looks up keys in **dbm** or NIS databases.

L*n* Set the default logging level to *n*. *n* can be 0-10, 12, or 16, with 0 denoting no logging, and higher number denoting increasingly detailed logging.

M*xvalue* Set the macro *x* to *value*. Use this option only from the command line (for example, **sendmail -oMDARPA**; recall that in the configuration file, you use **D***xval* to set macros (for example, **DDARPA**).

m Mail to the sender even if the sender is the expansion of an alias. For example, if you send mail to a mailing list that includes your name, you receive a copy of the mail only if this option is set. Otherwise, **sendmail** removes the sender's name if it is found in the expansion of an alias.

N*netname* Specify the name of the home network. The argument of an SMTP **HELO** command is checked against *hostname.netname*, where *hostname* is requested from the kernel for the current connection. If the argument to **HELO** does not match the name obtained from the kernel, that name is added to `Received:` lines to assist in message tracing.

o Turn on old-style address parsing, which means that within headers, spaces delimit address names. This option actually turns on an adaptive algorithm: if any recipient or sender address contains a comma, parenthesis, angle bracket, or semicolon, **sendmail** turns off the old-style address parsing. If this option is not set, only commas delimit names. Headers are always produced with commas between the names.

Q*directory* Use the named *directory* as the queue directory. By default, the queue directory is **/usr/spool/mqueue**.

q*factor* The **sendmail** program divides the value of *factor* by the difference between the current load average and the load average limit to determine the maximum priority of messages to be sent immediately. When the resulting quotient is less than the priority of the message, the job is queued rather than run immediately. That is, given

$$\frac{\text{factor}}{(\ LA_{current} - LA_{limit}\ ) + 1} = \alpha$$

sendmail queues the message if $\alpha <$ the message priority. The smaller the *factor*, the more likely jobs are queued rather than run immediately. The default *factor* is 10000. Set LA_{limit} with the **x***N* option.

Changing the default *factor* is useful only on a host that handles a large volume of mail (for example, 5000 messages a day).

As reported by **ruptime**(1), the load average reflects jobs that would run if they were given to a processor. Jobs awaiting any event (such as disk I/O) are not included in the load average calculation.

r*time* Set the read timeout to *time*. Specify *time* as the concatenation of a numeric value and a single letter to designate the unit of time. For example, to set the read timeout to 30 minutes, you would specify **Or30m**; to set the read timeout to 2 hours, you would specify **Or2h**. A timeout between 1 and 2 hours is recommended.

It is possible to time out when reading the standard input or when reading from a remote SMTP server, because a program that never returns or an uncooperative server may cause **sendmail** to accumulate many processes. You could set the read timeout to a shorter time (such as an hour), which

reduces the chance of these idle processes piling up on your system.

Unlike other implementations of **sendmail**, this one associates the read timeout with the reception of an entire message.

Even though the SMTP protocol does not specify a read timeout, sites that run **sendmail** should set it. See *RFC 1047 (Duplicate Messages and SMTP)* and *RFC 1123 (Requirements for Internet Hosts — Application and Support)* for more information.

/ Set the split rewriting option, which processes sender and recipient envelope addresses with rulesets 1 and 2 respectively, and sender and recipient header addresses are processed with rulesets 5 and 6 respectively. (You must define rulesets 5 and 6.) This is contrary to the default, when both envelope and header addresses are processed with rulesets 1 and 2.

S*file* Log mail statistics in *file*. The statistics logged are the number of message to and from each mailer and the number of kilobytes transferred to and from each mailer. The default value is **/etc/mailstats.st**. If you use a non-default statistics file, first create an empty file.

s Always create the queue file, even if you are going to attempt immediate delivery. If the **s** option is set, mail delivery is more reliable, but more resources (CPU, disk space) are consumed. Use this option to reduce the chance of mail getting lost due to a system crash. As a rule, you should set this option.

T*time* Set the queue timeout to *time*. Mail that has been in the queue longer than this interval is returned to the sender the next time that the queue is run. The timeout is typically set to three days.

The time of submission, rather than the amount of time left until timeout, is set in the queue. As a result, you can flush messages that have been hanging for a short period by running the queue with a short message timeout. For example, the following line runs the queue and returns anything that is one day old or older to the sender:

```
sendmail -oT1d -q
```

u*id* Set the default user ID for mailers to *id*, where *id* is either a user ID number or a username. For example, you can specify **Oumail** or **Ou8**; if you specify the

former, **sendmail** looks up the password entry of **mail** and uses the uid it finds. Thus, the administrator does not have to keep the uid for **mail** consistent with the **Ou** line in **sendmail.cf**.

To override this, use the **S** mailer flag. See "Using Mailer Flags" later for more information about the **S** mailer flag.

v Run in verbose mode. The **sendmail** program tells you more about what it does as it runs when this option is set. This option may override some other options (for example, the **c** option).

X*n* Set the load average above which to refuse incoming messages to *n*, where *n* is a real number. The default value for *n* is 3.0. The **sendmail** program does not accept any incoming connections until the load average falls below *n*. See also the description of the **q***factor* option.

x*N* Set the load average above which to queue messages. *N* is a real number. The default value for *N* is 2.0. The **sendmail** program queues messages to conserve system resources. See also the description of the **q***factor* option.

Y If set, **sendmail** uses a distinct process to deliver each job that is run from the queue. Use this option if your system has little memory, since **sendmail** uses considerable memory when processing the queue.

If your host handles a large volume of mail, the following options are useful to control mail flow. They directly affect the priority value of a job. The highest priority values are processed last. If your host does not handle a large volume of mail, you probably do not need to change the value of these options.

y*factor* The indicated *factor* (an integer) is added to the priority value (thus lowering the priority of the job) for each recipient. The value of *factor* penalizes jobs with large numbers of recipients. The default value is 1000.

z*factor* The indicated *factor* (an integer) is multiplied by the message class (determined by the **Precedence:** field in the header and the **P** lines in the configuration file) and subtracted from the priority value. The higher the *factor*, the more messages with a higher priority value are favored. The default value is 1800.

Z*factor* The *factor* (an integer) is added to message's priority value every time that **sendmail** runs through the job queue. The default value is 9000.

Examples of sendmail Options

The following examples show how you can define certain **sendmail** options. These examples are taken from a modified **/etc/sendmail.cf.proto**. You should choose options that fit the needs of your site.

`OA/etc/aliases`	Defines option `A`, which is the name of the alias file. Here, the alias file is called `/etc/aliases`.
`Og1`	Defines option `g`, which is the default group ID number for mailers. In this case, the default group ID number is 1.
`OH/etc/sendmail.hf`	Defines option `H`, which is the name of the help file for SMTP. Here, the help file is called **/etc/sendmail.hf**.
`OQ/usr/spool/mqueue`	Defines option `Q`, which is the name of the queue directory. Here, the queue directory is called **/usr/spool/mqueue** (default).
`Os`	Sets option `s` to TRUE. This causes the queue file to be created for all delivery modes.
`OT3d`	Defines option `T`, which sets the queue timeout. Here, the queue timeout is set to three days.

Defining Classes

Define classes for use in the left-hand side of a rewriting rule. Rewriting rules are described in detail later in the chapter. You can create a rewriting rule that tells **sendmail** to check whether an input value corresponds to the member of a class.

For example, the following line defines class **T** to consist of top level Internet domain names.

```
CT mil edu com net gov org
```

The following rewriting rule checks the top level domain name of an address against elements in the class **T**:

```
R$*@$*.$=T      $#inet$@$2.$3$:$1@$2.$3
```

This rule passed the inet mailer `$2.$3`, the second and third elements in a message address. This breaks down as follows:

```
mailer:   inet
host:     $2.$3      (for example: abc.ef.com)
user:     $1@$2.$3   (for example: vance@abc.ef.com)
```

For more information about rewriting rules, see the sections "Defining Rewriting Rules" and "The Left-hand Side of a Rewriting Rule."

Unlike previous versions of **sendmail**, this version lets members of classes contain characters taken from the set of delimiters defined by the **o** macro. The following line defines class **H** to consist of two Internet domain names, each of which are delimited by periods.

```
CH tnt.acme.com unc.edu
```

Defining Precedence

When you define precedence, you tell **sendmail** how to order the priority of message processing. The syntax for the precedence field is as follows:

P*name=num*

where **P** tells **sendmail** that the precedence is being set, *name* classifies a message, and *num* defines the level of precedence. Higher values of *num* mean higher precedence, and negative values mean that messages are not returned to the sender if an error occurs. The default level of precedence is zero.

The following list shows the entries in the prototype configuration file:

```
Pfirst-class=0
Pspecial-delivery=100
Pjunk=-100
```

Here, "special-delivery" messages have higher precedence than "first-class" messages. "Junk" messages are not returned to the sender if an error occurs.

To set a message's precedence, insert the `Precedence:` header and specify a value as defined in the configuration file. For example, a message must contain the following header to be treated as "special-delivery":

```
Precedence: special delivery
```

Defining Trusted Users

Trusted users are permitted to override the sender address by using the **-f** flag. The syntax of the line to define a trusted user is:

T*user1 user2...*

or

T*user1*
T*user2...*

where **T** indicates that the line defines trusted users and *user1* and *user2* are the names of the trusted users.

The following lines in a configuration file define `root`, `daemon`, and `uucp` as trusted users:

```
Troot
Tdaemon
Tuucp
```

Defining Headers

The syntax for defining **sendmail** headers is as follows:

H[?*mflags*?]*hname*: *htemplate*

where **H** defines the line as a header line, *mflags* are optional mailer flags, *hname* is the header name, and *htemplate*, which may contain macros, is the body of the header. *hname* may consist of any printable ASCII character except the colon (:). When specifying a header, you must follow *hname* with a colon. *htemplate* may consist of any ASCII character except CR or new line. You specify mailer flags when you describe the mailers to **sendmail**. For details, see the section "Defining Mailers and Rewriting Rules."

When **sendmail** receives a message, it determines which mailer handles delivery of the message and what mailer flags are set for that mailer. If an *mflag* is specified in the header line definition, and if that *mflag* is a mailer flag specified in the chosen mailer definition, **sendmail** places the specified header line (*hname*: *htemplate*) in the message. Remember that the *mflag* is optional; if you do not use one in the header definition, the specified header always appears in a message.

Examples of sendmail Headers

The H lines below show sample header lines from a modified **/etc/sendmail.cf.proto** file. You can tailor the header lines to your needs.

`H?P?Return-Path: <$g>`	Mailers with the `P` flag set get a `Return-Path:` line, defined here as `$g` (the sender address).

`HReceived: $?sfrom $s $.by $j ($v/$V) id $i; $b`

All mailers get a `Received:` header. See "Defining Macros" earlier in this chapter for definitions of the `s`, `j`, `v`, `i`, and `b` macros. The words "from," "by," and "id" appear on the header line.

`H?D?Date: $a`

`H?D?Resent-Date: $a`

If the mailer has the `D` flag set, **sendmail** inserts a `Date:` line, defined here as `$a` (the origination date) into the message. If this is a resent message, **sendmail** inserts a `Resent-Date:` line.

`H?F?From: $q`

`H?F?Resent-From: $q`

If the mailer has the `F` flag set, **sendmail** inserts a `From:` line into the message. If this is a resent message, **sendmail** inserts a `Resent-From:` line.

`H?x?Full-Name: $x`

Mailers with the `x` flag set get a `Full-Name:` line, defined here as `$x` (the full name of the sender).

`HSubject:`

All mailers get the `Subject:` line.

`H?M?Message-Id: <$t.$i@$j>`

`H?M?Resent-Message-Id: <$t.$i@$j>`

If the mailer has the `M` flag set, **sendmail** inserts the `Message-Id:` line into the message. If the message is resent, **sendmail** inserts the `Resent-Message-Id:` line. See "Defining Macros" earlier in this chapter for definitions of the **t**, **i**, and **j** macros. The characters **@** and **.** are separators that appear in the header line.

Special Header Lines

Some header lines have interpretations built into **sendmail** that you cannot change. These header lines are described below.

Return-Receipt-To: *addr*

If you send a message that contains the **Return-Receipt-To:***addr* header when **sendmail** attempts local delivery (that is, if the **l**

mailer flag is set in the mailer descriptor), **sendmail** sends a message to *addr* that the mail was delivered. (See "Using Mailer Flags" later in the chapter for details about the **l** mailer flag.)

Errors-To: *addr*

If errors occur anywhere during processing, this header causes error messages to go to *addr* rather than to the sender. This header is intended for mailing lists.

Apparently-To: *addr*

If a message comes in with no recipients listed in the message (in a `To:` line), **sendmail** adds an `Apparently-To:` header line for recipients. At least one `To:` line is required under *RFC 822 (Standard for the Format of ARPA Internet Text Messages)*.

Resent Messages

When **sendmail** receives a message that contains a header of the following form, it treats the message as a "resent" message:

Resent-*header*: mumble

The *header* can be `Sender`, `From`, `To`, `Cc`, `Bcc`, `Message-Id`, or `Date`. A mail-sending program (for example, **mailx**) can add any one of these headers when it processes mail. If **sendmail** adds any other header specified in the mailer when it processes a "resent" message, it adds the "resent" version of the header instead of the normal version.

Step 2: Defining Mailers and Rewriting Rules

The second step in editing the **sendmail** configuration file is to define the mailers you want and define rewriting rules for each mailer.

Defining Mailers

A mailer is a service that **sendmail** uses to deliver mail. To get **sendmail** to use a mailer to deliver messages, you must define the mailer's characteristics in the configuration file.

Each mailer must have an internal name.

Table 4–5 shows the internal names, agents, and pathnames of some sample mailers. For example, **smtp**, a server that listens for network connections, is the agent for the mailer with the internal name `tcp`. The **local** mailer uses the local mail program as its agent (most often **/bin/mail**).

Table 4–5 sendmail Mailers

Internal Name	Agent	Pathname
local	localmail	/bin/csh
prog	shell program	/bin/sh
tcp	**smtp** over TCP/IP	[IPC]
uucp	**uucp** transport	/usr/bin/uux

You must specify the internal names **local** and **prog** in the **sendmail** configuration file. You can specify no more than 25 mailers.

The syntax for defining a mailer is as follows:

M*name*,[*field=value*]...

where *name* is the internal name of the mailer, and *field=value* pairs define the characteristics of the mailer. Permitted fields are as follows:

Path (**P**) The pathname of the mailer. If the mailer is accessed via an Inter-Process Communication (IPC) connection, use the string **[IPC]** instead.

Flags (**F**) Special flags that indicate headers that the mailer gets or information about the mailer. Mailer flags and their functions are listed below. For examples of how mailer flags are used in headers, see "Defining Headers" earlier in this chapter.

Sender (**S**) A rewriting ruleset that is applied to the sender addresses after the sending domain is appended and the general rewriting rulesets 3 and 1 are applied. See "Defining Rewriting Rules" later in the chapter.

Recipient (**R**) A rewriting set for recipient addresses that is applied after the sending domain is appended and the general rewriting rulesets 3 and 2 are applied. See "Defining Rewriting Rules" later in the chapter.

Argv (**A**) An argument vector to pass to the mailer. The argument vector may include **sendmail** macros.

Eol (**E**) The end-of-line string for the mailer. The default is a string containing only a new line. You can use the backslash escapes (\r, \n, \f, \b).

Maxsize (**M**) The maximum message length, in bytes, which may be sent to this mailer.

Using Mailer Flags

There are two classes of mailer flags: flags that indicate that a message sent with the mailer should get a header, and flags that tell **sendmail** something about the mailer itself. These two classes of flags can overlap.

Unless you change the associated headers in the **sendmail** configuration file, the following flags indicate that a message sent with the mailer should get the specified header.

Flag	Header Specified
D	`Date:` header line.
F	`From:` header line.
M	`Message-Id:` header line.
P	`Return-Path:` line.
x	`Full-Name:` header line.

Here are the flags that tell **sendmail** something about the mailer itself.

Flag	Description
A	This is an Arpanet-compatible mailer, and all appropriate modes should be set.

C	If receiving mail from a mailer that has the **C** flag set, append the "@domain" clause from the sender to any addresses in the header that do not include an at sign (@) after the header is rewritten by ruleset 3 (see "Defining Rewriting Rules" later in this chapter). Here is an example:
	`From: usera@hosta`
	`To: userb@hostb, userc`
	is rewritten automatically as:
	`From: usera@hosta`
	`To: userb@hostb, userc@hosta`

E	Insert the > character at the beginning of any line in the message that begins with `From`.
I	This mailer "speaks" SMTP to another **sendmail**. Therefore, it can use special protocol features (see *RFC 821*). If you omit the **I** flag, the transmission still operates successfully, although perhaps not as efficiently as possible.

L	Limit the line lengths as specified in *RFC 821*.
S	Do not reset the user ID before calling the mailer. You can use this flag in a secure environment where **sendmail** runs as root. This flag is suppressed if given from an environment that is not secure.
U	This mailer gets DG/UX-style `"From "` lines with the UUCP-style `remote from` *host* on the end.
V	Change header lines so that the UUCP path is relative to the recipient host. Routes through the *remote* host, that is, addresses that begin with *remote*!... are stripped of that part unless the ultimate recipient resides on *remote*. The **sendmail** program prefixes all other addresses with *localhost*!... if *localhost* is not already there. The **sendmail** program fetches *localhost* from the **k** macro. The default value is your local hostname, as supplied by **gethostname**(3).
X	This mailer uses the hidden dot algorithm as specified in *RFC 821*; basically, any line beginning with a dot has an extra dot prepended (to be stripped at the other end). This ensures that lines containing only a dot do not terminate the message prematurely.
e	This mailer is expensive to connect to, so try to avoid connecting routinely; any necessary connection occurs during a queue run.
f	The mailer expects the **-f***name* command line option, where *name* is the host name of the sender. The mailer returns an error if the executing user does not have special permissions. For information about **-f***name*, see "Command Line Options" later in the chapter.
h	This mailer preserves uppercase form of hostnames. If not set, hostnames are mapped to all lower case before attempting to deliver mail.
l	This mailer is local; that is, final delivery of a message is performed on the system running this **sendmail** process.
m	This mailer can send to multiple users on the same host in one transaction. When a **u** macro occurs in the argument vector (argv) of the mailer definition, the expansion of the macro is repeated as necessary for all qualifying users.

n	Do not insert a DG/UX-style `"From "` line on the front of the message.
p	Use the return-path in the SMTP `"MAIL FROM:"` command rather than just the return address; although this is required in *RFC 821*, many hosts do not process return paths properly. *RFC 1123* does not recommend using the return–path.
s	Strip quote characters off the address before calling the mailer.
u	This mailer preserves uppercase form of usernames. If not set, usernames are mapped to all lower case before attempting to deliver mail.

Examples of sendmail Mailer Definitions

Here is an example of a **sendmail** mailer definition:

```
Mlocal, P=/bin/mail, F=DFMlmns, R=25/10, S=11/10,
A=mail $u
```

The mailer is called `local`, and it is found in `/bin/mail` (as specified in the **P** field).

The **F** field of the mailer specifies the mailer flags `D`, `F`, `M`, `l`, `m`, `n`, and `s`. Thus, the mailer puts the following header lines into messages: `Date:`, `From:`, and `Message-Id:`, the mailer is going to perform local delivery, the mailer can send to multiple users on the same host in one transaction, the mailer does not require a DG/UX-style `From` line on the front of the message, and the mailer strips quote characters from the address before **sendmail** calls it.

The local mailer uses split envelope/header rulesets for the **R** field. The example indicates that ruleset 25 is applied to recipient envelopes and ruleset 10 is applied to recipient headers (see "Defining Rewriting Rules" later in this chapter).

The local mailer uses the split envelope/header ruleset feature for the **S** field as well. The example indicates that ruleset 11 is being applied to sender envelopes and ruleset 10 is applied to sender headers (see "Defining Rewriting Rules" later in this chapter).

The **A** field indicates that the argument vector (argv) to pass to the mailer is the word "mail" and the macro expansion of **u** (the username).

Defining Rewriting Rules

The **sendmail** program applies rewriting rulesets to message addresses to determine which mailer it should use, and to change

the appearance of message headers, for example, to add the local hostname to a sender's address. Because **sendmail** uses rewriting rulesets to parse addresses, **sendmail** can accept addresses that use arbitrary syntax.

Rewriting rulesets consist of one or more rewriting rules. Use the following syntax to specify the beginning of a rewriting ruleset:

S*n*

where **S** indicates the start of the ruleset, and *n* is an integer between 0 and 50 that uniquely identifies the ruleset. Thus, **S10** indicates the start of rewriting ruleset 10. After the **S** line, you specify rewriting rules with the following format:

R*lhs*<**TAB**>*rhs*<**TAB**>*comments*

where **R** specifies a rewriting rule line, *lhs* defines the left-hand side of the rule, *rhs* defines the right-hand side of the rule, and *comments* can be used to remind you what the rewriting rule does. You must separate the fields of a rewriting rule line by at least one tab character; you may put embedded spaces in each field.

The appearance of another **S** designates the beginning of the next ruleset, and therefore ends the previous ruleset. If two rulesets are defined with the same number, the last definition is used.

There are two general types of rewriting, one for each type of address: envelope and message header. Envelope and message header addresses in **sendmail** can be compared to the addresses on a paper envelope and on the enclosed letter. The address on the envelope is used by some agent to forward the mail to its proper destination, and the address on the enclosed message is read and used by the recipient or recipients of the mail.

For example, suppose `vance` sends mail to `mail-list` from `sys01`. On `sys01`, **sendmail** examines the recipient address and determines that it is an alias for `goolsby`, `dodd`, `pafford`, and `jones`. The **sendmail** program expands `mail-list` to `goolsby` and so on in the envelope address, but leaves the `To:` header address in the message as `To: mail-list`.

Each type of address has a specific type of rewriting:

- Envelope rewriting parses the addresses of the sender of the message and of all intended recipients.
- Message header rewriting parses addresses associated with headers, including all headers associated with the recipient and all headers associated with the sender

Headers associated with the recipient include `To:`, `Resent-To:`, `Cc:`, `Resent-Cc:`, `Bcc:` and `Resent-Bcc:`.

Headers associated with the sender include `From:`, `Resent-Sender:`, `Resent-From:`, `Sender:`, `Reply-To:`, `Full-Name:`, `Return-Receipt-To:`, and `Errors-To:`.

There are seven sets of fixed rewriting rules, numbered 0 through 6:

Ruleset 0 Processes an address to determine which mailer to use, the sender or recipient of the message, and, if the message must go to a remote system, the name of the system.

Ruleset 1 Applied to envelope and header addresses associated with the sender.

Ruleset 2 Applied to the envelope and header addresses associated with the recipient.

Ruleset 3 Maps addresses of any form into a canonical form. A canonical form is a fully qualified form. When **sendmail** parses an address, it starts with this rule.

Ruleset 4 Maps addresses from canonical form to external form.

Ruleset 5 If you specify the / option in the configuration file, **sendmail** applies ruleset 5 to the header address associated with the sender. If you do not specify the / option, sendmail uses ruleset 1 to process the header address associated with the sender. The / option is not in effect by default.

Ruleset 6 If you specify the / option in the configuration file, **sendmail** applies ruleset 6 to the header address associated with the recipient. If you do not specify the / option, sendmail uses ruleset 2 to process the header address associated with the recipient.

Figure 4–2 shows how an address is parsed when **sendmail** processes a recipient's address, a sender's address, a recipient header, and a sender header.

The top part of Figure 4–2 illustrates envelope address rewriting. Initially, the envelope sender address is parsed with rewriting rulesets 3, 0, 2, R, and 4 for some internal bookkeeping, then the same address is parsed with rewriting rulesets 3, 1, and 4 to produce **$f**, the sender (from) address relative to the system where this processing occurs. This address is then parsed with rewriting rulesets 3, 1, S (precise value depends on the mailer used), and 4 to produce **$g**, the sender address relative to the recipient. The envelope recipient address(es) are parsed with rewriting rulesets 3, 0, and 4 to produce an address that is then parsed with rewriting

rulesets 3, 2, R (precise value depends on the mailer used), and 4 to produce the parsed address relative to the sender.

The lower part of Figure 4–2 illustrates header address rewriting. Sender header rewriting uses rulesets 3, 1, S, and 4. Recipient header rewriting uses rulesets 3, 2, R, and 4.

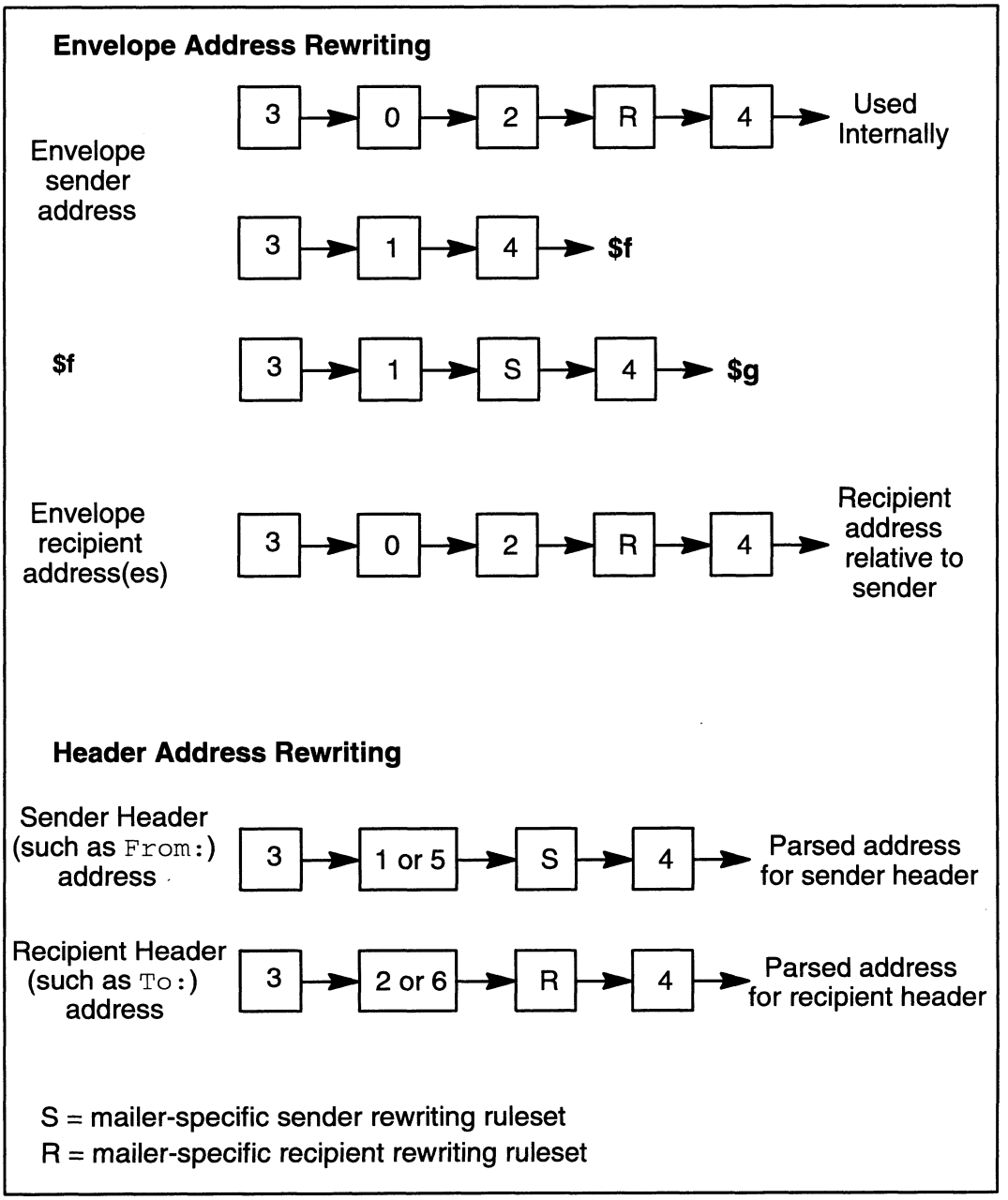


Figure 4–2 Relationship of Rulesets for Address Parsing

As you now know, a ruleset consists of rules, and a rule consists of a left-hand side and a right-hand side. The **sendmail** program tries to match an address with the pattern on the left-hand side of a rule in the ruleset. When an address matches the pattern on the

left-hand side of a rule, that address is replaced by the expansion of the right-hand side of that rule.

As Figure 4–2 shows, when **sendmail** parses an address, it always starts with ruleset 3. If no match occurs between the address and the pattern on the left-hand side of any rule in ruleset 3, **sendmail** parses with the next ruleset in the appropriate chain.

The next section generally describes how **sendmail** parses an address with a rewriting rule.

The Logic of a Rewriting Rule

The left-hand side of a rewriting rule contains a pattern that is compared to the address. The pattern consists of symbols that **sendmail** matches to tokens in the address. The symbols you use to define this pattern are described in the next section. A token is any literal value or defined separator in the address.

For example, consider the following address:

```
jack@spectre.aa.com
```

Assuming that the **o** macro defines `@` and **.** as separators, this address consists of seven tokens:

```
jack   @   spectre   .   aa   .   com
```

Each symbol in the rewriting rule's left-hand side is applied to the tokens of the address as appropriate. If an address matches the symbol pattern on the left-hand side of a rewriting rule, that address is replaced with the expansion of the symbol pattern on the right-hand side. If the left-hand side does not match, then:

- the next rule is applied, or
- *if* this rule is the last one in the ruleset, the ruleset returns the address.

A rule is applied repeatedly until it fails. The **sendmail** command continues to apply a rule unless one of the following conditions is true:

- The rule no longer succeeds (that is, the left-hand side does not match).
- The right-hand side begins with the following symbols: **$:**, **$@**, or **$#**.

The next two sections describe the syntax of the left-hand and right-hand sides of a rewriting rule.

The Left-hand Side of a Rewriting Rule

The symbols you can use on the left-hand side of a rewriting rule are as follows:

Symbols	Description
$*	Match zero or more tokens.
$+	Match one or more tokens.
$-	Match exactly one token.
$=*x*	Match any token in class *x*.
$~*x*	Match any token not in class *x*.
literal	Any *literal* value, for example `vance`, or `@`.

If any of these symbols (except *literal*) matches the input, they are assigned to the symbol $*n*, where *n* is the index number in the left-hand side (see the description in the next section), for replacement on the right-hand side. For example, suppose that the left-hand side:

```
$-@$+
```

is applied to the input:

```
karl@spectre.aa.com
```

As you saw in the preceding section, the tokens of this address are as follows:

```
karl    @    spectre    .    aa    .    com
```

The first token, `karl`, matches the symbol `$-`, which means to match exactly one token. The `@` token in the address matches the `@` literal token on the left-hand side. The remaining tokens of the input address match the `$+` symbol, which means match one or more tokens.

Suppose the same left-hand side is applied to the following input:

```
oldhost!amh
```

Assuming that you have used the **o** macro to define **!** as separators, this address consists of three tokens.

```
oldhost    !    amh
```

As before, the first token, `oldhost`, matches the symbol `$-`. However, the rule fails because the second token of the address, `!`,

does not match the @ of the left-hand side. Because the rule fails, the right-hand side is not substituted.

The Right-hand Side of a Rewriting Rule

When the left-hand side of a rewriting rule matches the input, the input is replaced by the right-hand side. The right-hand side consists of literal tokens and special symbols that begin with **$**. The symbols that you use on the right-hand side are:

$*n* Substitute the corresponding value from a $*, $+, $-, $=x, or $~x matched on the left-hand side.

$[***name***$:*default***$]**
Replace a hostname with its fully qualified name. The brackets are mandatory. For example, if **bucky** and **harpo** (**128.223.8.48**) are part of the DNS domain named **tnt.acme.com**, **$[bucky$]** becomes **bucky.tnt.acme.com** and **$[128.223.8.48$]** becomes **harpo.tnt.acme.com**. If the lookup fails, replace *name* with *default*. The **$:***default* value is optional.

For details about fully qualified names and the DNS, see Chapter 5.

$(***x key***$@*arg***$:***default***$)**
Lookup *key* in **dbm**(3) database *x*. If you specify **$@***arg* and if **sendmail** finds *key*, **sendmail** uses the *key* as a format string of an **sprintf**(3) expression, with *arg* as the argument. If the lookup fails, for example, because there is no record for *key*, the *default* is returned. Both the **$@***arg* and the **$:***default* are optional.

$&*x* Force runtime evaluation of macro *x*.

$>*n* "Call" ruleset *n*. This symbol causes the remainder of the line to be substituted as usual and then passes it as the input to ruleset *n*. The returned value of ruleset *n* then becomes the substitution for this rule.

$: When this symbol is at the beginning of the right-hand side, it means to substitute this right-hand side and exit this rule, then continue with the ruleset.

$@ When this symbol is at the beginning of the right-hand side, it means to substitute this right-hand side and exit this ruleset.

$#*mailer* Resolve to the *mailer* that processes the mail. Use this syntax only in ruleset 0 (see "Step 3: Modifying Ruleset 0 to Call Mailers" in this chapter). It causes the ruleset to be exited and means that the invocation address has been completely resolved. The *mailer* must be a single word.

The complete syntax is:

$#***mailer***$@*host***$:***user*

which specifies the necessary arguments for the mailer. If the mailer is local, the **$@***host* portion can be omitted.

$@*host* Specify *host*. The use of **$@** to signal that a hostname follows applies only when a right-hand side starts with **$#**. The *host* must be a single word, for example **$2.$1** is invalid, but **$2** is valid.

$:*user* Specify *user*. The use of **$:** to signal that a username follows applies only when a right-hand side starts with **$#**. The *user* can be multi-part.

The **$@** and **$:** prefixes can also precede a **$>** symbol. You can use multiple instances of the **$>** symbol. For example:

```
R$*     $:$>7$>8$1
```

Here, the left-hand side (`R$*`) matches anything and the value is available as **$1**. The value in **$1** is passed to ruleset 8. The return value of ruleset 8 is given to ruleset 7, and processing continues with the next rule of the current ruleset. The **$:** is included to avoid an infinite loop.

Consider the address introduced earlier:

```
jack@spectre.aa.com
```

You have seen how the following left hand side of a rewriting rule

```
$-@$+
```

is applied to the tokens of this address.

```
$-    matches    jack
@     matches    @
$+    matches    spectre.aa.com
```

The first token is available on the right-hand side as **$1**, because `$-` was the first symbol of the left-hand side. The literal `@` matches `@`. The remaining tokens are available on the right-hand side as **$2** because `$+` is the second symbol on the left-hand side.

Because the address matches the left-hand side, the rule passes and the right-hand side is substituted. Thus, the following values are available to the right-hand side:

```
jack                    as $1
spectre.aa.com          as $2
```

Defining Rewriting Rulesets for Your Mailers

When you defined your mailers, you defined **S** and **R** fields. You now need to write the rewriting rules that tell **sendmail** what to put on the "To:" and "From:" lines. For example, consider the following mailers:

```
Mtcp,  P=[IPC], F=msDFMueXL, S=15, R=14, A=IPC $h,
E=\\r\\n

Muucp, P=/usr/bin/uux, F=sDFMhuU, S=16, R=14, M=100000,
       A=uux - -r -$h!rmail ($u)
```

As described earlier, the **S** field defines a rewriting set for the sender addresses (rulesets 15 (S=15) and 16 (S=16)) and **R** defines a rewriting set for recipient addresses (ruleset 14 (R=14)). What you define for these rulesets depends on how you want **sendmail** to change the addresses.

Suppose that the two mailers shown above used the following rewriting rules:

```
S14

S15

R$*@$*              $@$1@$2
R$*                 $:$1@$w

S16

R$w!$*              $@$w!$1
R$*                 $:$w!$1
```

Here, both mailers call ruleset 14 for recipient addresses. Because ruleset 14 does not contain any rule definitions (**R** lines), **sendmail** does not change the `To:` header line for recipient addresses when the mailer invokes the ruleset.

Example: tcp Mailer Calling Ruleset 15

The example tcp mailer shown above calls ruleset 15 for sender addresses. The **sendmail** program scans the left-hand side of the first line of ruleset 15 for a match. The left-hand side of the first line in ruleset 15 specifies that an address contain an @ sign. For example:

```
jones@sys13
```

would match the left-hand side `$*@$*`. Because the address matched the left-hand side of the rule, **sendmail** applies the rules on the right-hand side to this address. The right-hand side does the following:

- `$@` tells **sendmail** to substitute the right-hand side and exit from the ruleset.
- `$1` is the first token matched and is the right-hand side equivalent of `jones`.
- `@` is a literal token in the right-hand side, which is placed in the parsed address.
- `$2` is the second token matched and is the right-hand side equivalent of `sys13`.

Therefore, the right-hand side `$@$1@$2` leaves the incoming address `jones@sys13` unchanged.

The second line of ruleset 15 tells **sendmail** to scan for all other tokens (the `$*` symbol). If an address came in as just `jones`, the address would first fail to match the first line of the rule. The address would then match the left-hand side of the second line, and the right-hand side would do the following:

- `$:` causes the rule to terminate immediately, but lets the ruleset complete. (Otherwise, `$*` would always match, and **sendmail** would go into a loop.)
- `$1` is the first token matched and is the right-hand side equivalent of `jones`.
- `@` is a literal token in the right-hand side, which is placed in the parsed address.
- `$w` is a macro that specifies the sender's hostname. For this example, the hostname is `sys13`.

Therefore, the right-hand side `$:$1@$w` changes the incoming address `jones` to `jones@sys13`.

Example: uucp Mailer Calling Ruleset 16

For the example uucp mailer, **sendmail** scans ruleset 16 for a match. In the left-hand side of the first line, **sendmail** looks for a specific address with a ! sign. For example:

```
sys13!jones
```

would match the left-hand side `$*w!$*`. Because there was a match, **sendmail** would apply the rules on the right-hand side to this address. The right-hand side does the following:

- `$@` tells **sendmail** to substitute the right-hand side and exit the ruleset.
- `$w` is the macro for the sender's hostname (`sys13`).
- `!` is a literal token
- `$1` is the first token matched and is the right-hand side equivalent of `jones`.

Therefore, the right-hand side `$@$w!$1` leaves the incoming address `sys13!jones` unchanged.

The left-hand side of the second line of ruleset 16 tells **sendmail** to scan for all other tokens (the `$*` symbol). If an address came in as just `jones`, the address would first fail to match the first line in the rule. The address would then match the left-hand side of the second line of the rule, and the right-hand side would do the following:

- `$:` causes the rule to terminate immediately, but lets the ruleset complete. (Otherwise, `$*` would always match, and **sendmail** would go into a loop.)
- `$w` is a macro that specifies the sender's hostname.
- `$1` is the first token matched and is the right-hand side equivalent of `jones`.

Therefore, the right-hand side `$:$w!$1` changes the incoming address `jones` to `sys13!jones`. A longer address such as `sys5!jones` would be changed to `sys13!sys5!jones`.

Step 3: Modifying Ruleset 0 to Call Mailers

In the sample configuration file, ruleset 0 is divided into two parts: the ruleset 0 preamble and the system-dependent part. You do not need to change the preamble, but you must modify the system-dependent part so that **sendmail** recognizes the incoming address and calls the appropriate mailer.

This example uses the local, tcp, and uucp mailers described earlier.

Depending on your mailers, writing rules for ruleset 0 can be a simple process. For the local, tcp, and uucp mailers, the following left- and right-hand sides, which include appropriate comments in the third column, should successfully sort through the addresses and call the proper mailers:

```
R$*@$w      $#local$:$1       tcp style for this system
R$*!$*      $#uucp$@$1$:$2    uucp going elsewhere
R$*@$*      $#tcp$@$2$:$1     tcp going elsewhere

R$+         $#local$:$1       default
```

For example, if the address `sys10!jones` came in, **sendmail** would match the second line of ruleset 0 and would call the uucp mailer. The fourth line in ruleset 0 is a default line that covers any addresses not identified in the first three lines. You should have a similar line in your ruleset 0 configuration.

The only time you might have difficulty with your rules is when you receive a complex address. For example, you could receive the following address:

```
sys10!summit!joe@nic.arpa
```

Using ruleset 0 as specified above, **sendmail** would misinterpret this to mean that it should use **uucp** to send a message to `sys10` for `summit!joe@nic.arpa`. You can fix the problem by switching the order of lines 2 and 3 of the rewriting ruleset. Once the lines are switched, **sendmail** properly uses tcp on host `nic.arpa` to send a message to user `sys10!summit!joe`.

Step 4: Testing Your Configuration

This section shows you how to test your configuration by taking the following steps:

- Attempt to send or receive mail with your configuration and see if you get the results you expect.
- Use **sendmail**'s test mode to check your rewriting rules (we describe this step in the following section).

