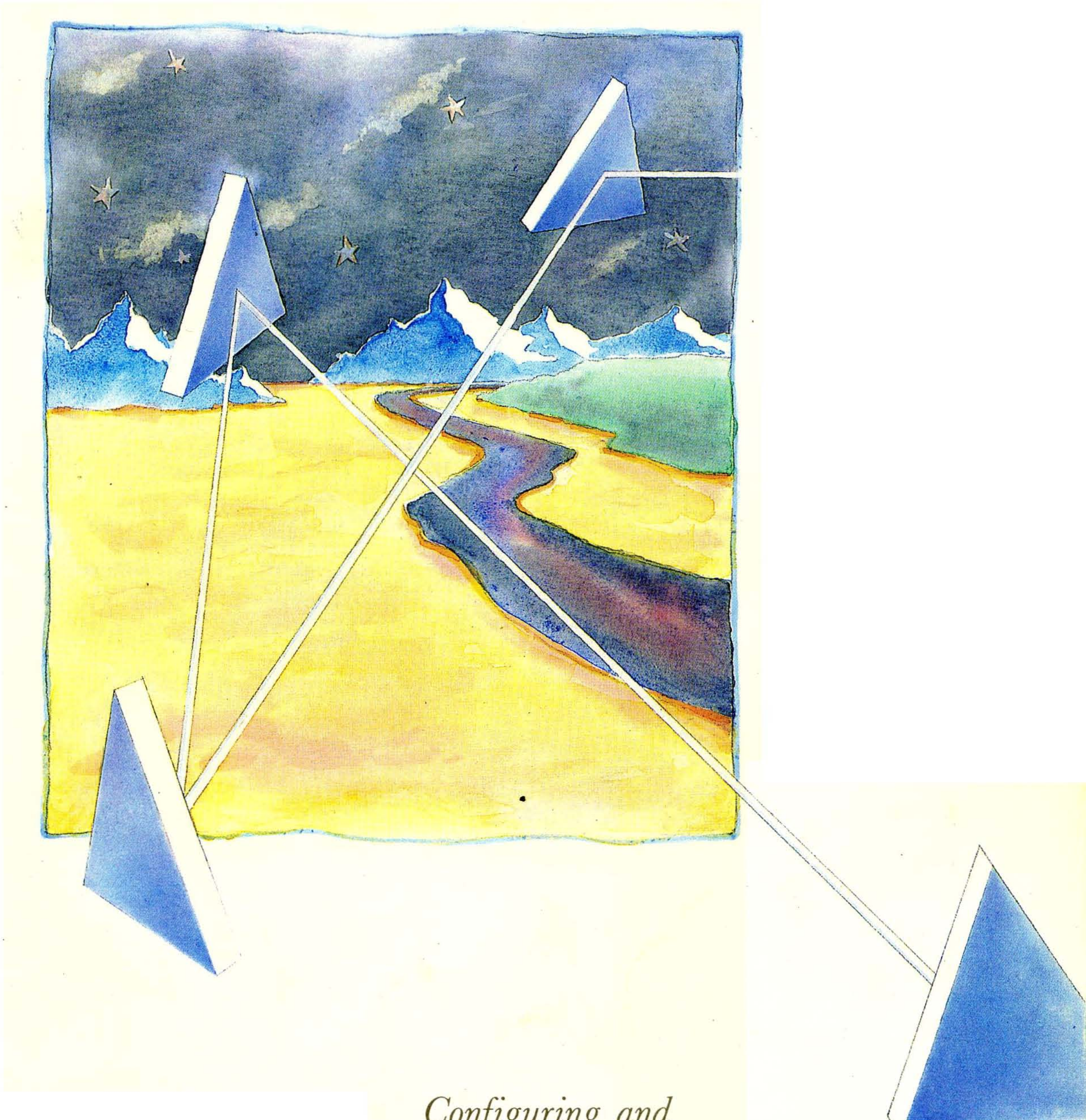


Configuring and Managing TCP/IP

008543-A00

apollo

Configuring and Managing TCP/IP

Order No. 008543-A00

Apollo Computer Inc.
330 Billerica Road
Chelmsford, MA 01824

First Printing: March, 1986
Latest Printing: July, 1988

This document was produced using the Interleaf Technical Publishing Software (TPS) and the InterCAP Illustrator I Technical Illustrating System, a product of InterCAP Graphics Systems Corporation. Interleaf and TPS are trademarks of Interleaf, Inc.

Apollo Computer Inc. reserves the right to make changes in specifications and other information contained in this publication without prior notice, and the reader should in all cases consult Apollo Computer Inc. to determine whether any such changes have been made.

THE TERMS AND CONDITIONS GOVERNING THE SALE OF APOLLO COMPUTER INC. HARDWARE PRODUCTS AND THE LICENSING OF APOLLO COMPUTER INC. SOFTWARE PROGRAMS CONSIST SOLELY OF THOSE SET FORTH IN THE WRITTEN CONTRACTS BETWEEN APOLLO COMPUTER INC. AND ITS CUSTOMERS. NO REPRESENTATION OR OTHER AFFIRMATION OF FACT CONTAINED IN THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO STATEMENTS REGARDING CAPACITY, RESPONSE-TIME PERFORMANCE, SUITABILITY FOR USE OR PERFORMANCE OF PRODUCTS DESCRIBED HEREIN SHALL BE DEEMED TO BE A WARRANTY BY APOLLO COMPUTER INC. FOR ANY PURPOSE, OR GIVE RISE TO ANY LIABILITY BY APOLLO COMPUTER INC. WHATSOEVER.

IN NO EVENT SHALL APOLLO COMPUTER INC. BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING BUT NOT LIMITED TO LOST PROFITS) ARISING OUT OF OR RELATING TO THIS PUBLICATION OR THE INFORMATION CONTAINED IN IT, EVEN IF APOLLO COMPUTER INC. HAS BEEN ADVISED, KNEW OR SHOULD HAVE KNOWN OF THE POSSIBILITY OF SUCH DAMAGES.

THE SOFTWARE PROGRAMS DESCRIBED IN THIS DOCUMENT ARE CONFIDENTIAL INFORMATION AND PROPRIETARY PRODUCTS OF APOLLO COMPUTER INC. OR ITS LICENSORS.

Preface

Configuring and Managing TCP/IP provides network system administrators and knowledgeable users with background information about the TCP/IP protocols and also with detailed instructions about configuring and managing TCP/IP Domain® hosts and Internets built around Domain systems. This manual assumes that readers are familiar with Domain network administration.

Manual Organization

This manual is organized as follows:

Chapter 1 Provides an introduction to the TCP/IP protocols, discusses background material about TCP/IP, and gives an example of a TCP/IP network configuration.

Chapter 2 Describes how to select TCP/IP Internet addresses.

Chapter 3 Discusses network configuration issues, including the formats of the TCP/IP administrative files and how to create them, the TCP/IP daemons and how to invoke them, and how to configure a TCP/IP network as well as single hosts or gateways.

Chapter 4 Describes the name server daemon, **named**, and its database files, and explains how to configure the network to use **named**.

Chapter 5 Provides maintenance and troubleshooting information about TCP/IP Internets.

Appendix A Contains a template of the **/etc/rc.local** startup file and describes the items in the file and how to change them.

Appendix B Describes TCP/IP error messages and lists the Apollo® TCP/IP implementation constants.

Appendix C Describes the contents of the **named** database files and the Standard Resource Record Format used in **named** files.

Appendix D Provides descriptions of all the TCP/IP system administration commands and utilities mentioned in this book. It is intended for users who operate in the Aegis environment.

Appendix E Provides descriptions of the formats for all the TCP/IP files mentioned in this book. It is intended for users who operate in the Aegis environment.

Related Manuals

The file **/install/doc/apollo/os.v.***latest software release number*__**manuals** lists current titles and revisions for all available manuals.

For example, at SR10.0 refer to **/install/doc/apollo/os.v.10.0__manuals** to check that you are using the correct version of manuals. You may also want to use this file to check that you have ordered all of the manuals that you need.

(If you are using the Aegis™ environment, you can access the same information through the Help system by typing **help manuals.**)

Refer to the *Domain Documentation Quick Reference* (002685) and the *Domain Documentation Master Index* (011242) for a complete list of related documents.

The system administrator's manuals for each Domain/OS environment are listed below. We refer to these titles collectively as *Managing System Software* manuals.

- *Managing SysV System Software* (10851)
- *Managing Aegis System Software* (10852)
- *Managing BSD System Software* (10853)

The programmer's and command reference manuals for each Domain/OS environment are listed below. We refer to these titles collectively as *Command Reference* and *Programmers Reference* manuals.

- *SysV Command Reference* (005798)
- *SysV Programmer's Reference* (005799)

- *BSD Command Reference* (005800)
- *BSD Programmer's Reference* (005801)

- *Aegis Command Reference* (002547)
- *Domain/OS Call Reference* (007196)

For more information on TCP/IP programming and applications, see

- *Using TCP/IP Network Applications* (008667)

For more information about Domain networks, see

- *Planning Domain Networks and Internets* (009916)
- *Managing Domain/OS and Domain Routing in an Internet* (005694)

For information about installing software, see *Installing Software with Apollo's Release and Installation Tools* (008860).

Problems, Questions, and Suggestions

We appreciate comments from the people who use our system. To make it easy for you to communicate with us, we provide the Apollo Product Reporting (APR) system for comments related to hardware, software, and documentation. By using this formal channel, you make it easy for us to respond to your comments.

You can get more information about how to submit an APR by consulting the appropriate Command Reference manual for your environment (Aegis, BSD, or SysV). Refer to the **mkapr** (make apollo product report) shell command description. You can view the same description online by typing:

$ **man mkapr** (in the SysV environment)

% **man mkapr** (in the BSD environment)

$ **help mkapr** (in the Aegis environment)

Alternatively, you may use the Reader's Response Form at the back of this manual to submit comments about the manual.

Documentation Conventions

You can perform the tasks described in this book in any Domain/OS environment. We use the dollar sign ($) to indicate the shell prompt; however, $ does not imply a particular shell type. With few exceptions, the commands shown after the shell prompt work in all Domain/OS shells. We specify commands for particular Domain/OS environments when necessary.

Unless otherwise noted in the text, this manual uses the following symbolic conventions.

literal values Bold words or characters in formats and command descriptions represent commands or keywords that you must use literally. Pathnames are also in bold. Bold words in text indicate the first use of a new term.

user–supplied values Italic words or characters in formats and command descriptions represent values that you must supply.

sample user input In examples, information that the user enters appears in color.

`output` Information that the system displays appears in this typeface.

[ ] Square brackets enclose optional items in formats and command descriptions.

| A vertical bar separates items in a list of choices.

. . . Horizontal ellipsis points indicate that you can repeat the preceding item one or more times.

——— ⊞ ——— This symbol indicates the end of a chapter.

Contents

Chapter 1 Introduction to TCP/IP

Chapter 2 Selecting Internet Addresses

Chapter 3 Configuring a TCP/IP Network

Chapter 4 Configuring the Name Server

Chapter 5 Managing and Troubleshooting a TCP/IP Internet

Appendix A The /etc/rc.local Startup File

Appendix B TCP/IP Implementation Constants and Error Messages

Appendix C Named Database Files and the Standard Resource Record Format

Appendix D TCP/IP System Administration Commands and Utilities

Appendix E TCP/IP File Formats

Glossary

Index

Figures

Tables

Procedures

Chapter 1

Introduction to TCP/IP

The Transmission Control Protocol (TCP) and the Internet Protocol (IP), commonly referred to as TCP/IP, provide services that allow different computers to communicate with each other. This chapter provides an introduction to TCP/IP communications concepts. While it is not an introduction to networking, it does explain some of the concepts required to understand TCP/IP. This chapter concludes with a sample TCP/IP internetwork configuration.

1.1 Overview of TCP/IP

TCP/IP is a standard protocol, originally developed by the Defense Advance Research Projects Agency (DARPA). It works on various types of computers so users can share the resources among many different machines. The most common applications that use TCP/IP communications are remote log in and file transfer.

The TCP/IP protocols were designed to provide communication services over a variety of physical networks — from computer networks to radio networks. TCP/IP can provide this broad communications service by defining protocols for *how* to send and receive messages, but does not define *what* the physical devices must do to send and receive the messages. By leaving the device details to those who want to implement TCP/IP, computers on numerous types of networks can use TCP/IP to communicate with each other.

Many computer manufacturers use TCP/IP as a way to communicate with competitors' computer systems because it is an industry–wide protocol.

Our implementation of TCP/IP follows the BSD UNIX* model, although TCP/IP operations are supported in all three Domain environments – Aegis, BSD, and SysV. You can use TCP/IP in all three environments as a communication link between various operating sys-

* UNIX is a registered trademark of AT&T in the USA and other countries.

tem utilities. When you install TCP/IP, you can perform the following functions on your Domain network:

BSD Utility	**Allows users to:**
ftp	Access the File Transfer Protocol, which lets you transfer files to and from a remote network site.
lpr	Queue and print files.
rcp	Copy files between machines.
rexec	Return a stream to a remote command.
rlogin	Log in to a remote host.
rsh	Execute a shell command on a remote host.
ruptime	Get status of UNIX computers on local network. The status information is by host and includes the time since the system last came up or went down and the number of users logged in.
rwho	Determine who is logged in on all machines in the local network.
telnet	Access the Telnet Protocol, which lets you communicate with another host.

For more information about using these utilities, see the manual *Using TCP/IP Network Applications*.

TCP/IP enables Apollo workstations on Domain internets to communicate with computers on many other types of computer networks as long as the remote devices also support TCP/IP. There are many network configurations in which TCP/IP can be used to provide the basic communication protocols. Some configurations may require special hardware. Contact your sales representative for information about our most current hardware products. For more information about creating a Domain internet and connecting to heterogeneous networks and internets, see *Planning Domain Networks and Internets* and *Managing Domain/OS and Domain Routing in an Internet*.

This book provides procedures to configure and manage the software in a Domain TCP/IP network. The procedures explain how to:

- Select unique addresses for each node in your Domain TCP/IP network or internet
- Create the files TCP/IP requires for name/Internet address resolution and routing
- Configure TCP/IP hosts and gateways

- Update and maintain your Domain TCP/IP network or internet once you've installed it
- Locate problems within your Domain TCP/IP network or internet

1.2 TCP/IP Gateways and Hosts

A TCP/IP **gateway** node contains the appropriate software and hardware to provide the physical link between different networks, allowing computers in separate networks to communicate with each other via TCP/IP. Computers that communicate with one another via TCP/IP communications are called TCP/IP **hosts**.

Gateways have physical interfaces on two or more networks and can route a message from one network to the other. Figure 1-1 shows how a gateway routes information. (Figure 1-1 shows the usual network software layering scheme, but only the layers relevant to this discussion are indicated.) For a computer in Network A to communicate with a computer in Network B via TCP/IP, data from Network A must pass through the gateway on its way to Network B.

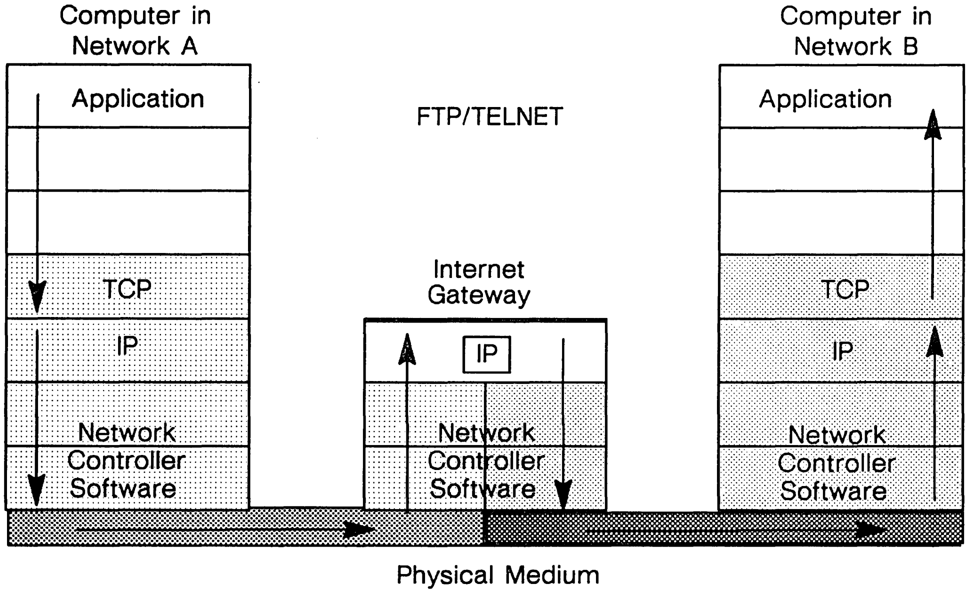


Figure 1-1. Internet Gateway Layers

For our TCP/IP product, a TCP/IP gateway node can be any node containing at least two network interfaces and the TCP/IP software. A TCP/IP host is any node that has TCP/IP software to communicate with hosts on other computer networks. That is, the gateway performs the physical and routing functions to connect the networks and the host provides the applications such as FTP or Telnet. A node can be both a gateway and a host as long as it provides the hardware and software for both.

1.3. TCP/IP Internets

Although two different networks may be physically connected by a TCP/IP gateway, both networks must follow certain protocols in order to communicate. The TCP/IP protocols, originally developed by DARPA, control how networks assign addresses and route messages through the gateways. You can connect your network with any other network that conforms to the DARPA TCP/IP standards. By connecting your network to other DARPA-conforming networks, you are creating a TCP/IP internet.

Note the difference between a Domain internet and a TCP/IP internet: A Domain internet refers to a network of networks all running the Domain distributed environment. The physical networks that can comprise a Domain internet include the Apollo Token Ring and the IEEE 802.3 network (commonly referred to as an ETHERNET* network). Apollo workstations in a Domain internet usually communicate with each other using Domain protocols and with other vendors' systems using non-Domain protocols such as TCP/IP. A TCP/IP Internet is a network of several vendors' networks, and computers within the internet communicate via TCP/IP.

One of the largest TCP/IP internets to which you can connect your network is the nationwide network called the ARPANET. The ARPANET, the first large scale network using TCP/IP, was developed by the Department of Defense (DoD) and Bolt, Baranek and Newman (BBN).

The Domain TCP/IP product conforms to DARPA internet standards. It also supports utilities defined by standard BSD UNIX. Therefore, you can connect your Apollo workstations running TCP/IP to non-Apollo equipment running TCP/IP and located either on your Domain network or internet or on other vendors' networks.

We describe the DARPA standards in detail in Chapter 2, "Selecting Internet Addresses." For specific information about TCP/IP internets, contact the Network Information Center (NIC) at SRI International. The NIC maintains specifications and detailed information about the TCP/IP protocols.

* ETHERNET is a registered trademark of Xerox Corporation.

1.4. Example of TCP/IP Configuration

Figure 1–2 illustrates a simple TCP/IP implementation in a Domain internet that is comprised of two Apollo Token Ring (ATR) networks and one IEEE 802.3 network.

In this figure, the Apollo workstations on the ATR and IEEE 802.3 networks communicate with each other using Domain protocols and with the VAX* computers using TCP/IP.

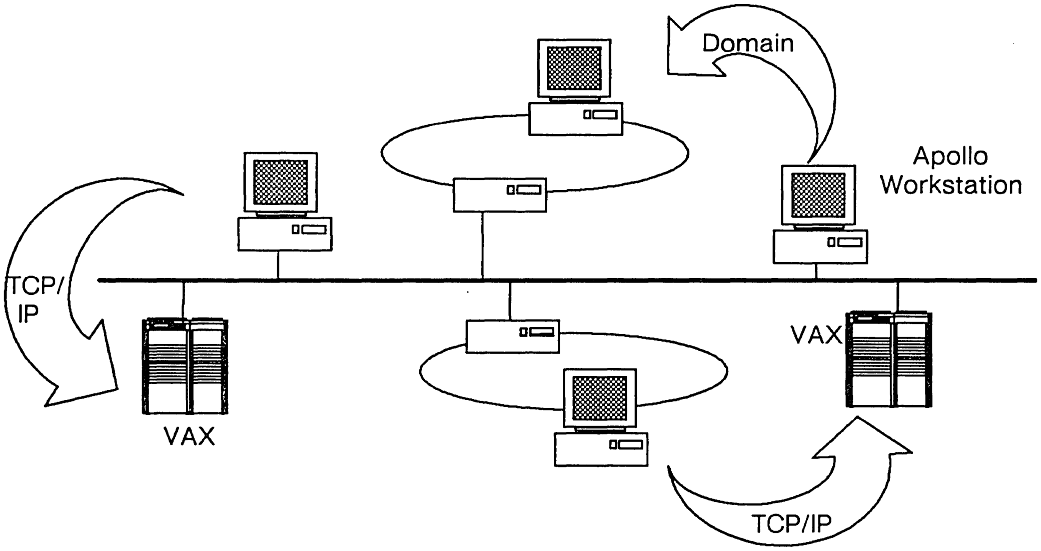


Figure 1–2. Domain Internet Using Domain and TCP/IP Protocols

* VAX is a registered trademark of Digital Equipment Corporation.

Chapter 2

Selecting Internet Addresses

This chapter explains the format of Internet names and addresses and describes the process of selecting Internet addresses for all the nodes using TCP/IP on your network system.

Note that most of the information in this chapter is helpful if you are configuring TCP/IP on an entire network for the first time. If you are familiar with TCP/IP, or if you are updating an established TCP/IP configuration, go on to the configuration procedures in Chapter 3, "Configuring TCP/IP."

2.1 Drawing the Internet

To begin configuring TCP/IP, you must first determine which nodes in your network will use TCP/IP. Then you must select Internet Protocol (IP) addresses for each node. The **Internet address** allows communication between computers on different physical networks by providing a standard addressing mechanism that all the computers can understand.

As a first step, draw a picture of the network (or internet) you are configuring. (See Figure 2–1 for an example.) Decide which nodes will run TCP/IP as hosts and as gateways. Any node on which users run TCP/IP applications or applications that require TCP/IP, such as The X Window System* or NFS, should be considered a TCP/IP host. Any node which routes TCP/IP packets between networks is called a TCP/IP gateway node. To operate as a TCP/IP gateway, a node must have the necessary network controller boards to physically interface with the networks to which it is connected. In addition to helping you configure TCP/IP addresses, drawing this picture will help you later when you are trying to locate communication problems within your network.