Testing the Rewriting Rules

To test the rewriting rules, invoke **sendmail** with the following option:

```
% sendmail -bt ↵
```

When invoked this way, **sendmail** reads the configuration file **/etc/sendmail.cf** (or the frozen configuration file, if it exists and is current) and enters test mode. (For details about the frozen configuration file, see "Step 5: Setting Up a Frozen Configuration File" later in the chapter.) If you want to test some other configuration file, invoke **sendmail** the following way:

```
% sendmail -bt -Cconfigurationfile -oQqueuefile ↵
```

When you test an alternative configuration file and you are not running as root, you must specify a different queue file. This is because **sendmail** does not set its user ID to root when you use a

different configuration file, and because only root has write permissions for **/var/spool/mqueue**, **sendmail** cannot write its queue files there.

When you are in test mode, you get the following prompt:

```
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
>
```

At this prompt, use the following format to enter rulesets to be tested:

rwset[,*rwset*, ...] *address*

where *rwset* is the number of the rewriting ruleset or rulesets that want to test, and *address* is the address to which you want to apply the set. The test shows you the final address and the steps that **sendmail** took to get the final address.

For example, suppose that you want to test the case of user `smith` sending a message to `jones@sys56`. Suppose that the mailer definition line in the **sendmail.cf** file specifies S=15 and R=14.

Remember that there are two general types of rewriting: envelope and message header. Earlier in this chapter, you saw that envelope rewriting uses rulesets 3, 0, and 4, and that message header rewriting depends on whether the header is associated with the sender or the recipient. `To:` header rewriting uses rulesets 3, 1, a mailer-specific sender ruleset (defined here to be ruleset 15), and ruleset 4. `From:` header rewriting uses rulesets 3, 2, a mailer-specific recipient ruleset (defined here to be ruleset 14), and ruleset 4.

With this in mind, you first enter the following command:

```
% sendmail -bt↵
```

The system responds:

```
ADDRESS TEST MODE
Enter <ruleset> <address>
[Note: No initial ruleset 3 call]
>
```

You then enter:

```
> 3,0,4 jones@sys56↵
```

The output of this test is as follows:

```
rewrite: ruleset  3   input: jones @ sys56
rewrite: ruleset  7   input: jones @ sys56
rewrite: ruleset  9   input: jones @ sys56
rewrite: ruleset  9 returns: < jones > @ sys56
rewrite: ruleset  7 returns: < @ sys56 > , jones
rewrite: ruleset  8   input: < @ sys56 > , jones
rewrite: ruleset  8 returns: < @ sys56 . tnt . acme . com > , jones
rewrite: ruleset  3 returns: < @ sys56 . tnt . acme . com > , jones
rewrite: ruleset  0   input: < @ sys56 . tnt . acme . com > , jones
rewrite: ruleset 29   input: < @ sys56 . tnt . acme . com > , jones
rewrite: ruleset 29 returns: < @ sys56 . tnt . acme . com > , jones
rewrite: ruleset 26   input: < @ sys56 . tnt . acme . com > , jones
rewrite: ruleset  2   input: < @ sys56 . tnt . acme . com > , jones
rewrite: ruleset  2 returns: < @ sys56 . tnt . acme . com > , jones
rewrite: ruleset  4   input: < @ sys56 . tnt . acme . com > , jones
rewrite: ruleset  4 returns: jones @ sys56 . tnt . acme . com
rewrite: ruleset 26 returns: $# LOCAL $@ sys56 . tnt . acme . com $: jones
rewrite: ruleset  0 returns: $# LOCAL $@ sys56 . tnt . acme . com $: jones
rewrite: ruleset  4   input: $# LOCAL $@ sys56 . tnt . acme . com $: jones
rewrite: ruleset  4 returns: $# LOCAL $@ sys56 . tnt . acme . com $: jones
```

First, **sendmail** applies ruleset 3 to the input `jones@sys56`. Next, ruleset 3 calls ruleset 7, which in turn calls ruleset 9. Ruleset 9 returns `<jones>@sys56`, and ruleset 7 transforms this to `<@sys56>,jones`. Then ruleset 3 calls ruleset 8, which turns its input into `<@sys56.tnt.acme.com>,jones`. Ruleset 3 returns this.

Next, **sendmail** applies ruleset 0 to `<@sys56.tnt.acme.com>,jones`. Ruleset 0 calls ruleset 29 and then ruleset 26, which in turn calls ruleset 2 and then ruleset 4. Ruleset 2 does not transform its input. Ruleset 4 transforms its input to `jones@sys56.tnt.acme.com`. Ruleset 26 takes this input and returns `$#LOCAL$@sys56.tnt.acme.com$:jones`. Ruleset 0 returns this. Then, **sendmail** applies ruleset 4 to this output.

The input line to test rewriting of the sender header would be:

> 3,1,15,4 smith ↵

and the input line for the recipient header would be:

> 3,2,14,4 jones@sys56 ↵

If you need more information during testing, invoke **sendmail** as follows:

% sendmail -bt -d21.12 ↵

This line causes **sendmail** to print out information about each individual rule that is tried.

Step 5: Setting Up a Frozen Configuration File

Each time **sendmail** starts execution, it reads its configuration file. On systems that handle lots of mail traffic, multiple reads can degrade system performance. To alleviate this degradation, a superuser may set up a quick version of the **sendmail** configuration file with the following command:

```
# sendmail -bz ↵
```

This creates a "frozen" configuration file named **/etc/sendmail.fc**. This frozen file is an image of the data portion of **sendmail**'s executable image after reading in the configuration file. If the frozen configuration file exists, it is used instead of **/etc/sendmail.cf**.

You must create a new frozen configuration file in the following circumstances:

The hostname changes

You are using the NIS and your NIS domain name changes

The **sendmail.cf** changes

You install a new **sendmail** executable file

The frozen configuration file is ignored if you specify a **-C** flag on the **sendmail** command line.

Changing Files After You Configure sendmail

The following files contain special lines to accommodate **sendmail**. If you have problems loading files for TCP/IP for AViiON Systems, check whether these lines appear as shown.

/etc/services — The following line tells the **smtp** server to listen for incoming calls on TCP port number 25

smtp 25/tcp mail

/etc/passwd — The following line creates a password entry for the programs that deliver mail.

mail:*:8:1:
Sendmail:/usr/mail:/usr/bin/mail

User IDs below 100 are reserved for servers or daemons. Because **mail** is a server, the number 8 is arbitrarily selected. The number that you select must match the value specified in **/etc/sendmail.cf**. See the description of option **u** in the section "Defining Options" earlier in this chapter.

/etc/tcpip.params The following line gives arguments to the **smtp** daemon that tell it to attempt to deliver queued messages every 30 minutes.

DAEMON_NAME=smtp
DAEMON_ARGS="-q30m"

Using sendmail

To send and receive electronic mail, we recommend that you use **mailx** (or another mail program with an easy-to-use interface) rather than **sendmail**. However, if you want to use **sendmail** to send e-mail, this section shows you how.

To send a local message with **sendmail**, enter the following lines, filling in the appropriate values:

- **sendmail** *recipient*
- **To:** *recipient*
- Optional. **From:** *yourname*
- Press New Line.
- *Your message. End every line of text by pressing New Line.*
- Use **<Crtl-D>** or enter a period (.) as the only character of the last line of the message to end and send your message.
- If make a mistake, you get an error message. Start over.
- If you want to send a file, redirect it on the command line. Use the following format,

sendmail *recipient* **<** *file*

To send a message to a user on a remote computer system, specify the recipient's address with the @ sign (for example, **owen@bucky).**

Command Line Options

The following list details the command line options that you can use when you edit the `DAEMON_ARGS` on the `DAEMON_NAME=smtp` line in

the **/etc/tcpip.params** file, or when you invoke **sendmail** otherwise.

-b*x* Set operation mode to *x*. Operation modes are as follows:

- **a** Run in ARPANET mode. Special processing for ARPANET mode includes reading the `From:` line from the header to find the sender, printing ARPANET style messages (preceded by three-digit reply codes for compatibility with the FTP protocol), and ending lines of error messages with **<CRLF>**(Ctrl-M Ctrl-J).
- **d** Run as a server.
- **i** Initialize the alias database. You can also initialize the alias database by running **/usr/bin/newaliases**.
- **m** Deliver mail in the usual way (default).
- **p** Print the list of messages in the mail queue. You can also use the **mailq** command to print the contents of the queue.
- **s** Speak SMTP with standard input and standard output, as described in *RFC 821*.
- **t** Run in address test mode. This mode tells **sendmail** to read addresses and shows the steps in parsing the addresses.
- **v** Just verify addresses, do not collect or deliver messages.
- **z** Create a frozen configuration file.

-C*file* Use the configuration file named *file*. When given this option, **sendmail** runs as the invoking user, not as root.

By default, **sendmail** looks for a frozen configuration file or **/etc/sendmail.cf**.

-d[[*x***[-m][.***l***]] [,..]]**

Set the debugging value *x* to *l*, where *x* represents a debugging stream or subsystem, *x-m* represents a range of debugging subsystems, and *l* represents a verbosity level (higher values result in more debugging output; the default *l* is 1). Separate repeated arguments with a comma. Here are some examples.

sendmail -d21.12 *user*
or
sendmail -bs -d5-10.10,4.1 *user@host*

With no options, **-d** sets all debugging flags to level 1.

-F Use the full name of the sender on the `From:` line.

-f*name* Sets the name of the "`From`" person. The **-f** can be used only by the trusted users as specified in the configuration file (for example, root, daemon, or network), or if you specify your own name.

-h*N* Set the current value of the hop count to *N*. The current value of the hop count is incremented each time the mail is processed. Thus, you can use this option to act as though the mail has been processed more times than it actually has. When the hop count reaches its maximum value, **sendmail** returns the mail with an error message.

-n Do not do aliasing or forwarding.

-o*xvalue* Set option *x* to the specified value. These options are described earlier in the "Defining Options" section.

-q[*time***]** Specify how often SMTP runs through the mail queue.You must be the superuser to use this option. If you are running **sendmail** in interactive mode or background mode, the *time* you specify can be relatively long. If you run in **q** mode, the *time* you specify should be shorter than the timeout interval (set with option **T**). You want a fair number of delivery attempts before **sendmail** returns the message to the sender.

You must specify the *time* interval. For example, "30m" represents thirty minutes, whereas "2h30m" represents two hours and thirty minutes. The time intervals are:

s seconds

m minutes

h hours

d days

w weeks

You can use this option to start the **smtp** daemon.

```
# sendmail -bd -q30m ↵
 or
# smtp -q30m ↵
```

Use the -q option without specifying a *time* to force the mail queue. For more information, see the upcoming section "Forcing the Mail Queue."

-t Use the `To:` and `Cc:` lines of the message to determine where the mail should go.

-v Run in verbose mode. Alias expansions are announced.

-Z*file* Use a different frozen configuration file. When given this option, **sendmail** runs as the invoking user, not as root.

Managing the Mail Queue

The **sendmail** program keeps a queue of messages. When **sendmail** processes messages in the queue, it is called a queue run. A queue run may involve one or more messages, because it may be more efficient to process several messages at once rather than each message individually. If a message cannot be delivered immediately — perhaps the recipient host is down — the queue provides temporary storage capacity.

Each message in the queue has a data file, a control file, and a transcript file. By default, these temporary files reside in **/usr/spool/mqueue**. These files begin with the letters **d**, **q**, and **x**.

- The **d***files* are data, which contain the text of the message.
- The **q***files* contain control information about the message. These files tell the precedence of the message, time the message was received in the queue, name of the data file, sender, recipient, error messages, and header information. To determine the text of the numerical error number given, look in the **errno.h** file.
- The **x***file* might contain some user and some system error messages. It is the transcript file that **sendmail** produces as it processes a job. The file contains some of the same information as that produced on your screen when you invoke **sendmail** with the verbose option.
- The star (*) displayed next to the queue ID while you examine the queue (**sendmail -bp**, or **/usr/bin/mailq**)means that **sendmail** is currently processing the control file.

The **sendmail** program should process the mail queue automatically. However, sometimes you may need to intervene. For example, if a major host is down for a period of time, the queue may become clogged. In this case, **sendmail** may perform inefficiently. First, print the contents of the mail queue, and then force the queue when the host comes back up.

Printing the Mail Queue

You can print the contents of the mail queue using the **mailq** command or by specifying the **-bp** option to **sendmail**.

```
% mailq ↵
or
% sendmail -bp ↵
```

The **mailq** command produces a listing of the queue IDs, the size of the message text in bytes, the date that the message entered the queue, the sender of the message, any error message generated, and the message recipients. Here is an example of the output of **mailq**:

```
                Mail Queue (1 request)
--QID--   --Size-----Q-Time---------------Sender/Recipient---
AA05305*   17        Tue Jun 4 10:47      pafford
                                          goolsby@sales
```

Forcing the Mail Queue

If you invoke the **sendmail** with the **q** flag, the program automatically runs the queue at intervals. The algorithm that **sendmail** uses reads and sorts the queue, and then attempts to process all messages in order.

The **sendmail** program does not ensure that only one queue process exists at any time, since there is no guarantee that a message cannot take a very long time to process. Due to the locking algorithm, it is highly unlikely that one message will freeze the queue. However, an uncooperative recipient host or a program recipient that never returns can cause **sendmail** to accumulate many processes. You can partially alleviate this problem by setting a timeout using the **r** option.

In some cases, you may find that a major host going down for a couple of days may create a prohibitively large queue. This may result in **sendmail** spending an inordinate amount of time and memory sorting the queue. If **sendmail** slows down for this reason, you can move the queue and rename it. Then, when the major host is up and running again, you can move the queue to its original location.

To force the message queue, use **sendmail** with the following option:

```
# sendmail -q -v ↵
```

Using the **-v** option causes **sendmail** to tell you more about what it is doing as it processes the queue.

Using the Aliases Database

The **sendmail** program uses the alias database to redirect mail to a different address or set of addresses. You can use it to provide alternate names for users, for example:

jack: vance

You can also use it to provide groups, for example:

mail-list: goolsby, dodd, jonesjg, owen, pafford

Each alias consists of a name and a list of substitutions. Any mail destined for the name to the left of the colon goes instead to each member of the list of substitutions.

The alias database exists in two forms. The first form, or the *text* form, is maintained in the file **/etc/aliases**. These aliases are of the form:

```
name: name1, name2, ...
```

Only local names may be aliased. For example, the following line does send mail for riversk at sys1 to riversk at sys2:

```
riversk@sys1: riversk@sys2
```

Continue aliases over a line by starting any continuation lines with a space or a tab. Blank lines and lines beginning with a pound sign (#) are interpreted as comments.

The second form, or the *dbm* form, is processed by the **dbm**(3) library functions, and is created from the first form. This form is specified in the files **/etc/aliases.dir** and **/etc/aliases.pag**. The **sendmail** program uses this form to resolve aliases.

When **sendmail** attempts to reconcile the name of an addressee for any possible alias, it first interrogates its local alias database. If it does not find the alias there, **sendmail** queries the Network Information Service's (NIS) **mail.aliases** map. If the name alias does not exist in **mail.aliases**, **sendmail** attempts to deliver the mail to the named addressee. If you do not have the NIS installed on the host, or if the NIS becomes unavailable, **sendmail** checks its local alias database.

In a large facility, the Network Information Services may offer more efficient administration of aliases than **/etc/aliases**. NIS provides a single distributed database rather than a database that must be sent to every host that wants to use it. Thus, NIS lets you keep aliases up-to-date with less effort. For information about the NIS and **mail.aliases**, see *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System.*

Rebuilding the Aliases Database

After you edit the **/etc/aliases** file, you must use the following command to rebuild the alias database to incorporate the updates:

```
% newaliases ↵
or
% sendmail -bi ↵
```

Be sure to run **newaliases** each time you edit **/etc/aliases**. You do not have to run **newaliases** after changing an NIS alias database.

Potential Problems with the Aliases Database

A number of problems can occur with the alias database. They all result from a **sendmail** process accessing the **dbm** form of the database while it is only partially built. This can happen under two circumstances: one process accesses the database while another process is rebuilding it; or the process rebuilding the database dies before completing the rebuild.

The **sendmail** program uses two techniques to try to relieve these problems. First, it ignores interrupts while rebuilding the database; this prevents someone from aborting the process and leaving a partially rebuilt database. Second, at the end of the rebuild, **sendmail** adds an alias of the form:

```
@: @
```

which is not normally legal. Before **sendmail** accesses the database, it checks to ensure that this entry exists (**sendmail** performs the check only if the **a** option is specified in the configuration file). It waits as long as is specified with the **a** option (by default, five minutes) for this entry to appear, at which point it forces a rebuild itself (**sendmail** performs this operation only if the **D** option is specified in the configuration).

We recommend, however, that you do not have automatic rebuilds; you should edit **/etc/aliases** and then run **/usr/bin/newaliases**.

Owning a Mailing List

If an error occurs when mail is sent to a certain address, say *x*, **sendmail** looks for an alias of the form `owner-`*x* to receive the errors. This is typically useful for a mailing list where the submitter of the list has no control over the maintenance of the list itself; in this case, the person maintaining the list also owns it. For example:

```
unix-wizards: vance@sys1, pafford@sys2, nosuchuser,
              sam@sys3
owner-unix-wizards: owen@sys1
```

The **sendmail** command would send `owen@sys1` the error message that occurs when someone sends to `unix-wizards` due to the inclusion of the illegal user, `nosuchuser`, on the list.

Mailing Messages to a File or Program

You can send messages directly to files or programs as long as they are listed as alias members in **/etc/aliases**. Mail recipient files provide archival storage of messages. For example, you may want to log all messages sent to a particular mailing list. A mail recipient file specified in **/etc/aliases** need not previously exist. But if it does already exist, the file must not have execute permissions set.

You can also send a message to a program that captures the message as input for further processing. For example, a program might take all incoming messages and build a public bulletin board.

File and program mail recipients are distinguished by special characters. Every address that passes through the initial parsing for local addresses is scanned for the following prefix characters:

/ If the address begins with /, then it is treated as a filename.

| If the address begins with |, then the remainder of the address is executed as a shell command.

Suppose you want to send messages to a file named **stuff** in directory **cdg**. You want that same message to go to a program named **bboard**. So, you put both of these in **/etc/aliases** as follows:

salesgrp: Groucho, Harpo, Chico@sys1, /cdg/stuff, |bboard

Now, all you have to do is invoke **sendmail** with **salesgrp** as the recipient. Remember, **salesgrp** is an alias. You cannot, for example, send mail to **/cdg/stuff**, you must send it to **salesgrp**.

Using Include Files

You can use an "include" file as an alias member in **/etc/aliases**. For instance, you might have a file **/udd/worthy/programmers** that contains all the members of the Programmers basketball team.

To send mail to this group, put the following in **/etc/aliases**:

programmer: :include:/udd/worthy/programmers

This is a convenient and secure method for updating and maintaining an alias list, because regular users can update and maintain alias lists without having access to the file **/etc/aliases**. Here, regular users would update **/udd/worthy/programmers** instead. Each line in an include file such as **/udd/worthy/programmers** should be a comma-separated list of mail addresses.

Using Per-User Forwarding

As an alternative to the alias database, any user may put a file named **.forward** in his or her home directory. If this file exists, **sendmail** redirects mail for that user to the list of addresses listed in the **.forward** file. For example, if the home directory for user `jones` on sys12 has a **.forward** file with contents:

```
jones@sys13
jones@sys10
```

then any mail arriving for `jones` is redirected to the specified accounts, in this case, to both `sys13` and `sys10`.

You can forward mail to a user on a local host even if the aliases file specifies that mail for the user should be sent to another host. To do this, prepend a backslash to the username in **.forward**. The **sendmail** program strips the backslash and suppresses aliasing.

For example, suppose the aliases file on **sysA** contains the following entry:

```
owen:    owen@sysD
```

but that owen has the following **.forward** file in his home directory on **sysA**:

```
\owen
```

Any mail sent to `owen@sysA` would normally go to `owen@sysD` (because of the entry in the aliases file), but because of the entry in

the **.forward** file, the mail is placed in owen's mailbox on `sysA` instead. This applies only to mail sent to `owen@sysA`.

When every host on a network has an aliases file that contains an alias for every user, take care in using the **.forward** file. For example, if user `goolsby` set up a **.forward** file to forward mail for `goolsby` to `goolsby@sysA`, but the aliases file redirects any mail for `goolsby` to **sysD**, the mail will get caught in a loop between **sysD** and **sysA**. Mail would arrive for `goolsby` on **sysD**, the **.forward** file would forward it to **sysA**, the aliases file on **sysA** would direct it back to **sysD**, and so on. When the hop count was exceeded, **sendmail** would finally reject the message.

Using sendmail Logging

The **sendmail** program uses the logging facility provided by **syslogd**. Log output from **sendmail** is arranged as a succession of levels. At the lowest level, only extremely unusual events are logged. At the highest level the output is very verbose, with very routine events being recorded. In general, log levels under 10 are useful. Use log levels higher than 10 for debugging.

The log levels and their output are as follows:

0	Produce no log.
1	Log major problems only.
2	Log message collections and failed deliveries.
3	Log successful deliveries.
4	Log messages that have been deferred (for example, because the receiving host is down).
5	Log normal reception and delivery of messages.
6	Log unusual but harmless events (for example, trying to process a locked queue file).
9	Log internal queue ID to external message ID mappings. Use this level to trace a message as it travels between hosts.
10	Record host without an **MX** record.
12	Log miscellaneous messages for debugging.
16	Log verbose information regarding the queue.

See the **syslog.conf**(5) man page for details about how to set up logging output.

Interpreting sendmail Exit Status Messages

When an error occurs, **sendmail** returns an error number and an exit status message. Table 4–6 describes these error numbers and their status messages.

Table 4–6 sendmail Exit Status Messages

Error	Status Message	Meaning
0	EX_OK	Successful completion on all addresses.
64	EX_USAGE	The command was used incorrectly; for example, with the wrong number of arguments or an invalid flag.
65	EX_DATAERR	The input data was incorrect.
66	EX_NOINPUT	An input file did not exist or was not readable. This could also result from errors like "No message" to a mailer.
67	EX_NOUSER	Username not recognized.
68	EX_NOHOST	Hostname not recognized.
69	EX_UNAVAILABLE	Necessary resources were not available.
70	EX_SOFTWARE	Software error, including incorrect arguments.
71	EX_OSERR	A temporary operating system error; for example, cannot fork.
72	EX_OSFILE	A system file (such as **/etc/passwd** or **/etc/utmp**) does not exist, cannot be opened, or has an error (such as a syntax error).
73	EX_CANTCREAT	A user-specific output file cannot be created.
74	EX_IOERR	An error occurred while reading from or writing to some file.
75	EX_TEMPFAIL	A temporary failure; for example, a mailer could not create a connection.
76	EX_PROTOCOL	The remote system returned something that was "not possible" during a protocol exchange.
77	EX_NOPERM	You did not have permission to perform the operation. This message is not intended for file system problems, which should use `NOINPUT` or `CANTCREAT`.
78	EX_CONFIG	Configuration error.

End of Chapter

5 Configuring and Using the Domain Name System

This chapter describes how to configure and use the domain name system (DNS), which is a distributed database that lets hosts on the Internet share information. This information can include the name and Internet address of a host, the Internet address of a host that knows how to deliver mail to locations outside its local network, a list of well-known services associated with a host (for example, **telnet**, **ftp**, and so on), or the operating system and hardware for a host.

If this is the first time that you have worked with the domain name system, read the entire chapter before you attempt set up the system. If you have some experience with the DNS, you may want to start with the section "Setting Up the Domain Name System."

Understanding the Domain Name System

The following sections explain the principles of the domain name system. The first section describes the rationale for using the DNS. The next section provides an overview of the domain name space, which is an abstract name space designed to accommodate a large set of names and associated information. The last section describes how the name space can be divided into administrative subdivisions called zones.

Why Use the Domain Name System?

Formerly, when a program needed to contact a host on the network for which it only had a hostname, it accessed a fixed list of hostname/Internet address pairs, obtained the required pair, and used the obtained address to contact the host. Traditionally, hostname/Internet address pairs for the entire Internet were maintained by the Defense Data Network – Network Information Center (known as the NIC) in a file called **hosts.txt**. Every host administrator was responsible for obtaining a new copy of **hosts.txt**, either from the NIC or from some intermediary, when it was updated. Programs accessed a file derived from **hosts.txt** when they needed a particular hostname/address pair.

As the number of hosts on the Internet increased, the administration of the **hosts.txt** file became more difficult. The domain name system was developed to ease this administrative burden.

With the DNS, to get a particular hostname/address pair, a program needs to contact a name server process and request the information.

Because information is so distributed, a name server keeps only a small database that contains information about a subset of the entire set of hostname/address pairs and a list of other name servers that maintain information about other subsets. A name server also needs a method for finding the appropriate name server to query for any other piece of information.

Understanding the Domain Name Space

Every computer system can be assigned a symbolic name that uniquely identifies it. When you connect computer systems to form a network, you can assign names using any name structure that you want. However, as more systems connect, choosing unique and meaningful symbolic names becomes more difficult. A hierarchical domain name space was invented for the Internet community to accommodate a large, complicated set of names.

As Figure 5–1 shows, the domain name space is a tree structure. A *domain* is a subset of the name space from a particular point in the hierarchy to the leaves of the tree. This diagram does not accurately represent the domain name space, but it does help explain the structure of the name space.

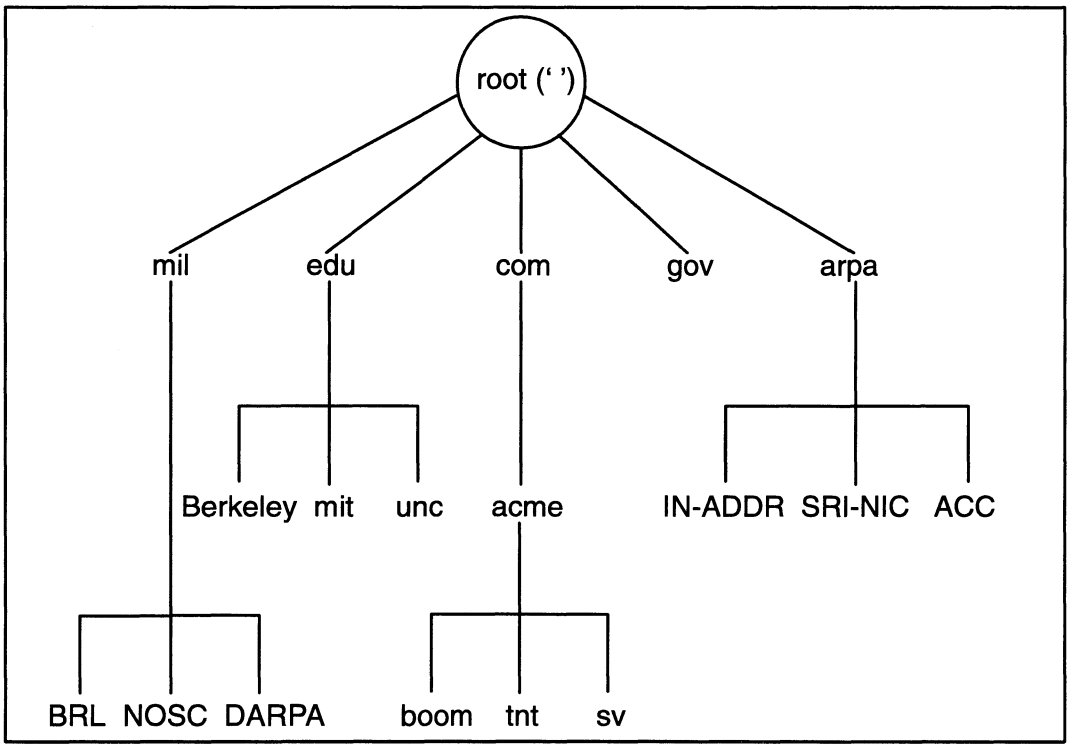


Figure 5–1 A Hierarchical Domain Name Space

Notice that each node (branch point and leaf) of this tree structure has a label. The root of this tree is labeled with the null character ("). Next on the tree come labels that correspond to top-level domains, such as **com**, for commercial organizations, **mil**, for military groups, **edu**, for educational institutions, **gov**, for government agencies, and **arpa** for the ARPANET. After these come nodes labeled **Berkeley** under **edu**, **acme** under **com**, and **SRI-NIC** under **arpa**. (The DNS is case-insensitive.) Finally come labels such as **boom**, **tnt**, and **sv** under **acme** (hypothetical domains created specifically for this chapter).

You obtain a *domain name* from any ordered grouping of labels, tracing from a given node of the tree up to the root, separating the labels with dots. You name a low-level domain by tracing from a leaf node to the root. For example, **tnt.acme.com.** could be a low-level domain for a single facility of a large company. The next level domain up the tree would be named **acme.com.**, for the entire company. In addition to **tnt.acme.com.**, the domain **acme.com.** includes the domains **boom.acme.com.** and **sv.acme.com.**, which could serve other facilities of the same company. The top-level domain would be **com.**, which, as you read, is for all commercial organizations. And, as mentioned earlier, the root domain, which is the parent of all domains, is labeled with the null character.

A trailing dot in a domain name specifies a name relative to the root domain. When you specify a domain name without this trailing dot, it denotes a partial domain name that is relative to some known origin. When you do use the trailing dot, you use something called a fully qualified name; the origin is the root domain. More about this later in the chapter.

The domain name space is organized so that the order of labels in a domain name reflects the relationship of domains to one another. Thus, the low-level domain **tnt.acme.com.** is a subdomain of **acme.com.**, **com.**, and the root domain. The domain **acme.com.** is a subdomain of **com.** and the root, and so on. The sequence of labels within any domain name reflects the naming hierarchy; each position in the sequence refers to a different hierarchical level. Different hierarchical levels are served by one or more name servers, some of which have authority over specified subdivisions of the name space.

The Internet authority that has responsibility for the root domain is the NIC. Table 5–1 shows how the NIC has partitioned the top-level domains.

Table 5–1 Top-level Internet Domains

Domain Name	Description
arpa	ARPANET domain. Hosts that fall into this domain eventually move to other branches of the domain tree.
com	Commercial organizations
edu	Educational institutions
gov	Government agencies. Restricted to the United States.
int	International organizations
mil	Military groups. Restricted to the United states.
net	Network administrative and service entities (for example, information centers or operations centers)
org	Organizations that do not clearly fall within the other top-level domains (for example, technical support groups or professional societies)
country_code	Countries (for example, **fr** for France or **us** for the United States)

Below these top-level domains are a variety of domains for which responsibility is widely distributed. It is at this level that you will likely establish a domain once you have acquired permission from the proper authority. How you do this is discussed at the end of the chapter.

Delegating Authority in the Domain Name Space

Control and management responsibility for a domain can be changed along any arc of the domain name space by assigning something called a *delegation* at a particular node. Delegations divide the name space into administrative subdivisions called *zones*.

Assume that our large company wanted to manage its own name space, and got the owner of the root domain, the NIC, to delegate the **acme.com.** zone. Figure 5–2 shows how the owner of the **acme.com.** zone may divide it into different zones:

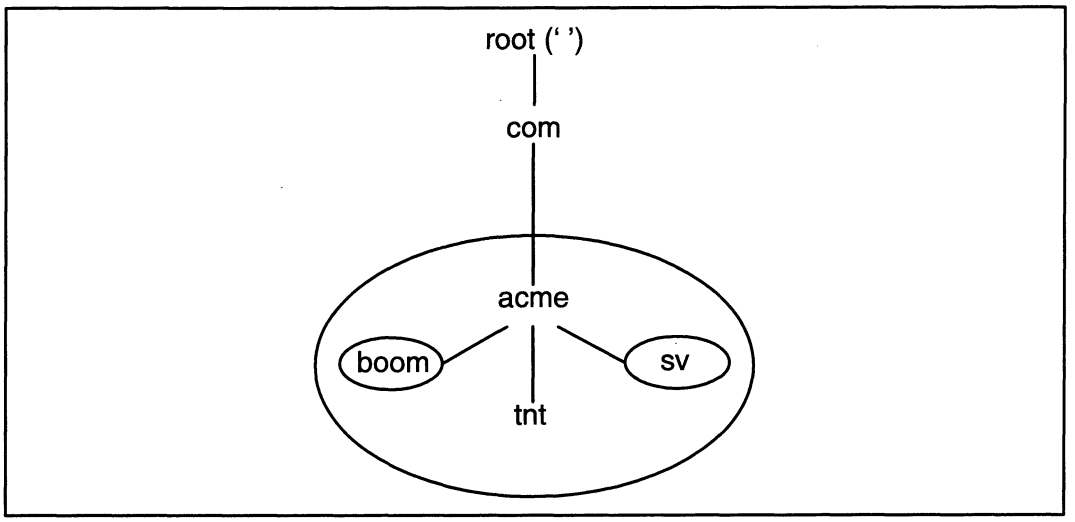


Figure 5–2 Delegation of Zones

You identify zones by the names of the associated topmost nodes (closest to the root). In the figure, there are four zones: the root (remember, **com.** is managed by the authority for the root), **acme.com.** (**tnt.acme.com.** is managed by the authority for **acme.com.**), **boom.acme.com.**, and **sv.acme.com.**.

A zone is always connected through nodes of the tree. It can begin at any level of the hierarchy and can include domains from a variety of levels. The owner of a zone can create new names and associated data, and can create new subordinate domains. The owner of **acme.com.** created three subdomains: **boom**, **tnt**, and **sv**. Such subdivisions can continue indefinitely. Once it delegates the authority, the only direct control that a parent zone retains over its children is its link to the child zone and all of its descendants. The parent zone can delete the link to excommunicate the child zone and all descendants of the child.

Typically, control of a subdomain is delegated to a different authority, but this is not required. In the example above, the authority for **acme.com.** delegated **boom** and **sv** to other organizations within the company. These subdomains, because they have a different authority from their parent domain, can properly be called subzones. Now, the authorities for **boom.acme.com.** and **sv.acme.com.** can create subdomains independently of their parent zone. The subdomain **tnt.acme.com.**, though subordinate to **acme.com.**, is controlled by the same authority as its parent zone. Only the authority for **acme.com.** can create subdomains for **tnt.acme.com.**

So, the term domain refers to a node or portion of the name space without regard to its management. In contrast, the term zone refers

to the administrative authority and control of data about the name servers and hosts in a division of the name space.

DNS Components

The domain name system comprises two components: the *resolver* and the *name server.* The resolver consists of procedures that are linked in with the programs that call it. The name server runs as a process called **named**. When a program wants a hostname/address pair or some other information, it makes a procedure call to a resolver routine. When the resolver gets a request from a calling program, it queries **named**.

Whenever a name server gets a query, it checks whether the name lies in the zone for which it has authority. If it does have the authority and does have the information, a name server consults its database and answers the query with an *authoritative* response. If it does not have the information, it tells the sender that it does not know the answer.

If the name does not lie in the zone for which it has authority, a name server consults its database and, if it has the information, responds to the query with a *nonauthoritative* answer. If a name server does not have the information, it takes one of two approaches (see **resolver**(3) for details):

- The *iterative* approach. The name server refers the program to another name server and lets the program pursue the query.
- The *recursive* approach (the default). The name server pursues the query itself. This means that the name server forwards the query to other name servers until it finds one whose database contains the requested information. Once it gets the information from another name server, the first name server passes it on to the requesting program.

Name servers at different sites operate cooperatively. Figure 5–3 illustrates the name servers operating for the **acme.com** zone and its subordinate zones.

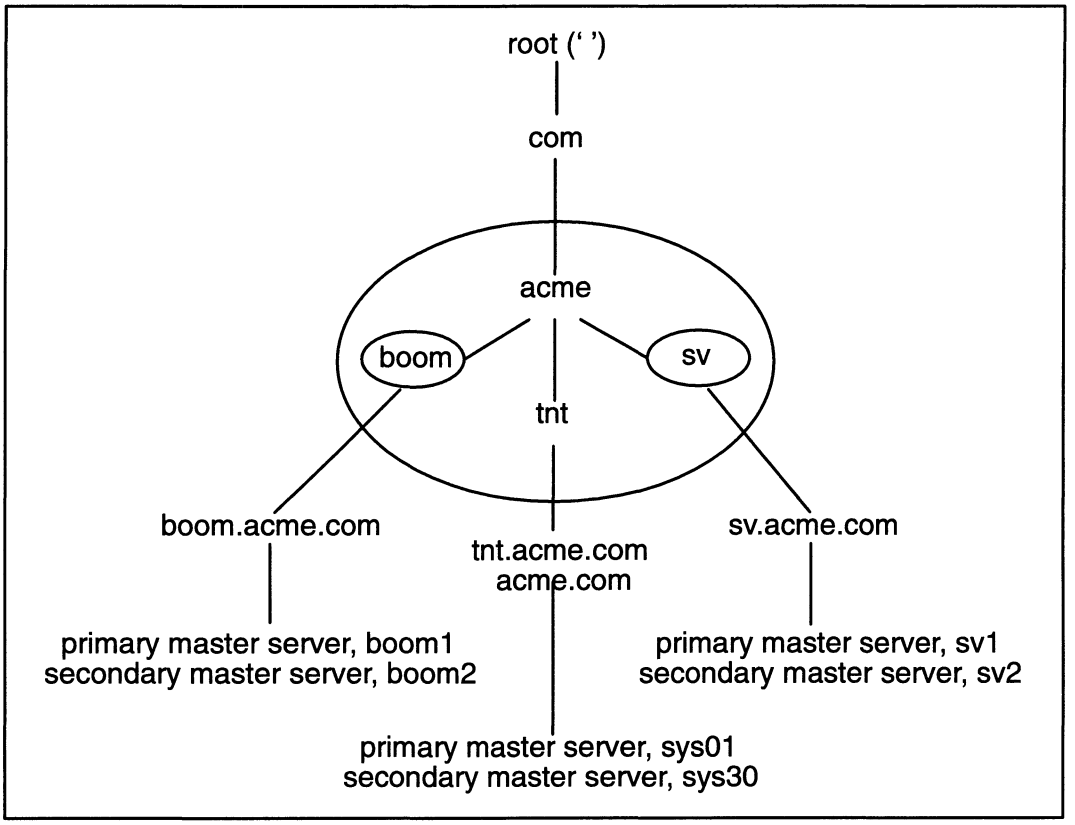


Figure 5–3 Name Servers Operating for Three Zones

Figure 5–3 shows different types of name servers, for example, primary and secondary master servers. Do not be concerned about these different types; they are described later in this section. You can set up a variety of types of name servers within a zone, at least one name server in each zone must maintain authoritative information about the zone.

A zone should be accessible through at least two name servers to ensure the availability of data about it when hosts or communication links fail. These name servers can be inside or outside the zone itself. Many zones have more redundancy than that. However, the more authoritative name servers you have per zone, the higher your maintenance cost.

Files Shipped with the Domain Name System

TCP/IP for AViiON Systems includes the resolver and the name server files listed in Table 5–2. Each file is described later in the chapter.

Table 5–2 Files That Support the Domain Name System

File	Description
/usr/lib/libc.a	This library contains the resolver routines.
/etc/resolv.conf	This file, which the resolver reads during initialization, provides information about the current domain name and specifies name servers to query if no local name server is running.
/usr/bin/named	The name server of the domain name system.
master files	The name server is supported by five master files: a boot file, a cache file, and three data files (hosts, local hosts, and reverse).
/usr/bin/nslookup	This program can query Internet name servers for information about various hosts and domains or can print a list of hosts in a specific domain.
/etc/svcorder	This file lets you configure the order and the means of performing name/address resolution through the contents of this file.

Understanding the Resolver

The resolver is the interface between user programs and name servers. Specifically, the resolver comprises a group of functions that are included with **/usr/lib/libc.a**. These functions include routines to initialize the resolver, format and forward queries to **named**, and relay responses to the requesting program (for example, **telnet** or **ftp**) in a form that the program can use.

Resolver functions are linked in with the programs that call them. When a program calls the resolver, the resolver attempts to access a configuration file, **resolv.conf**. This file specifies the current domain and lists the name servers to which the resolver should send requests.

The resolver formulates a query from information a network program provides it and forwards this query to a name server on its list. In the Internet, queries are carried in UDP datagrams or over TCP connections. A query contains a name to be resolved, a declaration of the class of the name, the type of mapping needed, and a code that specifies whether a name server should translate the name completely. For details about the structure of a query, see *RFC 1035 (Domain Names— Implementation and Specification)*.

The resolver sends queries to the name servers on its list in the order specified until one returns a response. The response by a

name server to a query either answers the question posed or signals some error condition. If an error occurs, the resolver deals with it.

If no servers are available, the resolver retries each of them a specified number of times before it returns a failure. Because a resolver may need to consult several name servers, the amount of time that a resolver takes to resolve a name/address pair can vary quite a bit, from milliseconds to several seconds.

If the resolver cannot find a configuration file, it accesses the name server on the local host. If there is no name server on the local host, the resolver returns a failure.

Understanding the Name Server

The name server runs as a process called **named**. When **named**, starts, it reads a boot file to determine which zones it has authority over, where it should get information to initially load a cache that it maintains, and where the files that contain the zone information reside. **named** listens for both UDP and TCP queries and answers queries sent from a resolver or from another name server. **named** processes those queries and sends back responses.

named maintains a cache of records that it consults when it tries to answer queries. Maintaining a cache (as opposed to querying other name servers or accessing a file) saves **named** time when it looks up data for which it frequently gets queries. When you start up **named**, it reads records to be cached from the file **root.cache**, and places them in a cache. After startup, the cache grows dynamically in memory, made up of records returned as responses from previous queries. Each record has a time-to-live field whose value indicates how long the information contained in the record should be kept in the cache. A maintenance function of **named** periodically cleans the cache of records older than the time-to-live value.

As you have seen, you can parcel the domain name space into administrative subdivisions called zones, and you can distribute authority for zones among name servers. While a name server can have several optional functions and sources of data, its essential task is to answer queries using the authoritative data for its zone. By design, name servers answer queries simply; the response always contains either the answer to the question or a response that the name server does not have the answer.

A master name server generally supports one or more zones, but zones are typically delegated so that a name server has authoritative data about only a small section of the domain tree. A name server may also have some cached nonauthoritative data about other parts of the tree. This saves time since accessing

memory is quicker than contacting another **named** process. A name server marks its responses to queries so that the requester can tell whether the response comes from authoritative data or not. Authoritative name servers are responsible for making sure that their data is updated in a timely way.

How Name Servers and Resolvers Work Together

Figure 5–4 shows interaction between a resolver and name servers on a single host that runs both programs.

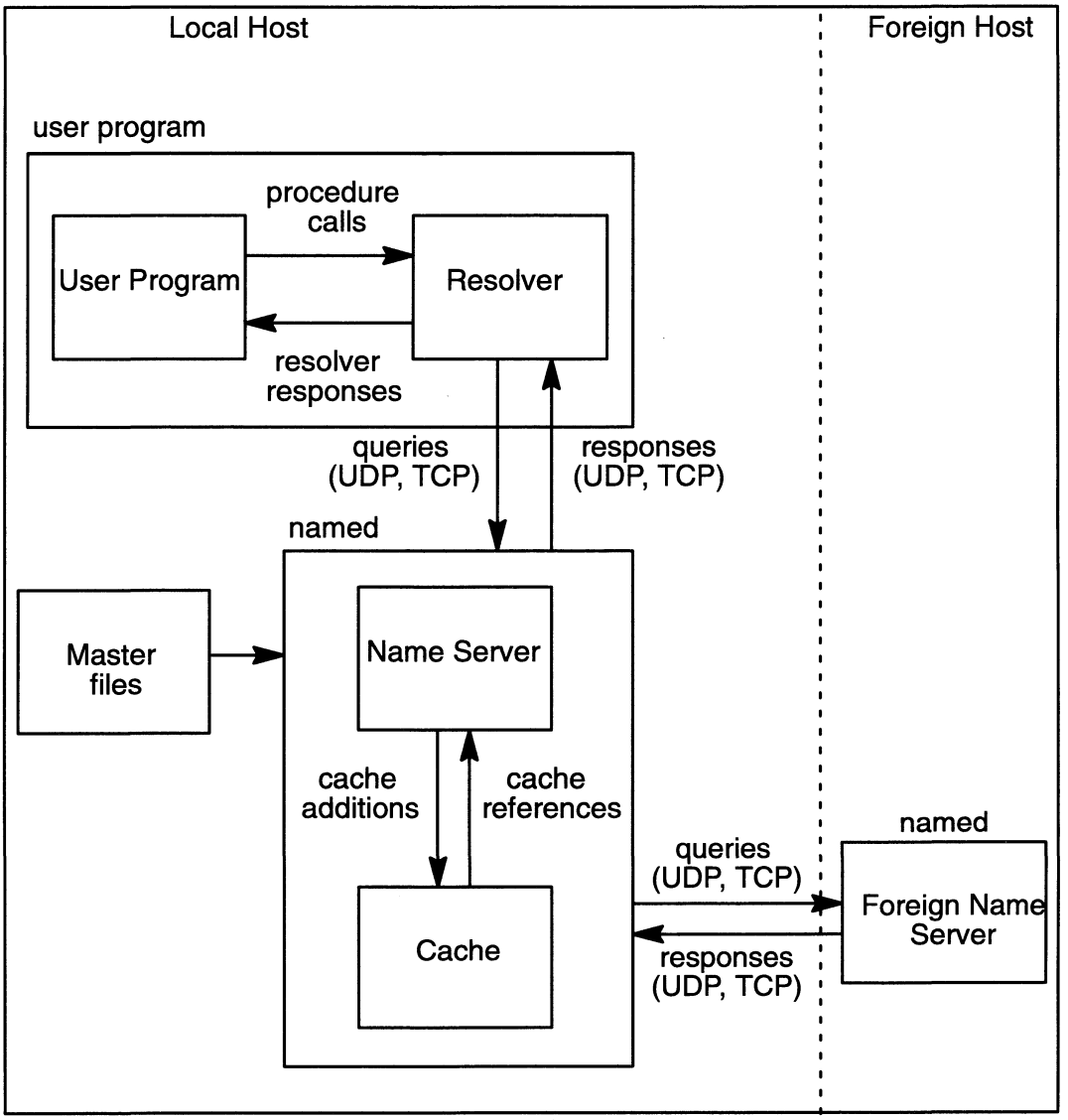


Figure 5–4 Information Flow in a Host with a Resolver and a Name Server

In this figure, a user program requests name/address resolution through procedure calls to the resolver. The resolver queries the name server through UDP packets or a TCP connection. Which

protocol the resolver uses depends on whether a particular option is enabled; see **resolver**(3) for details. The name server gets data from the master files that **named.boot** told it to load. These master files include the boot file, the cache file, and three data files.

The local name server communicates with other name servers through UDP or TCP to obtain information for queries it cannot answer by accessing its local database, and to obtain updates from zone transfers. A zone transfer causes a name server to clean the file of records older than the specified time-to-live value, and to obtain a new copy of the database. With zone transfers, redundant name servers for a particular zone automatically remain consistent.

You generally do not maintain a name server on every host in your local network. Figure 5–5 shows information flow in a local host that supports just the resolver and a remote host that runs **named**.

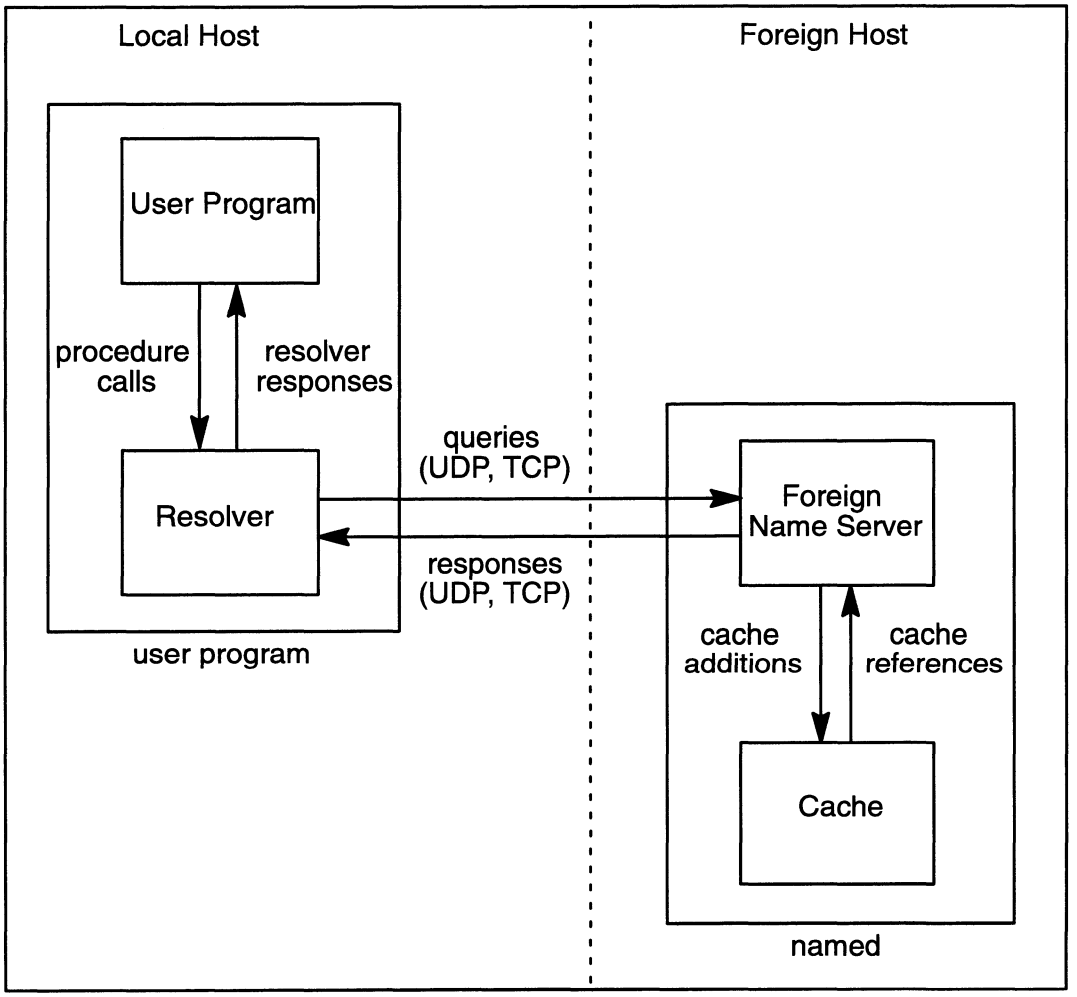


Figure 5–5 Flow Between a Local Resolver and a Foreign **named** Host

In the figure, only the domain name resolver is running on the local host, so when the system starts, the resolver checks the

/etc/resolv.conf file to determine the address of a name server that services its queries. A user program makes a call to its local resolver. The resolver then sends a query to a foreign name server for processing. This foreign server could be running on a host in another part of the country, but it is just as likely that it is running on another host on the same local network. If the foreign server has the requested information, it answers the query. Otherwise, it forwards the query to another foreign server.

Distinguishing Domain Name Server Types

Once you obtain the delegation for a zone, you must choose distinct roles for the name servers within the zone. The domain name system permits primary and secondary master servers, forwarder and slave servers, caching-only servers, and remote servers.

A name server can play more than one role at a time, depending on how a zone is set up and on which zone is involved. For example, one DNS client could view a name server as a remote server, but another client could view that name server as a caching-only server. Similarly, a name server could be the primary master server for one zone, but the secondary master for a different zone.

What Is a Master Server (Primary or Secondary)?

A *master server* for a zone is the name service authority for that zone. That is, a master server maintains all authoritative data for its zone. Each zone should have at least two master servers, a primary master and one or more secondary masters to provide backup service should the primary master become unavailable or overloaded.

You are not limited to two master servers. A single name server can be a master server for multiple zones, being primary for some zones and secondary for others. Thus, the system you implement can have as many primary and secondary master servers as you require.

A *primary master server* is a name server that loads the authoritative data for a zone from files on disk. Remember, having authority for a zone means having control and management responsibility for the name servers and hosts within the zone. The administrator of a primary master name server for a given zone can change the authority along any arc of the zone's name space by assigning a delegation at that point. Then, a primary master server for the new subzone can be set up at that node.

A *secondary master server* is a name server that receives its data from a primary master server. At boot time, the secondary server

requests all the authoritative data for its zone from the primary server. The secondary server then periodically checks with the primary server to see if it needs to update its data. The process whereby a secondary server gets data from a primary server is called a *zone transfer.*

Depending on how you set it up, a secondary master server puts data for the zone into a file as a backup. When you first start such a secondary name server, it loads data from this backup file if possible. Then, it consults the primary master server to check that the zone information is still up-to-date. It does this by checking a specific field of a specific record in one of the master files. This process is covered in detail later in the chapter.

What Is a Slave Server and a Forwarder Server?

A *slave server* runs on the loopback interface (**127.0.0.1**) and has no authority for a zone. Such a server always directs queries to a fixed list of *forwarder servers* in the same zone when it cannot answer a query with information from its local database. A slave server does not direct queries to master servers for the root zone or for foreign zones.

A forwarder server is simply the target of a slave server's queries. It can be a primary or secondary master server, or a caching-only server. For each slave server, you can specify one or more forwarder servers, and a slave-server tries each of them in turn until it exhausts the list.

Typically, you would use a slave and forwarder configuration when you do not want name servers at a given site to direct queries to or get queries from name servers on the rest of the Internet. A typical scenario would involve a number of workstations and a departmental computer with access to the Internet. The workstations may be prohibited from having Internet access. To give the workstations the semblance of access to the Internet domain system, they could be set up as slave servers to the departmental machine, which would forward queries and interact with name servers outside the zone to resolve a query.

An added benefit of using a slave/forwarder configuration is that the forwarders develop a complete cache of information that all the slaves can use. Queries may take less time than in a configuration using a remote server, with the cost being the resources to run the name server on the local system.

What Is a Caching-Only Server?

A *caching-only server* runs on the loopback interface and has no authority for a zone. This name server services queries and gets

information from other name servers who do have zone authority. In contrast to a slave server, a caching-only server can direct queries to master servers for the root zone or for foreign zones.

On a caching–only server, `localhost` is cached but `hostname` is not. To force the value of **hostname**(1M) into the cache, type the command:

```
ping 'hostname'
```

What Is a Remote Server?

A *remote server* is a name server designated to handle queries from resolvers on one or more foreign hosts. From another perspective, using a remote server means running only the resolver on a local host and querying a name server on a remote host. Any type of name server can act as a remote server. The term "remote server" does not connote any degree of zone authority. A simple scenario should illustrate the utility of remote servers.

Suppose you want to provide domain name service on a workstation or on a host that has a limited amount of memory or CPU cycles. You could set up **resolv.conf** to contain addresses of remote servers. You would then run all of the networking programs that use domain name service without invoking **named** on the constrained host. When these networking programs attempt to resolve a name/address pair, they call the appropriate resolver routines. The resolver then directs all queries to the remote name servers.

Thus, workstations on the network can take advantage of the caches on the other hosts running the name server. The workstation avoids the overhead associated with running the name server. Queries usually take a small amount of time to get from workstations to the remote server.

As you may gather from the preceding sections, the essential ideas of the domain name system are simple and powerful, but implementation of the system can become complex. The software permits you considerable latitude with its configuration and implementation within a given zone.

Setting Up the Domain Name System

Before you run any domain name system software, you should decide what type of name servers you intend to run and where you intend to run them. You should also decide how many hosts within the zone you want serviced.

Before you set up any resolver configuration files or name server master files, you should answer the following questions.

- What is your domain name? What is the name of your parent domain? Will there be any subordinate domains to your domain? It may help to draw a tree that illustrates the name space you intend to administer, indicating where the name servers will reside.
- In what order will you use the Network Information System (NIS), domain name system, and the **/etc/hosts** file for address resolution?

You should also decide which hosts will rely on a local **named** and which hosts will rely on a remote **named**.

- If will not run the name server on a given host and will use only the resolver, you must obtain the Internet addresses of the remote servers to be queried.
- If you will run the name server on a host, you must decide what type of server you intend to run.

If a name server will be a primary master server, you must obtain the names and addresses of all hosts in the zone over which the name server has authority. You must also obtain names and addresses of all secondary master servers for the zone, whatever backup name servers serve the zone, and of primary name servers for the root domain. Optionally, you can obtain names and addresses of master servers in other zones with which you want to communicate.

If a name server will be a secondary master server, you must obtain the names and addresses of the primary master server and whatever backup name servers serve the zone. Optionally, you can obtain names and addresses of master servers in other zones with which you want to communicate.

If a name server will be a slave server, you must obtain the names and addresses of its forwarder servers. A primary or secondary master server or a caching-only server can act as a forwarder server.

If a name server will be a caching-only server, you must obtain names and addresses of root servers and other master servers to be queried.

How do you decide which type of name server to run where? The answer depends on how your local networks are laid out, and on how you decide to set up your zones. Once you answer these questions, you can use the sections that follow to set up the appropriate components of the domain name system.

Setting Up a Resolver

You should always edit the resolver's configuration file, **resolv.conf**, to specify the default domain for the resolver. If you will use remote servers from a host, you must also edit **resolv.conf** to specify the Internet addresses of those name servers. You can designate up to three name servers on the network that should be sent queries.

Here is an example of a **resolv.conf** file:

```
domain tnt.acme.com
nameserver 128.223.1.26
nameserver 128.9.0.32
```

The resolver reads the **/etc/resolv.conf** at initialization. Afterwards, any time it gets a request, the resolver queries name servers in the order you have specified: in the example above, 128.223.1.26 is queried first by the resolver. Once **/etc/resolv.conf** is in place, domain name service is complete for hosts that use only remote servers to answer queries. In all other circumstances, you must also set up a name server.

Setting Up a Name Server

Setting up a name server requires that you edit its master files. The name server's master files include a boot file, a cache file, and three data files.

Understanding the Name Server's Master Files

The boot file is called **/etc/named.boot**. When you start **named**, it reads this file to find out what kind of server it is, which zones it has authority over, and where to find its cache file and its data files (if any). You can specify another location for the boot file on the **named** command line; this is described later in "Using the Domain Name System."

By default, the cache file is named **root.cache**. This file should contain the names and addresses of the authoritative name servers for the root domain of the network. These names and addresses become the initial entries for **named**'s cache. The entries are always kept in the cache.

Every name server requires at least one data file, namely, the local host file. A resolver sends queries to a name server through the loopback interface (**127.0.0.1**) when you do not set up the resolver's configuration file. You can name the local host file whatever you wish. We refer to it as *localhost.rev.*

When you set up a primary master server, you also must set up two other data files: the hosts and reverse files. You can name these files whatever you wish. We refer to them as *named.hosts* and *named.rev*.

Table 5–3 summarizes the name server's master files. For more information about how to specify another location for these data files, see "Defining the Name Server Directory" later in this section.

Table 5–3 Master Files for a Domain Name Server

File	Description
/etc/named.boot	The name server reads this file at startup to find out what kind of name server it is, which zones it has authority over, and where to find its cache file and its data files.
root.cache	Specifies the names and addresses of authoritative name servers for the root domain of the network.
localhost.rev	Specifies the address for the local loopback interface. Copy the prototype file to the directory that you specify in the boot file and edit appropriately for your site.
named.hosts	Contains resource records about the name servers and hosts in a particular zone. Also contains glue records about the name servers for subordinate domains. Copy the prototype file to the directory that you specify in the boot file and edit appropriately for your site.
named.rev	Allows Internet-to-hostname mapping. Copy the prototype file to the directory that you specify in the boot file and edit appropriately for your site.

Before you edit a name server's master files, you should understand their contents. The **named.boot** file, the **root.cache** file, and each of the data files contain a sequence of records. A record is generally one line long, but you can use parentheses to continue a list of items across a line boundary. For example, you will see how the start of authority record in a name server's data files always covers more than one line.

The contents of a record must obey certain syntax rules. Text literals in a record can contain CRLF (Ctrl-M Ctrl-J). Any combination of tabs and spaces acts as a delimiter between items in a record. The name server treats anything from a semicolon (;) to the end of a line as a comment.

Records in the boot file and the cache file have relatively simple syntax rules. Records in the data files, which are called resource records, have a more complicated set of syntax rules. These syntax rules are described in detail later in the chapter.

Specifying Domain Names in Master Files

Any domain name should conform to the syntax rules listed below. The | symbol represents an OR operation.

domain_name = *subdomain* | " "

subdomain = *label* | *subdomain.label*

label = *letter* [[*ldh-str*] [*letter* | *digit*]]

ldh-str = *let-dig-hyp* | *let-dig-hyp ldh-str*

let-dig-hyp = *letter* | *digit* | *hyphen*

where:

letter is an upper- or lowercase alphabetic letter.

digit is a number from 0 to 9.

hyphen is a hyphen.

. is the delimiter.

A *label* must follow the same rules as for ARPANET host names. Labels must start with a letter, end with a letter or digit, and have as interior characters only letters, digits, and hyphen. The length of a label cannot exceed 63 characters. Both upper- and lowercase letters are allowed in domain names. The name server is case insensitive.

Select a domain name that satisfies both the rules of the domain system and any rules for the thing you are naming, whether the rules are published or implied by existing programs. For example, a mailbox name should meet both the restrictions in *RFC 822 (Standard for the Format of ARPA Internet Text Messages)* and *RFC 1034 (Domain Names — Concepts and Facilities)*. A hostname should meet the restrictions in *RFC 952 (DOD Internet Host Table Specification)*.

There are exceptions to these syntax rules. For example, **3COM.COM** and **3M.COM** are valid domain names. However, to ensure the fewest difficulties with the widest range of application programs, follow these rules.

Editing a Name Server's Boot File

When **named** starts up, it reads its boot file to find out what type of server it is, which zones it has authority over, and where to get its initial data. As you have read, the boot file is named **/etc/named.boot** by default.

The file **named.boot** consists of a series of lines that define the operating characteristics of the name server. Before you start **named**, edit **named.boot** to define how your name server should operate. That is, edit the file to do the following:

- Define the name server directory.
- Specify the name of the cache file (**root.cache** by default).
- Specify the name of the *localhost.rev* file.
- If you are setting up a master server, specify zone authority; that is, specify domains for which this server is the primary master server, or specify domains for which this server is the secondary master server.
- If you are setting up a caching-only server, omit lines to define a primary or secondary master server.
- If you are setting up a slave server, include a line that indicates so and specify the forwarder servers.

Minimally, **named.boot** should contain records to define what sort of server it is. For example, if the name server is a primary server, the boot file should contain records that indicate such and that specify the zone of authority.

You can specify a different name and location for the boot file on the **named** command line when you start it up. For example,

```
% named bootfile ↵
```

where *bootfile* may be a filename or a full pathname.

Defining the Default Domain for the Name Server

You can specify a default domain for the name server using a line in the boot file such as this:

```
domain  tnt.acme.com
```

Most often, you do not need to specify a default domain for the name server.

Defining the Name Server Directory

The directory line specifies where the name server should run. This lets you specify other filenames in the boot file using relative pathnames. If you put the following line in your boot file, then the name server looks in **/etc/domain** for any filenames that you specify in later lines of the boot file.

directory /etc/domain

There are two purposes for the directory line. First, you use it to make sure **named** is in the proper directory when it tries to include files by relative pathnames or with **$INCLUDE**. Second, you use it to let **named** run in a location where it is reasonable to dump core if necessary.

The directory line is optional. There are some prototype files in **/etc/domain** you can use as examples.

Specifying the Location of the Cache

Every name server, regardless of type, must have a **cache** line in its boot file to tell it where to go to find initial entries for its cache. You do not need to specify a full pathname for the cache file if you specify a directory line.

cache . root.cache

Here, the name server looks for initial cache entries in the file **root.cache**. The free-standing dot indicates that the root domain can be reached through the information in this file. The servers listed in this file may be root servers, or they may be other name servers that know how to get to the root. You must specify one cache entry with a free-standing dot because every name server has to prime its cache, providing the server with information about the root.

The **named** process reads all cache files listed when you boot it. Any values still valid are reinstated in the cache. The root name server information in the cache files is always used.

Suppose you save a file of records and used that file as a cache file in addition to the root server's cache. Entries from your file that have not timed out would be kept, but the root server's cache entries are always kept. For information on cache files, see the section "Understanding the Cache File."

Specifying Zone Authority

If you are setting up a master name server, you must include an authority line in the boot file, which specifies the zone for which the server. This example specifies a zone for a primary server:

primary tnt.acme.com hosts.tnt

The authority line contains three fields: a field to specify type of authority, a zone field, and a filename field. In this example, the first field specifies that the name server is primary for the zone specified in the second field, **tnt.acme.com**. The third field indicates that the *named.hosts* file is **hosts.tnt**. This file is where the name server gets data about the hosts over which it has authority.

Specifying a secondary server is similar to the line to specify a primary server, except that you must also list Internet addresses of other name servers (usually primary servers) from which the secondary server obtains the data about the zone.

secondary tnt.acme.com 128.223.0.10 128.223.0.4 hosts.tnt.bak

In the example above, the first two fields specify that the name server is a secondary server for the **tnt.acme.com** zone. The two network addresses that follow specify the Internet addresses of the name servers that are primary for the zone. You can specify as many as 10 Internet addresses. The secondary server gets its data across the network from these servers. The secondary server tries each name server in the order listed until it receives data.

Optionally, you may specify a filename after the list of primary server addresses. Data for the zone is put into this file as a backup. In the example above, the file specified is **hosts.tnt.bak**. When you first start a secondary name server, it loads data from this backup file, if possible. Then, it consults a primary server to check that the zone information is still up to date.

Specifying a Caching-Only Server

To set up a caching-only server, do not specify any authority lines for the local zone in **named.boot**.

Specifying Forwarder Servers

Any server can make use of a forwarder, which is a server that tries to resolve queries on behalf of other systems. Specify forwarders in the boot file by Internet address as follows:

forwarders 128.223.20.10 128.223.0.4

There are two reasons to specify a forwarder. First, other systems may not have full network access, and thus may be prevented from sending any IP packets onto the rest of the network. Second, a forwarder sees a union of all queries as they pass through its server.

Thus, a forwarder builds up a rich cache of data in contrast to the cache in a typical workstation server. In effect, a forwarder becomes a metacache from which all hosts can benefit. Having such a metacache reduces the total number of queries from the site with a forwarder to the rest of the net.

Specifying a Slave Server

Specify a slave server if you lack access to the Internet, and the use of forwarder servers is the only way to resolve queries. You also can use a slave server to force the name server to use the listed forwarder servers.

Activate a slave server by placing the following line in the boot file:

slave

If you activate a slave server, you must specify forwarder servers. When a name server acts as a slave, it forwards each query to each of the forwarders until it finds an answer or the list of forwarders is exhausted.

Specifying the sortlist Parameter

Use the **sortlist** parameter to indicate which address should be returned first when a query is made for a host that has several address records associated with it. Address records are discussed later in the chapter.

You specify one or more network addresses with **sortlist**. For example,

sortlist 128.223.0.0 10.0.0.0

With this **sortlist**, a name server that gets an address query for a host with addresses 128.223.5.12 and 10.3.5.4 return the 128.223.5.12 address first because 128.223.0.0 was specified first in the sort list.

When you specify network addresses with the **sortlist** parameter, put an address that applies locally first on the list. Consider the following network configuration.

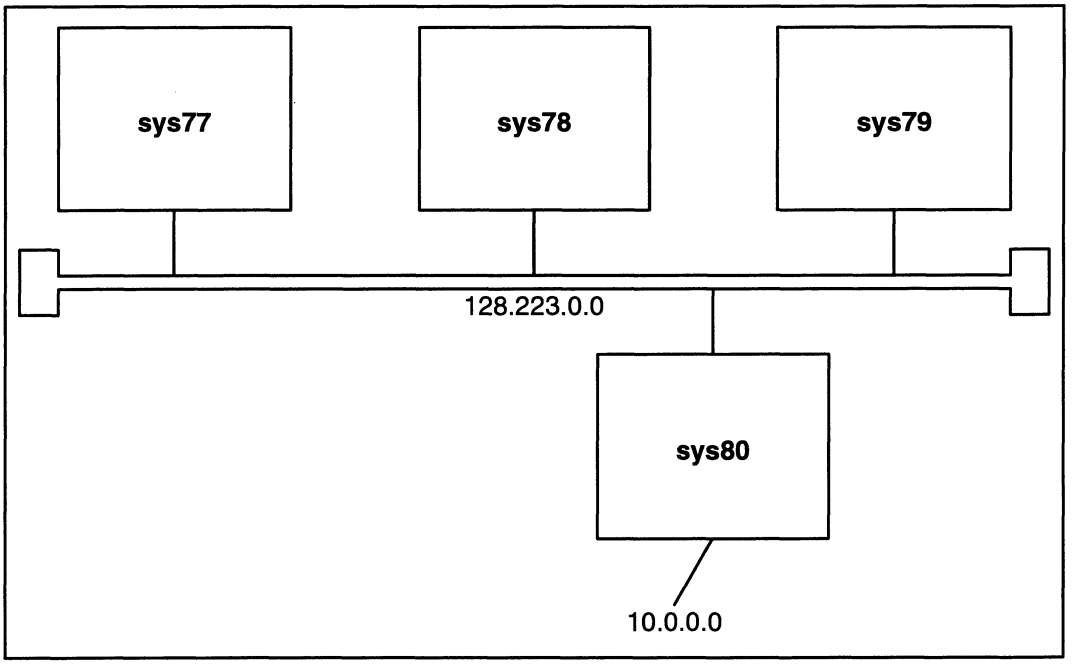


Figure 5–6 Network Configuration Where Sortlist Would Be Useful

The host **sys80** has a connection to **128.223.0.0** and a connection to **10.0.0.0**. The host **sys80** is the authoritative server for a zone that includes **sys77**, **sys78**, and **sys79**. The **sortlist** line should be as above, with **128.223.0.0** appearing first in the list. This causes queries by **sys77**, **sys78**, and **sys79** for the address of **sys80** to be returned with the **128.223.***x.y* address first.

Sample Boot Files

The following example shows a sample boot filecalled **named.boot**, for the primary master server.

```
;
; For tnt.acme.com., the SOA is defined in tnt.hosts
; For 223.128.IN-ADDR.ARPA, the SOA is defined in tnt.rev
; For 0.0.127.IN-ADDR.ARPA, the SOA is defined in localhost.rev
;
sortlist 128.223.0.0  10.0.0.0

directory       /etc/domain
;
; type          domain                  source/host file
;
cache           .                       root.cache
primary         acme.com                acme.hosts
primary         tnt.acme.com            tnt.hosts
primary         boom.acme.com           boom.hosts
primary         sv.acme.com             sv.hosts
primary         223.128.IN-ADDR.ARPA    tnt.rev
primary         223.128.IN-ADDR.ARPA    boom.rev
primary         223.128.IN-ADDR.ARPA    sv.rev
primary         0.0.127.IN-ADDR.ARPA    localhost.rev
```

Here, a sort list has been set up to return an address with a network portion of 128.223 before an address with a network portion of 10. The name server should look in **/etc/domain** for all data files. It looks in **root.cache** for information with which to prime or initially load its cache. The free standing dot implies that the root domain can be reached through the information in **root.cache**. The authoritative data for all the name servers and hosts for the **acme.com** zone is to be found in **acme.hosts** (since this is a primary entry).

Authoritative data for all the name servers and hosts for the **tnt.acme.com** zone is to be found in **tnt.hosts**, data for **boom.acme.com** is to be found in **boom.hosts**, and data for **sv.acme.com** is to be found in **sv.hosts**. Authoritative data for the **223.128.IN-ADDR.ARPA** domain is to be found in **boom.rev**, **tnt.rev**, and **sv.rev** (again, primary entries). The loopback information is to be found in the file **localhost.rev** (another primary entry).

The following example shows a sample boot file, also called **named.boot**, for a secondary master server.

```
; secondary server
;
sortlist      128.223.0.0  10.0.0.0
;
directory     /etc/domain
;
; type        domain                    source/host file   backup
file
;
cache         .                         root.cache
secondary     acme.com                  128.223.1.26
acme.hosts.bak
secondary     tnt.acme.com              128.223.1.26
tnt.hosts.bak
secondary     boom.acme.com             128.223.1.26
boom.hosts.bak
secondary     sv.acme.com               128.223.1.26
sv.hosts.bak
secondary     223.128.IN-ADDR.ARPA      128.223.1.26
tnt.rev.bak
secondary     223.128.IN-ADDR.ARPA      128.223.1.26
boom.rev.bak
secondary     223.128.IN-ADDR.ARPA      128.223.1.26
sv.rev.bak
primary       0.0.127.IN-ADDR.ARPA      localhost.rev
```

Here, source hosts are specified instead of source files. This is because the secondary master server that boots from this file obtains authoritative information through a zone transfer from the primary name server with the Internet address 128.223.1.26, rather than from data files. The files with **.bak** as an extension contain backup data for the zone. If you were to restart the secondary name server, it would load zone data from these files. The secondary name server would then consult the specified primary server (here 128.223.1.26) to check whether the data were still up to date.

Editing a Name Server's Cache File

As mentioned above, **root.cache** contains names and addresses of the authoritative name servers for the root domain of the network. If you are setting up a master name server, you need only copy the prototype file to the directory you specify in the boot file. If you are setting up a slave server, this file must contain at least the names and addresses of local name servers.

The cache file becomes the initial data in the name server's cache.

The prototype **root.cache** file that you can use to initialize the caches shown below. The time-to-live values (ttl) are set very high

(99999999) to ensure that information about the authoritative name servers does not expire. The time-to-live values specify, in seconds, how long the data is kept in the domain's data base.

```
;
; Initial cache data for root domain servers.
;
;name           ttl             class     type      domain_name

               99999999        IN        NS      NS.NIC.DDN.MIL.
               99999999        IN        NS        A.ISI.EDU.
               99999999        IN        NS        AOS.BRL.MIL.
               99999999        IN        NS        C.NYSER.NET.
               99999999        IN        NS  GUNTER-ADAM.AF.MIL.
               99999999        IN        NS        NS.NASA.GOV.
               99999999        IN        NS        TERP.UMD.EDU.
;
;  Prep the cache. Order does not matter
;
;name           ttl             class     type          address

NS.NIC.DDN.MIL.    99999999        IN        A     192.67.67.53
A.ISI.EDU.         99999999        IN        A      26.3.0.103
A.ISI.EDU.         99999999        IN        A     128.9.0.107
AOS.BRL.MIL.       99999999        IN        A     128.20.1.2
AOS.BRL.MIL.       99999999        IN        A     192.5.25.82
C.NYSER.NET.       99999999        IN        A     192.33.4.12
GUNTER-ADAM.AF.MIL.99999999        IN        A      26.1.0.13
NS.NASA.GOV.       99999999        IN        A     128.102.16.10
NS.NASA.GOV.       99999999        IN        A     192.52.195.10
TERP.UMD.EDU.      99999999        IN        A     128.8.10.90
```

Editing a Name Server's Data Files

The standard files to specify authoritative data for a zone are the *named.hosts* file and the *named.rev* file. The standard file to specify the loopback interface to the name server is the *localhost.rev* file. As the example in the previous section showed, you can call these files whatever you want. In that example, the files were named **tnt.hosts** and **tnt.rev** for the **tnt.acme.com** domain, they were named **boom.hosts** and **boom.rev** for the **boom.acme.com** domain, and so on. The single local host file was called **localhost.rev**.

You specify the location of the data files using the **directory** parameter in the boot file. As the sample **named.boot** files showed, you should set up *named.host* and *named.rev* files for each zone, and a *localhost.rev* file for each name server.

The *named.hosts* file contains all the data about the name servers and hosts in a zone. It contains resource records that specify the operating parameters of name servers.

The *named.rev* file allows address-to-hostname mapping. It specifies the local **IN-ADDRARPA** domain, which was created especially to allow this mapping. Leaf-node names in the **IN-ADDR.ARPA** domain have a one-to-one correspondence with Internet addresses. To map an Internet address to its **IN-ADDR.ARPA** domain equivalent, use the decimal number for each octet of the Internet address from right to left as labels, and add the **IN-ADDR.ARPA** suffix.

For example, the Internet address **128.223.8.99** would be represented in the *named.rev* file as **99.8.223.128.IN-ADDR.ARPA**. This reversal of the address allows for the natural grouping of hosts in a network. This way, the **IN-ADDR.ARPA** domain is organized into subdomains that correspond to the division of Internet addresses into networks. For example, **223.128.IN-ADDR.ARPA** corresponds to network 128.223 (class B). If you query the name that corresponds to a network number in the **IN-ADDR.ARPA** domain, you get a list of pointers (**PTR** records) that point to gateways for that network. **PTR** records are discussed later in the chapter.

The *localhost.rev* file specifies the address for the local loopback interface, which has the network address **127.0.0.1**.

Creating Resource Records in the Data Files

Resource records have a standard syntax and a standard format. The standard syntax is as follows:

[*blank*] [*;comment*]

$ORIGIN *domain_name* [*;comment*]

$INCLUDE *filename* [*domain_name*] [*;comment*]

domain_name record [*;comment*]

[*blank*]*record* [*;comment*]

Blank lines, with or without comments, are allowed anywhere in the file. The syntax of *domain_name* was covered earlier in the chapter.

Two special resource records are optional control entries: the **$ORIGIN** and **$INCLUDE** entries. Use **$ORIGIN** to change the origin for relative domain names. Start the entry in column 1, and

follow it with a domain origin. Use the entry to specify more then one domain in a single data file. For example, the following entry changes the origin to **a.isi.edu.**

$ORIGIN a.isi.edu.

$INCLUDE inserts the specified file into the current file, and can optionally specify a domain name that sets the relative domain name origin for the included file. An include entry starts in column 1. You can have more than one include entry in a master file. This control entry is particularly useful to separate different types of data into multiple files.

The **$INCLUDE** control entry does not cause data to be loaded into a different zone or tree. The entry simply lets data for a given zone be organized in separate files. For example, you may use the **$INCLUDE** entry to separate host information from information about zone authority. Suppose *named.hosts* contained the following **$INCLUDE** entry:

$INCLUDE tnt.hosts

The name server would interpret the entry as a request to load the file **tnt.hosts** from the directory specified in the boot file. (Alternatively, you can specify a full pathname.) You may use such a file to keep a frequently changing lists of hosts separate from stable information such as that about zone authority and the name of the name server.

The last entries in the resource record include *record*, which has the following standard format:

[*name*]] [*ttl*] [*class*] *recordtype* *record-specific-data*

The first field is the name of the record. If you include it, you must always start it in column one. This field is optional in most records; it is required in the start of authority (**SOA**) record. If you omit the *name* field, the record takes on the name of the previous record.

The second field is an optional time-to-live (*ttl*) field. If you omit this field, you must specify the minimum time-to-live in the start of authority resource record (see "Specifying the Start of Authority Resource Record (SOA)").

The third field is *class*. There are three values for *class*: **IN** for Internet, **CH** for Chaos, and **HS** for Hessiod. Always use the **IN** class; the other values are inappropriate for TCP/IP for AViiON Systems applications. If you omit the *class* field, the record takes on the class of the previous resource record.

The fourth field specifies the *recordtype*. You must specify a *recordtype* for every resource record. Table 5–4 shows the most commonly used record types.

Each resource record type (and others) are described in the following sections.

The number and type of *record-specific-data* fields depend on the type of the resource record you specify.

Case is preserved in names and data fields when loaded into the name server. However, as you have learned, all comparisons and lookups in the name server data base are case insensitive.

Table 5–5 shows the characters that have special meanings in resource records.

Table 5–4 Commonly Used Resource Record Types

Record Type	Description
SOA	Start of authority; designates the beginning of a zone.
NS	Name server; ists a name server responsible for a given zone.
A	Address; lists the Internet address for a given machine.
HINFO	Host information; describes host specific data.
CNAME	Canonical name; specifies an alias for a fully-qualified name.
WKS	Well known services; describes the advertised services supported by a protocol at a specified address.
MX	Mail exchange; specifies a machine that can deliver mail to a machine not directly connected to the network.
PTR	Pointer; lets special names point to a location in the domain.
MB	Mailbox; lists the machines where a user wants to receive mail.
MR	Mail rename; lists aliases for a user
MINFO	Mailbox information; creates a mail group for a mailing list.
MG	Mail group; lists members of a mail group

Table 5–5 Characters with Special Meaning in Resource Records

Character	Meaning
.	A free-standing dot in the name field refers to the current domain.
@	A free-standing @ in the name field denotes the current origin.
. .	Two free-standing dots represent the null domain name of the root when used in the name field.
\DDD	Each **D** is a decimal digit. Refers to an octet that corresponds to **DDD**. The name server assumes that the octet is text, and it does not check it for special meaning. This lets you place a number where the name server expects a string.
( )	Group data that crosses a line. In effect, line terminations are not recognized within parentheses.
:	Starts a comment; the remainder of the line is ignored.
*	Wildcards.
"	Let you put spaces in a field value.

If you do not terminate a name with a dot (.), the name server appends the current origin to the name. This may be useful when you want to append the current domain name to the data, but it may cause problems other times. If the hostname is not in the domain for which you are creating the data file, specify the fully qualified name.

The standard resource record format is explained in detail in *RFC 1033 (Domain Administrators Operations Guide)*, *RFC 1034*, and *RFC 1035*.