Figure 2–1 shows a network configuration consisting of three networks, two Apollo Token Ring networks and one IEEE 802.3 network. In the figure, the TCP/IP host name for the

* The X Window System is a trademark of MIT–Project Athena.

workstation on Ring A is HostA and the gateway's name is HostB. On Ring B, the TCP/IP host name for the workstation is HostC and the gateway's name is HostD. On the IEEE 802.3 network, the workstation host names are HostE and HostF and the computer host names are VAX01 and VAX02.

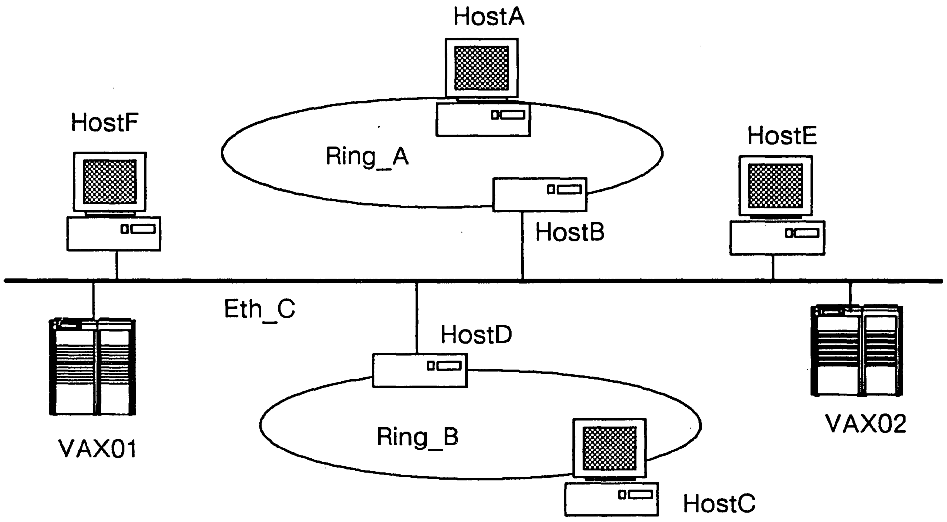


Figure 2-1. Drawing an Internet

After drawing the picture of your network and deciding which nodes are hosts and which nodes are gateways, you can assign names and addresses to each host and gateway. These names and addresses must follow the standard DARPA formats described in the following sections.

2.2 Selecting Internet Names and Addresses

Whenever you refer to an object on the network, whether it is a host or a file, you usually use a name because names are easy to remember. The operating system, however, converts names to addresses because addresses are more meaningful to it. For example, you refer to your Domain node by a name such as //HostA while the operating system refers to it by an address such as 06d49.

2.2.1 Internet Naming Conventions

The following naming conventions should be used when assigning Internet names:

- The name must begin with an alphabetic character

- The name can be up to 32 characters long

- The name cannot contain embedded spaces or comment characters (#)

- Valid characters include A – Z, 0 – 9, period (.), underscore (_), and hyphen (–). Both upper and lower case characters are acceptable

For simplicity, on Domain networks, we recommend that the TCP/IP host name be the same as the node name without the slashes. (In fact, the TCP/IP software uses the node name as the default TCP/IP hostname.) For example, the TCP/IP host name for the node //HostA can be HostA. (Note that you cannot include slashes in TCP/IP host names.)

Within Domain networks using the Domain internet routing service, you can transfer messages simply by specifying the local name. (For information about Domain routing, see *Managing Domain/OS and Domain Routing in an Internet.)* However, to communicate with hosts on Domain networks using the TCP/IP protocols or with foreign networks, you need an additional addressing layer. For TCP/IP, this is the **Internet address.**

2.2.2 Format of the Internet Address

A typical Internet address consists of two fields; the left field (or the **network number**) identifies the network, and the right field (or the **host number**) identifies the particular host within the network.

The DARPA Internet address is 32–bits long and can be interpreted differently to accommodate networks of varying sizes. The Type A address allows you to have 256 networks, each with many hosts (up to 16,777,214). Type B only allows a network to have up to 65,534 hosts, but it allows you to have 65,534 physical networks. Type C allows you to have millions of physical networks with up to 254 hosts on each.

You would choose to use a particular address type depending on the number of subnets within your internetwork and the number of hosts on each subnet. You can recognize a type by the value of the Most Significant Bits (MSB) or the leftmost bits in the address. For example:

- Type A addresses have a 7–bit network number, a 24–bit host number, and the MSB is 0.

- Type B addresses have a 14–bit network number, a 16–bit host number, and the two MSB's are 10.

- Type C addresses have a 21–bit network number, an 8–bit host number, and the three MSB's are 110.

Figure 2–2 shows how a 32–bit Internet address is divided into network and host numbers. It also shows how the most significant bits (MSB) in each network number identify the address type.

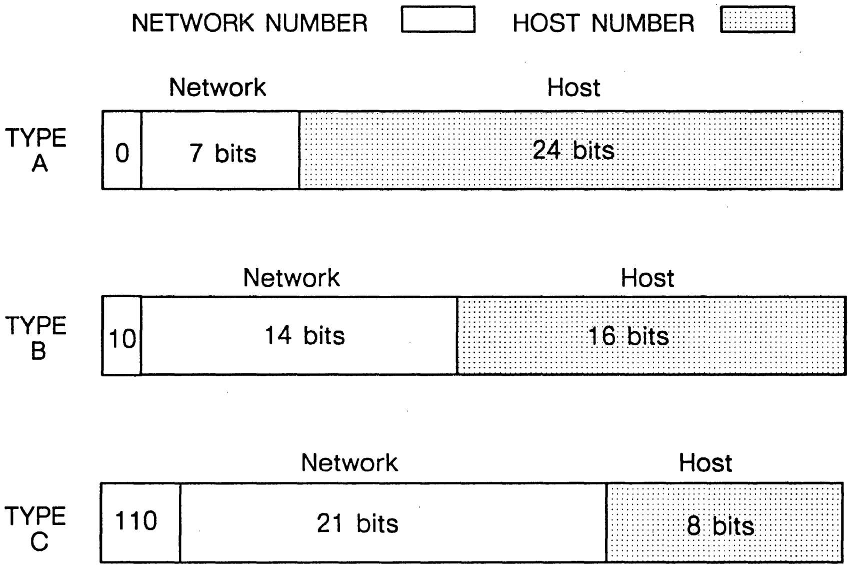


Figure 2–2. Type A, B, and C Internet Addresses

When selecting Internet addresses for your network, you don't need to calculate the size of the network and host fields. Instead, after choosing the type of address you want to use, you simply supply decimal numbers within a specific range.

> **NOTE:** You must supply *decimal numbers* to conform to the DARPA Internet addressing standard format.

The standard DARPA Internet addressing format is:

W.X.Y.Z

where W, X, Y, and Z are decimal numbers between 0 and 255. Each of these decimal numbers represents one byte of the Internet address. The four bytes together represent both the network and host address. However, which numbers refer to the network and which numbers refer to the host depends on the Internet address type (Type A, B, or C).

For example, Type C addresses have a one–byte host address so your host number can be any number within the range of 1 and 254. (DARPA reserves numbers 0 and 255.) Type C addresses have a 21–bit network address and a 3–bit MSB, so the network number will be 3 bytes long; and can fall within the range of 192.0.1 and 223.255.254. The number starts after 192 because the first three bits (0 through 192 in decimal) are reserved to signify the address type.

Table 2–1 summarizes the ranges you can specify for network and host numbers of each type. By using this table to select numbers, you also avoid using the Internet addresses that DARPA reserves for its own use. For example, DARPA reserves network and host numbers that have a value of zero (all four numbers are 0) and a value of one (all four numbers have the decimal value of 255). It also reserves Type C network numbers greater than 223.255.254. If you use reserved numbers, TCP/IP might generate errors.

Table 2–1. Ranges of Values for Type A, B, and C Internet Addresses

Address Type	Size in Bytes		Decimal Number Range of Values	
	Network	**Host**	**Network Portion**	**Host Portion**
A	1	3	1 - 255	0.0.1 - 255.255.254
B	2	2	128.1 - 191.254	0.1 - 255.254
C	3	1	192.0.1 - 223.255.254	1 - 254

2.2.3 Creating Internet Addresses with Subnet Numbers

In addition to selecting Internet addresses that consist of your network and host numbers, you can also designate an intermediate number called a **subnet** number. Using subnets allows you to effectively extend the network field of the internet address beyond the limit defined by the Type A, B, C scheme. Subnets let you set up a hierarchy of Internet addresses within your network. That is, you can have one network number for your entire internet, and various subnet numbers for each network within your internet. TCP/IP treats the network and subnet fields together as the network portion.

The following example illustrates the advantage of having subnet numbers. Consider two hosts on the ARPANET — one at the University of Southern California (USC) and the other at Massachusetts Institute of Technology (MIT). Since both hosts are part of a large campus internet that consists of numerous networks, sending messages is complicated without subnet numbers. To send a message from the USC host to the MIT host, the USC sender must know the specific network within the internet at MIT. That is, the sender at USC must know the network topology of the receiver at MIT. Moreover, if the MIT network changes, the USC sender might need to learn a *new* network address.

If the two colleges assign subnet numbers, sending messages between them is easy. The USC host sends a message to the MIT host simply by specifying an Internet address whose network number represents the entire MIT internet. When the message reaches the MIT gateway, the gateway checks whether subnets are implemented, and if so, relays the message to the appropriate network within the MIT internet.

To create subnets on your internet, you use the same Internet address format but you cause it to be interpreted differently. The 4–byte Internet address represents a network, subnet, and host number rather than representing the network and host number. Note that the size of the network number remains the same. You create a subnet by dividing the host number into a subnet and host number.

Figure 2–3 shows some possible ways you can subdivide an Internet address into network, subnet, and host numbers. You can actually subdivide it any way you want depending on the number of subnets (networks within the internet) and hosts you have.

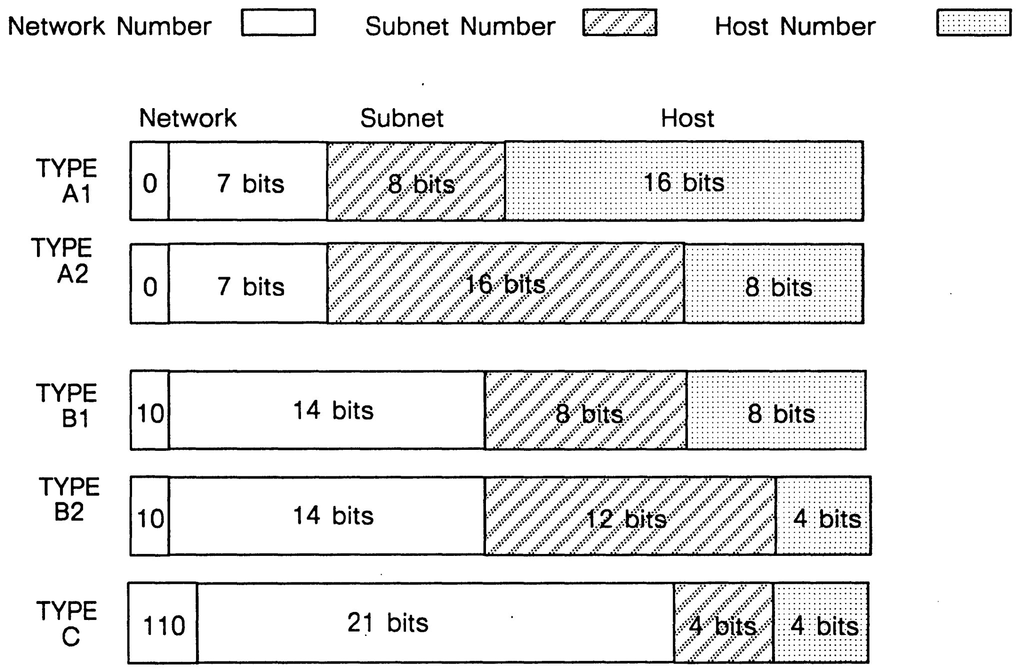


Figure 2–3. Internet Addresses with Subnet Numbers

To create a subnet, you subdivide the host portion of your Internet address. Table 2–2 lists the range of subnet and host values for each type. Note that since Type C host numbers are only 8–bits long, you're limited to 15 subnets and 14 hosts. For this reason, most users implement subnets with Type A or B addresses.

Table 2–2. Range of Subnet and Host Values for Type A, B, and C Addresses

Address Type	Size in Bits		Decimal Number Range of Values	
	Subnet	**Host**	**Subnet Portion**	**Host Portion**
A1	8	16	1 - 255	0.1 - 255.254
A2	16	8	0.1 - 255.255	1 - 254
B1	8	8	1 - 255	1 - 254
B2	12	4	1 - 255	1 - 240
C	4	4	1 - 15	1 - 14

As we stated earlier, using subnets does not change the Internet address format. Instead, you are changing how TCP/IP *interprets* the Internet address. You do so by supplying a bit mask or subnet mask to each node, which tells TCP/IP that your network system uses subnets, and which part of the Internet address corresponds to the subnet numbers. This subnet mask is supplied to TCP/IP nodes by using the **/etc/ifconfig** command within the node's startup file, **/etc/rc.local**. We describe **/etc/ifconfig** in more detail in Chapter 3 of this manual and **/etc/rc.local** in Appendix A. The following section describes how to specify a subnet mask.

To understand how to divide an internet into TCP/IP subnets, refer to Table 2–3. The table lists the Internet addresses corresponding to Figure 2–1. To assign the Internet addresses, we first assigned a network number to correspond to the entire Domain internet. Then we assigned a subnet number to each network within the internet.

Even if the TCP/IP gateways will not be used as TCP/IP hosts, they should be listed as hosts in Table 2–3. We inserted *nnnn* for the local names of non–Apollo hosts on the IEEE 802.3 network.

To ensure that you understand how to assign these addresses, pencil in the addresses from Table 2–3 onto Figure 2–1.

2.2.4 Specifying Subnet Masks

If your network system contains subnets, you must supply a bit mask or subnet mask with an Internet address to indicate to gateways how they should interpret the address. The subnet mask identifies which parts of the Internet address correspond to a subnet number and which parts correspond to the host number.

Subnet masks are set with the **/etc/ifconfig** command, which also specifies the node's Internet address. The subnet mask is specified as a single hexadecimal number with a leading 0x (for example, 0xffffff00) or as a dot–notation Internet address (w.x.y.z). The mask contains 1's for the bit positions in the 32–bit address which are to be used for the network and subnet parts and 0's for the host part. The mask value must be preceded by the keyword **netmask**. For a type B address where the first two bytes indicate the network number, the third byte indicates the subnet number, and the fourth byte is the host number, the mask would be specified with the **inconfig** command in one of two ways:

```
/etc/ifconfig dr0 129.9.6.1 netmask 0xffffff00
/etc/ifconfig dr0 129.9.6.1 netmask 255.255.255.0
```

The node's Internet address and, if necessary, subnet mask, usually are specified in the node's **/etc/rc.local** file. See Appendix A of this manual for a description of the **/etc/rc.local** file.

Table 2–3. Internet Addresses for Sample Subnet Configuration

			Internet Number:		
Object Type	**Local Name**	**Internet Name**	**Network**	**Subnet**	**Host**
Domain Internet			129.9.	0.	0
Network	Ring A		129.9.	1.	0
Network	Ring B		129.9.	2.	0
Network	Eth C		129.9.	3.	0
Gateway	//HostB	HostB	129.9.	1.	23
		HostB	129.9.	3.	23
Gateway	//HostD	HostD	129.9.	2.	25
		HostD	129.9.	3.	25
Host	//HostA	HostA	129.9.	1.	21
Host	//HostE	HostE	129.9.	3.	21
Host	//HostF	HostF	129.9.	3.	12
Host	//HostC	HostC	129.9.	2.	3
Host	*nnnn*	VAX01	129.9.	3.	221
Host	*nnnn*	VAX02	129.9.	3.	222
Host	//HostB	HostB	129.9.	1.	23
		HostB	129.9.	3.	23
Host	//HostD	HostD	129.9.	2.	25
		HostD	129.9.	3.	25

2.2.5 Creating Internet Addresses for Internets without Subnet Numbers

You can configure TCP/IP for your internet *without* implementing subnets. You must first decide what type of addressing to use, and then assign each network a different network number. Hosts and gateways are assigned internet addresses based on the their network number.

If you **ever** plan to use TCP/IP within a DARPA Internet, you should request a network number assignment from the Network Information Center (NIC) run by SRI International. This will avoid renumbering your networks and changing host and gateway internet addresses in the future.

2.2.6 Assigning Internet Addresses

Now that you know the format of Internet numbers, you can select them for each TCP/IP host and gateway in your network. The following procedure provides step–by–step instructions on how to select and assign Internet addresses.

Procedure 2-1. *Assigning Internet Addresses*

This procedure is relevant for all networks using TCP/IP, whether or not they will connect to the ARPANET.

Task 1: Make a List of Host Names

Select a host name for each host on the network. For Domain hosts, use the node name without the slashes; for example, HostA. (The TCP/IP software uses the node name as a default host name.) If you have a diskless host without a host name, catalog the diskless node in the root directory and uncatalog it as diskless_$<nodeid>. See the appropriate *Managing System Software* manual for information on cataloging nodes. For non-Domain hosts, use any appropriate mnemonic host name.

Host names must start with an alphabetical character and can include any of the following: A-Z, 0-9, period (.), underscore (_), and hyphen (-). They can have up to 32 characters and both upper and lower case characters are acceptable (host names are not case-sensitive). Slashes and embedded spaces are not permitted in Internet host names.

You can assign more than one name to a single host or gateway. These additional names are called **aliases.** You might use aliases when a node serves as both a gateway and a host, or when you want to identify hosts according to their networks. For example, you could assign the node //HostB to have the Internet name HostB and the aliases, HostB.Gate and HostB.Network. TCP/IP primary host names and aliases must be included in the network's **/etc/hosts** file.

Task 2: Decide on Type A, B, or C Internet Address Format

Decide on the type of Internet address you want, Type A, B, or C. If you have a large number of hosts and a few networks, select Type A or B. If you have many networks and fewer hosts, select Type C.

Note that if you plan to use TCP/IP to communicate within a DARPA Internet such as ARPANET, you must apply for a network number from the Network Information Center (NIC) at SRI International. They will usually assign a Type B address if you plan to implement subnets. Otherwise, they will provide you with a Type C address.

Apply to NIC for a network number if you *ever* intend to attach your network to the DARPA Internet, even if you do not initially intend to do so. This way, you won't have to change your host addresses when you start using the DARPA Internet.

Task 3: Select a Network Number

If you do not plan to request a number from NIC, select a network number that will be unique across all interconnected networks. Note that the size of the network number depends on the type of Internet address format you selected.

If you are implementing subnets within a Domain internet, choose a network number to represent the internet as a whole. Individual networks within the internet share the same network number but have different subnet and host numbers.

Task 4: Select Subnet Numbers

If you are implementing subnets within a Domain internet, select a unique subnet number for *each* network within the internet.

Task 5: Assign Internet Addresses for Each Gateway

On your network drawing, assign Internet addresses to each gateway in the network. For TCP/IP purposes, a gateway connects two different networks. TCP/IP addresses must be included in the network's **/etc/hosts** file. See the following chapter for more information about how Internet addresses actually are assigned to hosts and gateways.

Note that you must assign *two* Internet addresses to gateway nodes since they belong to more than one network. For example, the node HostB is on Net A and Eth C, so it should be assigned two Internet addresses, 129.9.1.23 and 129.9.3.23.

Task 6: Assign Internet Addresses for Each Host

On your network drawing, assign an Internet address for each host. When you record the Internet addresses, remember that they are expressed in *decimal,* not *hexadecimal.* Each host on the same network must have the same network number, but a different host number. You should list gateways as both hosts **and** gateways.

As with gateways, host TCP/IP addresses must be included in the network's **/etc/hosts** file.

Chapter 3

Configuring a TCP/IP Network

This chapter discusses issues important to system administrators who configure and administer TCP/IP networks. Topics covered include:

- The TCP/IP files and their locations
- Methods of TCP/IP Internet name–address resolution and how to choose the appropriate method for your network system
- The TCP/IP daemons, or server processes, and how to invoke them
- Procedures to configure a TCP/IP network, to configure a single user's TCP/IP node, and to test communications on the network

If you are a **system administrator** who is familiar with the concepts discussed in the first parts of this chapter, you may wish to proceed directly to Procedure 3–1 which describes the tasks required to configure an entire network for TCP/IP.

If you are an **individual user** who is configuring TCP/IP on your own node, you may wish to proceed directly to Procedure 3–4 ,which describes how to configure a single TCP/IP host or gateway node.

3.1 TCP/IP Configuration Files

The TCP/IP files discussed in this section fall into two categories: **administrative files,** which contain information that must be the same for all hosts on the network, such as Internet name to Internet address mapping; and **local files,** which contain information that is specific to an individual node. Administrative files reside on the network's TCP/IP administrative node and are linked to by the other TCP/IP hosts and gateways on that network. Local files reside on each TCP/IP node.

Table 3–1 briefly describes all the TCP/IP configuration files and their locations. More detailed descriptions and •formats of the TCP/IP **administrative** files follow the table. See Appendix A for more information about the local startup file, **/etc/rc.local**; see Section 3.3.3, "inetd," for information about **/etc/inetd.conf**; and see Chapter 4 and Appendix D for information about the UNIX name server (**named**) database files.

Table 3–1. TCP/IP Configuration Files and Their Locations

TCP/IP File	**Location**	**Description**
/etc/hosts	On administrative host, with links on all other TCP/IP nodes.	Administrative file that relates host names and Internet addresses.
/etc/networks	On administrative host, with links on all other TCP/IP nodes.	Administrative file that associates Internet network addresses to network names for all accessible networks.
/etc/gateways	On TCP/IP gateways. Or on administrative host with links on gateways.	Administrative file that contains static routes to be loaded into static routing tables.
/etc/hosts.equiv	On administrative host, with links on all other TCP/IP nodes.	Administrative file that lists equivalent hosts for log–in purposes.
/etc/resolv.conf	On administrative host with links on each TCP/IP host not running UNIX name server, **named**, locally.	Administrative file for networks using **named**. Is pointer file to remote servers for hosts not running **named** locally.
/etc/rc.local	On each TCP/IP host. This file is a link to **'node_data/etc/rc.local**.	Local start–up files for node. Usual way to start TCP/IP server processes.
/etc/inetd.conf	On each TCP/IP host. This file is a link to **'node_data/etc/inetd.conf**.	Local configuration file for **inetd**. Lists all services that can be invoked by **inetd**.
/etc/daemons/<server>	On each TCP/IP host. Link to files of the same name in directory **'node_data/etc/daemons**	Local files that enable each <server> process. Allow users to control what servers operate on their nodes.
/etc/named.boot **/etc/named.ca** **/etc/named.hosts** **/etc/named.rev** **/etc/named.local**	On each TCP/IP host running **named**. Link to files of same name in **'node_data/etc** directory.	Local **named** boot and data files.

3.1.1 File Links

When the TCP/IP software is installed (as part of the standard operating system software), many of the TCP/IP configuration files are installed as some form of link. Local TCP/IP configuration files are installed in the **/etc** directory as links to files of the same name in each node's **'node_data/etc** directory, where they actually reside.

Administrative files are installed as links to the same file on a generic administrative node by means of another link called "tcp_admin." For example, the file **/etc/hosts** is installed as a link that resolves to **tcp_admin/etc/hosts**. Then, as part of the configuration process for each TCP/IP host or gateway, the "tcp_admin" link must be edited to point to the correct TCP/IP administrative node. On administrative nodes, the administrative file links must be **removed** so that the administrative files can be physically located on the administrative nodes.

3.1.2 Administrative Nodes

On Domain networks, TCP/IP administrative nodes are the machines on which the TCP/IP administrative files are physically located. These files provide TCP/IP information that must be the same for all hosts on the network, such as Internet name to address mapping and host equivalencies. TCP/IP hosts and gateways have file system links to the administrative files on the administrative nodes. The use of administrative nodes, made possible because of the Domain distributed file system, greatly simplifies management of TCP/IP networks. File updates only need to be made on the administrative node; hosts and gateways automatically pick up the changes.

You will need to configure at least one TCP/IP administrative node per network. If you have a network system composed of several subnets, each subnet should have its own administrative node. Administrative nodes do not *have* to be TCP/IP hosts themselves, although they can be hosts and gateways also, if you so configure them.

You can have more than one TCP/IP administrative node on a network, but if you do, you must make sure that all administrative nodes on the network always have *identical* information. The advantage of having a single administrative node is that you need to maintain only *one* database. However, you may also need to provide an alternate database in case the primary administrative node crashes or is inaccessible because of a network failure. If the administrative node is not available when a TCP/IP node starts up, the node will not be able to run TCP/IP.

We recommend that you locate at least one primary and one backup administrative node in every network. You must update *every* TCP/IP administrative node on every subnet whenever you change your network configuration.

Note that TCP/IP hosts and gateways can be linked to only *one* administrative node, so you must change the link to change the administrative node.

3.1.3 The /etc/hosts Administrative File

The **/etc/hosts** file contains the primary name, alias, and IP address of every TCP/IP host that can be accessed by name by hosts on your network system. During normal operation, the TCP/IP software uses this file to resolve IP name/address queries, if the UNIX name server, **named**, is not running on the network. In addition, at start-up every TCP/IP host and gateway reads the ASCII version of this file to resolve its own IP address.

The hosts that must be listed in this file include all the hosts on your own network system. You may choose to include all the hosts on remote networks with which hosts on your network will communicate, if your network system does not use **named**. Every time you add or remove hosts from your internet, you must update this file.

Each line has the following format:

```
internet-address    host-name  alias
```

Host names and aliases can contain any printable characters other than field delimiters (blank spaces), newlines, or comment characters (#). See Chapter 2, "Selecting Internet Addresses," for a more complete definition of internet naming conventions. The address and name (and aliases, if used) must be separated by one or more blanks or TAB characters. For example:

```
127.0.0.1          localhost
129.9.3.21         HostA
129.9.3.22         HostB
129.9.3.23         HostC   rodney
129.9.1.1          HostD   alice
```

Include the standard `localhost` entry in order to allow users to access the software loopback interface on each host. The software loopback interface is a physical interface simulator built into the TCP/IP software and is used for troubleshooting. See Chapter 5 for information about using the loopback interface.

Local hosts must be included in this file if their users want to run TCP/IP daemons, described in Section 3.3, "TCP/IP Daemons," or any of the following application programs or library routines: **lpr**, **rcmd**, **rcp**, **rlogin**, **rsh**, **rexec**, **ftp**, or **telnet**. Hosts that use other Apollo products that rely on TCP/IP, such as Domain/Access or NCS, also must be listed in **/etc/hosts**.

If users on your network will communicate with the ARPANET, your local **/etc/hosts** file should contain the names of all the ARPANET hosts (unless you run **named**). See Procedure 3-2 for information about obtaining the ARPA master file from the Network Information Center at SRI, International and creating **/etc/hosts** with the **htable** utility.

If your network uses **/etc/hosts** for Internet name-address resolution, you can use the **mkhosts** utility to generate a hashed database from the **/etc/hosts** file that speeds the name/address translation process. See Section 3.2, "Methods of Internet Name–Address Resolution," and also Procedure 3–2 for information about using **mkhosts**.

3.1.4 The /etc/networks Administrative File

The **/etc/networks** file contains the names and IP internet addresses of networks that can be accessed by hosts on your network. It is used by **route** and other other TCP/IP applications such as **netstat** for converting between network names and network internet addresses. Each entry is a single line of the format:

```
network-name        internet address
```

The two values are separated by one or more spaces or TAB characters. The `network-name` can be any name the describes the network. For example,

```
Network-A           129.9.1
Network-B           129.9.2
Network-C           129.9.3
Network-D           129.9.4
```

This file is created by **htable** at the same time that **/etc/hosts** is created. See Procedure 3–1 for information about using **htable** to create these files.

In a single network environment (where there are no gateways to communicate with other networks) this file consists of one line defining the local network and one line defining the software loopback. For example, in a single IEEE 802.3 network, the file would only contain the following lines:

```
my-network  192.3.5
loopback    127.0.0
```

3.1.5 The /etc/gateways Administrative File

The **/etc/gateways** file contains routing information used by **routed**, the routing daemon. This file contains entries about remote networks and hosts and their gateways that may not support the **routed** protocol. The file is read by the **routed** process on gateways at startup, the information is added to the gateways' internal routing tables, and it remains there permanently. The information also may be sent to other gateways running **routed**.

If you want to create local static routes, use the **route** command rather than this file. Do not create or use an **/etc/gateways** file on a network unless that network contains a passive gateway that does not support the Routing Information Protocol (RIP) and cannot run **routed**. For example, for members of the ARPANET, this file should include information about gateways that use the Exterior Gateway Protocol (EGP) instead of RIP to obtain gateway information. Apollo gateways always support RIP and should not be listed in any **/etc/gateways** files.

Routed propagates the routing information in **/etc/gateways** to other gateways and hosts that run **routed**. Routes and priority routes also can be added to and deleted from a host's routing tables with the **route** command. For more information about **routed** and **route**, see the appropriate *Managing System Software* manual for your environment.

The **/etc/gateways** file can be created by **htable** at the same time that **/etc/hosts** and **/etc/networks** are created. Procedure 3–2 describes using **htable** to create these files.

Each entry in the **/etc/gateway** file is a single line in the following format:

[net|host] name1 **gateway** name2 **metric** hops **[active|passive|external]**

The command line terms are defined as follows:

[net|host] Type of routing destination; enter either **net** or **host**.

name1 Name or IP internet address of the destination **host** or **net.** Must be located in **/etc/hosts** or **/etc/networks.**

gateway A keyword specifier for name2.

name2 Host name or IP internet address of **gateway**. This is the gateway to which the packets should be addressed, and is the next gateway in the route to the destination (the next hop). The name2 gateway must be on the same network as the gateway that uses this file.

metric A keyword specifier for **hops**. If **metric** is not specified, the hop count is 0.

hops Number of gateways data packets must travel through to reach the destination network (the hop count).

[active|passive|external]
Indicates whether the name2 gateway uses **routed. Active** specifies that the gateway is running **routed** to exchange routing information. **Passive** indicates a gateway that does not use **routed** and is not exchanging routing information. **External** indicates that the gateway uses a protocol external to the Routing Information Protocol used by **routed**.

In the following example, these **/etc/gateways** entries provide routing information about the ARPANET and a foreign host. The "arpanet" network is connected to the local gateway, "vaxd03," which does not use **routed**. The remote host, "vax–x25.arpa," can be reached through the local gateway, "vaxd03," but is two hops away.

```
net arpanet gateway vaxdo3 metric 1 passive
host vax–x25.arpa gateway vaxdo3 metric 2 passive
```

3.1.6 The /etc/hosts.equiv Administrative File

This file lists hosts that share user accounts and eliminates the need to supply a log–in name and password when using the following programs and functions: **lpr, lprm, lpq, rcmd, rcp, rlogin, rsh**.

> **NOTE:** All nodes that use **lpr,** including the node that runs the line printer daemon **lpd**, must be configured for TCP/IP communications. You must place the names of all nodes that will print files using **lpr** in the **lpd** node's **/etc/hosts.equiv** file.

The **/etc/hosts.equiv** file contains the name of each equivalent TCP/IP host, one name per line. For example:

```
HostA
HostB
HostC
HostD
HostE
```

The function performed by this file is similar to that performed by the **.rlogin** file on individual hosts, described in *Using TCP/IP Network Applications*.

3.1.7 The /etc/resolv.conf Administrative File

This administrative file, the resolver configuration file, is required only if your network system uses the UNIX name server, **named,** for Internet name–address resolution, and if remote servers answer queries from hosts not running the server locally. Section 3.2, "Methods of Internet Name–Address Resolution," discusses when to use **named** in your network system and Chapter 4 describes using **named** in more detail. The resolver configuration file contains name–value pairs that provide information to resolver routines the first time they are invoked by a process. The format of the file is:

```
domain <site-name>.com
nameserver <primary-name-server-ip-address>
nameserver <secondary-name-server-ip-address>
```

The terms are defined as follows:

`domain` Keyword that indicates that the following domain name is the default domain.

`site-name` Default domain name. This name is appended to host names that do not have a dot in them. If no domain entries are present, the domain returned by the **gethostname** utility is used.

nameserver — Keyword that indicates that the following Internet address is the address of a name server that the resolver should query.

ip-address — Internet address, in dot notation (w.x.y.z) of name server. The name servers are queried in the order listed. At least one name server should be listed. If no entries are present, the default is to use the name server on the local machine.

The following is an example of an **/etc/resolv.conf** file:

```
domain apollo.com
nameserver 192.9.13.48
nameserver 192.9.13.49
```

3.2 Methods of Internet Name–Address Resolution

TCP/IP requires that Internet name–Internet address translation services be provided for TCP/IP hosts. There are three methods of providing this service and each method has advantages and limitations.

- Using the ASCII version of the administrative file **/etc/hosts.**
- Using the hashed version of **/etc/hosts**, created with the **mkhosts** utility.
- Using the UNIX name server process, **named,** and distributed name server databases.

3.2.1 Using Only /etc/hosts

Name/address resolution based solely on **/etc/hosts** allows you to maintain a central database, located on a TCP/IP administrative node (and copied to other administrative nodes as necessary). By linking hosts to this central database on their administrative node, you can provide up-to-date information on your entire network by maintaining only a single set of files. Using only the ASCII version of **/etc/hosts** is the simplest method because it requires no further processing of the file after editing. It is recommended for small networks.

Even if you use one of the other methods of Internet Name–Address resolution, you must keep an ASCII version of **/etc/hosts** available to the TCP/IP nodes on the network. They use it for resolving their IP addresses and as a backup method of name–address resolution.

3.2.2 Using Mkhosts Utility

The hashed host database created from **/etc/hosts** with the **mkhosts** command allows faster access to the name/address information. When you create a hashed host database, which consists of the files **/etc/hosts.pag** and **/etc/hosts.dir**, the library routines that are used to resolve name/address queries always use those files rather than **/etc/hosts**. Therefore, when you make changes to the **/etc/hosts** file (in the ASCII version), you must run **mkhosts** again to recreate the hashed database in order to make the changes available to the software. You must also keep the ASCII version of **/etc/hosts** available to TCP/IP hosts and gateways because they read **/etc/hosts** at start up to find their own Internet Address. The **/etc/hosts** file also can be used as a backup name/address resolution method.

Note: You must run **mkhosts** again every time you change **/etc/hosts**.

3.2.3 Using Named

In large internets, with many administrative nodes and distributed authority over network administration, keeping the administrative files accurate can be difficult to do as well as time consuming. The BSD name server simplifies network administration by employing a system of local host databases, each responsible for name/address resolution within its own area, thus distributing the maintenance task. We recommend using **named** in any large internet where TCP/IP resource management is distributed over several groups.

Even if you choose **named**, however, you will need to maintain an **/etc/hosts** file on each network since that file is accessed by the TCP/IP software when the nodes start up and as a backup for name/address resolution if the naming server is not running or is not accessible. You also may need to maintain a centralized **/etc/hosts** for the whole internet if you have programs or shell scripts that explicitly reference **/etc/hosts** for name/address resolution. We provide two conversion utilities, **hostns** and **nshost**, which allow you to convert between **/etc/hosts** and the **named** database files. (See Chapter 4 for information about these utilities.)

3.3 TCP/IP Daemons

This section contains information about daemons, or server processes, relevant to TCP/IP. Daemon is a UNIX term applied to server processes. Typically, they run as background processes on hosts, transparently providing various services. Daemons

- Manage user access to network resources.
- Answer requests for data.
- Gather statistics about the state of the network.
- Manage communication pathways outside the network.

The Domain implementation of TCP/IP supports a number of daemons, most of which are the standard UNIX programs. Some, however, like **tcpd**, are specifically Domain processes. Other UNIX daemons have had Domain options added to improve functionality in Domain networks. Some of the TCP/IP daemons you invoke at startup in **/etc/rc.local,** while others are invoked by **inetd** when needed. This section describes the UNIX and Domain TCP/IP daemons and how they are invoked. For more information about these server processes, see the appropriate *Managing System Software* document for your environment.

3.3.1 tcpd

The TCP/IP protocol server process, **tcpd**, is a Domain daemon which initializes several internal tables required for operation of the protocols and enables a node's socket-call interface. **tcpd** must be started on every node that uses TCP/IP.

3.3.2 routed

The network routing daemon, **routed**, manages the network routing tables. This is the standard BSD daemon that uses a variant of the Xerox NS Routing Information Protocol and dynamically maintains and updates the network routing tables. In a Domain network system, **routed** normally is run continuously only on gateway nodes. On non-gateway nodes, **routed** is started when the node is started, in order to create the node's internal routing tables, but then is stopped, using a Domain option, after the node's routing tables are created.

3.3.3 inetd

The **inetd** daemon is a super daemon that invokes Internet services such as **ftpd** or **rlogind** as necessary. Since it is a single process, **inetd** can efficiently manage many types of Internet connections. You must run **inetd** on every node that requires any of the following servers: **ftpd, telnetd, rexecd, rlogind, rshd,** or **tftpd.**

When **inetd** is started, it reads its configuration information from the file **/etc/inetd.conf**. This file is located on each TCP/IP node and is a link to **'node_data/etc/inetd.conf**. The file must have one entry for each service that **inetd** invokes, in the following format:

```
service-name socket-type protocol wait/nowait user server-prog prog-args
```

For example, the following lines describe the telnet and ftp services.

```
telnet  stream  tcp  nowait  root  /etc/telnetd  telnetd
ftp     stream  tcp  nowait  root  /etc/ftpd     ftpd
```

The `service-names` listed in **/etc/inetd.conf** must be found in the **/etc/services** file and the `protocols` listed must be found in the **/etc/protocols** file. Templates for these files can be found in the **/etc** directory.

3.3.4 ftpd

The **ftpd** daemon accepts File Transfer Protocol (FTP) connections and services FTP requests. Although an FTP connection can be made only to a host (destination) that runs **ftpd,** the **ftp** server process, the host making the request (source) does not have to run **ftpd. ftpd** is invoked by **inetd.**

3.3.5 telnetd

The **telnetd** daemon accepts Telnet connections. This server process must be run on each host that accepts inbound Telnet sessions. The **telnet** command may be issued on a host that does not run **telnetd.** As with **ftpd, telnetd** is invoked by **inetd.**

3.3.6 rexecd

The **rexecd** daemon services requests from the **rexec** library function. It allows users to execute UNIX commands remotely on the host running the **rexecd** server. **rexecd** must receive a valid user ID and password from **rexec. rexecd** is invoked by **inetd.**

3.3.7 rlogind

The **rlogind** daemon services requests from the **rlogin** program. That is, users can log in remotely on any host running **rlogind. rlogind** requires pseudo–ttys. It does not request a password if the remote host is listed in the listed in **/etc/hosts.equiv. rlogind** is invoked by **inetd.**

3.3.8 rshd

The **rshd** daemon is the remote shell server. It services requests from the **rsh** program and the **rcmd** library function. Users can execute UNIX commands remotely on any host running **rshd.** If the remote host is listed in **/etc/hosts.equiv, rshd** does not request a password. **rshd** is invoked by **inetd.**

3.3.9 tftpd

The **tftpd** daemon supports the DARPA Trivial File Transfer Protocol (TFTP). **tftpd** listens for and accepts TFTP requests. Requests can be made only to hosts that run **tftpd,** but requests can be made by hosts that do not run **tftpd. tftpd** is invoked by **inetd.**

3.3.10 rwhod

The **rwhod** daemon is the Internet system status server. It maintains the database of status information that the **rwho** and **ruptime** programs use. **rwhod** is invoked in the

/etc/rc.local startup file. In a Domain network, one master node runs **rwhod** normally and updates the common **/usr/spool/rwho** directory while the other nodes invoke **rwhod –t** so that they only transmit their status information and do not attempt to update the directory. See the appropriate *Managing System Software* document for your environment for more information about **rwhod**.

3.3.11 Invoking TCP/IP Daemons

The TCP/IP daemons **tcpd**, **routed**, **rwhod**, **named**, and **inetd** usually are invoked by the startup file, **/etc/rc.local**. This file is protected, so only those people with the correct permissions can edit it. In order to allow users to control which servers get started on their Domain nodes, daemons are started only if they are invoked in **/etc/rc.local** *and* a file exists in the node's **/etc/daemons** directory with that daemon's name. If a file with the name of the server process exists within that directory, the process is started. If the file does not exist, the server process is not started. For example, if you wish to run the UNIX name server process, create the file **/etc/daemons/named**. Note that daemons invoked by **inetd** (**ftpd**, **rshd**, **rexecd**, **rlogind**, and **telnetd**) do not need files in the **/etc/daemons** directory.

3.4 Configuring a TCP/IP Network

This section contains several procedures to help you configure a TCP/IP network. Procedure 3–1 includes the following tasks:

- Designating hosts, gateways, and administrative nodes
- Selecting IP internet addresses
- Configuring the administrative nodes
- Editing the administrative files
- Configuring hosts and gateways
- Testing TCP/IP communications on each node

Additional procedures are included to give you more information about specific tasks listed in Procedure 3–1. For example, Procedure 3–2 helps you create your administrative files if you are joining the ARPANET and Procedure 3–3 helps you test TCP/IP communications.

Users who want to install and configure TCP/IP on their nodes may turn directly to Procedure 3–4.

Procedure 3-1. *Configuring a TCP/IP Network*

This procedure describes how to configure your TCP/IP network.

Task 1: **Designate hosts, gateways, and administrative nodes**

If you have not already done so, list the TCP/IP hosts and gateways you plan to configure. Select an administrative node to contain the administrative files, **/etc/hosts, /etc/networks**, **/etc/gateways**, **/etc/hosts.equiv**, and **/etc/resolv.conf**. (This node need not be a TCP/IP host.)

If you have more than one subnet in your network system, select one administrative node for each subnet.

Task 2: **Select Internet Addresses — All TCP/IP Hosts and Gateways**

If you have not already done so, select Internet addresses for all hosts and gateways. (Refer to Chapter 2 for information about selecting Internet addresses.)

Task 3: **Select Method of Internet Name-Address Resolution**

Decide what method on name-address resolution is appropriate for your network. The method chosen will determine which administrative files and daemons are required. See Section 3.2 for information on choosing the appropriate method.

Task 4: **Install the Software — All TCP/IP Hosts and Gateways**

Install the software, as described in *Installing Software with Apollo's Release and Installation Tools*. TCP/IP files and utilities are installed as part of the operating system. You must choose to have all the TCP/IP files loaded during the installation on TCP/IP hosts and gateways.

Task 5: **Configure the Administrative Node(s)**

When the software is installed on all nodes, the TCP/IP administrative files are installed as links to the "tcp_admin" node. (Section 3.1.1 discusses file links.) To configure an administrative node, you must

1. Delete file links for the administrative files, including **/etc/hosts**, **/etc/networks**, and **/etc/hosts.equiv**. Include **/etc/gateways** and **/etc/resolv.conf** only if required by your network system.

If the node also will be a TCP/IP host or gateway, then do the following:

2. Edit the **/etc/rc.local** file, if necessary. (Appendix A describes the **/etc/rc.local** file in detail.)

3. Create a file in the node's **/etc/daemons** directory for every daemon that will run on that node.

Task 6: Create Administrative Files on Administrative Nodes

If your network system will **not** communicate with the ARPANET:

1. On the administrative node, manually create the administrative files required by your network system following the file formats described in Section 3.1, "TCP/IP Configuration Files." The files required depend on the method of name–address resolution you chose in Task 3.

   - If you are using the ASCII version of **/etc/hosts**, create **/etc/hosts**, **/etc/networks**, and **/etc/hosts.equiv.**

   - If you are using the hashed host database, create **/etc/hosts**, **/etc/networks**, and **/etc/hosts.equiv.** Then run the **mkhosts** utility on **/etc/hosts** to create **/etc/hosts.pag** and **/etc/hosts.dir**.

   - If you are using **named** and remote servers, create **/etc/hosts**, **/etc/networks**, **/etc/hosts.equiv** and **/etc/resolv.conf.** (See Chapter 4 for more information about **named** startup and database files.)