Specifying the Start Of Authority Record (SOA)

The start of authority or **SOA** record designates the beginning of a zone. It contains important information about the name server that serves as the authority for the zone. Specify only one **SOA** per zone.

```
[name]  [ttl]   [class] recordtype  origin              person_in_charge

@               IN      SOA         sys01.tnt.acme.com. mt@sys01.acme.com. (
                                    890604              ; Serial
                                    10800               ; Refresh every 3 hours
                                    3600                ; Retry 1 hour
                                    3600000             ; Expire 1000 hours
                                    86400 )             ; Minimum 24 hours
```

Here, the special character @ appears in the *name* field, denoting the current origin. No *ttl* is specified in this record, so the record acquires the time-to-live value specified in the Minimum parameter,

which is explained in a moment. In this example and in all the examples that follow, the *class* is **IN**, for Internet records. The *origin* is the name of the host on which this data file resides (here, sys01). The *person_in_charge* is the mailing address for the person responsible for the name server.

The fields after *person_in_charge* contain information about the name server that serves as the authority for the zone. You must specify a value for each of these fields.

The first field is the Serial number, which is the version number of this data file. This number should be incremented whenever a change is made to the data. If you use a serial number that contains a decimal point, you cannot use an ordinate greater than 9999.

The second field is the Refresh parameter, which indicates how often, in seconds, a secondary name server should check with the primary name server to see if an update is needed.

The third field is the Retry parameter, which indicates how long, in seconds, a secondary server should wait before it checks for a refresh after some kind of failure.

The fourth field, Expire, is the upper limit, in seconds, that a secondary name server is to use the data before it expires for lack of getting a refresh.

The fifth field, Minimum, is the minimum number of seconds to be used for the time-to-live field on records; if a record has no explicitly specified time-to-live, it has at least this long to live.

Specifying a Name Server Record (NS)

A name server or **NS** record lists a name server responsible for a given zone.

```
[name]  [ttl]   [class] recordtype  name_server

                IN      NS          sys01.tnt.acme.com.
```

The *name* field specifies the zone that the listed name server services. If you do not specify a *name*, the *name_server* services the last explicitly named zone, whether in a previous **SOA** record or in a previous **NS** record.

Specify one **NS** record for each authoritative server for the domain. If the specified *name_server* does not reside in the zone, you must include an address (**A**) record for that name server. Such an **A** record is called a *glue record*.

Specifying an Address Record

An **A** record lists the address for a given machine.

```
[name]  [ttl]   [class] recordtype  address

sys01           IN      A          128.223.1.26
```

Use the *name* field to specify the hostname and the *address* field to specify the network address.

Specify one **A** record for each address of each host that you want to serve.

Specifying a Host Information Record

A **HINFO** record describes host specific data.

```
[name]  [ttl]   [class] recordtype  Hardware OS

                IN      HINFO      AV5100 DG/UX
```

This line lists the hardware (AV5100) and operating system (DG/UX) that are running at the host specified in the preceding **A** record.

Only a single space can separate the hardware information from the operating system information. To include a space in the hardware model, you must quote the name, for example, **"AV 5100"**.

We recommend that you specify one **HINFO** record for each host in a zone.

Specifying a Canonical Name Record

A **CNAME** record specifies an alias for a canonical name. A canonical name is a fully qualified name.

```
alias   [ttl]   [class] recordtype  Canonical name

sys33           IN      CNAME      sys01.tnt.acme.com.
```

The **CNAME** record is the only record associated with the alias name; all other resource records should be associated with the canonical name and not with the alias. Any resource records that include a domain name as their value (for example **NS** or **MX**) should list the canonical name, not the alias.

Specifying a Well Known Services Record

A well known services (**WKS**) record describes the advertised services supported by a particular protocol at a specified address.

```
[name]  [ttl]    [class]  recordtype  address      protocol   list of services

                 IN       WKS         128.223.1.   UDP        (who route timed
                                      2                       domain)
                 IN       WKS                      TDP        (echo telnet
                                      128.223.1.              netstat ftp
                                      2                       smtp hostnames
                                                              domain nameserver)
```

The list of services should correspond to the list of services specified in **/etc/services**.

The **WKS** record is optional. If you use it, specify only one **WKS** record per protocol per address record.

Specifying a Mail Exchange Record

A mail exchange (**MX**) record specifies a machine that can deliver mail to a machine that is not directly connected to the network.

[name]	*[ttl]*	*[class]*	*recordtype*	*preference value*	*mailer exchange*
Munnari.OZ.AU.		IN	MX	0	Seismo.CSS.GOV.
*.IL.		IN	MX	0	RELAY.CS.NET.

In the first line, Seismo.CSS.GOV. is a mail gateway that knows how to deliver mail to Munnari.OZ.AU., but other machines on the network cannot deliver mail directly to Munnari. These two machines may have a private connection or use a transport medium different from other hosts. The preference value is the order that a mailer should follow when there is more than one way to deliver mail to a single machine. See *RFC 974 (Mail Routing and the Domain System)* for more detailed information.

You can use wildcard names in **MX** records. There are likely to be servers on the network that simply state that any mail to a domain should be routed through a relay. In the second line above, all mail to hosts in the domain **IL** is routed through **RELAY.CS.NET**.

Specifying a Domain Name Pointer Record

A domain name pointer (**PTR**) record lets special names point to some other location in the domain. Here is an example from the *named.rev* file for the **IN-ADDR.ARPA** domain for network 128.223.

[name]	*[ttl]*	*[class]*	*recordtype*	*real_name*
26.1		IN	PTR	sys01.tnt.acme.com.

Here, the host part of an Internet address appears in the *name* field; the network part of the address is implied from the zone for which the data file applies. You specify the network part, for example `223.128.IN-ADDR.ARPA`, in the boot file for the zone. Thus, the network part of the address is implied for the **IN-ADDR.ARPA** domain.

Suppose that after several entries such as these, you wanted to change the implied network part of addresses. Subsequent lines in *named.rev* file would look like this:

```
$ORIGIN     223.128.IN-ADDR.ARPA.
50.1        PTR     sys50.boom.acme.com.
52.1        PTR     sys52.boom.acme.com.
```

or equivalently:

```
50.1.223.128.IN-ADDR.ARPA.    PTR    sys50.boom.acme.com.
52.1.223.128.IN-ADDR.ARPA.    PTR    sys52.boom.acme.com.
```

In either case, these resource records state that address `128.223.1.50` belongs to a host called `sys50.boom.acme.com.`, and `128.223.1.52` belongs to `sys52.boom.acme.com.` (The **$ORIGIN** entry was explained earlier in the chapter.)

The system administrator for a given network usually has authority over the **IN-ADDR.ARPA** subdomain which has all of the addresses for the networks. Thus, she or he can edit the master file to insert the name to address mapping. However, since gateways must have addresses on other networks, the **PTR** resource records for a gateway typically have to be divided and separately managed by more than one administrator.

You can also use the **PTR** record to associate a generic pointer from a domain name to the owner name. The following line shows how to set up reverse pointers for the special **IN-ADDR.ARPA** domain.

```
[name]                          [ttl]    [class]  recordtype     real_name
32.0.9.128.IN-ADDR.ARPA.                 IN       PTR            Venera.ISI.EDU.
```

Here, `32.0.9.128.IN-ADDR.ARPA` points to `Venera.ISI.EDU`. This means that information about 32.0.9.128.IN-ADDR.ARPA can be found through Venera.ISI.EDU. **PTR** names should be unique to the zone.

In addition to pointing to some other location in a domain or associating a pointer from a name to the owner name, you can use a **PTR** record to do the following:

- Map an IP address or network number to a network name.
- Map a network name to a network address.
- Map an organization to its network names and numbers.
- Map an organization to subnets, even when the subnets are nested.

To accomplish this, you create names in the **IN-ADDR.ARPA** tree that correspond to addresses where the host numbers are zero. You set up a **PTR** record that points to a name in its data section. This name could refer to a network or a subnet. If the name refers to a subnet, you set up an **A** record that points to the network mask.

For example, in the records that follow, cmu-net.cmu.EDU refers to a subnet of the Class B network 128.2.

```
0.0.2.128.IN-ADDR.ARPA.                    PTR     cmu-net.cmu.EDU.
                                           A       255.255.255.0
```

Consider a more involved example concerning the Class B network named RDU that is in the **com.** zone and that has the network number 128.226. Suppose that this network is organized into two levels of subnet, the first level using an additional 8 bits of address, and the second level using 4 bits, yielding network (subnet) masks of **0xffffff00** and **0xfffffff0**. Assume that the fully qualified name for network RDU is xyz-net.RDU.com., the first-level subnets are named div1-subnet.RDU.com. and div2-subnet.RDU.com., and a second-level subnet is called inc-subsubnet.RDU.com.

To encode this information for use by the DNS, you would create the following records in RDU's hosts file for the RDU.com. zone:

```
; Define network entry
   xyz-net.RDU.com.              PTR
0.0.226.128.IN-ADDR.ARPA.

; Define first level subnets
   div1-subnet.RDU.com.          PTR
0.1.226.128.IN-ADDR.ARPA.
   div2-subnet.RDU.com.          PTR
0.2.226.128.IN-ADDR.ARPA.

; Define second level subnets
   inc-subsubnet.RDU.com.        PTR
16.2.226.128.IN-ADDR.ARPA.
```

You would also create the following records in RDU's reverse file for the **226.128.IN-ADDR.ARPA** zone:

```
; Define network number and address mask
   0.0.226.128.IN-ADDR.ARPA.     PTR  xyz-net.RDU.com.
                                 A    255.255.255.0  ;
0xffffff00

; Define one of the first level subnet numbers and masks
   0.1.226.128.IN-ADDR.ARPA.     PTR  div1-subnet.RDU.com.
                                 A    255.255.255.240 ;
0xffffff0

; Define another first level subnet number and mask
   0.2.226.128.IN-ADDR.ARPA.     PTR  div2-subnet.RDU.com.
                                 A    255.255.255.240 ;
0xffffff0

; Define second level subnet number
   16.2.226.128.IN-ADDR.ARPA.    PTR
inc-subsubnet.RDU.com.
```

For more information about DNS encoding of network names and other entities, see *RFC 1101 (DNS Encoding of Network Names and Other Types)*.

Specifying a Mailbox Record

An **MB** record lists the machines where a user wants to receive mail.

[name]	*[ttl]*	*[class]*	*recordtype*	*machine*
bucky		IN	MB	spectre.tnt.acme.com.

The *name* field is the user's login; the *machine* field is the machine to which mail should be delivered. Names must be unique to the zone.

This record type is experimental and should be used with discretion.

Specifying a Mail Rename Record

A mail rename **(MR)** record lists aliases for a user.

[name]	*[ttl]*	*[class]*	*recordtype*	*corresponding-MB*
Postmaster		IN	MR	bucky

The *name* field lists the alias for the name listed in the fourth field, which should have a *corresponding-MB* record.

This record type is experimental and should be used with discretion.

Specifying a Mailbox Information Record

A mailbox information (**MINFO**) record creates a mail group for a mailing list.

```
[name]   [ttl]   [class]  recordtype   requests              maintainer
BIND             IN       MINFO        BIND-REQUEST          mt@sys01.tnt.acme.com.
```

This record is usually associated with an **MG** record but may be used with an **MB** record. The *name* specifies the name of the mailbox. The *requests* field is where mail such as requests to be added to a mail group should be sent. The *maintainer* is a mailbox that should receive error messages. Use the **MINFO** record to specify mailing lists when errors in member's names should be reported to a person other than the sender.

Specifying a Mail Group Member Record

A mail group (**MG**) record lists members of a mail group.

```
[mail group name]     [ttl]     [class]  recordtype      member name
                                IN       MG              bind
```

Here is a sample mailing list:

```
Bind        IN    MINFO    Bind-Request                mt@sys01.tnt.acme.com.
            IN    MG       bucky
            IN    MG       lj@manatee.tnt.acme.com.
            IN    MG       cdg@harpo.tnt.acme.com.
            IN    MG       jgj@illusion.tnt.acme.com.
```

This record type is experimental and should be used with discretion.

A Sample named.hosts File

This example shows a sample hosts file for a primary server called **hosts.tnt**. This hosts file services the **tnt.acme.com.** zone. Remember, you specify the location of the hosts file with the **directory** parameter in the boot file, **named.boot**.

```
; name   ttl   class   record      origin              Person
;
@              IN       SOA        sys01.tnt.acme.com.
mt@sys01.acme.com.(
                                                       890604
; Serial
                                                       10800
; Refresh 3 hours
                                                       3600
; Retry 1 hour
                                                       3600000
; Expire 1000 hours
                                                       86400 )
; Minimum 24 hours
               IN       NS         sys01.tnt.acme.com.
               IN       NS         sys30.tnt.acme.com.
               IN       MX         10
sys01.tnt.acme.com.
sys01          IN       A          128.223.1.26
sys30          IN       A          128.223.8.30
                        MX         10
sys30.tnt.acme.com.
               IN       HINFO      AV5100 DG/UX
sys22          IN       A          128.223.1.2
                        MX         10
sys01.tnt.acme.com.
sys33          IN       CNAME      sys01.tnt.acme.com.
               IN       HINFO      MV7800 DG/UX
bucky          IN       A          128.223.1.3
                        MX         10
bucky.tnt.acme.com.
               IN       HINFO      AV5100 DG/UX
sun2           IN       A          128.223.1.5
                        MX         10         sun2.tnt.acme.com.
               IN       HINFO     "SUN 3/260" "SUNOS"
```

First, consider the start-of-authority or **SOA** record. This record indicates that authoritative data for current domain (implied by the @ character) can be found at sys01.tnt.acme.com. The primary name server for the domain is sys01, and the mail address for the administrator responsible for the domain is mt@sys01.tnt.acme.com.

The record also specifies a serial number of 890604. This value corresponds to the date when the file was last updated. Using the date is a convention, not a requirement. The values you choose for your serial number are left to your discretion. If you choose to use the date as a serial number, it limits you to updating the secondary copies of the database no more than once a day.

Other parameters have to do with when the name server should perform some action. The value for the refresh parameter, 10800, indicates that a zone refresh should take place every 3 hours (10800 seconds). The value for the retry parameter, 3600, indicates that secondary servers should check whether the zone data has been updated every hour (3600 seconds). The retry value applies once the value for refresh has elapsed. The value for the expire parameter, 3600000, indicates that zone date expires after 1000 hours. The value for the minimum parameter, 86400, indicates that all data must have a time-to-live of at least 24 hours (86400 seconds).

After the **SOA** record come two name server records, one for sys01.tnt.acme.com., and one for sys30.tnt.acme.com. Specify at least two name servers, on different physical networks, for each domain in case of system crashes or network problems. Each **NS** record has a corresponding **A** (address) record that specifies the name server's Internet address.

Each name server has host information (**HINFO**) and mail exchange (**MX**) records. The **HINFO** records show that two name server systems are Data General AViiON®5100 series systems. One is a Data General ECLIPSE MV/7800™, and one is a SUN 3/260 running SUN/OS. The **MX** records show that sys30.tnt.acme.com. uses itself as a mail exchange; sys33.tnt.acme.com., which has the CNAME sys01.tnt.acme.com., uses itself as a mail exchange; and that bucky.tnt.acme.com., and sun2.tnt.acme.com. use themselves as mail exchanges.

A Sample localhost.rev File

This example shows a sample local host file called **/etc/localhost.rev**. Remember, this file specifies the domain authority for the local loopback interface. Each name server should have its own **localhost.rev** file.

```
;
; name    ttl      class    record      origin
Person
;
@                  IN       SOA         sys01.tnt.acme.com.
mt@sys01.acme.com. (
                                                    890604
; Serial
                                                    10800
; Refresh 3 hours
                                                    3600
; Retry 1 hour
                                                    3600000
; Expire 1000 hours
                                                    86400 )
; Minimum 24 hours
                   IN         NS          localhost
localhost          IN         A           127.0.0.1
1                  IN         PTR         localhost
```

The pointer record (**PTR**) points to the host 1. Remember that the boot file specifies a domain of 0.0.127.IN-ADDR.ARPA for **localhost.rev**. That is, all hosts specified in **localhost.rev** have as their default domain 0.0.127.IN-ADDR.ARPA. Thus, the local host becomes 1.0.0.127.IN-ADDR.ARPA, which resolves to the address of the loopback interface, **127.0.0.1**.

A Sample named.rev File

This example shows a sample *named.rev* file called **tnt.rev**. Remember, the *named.rev* file specifies the local **IN-ADDR.ARPA** domain. This domain was created especially to allow address-to-hostname mapping. The boot file for this name server specified that authoritative data for the 223.128.IN-ADDR.ARPA domain is to be found in **boom.rev**, **tnt.rev**, and **sv.rev** (they were **primary** entries).

```
;    tnt.rev
;
; name  ttl   class     record     origin
Person
;
@             IN        SOA        sys01.tnt.acme.com.
mt@sys01.acme.com. (
                                                 890604
; Serial
                                                 10800
; Refresh 3 hours
                                                 3600
; Retry 1 hour
                                                 3600000
; Expire 1000 hours
                                                 86400 )
; Minimum 24 hours
              IN        NS         sys01.tnt.acme.com.
              IN        NS         sys30.tnt.acme.com.
26.1          IN        PTR        sys01.tnt.acme.com.
3.1           IN        PTR        bucky.tnt.acme.com.
5.1           IN        PTR        sun2.tnt.acme.com.
30.8          IN        PTR        sys30.tnt.acme.com.
```

The **PTR** records point to the hosts sys01, bucky, sun2, and sys30 in the **IN-ADDR** domain. The host portion of these host's Internet addresses are reversed in the *name* field; the network portion of these addresses are specified in **named.boot** on the line for **tnt.rev**.

Essential Files for Different Types of Name Servers

Table 5–6 shows the files that you must set up for each type of name server, and what the file should contain. These are minimum requirements.

Table 5–6 Essential Files for Each Type of Name Server

Server Type	Essential Files
Master	Primary and secondary servers: *localhost.rev* containing **SOA** record for local domain, **NS** record for the loopback interface, **A** record for the loopback interface, **PTR** record for the loopback interface **root.cache** containing records of root and other master name servers **/etc/named.boot** containing location of other master files Primary servers only: *named.hosts* containing **SOA** record for local domain, **NS** records for master servers, **A** records for master servers and hosts within zone, **HINFO** records for hosts within zone *named.rev* containing **SOA** record for local domain, **NS** records for master servers, **PTR** records for hosts within zone
Slave	**/etc/named.boot** containing location of *localhost.rev* and **root.cache**, list of forwarder servers, line specifying **slave** *localhost.rev* containing **SOA** record for local domain, **NS** record for the loopback interface, **A** record for the loopback interface, **PTR** record for the loopback interface **root.cache** containing records of forwarder name servers

Continued

Table 5–6 Essential Files for Each Type of Name Server

Server Type	Essential Files
Forwarder	Files required for master server
Caching-only	**/etc/named.boot** containing location of *localhost.rev* and **root.cache**
	localhost.rev containing **SOA** record for local domain, **NS** record for the loopback interface, **A** record for the loopback interface, **PTR** record for the loopback interface
	root.cache containing records of root and master name servers

Using the Domain Name System

The following sections describe the user interfaces to the domain name system. One section describes how to use the **named** command to start the name server and another tells how to use the **nslookup** program to access name servers. Other sections describe how to integrate the domain name system with other means of name/address resolution, how to debug a name server, and how to register your domain with the proper authorities.

One way to gather information about operating the domain name system is to become part of the BIND mail group (**bind@ucbarpa.Berkeley.EDU**) or of the NAMEDROPPERS group (**namedroppers@nic.ddn.mil**). To become part of the BIND mail group, send a request to **bind-request@ucbarpa.Berkeley.EDU**. To become part of the NAMEDROPPERS group, send a request to **namedroppers-request@nic.ddn.mil**. These groups exchange messages about bugs detected and solutions discovered and discuss future design decisions, operational problems, and other related topics.

Starting the Name Server

After you set up DNS, start **named** with the **-d** option to begin the debugging process. When you finish debugging, start **named** with **sysadm** so that **named** starts when the the system boots. (See "Manage DNS" in Chapter 3.)

The **named** command has the following syntax:

named [**–d** [*debuglevel*]] [**–p** *portnumber*] [[**–b**] *bootfile*]

Without any arguments, **named** reads the default boot file **/etc/named.boot**, reads any initial data, and listens for queries.

Options are as follows:

–d [*debuglevel*]	Write debugging information. The *debuglevel*, which is a value from 1 to 10, determines the level of messages printed. The default is 1, which supplies the minimum information. The name server writes debugging information to **/var/adm/named.log.**
–p *portnumber*	Tells **named** to listen at *portnumber.* The default is 53.
[**–b**] *bootfile*	Use the specified boot file. The boot file contains information about where the name server is to get its initial data. The default is **/etc/named.boot**.

Any additional argument is taken as the name of the boot file.

Debugging the Name Server

When you run **named** with the **-d** option, the name server transcribes a record of its activity to a file named **/var/adm/named.log**. Specify a number after the **-d** to determine the amount and detail level of the information transcribed to the file. The default debugging level is 1 (minimum information), and the highest debugging level is 10 (the maximum).

When you first bring up the name server, you can check the file to determine whether the name server read its data files correctly. The following example shows what **named** puts into the debugging file with debug level 1.

```
Debug turned ON, Level 1
Version = named 4.8 #52: Tue Jun 4 14:27:57 EDT 1991
.
.
.
zone[0] type 3: '.', source = root.cache
db_load(root.cache, , 0)
z_time 0, z_refresh 0
zone[1] type 1: 'acme.COM', source = named.hosts
db_load(named.hosts, acme.COM, 1)
z_time 0, z_refresh 0
zone[2] type 1: '223.128.IN-ADDR.ARPA', source = named.rev
db_load(named.rev, 223.128.IN-ADDR.ARPA, 2)
z_time 0, z_refresh 0
zone[3] type 1: '0.0.127.IN-ADDR.ARPA', source =
localhost.rev
db_load(localhost.rev, 0.0.127.IN-ADDR.ARPA, 3)
```

If there is an error reading any of the data files, you see an error message somewhere in **named.log**.

The following section of **named.log** contains a record of how the name server handles a query from the loopback interface (127.0.0.1).

```
Ready to answer queries.
prime_cache: priming = 0
sysquery: send -> 26.0.0.73 7 (53), nsid=1 id=0 0ms
resend(addr=1 n=0) -> 128.102.16.10 7 (53) nsid=1 id=0 0ms

datagram from 127.0.0.1 port 1089, fd 7, len 30
req: nlookup(big-k.acme.com) id 1 type=1
req: found 'big-k.acme.com' as 'big-k.acme.com' (cname=0)
req: nlookup(sys46.acme.COM) id 1 type=1
req: found 'sys46.acme.COM' as 'sys46.acme.COM' (cname=1)
req: answer -> 127.0.0.1 8 (1089) id=1 Local
```

By scanning later sections of **named.log**, you can tell what routines the name server used to do the work requested, for example, "answer" to send an answer to the querying program. A line that begins with "datagram from" indicates that the name server is handling a specific query. You can study the lines that follow this one to see how the name server handled the query and whether it updated its database. Also, you can use the **grep** command to find a particular hostname to see if there is any name server activity associated with the host.

If there is an error handling a query, you see an error message among lines like these.

Occasionally, the name server sends queries to root servers to determine whether it needs to update resource records in its cache. You can check **named.log** to see whether **named** obtained the correct information or had a problem obtaining the information.

View the contents of **/var/adm/named.log** whenever **named** fails. The **named** process totally fails when the name lookup function hangs until it times out, or when the name lookup fails immediately on all foreign lookups. When this happens, check to see whether **named** is running at all. If it is not running, restart it.

When some zones can't be read or when incorrect data is returned, it means one of three things:

- Servers for some zones (or the root zone) cannot be reached.
- Servers for other zones are confused and have returned incorrect server or host information.
- The local database or cache has been corrupted. If this is the case, kill **named** and restart it.

To determine which of these three factors caused the error, use the **nslookup** program in interactive mode and do the following:

- Set **debug** or **d2** and **norecurse**.
- Query local and remote servers.
- Cause **named** to dump its cache (see **named**(1M) for details)
- Examine **/var/adm/named.log**, use **grep** to search for the hostnames of the name servers queried, and find out if and how the name server failed.

Configuring Name/Address Resolution

You can configure the order and the means of performing name/address resolution through the contents of the file **/etc/svcorder**. This file contains one record made up of a single field and, optionally, a comment.

This field determines the order in which the different address resolution methods are tried. The field consists of any combination of a subset of the following keys separated by colons (:), **YP** or **yp** for Network Information Service (which used to be called the Yellow Pages®), **RES** or **res** for the resolver, **EHOSTS** or **ehosts** for the **/etc/hosts** file. Optionally, you can specify a comment after the field. You separate the comment and field with white space, and begin a comment with a # character.

The following line specifies that address resolution should be attempted by NIS first.

yp:ehosts:res

If NIS is running and cannot resolve the name/address pair, name/address resolution goes no farther. You can, however, configure NIS so that it uses the domain name system when it cannot resolve a name/address pair. For details about how to do this, see *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System*.

If NIS is not running, **/etc/hosts** is consulted for name address resolution. If **/etc/hosts** does not produce an answer, the domain name system is consulted. If the **/etc/svcorder** file does not exist, different resolution methods are tried as above.

When you specify no resolution methods, the system checks for the presence of the NIS and uses that service. However, you should use **/etc/svcorder** to specify resolution methods, since it gives you the flexibility to configure name/address resolution any way you wish.

Using nslookup

Use **nslookup**(1M) to query Internet domain name servers directly. You can run **nslookup** in noninteractive or interactive mode.

In noninteractive mode, **nslookup** contacts the domain name resolver to resolve the address associated with a hostname that you typed on the command line. In interactive mode, **nslookup** services numerous types of requests and lets you specify the domain name server it should use to resolve a query. This can be helpful when you debug problems with a particular server.

When you use **nslookup** in noninteractive mode, its syntax is as follows:

nslookup *host_to_find* [*server_to_use*]

You use **nslookup** in noninteractive mode when you specify the name of the *host_to_find* as the first argument. Optionally, you can specify the name or address of a name *server_to_use* as a second argument.

When you use **nslookup** in interactive mode, its syntax is as follows:

nslookup [**-** [*server_to_use*]]

When you specify no arguments or a hyphen, **nslookup** accesses the default name server. When you specify a hyphen and the hostname or Internet address of a name server, **nslookup** accesses that name server.

nslookup displays the following information in interactive mode:

```
% nslookup ↵
Default Server: server_name
Address: server_addr

>
```

Enter a **nslookup**(1M) command at the > prompt. The command line must be less than 80 characters long. Any unrecognized command is interpreted by **nslookup** as a hostname. You can interrupt a command by typing Ctrl-C. To exit from interactive mode, type Ctrl-D. The **nslookup** commands are as follows:

host [*server*]

Looks up information for *host* using the current default server or using *server* if specified.

server *server*[.]
lserver *server*[.]

Changes the default server to *server*. Use a **.** to prevent the default domain from being appended to *server*, even when you have previously specified **set defname**. When you change the default server to *server*, **lserver** uses data from the initial server to look up information about the domain, while **server** uses the current default server to obtain the information. If an authoritative answer cannot be found, the names of servers that may have the answer are returned.

root

Changes the default server to the name server for the root of the domain name space. Currently, **nic.ddn.mil** is used. (This command is a synonym for **lserver nic.ddn.mil**.) You can change the name of the root server with **set root.**

ls [**-a** | **-h** | **-d**] *domain* [**>** *filename*]
ls [**-a** | **-h** | **-d**] *domain* [**>>** *filename*]

List the information available for *domain*. The default output contains host names and their Internet addresses.

Use the **–a** option to list aliases of hosts in the domain. Use the **–h** option to list CPU and operating system information for the domain. Use the **–d** option to list all contents of a zone transfer.

Use the > symbol to redirect output to a file, and the >> symbol to append output to a file. When output is redirected, hash marks are printed for every 50 records received from the name server. You must use a space to separate the redirect operator from the output file name.

view *filename*

Sorts and lists the output of the redirected **ls** command(s) with **more**(1).

help *or* **?**

Prints a brief summary of commands, which comes from **/usr/lib/help/nslookup/nslookup.help**.

set all

The **set** command changes state information that affects lookups. **set all** prints the current values of the available options. Information about the current default server and host is also printed.

set debug *or* **set nodebug**

Turns debugging mode on or off. (Default = **nodebug**, abbreviation = [**no**]**deb**.)

set db *or* **set nodb**

Turns exhaustive debugging mode on or off. Essentially, all fields of every packet are printed. (Default = **nod2**.)

set defname *or* **set nodefname**

Appends the default domain name to every lookup. (Default = **defname**, abbreviation = [**no**]**def**)

set search *or* **set nosearch**

With **defname**, searches for each name in parent domains of the current domain. (Default = **search**)

set domain=*name*

Changes the default domain name to *name*. The default domain name is appended to all lookup requests if the **defname** option has been set. The search list is set to the parents of the domain with at least two components in their names. (Default = *value* in hostname or **/etc/resolv.conf**, abbreviation =**do**)

set querytype=*value*
set type=*value*

Sets the type of information returned from a query to one of the following:
SOA – Information from the start of authority record.
A – The host's Internet address (the default).

CNAME – The canonical name for an alias.
HINFO – The host CPU and operating system type.
MB – The mail box.
MX – The mail exchange.
MG – The mail group member.
MINFO – The mailbox or mail list information.
MR – The mail rename domain name.
NS – Name server for the zone.
PTR – Pointer record, which is how you check reverse lookup.
Check *RFC 1035* for the other types of information that can be returned from a query. (Abbreviation = **q**)

set recurse *or* **set norecurse**

Tells the name server to query other servers if it does not have the information. (Default = **recurse**, abbreviation = [**no**]**rec**)

set retry=*number*

When a reply to a request is not received within a certain amount of time (set with **set timeout**), the request is resent. The retry number controls how many times a request is resent before giving up. (Default = 2, abbreviation = **ret**)

set root=*host*

Changes the name of the root server to *host*. This affects the **root** command. (Default = **nic.ddn.mil**, abbreviation = **ro**)

set timeout=*number*

Changes the time-out interval for waiting for a reply to *number* seconds. (Default = 10 seconds, abbreviation = **t**)

set vc *or* **set novc**

Always use a virtual circuit when sending requests to the name server. That is, always use a TCP rather than a UDP connection. (Default = **novc**, abbreviation = [**no**]**v**)

Setting Up Your Own Zone

Now that you understand how the domain name system works, you can to set up your own zone. First name your domains, and then contact the proper authority to register your zone.

Naming a Domain

Generally, the domain name system lets you design domains that have a single implicit semantic content. You may find it useful to

create a name space whose structure parallels the structure of your organization or business.

Contacting the Proper Authorities

Before you set up a zone to be on a public network, contact the organization in charge of the network to which you are connected and request the appropriate domain registration form. For top-level domains, this is usually the NIC (**hostmaster@nic.ddn.mil**). An organization that belongs to multiple networks (such as BITNET) should register with only one network. If you are on the BITNET and need to set up a zone, contact **INFO@BITNIC**. Organizations such as BITNET have special arrangements with the NIC to act as agents for their clients to the NIC.

End of Chapter

6 Configuring and Using the Simple Network Management Protocol

This chapter describes how to configure the Simple Network Management Protocol (SNMP) software and use the SNMP management commands. This information supplements section "Manage SNMP" in Chapter 3, which explains how to configure SNMP with **sysadm**.

This chapter describes the SNMP software and explains the shell command interface. Optionally available from Data General is a Motif–based SNMP interface. For this interface, see *Managing a Network with OS/EYE*NODE for AViiON Systems* (093–000812).

The SNMP software comprises three parts:

- The **snmpd** daemon, or agent (these terms are interchangeable), which runs on each SNMP host
- A hierarchical database of SNMP objects, called the Management Information Base (MIB)
- Several programs, called managers, run by network managers to query or set MIB values

SNMP is defined in *RFC 1157*. Figure 6–1 illustrates an SNMP network.

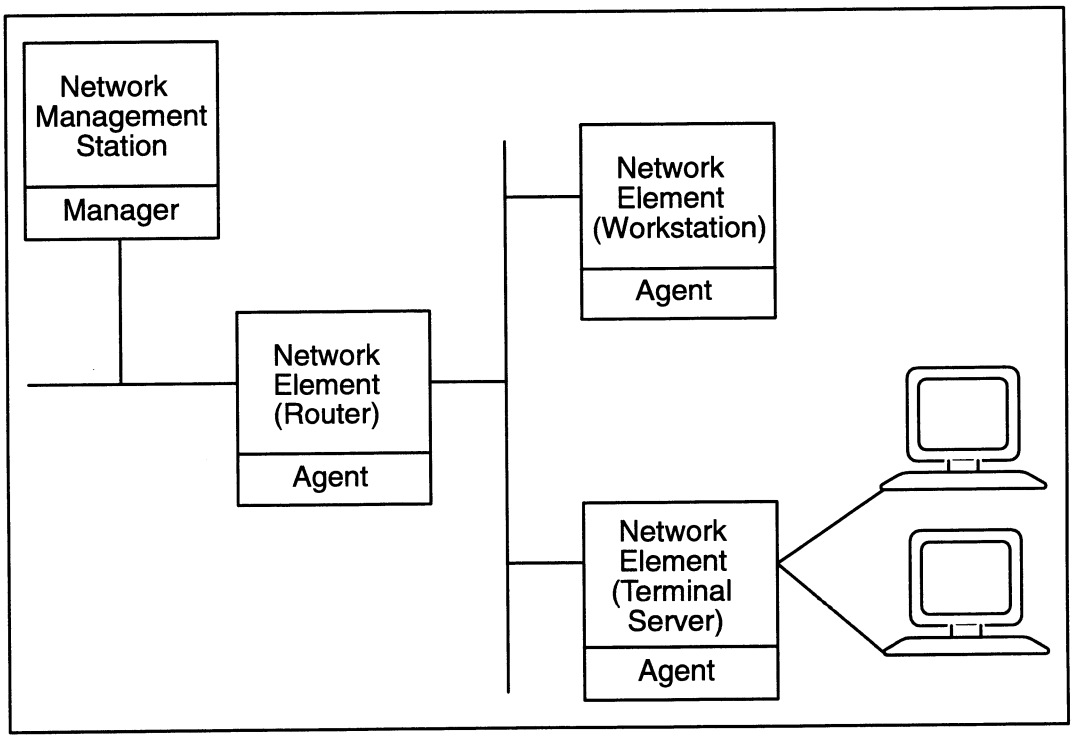


Figure 6–1 A Network Management Station and Network Elements

The network management station is simply the host from which an SNMP manager issues SNMP management commands. The network elements are different types of network host or device (servers, workstations, routers, terminal servers) managed through SNMP. Each element runs an SNMP agent, which answers queries or performs requested operations. The DG/UX SNMP agent (daemon) is **snmpd**.

The Management Information Base (MIB) Object Tree

The information that network management stations collect and maintain to monitor and control network elements is is called the *Management Information Base (MIB)*. The MIB is made up of *objects* organized in a treelike structure illustrated in Figure 6–2. The model for this tree and the general object types allowed for TCP/IP-based internets are defined in *RFC 1155*. MIB objects are defined in *RFC 1213*. Many use the term MIB in a general way to encompass any group of objects within the tree or even the entire tree.

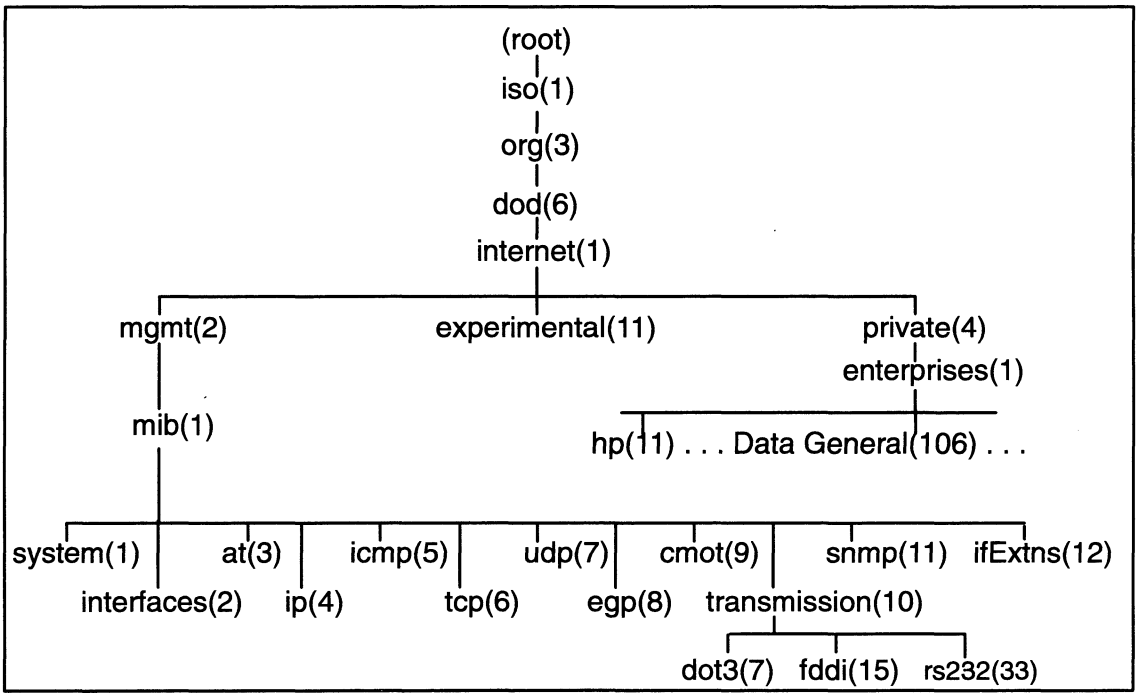


Figure 6–2 Major Branches of the Network Management Object Tree

Each object type has an *object ID*, which is a sequence of labels (integers) obtained as you traverse the tree from the root to the object, and an *object descriptor*, which is a string that names it. For example, the object with the object descriptor **interfaces** has the object ID **1.3.6.1.2.1.2**.

Object syntax and encoding are defined by an ISO specification called *Abstract Syntax Notation 1*, or ASN.1. For more information about ASN.1, see ISO 8824 and ISO 8825.

Figure 6–2 shows the standard internet object MIB subtree. MIBs for specific network media, such as FDDI *RFC 1285*, Ethernet (dot3) *RFC 1398*, and RS–232 *RFC 1317* are in the transmission subtree. *RFC 1229* is an extension of the standard MIB. Proprietary object definitions fall into the private-enterprise subtree, which appears at the right. The directory `/usr/etc/snmp` contains the files defining the MIBs supported by DG/UX. Appendix A illustrates the MIB subtree and describes the objects defined and maintained by Data General.

Object types are defined in the MIB. An object type may have one or more object instances. Instances of object types are maintained by **snmpd** and manipulated by the SNMP management commands. Object types are similar to the types provided by a programming language (such as **int** and **char**), while instances of object types are similar to specific variables of a given type (for example **myint** or **name**).

Object types are aggregate or nonaggregate. An aggregate type is a collection of object types, whereas a nonaggregate object type is a distinct entity. You can think of aggregate object types as tables, with the constituent object types forming columns of the tables and the object instances of those object types forming rows. Figure 6–3 illustrates the distinction between subtrees, aggregate object types, nonaggregate object types, and instances of aggregate object types.

In this figure:

- **ifNumber** is a nonaggregate object of the subtree interfaces, representing the number of network interfaces on the system.
- **ifTable** is an aggregate object of the subtree interfaces, where each row in the table represents a particular interface.
- **ifIndex**, **ifDescr**, and **ifPhysAddress** are nonaggregate objects of **ifTable**, where each object is a field in a table row.
- The numbered items are instances of their respective objects, containing interface indexes, descriptions, and addresses.

To refer to an instance of a nonaggregate object, you append a .0 to the object name. Thus, in Figure 6–3, instance **ifNumber.0** contains the number of interfaces on a given host. For each interface, the **ifTable** object contains separate **ifDescr** and **ifPhysAddress** objects (columns) and instances (rows) containing names, unit numbers, and physical addresses of the interfaces.

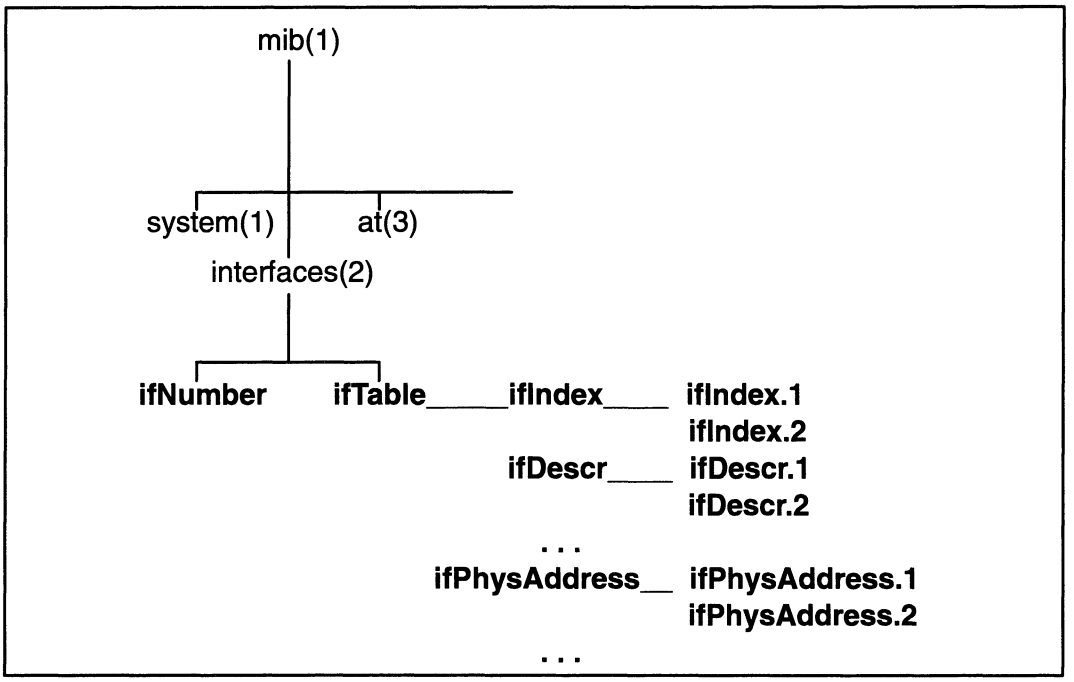


Figure 6–3 Subtrees, Object Types, and Instances

SNMP lets you query or qualify individual instances of an object, according to its type, from a management station. For example, the **system** branch contains a nonaggregate object **sysDescr** with object ID **1.3.6.1.2.1.1.1**. To query an agent for its system description, you specify an instance of the object type **sysDescr**. For nonaggregate objects, the instance is identified by appending a 0 to the object ID for the type. So for this example, the agent specifies object ID **1.3.6.1.2.1.1.1.0** (**sysDescr.0**) as the instance ID.

The object classes that compose the **mib** subtree, that is, **system**, **interfaces**, **at**, **ip**, and so on, are commonly called *groups*. These groups are defined in the MIB and represent the range of functions provided by network elements. With this set of groups, agents can limit support to object types that are applicable to the network element on which it runs. For example, an agent running on a gateway likely would support the **egp** (External Gateway Protocol) group, while an agent running on a workstation may not.

SNMP Messages

Agent and management software exchange information through SNMP messages. All SNMP messages are encoded in ASN.1. SNMP uses the User Datagram Protocol (UDP) to transmit messages. SNMP messages may contain more than one object, but if an operation fails for any object, the entire request fails. SNMP messages contain a request ID field that can be used to identify duplicated or lost packets.

The messages that agent and management software exchange are called Protocol Data Units (PDUs). SNMP supports five types of PDUs.

- **GetRequest-PDU.** Request the value for a specified object instance.
- **GetNextRequest-PDU.** Request the next instance, using lexicographical ordering, following the specified object. This PDU is useful to walk through aggregate objects or through the entire MIB.
- **SetRequest-PDU.** Request to set the value for an object instance.
- **GetResponse-PDU.** Response for any of the above requests.
- **Trap-PDU.** An asynchronous message from the agent to the management station for one of the following reasons:

coldStart	The agent is re–initializing itself because the configuration or the implementation may have been altered, or the agent has been restarted.
linkDown	The agent recognizes a failure in one of the communication links, or the administrator has taken down the interface with **ifconfig**(1M).
linkUp	The agent recognizes that one of the communication links has returned to operation, or the administrator has restarted the interface with **ifconfig**(1M).
authenticationFailure	The request failed authentication.

Configuring snmpd

The configuration file for **snmpd** is **/etc/snmpd.conf**. Use **sysadm** to modify this file (see "Manage SNMP" in Chapter 3) on each SNMP host. Alternatively, you can become superuser and edit **/etc/snmpd.conf** or you can use the **adm** scripts. If you edit the file rather than use **sysadm** or the **adm** scripts, you must stop and restart **snmpd** for your changes to take effect. For more information, see **snmpd.conf**(4), **admsnmpobject**(1M), **admsnmpcommunity**(1M), and **admsnmptrap**(1M).

The **/etc/snmpd.conf** file has the format:

record-type record–data

where *record-type* can be one of the following:

OBJECT	override default values of some objects
COMMUNITY	specify a community name
TRAP	specify a destination for TRAP messages
SUBAGENT	specify a path to an EMANATE subagent

The following sections explain how to define these four record types in **/etc/snmpd.conf** with appropriate *record–data*.

Defining Objects

Objects have predefined defaults. Enter object records to override the defaults. Object records have this format:

OBJECT *object value*

where *object* is one of the following variables and *value* is its assigned content:

sysDescr describes the agent

sysObjectID identifies the vendor

sysContact names the administrator and contact information such as an email address or phone number

sysLocation gives the host's location

sysName contains the host name: by default, the value returned by the **hostname**(1) command

snmpEnableAuthenTraps contains a 1 to enable or a 2 to disable traps

Except for **sysContact** and **sysLocation**, the default values are probably sufficient.

Example:

```
OBJECT sysDescr DG/UX TCP/IP SNMP AGENT
OBJECT sysObjectID DataGeneral
OBJECT sysContact James G. Jones jonesjg@acme.com
OBJECT sysLocation Dodd Hall, Second Floor
OBJECT sysName wyvern.tnt.acme.com
OBJECT snmpEnableAuthenTraps 2
```

Defining Communities

An agent needs a way to authenticate requests; otherwise, any host could control and monitor any network element. Authentication is

based on a community string, which is a shared secret between an agent and a managing host. An agent uses the community string and the address of the requesting host to determine the level of access for a request. Community records have this format:

COMMUNITY *community host access*

where:

community is an ASCII string of up to 64 characters naming the community

host is the host's name or Internet address in dot notation (for example, **128.223.1.2**), or **0.0.0.0** to allow any host to make a request of the community.

access is **READ** for read-only access, **WRITE** for read and write access, or **NONE** for no access

This example gives any host read-only access to the community named **public** and gives host **128.223.2.1** read-write access to the same community:

```
COMMUNITY public 0.0.0.0       READ
COMMUNITY public 128.223.2.1   WRITE
```

If an unknown community string is specified in a request, an agent does not respond. Instead, it sends an authentication trap (if traps are enabled). The current authentication scheme is known as "trivial authentication" because the community string is not encrypted as it crosses the network.

Defining Traps

A trap is an asynchronous message sent from an agent to indicate a change in its state. When traps are enabled, an agent sends a trap whenever the agent re–initializes, a communication link goes up or down, or a request fails to pass authentication. (See **Trap–PDU** above in "SNMP Messages" for a description of trap messages.) Trap records indicate where such messages are sent (typically to a management station.) Trap records have this format:

TRAP *community host port*

where:

community is an ASCII string of up to 64 characters naming the community that should receive the message. The community name must be valid for the network management station receiving the traps.

host is the receiving host's name or Internet address in dot notation (for example, **128.223.1.2**)

port is the port number on which network management station is listening for traps. The default is **162**.

Example:

```
TRAP public 128.223.8.48    162
```

Subagents

The SNMP agent supports the EMANATE extensibility mechanism created by SNMP Research Inc. Subagent records specify a pathname to a shared library subagent. Do not add, delete, or modify subagent records. The subagents provided with DG/UX are specific to DG/UX and cannot be used with any other operating system.

Using snmpd

To use the SNMP agent, run **snmpd**. The **rc** scripts automatically start **snmpd** while going to run level 3. **snmpd** handles requests from the network sequentially in a first-in, first-out order. The syntax is as follows:

snmpd [**-v**] [**-d**] [**-p** *interval*] [**-P** *port*]

When you invoke **snmpd**, it disassociates itself from the controlling terminal, reads its configuration files, and begins servicing network requests. The following options change the default behavior of **snmpd**.

-v Causes **snmpd** to remain attached to the controlling terminal and to print additional information about the packets received and transmitted. Use this option to verify information that the agent and the manager exchange, or to debug the agent when it does not function as it should.

-d Causes **snmpd** to remain attached to the controlling terminal, but does not print information about packets. For debugging, use this option to view error messages without getting packet information.

-p *interval* Sets the polling interval (in seconds) that **snmpd** uses to check for changes in the state of the interfaces. The default polling interval is 60 seconds. When **snmpd**

determines that an interface has changed state, it sends a **linkUp** or **linkDown** trap to all the managers configured in the **snmpd.conf** file.

To indicate that **snmpd** should not poll, specify a value of **0**. If no traps are configured, **snmpd** does not poll regardless of the *interval* specified. Regardless of whether **snmpd** polls, an explicit request to check the state of an interface returns accurate information.

-P *port* Sets the UDP port number for incoming requests that **snmpd** listens at. The default is port 161.

snmpd listens on the UDP port for incoming SNMP requests. Once it receives a request, **snmpd** checks it using the trivial authentication scheme. Trivial authentication verifies that the community string is one configured in the **snmpd.conf** file, and that the requesting host and the community specified have the proper access for the type of request made. For example, trivial authentication checks that the host and community specified have read-write access for a **setRequest** operation.

Handling Errors

A single SNMP request may contain many objects, but if any fail, the entire request fails. The following error conditions and actions may occur:

- `snmpsetany: no response - try again` (return code –1)

  The agent does not respond to requests that do not pass authentication. If **snmpEnableAuthenTraps** is enabled , an `authenticationFailure` trap occurs is sent to the management stations configured in the trap records of `snmpd.conf`.

- `No such variable name` (return code 1)

  Error status `noSuchName` occurs if a request contains an unknown instance, or if a set request is received from a recognized host and a known community without write access.

- `Bad variable value` (return code 1)

  Error status `badValue` occurs if the value specified for an object in a set request is not of the correct type or is out of range.

Using SNMP Commands

TCP/IP for AViiON Systems provides several commands that let you query the MIB object tree and perform network and system management functions. Use these options and arguments with most of the commands:

–n	Display object IDs in numeric rather than symbolic format
–t	Display the data types of objects in addition to their names and values
host	The name or IP address of the host being queried
community	The authentication string
object	The name or dot–notation address of the MIB object to be set or displayed

snmpgetone

Use **snmpgetone** to retrieve individual object instances.

snmpgetone [**–n**] [**–t**] *host community object* ...

The following command returns the current values of object instances **sysDescr.0** and **ifNumber.0** on host **dg1**;

```
% snmpgetone dg1 public sysDescr.0 ifNumber.0 ↵
sysDescr.0 = DG/UX TCP/IP SNMP AGENT
ifNumber.0 = 2
```

snmpgetnext

Use **snmpgetnext** to retrieve a set of objects or object instances.

snmpgetnext [**–n**] [**–t**] *host community object* ...

Because the function uses the **GetNext-PDU** operator, the value returned is the next higher object ID in the sequence of the specified object.

The following command returns the object instances **sysDescr.0** and **ifNumber.0**:

```
% snmpgetnext dg1 public sysDescr ifNumber ↵
sysDescr.0 = DG/UX TCP/IP SNMP AGENT
ifNumber.0 = 2
```

snmpgetmany

Use **snmpgetmany** to retrieve an object, subtree, or the entire MIB tree.

snmpgetmany [**–n**] [**–t**] *host community object* ...

This following command traverses the **ifTable** aggregate object:

```
% snmpgetmany dg1 public ifTable ↵
```

```
ifIndex.1 = 1
ifIndex.2 = 2
ifDescr.1 = inen0
ifDescr.2 = loop0
ifType.1 = 6
ifType.2 = 24
ifMtu.1 = 1500
ifMtu.2 = 1500
ifSpeed.1 = 10000000
ifSpeed.2 = 10000000
. . .
```

The **snmpgetmany** command stops retrieving object classes when it receives a `No Such Name` error, which indicates that it has reached the end of the MIB, or when it reaches the end of a subtree.

The following command retrieves the entire MIB tree named **iso**:

snmpgetmany *hostname* **public iso**

snmpgettab

Use **snmpgettab** to retrieve, one row at a time, all the objects that comprise a specified MIB table. The command has two formats:

snmpgettab –help
snmpgettab [**–n**] [**–t**] *host community table*

The *table* argument identifies a MIB table. Use the first format to display the valid *table* arguments.

The following example displays the ARP table on a host named `dg1`:

```
% snmpgettab dg1 public ipNetToMediaTable ↵
```

snmpsetany

Use **snmpsetany** to set object instances to a specified value.

snmpsetany *host community* [*instance type value*] ...

The *instance* is a name in dot notation identifying the instance to be set to the specified *value*. The *type* option indicates the format of *value*. Table 6–1 lists the valid *type* options.

Table 6–1 Object Data Entry Types

Option	Value
–i	integer
–o	octet (hexadecimal notation)
–d	object identifier (dot notation)
–a	Internet address (dot notation)
–c	counter
–g	gauge
–t	time-ticks
–s	octet (display string notation)
–D	same as **–s**
–N	NULL

Consider the following sequence of commands. Working on host **dg1**, you use the **snmpgetmany** command on the object **ifOperStatus**, which returns the current status of the interface. It shows that interface 2 is down (has a value of **2**):

```
% snmpgetmany dg1 public ifOperStatus ↵

ifOperStatus.1 = 1
ifOperStatus.2 = 2
```

You then use the **snmpgetone** command to determine which interface is down. This shows the loopback device to be down, which **ifconfig** confirms:

```
% snmpgetone dg1 public ifDescr.2 ↵

ifDescr.2 = loop0
% ifconfig loop0 ↵

loop0: 127.0.0.1 flags=48<LOOPBACK,RUNNING>
netmask=0xff000000  metric=0
```

You then use **snmpsetany** with object **ifAdminStatus** to bring the loopback device back up, and ifconfig to confirm:

```
% snmpsetany dg1 my_community ifAdminStatus.2 -i 1 ↵

ifAdminStatus.2 = 1
% ifconfig loop0 ↵

loop0: 127.0.0.1
flags=449<UP,LOOPBACK,RUNNING,STARTED>
netmask=0xff000000  metric=0
```

snmptraprecv

As the superuser, you can run **snmptraprecv** to receive traps from network elements that generate them.

snmptraprecv [**–d**] [*port*]

The **snmptraprecv** command binds to the SNMP trap port (UDP port 162) or to the specified *port* to listen for traps. It prints messages corresponding to the traps it has received.

The following example starts **snmptraprecv** in the background and restarts **snmpd**, which sends a **coldStart** trap. The resulting display shows two trap messages for host 128.223.2.2, one to community `public` and the other to `my_community`.

```
# snmptraprecv & ↵
[1] 893
# snmpd ↵
#
Community: public
Enterprise: DataGeneral
Agent-addr: 128.223.2.2
Cold start trap.
Time Ticks: 36
iso.1.1.1 = NULL

Community: my_community
Enterprise: DataGeneral
Agent-addr: 128.223.2.2
Cold start trap.
Time Ticks: 36
iso.1.1.1 = NULL
```

snmptrapsend

As the superuser, you can run **snmptrapsend** to send traps to stations that monitor for them. The command provides a

low-overhead way to generate a trap. An application can generate traps with this command without requiring support from the SNMP agent. The command has two formats:

snmptrapsend *host community standard_trap* [*port*]
snmptrapsend *host community enterprise_trap/id* [*object type value ...*] [*port*]

The standard_trap argument may be an integer 0–6, or one of the following values:

host	A host name or Internet address in dot notation indicating the trap destination.
community	The name of the community.
port	The UDP port (default 162) that the command binds to.
standard_trap	An integer 0–6, or one of the following values:
coldStart(0)	Agent is re–initializing itself because the configuration or the implementation may have been altered.
warmStart(1)	Agent is re–initializing itself, but the configuration or implementation has not been altered.
linkDown(2)	Agent recognizes a failure in one of the communication links in the agent's configuration.
linkUp(3)	Agent recognizes that one of the communication links in the agent's configuration has returned to operation.
authenticationFailure(4)	The request failed authentication.
egpNeighborLoss(5)	The Exterior Gateway Protocol (EGP) peer has been marked down.
enterpriseSpecific(6)	This type of protocol data unit (PDU) is vendor specific, and so has no standard meaning.
enterprise_trap / id	An object identifying the enterprise that defined the trap. For example, `dataGeneral.1.2/108` identifies the MIB subtree **dgHwIdAviionHw** defined by Data General. (See Appendix A for a description of the Data General MIB hierarchy.)

object	The identifier of an object, which may be a standard SNMP object or an enterprise–defined object.
type	An option switch indicating the data type of a supplied object value. Valid options are the same as those described above for **snmpsetany**: see Table 6–1.
value	The supplied object value, which must be in the format indicated by the *type* switch.

In the following example, a **linkDown** trap is sent to the **snmptraprecv** running from the example in the previous section:

```
# snmptrapsend ↵

usage: snmptrapsend host community type [port]
# snmptrapsend dg1 public linkDown ↵
Community: public
Enterprise: experimental.1.42.42.42.42
Agent-addr: 128.223.2.2
Link down trap.
Time Ticks: 0
iso.1.1.1 = NULL
```

End of Chapter

7 Configuring and Using the Serial Line Internet Protocol (SLIP)

This chapter tells you how to install, manage, and use the Serial Line Internet Protocol (SLIP) on DG/UX systems. SLIP enables you to run TCP/IP applications such as **telnet** and **ftp** over point-to-point serial lines. Systems running SLIP generally use modems to communicate across telephone lines.

Understanding SLIP

SLIP provides a point-to-point serial link over which you can run TCP/IP applications. The link is usually established through modems, though hosts can be connected directly with serial cables. Because SLIP works over asynchronous lines, applications run significantly slower than over Ethernet or other local area network (LAN) media.

Despite its relative slowness, SLIP can often be more convenient than networking alternatives. Serial lines are inexpensive and easier to install than other sorts of media, and you can use existing telephone lines to establish SLIP links. Where your LAN cabling is restricted due to terrain, distance, and other factors, you can often make network connections with serial lines.

Data General's SLIP is based on *RFC 1055 (A Nonstandard for Transmission of IP Datagrams over Serial Lines)*. Header compression as defined in *RFC 1144 (Compressing TCP/IP Headers for Low-Speed Serial Links)* is also supported as an option. If you use header compression, performance of interactive applications over the SLIP link will improve.

Figure 7–1 shows a connection between **sys23**, a remote SLIP client, and **sys05**, a SLIP server connected to an Ethernet LAN. This link allows either host to run TCP/IP applications such as **telnet** and **ftp**. Furthermore, the link allows the SLIP client **sys23** to connect to any system available to the SLIP server **sys05,** assuming appropriate routes are installed.

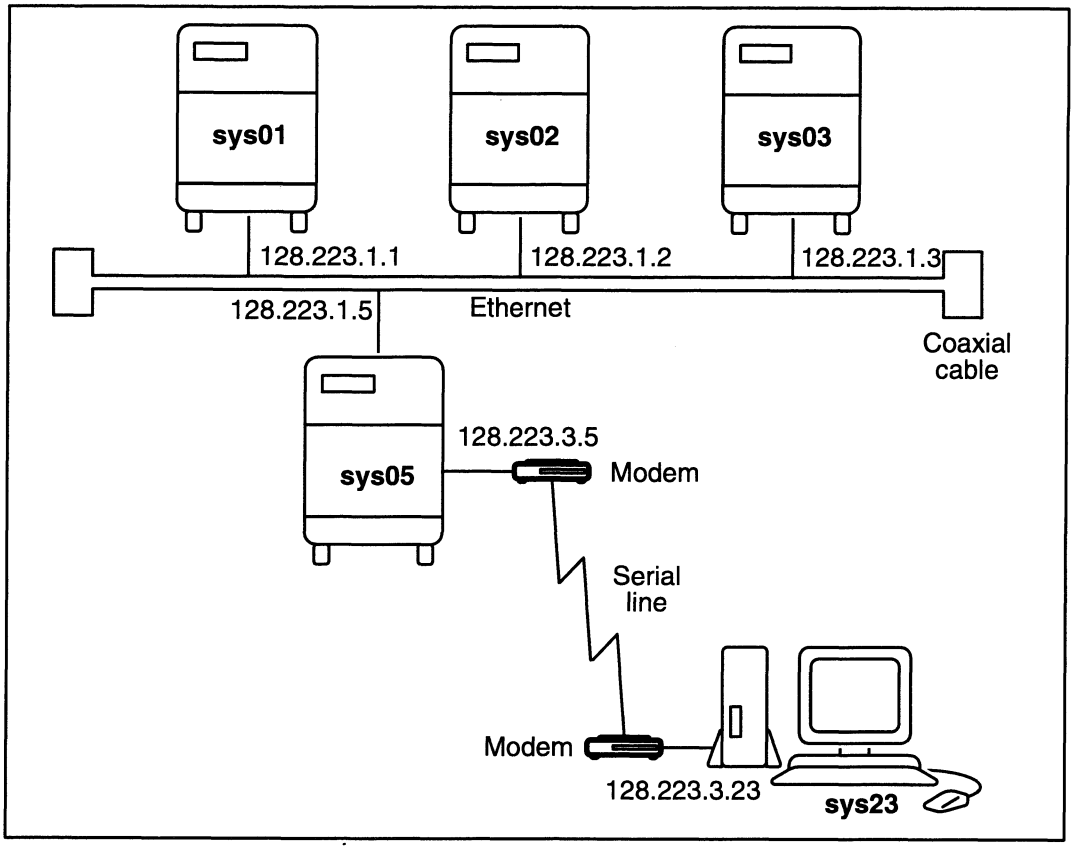


Figure 7–1 SLIP Connection to a Host on an Ethernet LAN

SLIP Setup

The following sections describe how to set up SLIP on a DG/UX system. The first section gives an example of the modem setup required for a SLIP connection. The next two sections describe how to prepare SLIP on calling (client) and receiving (server) hosts respectively. The last section tells how to make two cables that you can use to directly connect two hosts with a SLIP link.

See Chapter 3, "Manage SLIP," for SLIP setup and maintenance procedures. For modem setup and installation, see *Managing Modems and UUCP on the DG/UX System* (093–000698).

Modem Requirements

The functionality and operating efficiency of SLIP depend on your modem configuration. Table 7–1 shows typical front switches settings for a Data General model 18901 modem (part number 119-2087). These settings apply for both call-out and call-in SLIP services.

IMPORTANT: These tables are for the Hayes Smartmodem™ AT command set. If your system does not use the Hayes™ commands, substitute the equivalent command.

Table 7–1 Modem Front Switch Settings

Switch	Setting	AT Command	Remarks
1	Up	AT&D2	Respond to DTR
2	Down	none	AT Command Mode
3	Up	none	AT Command Mode
4	Up	ATE1	Command echo enabled
5	Up	ATS0=1	Auto-answer enabled
6	Down	AT&C1\D0	CD follows modem carrier, DSR and CTS on
7	Down	AT&C1\D0	CD follows modem carrier, DSR and CTS on
8	Down	AT–H0	Smart mode
9	Up	none	Not used
10	Up	none	Asynchronous operation

Table 7–2 shows typical back switch settings for the same type of modem.

Table 7–2 Modem Front Switch Settings

Switch	Setting	AT Command	Remarks
1	Down	AT\Q3	Bidirectional hardware flow control
2	Up	AT\Q3	Bidirectional hardware flow control
3	Down	AT\N3	Microcom Networking Protocol™ (MNP®) and LAPM auto-reliable mode
4	Up	AT\N3	MNP and LAPM auto-reliable mode
5	Down	AT\V2	Use V.42 result codes
6	Up	none	Restore configuration stored with **AT&W** or **AT*W** in the event of a reset, **ATZ**, or power up
7	Down	none	Read switches in the event of reset, **AT&F**, **ATZ** or power up
8	Down	AT\J0	BPS rate adjust disabled

Once you have set your modem up correctly, you can issue the **AT\S** command, or its equivalent on your system, to review additional configuration options. Table 7–3 shows typical configuration options for the the Data General model 18901 modem. If you are not using this modem, use the table as a general guide to set up your modem.

Table 7–3 Modem Configuration Options

Option	State	AT Command
MODEM BPS	9600	AT%G0
MODEM FLOW	OFF	AT\G0
modem mode	AUT	AT\N3
AUTO ANS.	on	ATS0=1
serial bps	19200	AT%U0
bps adjust	off	AT\J0
answer messgs	off	atq2
serial flow	bhw	at\q3
pass xon/xoff	off	at\x0
parity	8n	at
break	5	at\k5
exit char	043	ats2=43
cmd echo	on	ate1
results	on	atq0
result type	mnpx	atv1\v2
conn mnp–	0	at–m0
speed match	1	at%l1
equalizer	on	at:e1
fallback	2	at–q2
data echo	off	at\e0
inact timer	00	at\t0
auto retrain	on	at%e1
compression	all	at%c3
max blk size	256	at\a3
auto buff	0	at\c0
auto char	000	at%a0
emulating hp	off	at\h0
pause time	002	ats8=2
dtr	2	at&d2
carr det	1	at&c1

Continued

Table 7–3 Modem Configuration Options

Option	State	AT Command
dsr	0	at\d0
ring ind	1	at\r1
spkr ctrl	1	atm1
lease line	0	at&l0
async/sync	0	At&m0
cts/rts	0	at&r0
lng spc disc	off	aty0
sim ring	0	at:r0
cd delay	000	at:u0
cts delay	000	at:v0
dsr delay	000	at:x0
disc delay	000	at%d0
rem char	042	at*s42
rem enable	off	at*e0
rem sec	off	at*r0
rdlb enable	on	at&t4
dial mode	4	atx4
pulse dial	60%	at&p0
v.24 tst modes	off	at%h0
guard tone	0	at&g0
async protocol	none	at:k0
kermit mark	001	at:q1
par chk	0	at–p0
manual dial	0	at:d0
cont 1200 bps	off	at*h0
detect phase	on	at–j1
mnp ext svc	on	at–k1
run diags	on	at$d1
read switches	on	at$k0
o/a button	on	at$o0
s/a switch	on	at$s0
td enable	on	at$t0
autologon view	on	at$v1
autologon ans	0	at$a0

Continued

Table 7–3 Modem Configuration Options

Option	State	AT Command
clock source	mdm	at&x0
bell	on	atb1

Preparing the Call-Out Side

To prepare the call-out (client) side, you should ensure that a TTY port is available for SLIP to use. SLIP cannot use a port that is locked by another service. SLIP will tell you if the port it is trying to use is locked by another service. SLIP will also arbitrate port locking with other services.

For call-out service, you must set up dialstring entries in the **/etc/slipdialinfo** database and link parameters in the **/etc/slipusers** database. See "Manage SLIP " in Chapter 3 for the **sysadm** procedures. See "SLIP Database Formats" later in this chapter for a description of the formats of these files.

Preparing the Call-In Side

To prepare the call-in (server) side, you need to do some additional TTY port set up operations. You will need to add a port service for the TTY port and decide how to set up the initial process for your SLIP users.

For call-in service, you must set up link parameters in the **/etc/slipusers** database, as explained in "Manage SLIP" in Chapter 3.

Adding a Port Service for SLIP

Follow these steps to add a port service:

1. Become the superuser and execute **sysadm**.
2. If you do not already have a **ttymon** port monitor, then you need to add one. You can select Device—>Port—>Port Monitor—>List to see if you have a **ttymon** port monitor.

   Follow these steps to add the port monitor:

   a. Select Device—>Port—>Port Monitor—>Add

   sysadm asks for the "Port monitor type."

   b. Select "ttymon."

   sysadm asks for the "Port monitor tag:"

c. Enter any tag you want. This example uses the tag "modem."

d. Take the defaults for the rest of the parameters.

3. Select Device—>Port—>Port Service—>Add

   sysadm asks for the "Controlling port monitor for service."

4. Select your **ttymon** port monitor. This example uses "modem."

   sysadm asks for the "Port service tag."

5. Enter any tag you want. This example uses the tag "slip."

6. Take the defaults for all queries until "Path name of terminal device."

7. Enter the TTY port you want to use for SLIP service. This example uses "/dev/tty00."

   sysadm asks for the "TTY Definition Label."

8. Select the baud rate for your SLIP service. If you are using a modem, then you must select one of the labels that start with "M." This example uses "M19200."

9. Take the default for "Service command."

10. Select "no" for "Hangup?"

11. Select "yes" for "Connect on Carrier?"

12. Select "yes" for "Bidirectional." However, if you do not want to have both call-out and call-in service on your system, you can take the default.

    sysadm asks for "Wait-read value:"

13. Enter the value you want. This example uses "0," so a login prompt will be returned for any character sent.

    sysadm asks for "Timeout value:"

14. Enter the amount of time in seconds that the port monitor will be open and inactive before hanging up. This example uses "30," so the port monitor will never time out after 30 seconds.

15. Take the defaults for the rest of the queries.

    For more information about managing ports and services, see Chapter 9 in *Managing the DG/UX™ System*.

Handling your SLIP Users

You can set up your user account so that SLIP is the initial process invoked when a user logs in to the system. You might find this preferable in the following circumstances:

Some users will be accessing your system only through SLIP

You want to provide separate user accounts just for SLIP access

This reduces the steps needed to make a SLIP link and consequently simplifies the management of dialstrings in **/etc/slipdialinfo**. Also, invoking **slipd** as the initial process prevents general shell access and provides extra security should someone try to log in with a stolen username and password.

You can set up SLIP as the initial process for one of your users in the **/etc/passwd** file. Usually, an entry in the file looks like the following:

```
frazier:iLJliKHkhiLHi:964:50:User Frazier LOGIN:/home/frazier:/bin/sh
```

To invoke SLIP as the initial process for this user, change this entry as follows:

```
frazier:iLJliKHkhiLHi:964:50:User Frazier SLIP LOGIN:/home/frazier:/usr/bin/slipd
```

See "Manage SLIP" in Chapter 3 for the **slipdialinfo** maintenance procedures.

Using SLIP Without a Modem

You can directly connect two hosts together with a serial cable and use SLIP to communicate between them. This is still a host-to-host connection and cannot be used to form any sort of multi–access LAN.

Cable Requirements

Data General does not supply a cable specifically for SLIP connections. However, you can make your own cable using standard 25-Pin D Shell Connectors.

Figure 7–2 shows the cable wiring diagram for a SLIP cable used to connect two DTE (Data Terminal Equipment) standard devices or between two DCE (Data Communications Equipment) devices.

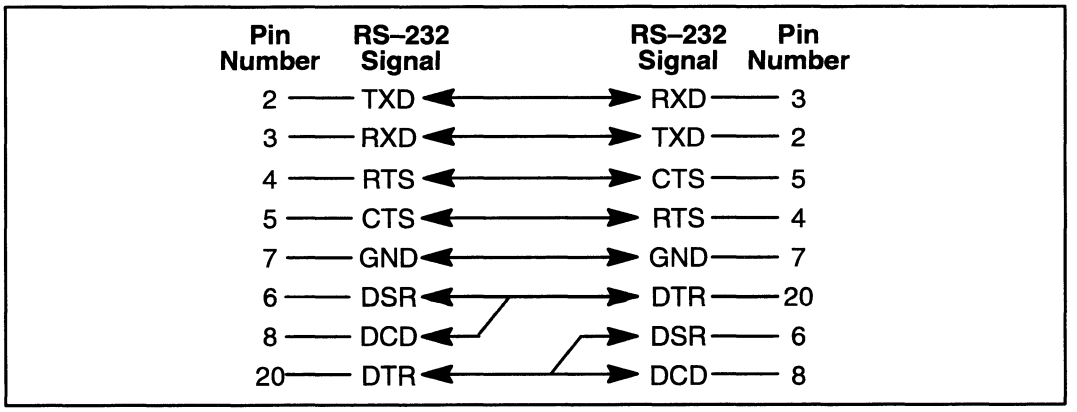


Figure 7–2 Pin Configuration for DTE–DTE or DCE–DCE Connections

Figure 7–3 shows the cable wiring diagram for a SLIP cable used to connect a DTE device (left) to a DCE device (right).

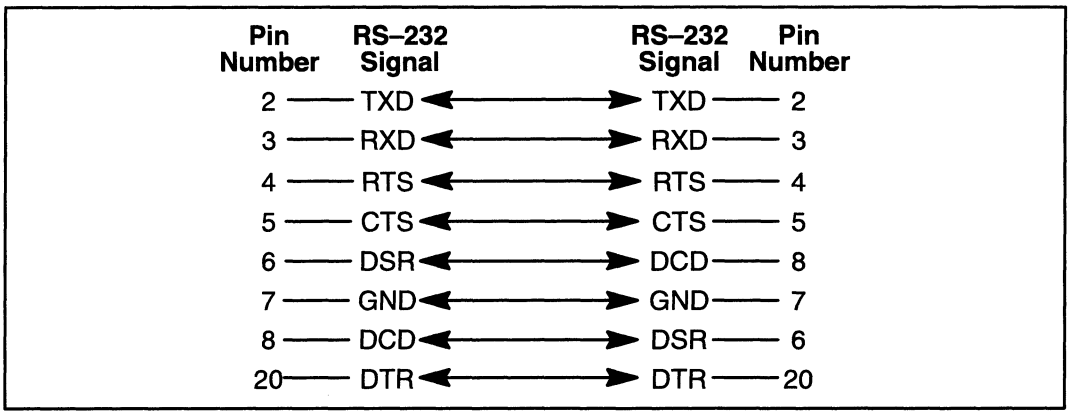


Figure 7–3 Pin Configuration for DTE–DCE Connections

IMPORTANT: AViiON 4600 and 4620 systems use 9-pin connectors instead of 25-pin connectors. You need a 9-Pin to 25-Pin Adapter to make a direct cable SLIP connection.

Contact your Data General sales representative for appropriate adapter and cabling information.

Managing SLIP

Section "Manage SLIP" in Chapter 3 explains how to set up, start, and stop a SLIP connection from **sysadm**. This section describes how to manage SLIP from the command line, describes SLIP configuration files, and discusses security issues.

Starting SLIP with slipd

IMPORTANT: Before you can start a SLIP connection, there must be valid entries in the **/etc/slipusers** and **/etc/slipdialinfo** files on both the local and remote systems.

- Once you start SLIP, the connection remains active until you explicitly tell it to stop (or until the other end hangs up).

  Establishing a connection with **slipd** has the following advantages over using **sysadm:**

  - You can supply command line switches to override the link initialization parameters in the **/etc/slipusers** file.
  - You can use the **–d** option to display debug traces during the operation. (Note that this forces the SLIP operation to remain in the foreground. See "Troubleshooting SLIP" in Chapter 8 for more information on **slipd** debug mode.)

To start a SLIP connection, you type a command like the following:

```
slipd -b2400 -ltty00 goolsby ↵
```

The time required to establish a connection varies. To let you know the operation is proceeding, SLIP displays this message and prints a series of dots at the end:

```
SLIPD (DG/UX TCP/IP Release 5.4R3.00) ready - Auto dial mode to 'system'
  Establishing remote connection, please be patient...
```

If SLIP is able to connect to the remote system, it displays the following message:

```
Slipd READY - Connected to 'system'
```

If SLIP encounters a problem establishing the connection, it displays the following message:

```
Unable to establish remote connection.
```

In this case, wait a while and then try to establish the connection again. The other system's modem might be busy. If you still cannot make a connection after several tries, check to be sure your modem is set up correctly and the information in your SLIP databases is correct.

If SLIP is able to make the remote connection but cannot establish the SLIP link within 90 seconds, it displays the following message:

```
Timeout connecting to remote host
```

See the "Troubleshooting SLIPD" in Chapter 8 for assistance in isolating the problem.

You must enter the baud rate (**-b**), tty port (**-l**), and *dialsystem* to start a call-out sequence when you invoke **slipd** from the command line. You can also specify the local host address (**-h**), remote host address (**-r**), network mask (**-m**), and the link initialization flags (**-f**) for *dialsystem*. These link parameters override those entered in the **slipusers** database for *dialsystem*.

For more information, see the **slipd**(1C) manual page.

How SLIP Makes a Connection

SLIP performs the following steps while making a connection:

1. The user invokes the **slipd** command to call-out to a SLIP server.
2. The client **slipd** verifies the command parameters and then uses the specified *dialsystem* as a key to get link parameters in **/etc/slipusers**.

   If the *dialsystem* entry does not exist (or, in the case of a server, is not specified), **slipd** tries to get the link parameters using the user's login name as the key.

   If that fails, **slipd** displays the following error message and exits:

   ```
   Unauthorized user: 'dialsystem'
   ```

3. The client **slipd** tries to open the specified *tty* device and **/dev/ip**.

   If the *tty* is locked, (due to another service running on the port), **slipd** displays the following error message and exits:

   ```
   Error obtaining tty lock
   ```

4. The client **slipd** uses *dialsystem* as a key to get send and expect dialstrings from the **$HOME/.slipdialinfo** database.

   If the **.slipdialinfo** file does not exist or there is no matching record in that file, **slipd** looks in the **/etc/slipdialinfo** database.

   If no matching record is found in that file, **slipd** displays the following error message and exits:

   ```
   Specified dialsystem not found
   ```

5. The client **slipd** communicates with the TTY port and issues the commands in the dialstrings. This usually involves dialing the remote machine, logging in, and invoking the server **slipd** process.

   Specifying the debug (**–d**) option when **slipd** is invoked forces the client to display all of the dialstring dialogue.

6. The server **slipd** is invoked on the server and uses the login name as a key to get the link initialization parameters from `/etc/slipusers`.
7. The server **slipd** tries to open the *tty* port and **/dev/ip**.

   Typically, the server **slipd** process should run with input/output to the **stdin/stdout**. This is the default, unless you specify the **–l** option.

   After validating the link parameters and opening the devices, the server **slipd** announces its transition to IP mode by reporting:

   ```
   SUNSLIP
   Attaching remote-host (remote-addr) to network via local-host
   (local-addr)
   ```

8. The client **slipd** is configured to see the IP mode transition banner as its last 'expect' string and also makes the transition to IP mode.
9. Both the client and server **slipd** processes install point-to-point host routes and (if specified) a default route to its peer.

   If the client **slipd** is not running in debug mode, it backgrounds itself to free up the controlling terminal.

Stopping SLIP

IMPORTANT: If you stop a SLIP connection while an application such as **ftp** is running over the link, the application hangs.

To stop a SLIP connection, you send a SIGHUP signal to the **slipd** process. You can kill the process by using the **ps** command to find the process id and the **kill** command to stop the process. You can also stop all **slipd** processes with the **dg_kill** command.

For example, enter the following command line to find the process id of **slipd**:

```
ps -ef | grep 'slipd' ↵
```

The system might display the following:

```
root  2745  2494  7 16:03:00 p6       0:00 grep slipd
root  2730     1  7 15:56:17 00       0:00 slipd -b19200 -ltty00 goolsby
```

You would then enter this command to kill the process:

```
kill -1 2730 ↵
```

If you want to kill all **slipd** processes without having to check for the process id, then you could enter the following command line

```
dg_kill -1 slipd ↵
```

SLIP Database Formats

SLIP uses two files to make connections, **/etc/slipdialinfo** and **/etc/slipusers**. Chapter 3 explains how to manage these files with **sysadm**. This section describes the format of these files. (If you prefer, you can maintain these files with a text editor, or by using the **admslipdial**(1M) and **admslipuser**(1M) commands, as explained in their manual pages.)

Slipdialinfo contains the send-expect dialstrings SLIP uses to initialize a link. The file has the following format:

```
dialsystem   send   expect   [send   expect]   ...
```

dialsystem is a label used to find the dialstrings for establishing a SLIP link with the remote host.

send and *expect* are the strings sent to the modem (port) and received in response. Items in an entry are separated by any number of spaces or tabs. The *send* and *expect* strings may contain:

\\	the "\" character
\w	pause for 1/2 second
\r	carriage return
\n	newline
\s	space

The last *expect* string must be "SLIP", since this is the last string the SLIP server sends before making the transition to IP mode. Data General's SLIP implementation sends a startup string compatible with PC-NFS™ (that is, SUNSLIP plus link address information).

Comment lines should start with a "#" character and blank lines are ignored.

The following example is included in the **/etc/slipdialinfo.proto** prototype file:

```
# Dialsystem                    Auto-login Send/Expect Dialstrings
# ------------------------------------------------------------------------
  frazier        ATDT1234 CONNECT \r ogin: name word: pswd $ exec\sslipd SLIP
```

If you had set up entries in **/etc/passwd** to invoke SLIP as the initial process for a user (see "Preparing the Call-In Side", then you would change this entry to the following:

```
# Dialsystem                    Auto-login Send/Expect Dialstrings
# ------------------------------------------------------------------------
  frazier        ATDT1234 CONNECT \r ogin: name word: pswd SLIP
```

In addition to the main **/etc/slipdialinfo** file, individual SLIP users can have a **.slipdialinfo** file in their home directories. This file allows non-superusers to manipulate their own dialstrings. The format of **.slipdialinfo** is identical to that of **/etc/slipdialinfo**.