2. Create an **/etc/gateways** file for a network only if there are gateways within that network that do not support the Routing Information Protocol.

If your network system **will** communicate with the ARPANET, follow Procedure 3–2 to create your administrative files.

Task 7: Configure TCP/IP Hosts and Gateways

There are four primary tasks involved in configuring TCP/IP hosts and gateways.

1. On each host and gateway, make sure the link named "tcp_admin" in the **/etc** directory resolves to the TCP/IP administrative node for your network. The administrative nodes were configured in Task 4 of this procedure. If the link does not exist, create it. For example:

   $ crl tcp_admin *//admin_node* (in the Aegis environment)

   $ ln –s *//admin_node* **tcp_admin** (in either UNIX environment)

   Note: If your administrative node is running SR10, then the "tcp_admin" link should resolve to *//admin_node*. If your administrative node is running SR9.7, the link should resolve to *//admin_node/* **bsd4.2.**

2. Edit the **/etc/rc.local** startup files on nodes that require it. The **/etc/rc.local** file was designed to require **no** changes if you are configuring a TCP/IP host with one physical interface. (See Appendix A for a description of **/etc/rc.local**.)

   - Configure network interfaces and IP addresses on all gateways.

   - If subnetting is used, all nodes must have subnet masks added to **ifconfig** command line.

3. Create a file in the node's **/etc/daemons** directory for every TCP/IP daemon that should run on that node.

4. Check that all hosts and gateways are included in the network's **/etc/hosts** file.

Task 8: Catalog All TCP/IP Hosts and Gateways

The TCP/IP software can use node names as default IP names, if the nodes are cataloged on the network. Follow the procedures defined in the *Managing System Software* document for your environment.

Task 9: Reboot the Nodes

Shut down and reboot the nodes to enable TCP/IP services.

Task 10: Test TCP/IP Communications on All Hosts and Gateways

For each TCP/IP host and gateway you configure, check that the TCP/IP software is running correctly on that node using the **netstat** and **ping** commands. See Procedure 3–3 if you need more information about using these commands for this purpose.

3.4.1 Creating Administrative Files

The following procedure describes how to create the **/etc/hosts**, **/etc/networks**, and **/etc/gateways** administrative files and how to create the optional hashed database with the **mkhosts** utility.

If your network system will communicate with the ARPANET, the process is more complicated than manually creating files because you must include information about the ARPANET in your administrative files. This information is obtained from the Network Information Center (NIC) of Stanford Research Institute (SRI), International, which maintains a master file containing the names and Internet addresses of all networks and gateways on the ARPANET and several other networks which conform to the DARPA–Internet standard. When your software is installed, you receive a copy of that master file in the file **/etc/hosts**, but you should occasionally update your copy.

Procedure 3–2. *Creating Administrative Files*

This procedure is relevant to network installations that will communicate with the ARPANET. If your network will not communicate with the ARPANET, create **/etc/hosts, /etc/networks, /etc/gateways,** and **/etc/hosts.equiv** manually.

Task 1: Update the /etc/hosts.txt File

Use the **gettable** program to copy the master file from the NIC to the **/etc/hosts.txt** file on a TCP/IP administrative node. (See the appropriate *Managing System Software* manual for a reference description of the **gettable** utility.) Use the following steps to copy the file:

1. Set your working directory to the **/etc** directory on the TCP/IP administrative node. For example,

   $ **wd** *//admin_node/***etc** (in the Aegis environment)
   $ **cd** *//admin_node/***etc** (in either UNIX environment)

2. Run **gettable** with the host name SRI.NIC.ARPA. The output file from **gettable** is **hosts.txt.**

   $ **gettable SRI.NIC.ARPA**

Task 2: Create Your Local Administrative Files

Manually create the following files in the **/etc** directory:

localhosts: Contains the name and Internet address of every host on your network system. Use the format for **/etc/hosts**.

localnetworks: Contains the name and Internet address of every network on your network system. Use the format for **/etc/networks.**

localgateways: Contains static routing information needed by **routed.** Use the format for **/etc/gateways.** Create this file only for local networks that contain passive gateways or gateways that do not use **routed.**

Note: An alternative to creating the **/local...** files is to directly edit the output files of **htable** to add your local network information. If you choose to do this, skip this Task and go to Task 3.

Task 3: Run the htable Program

Still in the **/etc** directory on the administrative node, run **htable** to convert the **hosts.txt** and **local...** files into the format required by the TCP/IP software. If you wish to create a **gateways** file, you must use the **–c** option with **htable**. **Htable** automatically adds the contents of the **local...** files to the top of its output files.

```
$ htable hosts.txt
```

If you have not created the **local...** files, or if **htable** cannot find them, you may see the messages:

```
Warning, no localhosts file.
Warning, no localgateways file.
Warning, no localnetworks file.
```

Task 4: **Check the Files.**

Examine the **hosts, networks,** and **gateways** files that **htable** created to ensure that your network system information was added correctly to the output files.

If you did not perform Task 2, manually add your local network information to the files created by **htable.**

Task 5: **Run the /etc/mkhosts Program.**

If your network system will not use **named,** you may run the program **mkhosts** on the file **/etc/hosts** to create a hashed host database that allows quicker name–address resolution. You must run this program every time you change the ASCII version of the **/etc/hosts** file.

```
$ mkhosts /etc/hosts
```

The output files created by **mkhosts** are **/etc/host.pag** and **/etc/host.dir**. If these hashed files exist, the TCP/IP software will use them for name/address resolution. Therefore, it is very important that you rerun **mkhosts** every time you change **/etc/hosts**.

Task 6: **Create the /etc/hosts.equiv File**

Manually create the **hosts.equiv** file in the **/etc** directory of the administrative node. This involves adding to the file the host names of every TCP/IP host in your network system that will use the following programs and functions: **lpr, lprm, lpq, rcmd, rcp, rlogin, rsh**. See Section 3.1.6, "The /etc/hosts.equiv Administrative File," for the format of this file.

Task 7: **Create the /etc/resolv.conf File**

If your network system will use **named** and remote name servers, manually create the **resolv.conf** file in the **/etc** directory of the administrative node. Create links on the TCP/IP hosts and gateways to this file on the administrative node. Chapter 4 of this manual contains more information about using **named.**

3.4.2 Testing TCP/IP Communications

The following procedure explains how to use the **netstat** and **ping** commands to test TCP/IP communications after you have installed and configured TCP/IP on a host or gateway.

Procedure 3–3. *Testing TCP/IP Communications*

For each TCP/IP host and gateway you configure, check that the TCP/IP software is running correctly on that node using the **netstat** and **ping** commands.

Task 1: Check Required Processes

Check that all the required processes are running on the node. Use the appropriate process status command for the node's system environment to list the processes currently running. For example:

$ **ps –e** (in the SysV environment)

% **ps ax** (in the BSD environment)

$ **pst –un** (in the Aegis environment)

The **tcpd** server process and other daemons invoked by the **/etc/rc.local** file should be listed in the process status command output. If not, check that

- The correct command lines in the **/etc/rc.local** file have been uncommented and/or edited.

- There are files in the **/etc/daemons** directory for all required daemons.

Task 2: List Active Sockets

Use the **netstat** command with the **–a** option to list active sockets on the node, and with the **–i** option to list the node's network interfaces. (See the **netstat** Manual Page for more information.) When you use **netstat –a**, the screen displays output similar to the following:

```
Active connections (including servers)
Proto  Recv-Q  Send-Q  Local Address Foreign Address (state)
tcp      0       0      *.telnet          *.*        LISTEN
tcp      0       0      *.ftp             *.*        LISTEN
tcp      0       0      *.exec            *.*        LISTEN
tcp      0       0      *.login           *.*        LISTEN
udp      0       0      *.route           *.*        LISTEN
udp      0       0      *.who             *.*        LISTEN
```

When you use **netstat –i**, the screen displays output similar to the following:

```
Name  Mtu   Network    Address    Ipkts Ierrs Opkts Oerrs Collis
dr0  1268  ring_15    my_node     316    0    51     0     0
lo0  9216  localnet   localhost    0     0    0      0     0
```

An asterisk following the local loopback name (that is, lo0*) means that the loopback interface is not enabled. Use the **ifconfig** command to enable it.

If no other interface but the local loopback is shown with the **netstat -i** command, the node's network interface has not been enabled. This may mean that the TCP/IP administrative node was not available when the node booted.

Task 3: Check Local Physical Interface

Use the **ping** command to check that the node's local network interface is up and running and the TCP/IP software is operating. Use the "count" option to limit the number of datagrams sent. For example:

$ **ping** *my_node* **count** *2*

The node is operating properly if you see a response to the **ping** command like the following:

```
PING my_node.my.domain: 0 data bytes
8 bytes from 192.9.9.4: icmp_seq=0
8 bytes from 192.9.9.4: icmp_seq=1

----my_node.my.domain PING Statistics----
2 packets transmitted, 2 packets received, 0% packet loss
```

If you see the following response, check that **tcpd** is running as described in Task 1.

```
ping: socket: I/O error
```

If you see the following response, check that the network interface is defined correctly in the **/etc/rc.local** file:

```
PING my_node.my.domain: 0 data bytes
Timed out (1 second) waiting for echo reply
Timed out (10 seconds) waiting for echo reply

----my_node.my.domain PING Statistics----
2 packets transmitted, 0 packets received, 100% packet loss
```

Task 4: Check Remote Physical Interface

Use the **ping** command to check remote hosts in the same way.

Task 5: Initiate Communications on Same Network

Initiate communications with another host on the same network, using **telnet** or **ftp**. If that connection can be made, the host can communicate over the network.

$ **telnet** *other_node*

You may receive the following error messages:

```
Trying...
Destination refused.
```

The "destination refused" message is returned when the remote host is not configured to accept incoming requests for the service you are using. In this case, reconfigure the remote host by editing the remote host's **/etc/inetd.conf** file to enable **telnet**.

```
Network is unreachable.
```

The "network is unreachable" message is returned when the local host's network interfaces are not properly configured, or when there is no information in its routing table. Make sure that you have executed **ifconfig** from the host's **/etc/rc.local** file to configure the network interfaces. (See Appendix A for information about the **/etc/rc.local** file.)

```
I/O error.
```

The "I/O error" message is returned when the local host is not running **tcpd**. Refer to Appendix A for information about starting **tcpd** from the **/etc/rc.local** file. See Task 1 for information about listing processes on the host.

Task 6: Initiate Communications on Remote Network

Initiate communications with a host on another network, using **telnet** or **ftp**. If the connection is made, the host can communicate over the network.

You may receive the following error message:

```
Network is unreachable.
```

Check that a routing table exists, with the **netstat –r** command. Make sure that **routed** is running. Check that the file **/etc/daemons/routed** exists.

Also make sure that the remote host is running and that you have used the **route** command to configure static routes where necessary. (See Appendix A for information about executing **route** and **routed** from the **/etc/rc.local** file. See Chapter 5 for information about troubleshooting TCP/IP.)

3.4.3 Configuring TCP/IP Hosts and Gateways

This section contains information about configuring Domain nodes to be TCP/IP hosts and gateways. The following procedure describes the basic steps involved in configuring a host or gateway. If you are configuring a gateway, you also must edit the **/etc/rc.local** startup file. Appendix A describes the **/etc/rc.local** file in detail.

Procedure 3–4. *Configuring a TCP/IP Host or Gateway*

This procedure describes how to configure individual TCP/IP hosts and gateways

Task 1: Obtain Hostname for Node

The **/etc/rc.local** startup file is set up to use the node name as the TCP/IP hostname. To use this feature, make sure the node (disked or diskless) is cataloged in the root directory and given a name. For diskless nodes, you also must uncatalog the node's //diskless_$<nodeid> name, if it exists, from the root directory. Directions for cataloging nodes can be found in the *Managing System Software* document for your environment.

If you do not wish to use the node name as the TCP/IP hostname, you must edit the **hostname** command line in the **/etc/rc.local** file.

Task 2: Obtain Internet Address(es) for Node

Ask your System Administrator to assign one or more Internet Addresses for the node (one per host, two or more per gateway). If the node is a TCP/IP host, the TCP/IP software will automatically learn its Internet Address at startup by reading the network's **/etc/hosts** file.

If the node is a TCP/IP gateway, you must edit the **/etc/rc.local** file. See Task 3.

Task 3: Edit /etc/rc.local File, if Necessary

If you are configuring a TCP/IP host which has one network interface to either an Apollo Token Ring or an IEEE 802.3 network and which uses its node name as its TCP/IP hostname, you do not need to edit this file. Go on to Task 4.

You will need to edit the node's **/etc/rc.local** if:

- You are configuring a TCP/IP gateway. Edit the **ifconfig** command line to add the gateway's assigned Internet Addresses.
- The network is subnetted. Edit the **ifconfig** command line to add the subnet mask.

- You want Ethernet trailers enabled for that node. Edit the **ifconfig** command line to replace the "–trailers" parameter with "trailers".
- You do not want to use the default options for **rwhod, named,** or **routed.**

Appendix A contains information about the contents of the **/etc/rc.local** file and how to edit it.

Task 4: Check Node's /etc/tcp_admin Link

TCP/IP hosts and gateways must link to the administrative files located on a TCP/IP administrative node. Check that the node's **/etc/tcp_admin** link, created during software installation, resolves to the correct administrative node. Ask your System Administrator for the correct link.

If your administrative node is running SR10, then the **/etc/tcp_admin** link should resolve to *//admin_nodename*. If your administrative node is running SR9.7, the link should resolve to *//admin_nodename/***bsd4.2**.

Task 5: Check /etc/hosts File

The node's hostname and Internet Address(es) must be in the **/etc/hosts** file for that network. If they are not, ask your System Administrator to add them. TCP/IP cannot run on that node until its hostname and Internet Address(es) are listed in the **/etc/hosts** file.

Task 6: Check Node's /etc/daemons Directory

You must have files for **tcpd** and **routed** in this directory since these daemons are required for the proper operation of TCP/IP. If you want to invoke other daemons, such as **inetd, named,** or **rwhod,** add files for them in this directory.

Task 7: Check Node's /etc/inetd.conf File, if Necessary

If you invoke **inetd,** check the node's **/etc/inetd.conf** file to make sure that the required services are not commented out.

Task 8: Reboot the Node

If you have made any changes to the **/etc/rc.local** file, added any files to the **/etc/daemons** directory, or required any changes to be made to the network's **/etc/hosts** file, you must reboot the node.

Chapter 4

Configuring the Name Server

This chapter describes the operation of the BSD name server program, **named**, and the contents and formats of the data files required by the server. Also included are instructions on how to configure your network to use **named** and how to use the **hostns** utility to create name server data files from an existing **/etc/hosts** file. This chapter is based on the *Name Server Operations Guide for BIND* (Berkeley Internet Name Domain) (SMM11).*

> **NOTE:** The 4.3BSD name server program is unrelated to Apollo's Domain network Naming Server, ns_helper. The information in this manual *only* concerns **named**.

4.1 The Name Server Daemon

The **named** program is a UNIX daemon that provides a mechanism for translating host names into addresses usable by the TCP/IP software on your node. The name server daemon is the 4.3BSD implementation of the DARPA Internet Name Resolution Protocol (see RFCs 882 and 883). This protocol was developed to solve the problem of maintaining the Network Information Center's (NIC) global Internet host name–address mapping table. Rather than maintaining a single global address resolution table, the Internet Name Resolution Protocol allows responsibility for authoritative name–address information to be distributed among multiple name servers. Name servers are the source of name–address resolution information about hosts located only within their area of responsibility; queries about other hosts are passed on to other name servers. Name servers now operate throughout the ARPANET, replacing the name–address mapping function of the global table.

Using **named** is recommended for members of the ARPANET. However, we also recommend using **named** in any large internet where network resource management is distributed

* The operations guide exists on line in the file /usr/doc/smm/11.named.

over several groups and maintaining a single source of host name and address information would be difficult.

The **named** program replaces the look–up function of the **/etc/hosts** file. When TCP/IP applications call the library routines **gethostbyname** and **gethostbyaddr**, normally those routines search the **/etc/hosts** file to translate host names to addresses. If you choose to run **named**, however, these routines ask the name server to supply the requested information.

The chief advantage of using **named** over searching **/etc/hosts** is eliminating the need to maintain a single database for a whole network system. For small networks, maintaining an accurate **/etc/hosts** file may be a negligible administrative task. For large TCP/IP internets, however, maintaining this file can be a burden. In addition, **named** solves the problem of managing networks that cross organizational boundaries by distributing authority for accurate naming information across a system of loosely coupled local databases.

4.2 Summary of Name Server Operation

This section explains concepts and terms relevant to the name server daemon and describes briefly how **named** operates.

4.2.1 The Domain Name Space

The DARPA Internet Name Server protocol requires consistent naming conventions that are used to refer to network resources. The chosen solution is a tree–structured naming system called the Domain Name Space. A **domain** within the name space is a group of resources that usually fall within the boundaries of one administrative unit, such as a single department in a company, and that share a common name server to which all unresolved queries are directed. Large domains can contain thousands of hosts and incorporate many smaller domains or **subdomains**.

Domain names reflect the tree structure of the name space. All the parts of the domain are given labels, and the domain name of any part of the domain is the concatenation of all the labels from the highest level, called the root, to the lowest level. The labels are combined from the lowest on the left to the highest on the right and are separated by dots. For example, the name of a host on subdomain C within subdomain A of the Commercial (COM) domain of the ARPANET is "[hostname].C.A.COM." The root label and its associated dot are omitted from printed domain names.

NOTE: Name server domains are unrelated to Apollo Domain networks. A name server domain is an administrative division of a TCP/IP internet running **named**.

Figure 4–1 illustrates a simple domain name tree. In Figure 4–1, the Subdomain A contains two subdomains. The Root Domain contains a name server that receives all unresolved requests from all subdomains.

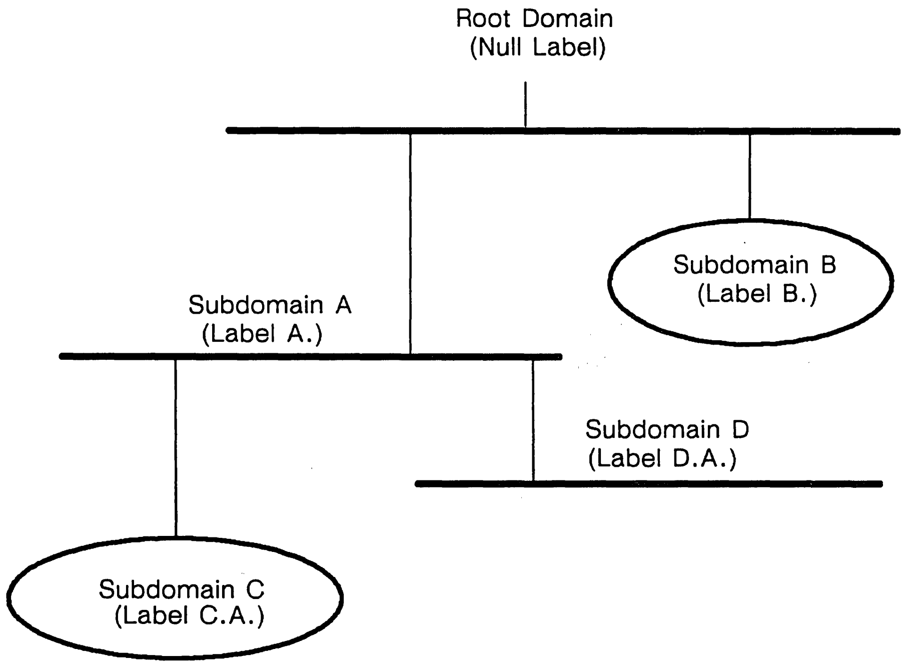


Figure 4–1. Simple Name Server Domain Tree

The ARPANET is organized in several large domains that correspond to classes of networks, such as educational, commercial, and military. The root name server for the ARPANET is the Network Information Center (NIC) at Stanford Research Institute (SRI), International. Figure 4–2 shows part of the ARPANET domain tree. The ARPANET domain names are

- EDU for educational networks
- COM for commercial networks
- MIL for military networks
- GOV for non–military government networks

Individual members of an ARPANET domain may create their own domain trees. The subdomains can correspond to the divisions within the organization. For example, a commercial network may partition its network into personnel, accounting, and R&D.

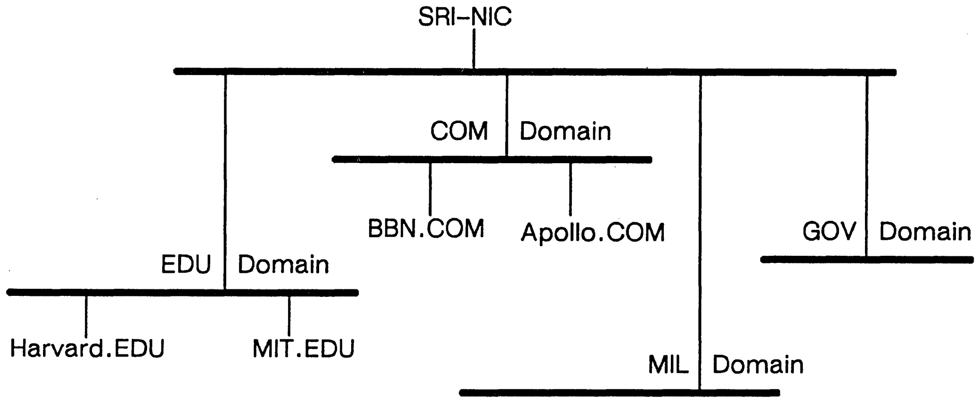


Figure 4–2. The ARPANET Domain Name Tree

4.2.2 Resolving Name–Address Queries

The domain name tree provides the structure for name server operations. Each domain has one or more **master name servers** that coordinate name–address resolution for hosts in a portion of the domain. Master name servers are either **primary** or **secondary** servers. A primary master server loads its database of current name–address information from a file on disk. A secondary master server receives its data from a primary master server at boot time and periodically checks with the primary server to see if it needs to update its data. The portion of the network for which a name server has current and authoritative information is called its **zone of authority**.

Master servers resolve queries for their zones and redirect unresolved queries to another master server or to the domain root name server. The root name server contains a database of all name servers in the domain and their zones of authority. The root name server determines the appropriate local master server and redirects the query to that server. When the local name server resolves the query, it replies to the requesting host.

In addition to master servers, there are **caching only** and **remote** servers. A caching only server is not authoritative for any domain. Rather, it services queries and asks other servers, who have the authority, for the information needed. Remote servers are used on networks where it is undesirable for all hosts to run the name server process. All nodes on the network can run networking programs that use the name server, but name queries are serviced by the remote name server running on one node.

Figure 4–3 shows how **named** handles a request for name–address resolution in a domain name tree.

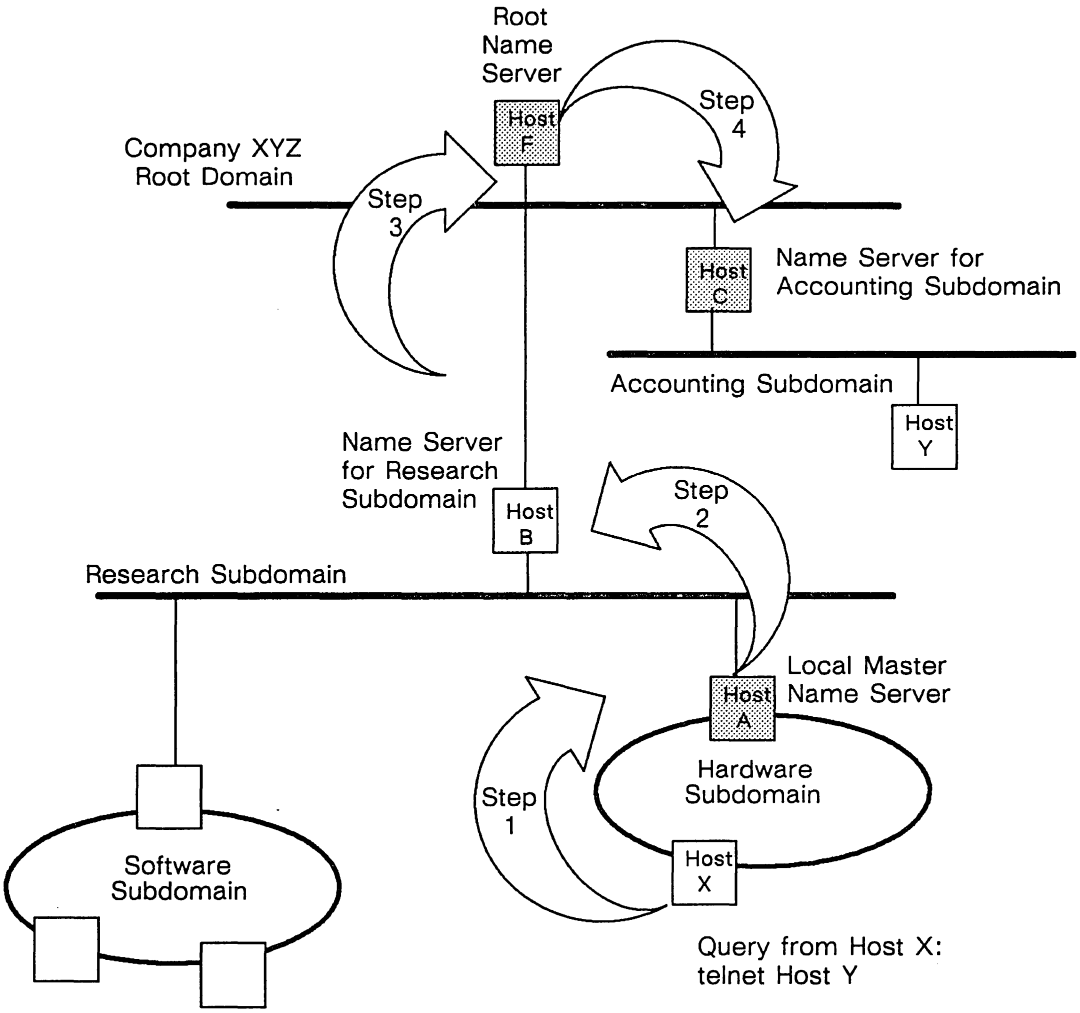


Figure 4–3. Name Server Operation

Figure 4–3 illustrates the following steps:

1. Host X sends a query for name–address information about Host Y to Host A, the local master name server for the hardware subdomain.

2. Host A redirects Host X's query to Host B, A's authoritative name server, because Host Y is not in Host A's zone of authority.

3. Host B can't resolve the query and redirects the query to the Company Domain root name server, Host F, B's authoritative name server.

4. The root name server, HostF, maintains information about all master name servers and their zones. It sends the request directly to Host C (the master name server in Host Y's zone) for resolution. Host C responds directly to Host X.

4.3 The Name Server Database

The name server daemon, **named**, uses two types of files, boot (or startup) files and data files. The two boot files, which must reside locally on every node running the name server, are

- **/etc/named.boot**
- **/etc/named.ca**

There are three data files, two of which must reside locally only on the primary master server for the network. Secondary master name servers obtain copies of these data files from their primary master server. The third data file, **/etc/named.local**, resides on all servers. The data files are:

- **/etc/named.hosts**
- **/etc/named.rev**
- **/etc/named.local**

These files refer to hosts within the domain name space by their domain names and by their Internet addresses. Domain names are a concatenation of all the domain labels from lowest to highest. The format of Internet addresses is explained in Chapter 2 of this manual.

Detailed information about the contents of these files and the Standard Resource Record Format used in the files is provided in Appendix C.

4.3.1 The named.boot File

The **/etc/named.boot** file is the first file **named** reads when it executes. This file specifies the type of server **named** will be (that is, primary or secondary master server, or caching only server), the server's zone of authority, and the locations of the source files for its name–address data. This boot file must be located on all hosts running **named**. Figure 4–4 shows a sample **named.boot** file for a *primary* master server.

```
;  named.boot -- boot file for 4.3BSD name server
;  HostE is primary master server on Software subdomain
;
; Type    Domain                Source File or Host
; ----    ------                ----------------
;
domain    sw.rd.
;
primary   sw.rd.                 /etc/named.hosts
;
cache                            /etc/named.ca
;
primary   4.9.192.in-addr.arpa   /etc/named.rev
;
primary   127.in-addr.arpa       /etc/named.local
```

Figure 4–4. The named.boot File for Primary Master Server

4.3.2 The named.ca File

The **named.ca** boot file is pointed to in the **named.boot** file and contains information that tells the server where to send queries that it cannot resolve itself. The data in this file is added to the server's name–address cache when the server is started. For primary and secondary master servers, this file would contain information about the servers in the next higher zone of authority. For caching only servers, this file would contain information about the primary and secondary servers for the current zone.

Figure 4–5 illustrates a cache file for a primary name server. A cache file for a secondary name server for the same zone of authority would be identical to this file.

```
;
;          named.ca  -- cache file for 4.3BSD name server
;          HostE is primary master server in sw.rd subdomain
;
;          Time to  Addr     Record   Record
;  Name     Live    Class     Type     Data
; ------- -------   ----     ------   ------
;
rd.         9999999   IN       NS      HostB.rd
;
..          9999999   IN       NS      HostF
;
HostB.rd.   9999999   IN        A      192.9.3.1
;
HostF.      9999999   IN        A      192.9.1.1
```

Figure 4–5. The named.ca File for Primary Master Server

4.3.3 The named.hosts File

The **named.hosts** file contains name–address data about the hosts in a particular zone of authority. The location of this file is pointed to in the name server's boot file. For primary name servers, this file must reside locally and its name is given in the boot file. For secondary name servers, the boot file specifies the address of the primary server on which this file resides.

Figure 4–6 illustrates a **named.hosts** file for a primary master name server. This file uses the Standard Resource Record Format described in Appendix C. See Appendix C for descriptions of other types of records that can be included in the **named.hosts** file, such as Host Information, Well Known Services, and several types of Mail records.

```
;
;        named.hosts file for sw.rd. subdomain
;        HostE is primary master server in sw.rd subdomain
;
;     Time to  Addr  Record   Record
;Name  Live   Class   Type     Data
;----  ----   -----  ------   ------
;
@              IN     SOA     HostE.sw.rd.   admin.HostE.sw.rd. (
                     1.1         ;serial
                     3600        ;refresh
                     300         ;retry
                     3600000     ;expire
                     3600 )      ;minimum
;
               IN     NS       HostE.sw.rd.
;
localhost      IN     A        127.0.0.1
HostE          IN     A        192.9.4.3
HostF          IN     A        192.9.4.2
HostD          IN     A        192.9.4.1
               IN     A        192.9.3.2
;
tom            IN     CNAME    HostF
dick           IN     CNAME    HostD
harry          IN     CNAME    HostE
;
admin          ANY    MB       HostE.sw.rd.
```

Figure 4–6. The named.hosts File for Primary Master Server

4.3.4 The named.rev File

This file specifies the "in–addr.arpa" domain, which is a special domain for allowing address to name mapping. Since Internet addresses do not necessarily fall within domain boundaries, this domain was created to allow inverse (i.e., Internet address to domain name) mapping.

The "in–addr.arpa" domain has four labels preceding it. Each label corresponds to an octet of an Internet address, in reverse order. All four octets must be specified, even if an octet is zero. For example, the Internet address of 192.9.0.1 is expressed as 1.0.9.192.in–addr.arpa.

Figure 4–7 illustrates the /etc/**named.rev** file that resides on a primary master server.

```
;
;        named.rev file for sw.rd. subdomain
;        HostE is primary master server in sw.rd subdomain
;
;      Time to   Addr   Record     Record
;Name   Live     Class  Type       Data
;----  -------   -----  ------      ------
@                 IN     SOA    HostE.sw.rd.  admin.HostE.sw.rd. (
                         1.1          ;serial
                         3600         ;refresh
                         300          ;retry
                         3600000      ;expire
                         3600 )       ;minimum
;
                  IN     NS         HostE.sw.rd.
;
$ORIGIN 4.9.192.in–addr.arpa.
;
1                 IN     PTR        HostD.sw.rd.
2                 IN     PTR        HostF.sw.rd.
3                 IN     PTR        HostE.sw.rd.
;
$ORIGIN 3.9.192.in–addr.arpa.
2                 IN     PTR        HostD.sw.rd
```

Figure 4–7. The named.rev File for Primary Master Server

4.3.5 The named.local File

This file specifies the address for the local loopback interface, known as **localhost**, with the standard network address of 127.0.0.1. Figure 4–8 illustrates a sample **named.local** file. This file should be located on all nodes running the name server.

```
;
;        named.local file for sw.rd. subdomain
;        HostE is primary master server in sw.rd subdomain
;
;     Time to    Addr  Record   Record
;Name  Live     Class   Type     Data
;----  ------   -----  ------   ------
;
$ORIGIN 127.in-addr.arpa.
@                IN     SOA    HostE.sw.rd.  admin.HostE.sw.rd. (
                        1.1          ;serial
                        3600         ;refresh
                        300          ;retry
                        3600000      ;expire
                        3600 )       ;minimum
;
                 IN     NS      HostE.sw.rd.
;
1.0.0            IN     PTR     localhost.
```

Figure 4–8. Sample named.local File

4.4 Preparing the Network

To prepare the network to run the UNIX name server, the system administrator must

- Partition the network into domains and zones of authority.
- Designate **primary** and **secondary master name servers** for each zone.
- Decide whether to run **named** locally on every node or remotely on some nodes.
- For every zone of authority, create and install the primary, secondary, and caching only name server files on all nodes running **named**. (You may use the **hostns** utility to create the files).

Use your *internet diagram* to record the zones, domains, and name servers you designate. You will need this information when you create the **named** files.

4.4.1 Partition the Network

As we stated, a **domain**, for the purposes of the name server utility, is a group of resources that usually fall within the boundaries of one administrative unit and that share a common name server to which all unresolved queries are directed. Large domains can contain many subdomains. A **zone of authority** is a portion of a domain for which a particular name server has authoritative information about host names and addresses. Domains and subdomains can contain several zones. For example, you could partition your network into a domain for each floor and a zone for each work group. In an internet, you could partition the network system into a domain for each network, as we illustrated previously in this chapter.

Once you have partitioned your network into domains and zones of authority, you must assign **labels** to all parts of the domain. **Domain names** are then a concatenation of all the labels assigned to the parts of the domain down to the specific object. The sequence of labels in a domain name indicates the path to the root, and proceeds from left to right. The root *may* occupy the rightmost position in the domain name. However, in many cases, the root is assumed and has a null label.

There are a number of official ARPANET domain names. Members of the ARPANET may apply to the SRI–NIC for an official domain name.* If you are not a member of the ARPANET, and don't expect to become a member, you may ignore the official domain names and choose your own names.

4.4.2 Designate Server Types

Master name servers maintain current host name–address data and answer and redirect queries. Master servers are often called **authoritative servers** because they always have access to authoritative data for their zones.

Master servers are either **primary** or **secondary** master servers. Primary servers maintain the name–address data files locally; secondary servers obtain the name–address data from primary servers. In addition, a zone of authority may contain **caching only servers** and **remote servers**. Caching only servers always refer to primary or secondary servers for answers to queries. Remote servers resolve queries from hosts that do not run the name server locally.

We recommend that you configure one primary master server and one or more secondary master servers for each zone in your network. This way, the system administrator only updates one primary server, while secondary servers automatically maintain their databases.

We also recommend that you locate either a primary or secondary master server on each TCP/IP gateway. If you assign your master name servers this way, queries are resolved by the gateway's server and are not routed through the network. If network traffic is a concern at your site, locate master servers on gateways. Remember that there is no difference between the *services* provided by primary and secondary servers. The primary server must

* Sites on the DARPA Internet that need information about setting up naming domains should contact HOSTMASTER@SRI–NIC.ARPA. You may also request to be placed on the on BIND mailing list, which is a mail group for people on the Internet running BIND. The group discusses design decisions, operational problems, and other related topics. The address for requesting information is: *bind–request@ucbarpa.Berkeley.EDU.*

have disk space for the database but the secondary server acquires this database and automatically requests updates.

4.4.3 Decide on Local or Remote Operation

If you do not wish to run **named** on every device, you can set up individual hosts to use a remote server to answer queries. To configure a host to use a remote server, create a link on that host to the file **/etc/resolv.conf** on the host's TCP/IP administrative node. This file designates the remote name servers in the zone of authority that should be sent queries. These remote servers can be primary, secondary, or caching only servers. Figure 4–9 illustrates the format of the **resolv.conf** file.

```
;
;          /etc/resolv.conf file for sw.rd domain
;
domain              sw.rd
nameserver          192.9.4.3
nameserver          192.9.4.1
```

Figure 4–9. Format of the /etc/resolv.conf File

If the **/etc/resolv.conf** file exists on a host, it is read when the node starts up and thereafter, all name queries are sent to the remote servers. Even if **named** is running on the local host, name queries are sent to the remote servers. Therefore, if you want the local server to answer queries, **do not** create this file on a host running the name server.

4.4.4 Create Named Files

There are two ways to create the **named** boot and data files for a zone of authority:

- Manually, using as templates the example files shown in Section 4.3, "The Name Server Database," and in Appendix C
- Automatically, using the **hostns** utility and an existing **/etc/hosts** file

The files created by **hostns** are intended for a zone's primary master server. They require only minor editing to make them complete for the primary server. You also can edit the files to make them appropriate for secondary and caching only servers in the same zone. We recommend using the **hostns** utility to avoid unnecessary typing and to ensure correct formatting of the files.

For network systems that have one **/etc/hosts** file containing the names and Internet addresses of all the hosts in the internet, options available with **hostns** allow the name server files to be created on a net or subnet basis. In addition, there is an **nshost** utility that allows you to reverse the process and create an **/etc/hosts** file from the name server files.

Procedure 4–1 describes the process of creating the name server files for a primary master server and then editing them for secondary and caching only servers. In order to have realistic examples for the procedure, we use an example network system whose network addresses and host names are listed in the **/etc/hosts** file shown in Figure 4–10. The topology of the example network system is shown in Figure 4–11.

```
#
#
#          /etc/hosts file for example network system
#
#
127.0.0.1          localhost
192.9.5.1          HostA
192.9.3.3          HostA
192.9.3.1          HostB
192.9.1.3          HostB
192.9.1.2          HostC
192.9.2.1          HostC
192.9.3.2          HostD
192.9.4.1          HostD  dick
192.9.4.3          HostE  harry
192.9.4.2          HostF  tom
192.9.1.1          HostG
192.9.5.3          HostW
192.9.5.2          HostX
192.9.2.2          HostY
```

Figure 4–10. Example /etc/hosts File

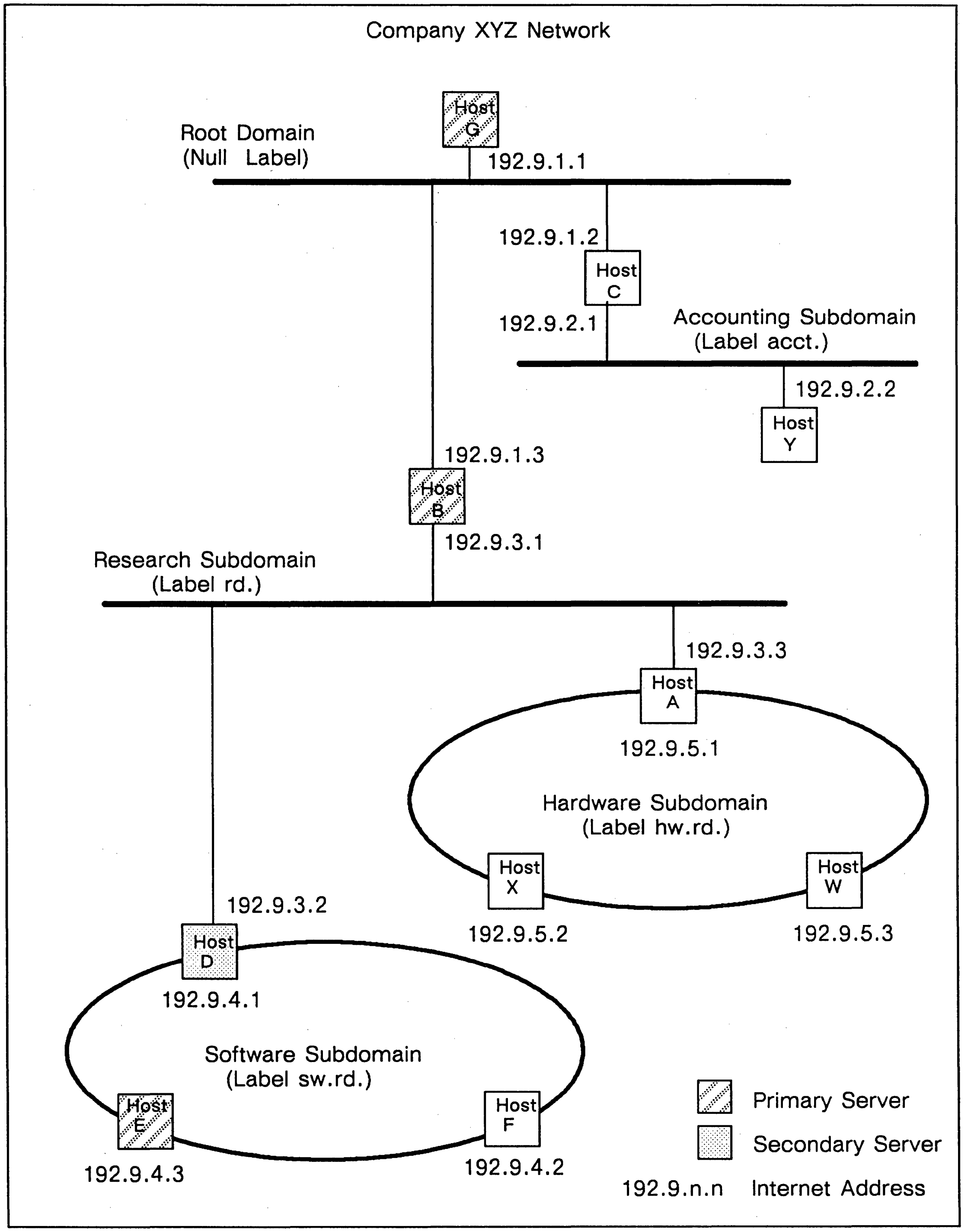


Figure 4–11. Sample Network Using the Name Server

Procedure 4–1. *Creating Named Database Files*

This procedure creates the **named** files for a primary server and then edits those files to make them suitable for a secondary server and caching only servers on a network.

Task 1: Partition the Network System

Partition your network system into domains and subdomains and designate name servers.

When we partition the example network system shown in Figure 4–11 into domains, we choose HostG as the root name server for the network, HostE as the primary master server for the Software subdomain, HostD as the secondary server for the Software subdomain, and HostB as the primary master server for the Research subdomain.

Task 2: Log In on Primary Server

On the primary server for a particular domain, log in as the person responsible for network administration. Set your working directory to the **/etc** directory.