Slipusers contains the link initialization flags, remote and local IP addresses, and the network mask used by **slipd** to form a link. The file has the following format:

```
params_tag flags remoteaddr localaddr netmask
```

params_tag is a label used to locate the link parameters for a SLIP connection. It usually should be the same as the corresponding *dialsystem* in the **slipdialinfo** database.

flags is used to control certain options in the SLIP link. Putting a "+" before a flag turns on the option. Putting a "–" before a flag turns off the option. The following flags are supported:

c controls IP packet header compression

e controls automatic IP packet header compression recognition and activation. If activated, both sides of the link must support header compression. With header compression activated, SLIP recognizes packets with compressed headers and automatically shifts to compressed header mode upon receiving such a packet.

r installs a default route via the SLIP interface, if one does not exist

remoteaddr and *localaddr* are the Internet address for the interfaces to be connected by the SLIP link. They may be either a valid hostname or an Internet address in dot format.

netmask is the network mask used for subnetting by the local system.

Comment lines should start with a "#" character and blank lines are ignored.

The following example is included in the **/etc/slipusers.proto** prototype file:

```
# Params_tag    Flags     Remote             Local              Netmask
# ----------    ------    ---------------    ---------------    -------------
 frazier        -c+e-r    sl-commtg3         sl-packfan         255.255.255.0
```

This entry could also have the Internet address for the local and remote systems, as follows:

```
# Params_tag    Flags     Remote             Local              Netmask
# ----------    ------    ---------------    ---------------    -------------
 frazier        -c+e-r    128.223.5.5        128.223.5.12       255.255.255.0
```

See the **slipusers**(4M) manual page for more information.

Security Issues

System administrators need to be aware of some SLIP-related security issues. These issues fall into three areas: access to **slipd**, access to the SLIP databases, and setting up SLIP with a shell as the initial process.

Access to slipd

By default, any user can start a SLIP link. This default behavior may create a security problem. A user could use the **-h** startup option to pretend to be any Internet address. If a remote system has a **.rhosts** file to allow remote shell access, then such a user might be able to gain illegal access to other people's accounts.

To restrict the ability to start a SLIP connection, you can create a users' group just for SLIP access. Add the users that you want to have SLIP access to this group. Use **chgrp** to change the **/usr/bin/slipd** group to your new group, and **chmod** to restrict access to root and group members. Now, only users you assign to this group can access SLIP.

For more information, see the **chgrp**(1), **chmod**(1), and **group**(4) manual pages.

Access to the SLIP Databases

Usually, a user's username and password are included in the **slipdialinfo** file's *send* strings. These passwords are not encoded in any way, so make sure the permissions on the SLIP databases provide root–only access. (The files are created with root–only access by default.) Remind users with **.slipdialinfo** files in their home directories to restrict their access permissions as well.

SLIP's Initial Process

If you choose to set up your user accounts so the initial process is a shell instead of **slipd**, then your **slipdialinfo** *dialstring* entry should invoke **slipd** with the **exec** command, **exec\sslipd**. This frees up space in the process table. When invoked with **exec**, **slipd** replaces the original **sh** or **csh** process.

End of chapter

8 Troubleshooting

Troubleshooting involves asking questions and performing tests to isolate the cause of a problem. For example, did the problem occur immediately after you set up the TCP/IP for AViiON Systems package? Is the problem due to hardware, software, or some interaction of hardware and software? Is the problem with your local host, or with a remote host?

When you encounter a problem, try to determine exactly when and where it occurs. First, does the problem occur just after you set up the package, or does it occur after the package has been running for a while? If this is the first time that you have run TCP/IP for AViiON Systems, go to "Isolating a Problem After Setup." If the problem occurs after TCP/IP has been running for a while, go to "Determining the Source of the Problem."

Isolating a Problem After Setup

Here is a general checklist to isolate problems that may occur shortly after you set up TCP/IP for AViiON Systems.

- Did you run hardware diagnostics on the system and on the communications board before you loaded any software? If you did not, do so now.
- What revision of each package are you using? Are you using versions of software that cannot work together? Check your release notice for product dependencies. Are all the requirements in the "Environment" section of the release notice met?
- Go through the setup procedure one more time. Try to pinpoint where the problem occurs. Make sure you have entered the correct values for your host name, Internet address, and network mask.
- If the **rc** scripts in run level 2 or 3 do not make the networking software come up fully operational, can you determine which line in a script has failed? Check your run level with **who -r**. Run level 2 allows only outgoing connections, and run level 3 allows incoming and outgoing connections. Record any error message that appears on the master console.
- You can also check the log files for error messages. The **/var/setup.d/log/tcpip.root** and **/var/setup.d/log/tcpip.usr** files contain the setup logs. The **/etc/log/init.log** file contains the **rc** scripts log.
- If the following message appears, either **/etc/services** does not exist or an entry for the service you have attempted is missing from the file (for example, telnet):

```
unknown service
```

If you are not running NIS and if you do not have the **/etc/services** file, you will get the error for the daemons that require the file: **ftpd**, **tftpd**, **rlogind**, **rshd**, and **rexecd**. Solve this problem by copying the prototype file. For example, as the superuser, execute the following command:

```
# cp /etc/services.proto /etc/services ↵
```

You can also get entries for any missing individual services from **services.proto**.

- If the message `socket: Address family not supported by protocol family` appears when you try to connect to a remote host, it probably means the kernel could not create sockets for network daemons because it lacks the necessary protocols. This usually happens for one of three reasons:

  The kernel was not rebuilt after you set up TCP/IP.

  The kernel was rebuilt, but you inadvertently booted the old kernel.

  The kernel was built incorrectly.

  You can verify that TCP/IP has been built into a kernel (for example, **dgux.aviion**) with the following command line:

  ```
  % sysdef /dgux.aviion | grep tcp ↵
  ```

  If "tcp" is output by this command, TCP/IP is built into the kernel.

  If TCP/IP was not built into the kernel, append TCP/IP-specific information (**system.tcpip.proto**) to the system file (**system**). Then, from **sysadm**, select `System-> Kernel-> Rebuild` to rebuild and install the kernel. Then reboot the system.

Determining the Source of the Problem

If TCP/IP successfully ran in the past, the best way to isolate a problem is through the process of elimination. Such a process involves determining where in the network the problem occurs, and what function of TCP/IP is giving you trouble. The following sections present a method for eliminating possible sources of network problems.

Assume that you are logged in to some local host, host A, and that you are trying to use TCP/IP to reach a remote host, host B, when something goes wrong. The following diagram illustrates the setup.

With this scenario, troubleshooting involves the following three basic steps:

1. Check the hardware.
2. Check the local host (host A).
3. Check the remote host (host B).

If you suspect that the problem is due to software rather than hardware, determine which layer is at fault. Is the problem at a high layer (a network application) or at the low layers (protocols, device driver)? If at a high layer, which application is responsible?

As you troubleshoot, take careful and complete notes of what you do and how the system responds to your actions. Save these notes because even if you cannot remedy your problem, they are valuable data for your support center when you call them for help.

Step 1: Check the Hardware

If the problem does not occur just after setup, but after TCP/IP has been running for a while, first check the hardware on the local host (host A). Check the following:

- Is your local network controller operating correctly? This should be apparent at boot time. For more information about how to check the controller, see "Using the ifconfig Command" later in this chapter. For details about the VLC, see **hken**(7). For details about the VLCi, see **cien**(7). For details about integrated Ethernet controllers, see **dgen**(7) and **inen**(7). For details about the VTRC, see **vitr**(7). For details about the VFC, see **pefn**(7).
- Have you made any changes to the hardware? Each network interface must be supported by the kernel. For a kernel called "dgux.aviion," the **sysdef /dgux.aviion** command shows which interfaces that kernel supports.

  If you build a VME-based controller such as **hken0** into the kernel, note whether the following message is displayed when the system is booting:

```
Configuring devices....
Could not configure hken0, skipping this device
```

If you see this message, the **/dev/hken***N* entries are not created. In this case, check the controller board and make sure it is jumpered correctly and properly seated. If you find any problems, correct them and reboot your system.

For other problems caused by changed hardware, seek further assistance from your DG support center.

- Are the cables and transceivers operating? If not, make the necessary adjustments or repairs. Is there a cable loose somewhere? If there is, reattach it.

  For more information about the correct installation of hardware, see *Ethernet / IEEE 802.3 Local Area Network Installation Guide* or *DG / Token Ring Local Area Network Installation Guide*.

Step 2: Check the Local Host

If there is no problem with the hardware, determine whether the problem exists on the local host. Consider the sample system introduced earlier in this section.

Assume you are logged in to the local host and that you are trying to connect to the remote host when a problem occurs. To determine whether the problem is with the local host, answer the following questions:

- Have you changed software?
- Is the local configuration information correct?
- Are the local interfaces running?
- Have you checked the connection to localhost?
- Can you connect to any host?
- Is the network subnetted? (Is routing the problem? Is the router operating correctly?)

- Is the Ethernet address correct?

Have You Changed Software?

Have you made any changes to the software on host A? If so, identify the changes, and then test communications without the changed piece(s).

Is the Local Configuration Information Correct?

Check the TCP/IP parameters on host A (**/etc/tcpip.params**) and other network files (for example, **/etc/hosts**). For more information about how this information should look, see Chapter 3.

Check the host database on host A to ensure that the correct address is available for the local interfaces and host B. A typical error message that would suggest a nonexistent entry in the host database is `Unknown Host`. Even if you do not get an `Unknown Host` error message, the problem may be an incorrect address for the remote host, which usually produces the error `Connection timed out` or `Destination unreachable.`

To check the host database, look in **/etc/hosts**. Use **ypcat** or **ypmatch** if you are running Network Information Service. For details about the NIS, see *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System*. Use the **nslookup**(1M) command if you are using the DNS.

Are the Local Interfaces Running?

Are your local interfaces configured, up, and running? Use **ifconfig** to find out. For more information about how to use **ifconfig**, see the section "Using the ifconfig Command" later in this chapter.

If **ifconfig** failed, did your protocol stack build? To find out, check the output of the **netinit** command in **/etc/log/netinit.log**. When **netinit**(1M) runs on a host with two network interfaces, its output looks something like this:

```
%OK open
%OK open
%OK link
%OK rename
%OK run
%OK open
%OK link
%OK rename
%OK run
```

Any log entries not flagged as `%OK` indicate that the stack did not build properly. If this is the case, examine the system configuration files to make sure that all Streams drivers and modules mentioned in the **/etc***hostname***.netinit** scripts are included in the kernel.

If in **/etc/log/netinit.log** the **run** commands show errors, examine the **run ifconfig** lines in **/etc***hostname***.netinit** scripts to ensure that the **ifconfig** parameters are correct. If they are not correct, modify the **/etc/tcpip.params** file to correct the **ifconfig** parameters.

Can You Connect to Localhost?

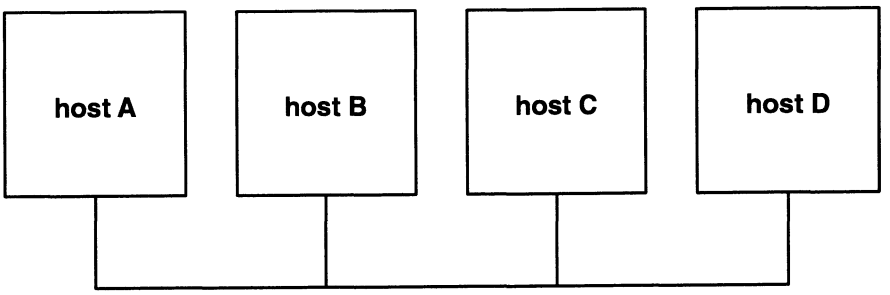
The reserved Internet address for **localhost** is **127.0.0.1**. Make sure that this address is defined in **/etc/hosts**. See Chapter 3 for details about **/etc/hosts**. If you are using the NIS, make sure this address is also defined in the NIS host database. See *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System* for details about the Network Information Service. If you are using the domain name system (DNS), make sure that you have an **A** record for this address in your **localhost.rev** file.

- Try connecting to **localhost**. For example, use **ping 127.0.0.1**. Connecting to **127.0.0.1** uses the loopback interface, which does not require a hardware interface.
- If you can connect to **localhost** and you cannot connect to a particular remote host, check to see if other machines can connect to that host. If other machines can connect to the host, then you might have a problem with your hardware. See the section "Checking the Hardware" earlier in this chapter for more information.
- If **ping 127.0.0.1** fails, refer to the earlier section "Are the Local Interfaces Running?" and follow the steps there.
- If your local interface is running, try **ping localhost**. If this command returns the error `can't find host localhost`, the mapping of localhost to address 127.0.0.1 has failed. Check the network databases, such as **/etc/hosts,** to make sure their content is correct.
- If **ping localhost** succeeds, try **telnet localhost**. If you cannot **telnet** to **localhost**, the server process is not running. Try restarting the server process (for example, **inetd**(1M)). Alternatively, restart the network or reboot the system.

Can You Connect to Any Host?

Is it that you cannot connect to a particular remote host (host B), or to *any* remote host?

- Try to connect to several hosts on the local LAN. For example, given the following configuration, try to connect to hosts C and D.

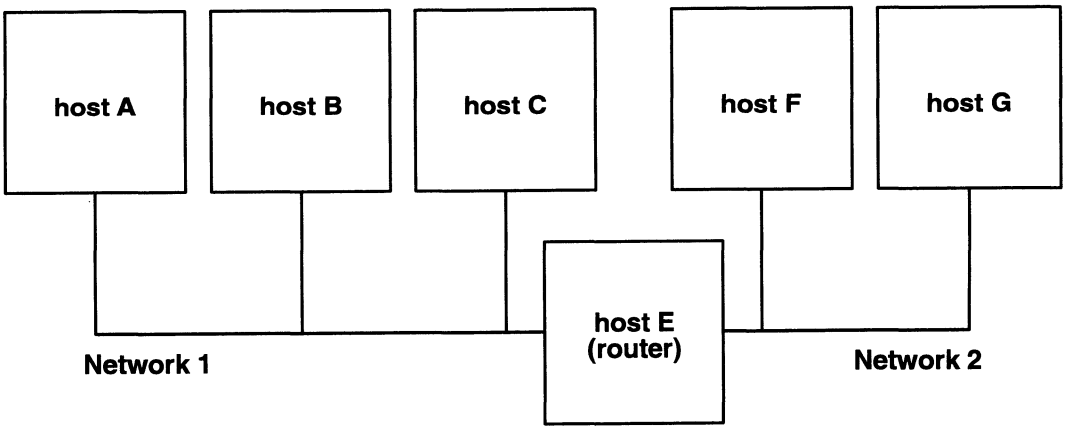


- If you cannot reach any other remote host, determine whether any other remote host can reach you. If a remote host (say, host C) on the local LAN cannot reach host A, but it can reach still another remote host (say, host B), the problem is on host A.
- If you determine that you can connect to other remote hosts (for example host C or host D), but not a particular host (for example host B), you can rule out problems with the local client programs.

  Verify that the remote host is up. Contact the system administrator of the machine to find out whether the host is up, or if any one else is having problems connecting to it.

Is the Remote Host on a Different Network or Subnetwork?

If the Internet address for the remote host (host B) is correct, the next step is to determine whether your LAN uses subnets and if the remote host in question is on the same or on a different physical network. Consider the following configuration:



- Check the network mask (also called subnet mask) on your local host (host A) and verify that it is the same as on all other hosts on your subnet (host B, host C). For example, use the **ifconfig** command to compare the network mask value, as follows:

ifconfig *interface*

where *interface* is the name of your Ethernet network interface.

- If the remote host is on the same subnet or physical network (host B or host C), routing is probably not the problem, because the routing entry for the local network is established when the networking device is configured. See the section "Step 3: Determine Whether the Problem Is with the Remote Host" for details.
- If the remote host is on a different subnet or physical network (host F or host G), the problem may be due to an incorrect route or to the behavior of the router (host E).

  Is Routing on the Local Host the Problem? It may be that the local route (on host A) to the remote physical network (Network 2) is not set up correctly. This problem usually results in an error message, such as the following:

  ```
  Network is unreachable
  ```

  To check the routes, use **netstat -r**. How to use **netstat** is covered in detail in the section "Using the netstat Command" later in this chapter.

  Set the routes with the **route** command, changing the routing parameters in **/etc/tcpip.params**, or with **routed**(1M).

  Is the Router Operating Correctly? It may be that the router (host E) is not operating correctly.

- Try to connect to another host on the same physical network as the remote host. For example, if you cannot connect to host F, try to connect to host G. If you can connect to the other host, you can assume the problem is not with either the local routes or the router.
- If you cannot connect to any other host on the remote physical network, check that the router (host E) is up and running with both interfaces active.
- If you cannot connect to a host through an IXE interface, the problem may be with the X.25 configuration.

Is the Ethernet Address for the Remote Host Correct?

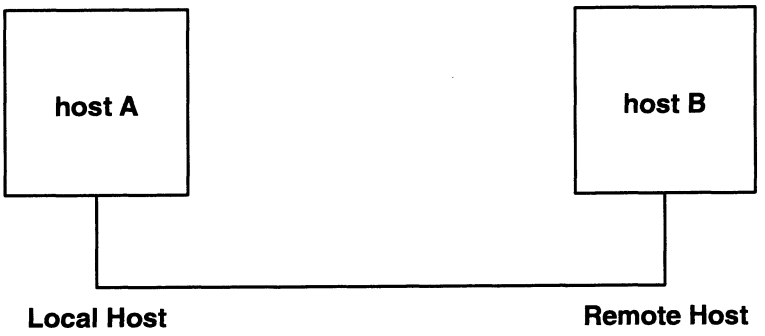
Use **arp -a** to display the Ethernet addresses. Is the Ethernet address for the remote host (host B) correct?

IMPORTANT: If you use subnets, the ARP table on the local host (host A) should have an entry for the router (host E). It may or may not have entries for remote hosts on a different physical network (for example, host F).

You may want to delete the ARP entry for the remote host or router and try the connection again.

Step 3: Check the Remote Host

Once again, consider the sample scenario introduced earlier in the chapter:



If the local host appears to be set up correctly, the remote host should be suspect. To check problems on the remote host, log in to the host from a terminal directly connected to it. Your approach to troubleshooting on the remote host should be similar to the approach you took on your host. The main difference is that you must verify that the network servers are working on the remote host.

Is the Remote Server Running?

Is the server for the network application that you are using up and running on the remote host? Run the application and try to connect to **localhost**. For example, try **telnet localhost**. Since the **localhost** interface uses no hardware, it works if the software is running correctly regardless of the network interface or of the cabling between systems.

Is the Remote Host's Setup Accurate?

If you can connect to **localhost**, the server is working correctly, so the problem may be with the configuration information, routing, or the network interface on the remote host. Have the remote host's system administrator verify the setup. Check routes with **netstat -r**, see "Using the netstat Command" later in this chapter for details. Check the network interface with **ifconfig**, see "Using the ifconfig Command" later in this chapter for details.

Is the Internet Address for the Local Host Correct?

Verify that the Internet address for the local host is correct on the remote host. Also, verify that the address of the host to which you want to connect is correct.

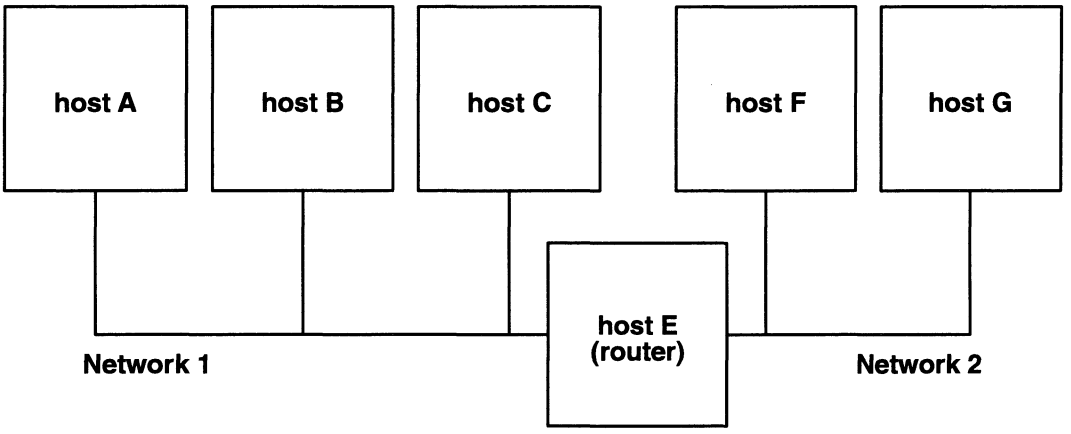
Look at **/etc/hosts** on the remote host. If the remote host uses NIS, use **ypcat**, or **ypmatch**. See *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System* for details about the NIS. If the remote host uses the DNS, use the **nslookup** command. For details about the DNS, see Chapter 5. Check **/etc/svcorder** to determine which databases the host is using.

Can the Remote Host Reach Other Hosts?

From the remote host, can you connect to the local host? If so, see "What Else?" later in this section. If not, see "Is the Ethernet Address for the Local Host Correct?" later in this section.

Is Routing on the Remote Host the Problem?

It may be that routing on the remote host is not set up correctly. Is the remote host to which you are trying to connect on the same or on a different physical network or subnet? Consider the following configuration:



If you cannot get to host A from host B, can you get to other hosts on the same physical network (for example, host C on Network 1)? If you cannot, have the administrator of the remote host and the local network verify that the routes are set up correctly. Since TCP/IP works bidirectionally, both local and remote hosts must know how to route to each other.

Verify that a route is set up on a remote host (for example, host F) for your network (Network 1). If the route is okay, check that the router (host E) is up and running. If the routing on the router is okay, try to connect to other hosts on the local network (for example, host C). If you cannot connect to any host on the local network from the remote host (host F on Network 2), you should suspect problems with the router (host E).

If you cannot get to host A from host F, can you get to other hosts on the same physical network (for example, host G on Network 2)? If

the network is subnetted, can a host on the other physical network (for example, host F) connect to any host on the same physical network as your host (for example, host B)? If not, there may be a problem with the routes between the remote host and the physical network on which your local host resides (Network 1).

Is the Ethernet Address for the Local Host Correct?

Use **arp -a** to display the Ethernet addresses. Is the Ethernet address for the local host (host A) correct? Note that you must issue **arp -a** on a host other than the local host to find the Ethernet address for the local host. If you issue the command on host A, it displays only the local ARP table for host A.

IMPORTANT: If you use subnets, the ARP table on the remote host (host B) should have an entry for the router (host E). It may or may not have entries for remote hosts on a different physical network (for example, host F).

You may want to delete the ARP entry for the local host or router and try the connection again.

Getting Outside Help

At this point, you have explored the most common sources of problems with networks, and have been unable to identify the cause of the problem. Seek further assistance from your DG support center.

Troubleshooting with Administrative Commands

The following section discusses ways to use administrative commands to troubleshoot at the low layers of TCP/IP. You can use these tools anywhere in the troubleshooting process.

Using the ifconfig Command

Generally, one of the first things you should do when you encounter a problem with TCP/IP is to use the **ifconfig** command to check the communication devices.

You can use the **ifconfig** command to assign an Internet address to a network interface, such as an Ethernet controller, configure parameters for that device, display information about the device, and stop and restart the device. To display the current configuration, use the following format:

ifconfig *interface*

where the *interface* in your machine is **cien***N*, **dgen***N*, **hken***N*, **inen***N*, **ixe***N*, **pefn***N*, or **vitr***N* and *N* is the device number. The command reports the current network mask and Internet address in each case. The output from this command should also list one or more of the following flags:

- The UP flag indicates that the device is started, attached, and running.
- The LOOPBACK flag indicates the interface is associated with a loopback network. This flag is only present for the **loop** interface.
- The BROADCAST flag indicates that the device has broadcast capability. For example, Ethernet is a broadcast medium; loopback is not. If the device has broadcast capability, the broadcast address is also reported. Broadcast on your host should match that of other hosts on the same physical network or subnet.
- The RUNNING flag indicates that the device is working properly; the flag is turned on when you start the interface and turned off when the system detects a controller failure.
- The NOARP flag indicates that the device does not support the address resolution protocol.
- The STARTED flag indicates that the device can send and receive data; you can turn the flag on by typing **ifconfig start**, and turn it off by typing **ifconfig stop**.
- The NOADDRMASKREPLY flag indicates that the device will not send replies to ICMP address mask request messages. This flag is not set by default.

Here is an example of output from **ifconfig cien0**:

```
cien0: 128.223.2.1 flags=443<UP,BROADCAST,RUNNING,STARTED>
            broadcast=128.223.2.255  netmask=0xffffff00  metric=0
```

If the message returned by **ifconfig** does not indicate that the interface is UP, BROADCAST, RUNNING, and STARTED, the device is not activated. The network mask for your machine should also be the same as that of other hosts on the same physical network or subnet. If the Internet address, broadcast address, or network mask is not correct, make the necessary corrections in the **/etc/tcpip.params** and **/etc/hosts** files.

Activating the Communication Controller

If the communication controller is not activated, initialize it by using the following format of the **ifconfig** command:

ifconfig *interface hostname*

or

ifconfig *interface host-internet-address* **broadcast** *address* **netmask** *netmask_value*

where, as before, *interface* can be **hken***N*, **cien***N*, **dgen***N*, **inen***N*, **ixe***N*, **pefn***N*, or **vitr***N* and you specify your *hostname* or *host-internet-address* as required. The **broadcast** *address* is the Internet broadcast address for the interface, if needed. The *netmask_value* is a 32-bit hexadecimal number that identifies which bits of the host's Internet address indicate the subnet number. After you have activated the device, check it as before.

If this procedure does not work, make sure **netinit** for your interface is running. If it is not, re–attach the interface using **admipinterface –o attach** or the **sysadm** procedure explained in Chapter 3, "Start or Stop, Attach or Detach an Interface."

Make sure that the device driver name for the communication controller is defined accurately in the system file as **hken**, **cien**, **dgen**, **inen**, **ixe**, **pefn**, or **vitr.** To check which devices are defined for the kernel, list the devices in the **/dev** directory as follows:

```
% ls -l /dev | more ↵
```

In **/etc/tcpip.params**, a number must be appended, starting with 0 (zero) for the first type of each interface, to the device name (**hken0**, **cien0**, **dgen0**, **hken0**, **inen0**, **ixe0**, **pefn0**, or **vitr0**) as appropriate. The following error message would appear if **ifconfig cien** *host* was used rather than **ifconfig cien0** *host*.

```
ifconfig: invalid subnet mask or broadcast address: ioctl
(SIOCGIFFLAGS)
```

If the **ifconfig** works after restarting manually, check all argument values in **/etc/tcpip.params**.

For more information, see **ifconfig**(1M) and **netinit**(1M).

Troubleshooting When the Network Hangs

When you investigate a network hang, find out how often the hang occurs. Can you determine the cause? Can you reproduce it? If you can reproduce a hang, check the output of **ifconfig** before and after the failure. For example, if your device is **cien0**, type the following commands:

```
% ifconfig cien0 ↵
<cause the hang>
% ifconfig cien0 ↵
```

Even if you cannot reproduce the hang, check the output from the **ifconfig** command after a hang has occurred. The values in the **ifconfig** flags may give you a clue about what is wrong.

Using the ping Command

Use the **ping** command to test whether hostname lookup succeeds, routing is in place, and the lower layers of TCP/IP are operational. The **ping** command sends an ICMP echo packet to the host you specify and expects the required ICMP response. If **ping** does not get the required response, it prints an error message such as `no answer from` *hostname*.

- To verify that IP is functioning without having to depend on a hardware interface, use **ping localhost**.
- To determine whether your host can communicate without using hostname lookup, use **ping** *IP_address*.
- If **ping** *hostname* returns the error `can't find host` *hostname*, hostname lookup has failed. Check network databases such as **/etc/hosts** for incorrect entries.
- If **ping** *remote_host* returns the error `Host is unreachable`, you may have a problem with the interface used to communicate with remote_host. Check the interface.
- If **ping** *remote_host* returns an error such as `Network is unreachable`, you may have a problem with routing. Check to be sure your routing is correct.
- If **ping** *remote_host* returns `no answer from` *remote_host*, then hostname lookup and routing has succeeded on your host, but there was no response from the remote host. In this case, the remote host is probably down. However, if your network is congested, the command may have timed out. You can specify a longer timeout a command similar to **ping** *hostname* **120**.

Using the traceroute Command

Use **traceroute** in conjunction with **ping** to isolate the break in a broken communication link. The following example shows **traceroute** output analyzing a link between a local interface **mirage–alt** and a remote host named **ftp.uu.net**:

```
# ping ftp.uu.net
ftp.uu.net is alive

# traceroute -m 12 ftp.uu.net
traceroute to ftp.uu.net (192.48.96.9), 12 hops max, 40 byte packets
 1  mirage-alt (128.998.10.27)  10 ms  10 ms  20 ms
 2  ncfddigw1 (128.998.1.204)  20 ms  10 ms  20 ms
 3  ncfddigw2 (128.998.250.8)  20 ms  10 ms  10 ms
 4  mass-alt3.us.dg.com (128.219.123.2)  90 ms  110 ms  160 ms
 5  sprint-gw.us.dg.com (128.219.137.29)  200 ms  290 ms  100 ms
 6  sl-dc-1-s2-384k.sprintlink.net (144.228.11.97) 170 ms 310 ms  330 ms
 7  icm-dc-2-f0.icp.net (144.228.1.36)  400 ms  390 ms  360 ms
 8  washington.dc.ws.net (191.41.177.248)  240 ms  150 ms  80 ms
 9  falls-church1.va.ws.net (139.39.128.6)  80 ms  370 ms  340 ms
10  ftp.uu.net (192.48.96.9)  460 ms  560 ms  430 ms
```

The following example illustrates **traceroute** output when the link is broken:

```
# ping ftp.uu.net
no answer from ftp.uu.net

# traceroute -m 12 ftp.uu.net
traceroute to ftp.uu.net (192.48.96.9), 12 hops max, 40 byte packets
 1  mirage-alt (128.998.10.27)  10 ms  10 ms  20 ms
 2  ncfddigw1 (128.998.1.204)  20 ms  10 ms  30 ms
 3  ncfddigw2 (128.998.250.8)  10 ms  10 ms  20 ms
 4  mass-alt3.us.dg.com (128.219.123.2)  90 ms  110 ms  160 ms
 5  sprint-gw.us.dg.com (128.219.137.29)  240 ms  320 ms  360 ms
 6  sl-dc-1-s2-384k.sprintlink.net (144.228.11.97)  400 ms  400 ms  400 ms
 7  icm-dc-2-f0.icp.net (144.228.1.36)  470 ms  400 ms  420 ms
 8  washington.dc.ws.net (191.41.177.248)  160 ms  80 ms  60 ms
 9  falls-church1.va.ws.net (139.39.128.6)  50 ms  80 ms  150 ms
10  * * *
11  * * *
12  * * *
```

This **traceroute** output indicates that the problem lies with host **falls–church1.va.ws.net**, rather than with the local **mirage–alt** interface.

Using the netstat Command

Use the **netstat** command to show the status for various network parameters. You can use **netstat** to see if the host "thinks" it is communicating or not.

Checking Incoming and Outgoing Packets

Run the **netstat -i** command to show incoming and outgoing packets and errors that the LAN controller sees. Then try to use **telnet** to access a remote machine. Even if the connection never

appears to be made, run the **netstat -i** command a few times to see if the incoming packets and outgoing packets change on your host.

For example, suppose on *local_hostname*, you run **netstat -i** to obtain output like the following:

```
Name  Mtu   Network       Address         Ipkts   Ierrs Opkts   Oerrs Collis
cien1 1500  128.223.1.0   sales           1037544 3     910644  0     46406
cien0 1500  128.223.2.0   sales-alt       311546  0     134806  0     3809
loop0 4136  127.0.0.0     localhost       398     0     398     0     0
```

Then, suppose you successfully **telnet** to *third_hostname* (through **cien0**). If you log off, and run **netstat -i** again, you will find that the statistics may have changed, as follows:

```
Name  Mtu   Network       Address         Ipkts   Ierrs Opkts   Oerrs Collis
cien1 1500  128.223.1.0   sales           1038912 3     911940  0     46523
cien0 1500  128.223.2.0   sales-alt       311784  0     134878  0     3810
loop0 4136  127.0.0.0     localhost       398     0     398     0     0
```

Another way to see the traffic involving your host while running an application program is to run **netstat 10**. This displays the number of packets transmitted and received every 10 seconds:

```
    input   (cien1)     output           input   (Total)    output
packets errs  packets errs  colls packets errs  packets errs  colls
312165  0     134962  0     3811  1353318 3     1049158 0     50606
19      0     1       0     0     49      0     31      0     0
16      0     16      0     0     93      0     88      0     0
11      0     11      0     0     132     0     134     0     12
18      0     19      0     9     108     0     74      0     17
4       0     94      0     24    182     0     171     0     24
12      0     81      0     9     159     0     146     0     9
22      0     2       0     0     63      0     41      0     0
5       0     15      0     0     54      0     55      0     10
14      0     10      0     0     116     0     98      0     2
27      0     11      0     8     125     0     89      0     9
12      0     0       0     0     68      0     59      0     0
25      0     1       0     0     48      0     36      0     8
21      0     3       0     8     71      0     50      0     8
5       0     4       0     0     61      0     93      0     0
```

The first line shows a cumulative total since boot time. If everything is going well with your network, do not be concerned with the number of errors, but if there are problems, take note of the `errs` and `colls`. Note that moderate numbers of collisions are normal on CSMA/CD networks such as Ethernet.

Checking the Routing Tables

Run **netstat -r** to check the routing tables. If you want to see the Internet numbers rather than the names defined in **/etc/hosts** and

/etc/networks, use the **-n** option. The list below shows sample output from **netstat -n -r**.

```
Routing tables
Destination      Gateway       Flags    Refcnt Use         Interface
127.0.0.1        127.0.0.1     U        0      1038        loop0
128.223.1        128.223.1.10  U        20     32840       cien1
128.223.2        128.223.2.1   U        20     506194      cien0
128.223.3.5      128.223.1.1   UGH      0      856         cien1
```

Local interfaces, such as **loop0**, **cien1**, and **cien0** in this example, have only the "U" flag present. If this is not the case, use ifconfig and check the /etc/tcpip.params file to be sure the interface is defined correctly. The "G" flag indicates an interface is a gateway.

Checking Network Statistics

Using **netstat -s** shows software statistics about the network. It shows problems at the IP, ICMP, UDP, and TCP protocol layers. Here is a partial example of output:

```
udp:
        0 bad header checksums
        0 incomplete headers
        0 bad data length fields
        1484 datagrams received
        38 datagrams received for non-existent port
        1371 transmit datagrams requested
        0 input datagrams dropped because of flow control


tcp:
        328 packets sent
                120 data packets (5466 bytes)
                0 data packets (0 bytes) retransmitted
                168 ack-only packets (86 delayed)
                0 URG only packets
                0 window probe packets
                0 window update packets
                40 control packets
        302 packets received
                143 acks (for 5491 bytes)
                35 duplicate acks
                0 acks for unsent data
                161 packets (29503 bytes) received in-sequence
                0 completely duplicate packets (0 bytes)
                0 packets with some dup. data (0 bytes duped)
                0 out-of-order packets (0 bytes)
                0 packets (0 bytes) of data after window
```

```
        0 window probes
        10 window update packets
        0 packets received after close
        0 discarded for bad checksums
        0 discarded for bad header offset fields
        0 discarded because packet too short
15 connection requests
10 connection accepts
25 connections established (including accepts)
60 connections closed (including 0 drops)
0 embryonic connections dropped
132 segments updated rtt (of 157 attempts)
0 retransmit timeouts
        0 connections dropped by rexmit timeout
0 persist timeouts
0 keepalive timeouts
        0 keepalive probes sent
        0 connections dropped by keepalive
```

When you use **netstat -s**, are there bad checksums or other errors? If so, be sure to point these out to your service representative. Networks may run fine with some ICMP errors, but they may be an indication of a problem.

Checking Network Connections

Using **netstat -a** shows you the status of network connections. The following example shows the output:

```
Active connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp        0      0  sales-alt-login       dg1-1021               ESTABLISHED
tcp        0      0  sales-alt-login       sun1-1022              ESTABLISHED
tcp        0      0  sales-telnet          sys02-2977             ESTABLISHED
tcp        0      0  sales-alt-1048        dg1-XV11               ESTABLISHED
tcp        0      0  sales-1043            sys56-nntp             ESTABLISHED
tcp        0      0  *-1034                *-*                    LISTEN
tcp        0      0  *-53                  *-*                    LISTEN
tcp        0      0  *-time                *-*                    LISTEN
tcp        0      0  *-daytime             *-*                    LISTEN
tcp        0      0  *-chargen             *-*                    LISTEN
tcp        0      0  *-discard             *-*                    LISTEN
tcp        0      0  *-echo                *-*                    LISTEN
tcp        0      0  *-exec                *-*                    LISTEN
tcp        0      0  *-login               *-*                    LISTEN
tcp        0      0  *-shell               *-*                    LISTEN
tcp        0      0  *-telnet              *-*                    LISTEN
tcp        0      0  *-ftp                 *-*                    LISTEN
```

```
tcp        0      0  *-smtp               *-*                    LISTEN
tcp        0      0  *-656                *-*                    LISTEN
tcp        0      0  *-649                *-*                    LISTEN
tcp        0      0  *-629                *-*                    LISTEN
tcp        0      0  *-1024               *-*                    LISTEN
tcp        0      0  *-994                *-*                    LISTEN
tcp        0      0  *-sunrpc             *-*                    LISTEN
udp        0      0  sales-alt-53         *-*
udp        0      0  sales-53             *-*
udp        0      0  localhost-53         *-*
udp        0      0  *-53                 *-*
udp        0      0  *-echo               *-*
udp        0      0  *-tftp               *-*
udp        0      0  *-715                *-*
udp        0      0  *-route              *-*
udp        0      0  *-syslog             *-*
udp        0      0  *-993                *-*
udp        0      0  *-sunrpc             *-*
```

The states of `telnetd`, `ftpd`, and other network daemons should be `LISTEN` when they are not connected, and `ESTABLISHED` when a connection is made. They go through other states, but `LISTEN` and `ESTABLISHED` are the most common. Most of the `udp` connections in the example above are from the DG/UX™ ONC™/NFS® product.

Using the arp Command

Unless a host on a TCP/IP network is an IXE host, it has an Internet address and an Ethernet, token ring, or FDDI address. The ARP/RARP (Address Resolution Protocol/Reverse ARP) protocols set up a table that contain Internet and Ethernet, token ring, or FDDI address pairs. You can use the **arp** command to display or change the values.

If your host cannot communicate with a machine on the local network, use **arp -a** to find out what your host thinks the Ethernet, token ring, or FDDI address of the remote host is. Use **arp -a** on the remote host to display its ARP table and see what it thinks your address pair is. Are the address pairs defined as you expect?

Here is a sample ARP table produced with **arp -a**:

```
   Current ARP table entries for device inen0

Hostname    Internet Address  Hardware Address          Status
sales       128.223.2.1       00:00:77:1a:00:6a         temporary
sun1        128.223.2.3       08:00:20:00:a7:5d         temporary
```

To change an incorrect or questionable ARP entry, remove it and add a new entry. You can add a new entry by hand if the machine has the **arp** command, or you can start a connection by using **ping** or some other application to let the ARP protocol get the values.

Remove a host's ARP entry with the following command lines:

```
% su ↵
# arp -d host ↵
```

where *host* is the hostname of the system as specified in **/etc/hosts**.

To set an Internet/Ethernet address pair,become the superuser and use the **arp -s** command, as follows.

```
# arp -i cien0 -s sales 00:00:77:1a:00:6a ↵
```

In all previous examples, the system named **sales** had two network interfaces: **cien0** and **cien1**. The command line above would associate the hostname **sales** with the address **00:00:77:1a:00:6a** through **cien0**.

For more information, see **arp**(1M).

Debugging Low Level Protocols

If you use a LAN analyzer (for example, the Network General Sniffer) to debug low level protocol problems, you can send your support center the output on a diskette (but make sure your support center has the hardware needed to read your diskette, and also send hardcopy output).