Task 3: Use Hostns Utility

Run **hostns** on the **/etc/hosts** file. Specify the domain name to be included in the **named** files with the **–d** option. Select the network to be included in the files with the **–n** option. For example,

$ **/etc/hostns –d** *sw.rd –n 192.9.4*

When you run **hostns** using the **–d** option, you may see the following warning message, which you can ignore:

```
Warning:  no entry for HostE.sw.rd in /etc/hosts
```

The files created by **hostns** for HostE are shown in Figures 4–12 through 4–16. The **named.ca** file created by **hostns** must be edited to add some information relevant to HostE. This process is described in Task 4.

```
; type    domain                  source file or host
;
domain    sw.rd.
primary   sw.rd.                   /etc/named.hosts
cache     .                        /etc/named.ca
primary   127.in-addr.arpa         /etc/named.local
primary   4.9.192.in-addr.arpa     /etc/named.rev
```

Figure 4–12. The named.boot File Created by Hostns

Task 4: Edit named.ca File

Edit the **named.ca** file to add the name and address of the authoritative server for the primary server (and to remove the default information added by **hostns**, if desired). For example, these lines should be added to the file shown in Figure 4-13:

```
rd.         9999999    IN    NS    HostB.rd.
HostB.rd.   9999999    IN    A     192.9.4.1
```

```
; name         ttl      class    type    record specific information
;
.              9999999   IN       NS     sri-nic.arpa
sri-nic.arpa   9999999   IN       A      10.0.0.51
```

Figure 4-13. The named.ca File Created by Hostns

Task 5: Edit named.hosts File

If necessary, edit the **named.hosts** file to add any other machine or mail information that your network requires. Figure 4-14 shows the **named.hosts** file created by **hostns**.

```
;name   ttl    class    type      record specific information
;
@               IN       SOA      HostE.sw.rd.    admin.HostE.sw.rd. (
                         1.1          ;serial
                         3600         ;refresh
                         300          ;retry
                         3600000      ;expire
                         3600 )       ;minimum
;               IN       NS       HostE.sw.rd.
localhost       IN       A        127.0.0.1
HostD           IN       A        192.9.4.1
dick            IN       CNAME    HostD
HostE           IN       A        192.9.4.3
harry           IN       CNAME    HostE
HostF           IN       A        192.9.4.2
tom             IN       CNAME    HostF
admin           ANY      MB       HostE.sw.rd.
```

Figure 4-14. The named.hosts File Created by Hostns

The **named.rev** and **named.local** files created by **hostns** do not need to be edited. They are shown in Figure 4-15 and 4-16.

```
;name   ttl    class    type      record specific information
;
@               IN       SOA     HostE.sw.rd.    admin.HostE.sw.rd. (
                        1.1         ;serial
                        3600        ;refresh
                        300         ;retry
                        3600000     ;expire
                        3600 )      ;minimum
;               IN       NS       HostE.sw.rd.
1               IN      PTR       HostD.sw.rd.
3               IN      PTR       HostE.sw.rd.
2               IN      PTR       HostF.sw.rd.
```

Figure 4–15. The named.rev File Created by Hostns

```
;name   ttl    class    type      record specific information
;
$ORIGIN  127.in-addr.arpa
@               IN       SOA     HostE.sw.rd.    admin.HostE.sw.rd. (
                        1.1         ;serial
                        3600        ;refresh
                        300         ;retry
                        3600000     ;expire
                        3600 )      ;minimum
;               IN       NS       HostE.sw.rd.
1.0.0           IN        PTR     localhost.
```

Figure 4–16. The named.local File Created by Hostns

Task 6: **Copy Files to Secondary Server**

To create the name server files required for the secondary server, copy the **named.boot, named.ca** and **named.local** files to the secondary server.

Task 7: **Edit named.boot File on Secondary Server**

Edit the boot file as required. Figure 4–17 illustrates the boot file required for HostD, our designated secondary server.

```
; type      domain                source file or host
;
domain      sw.rd.
secondary   sw.rd.                  192.9.4.3
cache                               /etc/named.ca
primary     127.in-addr.arpa        /etc/named.local
secondary  4.9.192.in-addr.arpa  192.9.4.3
```

Figure 4–17. A named.boot File Edited for HostD

Task 8: Copy Files to Caching Only Servers

To create the name server files required for caching only servers, copy the **named.boot, named.ca** and **named.local** files to all other hosts on the network that will run **named,**

Task 9: Edit named.boot and named.ca Files

Edit the boot and cache files as required. Figure 4–18 illustrates the boot file and Figure 4–19 illustrates the cache file required for caching only servers on this network.

```
; type      domain                 source file or host
;
domain      sw.rd.
cache                                /etc/named.ca
primary     127.in-addr.arpa         /etc/named.local
```

Figure 4–18. A named.boot File Edited for Caching Only Servers

```
; name          ttl     class    type    record specific information
;
.               9999999   IN      NS      HostE.sw.rd
.               9999999   IN      NA      HostD.sw.rd
HostE.sw.rd.   9999999   IN      A       192.9.4.3
HostD.sw.rd.   9999999   IN      A       192.9.4.1
```

Figure 4–19. A named.ca File Edited for Caching Only Servers

Chapter 5

Managing and Troubleshooting a TCP/IP Internet

This chapter describes how to manage an Apollo TCP/IP internet after all the nodes are configured. Included in this information are suggestions on how to verify the correct operation of TCP/IP on a node, how to add TCP/IP nodes to your internet, and how to remove them. This chapter also provides a brief troubleshooting procedure and then describes in detail the commands you can use to troubleshoot your internet.

5.1 Verifying Correct Operation

In order to verify that the TCP/IP software is running correctly on a single node, you should check that

- The server processes specified in the **/etc/rc.local** file are running.
- The node's **/etc/rc.local** file starts the server processes and defines its network interfaces correctly.
- The node is listed in the network's **/etc/hosts** file which resides on the TCP/IP administrative node.

5.1.1 Controlling Server Processes

To list server processes currently running on a node, use the appropriate process status command for the node's system environment. For example:

```
$ ps -e  (in the SysV environment)
% ps -ax  (in the BSD environment)
$ pst  -un  (in the Aegis environment)
```

If you use the BSD process status command, you get output that looks like this:

```
  PID STAT    TIME COMMAND
    2 R    2052:54 null
    3 S       2:41 wired-DXM
    4 S       7:00 purifier
    5 S       8:55 unwired-DXM
    6 S       0:01 netreceive
    7 S       1:20 netpaging
    8 S       3:37 netrequest
    1 S      15:47 /sys/dm/dm
    9 S      55:38 /etc/tcpd
   81 R       0:10 /etc/inetd
   86 S       0:00 ps ax
```

Output from the SysV process status command is very similar to the output of the BSD command.

If you use the Aegis process status command, **pst**, with the **-un** option, you get output that looks like this:

```
Node:  nnnn
Time:  Friday, May 27, 1988   2:27:57 pm (UTC)
-------------------------------------------------------------------
Processor|PRIORITY| Program | State | UNIX INFORMATION| Process Name
Time(sec)|ms/cu/mx| Counter |       |PID | PPID | PGID|
-------------------------------------------------------------------
270225.172 -- -- -- -------   -----  ----  ----  ----  <Null Process>
  4661.032 -- -- -- -------   -----  ----  ----  ----  <Aegis Processes>
    28.177 16/16/16 33E79C2   Wait      1     1     1  init
     0.187  3/14/14 <active>  Ready   170   123   123  uid.nnn...(pst)
     0.289  3/14/14 33E7884   Wait     93     1     0  uid.nnn...(inetd)
  5464.316  3/14/14 33E7884   Wait     85     1     0  tcpd
```

On all TCP/IP hosts, you should see at least **tcpd** listed. On TCP/IP gateways, **routed** also should be running. Other TCP/IP server processes, such as **named** and **inetd**, also could be running, depending on the requirements of the node.

Generally, the **/etc/rc.local** file is used to start the various server processes whenever a node reboots. Server processes started in the **/etc/rc.local** file are owned by **root**. The server processes also must have files in the node's **/etc/daemons** directory in order to be started by the startup file. However, in some cases, you may want to stop and start these processes manually. For example, you might want to stop **tcpd** and then restart it in a window so that you can monitor its activity.

To start a TCP/IP server processes, enter the daemon command. For example, to start **tcpd**, you would enter the following:

```
$ /etc/tcpd
```

To stop a daemon, use the appropriate process status command for your environment to get the process name or number to be stopped. Then use the appropriate command to stop the process. For example:

$ **kill** *PID* (in either UNIX environment)

$ **sigp** *process_name* (in the Aegis environment)

> **NOTE:** In order to stop any TCP/IP processes owned by root, you must be logged in as root.

5.1.2 Maintaining Configuration Files

Configuration files relevant to TCP/IP operation include files on individual nodes and on the TCP/IP administrative node. They are shown in Table 5–1.

Some items to keep in mind when configuring an individual TCP/IP host are:

1. All server processes started in the **/etc/rc.local** file must also have a file by the same name in the **/etc/daemons** directory.

2. Only services listed (and uncommented) in the **/etc/services** file can be invoked by **inetd**.

3. Only protocols listed (and uncommented) in the **/etc/protocols** file can be used by services invoked by **inetd**.

4. All of a TCP/IP host's physical interfaces must be specified with **ifconfig** commands. This is usually done in the **/etc/rc.local** file.

5. All of a TCP/IP host's Internet addresses and its hostname (and aliases) must be listed in the network's **/etc/hosts** file.

6. If you have used the **mkhost** utility to create a hashed database from the **/etc/hosts** file to improve access time to the name–address information, you must run the **mkhost** utility every time you change the contents of **/etc/hosts**.

Table 5-1. TCP/IP Configuration Files and Their Locations

File	**Location**	**Description**
/etc/rc.local	On each TCP/IP host. File is link to **'node_data/etc/rc.local.**	Start–up file for node. Usual way to start TCP/IP server processes.
/etc/inetd.conf	On each TCP/IP host. File is link to **'node_data/etc/inetd.conf.**	Configuration file for **inetd.** Lists all services that can be invoked by **inetd.**
/etc/protocols	On each TCP/IP host.	Lists valid protocols used by services invoked by **inetd.**
/etc/services	On each TCP/IP host.	Lists names of services invoked by **inetd.**
/etc/daemons/<server>	On each TCP/IP host. File is link to file of same name in directory **'node_data/etc/daemons**	Enables <server> process. Allow users to control what servers operate on their nodes.
/etc/hosts	On administrative host with links on all other TCP/IP nodes.	Administrative file that relates host names and Internet addresses.
/etc/networks	On administrative host, with links on all other TCP/IP nodes.	Administrative file that relates Internet addresses to network names for all accessible networks.
/etc/gateways	On TCP/IP gateways. Or on administrative host with links on gateways.	Administrative file that contains static routes to be loaded into static routing tables.
/etc/hosts.equiv	On administrative host, with links on all other TCP/IP nodes.	Administrative file that lists equivalent hosts for log–in purposes.
/etc/named.boot **/etc/named.ca** **/etc/named.hosts** **/etc/named.rev** **/etc/named.local**	On each TCP/IP host running **named.**	**Named** boot and data files.
/etc/resolv.conf	On administrative host with links on each TCP/IP host not running **named** locally.	For networks running **named,** is pointer file to remote servers for hosts not running **named** locally.

5.2 Adding, Removing, Renaming TCP/IP Nodes

The following sections describe how to add hosts and gateways to a network, how to remove hosts and gateways from a network, and how to change the name of a host or gateway node. These descriptions assume that you have the proper permissions (root) to edit start–up files and to stop processes.

5.2.1 Adding Hosts and Gateways to the Network

To add a host or gateway to an existing TCP/IP network that does not use **named**, perform the following steps:

1. Edit the node's **/etc/rc.local** file, if the node is a gateway or a non–standard host (standard hosts have one network interface to an Apollo Token Ring network). See Appendix A for more information about **/etc/rc.local**.

2. Create files for **tcpd**, **routed**, and any other daemons started in the **/etc/rc.local** file, in the node's **/etc/daemons** directory.

3. Add the node's name and Internet address(es) to the network's **/etc/hosts** file. (If you previously ran the **mkhost** utility on **/etc/hosts**, you must run it again every time you change **/etc/hosts**.)

4. Reinitialize TCP/IP on that node either by rebooting the node or by rerunning **/etc/rc.local** with the following command:

```
$ /etc/sys_sh  /etc/rc.local
```

If **named** is being used on the internet, refer to Chapter 4 for information about creating the required name server files from an **/etc/hosts** file.

5.2.2 Removing Hosts and Gateways from the Network

You must remove a TCP/IP host or gateway from the appropriate information files when you either physically remove the node from its network, or when you stop using TCP/IP on the node.

If you are not using **named** on your network or internet, delete the node's entry in the TCP/IP administrative node's **/etc/hosts** file. If you are using **named** on the internet, you must remove the node's entry from the **/etc/named.hosts** and **/etc/named.rev** file on each primary master server.

5.2.3 Changing a Host or Gateway Name

If the node uses its node name as its TCP/IP host name (the default state of the **/etc/rc.local** file), and you want to change the *node name*, perform the following steps:

1. Stop **tcpd.**
2. Uncatalog the old name and catalog the new name, as explained in the appropriate version of *Managing System Software.*
3. Edit the network's **/etc/hosts** file to reflect the new host name.
4. Reinitialize TCP/IP on that node either by rebooting the node or by rerunning **/etc/rc.local** with the following command:

```
$ /etc/sys_sh  /etc/rc.local
```

If the node currently uses its node name as its TCP/IP host name and you want to change the *host name* to a name different from the node name, or if the node's TCP/IP host name is specified in the **/etc/rc.local** file and you want to change it, perform the following steps:

1. Stop **tcpd** on that node.
2. Edit the **/etc/rc.local** file on that node to add the new hostname with the **hostname** command line.
3. Edit the network's **/etc/hosts** file to reflect the new host name.
4. Reinitialize TCP/IP on that node either by rebooting the node or by rerunning **/etc/rc.local** with the following command:

```
$ /etc/sys_sh  /etc/rc.local
```

5.2.4 Changing Host and Gateway Internet Addresses

Internet addresses are specified in the **/etc/rc.local** file with **ifconfig** command lines. Standard TCP/IP hosts with one network interface on an Apollo Token Ring or Ethernet can use the default version of that file, which causes the TCP/IP software to look up its IP address in the network's **/etc/hosts** file. To change the IP address of a standard TCP/IP host that uses the default version of **/etc/rc.local,** perform the following steps:

1. Stop **tcpd** on that node.
2. Edit the network's **/etc/hosts** file to reflect the new IP address for that host.
3. Reinitialize TCP/IP on that node either by rebooting the node or by rerunning **/etc/rc.local** with the following command:

```
$ /etc/sys_sh  /etc/rc.local
```

To change the addresses of TCP/IP gateways and non-standard TCP/IP hosts:

1. Stop **tcpd** on that node.

2. Edit the **/etc/rc.local** file on that node to add the new IP address with the appropriate type of **ifconfig** command line. Gateways have more than one IP address and require one **ifconfig** command line for each address.

3. Edit the network's **/etc/hosts** file to reflect the new IP address for that host.

4. Reinitialize TCP/IP on that node either by rebooting the node or by rerunning **/etc/rc.local** with the following command:

   ```
   $ /etc/sys_sh  /etc/rc.local
   ```

5.3 Internal TCP/IP Tables

After you've configured your network and TCP/IP is running, you have to make sure TCP/IP uses the most current host name and addressing information. In addition, TCP/IP software continuously updates its *internal* routing, address mapping, and interface files. The **tcpd** server process updates routing, address mapping and network interface information for each host. We recommend that you stop **tcpd** before changing the tables so that when you restart the server, **tcpd** automatically runs the utilities to update the tables.

5.3.1 The Internal Routing Table

Each TCP/IP node has an internal routing table which provides a list of accessible destination addresses and which gateways to use from the local network to reach each destination network. The table also indicates whether the gateway is an active or passive gateway. An **active gateway** exchanges routing information with other active gateways through a routing protocol such as the Routing Information Protocol (RIP). That is, an active gateway has a **routed** process running. A **passive gateway** has static routing tables and is not expected to exchange routing information. Domain TCP/IP gateways are active gateways.

The **routed** server process is used on all Apollo TCP/IP hosts and gateways to create the internal routing tables. On hosts, **routed** exits after creating the tables; on gateways, **routed** continues to run in order to update routing information to foreign networks. The **route** command can be used to manually manipulate a node's routing tables, if necessary. See Section 5.4.5, "Manual Manipulation of the Routing Table," for more information about using the **route** command.

The internal routing table on nodes lists *all* accessible destination networks. It indicates the *next* gateway in the route to each destination. You can display the routing table for an individual node using the **netstat –r** command. See Section 5.4.4.6, "Using the netstat –r Option," for an example of a routing table display.

TCP/IP continuously updates a node's routing table with information received over the networks to which the node is connected. The routing server, **routed**, updates the table with information, and, in turn, broadcasts the updated tables over the networks. In addition, the servers purge old information if it is not marked as static. This technique ensures that all nodes have the most current routing tables. We recommend that **routed** run continuously only on gateways. On hosts, start **routed** with the **–h** option which causes the routing server to create the routing tables at start–up but then to exit after they are initialized.

Routed uses the Routing Information Protocol (RIP). RIP enables gateways to exchange Internet routing information. It defines the information that the gateways broadcast, when to broadcast, and the packet format. The protocol also defines the procedures and timeouts used to update the routing tables and delete out–of–date entries. While TCP/IP hosts can listen to RIP messages, they are not required to run **routed** continuously.

After **routed** creates a host's routing table and then exits, the host's table is updated only when the host does not have a gateway entry that it should have. That is, a gateway keeps track of routing, and if it must redirect a packet to another gateway across the *local* network, the gateway enters that route and gateway in the host's routing table.

Because some gateways on a TCP/IP network may not support the RIP protocol, we provide a method of putting information permanently in a gateway's routing table. You can edit the **/etc/gateways** file to include the information when you configure TCP/IP, or you can use the **route** command. We describe the **/etc/gateways** format in Chapter 3, "Configuring a TCP/IP Network."

5.3.2 Address Mapping Files

TCP/IP uses an internal address mapping table to convert between Internet addresses and local network addresses. This table is maintained dynamically on all nodes, and is updated by using the Address Resolution Protocol (ARP). You can display the address mapping table for an individual node using the **netstat –h** command. See Section 5.4.4.3, "Using the netstat –h Option," for an example of an address mapping table display.

TCP/IP uses ARP to update and maintain the internal IP–to–local address mapping table that relates Internet and local network addresses. Whenever a host cannot find the local address that corresponds to the Internet address of a host or gateway that is on its local network, it uses ARP to get the address and add it to the table. Similarly, the host uses ARP to update its address mapping tables with local addresses on both networks that the gateway connects.

Use the **arp** program to add hosts to the IP–to–local address translation table and to display the current contents of that table. The IP–to–local address translation table maps addresses that are on the host or gateway's local networks. Therefore, hosts use the table to map between Internet addresses and local Domain addresses. Gateways map addresses between Internet addresses and local addresses for both networks that they connect.

For more information on the ARP protocol, see the Network Information Center (NIC) publication RFC 826 *An ETHERNET Address Resolution Protocol.*

5.3.3 The Physical Interface Table

In addition to the routing and address mapping tables, the TCP/IP software maintains a physical level interface table that is used in routing each packet. This table associates the host or gateway's networks with the physical level interface for each network. Essentially, it tells TCP/IP which physical network to use for a given network number. If a physical interface is not working, it is marked as being in error in the table, so that TCP/IP can either try another interface or not send the message. To initialize the physical interface table, the TCP/IP software uses the **ifconfig** program, usually in the **/etc/rc.local** file.

You can display the physical level interface table for an individual node using the **netstat –i** command. See Section 5.4.4.4, "Using the netstat –i, –I, and –t Options," for an example of an interface table display.

5.4 Troubleshooting

Your major concern in troubleshooting problems with a TCP/IP system is to determine which node is causing the problem.

In general, when determining the cause of your problem, you will rely on the error messages that you receive while running the TCP/IP software. Appendix B lists error messages that TCP/IP generates and indicates possible causes.

We also provide you with other tools to aid you in troubleshooting. These tools include the **tcpstat** (for the Aegis environment) and **netstat** (for UNIX environments) commands, **dtcb, mbd,** and **ping.**

The **tcpstat, netstat,** and **ping** commands are used to locate problems at the network level. Both **tcpstat** and **netstat** report network status information. **Ping** sends ICMP echo request packets to network hosts and are used to verify that devices are up and running. The **dtcb** and **mbd** commands are used at the host level to isolate particular software problems. The **dtcb** program dumps the contents of TCP control blocks while **mbd** dumps usage information on TCP memory buffer pools.

If you cannot fix the problem, and you determine that the cause of the problem is in the TCP/IP software or in the network controller hardware, you should call your service representative and report the problem.

5.4.1 Checking the TCP/IP Software

When you are troubleshooting, we suggest that you begin by examining the TCP/IP software on host and the gateway nodes. Procedure 5–1 describes the steps you should follow.

Procedure 5–1. *Checking the TCP/IP Software*

This procedure describes how to examine the TCP/IP software on hosts and gateways.

Task 1: **Check Status of tcpd**

Use the appropriate process status command for the node's environment to check that **tcpd** is running. Use of the process status command is described in Section 5.1, "Verifying Correct Operation."

Task 2: **Check Administrative Node**

Make sure the TCP/IP administrative node is running. To find out which administrative node the host or gateway is linked to, list the node's **/etc** directory. For example:

```
$ ls -S /etc   (in the SysV environment)
```

```
% ls -l /etc (in the BSD environment)
```

```
$ ld /etc -ll -lt (in the Aegis environment)
```

The administrative node is the node to which the **tcp_admin** symbolic link resolves. If you use either of the UNIX list directory commands, links are indicated in the output in the following way:

```
hosts -> //tcp_admin/etc/hosts
tcp_admin -> //admin_host
```

If you use the Aegis list directory command, links are indicated in the output in the following way:

```
hosts        "//tcp_admin/etc/hosts"
tcp_admin    "//admin_host"
```

Task 3: **Check Operation of Local Host**

Use the internal software loopback to check that local TCP/IP software is running correctly. See Section 5.4.3,"Using the Software Loopback," for information about using the software loopback.

Task 4: Monitor TCP/IP Activity

To monitor TCP/IP activity, perform the following steps:

1. Use the **tcpstat** command in an Aegis environment or the **netstat** command in either UNIX environment. See Section 5.4.4 for more information on using these commands.
2. Run **tcpd** in a window with the debug option. See Section 5.4.2 for more information about using **tcpd**.

Task 5: Connect to Host on Same Network

Try to establish a connection to another host node on the *same* physical network. Use the **crp** command to make sure the host is running the appropriate TCP/IP servers or daemons. Once you establish a connection, use **tcpstat** or **netstat** to monitor TCP/IP activity.

Task 6: Connect to Host on Remote Network

Try to establish a connection to a host across a gateway to another physical network. Use **tcpstat –r** or **netstat –r** to get the name of a gateway. Once you establish a connection, use **tcpstat** or **netstat** to monitor TCP/IP activity.

Task 7: Check Operation of Remote Host

If you find errors that indicate the remote host is not responding, check that the host is functioning and that all required servers are running. Table 5–2 gives examples of such errors, and possible reasons; Appendix B lists additional error messages.

Table 5–2. Common Error Messages from Remote Hosts

Message	Description
Destination not responding	Remote host is probably down or not running TCP/IP, or its routing table does not include the necessary gateway.
Destination unreachable	Gateway is probably down or not running TCP/IP. Local gateway table is not installed. To check, use **tcpstat –r** or **netstat –r**.
Destination refused	Remote host does not provide the services you are attempting to use, such as **ftp** or **telnet.**

5.4.2 Using the tcpd Debug Option

The **tcpd** server process can display various types of debugging information such as flow of control and indications of all messages received, including message size and the TCP header. To display this information, you must start the TCP server process using the **–d** option and supply a bitmask value that defines the type of information desired, as shown in the following example:

1. Stop **tcpd** on that node. You must be logged in as **root**.

2. Start **tcpd** with the debug option.

   $ /etc/tcpd –d[value]

The bitmask values are hexadecimal values that correspond to a 16–bit mask. Table 5–3 lists the debug information you get with each bit.

Table 5–3. Getting Additional Debug Information

Bit Value	Debug Information
0001 (default)	General Information
0002	IP Level Information
0004	ARP Information
0008	TCP Information
0010	Data in TCP packets
0020	UDP Information
0200	Broadcasts
1000	TCP Finite State Machine Information
2000	Device Level Information
4000	Additional Detail at any Level

To specify additional information, specify the bit values corresponding to the information you need. For example, to specify TCP (0008) and IP (0002) information, you add the bits 0002 and 0008 to get 000a. Specify the following hexadecimal value on the command line as follows:

```
$ /etc/tcpd  -d000a
```

To specify TCP, IP and device level (2000) information, add the bits 0002, 0008, and 2000 to get 200a. So, you specify the following hexadecimal value on the command line:

```
$ /etc/tcpd  -d200a
```

At times, you might want to get all the available debug information *except* for one certain type — such as broadcast information. (You might want to suppress broadcast information when **routed** is running on the local network because it generates many broadcasts.) To get all the available debug information except broadcast information, supply the following hexadecimal value on the command line:

```
$ /etc/tcpd  -df0ff
```

5.4.3 Using the Software Loopback

The TCP/IP software provides a software loopback interface, which is accessed by sending a message to Internet address 127.0.0.1.

By convention, address 127.0.0.1 is assigned the host name **localhost.** If you include this host in the network's **/etc/hosts** file when you configure TCP/IP and if you edit each node's **/etc/rc.local** file to uncomment the **ifconfig** line that defines the **localhost** interface, you can then use the name **localhost**, rather than the address, to access the software loopback. For example, you can issue a **telnet connect** command to **localhost**.

Messages that you send to the software loopback interface are redirected back to the host by the TCP software. The messages never go below the IP (network) protocol layer within the host. Therefore, the software loopback limits and isolates the processes that handle the message. (This does not necessarily mean any increase in speed; in fact, the opposite may be true.)

Sending to the software loopback is equivalent to sending to your own address. As with any other destination, you must have a process to receive the connection. For example, you can make a Telnet connection to **localhost** only if you run **telnetd** on your host.

5.4.4 Checking Network TCP/IP Statistics

Use the **tcpstat** or **netstat** programs to troubleshoot and correct problems with your TCP/IP system. Both programs report network status in the same ways: you may use the command **tcpstat** in an Aegis environment and **netstat** in a UNIX environment. Both commands use the same options and produce the same type of output. For simplicity, the following discussion uses only **netstat** in examples.

> **NOTE:** At SR10, **tcpstat** has been changed to operate identically with **netstat**. Therefore, if you have used **tcpstat** in a pre-SR10 Aegis environment, you will notice a change to the operation of **tcpstat**.

The **netstat** and **tcpstat** commands report network status. This section shows the syntax of the commands and their options, and gives information about typical uses of these commands during monitoring or troubleshooting of the TCP/IP configuration. The syntax of the **netstat** and **tcpstat** commands is:

$ netstat [*options*] [*interval*]

If you execute the command without options, the default display shows a list of active sockets for each protocol. For example:

$ netstat

```
Proto  Recv-Q  Send-Q  Local Address    Foreign Address    (state)
tcp         0       0  hosta.2483       max.ftp            ESTABLISHED
tcp         0       0  hosta.2165       hostb.telnet       ESTABLISHED
```

If you execute the command with an interval argument but without options, the display shows a column of statistics related to the host's primary network interface and a column summarizing information for all other interfaces. For example:

$ netstat 30

```
     input   (dr0)    output              input    (Total)    output
packets errs  packets  errs colls  packets   errs  packets errs colls
5546      0    693      1    0      5546       0      693   1    0
6         0    0        0    0      6          0      0     0    0
```

Table 5–4 lists and summarizes the options available with the **netstat** and **tcpstat** commands which cause additional information to be displayed.

The subsections following Table 5–4 show the use of the **netstat** options and describe the values that they report. The manual page for **netstat** is available on–line.

Table 5-4. The netstat Command Options

Option	Description
-A	In addition to the default display, show the address of any control blocks associated with sockets. Used for debugging.
-a	In addition to the default display, show the state of all sockets. Normally, sockets used by server processes are not shown.
-g	In addition to the default display, show the first gateway used in the connection.
-h	Show the state of the internal host table.
-i	Show the state of interfaces which have been configured.
-I (interf)	Show state of interface specified. When used with an interval argument, displays running count of statistics related to the interface.
-t	When used with -i, causes Timer column to be added to display.
-m	Show statistics recorded by memory management routines. (The network manages a private pool of memory buffers.)
-n	Show network addresses as numbers. Normally, the command interprets addresses and attempts to display them symbolically. This option may be used with any of the display formats.
-s	Show per protocol and routing statistics.
-r	Show routing table.
-T	Show all statistics.
-f (addr_f)	Limit statistics or address control block reports to those of the specified address family. The following address families are recognized: inet for AF_INET ns for AF_NS unix for AF_UNIX

5.4.4.1 Using the netstat -A Option

The -A option adds the address of protocol control blocks to the default display, as shown in the following example. This option is used for debugging purposes only.

$ netstat -A

```
PCB      Proto Recv-Q Send-Q  Local Address  Foreign Address  (state)
34a36b8  tcp        0      0  hosta.2483     max.ftp         ESTABLISHED
34a357c  tcp        0      0  hosta.2165     hostb.telnet    ESTABLISHED
```

5.4.4.2 Using the netstat –a and –g Options

The **–a** option adds to the default display the state of all sockets, including the sockets used by server processes. The **–g** option adds to the default display the first gateway used in any connections. The following example shows the output displayed using both options.

$ netstat –ag

```
Proto Recv-Q Send-Q  Local Address Foreign Address  Via Gateway  (state)
tcp        0      0  hosta.2483    max.ftp         gate01      ESTABLISHED
tcp        0      0  hosta.2165    hostb.telnet    public      ESTABLISHED
tpc        0      0  *.ftp          *.*              *         LISTEN
tpc        0      0  *.login        *.*              *         LISTEN
tpc        0      0  *.telnet       *.*              *         LISTEN
udp        0      0  *.*            *.*              *
```

5.4.4.3 Using the netstat –h Option

The **–h** option shows the state of the internal host address mapping table. This address mapping table associates Internet addresses with local addresses. Each association is an entry in the table. Before TCP/IP can make a connection, the local host's address mapping tables must contain an entry for the destination host. During **tcpd** operation, the Address Resolution Protocol (ARP) fills the mapping tables with correspondences as they are required.

If you use **netstat –h** after attempting to establish a connection, and no remote host name (or Internet address) appears, the correspondence was not in the address mapping table and the ARP has not been able to obtain the correspondence.

Inability to map will prevent any further progress towards a connection, so you might want to investigate why mapping has not taken place. Perhaps there are network problems on the network, since ARP must broadcast a packet over the network and receive a reply in order to get the correspondence. Also, the remote host must be running **tcpd** or, for non–Domain hosts, an equivalent TCP listening process.

If **netstat –h** shows that a mapping exists for the proper remote host, the problem is at a less fundamental level. Use other options to the **netstat** command to investigate further. **Netstat –h** can also show if one name is used for two different hosts. If the same name appears twice with different addresses, the name has been incorrectly assigned to two hosts.

The example that follows shows the output displayed when you enter **netstat –h**. The meanings of the fields of this display are shown in Table 5–5.

$ netstat –h

```
Host     Stat       Ref     Pct       Mapaddr
gate01   UG         1       0         0:0:be:99:0:0
public   U          1       0         0:0:ce:45:0:0
hosta    U          0       0         0:0:d6:9b:0:0
```

Table 5–5. Fields of netstat –h Display (Host Address Mapping Table)

Field	Description
Host	The name of the host for which mapping exists.
Stat	The type of entry in the host table, where: G Indicates a gateway entry. U Indicates in use. T Indicates a temporary entry.
Ref	The reference count; that is, the number of connections using one mapping entry. ARP creates mapping entries when it maps IP to local addresses.
Pct	The gateway probe count for gateway entries in the host table. **Tcpd** sends out periodic packets to the gateway to ensure that the gateway will respond. Non–gateway entries do not receive them, so they always show a zero. If a gateway does not respond to 4 consecutive probes, **tcpd** will look for another gateway to use.
Mapaddr	The local network address (physical address) of the remote host.

5.4.4.4 Using the netstat –i, –I, and –t Options

When you use **netstat –i**, the output shows information about the physical interfaces between the host and the network or networks. Most hosts display two reports — one for the network and one for the software loopback interface. Gateways show a line for each network to which they are attached. When you use the **–I** option and specify an interface, the same information is displayed, but only for that interface. When you use the **t** option in combination with **i**, a timer column is added to the display.

The following example shows a **netstat –it** display on a gateway. Table 5–6 explains the meanings of the fields of this display.

$ netstat –it

```
Name  Mtu   Network   Address    Ipkts  Ierrs Opkets  Oerrs Collis Timer
lo0   9216  localnet  localhost  0      0     0       0     0      0
dr0   1268  ring_15   gate01     21457  0     2176    4     0      0
eth0  1500  vax_ether gate01     36331  0     42621   1     0      0
```

Table 5–6. Fields of netstat –it Display (Interface Statistics)

Field	Description
Name	The name of the interface whose statistics are being displayed on line.
Mtu	Maximum transmission unit for interface.
Network	Name of network that interface is on.
Address	Name or address of host.
Ipkts	The number of input packets received by the interface.
Ierrs	The number of input errors for the interface.
Opkts	The number of output packets sent by the interface.
Collis	For Ethernet interfaces, the number of collisions. This usually is not an error condition; rather, it indicates the relative amount of activity on the Ethernet.
Timer	Indicates whether interface has timed out.

NOTE: If there is a problem with a connection and the **netstat –i** display shows large numbers of packet transmissions or errors (in the Ipkts, Opkts, Oerrs or Ierrs fields), the problem may actually be in the physical network.

5.4.4.5 Using the netstat –m and –s Options

The **–m** option shows information about buffer pools used for TCP/IP and the **–s** option reports error statistics on a protocol basis and routing statistics. An example of the output displayed when you use both options with the **netstat** command follows. The statistics requested by the **–m** option are listed first, followed by the statistics requested by the **–s** option.

$ netstat –ms

```
5/336 mbufs in use:
         0/112 (  80-byte) mbufs used/allocated
         5/112 (1560-byte) mbufs used/allocated
         0/112 (9216-byte) mbufs used/allocated
ip:
         0 bad header checksums
         0 with size smaller than minimum
         0 with data size < data length
         0 with header length < data size
         0 with data length < header length
```

```
icmp:
         0 calls to icmp_error
         0 errors not generated 'cuz old message too short
         0 errors not generated 'cuz old message was icmp
         0 messages with bad code fields
         0 messages < minimum length
         0 bad checksums
         0 messages with bad length
         0 message responses generated
tcp:
         0 bad header checksums
         0 bad header offset fields
         0 incomplete headers
udp:
         0 bad header checksums
         0 incomplete headers
         0 bad data length fields
routing:
         0 bad routing redirects
         4 dynamically created routes
         0 new gateways due to redirects
         1 destinations found unreachable
         0 uses of a wildcard route
```

5.4.4.6 Using the netstat -r Option

The **-r** option causes the **netstat** command to display the routing table for that node, as shown in the example below. The routing table display indicates the available routes and their status. The meanings of the fields in the routing table display are given in Table 5-7. See Section 5.4.5 for information about adding routes manually with the **route** command.

$ netstat -r

```
Routing Tables
Destination    Gateway      Flags    Hops  Refcnt  Use      Interface
vax_ether_15   gate01       UG       1      1       13      dr0
ring_270        absolute    UG       1      0      0        dr0
ring_228        absolute    UG       4      0      0        dr0
ring_16         ring_master UG       1      0      0        dr0
ether_12        efs         UG       1      0      0        dr0
ue_ether        main_street UG       2      0      0        eth0
```

Table 5-7. Fields of netstat -r Display (Routing Table)

Field	**Description**
Destination	Destination host or network of route.
Gateway	The gateway to use in forwarding packets to that destination.
Flags	Show status of route. U Indicates route is up. G Indicates route is to a gateway. D Indicates route was created dynamically by a redirect. P Indicates priority route. S Indicates static route added explicitly with **route**. X Indicates route marked for deletion.
Hops	The number of hops to reach that destination.
Refcnt	The current number of active uses of the route.
Use	Provides a count of the number of packets sent using that route.
Interface	Indicates the network interface used for the route.

5.4.4.7 Using netstat -n

Netstat -n modifies the information reported by other **netstat** options. With this option, hosts and gateways are identified by their Internet addresses instead of their ASCII Internet names. If you only specify the **-n** option you get the default report.

5.4.5 Manual Manipulation of the Routing Table

The **route** program can add regular, default, and priority routes to a node's routing table, delete entries, and forcibly clear the table. It normally is not needed since **routed** normally performs this task.

Priority and default routes can be added only with the **route** command. The TCP/IP server process will use priority routes before default routes or routes established by **routed**. **Tcpd** will use the default route when other routes occurring earlier in the routing table have failed, or when there are no other possible routes. The following examples show the commands to add and delete routes on the routing table of an individual node.

1. To add a route to network *ring_16* by way of the gateway with the address *n.n.n.n*, which requires *1 hop*:

```
$ /etc/route add net ring_16 n.n.n.n 1
```

2. To add a **priority** route to network *ring_16* by way of a gateway with the address *n.n.n.n,* which requires *1 hop*:

   ```
   $ /etc/route addp net ring_16 n.n.n.n 1
   ```

3. To add a **default** route:

   ```
   $ /etc/route add default n.n.n.n
   ```

4. To delete a route to network *ring_16* by way of gateway *n.n.n.n*:

   ```
   $ /etc/route delete net ring_16 n.n.n.n
   ```

See the manual page for **route** for more information about how to use this command to add or delete routes.

The ability to clear the routing table helps in troubleshooting because it removes any incorrect Internet address. That is, it clears any local address associations that may prevent messages from being delivered correctly. To use the **route** utility to clear the address mapping table, enter the following command:

```
$ /etc/route -f
```

5.4.6 Using the ping Utility

The **ping** utility is useful in tracking a single–point hardware or software failure in an internet. **Ping** uses the ECHO_REQUEST datagram to elicit an ECHO_RESPONSE from other hosts and gateways. Because of the load it could impose on a network, we recommend that you do not use **ping** during normal operations or from automated scripts.

When using **ping** for fault isolation, run it first on the local host to verify that the local network interface is up and running. Then, "ping" hosts and gateways further and further away. **Ping** sends one datagram per second and prints one line of output for every response returned. If an optional **count** is given, only that number of requests is sent. For example, if the following command is given, the following output is displayed:

```
$ /etc/ping hosta count 5
```

```
PING hosta: 0 data bytes
8 bytes from 192.9.11.24: icmp_seq=0.
8 bytes from 192.9.11.24: icmp_seq=1.
8 bytes from 192.9.11.24: icmp_seq=2.
8 bytes from 192.9.11.24: icmp_seq=3.
8 bytes from 192.9.11.24: icmp_seq=4.

----hosta PING Statistics----
5 packets transmitted, 5 packets received, 0% packet loss
```

5.4.7 Using the Software Debugging Utilities

Use the **dtcb** and **mbd** utilities to track problems in the operation of the TCP/IP software. The **dtcb** utility dumps the contents of the TCP control blocks (tcb) associated with a particular TCP connection. You can obtain the address of the tcb to be dumped with the **netstat –A** command. Two control blocks are dumped by **netstat –A**: the user control block, which contains the send and receive queues and user–related flags; and the tcp control block, which contains the connection sequence numbers, state, flags, and out–of–sequence queues. See the manual page for this command for an example of the output.

The **mbd** dumps usage information about the TCP memory buffer pools. You can obtain the usage statistics on TCP memory buffers with the **netstat –m** command. Use the **mbd** utility to analyze cases where buffers are being lost. The utility scans the entire buffer pool, finding all the chains of in–use buffers and then prints each chain of buffers. This information may help you to discover why buffers are being lost. See the manual page for the **mbd** command for an example of the output.

Appendix A

The /etc/rc.local Startup File

The **/etc/rc.local** file is a shell script located on every Domain host and is a link to the file **'node_data/etc/rc.local.** This script executes at system boot when it is called by **/etc/rc**. The file contains commands that configure a node for TCP/IP. The **/etc/rc.local** file can be edited by a system administrator who has the required permission to edit it. That is, someone who is logged in as **root**.

The **/etc/rc.local** file is designed to require **no** changes when a node is being configured as a standard TCP/IP host. We define a standard TCP/IP host to be a node with only one network connection, to either an Apollo Token Ring network or an ETHERNET, and which uses its Domain node name as its Internet host name. If the node is a gateway, you must edit the **/etc/rc.local** template to configure the gateway's network interfaces. You also may want to edit this file in order to start additional daemons, to change a host name, to add static routes, or to define the host who can write to the **/usr/spool/rwho** directory.

Figure A-1 illustrates the **/etc/rc.local** template. Items of interest within the table are numbered and explained on the facing pages.

```
   #
   #
   # /etc/rc.local  (DOMAIN/OS version SR10)
   #
①  # This script is called from /etc/rc during node startup
   # to enable tcp/ip communications.  It can also be run
   # by invoking '/etc/sys_sh -x /etc/rc.local' as root.
   #
   # For many nodes which have only one network controller,
   #
   #       * THERE IS NO NEED TO EDIT THIS FILE *
   #
②  # You need only make sure that the /etc/hosts file contains
③  # an entry for this node name, create files /etc/daemons/tcpd
   # and /etc/daemons/routed files, and TCP/IP will come up with
   # the following defaults:
   #
④  #     1 interface on either Domain ring or Ethernet:
   #            broadcast address <net>.255.255 (all 1's form)
   #            subnets not enabled
   #            ethernet trailers not enabled
   #            loopback interface not enabled.
   #     routed -h: this specifies routed will exit once
   #             the routing table has stabilized.
   #     rwhod: rwhod will not write to /usr/spool/rwho unless
   #            that directory is local to the node.
   #            (You can use -w to force writes to /usr/spool/rwho)
   #            Only one node per LAN, usually the TCP ADMIN node,
   #            needs to write to (a shared) /usr/spool/rwho dir.
   #
   #
   # For nodes with two network interfaces, you must edit
   # this startup file to invoke the ifconfig utility with
   # the explicit interface names and internet addresses.
   # For nodes in other non-default configurations,
   # futher changes may be required.
   #
   # Refer to "Configuring and Managing TCP/IP" No. 008543-A00.
   #
```

(Continued on Page A-4)

Figure A-1. The /etc/rc.local File Template

Item ① File Called at Startup by /etc/rc

This line indicates that the script is called by the **/etc/rc** file at node startup. If you make any changes to this file, the node must be rebooted in order to put the changes into effect.

Item ② Include Host Internet Name and Address in /etc/hosts

All TCP/IP hosts and gateways must be listed in the **/etc/hosts** file. See Section 3.1, "TCP/IP Configuration Files," for information about **/etc/hosts**.

Item ③ Create Daemon Files

Daemons are started only if they are invoked in **/etc/rc.local** *and* if a file with that name exists in the node's **/etc/daemons** directory. At a minimum, you should create the files **/etc/daemons/tcpd** and **/etc/daemons/routed** on each TCP/IP host and gateway that uses the default version of this startup file. See Section 3.3.11, "Invoking TCP/IP Daemons."

Item ④ Default Descriptions

This section of **/etc/rc.local** describes the default values specified in this template file. If you are configuring a TCP/IP host with only one network connection, to either an Apollo Token Ring network or an ETHERNET (IEEE 802.3), and the default values for all the other items are correct, you do not need to edit this file. However, for TCP/IP gateways, you must make some changes to this file. See the description for **ifconfig** in the appropriate *Managing System Software* manual or the on-line manual page.

The default values are:

1. One network interface, to either an Apollo Token Ring or an ETHERNET (IEEE 802.3) network. If you are configuring a TCP/IP *gateway*, you must specify all the node's network interfaces, using the **ifconfig** utility. See Items 8 through 11.

2. A broadcast address of <net>.255.255. Remove the broadcast parameter from the **ifconfig** command line if your network must use the <net>.0.0 form of broadcast address.

3. Subnets not enabled. If your network system is divided into subnets, define subnet masks using the **ifconfig** utility and the **netmask** parameter.

4. ETHERNET trailers not enabled. If the host you are configuring requires the use of "trailer" link level encapsulation, change the "–trailer" parameter to "trailer" on the **ifconfig** command line. See Item 11 below.

```
  #
  #
  #  Set this system's IP host name.  We recommend that
  #  you LEAVE THIS LINE COMMENTED OUT, and allow the IP
  #  host name to default to the node's Domain name.
  #  Note: for diskless nodes, ctnode -root <diskless name>,
  #  and uctnode -root diskless_$<nodeid>.
⑤ # if [ -f /etc/hostname ]; then
  #     /etc/hostname <ip-host-name>
  # fi
  #
  #
  #  Node must run the tcp daemon to enable the socket() interface.
  #  Tcpd must fork and exit before ifconfig runs,
  #  so DO NOT supply a trailing & on the command line.
⑥ if [ -f /etc/tcpd -a -f /etc/daemons/tcpd ]; then
      (echo " tcpd\c" >/dev/console)
       /etc/tcpd
  fi
  #
  #
  #  Run syslogd if needed for error logging (bsd4.3 & sys5.3 only).
  #  Should be started before rc.local, in case of internet errors.
  #
⑦ if [ -f /etc/syslogd -a -f /etc/daemons/syslogd ] ; then
       (echo " syslogd\c" >/dev/console)
       /etc/syslogd
  fi
  #
  #
  #  Enable network interface(s) for TCP.
  #  If node has only one network interface,
  #  then you can enable it using the pseudo-interface
  #  'net0'. If the node has more than one network
  #  interface, or more than one internet address,
  #  then you must enable each interface using its
  #  explicit interface name (eth0, dr0, eth1, dr1)
⑧ #  and its numeric IP address (e.g, 128.5.1.2).
  #
  #
```

(Continued on Page A-6)

Figure A-1. The /etc/rc.local File Template (Cont.)