You can also use **tcpdump**(1M) to analyze low level protocols, and **nfc**(1M) to convert between **tcpdump** binary and Network General Sniffer formats. With **tcpdump**(1M), you can display network activity by protocol or between specified hosts. The following examples illustrate how to display (1) Network Time Protocol (ntp) requests and (2) the ICMP echo path between two specified hosts:

```
# tcpdump -q -c 5 ntp
tcpdump: listening on inen0
15:46:14.25 ip[sage->brewery] udp[ntp->ntp:48] ntp[v3 sym_act strat 4 poll 6 prec 250]
15:46:14.26 ip[brewery->sage] udp[ntp->ntp:48] ntp[v3 sym_pas strat 3 poll 6 prec 250]
15:46:21.55 ip[melba->phantom] udp[ntp->ntp:48] ntp[v3 client strat 0 poll 6 prec 250]
15:46:31.69 ip[honeydo->brewery] udp[ntp->ntp:48] ntp[v3 sym_act strat 4 poll 7 prec 250]
15:46:31.70 ip[brewery->honeydo] udp[ntp->ntp:48] ntp[v3 sym_pas strat 3 poll 7 prec 250]
9789 packets received by filter
0 packets dropped by kernel
```

```
# tcpdump -e -p -S -v 'host capt-crunch and host honeydo'
tcpdump: listening on inen0
14:42:43.43
        ether: capt-crunch->8:0:1b:18:1f:a0,ip,98
        ip: capt-crunch->honeydo;(ttl 255,id 10612)
        icmp: echo request
14:42:43.44
        ether: 8:0:1b:18:1f:a0->capt-crunch,ip,98
        ip: honeydo->capt-crunch;(ttl 255,id 11722)
        icmp: echo reply
```

Interpreting Error Messages

The following sections describe how to interpret error messages. They also present ways to troubleshoot the problems.

connect: Network is unreachable

When using **telnet** or **ftp**, this error message may appear because a route to the machine to which you are trying to connect is not available or not setup. Check the routing table with **netstat -r** or **netstat -nr**. Add a route with the **route** command. See Chapter 3 for details about routing.

connect: Connection refused

This error message may appear because the daemon process is not up on the remote side. Start the daemon. See the appropriate manual page, for example, **inetd**(1M), for details.

hostname: Unknown host

This error may appear when the hostname for a machine to which you are trying to connect is not present in the Network databases. Check files such as **/etc/hosts** to see if this is the case. If so, add the host to those files with **sysadm**.

Interpreting Error Messages in errno.h

When you get an error message with which you are unfamiliar, find the corresponding text in **errno.h**. For example, use the following command:

```
% view /usr/include/sys/errno.h ↵
```

Search the file for the error number that was displayed on your screen. When the error number returned is preceded by a 0 (zero),

the number is in octal. Translate this number to decimal, because error numbers in the **errno.h** file are in decimal. The text of the error message appears after the line that contains the error number.

For example, when you use the **view** command to look at the **errno.h** file, you find this text associated with error number 70:

```
        */
#define ECOMM           70
        /*
         * Communication error on send.
         */
```

When reporting an error to your support center, be sure to specify which process it is coming from. For example, is the error from the DG/UX system, **telnet**, the **telnet** daemon, or **sendmail**?

Troubleshooting at the High Layer

The following sections describe known problems and their solutions for software at the high layer of TCP/IP.

Using telnet Options

One way to investigate problems with **telnet** is to turn option processing on. For example, use the following sequence of commands:

```
% telnet ↵
telnet>options ↵
Will show option processing.
telnet>open remote_hostname ↵
```

With the **options** set, **telnet** displays option-negotiation according to the TELNET protocol. This verbose output may help you pinpoint a breakdown in communication between the local and remote hosts. See *Using TCP/IP on the DG/UX™ System* or *RFC 854 (Telnet Protocol)* for details.

Resolving Problems with Pseudoterminals

The following message indicates a pseudoterminal problem.

```
All Network ports in use.
```

The message may also mean that the kernel on the destination host does not have pseudoterminals configured. If the host is running the DG/UX system, check the **/dev** directory for **/dev/ttyp***N*, where *N* is an integer. If the entries are there, the pseudoterminals are configured.

If the entries are not there, make sure the pseudoterminals were set up correctly. Entries for **pts()** (pseudoterminal server) and **ptc()** (pseudoterminal client) should be in the system file.

This message may also mean that all the pseudoterminals are in use. If this is true, try connecting again later.

Resolving Problems with ftp

If you have determined that your problem is with **ftp**, make sure you are not transferring a file to a directory that is too small. If you cannot transfer a file, try to transfer another file of the same size. Also, try to send the file that is giving you trouble to a third machine on your local network.

One way to investigate problems with **ftp** is to show activity as the file moves from host to host. You do this by turning the hash marks on. For example, use the following sequence of commands:

```
% ftp ↵
FTP user (DG/UX TCP/IP Release x.x.x) ready.
ftp>hash ↵
Hash mark printing on (2048 bytes/hash mark).
```

ftp>**open** *remote_hostname* ↵

With **hash** set, **ftp** displays checkpoints while a file is transferred.

Using ftp's Debug Option

It may also be useful to turn the debug option on. For example, use the following sequence of commands:

```
% ftp ↵
FTP user (DG/UX TCP/IP Release x.x.x) ready.
ftp>debug ↵
Debugging on (debug=1).
ftp>hash ↵
Hash mark printing on (2048 bytes/hash mark).
```

```
ftp>open remote_hostname ↵
```

With **debug** set, the **ftp** command shows you the FTP protocol commands as they execute.

Resolving Problems with sendmail

The following sections suggest ways to resolve problems with **sendmail**. After the list, specific problems that have been encountered by previous users are explored and their resolutions are presented.

Remember that you normally do not invoke **sendmail** directly. A mailer program should be used to interface with **sendmail**. We suggest that you use **mailx** to send and receive mail. If the problem is with **sendmail** and not your mail delivery program, perform the following tests:

Checking mail's User ID

Use the **grep** command to find `Ou` (uppercase O, lowercase u) in the **sendmail** configuration file, **/etc/sendmail.cf**. Look for **mail** in **/etc/passwd**. The user ID after `Ou` in the **sendmail** configuration file and the user ID in the **mail** entry in the password file should be the same (for example, 8). If they do not match, make them the same. Alternatively, `Ou` should be set to **mail**. Be sure that the user ID for **mail** in **/etc/passwd** is unique.

Showing the Conversation Between Hosts

Send a message showing the conversation between the two machines with the **-v** option.

```
% /usr/bin/sendmail -v user@host ↵
This is a test message.
. ↵
```

Choose the *user* and *host* to which you want to send the message. Verify that the output shows a connection to the destination host or a mail relay host, and that the mail text (DATA) is sent.

Printing the sendmail Queue

Use the command **/usr/bin/mailq** to print the **sendmail** queue. The * (asterisk) in the output display depicts messages that are currently active. Messages are held in **/var/spool/mqueue** for a

short time until they are delivered. Completed messages go to **/usr/mail/***username*. Only the superuser may look at these files.

Forcing the sendmail Queue and Recording Any Errors

Messages that did not get mailed to the remote machine for any reason stay in **/var/spool/mqueue** for the time-out period (often three days) specified in the configuration file before being mailed back to the originator. Every once in a while, the **smtp** daemon again attempts to mail the message. (Thirty minutes is usually defined on your **smtp** invocation line.) However, if you want to force the queue and see the errors, delete the **n*** and **x*** files and remail the messages in the queue. The following set of commands processes the saved messages in **/var/spool/mqueue** verbosely.

```
% su ↵
# cd /usr/spool/mqueue ↵
# rm n* x* ↵
# /usr/bin/sendmail -v -q ↵
```

Verify that the output shows a connection to the destination host or a mail relay host, and that the mail text (DATA) is sent.

Using telnet to Check the sendmail Port

Use **telnet** to look at the **sendmail** port. Do this to verify a host's **smtp** daemon is active and performing the functions needed to receive mail.

The following sequence of commands shows a **telnet** session between hosts named **durham** and **raleigh** through port 25. Assume that the **smtp** port (defined in **/etc/services)** is 25, as it is with DG software. Also assume that user **smith** is defined in **/etc/passwd** on the host **durham**, from which the message is sent. The four-letter commands are defined in *RFC 821 (Simple Mail Transfer Protocol)*.

```
% telnet raleigh 25 ↵
Trying...
Connected to raleigh.
Escape sequence: ^]
220 raleigh Sendmail 5.4/rtp-s03 ready at Tue, 4 Jun
1991 12:35:34 -0400
expn smith ↵
250 John Smith <smith>
help ↵
214-Commands:
214-    HELO    MAIL    RCPT    DATA    RSET
214-    NOOP    QUIT    HELP    VRFY
214-For more info use "HELP <topic> ".
214 End of HELP info
vrfy smith ↵
250 John Smith <smith>
vrfy dummy ↵
550 dummy... User unknown
mail from: smith ↵
250 smith... Sender ok
rcpt to: jones ↵
250 jones... Recipient ok
data ↵
354 Enter mail, end with "." on a line by itself
Hello Greg. This is a test of using telnet to check the
sendmail ↵
port. Hope you enjoyed your trip. ↵
. ↵
250 Mail accepted
quit ↵
221 raleigh closing connection
Connection closed by foreign host.
Connection closed.
%
```

System responses that start with "250" are fine. Other numbers may indicate where the problem lies.

Invoking smtp by Hand Requires Logname Set to Null

If for some reason you must restart the **smtpd** on your TCP/IP system (for example, when you change your **sendmail** configuration file), be sure to set the **LOGNAME** environment variable to null or the mail delivered to your local machine might look as though it came from the wrong person.

Normally, when you bring your system to multiuser mode, **smtp** is started in the **rc** script by the **init** program. The **init** program

usually does not have a **LOGNAME** value assigned in the environment, so there is no problem.

If you must invoke **smtp** by hand, use the following command lines. With the first command line, you become root. With the second, you set the environment variable to null and execute **smtp** with the options you require.

```
% su ↵
# env "LOGNAME=" smtp -q30m ↵
```

In the second line, **-q30m** means to flush the **sendmail** queue every 30 minutes.

Is sendmail Using Domain Names Correctly?

TCP/IP for AViiON Systems provides one prototype sendmail configuration file: **/etc/sendmail.cf.proto**. The name "proto" implies that it must be edited for your particular system. Chapter 4 explains the rules for the **sendmail** configuration files. If you are not able to send or reply to certain hosts using the prototype file, edit it to fit your site.

The **sendmail** program may have problems with fully or partially qualified hostnames. If the DNS is your primary method of host/address mapping, **sendmail** processes both fully qualified hostnames (*host.domain*) or simple hostnames (*host*) with the local domain implied. If the NIS or **/etc/hosts** is your primary method of host/address mapping, fully qualified names does not work unless you have set up the NIS to match both *host* and *host.domain*. Thus, if your NIS has only *host* entries, and **sendmail** tries to look up *host.domain*, the lookup will fail. It is advisable, particularly if your network connects to the Internet, to be able to lookup either simple or fully qualified names.

Another problem can occur because the hostname macro (**$w**) in **sendmail.cf** is set to recognize simple but not fully qualified hostnames. There should be rules in ruleset 0 of **sendmail.cf** to handle incoming addresses for both *user@host* and *user@host.domain*, mapping them to the local mailer. Check the notes in the prototype file and be sure that you are handling hostnames correctly.

Troubleshooting SLIP

This section describes how to diagnose some SLIP problems that you might encounter. The first section lists some known limitations in

this SLIP release. The next section tells how to use the **slipd** debug mode. The last three sections describe how to use the **ping**, **netstat**, and **cu** commands to gather information for troubleshooting.

This section does not cover any problems with your modem configuration. If you are having modem problems, make sure your modem is set up as described earlier in this chapter. Also, make sure your cable is connected to the correct port and that no connections are loose.

Your modem should support reliable connection (an MNP or LAPM error-free connection with another MNP or LAPM modem) and hardware flow control to maximize throughput. If you are using a reliable connection, you should also use hardware flow control or the data flow might overrun your modem. You should not use software flow control.

Known Limitations

This SLIP release does not support the following operations:

Operator-assisted calls

Modem callback

Debugging slip

You operate **slipd** in debug mode by including the **–d** flag on the command line. When you turn on debug mode, **slipd** runs in the foreground and retains control of your terminal.

When **slipd** runs in debug mode, it display each step the command is taking as it attempts to make a remote connection. These steps include accessing the SLIP databases and the send and receive string communication with the remote system. If you are having trouble making a SLIP connection, debug mode can show you where the connection process is failing.

The following page illustrates the debug information displayed while **slipd** is making a connection:

```
% slipd -d -b19200 -ltty00 sl_server ↵
Slipd (DG/UX TCP/IP Release 5.4R3.00) ready - Auto dial mode to 'sl-server'
  Looking for system 'sl-server' in /udd/frazier/.slipdialinfo
  Establishing remote connection, please be patient.
  -- Sending 'ATDT5551234' to modem --
  -- Expecting 'CONNECT' --
ATDT5551234  Timeout 0, waiting for expect string
  Timeout 1, waiting for expect string
  Timeout 2, waiting for expect string
  Timeout 3, waiting for expect string
  Timeout 4, waiting for expect string
  Timeout 5, waiting for expect string
  Timeout 6, waiting for expect string
  Timeout 7, waiting for expect string
  Timeout 8, waiting for expect string
CONNECT
  -- Index=1, Matched='CONNECT' --
  -- Sending '\w\r' to modem --
  -- Expecting 'gin:' --
14400/REL - MNP
server
DG/UX Operating System Release 5.4R3.00
login:
  -- Index=3, Matched='login:' --
  -- Sending 'sliplogin' to modem --
  -- Expecting 'word:' --
sliplogin
  Timeout 0, waiting for expect string
Password:
  -- Index=5, Matched='Password:' --
  -- Sending 'slippasswd' to modem --
  -- Expecting '$' --
DG/UX Release 5.4R3.00 AViiON
sl-server
Copyright (c) Data General Corporation, 1984-1993
All Rights Reserved
#============================================================================#
#      Welcome to the Data General SLIP server running DG/UX 5.4R3.00        #
#                                                                            #
#   ####   #                ####   ######  #####   #    #  ######  #####  #
#  #       #               #       #       #    #  #    #  #       #    # #
#   ####   #       #####    ####   #####   #    #  #    #  #####   #    # #
#       #  #                   #  #       #####   #    #  #       #####  #
#  #    #  #               #    #  #       #   #    #  #  #       #   #  #
#   ####   ######           ####   ######  #    #    ##    ######  #    # #
#                                                                            #
#  WARNING:  Access to and use of this system  and  the  DGC  network  is #
#            RESTRICTED TO AUTHORIZED INDIVIDUALS under the provisions of #
#            the "DGC  Information  Assets  Security  Policy"  for  the #
#            protection of  DGC proprietary and confidential information. #
#============================================================================#
  -- Index=7, Matched='$' --
  -- Sending 'exec\sslipd' to modem --
  -- Expecting 'SLIP' --
exec slipd
  -- Index=9, Matched='SUNSLIP' --
Slipd READY - Connected to 'sl-server'
```

In this example, SLIP made the connection with no problem. However, since **slipd** is running in debug mode, it is a foreground process and retains control of the terminal. You can terminate the **slipd** session by entering a SIGINT from the keyboard, (usually a ^C).

Checking Header Compression

Once you have made a SLIP connection, you can use the **ping** command to test for problems with header compression. If you are using header compression, both sides of the link must support compression or applications such as **telnet** will hang.

If you make a **telnet** connection over a SLIP link and the application hangs, try to **ping** the remote system. For example, enter the following:

```
% ping host ↵
```

Your system should respond:

```
host is alive
```

If you get this response and **telnet** connections still hang, turn off header compression in the associated **slipuser** entry, re-establish the SLIP link, and try **telnet** again.

Verifying That the Connection Is Up

Once you have made a SLIP connection, you can use the **netstat** command to verify that your connection is up and has routes available.

Use **netstat –i** to verify that your SLIP interface is up and running without errors. The following example shows partial output from **netstat –i** on a newly established SLIP connection:

```
Name  Mtu   Network   Address       Ipkts  Ierrs Opkts  Oerrs Collis
slip0 576   slip-lan  128.223.5.17  0      0     0      0     0
```

The name of the interface is `slip0`. It is up and running properly. If `slip0` did not appear in the list of interfaces, then you would know your SLIP link is not up. If there were entries in the `Ierrs` or `Oerrs` columns for `slip0`, then you would know that the interface is not functioning correctly.

Use **netstat –r** to verify that your SLIP interface is up and has routes available. The following example shows partial output from **netstat –r** on an established SLIP connection:

```
Destination     Gateway         Flags    Refcnt Use    Interface
128.223.5.17    128.223.5.17    UH       0      0      slip0
128.223.5.4     128.223.5.17    UH       0      0      slip0
```

`slip0` is up and has routes available. Again, if `slip0` did not appear in the list of interfaces, then you would know your SLIP link is not up.

See the **netstat**(1M) manual page for additional information.

Checking Your Modem

You can use the **cu** command to check the functionality of your modem. With **cu,** you can call-out on your modem and log in on a remote system to verify connectability.

In the following example, **cu** is used to access a remote system:

```
harpo% cu -ltty00 ↵
Connected
at ↵
OK
atdt1234 ↵
CONNECT 9600/REL - MNP
↵
chico
DG/UX Release 5.4R3.00  (chico)
login: smithj ↵
Password: ↵
DG/UX Release 5.4R3.00 AViiON
chico
Copyright (c) Data General Corporation, 1984-1992

All Rights Reserved

#============================================================================#
#                              WARNING                                       #
#                                                                            #
# ACCESS TO AND USE OF THIS SYSTEM IS RESTRICTED TO AUTHORIZED INDIVIDUALS!  #
#                                                                            #
#                  Data General AViiON DG/UX System                          #
#                              Chico                                         #
#                           "Why a duck?"                                    #
#                                                                            #
#============================================================================#

Thu Aug 13 14:42:27 EDT 1992
You have mail.
chico% exit ↵
chico% logout

Lost Carrier

Disconnected
harpo%
```

Note that this example uses the Hayes Smartmodem AT command set to communicate with the modem. If you are not using the Hayes commands, substitute the equivalent commands for your system.

This session verifies the following things about your system:

- Port is available
- Cable is correctly attached
- Modem can communicate
- Baud rates match on both sides of the link
- Remote system can be accessed through the modem link

For more information on the setup and use of **cu**(1), see *Managing Modems and UUCP on the DG/UX™ System* (069–000698).

Troubleshooting bootpd

If **bootpd** appears to be ignoring BOOTP requests, restart it with the debug switch, **bootpd –d**. The **bootpd** daemon is started by **inetd**. To add the **–d** switch, use the **sysadm** procedures described in Chapter 3, "Modify a Daemon." (Optionally, you can edit `/etc/inetd.conf`.)

With the **–d** option, **bootpd** reports requests and actions via the **syslogd**(1M) mechanism. The reported information includes: the hardware addresses of BOOTP clients making requests, whether **bootpd** located the addresses, and the BOOTP version. The reported information is sent to the destination explained in **syslogd**(1M).

Troubleshooting with the Results from rwhod

Every three minutes, **rwhod** broadcasts a packet that contains information about the system (for example, system name and length of time the system has been up) and the users. Every other machine on the network that is listening (and also running **rwhod**) is able to pick up that information and show it to the users by the **rwho** and **ruptime** commands.

This information about the machines is stored in **/var/spool/rwho** in files of the form **whod.***host*. You can view these files with the **od** command, as follows:

od -c whod.*host* **| more**

Another way to look at the contents is as follows:

strings whod.*host*

This command prints all printable characters, so it does not give you an accurate picture, but it does show the contents of the file.

The first 60 bytes of this structure is a header that contains the machine name. After that are groups of 24 bytes that give information about each user.

Generally, you see only **ruptime** and **rwho** values of machines that are on your local network. If you do not see all the information you expect about a machine, check the length of the **whod.***host* packet.

IMPORTANT: By default the **rwhod** is not enabled when TCP/IP for AViiON Systems is loaded onto your system. To use **rwhod**, edit the **/etc/tcpip.params** file and uncomment the **rwhod** line in the section of the file telling which daemons to start. You should also note that **rwhod** sends information only about the first 58 users on a system. This is the most information that will fit into the network buffer.

Gathering Data for Outside Help

Whenever a problem occurs, the tools that you need to solve this problem are a LAN analyzer, terminals on *local_hostname*, and terminals on *remote_hostname*. If you cannot remedy your problem, gather data about the following points. Your notes become valuable information for your support center as they attempt to help you solve your problem.

1. Make sure you can duplicate the original problem.
2. Set up a shell script to collect the data. It should run all the **netstat** commands and show the ARP table. Save the output of these commands in a file. For example, suppose your *script* contained the following commands.

   netstat -a
   netstat -i
   netstat -s
   netstat -r
   arp -a
   ping *local_hostname*
   ping *remote_hostname*
   ifconfig *device*

   You could place the output of *script* in a *file* as follows:

   % *script* > *file* ↵

3. Another script or terminal should run the **netstat** program with a numerical argument to count the packets the machine thinks are going into and out from the interface. In the example that follows, information is captured every 10 seconds and put in a file named **netstat10out**.

   % **netstat 10 > netstat10out** ↵

4. Depending on the problem, you may need to make an Ethernet trace of traffic during the problem. See the section "Using an Ethernet LAN Analyzer" earlier in this chapter. You may want to collect all traffic between the two hosts, only the ARP traffic, or all traffic on the network.

   When you are ready to gather the first set of data, run the statistics script, turn on the Analyzer, and run the **netstat 10** program. Then reproduce the problem. Finally, once the hang or time-out has occurred, run the statistics script a final time.

   Gather the second set of data in the same manner, except rather than reproducing the problem, make a connection to a machine that works so that you can contrast that output with the previous output.

Before You File a Software Trouble Report

Here is some additional information that should be useful when you gather information to file a Software Trouble Report (STR) or to contact your support center.

- Prepare a **readme** file explaining what is happening, what you've done, and what you're sending.
- Find a way to reproduce the problem. Give the command line you use to reproduce the problem. Track how often the problem occurs. If you have a problem using a command with a particular file, send the smallest portion of the file that causes the problem.
- If you encounter this problem at a particular time of day, for example, when the network traffic becomes heavy or when a particular daily job runs, describe the circumstances as clearly as possible.
- If there are problems connecting to a remote host, describe the physical layout of the network. Send in a diagram of the network. Are they all DG machines or are they from other vendors? If they are from other vendors, state the name, operating system, and revisions of software if you know them.
- If you have a problem specifically with **sendmail**, and your intended recipient gets something, even if incomplete, include the received message.

- If there is a core dump, send it with the corresponding executable files.
- Send any logfiles in **/usr/adm/streams**.
- Send your service representative a copy of all of the configuration files mentioned in the *Release Notice*, such as:

  The system file

  /etc/hosts, **/etc/networks**, **/etc/inittab**, **/etc/passwd**, **/etc/tcpip.params**

  /etc/sendmail.cf, **/usr/spool/mqueue**, **/etc/svcorder**

  If you use NIS, the system may not consult **/etc/hosts** or **/etc/networks**, and so they may not be useful to send. Instead, use **ypcat** to extract values in the NIS database for **hosts** and **networks**. In each case, redirect the output to a file. You may also want to use **ypcat** to extract values for **passwd**, **group**, **netgroup**, **services**, and **protocols**. For more information, see *Managing ONC™/NFS® and Its Facilities on the DG/UX™ System*.

  If you use the DNS, use **nslookup**'s interactive command **ls** *domain* to list the information available for a domain, redirecting the output to a file.

- Write down all error messages and whatever else appears on your console and the master console when the error occurs.
- Send in copies of any data files or log files that may help your support representative understand or reproduce the problem. Include any output from **netstat**.
- Include all information requested in the STR section of the *Release Notice*.

End of Chapter

A SNMP MIB Objects

This appendix describes the Management Information Base (MIB) maintained by Data General for SNMP. Figures A–1 and A–2 show the hierarchical arrangement of the MIB tree. The object names shown in the hierarchy are valid arguments with the SNMP commands. An object name references all the objects under it. Thus, argument `dataGeneral` references the entire hierarchy, while argument `dguxMgtLp` references the printer hierarchy.

The remainder of the chapter describes the low–level objects (those that contain data) in the branches `dgHwDeviceInfo` (Figure A–1) and `dguxSoftware` (Figure A–2). The objects are arranged under the lowest level hierarchies (shown in bold on the figures) that contain them.

The information in this appendix is extracted from the file `/usr/etc/snmp/dgc.mib.Z`. See this file for the most complete and up–to–date information about MIBs maintained by Data General. (The file is compressed: use **zcat** to un–compress it.)

The `/usr/etc/snmp` directory contains several other files defining additional MIB objects supported by **snmpd**. The file `snmp_info.dat` is used by **snmpd** and the SNMP commands to map object IDs and names. You may add new mappings. However, should you inadvertantly introduce errors, the SNMP daemon or commands may fail. Data General will update this file as required in future releases.

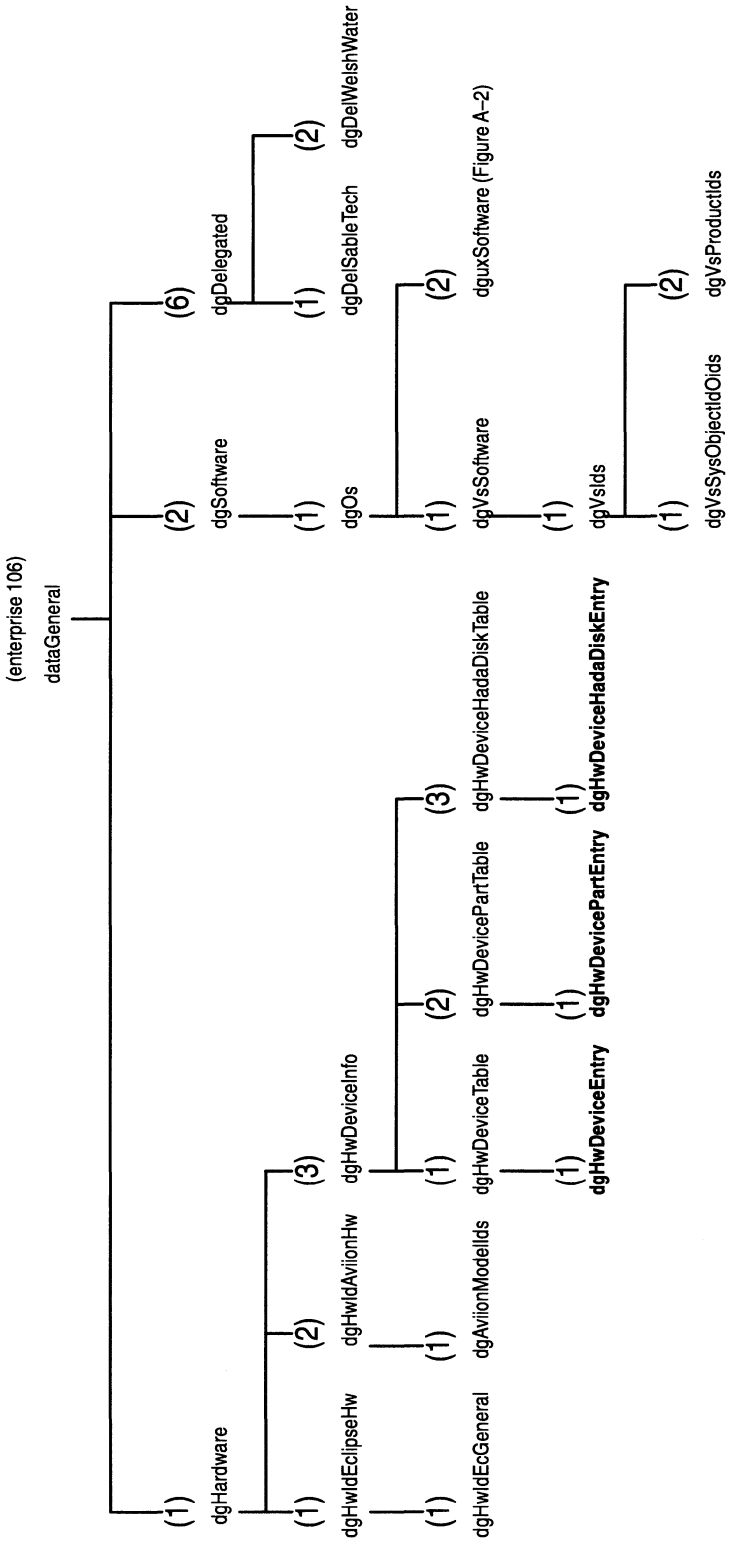


Figure A–1 Data General MIB Hierarchy

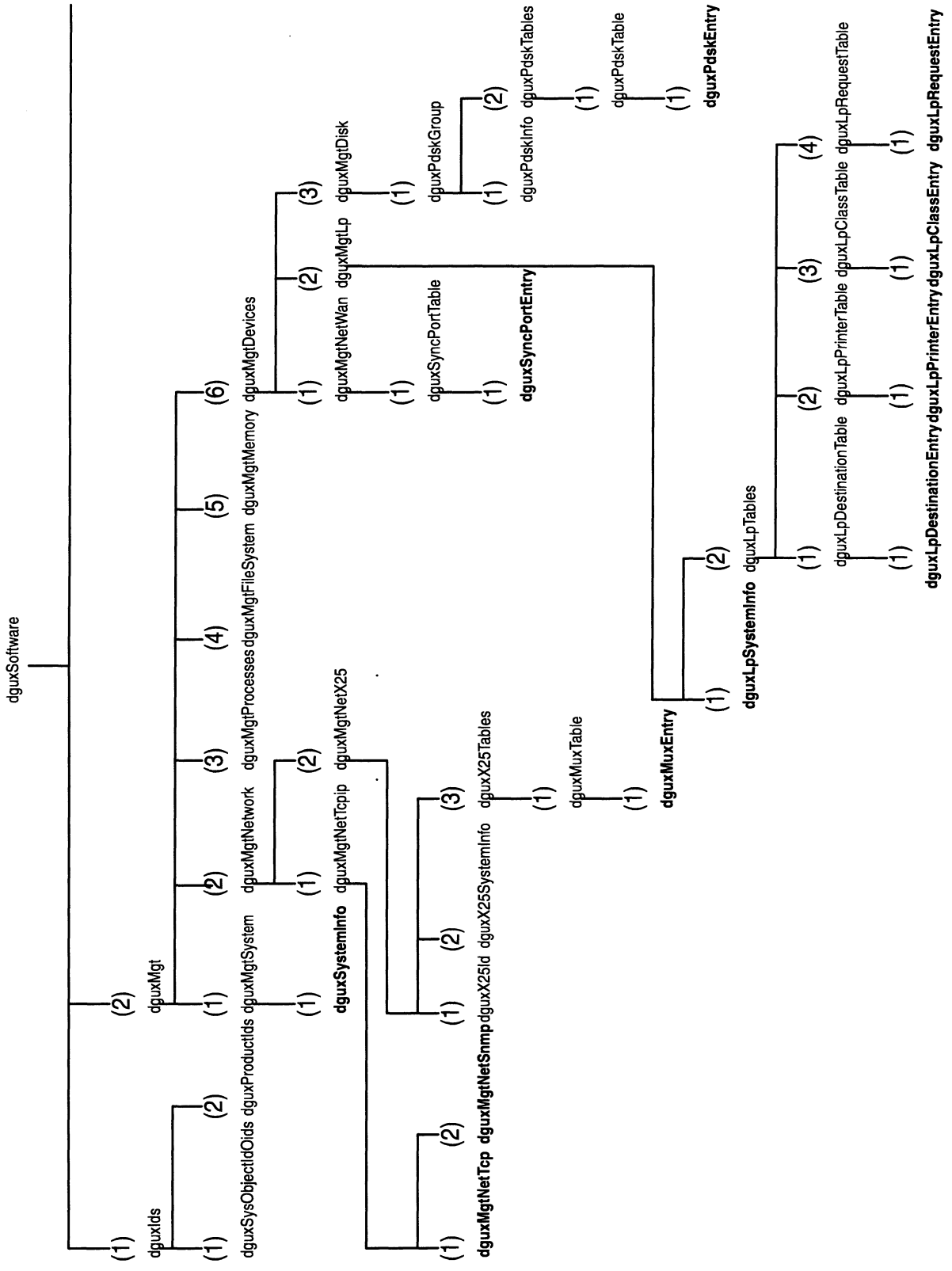


Figure A–2 DG/UX MIB Hierarchy

dgHwDeviceEntry

dgHwDeviceIndex (read–only)
The device's index. This index is independent of any other ordering of the devices and is strictly to identify object instances.

dgHwDeviceName (read–only)
The name of the device as known in the system. In DG/UX this is usually the name in `/dev/fru` for the management interface file.

dgHwDevicePartCount (read–only)
The number of parts that comprise the device.

dgHwDeviceType (read–only)
The device type:

hada	(4097), —'1001'H
syac	(4098), —'1002'H
scsiDisk	(4100), —'1004'H
scsiTape	(4101), —'1005'H
cisc	(4102), —'1006'H
insc	(4103), —'1007'H
dgsc	(4104), —'1008'H
hadaII	(4105) —'1009'H

dgHwDevicePartEntry

dgHwDevicePartIndex (read–only)
An index for the parts which make up a device. On DG/UX this index matches the index returned by a FRU_INQUIRY IOCTL call on the device.

dgHwDevicePartClass (read–only)
The type (sometimes known as the class) of part.

dgHwDevicePartPresent (read–only)
The value no or yes indicating whether this part for the device is present. Some multi–part devices are usable even if some parts are missing.

dgHwDevicePartAddrQualifier (read–only)
The qualifier needed to interpret the address of the part: vme, smdEsdi, scsi, integrated, diskArray, clusterId, memory, index, tapeArray.

dgHwDevicePartAddress (read–only)
The addres of the part.

dgHwDeviceHadaDiskEntry

dgHwDeviceHadaDiskStatus (read–only)
The current status of a disk in a HADA:

`other`: Not a known state (indicates internal software problem on HADA–I; not returned for HADA–II)

`off`: Turned off by disk array controller (this indicates a bad disk that should be fixed)

`notPresent`: Not there or didn't power up

`present`: Disk present but unresponsive (not reported by HADA–I)

`poweringUp`: running diagnostics (HADA–I could be formatting here as well)

`unBound`: Free disk waiting to be bound (also HADA–I BIND state)

`bound`: Bound disk waiting to be opened (also HADA–I ASSIGNED state)

`rebuilding`: data on disk being reconstructed

`binding`: binding (not reported by HADA–I)

`formatting`: formatting the disk (not reported by HADA–I)

`enabled`: A good disk being used (normal state)

dguxSystemInfo

dguxSystemModelOid (read–only)
The AViiON CPU model identifier.

dguxSystemSysIdNo (read–only)
The system identifier from **sysconf**(`_SC_BCS_SYS_ID`).

dguxSystemNumProcessors (read–only)
The number of available processors in the system. (All are not necessarily available; see dg_sys_info_machine_info.)

dguxSystemSoftwareId (read–only)
The DG/UX release identifier (defined under **dguxProductIds**). See **uname**(2).

dguxMgtNetTcp

dguxTcpKeepAliveTimer (read–write)
The keep alive timer in milliseconds (note some implementations may use one second resolution).

dguxTcpArpTimeout (read–write)
The ARP cache timeout in seconds. See RFC 1122 section 2.3.2.1.

dguxTcpIpSrcRtForward (read–write)
Contains `noForward` or `forward` indicating whether datagrams will be forwarded. See RFC 1122 section 3.3.5.

dguxMgtNetSnmp

dguxSnmpCacheTimeout (read–write)
The SNMP daemon data cache timeout in seconds, or 0 if the daemon does not cache data. This value specifies the maximum length of time the daemon keeps and reuses cached data before refreshing it. If fewer than this number of seconds has elapsed since the last request, out–of–date data will be retrieved. Setting a larger value increases the likelihood of retrieving old data, while setting a smaller value may cause slower response.

dguxSyncPortEntry

dguxSyncPortIndex (read–only)
The sync line's index value.

dguxSyncPortName (read–only)
The name of the sync line entry in `/dev`, such as `vsxb010`.

dguxSyncPortInFrames (read–only)
The number of good frames received on this port excluding any error frames.

dguxSyncPortOutFrames (read–only)
The number of frames transmitted on this port excluding any aborted or known error frames.

dguxSyncPortDsrTimeOuts (read–only)
The number of times DSR didn't come up within the timer window after DTR was raised.

dguxSyncPortCtsUpTimeOuts (read–only)
The number of times CTS didn't come up within the timer window after RTS was raised.

dguxSyncPortCtsDropTimeOuts (read–only)
The number of times CTS didn't drop within the timer window after CTS was lowered.

dguxSyncPortInBufferOverFlows (read–only)
The number of frames that were too large to fit within the receive buffer.

dguxSyncPortInNonOctetFrames (read–only)
The number of frames whose length was not a multiple of eight bits.

dguxMuxEntry

dguxMuxMajorIndex (read–only)
The MUX table entry index.

dguxMuxMinorIndex (read–only)
The MUX table pointer index.

dguxMuxStatus (read–write)
Status indicator: `active`, `notInService`, `notReady`, `createAndGo`, `createAndWait`, `destroy`. This object has the semantics described for the RowStatus textual convention in RFC 1443. The enumerations have been redefined here for the benefit of compilers that don't understand the RowStatus textual convention.

dguxMuxPointer (read–write)
The pointer to the multiplexed entity.

dguxMuxDescr (read–write)
The entry's description.

dguxLpSystemInfo

dguxLpSysSchedulerStatus (read–write)
Scheduler status, similar to **lpstat –r**: `unknown`, `setStart`, `setStop`, `running`, `notRunning`.

dguxLpSysNumRequests (read–only)
The number of printer requests remaining to be completed. Similar to **lpstat –o | wc**.

dguxLpSysDefaultDest (read–write)
The default printer for the system, or (0 0) if there is none. Similar to **lpstat –d**.

dguxLpDestEntry

dguxLpDestIndex (read–only)
The printer destination index.

dguxLpDestStatus (read–write)
The current status of the destination: `unknown`, `setAccept`, `setReject`, `schedulerDown`, `accepting`, `rejecting`.

dguxLpDestType (read–only)
This printer destination type: `unknown`, `printer`, `class`.

dguxLpDestNumRequests (read–only)
The number of printer requests remaining for this destination.

dguxLpDestName (read–only)
The name of the printer destination. This can be a class or a printer name.

dguxLpDestTlm (read–only)
The time when this destination was last modified (i.e., started accepting or rejecting requests).

dguxLpPrinterEntry

dguxLpPrinterStatus (read–write)
The current status of the printer: `unknown`, `setEnabled`, `setDisabled`, `schedulerDown`, `faulted`, `disabled`, `idle`, `printing`.

dguxLpPrinterName (read–only)
The name of the printer (same as **dguxLpDestName**).

dguxLpPrinterPath (read–only)
The device associated with this printer. For local printers, this is a pathname. For remote printers, this is a printername/hostname pair.

dguxLpPrinterTlm (read–only)
The time when the printer was last modified (disabled or enabled).

dguxLpClassEntry

dguxLpClassMemberIndex (read–only)
The class member indexes. The first index of this object identifies a class, the second index identifies the member of the class. Both are indexes into the **dguxLpDestination** table. The returned value is the value of the second index. For example, `dguxLpClassMemberIndex.1.2` indicates that the class, with index 1, has as a member the printer with an index of 2 (the returned value).

dguxLpRequestEntry

dguxLpRequestIndex (read–only)
The request's index.

dguxLpRequestSize (read–only)
The size of the request in bytes.

dguxLpRequestStatus (read–write)
This request's status: `unknown`, `setHold`, `setResume`,

`setImmediate`, `setCancel`, `schedulerDown`, `canceled`, `held`, `printing`, `queued`.

dguxLpRequestRank (read–only)
The position of the job in the request queue.

dguxLpRequestId (read–only)
The job's request ID.

dguxLpRequestUser (read–only)
The name of the user who submitted the request.

dguxLpRequestInfo (read–only)
Additional information about the status of the print request, such as the time submitted, assigned printer name, form name, or character set name.

dguxLpRequestStatusFlags (read–only)
Status flags, returned as bits in an octet string:

0 scheduler down
1 printing
2 canceled
3 failed
4 finished
5 assigned
6 queued remotely
7 queued to printer
8 form
9 character set
10 notify
11 held for change
12 held by admin
13 being held
14 filtered
15 being filtered

Bits are numbered starting with the most significant bit of the first byte being bit 0, the least significant bit of the first byte being bit 7, the most significant bit of the second byte being bit 8, and so on. A 1 bit indicates that the event has happened, while a 0 bit indicates that the event has not happened.

If the scheduler is down, the daemon sets a single bit set indicating the scheduler is down. This bit will be cleared and the appropriate bits set when the scheduler comes back up.

dguxPdskEntry

dguxPdskMajorNumber (read–only)
The disk's major device number.

dguxPdskMinorNumber (read–only)
The disk's minor device number.

dguxPdskName (read–only)
The disk's name in the `/dev/rpdsk` directory.

dguxPdskInBlocks (read–only)
The number of 512–byte blocks read from the disk.

dguxPdskOutBlocks (read–only)
The number of 512–byte blocks written to the disk.

dguxPdskInRequests (read–only)
The number of read requests completed by the disk.

dguxPdskOutRequests (read–only)
The number of write requests completed by the disk.

dguxPdskBusyTime (read–only)
The amount of time since system boot that the disk has been busy servicing requests.

dguxPdskResponseTime (read–only)
The amount of time since system boot that requests to the disk have waited for completion.

End of Appendix

Glossary

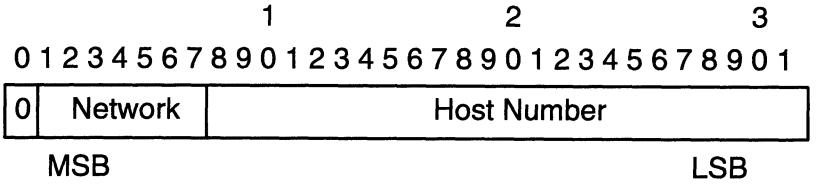
Abstract Syntax Notation One

An ISO specification of an abstract language to be used to define Simple Network Management System (SNMP) object types and encodings. The language is defined in ISO 8824 and the encoding is defined in ISO 8825.

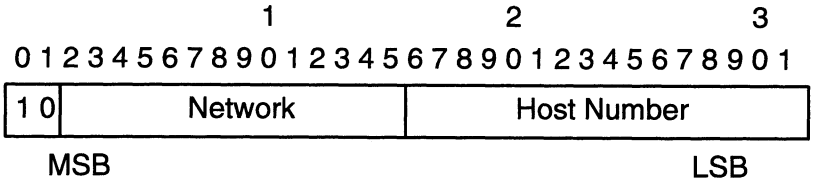
address class

A class of address recognized on the Internet network. Three classes are available: Class A, Class B, and Class C. Conceptually, the 32-bit address consists of a class identifier, a network number, and a host number. The format of the address depends on the class as the following diagram shows:

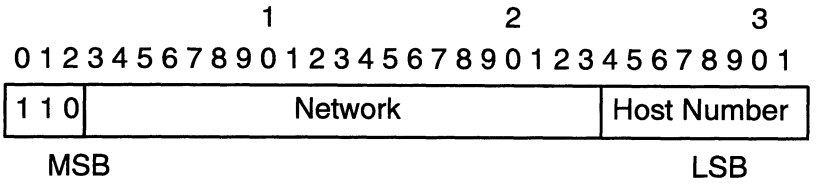
Class A Internet Address



Class B Internet Address



Class C Internet Address



The Defense Data Network — Network Information Center (NIC) assigns network numbers, and individual organizations assign host numbers.

Address Resolution Protocol (ARP)

A kernel-level protocol used to map an Internet address to a hardware (Ethernet or token ring) address. ARP can be used only across a single physical network and can be used only over networks that support broadcasts over the communications medium.

agent

A Simple Network Management Protocol (SNMP) agent. An SNMP agent is an entity contained within, or acting for, a network element that services queries for and performs operations requested by an SNMP *manager*. For TCP/IP for AViiON Systems, the agent is a user-level daemon called **snmpd**. In the client-server model, the agent is the server.

anchored connection

When peer processes on each end of a connection are anchored, you can transfer data without specifying the peer. In this context, anchored means that there is an actual connection between the processes (that is, one side has issued a **connect** system call and the other has issued an **accept** system call).

architecture

See network architecture.

ARPANET

A wide-area network funded by the Defense Advanced Research Projects Agency (DARPA). The ARPANET served as the basis for early networking research and as a backbone network during the development of the Internet network.

ASN.1

See Abstract Syntax Notation.

association

Binds communicating processes to one another over a network. An association is composed of a local address bound to a local port and a remote address bound to a remote port.

autonomous system (AS)

An independent network that is connected to the Internet and which has its own internal mechanisms for collecting and sharing routing information (called reachability information) with other other autonomous systems. Autonomous system IDs are defined by and obtainable from the Network Information Center (NIC).

Berkeley Software Distribution (BSD)

A general term for the version of UNIX created at the University of California at Berkeley. BSD UNIX offered UNIX-like utilities, such as **rcp**, to use the basic TCP/IP protocol and the socket abstraction that lets application programmers access the protocols. Data General's implementation of TCP/IP is based on the BSD implementation.

binding

Assigns a name to a socket so that a process can use the socket to communicate with another process. See the **bind**(2) manual page for details.

bridge

A device that copies raw packets from one network to another of the same type. Usually, bridges operate at the physical network level. Bridges differ from repeaters because bridges store and forward complete packets, while repeaters forward electrical signals.

broadcast

To send the same message to all systems on a network at the same time.

broadcast address

An Internet address used for all hosts on the network.

Any Internet address with a host portion that consists of all 1's is reserved for broadcast for systems compatible with BSD 4.3. Any Internet address with a host portion that consists of all 0's is reserved for broadcast for systems compatible with BSD 4.2. On networks where both BSD 4.2 and BSD 4.3 software is used, a host portion of all 0's works best.

A 48-bit Ethernet address that consists of all 1's.

BSD

See Berkeley Software Distribution (BSD).

bus network

A network that provides a single connective link between hosts; any of the hosts can transmit to any other, but only one host can transmit at a time.

Carrier Sense Multiple Access with Collision Detection (CSMA/CD)

A characteristic of network hardware that lets multiple stations contend for access to a transmission medium while preventing simultaneous transmission attempts. Ethernet is an example of such a medium.

client

An operating system (OS) client.

An executing program that sends a request to a server for services and waits for a response. Thus, there are network clients, Network Information Service clients, Network File System clients, and clients of the domain name system.

client-server relationship

Refers to a pattern of interaction among application programs that communicate over the network. Server programs provide services (such as remote login facilities) and client programs consume them. The relationship describes which program initiates a connection, sends data first, and controls the communication link.

CMC-130 Ethernet Intelligent Controller
See VLCi.

connection
The path between two processes that provides reliable, stream-oriented, process-to-process delivery service.

connection-based communication
Characteristic of the reliable, stream-oriented, process-to-process service offered by the Transmission Control Protocol (TCP). System calls are used to connect to and communicate with remote processes.

connectionless communication
Characteristic of the packet delivery service offered by the Internet Protocol or the User Datagram Protocol. Treats each packet or datagram as a separate entity that contains the source and destination address. Connectionless services may drop or duplicate packets or deliver them out of sequence.

controller
A hardware device that controls the operation of another device or system. For example, a local area network (LAN) controller controls signals from a network to the CPU.

daemon
An unattended background process, often perpetual, that performs a system-wide public function, for example, **inetd**.

DARPA
See ARPANET.

datagram
A self-contained package of data carrying the necessary information to route itself from source to destination. It is the unit of transmission in the IP protocol. To cross a particular network, a datagram is encapsulated inside a packet.

datagram socket
Sends datagrams in and receives datagrams from both directions simultaneously, preserving logical breaks in the data. Data travels in complete packets rather than streams or bytes. The packets may arrive out of order or may fail to be delivered. This service is known as connectionless communication.

Defense Data Network (DDN)
Used loosely to refer to the MILNET and ARPANET networks and the TCP/IP protocols that these networks use.

Defense Data Network — Network Information Center (NIC)
The part of the Defense Data Network (DDN) that has authority to assign Internet addresses.

device driver

A set of software used to manage a peripheral device. For example, **cien** is a device driver used to manage a CMC-130 Intelligent Ethernet Controller.

directory tree

Refers to the / (root), **usr**, **swap**, **share**, and other directories that make up the directory space of a DG/UX system. Since OS clients in DG/UX let / (root), **usr**, **swap**, and **share** be subdirectories in the OS server's file system, the terms / (root) file system and **usr** file system are no longer appropriate when referring to a client. The OS server still has a / (root) file system and a **usr** file system that contains the software associated with the primary release.

diskless client

An operating system (OS) client with no disk drive of its own.

domain

A subset of the hierarchical domain name space used to accommodate the large, complicated set of names used in the Internet. The domain name space has a tree structure. A domain extends from a particular point in the hierarchy to the leaves of the tree.

domain name

An ordered grouping of labels that name a domain. The order traces from a given node in the domain name space up to the root of the tree. Each label in the domain name is separated with a dot.

domain name system

A distributed database that lets hosts on the Internet share information. This information can include the names and Internet addresses of hosts, a list of well known services provided by hosts, and so on. The domain name system consists of a resolver and a name server. A resolver queries a name server for information.

dual counter-rotating ring

A networking architecture similar to token ring. The primary difference being a dual counter-rotating ring has two rings attached to each host on the network. Each ring has a token, one rotating clockwise, the other counterclockwise. The dual ring setup ensures the network remains functional if part of the network hardware develops a problem. If one of the hosts or rings fails, the network can re–configure itself to bypass that ring or host. In the event of multiple failures, the network breaks into smaller rings, each functional but isolated from each other.

Ethernet

A type of local area network developed by the Xerox Corporation. An Ethernet network consists of cable and interface hardware that connects hosts. Only one host can use the network at a time. Hosts send out packets of information over the network whenever they detect that other hosts are not using it. *See also* Carrier Sense Multiple Access with Collision Detection (CSMA/CD).

Ethernet address

A number that identifies a specific host on an Ethernet-based local area network. Ethernet addresses are set on a host during manufacture with hardware switches, and are guaranteed to be unique.

Fiber Distributed Data Interface (FDDI)

A networking architecture for which a standard is defined in ANSI X3T9.5. FDDI networks usually have a dual, counter-rotating ring topology and use fiber optic cable as the transmission media. The standard does allow for single ring configurations. Data transmission can be either asynchronous or synchronous. Each host on the network transmits data to the next, acting as a repeater. A host can transmit data only when it is in possession of a special message called a token.

file system

See logical disk file system.

File Transfer Protocol (FTP)

A user-level protocol accessed through the **ftp**(1C) command. FTP lets you transfer files from one host to another. The File Transfer Protocol uses TCP as the transport level protocol.

foreign release

A release supplied by a vendor other than Data General Corporation.

fully qualified name

A domain name that includes all node names from a specific point in the domain name space to the root. Often, this means that the name uses a trailing dot, which signifies that the origin is the root domain. The absence of a trailing dot in a domain name often signifies that the name is relative to some other origin.

gateway

A computer (or, in many cases, special-purpose equipment designed for use as a gateway) that converts from one protocol family to another.

handshaking

The exchange of control information between two processes communicating over a network where each process takes turns transmitting data.

heterogeneous OS server

An OS server that provides many system releases to many hosts. Clients share release software and maintain private files the same way clients of homogeneous OS servers do; clients have the ability to boot other releases supported by the OS server if the other releases are compatible with client hardware.

homogeneous OS server

An OS server that provides a single system release to many other clients that all share a single copy of some of the system release files but which also maintain private copies of dependent release files. Processing is done on each client's processor.

hop count

A measure of distance between two points in an internet. A hop count of *n* means that *n* gateways separate the source and destination.

host

A computer that is configured to share resources with other computers in a network. Refers to any computer: stand-alone, OS server, or OS client. *See also* local host, remote host.

host-dependent

The commands and files that are dependent or unique to an individual host. If a given file cannot or should not be shared between several hosts, then it is host-dependent. Every client host on the OS server needs its own copy of host-dependent files and needs to be able to write to its own data files. This class of files also contains the set of commands and data files required to boot a host. The **/etc/passwd** file is an example of a host-dependent file. Host-dependent files may or may not be release-dependent.

host ID

A unique number that identifies the host. In the DG/UX system, the host ID typically is the host's most commonly used Internet address. *See also* host number.

hostname

A string that represents a host. Hostnames are associated with Internet addresses in the **/etc/hosts** file.

host number

The host portion of a computer system's Internet address. *See also* address class, Internet address.

interface

A common boundary between two devices, programs, or systems. An interface gives two systems that handle information differently a way to interact.

Internet

The collection of networks and gateways, including the ARPANET and the MILNET, that use the TCP/IP protocol suite and function as a single, cooperative virtual network. Governed by the Internet Activities Board (IAB).

Internet address

A unique, 32-bit number that identifies a specific host on the Internet. Internet addresses are expressed in dot notation, and have the general form *a.b.c.d*, where each letter represents eight bits, or one octet. One part of the Internet address represents the network number, and one part represents the host number. There are three classes of Internet address: Class A, Class B, and Class C. The difference among classes depends on the length of the network number: Class A network numbers are one octet long, Class B network numbers are two octets long, and Class C network numbers are three octets long. Network numbers are assigned by the Network Information Center (NIC).

Internet Control Message Protocol (ICMP)

The part of IP that handles error and control messages. Gateways and hosts use ICMP to tell the source of datagrams about problems delivering the datagrams. ICMP also lets a host test whether a destination is reachable and responding.

Internet Protocol (IP)

A kernel-level protocol that defines unreliable, connectionless delivery of datagrams. An IP datagram contains the addresses of its source and destination, and the data transmitted. Connectionless service means that the protocol treats each datagram as a separate entity; the protocol can deliver packets out of sequence, or can drop packets. IP defines the exact format of data as it travels through a network, but delivery of data is not guaranteed.

internetwork

A technology that allows the interconnection of disparate physical networks into a coordinated functional unit. An internetwork (for example, the Internet) accommodates different networking hardware by adding physical connections and by implementing a standard set of protocol conventions.

interoperability

The ability of diverse computing systems to cooperate in solving computational problems.

iterative approach

With this approach, when a database does not contain the requested information, the name server refers the program to another name server and lets the program pursue the query. *See also* recursive approach.

IXE (IP to X.25 Encapsulation)

A program that performs the translations required to use Internet Protocols (TCP/IP) over X.25 networks, as described in *RFC 877 (A Standard for the Transmission of IP Datagrams over Public Data Networks*).

kernel

The nucleus of the DG/UX operating system. It controls access to the computer, manages the computer's memory, maintains the file system, and allocates the computer's resources among users. The kernel is sometimes described as the DG/UX system proper; resident code that implements the system calls.

line discipline

A module to handle protocol or data conversion for a stream. A line discipline, unlike a filter, is part of the kernel.

local area network (LAN)

A network within a small area or a common environment, such as within a building. Ethernet is a type of LAN.

local host

The computer your terminal is attached to. A local host can send and receive information from a remote host through connection-based or connectionless communication.

local port

See port.

logical disk file system

A file system directly associated with a specific logical disk; one logical disk equals one file system.

logical network

A network that may consist of one or more physical networks or may be a subdivision of a single physical network. You set up a logical network through your chosen addressing scheme.

Management Information Base (MIB)

A hierarchically organized collection of network management objects. The Internet-standard object types are defined in *RFC 1213*.

manager

An SNMP manager, which is a program that makes SNMP queries to SNMP agents for information about network elements. These applications are available from a variety of vendors, and provide different levels of support. In the client-server model, the SNMP manager is the client.

mapping

Associating the elements of two different representations of a system (like a directory tree) so that a correspondence exists between the two systems. Every element in one system can be mapped to an element in the other system.

MIB

A hierarchically organized collection of network management objects. The Internet-standard object types are defined in *RFC 1213*.

MILNET

Originally part of the ARPANET, the MILNET was partitioned in 1984 to give military installations reliable network service while the ARPANET continues to be a research network. The MILNET uses the same hardware and protocols as the ARPANET.

multiplexing

Using a device to handle several similar but separate operations simultaneously by alternating attention among them.

name server

Part of the domain name system. The name server runs as a daemon process called **named**, and responds to queries by consulting its database. If the answer is not in its database and the name server acts recursively, the name server forwards a query to other name servers.

native release

A release supplied by Data General Corporation.

netmask

See network mask.

network

The hardware and software that constitute the interconnections between computer systems, permitting electronic communication between the systems and associated peripherals. Networking for computer systems, means sending data from one system to another over some communications medium (coaxial cable, phone lines, and so on). Common networking services include file transfer, remote login, and remote execution.

network element

Any device on the network, which includes but is not limited to a host, gateway, router, bridge, modem, or terminal server.

network mask

A bit mask, associated with the network interface, that corresponds with the bits of the Internet address that determine the network portion of the address.

Network File System (NFS)

A service that lets many users share file systems over a network.

Network Information Center (NIC)

A branch of the federal government with general authority over the Internet. NIC assigns Internet addresses, domains, and autonomous system IDs.

Network Information Service (NIS)
A service that maintains a set of databases about hosts, networks, and services for an entire network..

network management station
A host or device that runs SNMP manager software to monitor a network.

network number
The network portion of an Internet address. The length of the network number depends on the address class.

NIC
The part of the Defense Data Network (DDN) that has authority to assign Internet addresses.

NIS domain
A named set of NIS maps, which are sets of keys and associated values.

A logical grouping of hosts in an NIS environment. Each host in a domain relies on the same server(s) for certain resource sharing and security services. Each domain has one master and zero or more slave servers.

NIS server
A computer that creates and maintains the following information for hosts in an NIS domain: advertised resources, hostnames and passwords, names and addresses for name servers of other domains (optional), host user and group information used for ID mapping (optional).

Open Network Computing/Network File System (ONC™/NFS®)
A package that consists of the Network Information Service (NIS), NFS, and other networking facilities.

OS client
A computer system that gets all or part of the operating system from an OS server through a local area network (LAN).

OS release
See release.

OS server
A computer system with its own disk that contains a bootable portion of the operating system, and that can provide all or part of the operating system to OS clients through a local area network (LAN).

package
A set of software traditionally called a product. DG/UX and DG/UX ONC/NFS are examples of packages.

packet
Refers to the unit of data sent across a packet-switching network. The format of a packet typically is defined by the protocol.

peer processes

Processes on different computer systems that run at the same level in the communications hierarchy. That is, both processes run at the user-level or both run at the kernel-level. Peer processes use protocols to exchange data that their peer can understand.

physical network

The hardware (computers, communication controllers, media) that makes up a network.

port

The point of connection between a device, such as a communications controller, and the CPU.

The number used to determine which process on a host receives information. Networking software uses ports to allow processes on different computers to communicate. A single process can use several ports, using each to communicate with the port of a different remote process. The local port exists on the local host. The remote port exists on the remote host.

primary release

The OS release that resides in the server's / (root) and **usr** logical disk file systems. Other (secondary) releases may reside on the OS server in a separate directory space.

process

A program being executed. When a program is executed by several people simultaneously, there are several processes but only one program. Each process is cataloged in the system's process table.

protocol

A set of rules that govern the transfer of data and communication between two or more entities in a network.

Protocol Data Unit (PDU)

A command supported by the SNMP protocol (for example **GetRequest-PDU**, **GetResponse-PDU**, and so on).

proxy ARP

Allows you to use the Address Resolution Protocol (ARP) to do away with explicit routing tables, which deal with Internet addresses, and route messages at the level of Ethernet addresses. Routers act as proxies for other hosts.

raw socket

Allows access to the underlying communication protocols (such as IP) that support higher level protocols. These sockets normally send information in datagrams, but their characteristics depend on the interface provided by the protocol.

recursive approach

With this approach, if a database does not contain the requested information, the name server forwards the query to other name servers until it finds one whose database contains the requested information. Once it gets the information, the first name server passes it on to the requesting program. By default, Data General's DNS takes a recursive approach. *See also* iterative approach.

redirect

A specific kind of message using ICMP (Internet Control Message Protocol). It contains information that generally translates to "In the future, to get to address xyz, please use gateway abc instead of me." Some TCP/IP implementations (including TCP/IP for AViiON Systems) use these redirects to add entries to their routing table.

release

A set of software intended for a specific architecture and operating system. A release encompasses all software required for a host including the release-dependent and host-dependent files.

A release identity is defined by the contents of the release's **usr** directory space. Additional software packages loaded into the release's **usr** directory space become part of the release.

release-dependent

The system commands and files that are dependent on a release. A file that cannot or should not be shared between releases is *release-dependent*. If, for example, manual pages and certain ASCII data files can be shared by more than one release, then they are *release-independent*. The master files and most commands are examples of release-dependent files. Release-dependent files may or may not be host-dependent.

remote host

The other computer that a local host sends information to and gets information from through connection-based or connectionless communication.

remote port

See port.

repeater

A device that copies each bit of a packet from one segment of a network to another. A repeater is like an amplifier. It increases the length of the physical network.

Request for Comments (RFC)

A series of technical papers that contain surveys, measurements, techniques, specifications, and proposed and accepted Internet protocol standards. RFCs are available from the Defense Data Network – Network Information Center (known as the NIC), Government Systems Inc., Attn: DDN Network Information Center, 14200 Park Meadow Drive, Suite 200, Chantilly, Virginia, 22021.

resolver

Part of the domain name system. The resolver is a set of routines that act as an interface between user programs and name servers. These routines are linked in with the programs that use them.

Reverse Address Resolution Protocol (RARP)

A kernel-level protocol used by a diskless computer at startup to find its Internet address. The diskless system broadcasts a request that contains its Ethernet address and the server responds by sending the machine its Internet address.

route

The path that network traffic takes from its source to its destination.

router

A device (a computer or special equipment) that forwards packets of a particular protocol type (for example, IP) from one network to another. It is possible to use a computer as a router as long as it has more than one network interface, and its software is prepared to forward datagrams.

routing table

A table that exists in the kernel that associates Internet addresses with specific hosts.

sendmail

A command that implements the Simple Mail Transfer Protocol (SMTP), which allows the dispatch of mail messages. The **sendmail** command uses TCP as the transport level protocol.

server

An OS server.

A server process that provides network services to a client process, for example, **telnetd**.

A Network Information Service (NIS) server, which provides NIS database information to NIS clients.

A Network File System (NFS) server, which provides file system access to remote NFS clients.

A name server, which is part of the domain name system.

shell

A command interpreter and programming language that acts as an interface to the UNIX® system. As a command interpreter, the shell accepts commands and acts on them. As a programming language, the shell's features include flow control and string-valued variable definition. When you log in to the system, you acquire a login shell. In this shell, you can run another shell program, which becomes a subshell to your login shell. The two most common shells are the Bourne shell and the C shell. For more information, see *Using the DG/UX™ System*.

Simple Network Management Protocol (SNMP)

A protocol created and adopted for short-term use on the Internet for network management. SNMP is based on a client-server model where agents (servers) provide information to managers (clients). The protocol is defined in *RFC 1157*.

SNMP agent

An entity contained within, or acting for, a network element that services queries for and performs operations requested by an SNMP *manager*. For TCP/IP for AViiON Systems, the agent is a user-level daemon called **snmpd**. In the client-server model, the agent is the server.

SNMP manager

A program that makes SNMP queries to SNMP agents for information about network elements. These applications are available from a variety of vendors, and provide different levels of support. In the client-server model, the SNMP manager is the client.

socket

A mechanism in the kernel that acts as an interface between processes on different computers. There are three types of sockets available with TCP/IP for AViiON Systems: stream sockets, datagram sockets, and raw sockets.

socket name

The concatenation of an Internet addresses with a port number. Also called a connection endpoint.

stand-alone system

A machine with its own disks (typically a mini-computer) that supports dumb terminals in a traditional timeshare environment where all terminals are running the same release. A stand-alone system does not use TCP/IP or NFS to service its terminals, and it has no operating system (OS) clients.

stream

A full duplex, processing and data transfer path in the kernel. It implements a connection between a driver in kernel space and a process in user space, providing a general character I/O interface for the user processes.

stream socket

Used to access TCP to send and receive data in continuous streams of bytes without logical breaks or duplication. Data can pass through the socket in both directions simultaneously, guaranteeing delivery in the original order in which the data is sent.

subnet mask

A bit mask, associated with the network interface, that corresponds with the bits of the Internet address that determine the network portion of the address.

subnets

An extension of the Internet addressing scheme that lets a site use a single Internet address portion of its host address field as a subnet field. Outside the site, routing divides the destination address into an Internet portion and a local portion. Routers and hosts inside a site that uses subnets interpret the local portion of the address by dividing it into a physical network portion and a host portion. Thus, a site can present a single local network number to the world, but still maintain distinct physical networks and routing internally.

tapeless

An OS server without a tape drive. To read a release tape, a tapeless server must access another host that does have a tape drive.

TELNET

A user-level protocol accessed through the **telnet** command. The TELNET protocol allows a user on one host to interact with a remote host as if the terminal were directly connected to the remote host. TELNET uses TCP as the transport level protocol.

token ring

A networking architecture for which a standard is defined in IEEE 802.5. Token ring networks use a ring topology. Data passes in a single direction through the ring. Each host on the network transmits data to the next, acting as a repeater. A host can transmit data only when it is in possession of a special message called a token.

transceiver

The interface between a controller and the cable in an Ethernet-based local area network.

Transmission Control Protocol (TCP)

A kernel-level protocol that defines reliable, end-to-end delivery of datagrams. TCP is connection-based because it establishes a connection between communicating hosts before transmitting data. TCP allows a process on one host to send data to a process on another through a byte stream. TCP uses IP to transmit information along an Internet network. TCP messages include a protocol port number that allows the sender to distinguish multiple programs on the remote host.

Transport Layer Interface (TLI)

A library of routines that uses STREAMS mechanisms to access transport-level services in the kernel.

trap

A Trap-PDU SNMP message sent from the SNMP agent to an SNMP manager. The message alerts the manager of some event, such as a system reboot or a change to an interface.

Trivial File Transfer Protocol (TFTP)

A user-level protocol accessed through the **tftp** command that allows file transfer with minimal capability and overhead. The **tftp** command depends on the UDP protocol.

During first stage boot with the AViiON station, the boot program, once it determines its Internet address, uses TFTP to transfer a file that contains the executable image of a second stage boot program.

User Datagram Protocol (UDP)

A kernel-level protocol that lets a process on one host send a datagram to a process on another. UDP is a connectionless transport protocol. UDP messages include a protocol port number that lets the sender distinguish multiple programs on the remote host.

UUCP

Stands for UNIX-to-UNIX copy. A standard telecommunications protocol for UNIX systems.

V/Ethernet 3207 Hawk Local Area Network Controller

See VLC.

VFC

A VME-bus Fiber Distributed Data Interface controller used in AViiON series systems. The controller was developed by the Interphase Corporation. For more information, see *V/FDDI (4211 Peregrine) User's Guide.*

V/FDDI 4211 High-Performance FDDI Node Processor for VMEbus

See VFC.

VLC

A VME-bus Ethernet controller used in AViiON series systems. The controller was developed by the Interphase Corporation. For more information, see *V/Ethernet 3207 Hawk Local Area Network Controller for Ethernet User's Guide.*

VLCi

A VME-bus Ethernet controller used in AViiON series systems. The controller was developed by CMC Corporation. For more information, see *VLCi Ethernet LAN Controller (CMC-130) Reference Guide.*

VTRC

A VME-bus token ring controller used in AViiON series systems. For more information, see *Configuring the VME Token Ring Controller (VTRC) for AViiON Systems.*

wide area network (WAN)

A network that extends across a wide area, such as across a street or across an ocean.

workstation

A system with its own processor, its own graphics terminal, and graphics software (shared or host-dependent). A workstation could be an OS server, a diskless client, or a client with a disk.

X.25

The CCITT (International Telegraph and Telephone Consultative Committee) recommendations made in 1980, 1984, and 1988 for three layers (levels) of communication: the physical layer, the frame layer, and the packet layer.

X-terminal

A graphics terminal with processing and communication facilities sufficient to run the X-server locally. An X-terminal acts as an end-user I/O device for the X-client programs that run on the host or hosts to which it is connected.

zone

A division of the domain name space that involves the management of authoritative data about the name servers and hosts with it. A zone can be a whole or partial domain.

End of Glossary

Index

D

E

F

G

H

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Permissions

Notices provided by the copyright holders

The following copyright notices and permissions apply to programs and their use as described in the following manuals:

Programming with TCP/IP on the DG/UX System (093–701024) Managing TCP/IP on the DG/UX System (093–701051)
Using TCP/IP on the DG/UX System (093–701023)

bootp.d

gated

Authors:
Jeffrey C Honig <jch@gated.cornell.edu>
Scott W Brim <swb@gated.cornell.edu>

Portions of this software may fall under the following copyrights:

Copyright (c) 1988 Regents of the University of California.
All rights reserved.

Redistribution and use in source and binary forms are permitted provided that the above copyright notice and this paragraph are duplicated in all such forms and that any documentation, advertising materials, and other materials related to such distribution and use acknowledge that the software was developed by the University of California, Berkeley. The name of the University may not be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

ISIS

Copyright 1991 D.L.S. Associates

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of D.L.S. not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. D.L.S. makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

D.L.S. DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL D.L.S. BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Authors: Robert Hagens and Dan Schuh

OSPF

Copyright 1989, 1990, 1991 The University of Maryland, College Park, Maryland.

All Rights Reserved

The University of Maryland College Park ("UMCP") is the owner of all right, title and interest in and to UMD OSPF (the "Software"). Permission to use,

copy and modify the Software and its documentation solely for non–commercial purposes is granted subject to the following terms and conditions:

1. This copyright notice and these terms shall appear in all copies of the Software and its supporting documentation.

2. The Software shall not be distributed, sold or used in any way in a commercial product, without UMCP's prior written consent.

3. The origin of this software may not be misrepresented, either by explicit claim or by omission.

4. Modified or altered versions must be plainly marked as such, and must not be misrepresented as being the original software.

5. The Software is provided "AS IS". User acknowledges that the Software has been developed for research purposes only. User agrees that use of the Software is at user's own risk. UMCP disclaims all warranties, express and implied, including but not limited to, the implied warranties of merchantability, and fitness for a particular purpose.

Royalty–free licenses to redistribute UMD OSPF are available from The University Of Maryland, College Park.
For details contact:

Office of Technology Liaison
4312 Knox Road
University Of Maryland
College Park, Maryland 20742
(301) 405–4209
FAX: (301) 314–9871

This software was written by Rob Coltun rcoltun@ni.umd.edu

ntp

Copyright (c) 1989,1990,1991,1992 Frank Kardel Friedrich–Alexander Universitaet Erlangen–Nuernberg

This code can be modified and used freely provided that the credits remain intact.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

slip

Copyright (c) 1989 Regents of the University of California. All rights reserved.

Redistribution and use in source and binary forms are permitted provided that the above copyright notice and this paragraph are duplicated in all such forms and that any documentation, advertising materials, and other materials related to such distribution and use acknowledge that the software was developed by the University of California, Berkeley. The name of the

University may not be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Van Jacobson (van@helios.ee.lbl.gov), Dec 31, 1989: – Initial distribution.

tcpdump

Note: two different copyright notes were found in Berkeley sources.

This program is subject to the 'standard' Berkeley network software copyright:

Copyright (c) 1988–1990 The Regents of the University of California. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that: (1) source code distributions retain the above copyright notice and this paragraph in its entirety, (2) distributions including binary code include the above copyright notice and this paragraph in its entirety in the documentation or other materials provided with the distribution, and (3) all advertising materials mentioning features or use of this software display the following acknowledgement: "This product includes software developed by the University of California, Lawrence Berkeley Laboratory and its contributors." Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Copyright (c) 1990, by John Robert LoVerso. All rights reserved. Redistribution and use in source and binary forms are permitted provided that the above copyright notice and this paragraph are duplicated in all such forms and that any documentation, advertising materials, and other materials related to such distribution and use acknowledge that the software was developed by John Robert LoVerso.

THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.

This implementation has been influenced by the CMU SNMP release, by Steve Waldbusser. However, this shares no code with that system. Additional insight gained from Marshall T. Rose's _The_Open_Book_. Earlier forms of this implemention were derived and/or inspired by an awk script originally written by C. Philip Wood of LANL (but later heavily modified by John Robert LoVerso). The copyright notice for that work is preserved below, even though it may not rightly apply to this file.

This started out as a very simple program, but the incremental decoding (into the BE structure) complicated things.

Los Alamos National Laboratory

@()snmp.awk.x 1.1 (LANL) 1/15/90

TIPS ORDERING PROCEDURES

TO ORDER

1. An order can be placed with the TIPS group in two ways:
   a. MAIL ORDER – Use the order form on the opposite page and fill in all requested information. Be sure to include shipping charges and local sales tax. If applicable, write in your tax exempt number in the space provided on the order form.

   b. Send your order form with payment to:

      Data General Corporation
      ATTN: Educational Services/TIPS G155
      4400 Computer Drive
      Westboro, MA 01581–9973

   c. TELEPHONE – Call TIPS at (508) 870–1600 for all orders that will be charged by credit card or paid for by purchase orders over $50.00. Operators are available from 8:30 AM to 5:00 PM EST.

METHOD OF PAYMENT

2. As a customer, you have several payment options:
   a. Purchase Order – Minimum of $50. If ordering by mail, a hard copy of the purchase order must accompany order.

   b. Check or Money Order – Make payable to Data General Corporation. Credit Card – A minimum order of $20 is required for MasterCard or Visa orders.

SHIPPING

3. To determine the charge for UPS shipping and handling, check the total quantity of units in your order and refer to the following chart:

Total Quantity	Shipping & Handling Charge
1–4 Items	$5.00
5–10 Items	$8.00
11–40 Items	$10.00
41–200 Items	$30.00
Over 200 Items	$100.00

If overnight or second day shipment is desired, this information should be indicated on the order form. A separate charge will be determined at time of shipment and added to your bill.

VOLUME DISCOUNTS

4. The TIPS discount schedule is based upon the total value of the order.

Order Amount	Discount
$0–$149.99	0%
$150–$499.99	10%
Over $500	20%

TERMS AND CONDITIONS

5. Read the TIPS terms and conditions on the reverse side of the order form carefully. These must be adhered to at all times.

DELIVERY

6. Allow at least two weeks for delivery.

RETURNS

7. Items ordered through the TIPS catalog may not be returned for credit.
8. Order discrepancies must be reported within 15 days of shipment date. Contact your TIPS Administrator at (508) 870–1600 to notify the TIPS department of any problems.

INTERNATIONAL ORDERS

9. Customers outside of the United States must obtain documentation from their local Data General Subsidiary or Representative. Any TIPS orders received by Data General U.S. Headquarters will be forwarded to the appropriate DG Subsidiary or Representative for processing.

TIPS ORDER FORM

Mail To: Data General Corporation
Attn: Educational Services/TIPS G155
4400 Computer Drive
Westboro, MA 01581 - 9973

BILL TO:	SHIP TO: (No P.O. Boxes - Complete Only If Different Address)
COMPANY NAME _____	COMPANY NAME _____
ATTN: _____	ATTN: _____
ADDRESS _____	ADDRESS (NO PO BOXES) _____
CITY _____	CITY _____
STATE _____ ZIP _____	STATE _____ ZIP _____

Priority Code _____ (See label on back of catalog)

Authorized Signature of Buyer (Agrees to terms & conditions on reverse side)	Title	Date	Phone (Area Code)	Ext.

ORDER #	QTY	DESCRIPTION	UNIT PRICE	TOTAL PRICE

A SHIPPING & HANDLING

☐ UPS

	ADD
1–4 Items	$5.00
5–10 Items	$8.00
11–40 Items	$10.00
41–200 Items	$30.00
200+ Items	$100.00

Check for faster delivery

Additional charge to be determined at time of shipment and added to your bill.

☐ UPS Blue Label (2 day shipping)
☐ Red Label (overnight shipping)

B VOLUME DISCOUNTS

Order Amount	Save
$0–$149.99	0%
$150–$499.99	10%
Over $500.00	20%

Tax Exempt # or Sales Tax (if applicable) _____

ORDER TOTAL	
Less Discount See B	–
SUB TOTAL	
Your local* sales tax	+
Shipping and handling – See A	+
TOTAL – See C	

C PAYMENT METHOD

☐ Purchase Order Attached ($50 minimum)
P.O. number is _____. (Include hardcopy P.O.)
☐ Check or Money Order Enclosed
☐ Visa ☐ MasterCard ($20 minimum on credit cards)

Account Number ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Expiration Date ☐☐☐☐

_____
Authorized Signature
(Credit card orders without signature and expiration date cannot be processed.)

THANK YOU FOR YOUR ORDER

PRICES SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.
PLEASE ALLOW 2 WEEKS FOR DELIVERY.
NO REFUNDS NO RETURNS.

* Data General is required by law to collect applicable sales or use tax on all purchases shipped to states where DG maintains a place of business, which covers all 50 states. Please include your local taxes when determining the total value of your order. If you are uncertain about the correct tax amount, please call 508–870–1600.

Form 702
Rev. 8/87

DATA GENERAL CORPORATION
TECHNICAL INFORMATION AND PUBLICATIONS SERVICE

TERMS AND CONDITIONS

Data General Corporation ("DGC") provides its Technical Information and Publications Service (TIPS) solely in accordance with the following terms and conditions and more specifically to the Customer signing the Educational Services TIPS Order Form. These terms and conditions apply to all orders, telephone, telex, or mail. By accepting these products the Customer accepts and agrees to be bound by these terms and conditions.

1. CUSTOMER CERTIFICATION
Customer hereby certifies that it is the owner or lessee of the DGC equipment and/or licensee/sub–licensee of the software which is the subject matter of the publication(s) ordered hereunder.

2. TAXES
Customer shall be responsible for all taxes, including taxes paid or payable by DGC for products or services supplied under this Agreement, exclusive of taxes based on DGC's net income, unless Customer provides written proof of exemption.

3. DATA AND PROPRIETARY RIGHTS
Portions of the publications and materials supplied under this Agreement are proprietary and will be so marked. Customer shall abide by such markings. DGC retains for itself exclusively all proprietary rights (including manufacturing rights) in and to all designs, engineering details and other data pertaining to the products described in such publication. Licensed software materials are provided pursuant to the terms and conditions of the Program License Agreement (PLA) between the Customer and DGC and such PLA is made a part of and incorporated into this Agreement by reference. A copyright notice on any data by itself does not constitute or evidence a publication or public disclosure.

4. LIMITED MEDIA WARRANTY
DGC warrants the CLI Macros media, provided by DGC to the Customer under this Agreement, against physical defects for a period of ninety (90) days from the date of shipment by DGC. DGC will replace defective media at no charge to you, provided it is returned postage prepaid to DGC within the ninety (90) day warranty period. This shall be your exclusive remedy and DGC's sole obligation and liability for defective media. This limited media warranty does not apply if the media has been damaged by accident, abuse or misuse.

5. DISCLAIMER OF WARRANTY
EXCEPT FOR THE LIMITED MEDIA WARRANTY NOTED ABOVE, DGC MAKES NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY AND FITNESS FOR PARTICULAR PURPOSE ON ANY OF THE PUBLICATIONS, CLI MACROS OR MATERIALS SUPPLIED HEREUNDER.

6. LIMITATION OF LIABILITY
A. CUSTOMER AGREES THAT DGC'S LIABILITY, IF ANY, FOR DAMAGES, INCLUDING BUT NOT LIMITED TO LIABILITY ARISING OUT OF CONTRACT, NEGLIGENCE, STRICT LIABILITY IN TORT OR WARRANTY SHALL NOT EXCEED THE CHARGES PAID BY CUSTOMER FOR THE PARTICULAR PUBLICATION OR CLI MACRO INVOLVED. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO CLAIMS FOR PERSONAL INJURY CAUSED SOLELY BY DGC'S NEGLIGENCE. OTHER THAN THE CHARGES REFERENCED HEREIN, IN NO EVENT SHALL DGC BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER, INCLUDING BUT NOT LIMITED TO LOST PROFITS AND DAMAGES RESULTING FROM LOSS OF USE, OR LOST DATA, OR DELIVERY DELAYS, EVEN IF DGC HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY THEREOF; OR FOR ANY CLAIM BY ANY THIRD PARTY.

B. ANY ACTION AGAINST DGC MUST BE COMMENCED WITHIN ONE (1) YEAR AFTER THE CAUSE OF ACTION ACCRUES.

7. GENERAL
A valid contract binding upon DGC will come into being only at the time of DGC's acceptance of the referenced Educational Services Order Form. Such contract is governed by the laws of the Commonwealth of Massachusetts, excluding its conflict of law rules. Such contract is not assignable. These terms and conditions constitute the entire agreement between the parties with respect to the subject matter hereof and supersedes all prior oral or written communications, agreements and understandings. These terms and conditions shall prevail notwithstanding any different, conflicting or additional terms and conditions which may appear on any order submitted by Customer. DGC hereby rejects all such different, conflicting, or additional terms.

8. IMPORTANT NOTICE REGARDING AOS/VS INTERNALS SERIES (ORDER #1865 & #1875)
Customer understands that information and material presented in the AOS/VS Internals Series documents may be specific to a particular revision of the product. Consequently user programs or systems based on this information and material may be revision–locked and may not function properly with prior or future revisions of the product. Therefore, Data General makes no representations as to the utility of this information and material beyond the current revision level which is the subject of the manual. Any use thereof by you or your company is at your own risk. Data General disclaims any liability arising from any such use and I and my company (Customer) hold Data General completely harmless therefrom.

Managing TCP/IP
on the DG/UX™
System

093–701051–06

Cut here and insert in binder spine pocket