Item ⑤ Specify the IP Host Name

If you do leave this command line unchanged, the TCP/IP software will use the node's Domain node name as its Internet name, if the node is cataloged on the Domain network. Make sure this node's name and Internet address is in the **/etc/hosts** file. We recommend that you keep the node name as the default Internet name.

For diskless nodes to use their node names, you must catalog the diskless node name in the root directory and uncatalog the diskless_$<nodeid> name.

If you do change this command, this command records this node's Internet host name in the internal TCP/IP software. Enter the host name in place of <ip-host-name>. The name entered here must be identical to the host name entered in **/etc/hosts** for this host. Host names can contain any printable characters other than field delimiters, newlines, or comment characters. Host names are returned by the **hostname** command.

Item ⑥ Invoke tcpd

This command starts **tcpd**, if the file **/etc/daemons/tcpd** exists. These lines must be uncommented to enable the TCP/IP socket interface, the basic mechanism for TCP/IP communications. See the **tcpd** description in the appropriate *Managing System Software* manual for information about options you can add to the command line.

This command is uncommented in the template file. To allow the TCP/IP server process to be invoked on this node, you must also create a file named **tcpd** in this node's **/etc/daemons** directory.

Item ⑦ Run syslogd

The system logging server, **syslogd**, reads and logs messages into a set of files described by configuration file **/etc/syslog.conf**. See the on–line man page for **syslogd**.

Item ⑧ Enable Network Interfaces

The **ifconfig** command configures the node's network interface(s). Hosts with one network interface on either an Apollo Token Ring or an ETHERNET, are automatically configured by the pseudo–interface, "net0". If you are configuring a gateway, you must enable each interface using its explicit interface name and IP address. The network interface types are:

dr0, dr1 Apollo Token Ring interfaces, which can be either an Apollo Token Ring controller (ATR) or the IIC controller in a Domain/Bridge–A or –B configuration.

eth0, eth1 IEEE 802.3 Network (ETHERNET) interfaces. An *eth* interface can be any Apollo IEEE 802.3 Network controller.

```
    #
⑨   #   If the node is running in a subnetted environment,
    #   you must supply 'netmask <subnetmask>' to ifconfig.
    #
⑩   #   If the node must interoperate with certain older
    #   TCP implementations, then remove 'broadcast 0x0ffffff'
    #   from the ifconfig command line.  The node will then
    #   send the '0-form' IP broadcast address, which is the
    #   one recognized by bsd4.2 and some other older tcp's.
    #
    #   Ifconfig must complete before routed starts up,
    #   so DO NOT supply a trailing & on the command line.
    if [ -f /etc/ifconfig ]; then
⑪   #  /etc/ifconfig lo0  localhost  broadcast 0x0ffffff up
      /etc/ifconfig net0 `/etc/hostname` broadcast 0x0ffffff up -trailers
    #  /etc/ifconfig dr0  <ip-address>  broadcast 0x0ffffff up
    #  /etc/ifconfig eth0 <ip-address>  broadcast 0x0ffffff up -trailers
    #  /etc/ifconfig dr1  <ip-address>  broadcast 0x0ffffff up
    #  /etc/ifconfig eth1 <ip-address>  broadcast 0x0ffffff up -trailers
    fi
    #
    #
    #  Enable dynamic routing.
    #  Options include -s: be a routing supplier, -q: be quiet, -t: trace
    #  rip traffic, -f: flush routing tables, -h: exit when routing tables
    #  stabilize.  See routed manual page for details.
⑫   if [ -f /etc/routed -a -f /etc/daemons/routed ]; then
        (echo " routed\c" >/dev/console)
        /etc/routed -h -f
    fi
    #
    #
    #  Enable additional (static) routes. Using 'addp' causes the static
    #  route to take precedence over dynamic routes added by routed.
    #  See route manual page for details.
⑬   # if [ -f /etc/route ]; then
    #    /etc/route add <destination ip addr> <gateway ip addr> <metric>
    # fi
    #
```

(Continued on Page A-8)

Figure A-1. The /etc/rc.local File Template (Cont.)

Item ⑨ Specify Subnet Mask

Add subnet masks to the **ifconfig** command for every host and gateway if your network system is subnetted, using the **netmask** <*mask*> parameter. See the description of **ifconfig** in the appropriate *Managing System Software* manual or in the on–line manual page for information about how to specify subnet masks.

Item ⑩ Specify Broadcast Address

Use the **broadcast** parameter with the **ifconfig** command to change the broadcast address used by the node being configured. The default broadcast address is the address with a host part of all 1's.

Item ⑪ Enable Loopback Interface

Uncomment this line to enable the node's software loopback interface to be accessed using the name **localhost**.

Item ⑫ Configure Dynamic Routing

This command starts the routing daemon, **routed**. The template file specifies the **–h** option, which should be used on hosts to stop the process after **routed** has acquired all known routes and routing information stabilizes. This option releases the host's memory and CPU time by reducing the number of active processes.

Gateways that are routing suppliers run **routed** continuously and ignore the **–h** option, unless the **–q** option also is supplied.

Item ⑬ Configure Static Routes

These lines add permanent routes to the node's internal routing tables. **Routed** will *not* delete these routes. See the **route** description in the appropriate *Managing System Software* manual for information about the options you can use on this line.

```
  #
  #  Establish IP-to-MAC address mappings for remote hosts
  #  that do not implement ARP (address resolution protocol).
  #
⑭ # if [ -f /etc/arp ]; then
  #     /etc/arp -s <ip address> <ethernet address>
  #     /etc/arp -f /etc/arp.conf
  # fi
  #
  #  Run inetd to manage internet daemons
  #  (ftpd, rshd, rexecd, rlogind, and telnetd).
  #  Edit /etc/inetd.conf to specify which
  #  daemons are enabled. (bsd4.3 and sys5.3 only)
  #
⑮ if [ -f /etc/inetd -a -f /etc/daemons/inetd ]; then
           (echo " inetd\c" >/dev/console)
           /etc/inetd
  fi
  #
  #  Enable the Berkeley Internet name server.
  #  This server replaces /etc/hosts as the source for
  #  internet name and address information.
  #
⑯ # if [ -f /etc/named -a -f /etc/named.boot \
  #    -a -f /etc/daemons/named]; then
  #    (echo " named\c" > /dev/console)
  #    /etc/named
  # fi
  #
  #  Enable rwho and ruptime information.
  #  Use rwhod -w to enable writes to /usr/spool/rwho.
  #  The default is to write to the directory, but not
  #  if it is a link to a directory on another node.
  #
⑰ if [ -f /etc/rwhod -a -f /etc/daemons/rwhod ]; then
        (echo " rwhod\c" >/dev/console)
        /etc/rwhod
  fi
```

Figure A-1. The /etc/rc.local File Template (Cont.)

Item ⑭ Establish Address Mapping

These commands allow you to enter in this node's address mapping table hosts that do *not* use the Address Resolution Protocol (ARP). The ARP protocol establishes correspondences between IP internet addresses and local ETHERNET addresses. ARP is widely implemented by hosts on IP internets. When hosts do not use ARP, the **arp** command lines provide the required IP–to–ETHERNET address mappings.

Edit and uncomment these lines to add a single host. You also can add hosts by editing the **/etc/arp.conf** file. See the **arp** description in the appropriate *Managing System Software* manual for more information about this program.

Item ⑮ Enable Other Internet Daemons

These lines enable the **inetd** program, which manages other IP internet daemons, if the file **/etc/daemons/inetd** exists. If you want the following daemons to run, they must be invoked through inetd: **ftpd, rshd, rexecd, rlogind, telnetd, tftpd**. The template file **/etc/rc.local** has these lines uncommented. You do not need to create files in the **/etc/daemon** directory for processes invoked by **inetd**

To specify the servers for the local host that can be invoked by **inetd**, edit the **/etc/inetd.conf** file. This file is a link to **‘node_data/etc/inetd.conf**. See the Manual Page for each daemon for information about options. See the **inetd.conf** Manual Page and the **/etc/inted.conf** template file for more information about configuring **inetd.**

Item ⑯ Enable the Name Server

This command starts **named**, the Berkeley Internet name server. Uncomment this command if you have configured your network to use **named**. See Chapter 4 for more information about **named.**

Item ⑰ Enable the System Status Server (rwhod)

This command enables **rwhod**, if the file /etc/daemons/rwhod exists. The default condition is to write to the /usr/spool/rwho directory if it is physically located on the node but not to write to it if the directory is a link. The –w option allows a node to write to the **/usr/spool/rwho** directory even though the directory is not physically located on that node.

On Apollo networks, every host does not need to maintain its own **/usr/spool/rwho** directory. One host per network should be allowed to write to the directory. All other hosts link to the master directory to access the information but do not write to it.

Choose one master host per network to write to the /usr/spool/rwho directory. (We recommend that you use the TCP/IP administrative host.) Then, either locate the directory physically on that node or start **rwhod** with the –w option on that node. The master host will receive broadcasts and update the /usr/spool/rwho directory normally. Configure all other hosts to run **rwhod** without any options and create links on those hosts to the **/usr/spool/rwho** directory on the master host.

See the **rwhod** description in the appropriate *Managing System Software* manual for more information.

Appendix B

TCP/IP Implementation Constants and Error Messages

This appendix contains some TCP/IP implementation constants that may be of interest to application programmers. It also contains the error messages you may see when attempting to use TCP/IP or any of the BSD4.3 socket IPC facilities. We have tried to include, where appropriate, information about what may be causing the error, and how to fix that condition. We also have tried to indicate whether a program error, an error in configuring TCP/IP, or an anomalous network condition is most likely to be responsible for generating a particular message.

B.1 Constants

B.1.1 Interface Constants

The following are network interfaces recognized by the TCP/IP software.

Type	Abbreviation	Maximum Transmission Unit
Apollo Token Ring	dr (dr0, dr1)	1268 bytes
ETHERNET (IEEE 802.3)	eth (eth0, eth1)	1500 bytes
Hyperchannel	hy	1500 bytes

B.1.2 IP Constant

Maximum Time to Live (MAXTTL)

Defaults to 30. Is selectable by **tcpd**.

B.1.3 ICMP Constant

Ping

The **tcpd** server process pings gateways which are in use by open connections on that device at 4 second intervals. It retries 3 times before demoting that routing entry to the end of the routing table.

B.1.4 UDP Constant

Maximum datagram size = 9100 bytes.

B.1.5 TCP Constants

Window Size

Default and maximum window size = 9100 bytes. (0x239c)
Minimum window size = 2048 bytes. (0x800)
See **tcpd** options.

Delayed ACK

Up to one ACK can be deferred. See **tcpd** options.

Retransmission Time

2 seconds, up to last round trip time.

Retransmit Too Long Timeout

30 seconds.

Zero Window Probe Interval

5 seconds, up to last round trip time.

B.2 Error Messages

Address already in use

This message is the result of a programming error or an error in configuring TCP/IP. The programming error occurs when an application attempts to reopen a connection on an address before a previous occurrence of that address closed or was timed out. If you have attempted to start the same server twice (on one host), you will also get this message. For example, this error message would occur if you tried to start **ftpd** on a host that already had **ftpd** running.

All network ports in use

No **ptys** (pseudo–tty devices that provide socket connection facilities) are available or all ptys already have **telnet** connections active. First, check the **/dev** directory to be sure at least one pty pair exists. The following example shows the command to list all device types in the **/dev** directory and typical output displayed by the command.

```
% ls /dev/*typ*

/dev/ptyp0 /dev/ptyp1 /dev/ttyp0 /dev/ttyp1
```

You must have at least one pair of ptys (for example, **ptyp0** and **ttyp0**) in **/dev**. If you don't, run **/etc/crpty** to create the ptys. We recommend that 8 or 16 ptys be installed. The example below creates eight pairs of ptys.

```
% /etc/crpty 8
```

Bad local address, port, link, or protocol

This message is usually caused by a programming error. You have misused the socket interface.

Bad net open mode

This message is usually caused by a programming error. You have attempted to perform a bad open operation on a socket.

Can't assign requested address

This could be caused by a program error if the program tries to perform an operation, usually a *bind*, on an Internet address that is different from the host's Internet address. It might also be an error in the **ifconfig** command line in the node's **/etc/rc.local** file. Check the **/etc/rc.local** file to make sure that the network address for this host defined with **ifconfig** agrees with the Internet address for this host in the network's **/etc/hosts** file.

Can't send after socket shutdown

This is the result of a programming error; the program attempted to send data to a socket that the program had already shut down with the **shutdown** call.

Connection refused

You tried to connect to a remote host for a service that didn't have a server running. For example, you may have tried to **telnet** to a host that didn't have **telnetd** running. In this instance, the remote host *actively* refused the connection.

Connection reset by peer

This error message generally results from a problem with the network. The connection on the network was broken for some period longer than a timeout set by the remote system. Try to reconnect.

Connection timed out

This error might result from a network problem or a configuration error. The message signifies that you attempted to connect to a machine and received no response. The network connection may be broken, the remote host may not be running TCP/IP, or there may be no gateway entry on the foreign host for the Domain gateway.

Check that **routed** is running on the gateway. Also check the foreign host to see if it recognizes your Domain network. If the foreign host runs UNIX, you can use the **netstat -r** command at that host to check the networks that it recognizes. If the Domain network doesn't appear on the listing, use the **route** command to add your network.

Destination address required

This error message results from a programming error. Some of the BSD socket calls require an address, and when this error returns, the address was not supplied in the program.

Destination dead

The remote host system to which you were connected is no longer responding.

Destination not responding (timeout)

The remote host system to which you were connected is no longer responding.

Destination refused

You tried to connect to a remote host for a service that didn't have a server running. For example, you may have tried to **telnet** to a host that didn't have **telnetd** running. In this instance, the remote host *actively* refused the connection.

Destination unreachable

The software could not find a path to the destination. Could be caused by a bad network or by a crashed gateway.

End of file (foreign peer closed)

Caused by a loss of connection during data transfer (**ftp**). The remote side of the connection has unexpectedly gone down.

Invalid request

This error message occurs in many cases; it is a catchall for a number of miscellaneous error conditions. The most likely cause of it is programmer error.

I/O error

This error message occurs in many cases; it is a catchall for a number of miscellaneous error conditions. The most likely cause of it, however, is that **tcpd** is not running, or is running in some undetectably altered state. Stop any TCP/IP proceses that are running on this host, and restart them.

Message too long

This error is caused by a programming error. The program attempted to send a datagram larger than legal size over a UDP socket.

Network dropped connection on reset

The remote host system to which you were connected crashed and rebooted. Reconnect to the remote system.

Network is unreachable

Your **tcpd** does not have a gateway entry for the network you're trying to reach. Check the physical path to the remote host; then use **route** to add the route to your routing table. If neither of these actions solves the problem, make sure there is a gateway entry for the foreign network in **/etc/hosts**; if not, add the entry.

Network not available

Your node cannot reach the network. This might be caused by an error in the **ifconfig** command line in the node's **/etc/rc.local** file. Check the **/etc/rc.local** file to make sure that the interfaces for this node are defined correctly. Also could be caused by a hardware failure.

No free network buffers

Since network buffers are held in global space, you have reached a resource limit. Probably, there are too many TCP connections open.

Operation attempted on unopened connection

This message is usually caused by a programming error. You have attempted to perform an operation without first opening a connection successfully.

Operation not supported on socket

This message is usually caused by a programming error. You have attempted to perform an operation on a type of socket that does not support that operation.

Protocol family not supported; Protocol not available; Protocol not supported.

These three messages are caused by programming errors. Your program has tried to create a socket type other than an Internet socket.

Protocol wrong type for socket

You specified a protocol which does not support the semantics of the socket type you requested in the program. For example, you specified the UDP protocol with type SOCK_STREAM.

Raw message send failed

This message is usually caused by a programming error. You have attempted to perform a bad send operation on a raw socket.

Read or write on closed connection

This message is usually caused by a programming error.

Socket is already connected

Your program attempted to connect to an already–connected socket, or a BSD **sendto** or **sendmsg** request specified a destination other than the connected socket.

Socket is not connected

Your program attempted to send or receive data to or from a socket that was not connected.

Socket operation on non–socket

This message results from a programming error. The program attempted to perform an operation on a file descriptor that was not associated with a socket; the attempted operation can only be performed on a socket.

Socket type not supported

You specified a socket type in your program that is not supported on this machine.

TCP/IP server is not running

You have attempted an operation that required **tcpd** but the server is not running. Start the server. You may have to start other TCP/IP server processes also.

Too many connections

You have reached a resource limit. Reduce the number of operations you are trying to perform.

Unknown error

This error message occurs in many cases; it is a catchall for a number of miscellaneous error conditions. The most likely cause of it is programmer error.

Unknown host name for your address

This is a configuration problem. There is no entry in the foreign host's host table for your Domain host. Add the Domain host's name to **/etc/hosts**, and be certain that the network address is entered in **/etc/networks**. On other types of hosts, you may have to edit an equivalent file.

Unknown host specifier

This is a configuration problem. There is no entry in your **/etc/hosts** file for this host. Add the host's name to **/etc/hosts,** and be certain that the network address is entered in **/etc/networks**.

Unknown service

This message could result if the destination network is down, or if the **/etc/services** file on either machine has been deleted or corrupted. See Appendix A of this manual for information about **/etc/services**.

Appendix C

Named Database Files and the Standard Resource Record Format

This Appendix describes the contents of the Berkeley Internet Domain (BIND) name server (**named**) database files and also explains the Standard Resource Record Format used in those files.

C.1 Database File Formats

The following examples of **named** files use the device names, domain labels, and Internet addresses of the network system shown in Figure C–1.

C.1.1 The named.boot File

The /etc/**named.boot** file is the first file **named** reads when it executes. This file specifies the type of server **named** will be (that is, primary or secondary master server, or caching only server), the server's zone of authority, and the locations of the source files for its name/address data. This boot file must be located on all hosts running **named**. Figure C–2 shows a sample **named.boot** file for a **primary** master server for one of the subnets in Figure C–1. The numbered lines are explained in detail below the figure. Figure C–3 shows a sample **named.boot** file for a **secondary** master server on the same subnet and Figure C–4 illustrates a sample **named.boot** file for a **caching only** server.

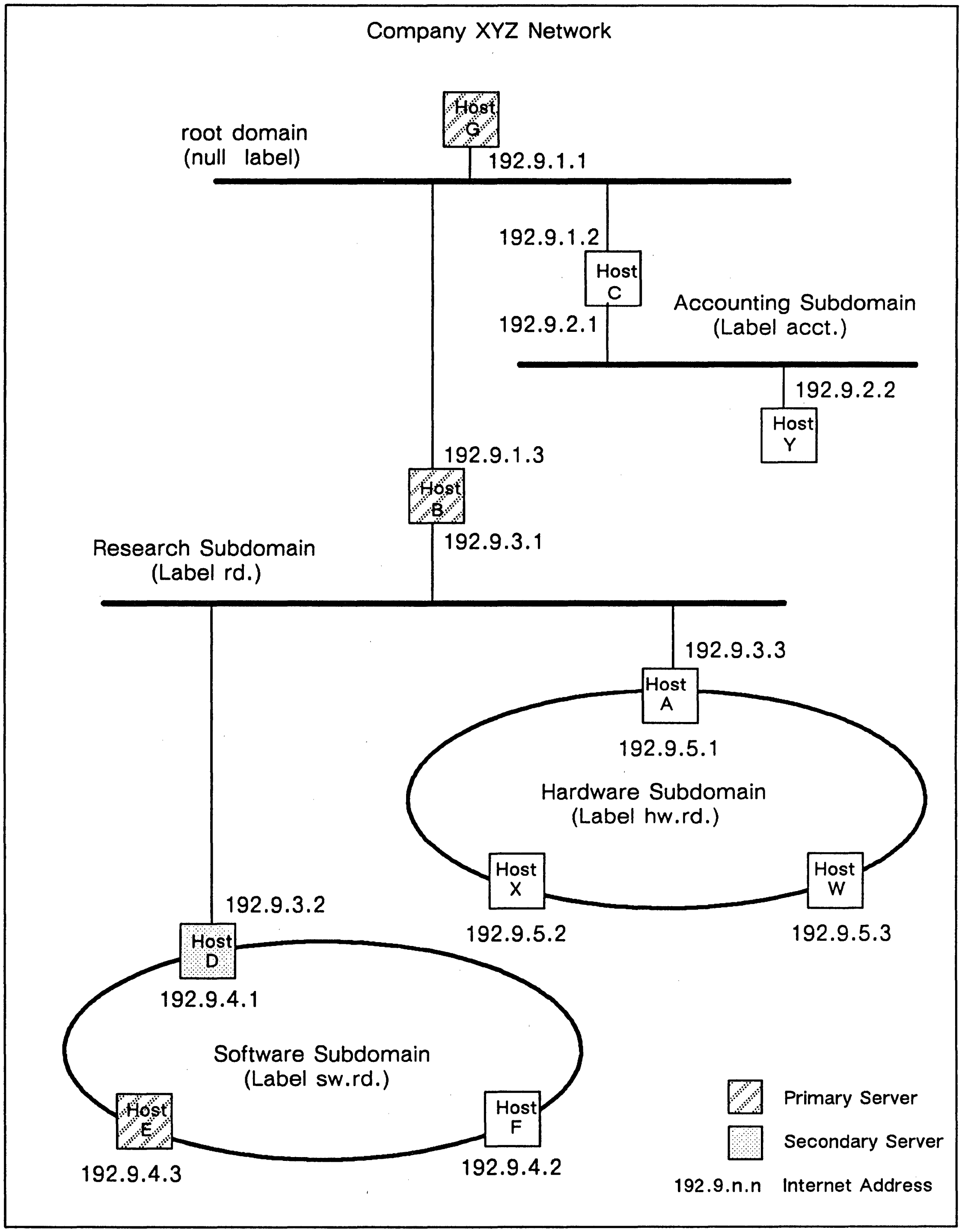


Figure C–1. Sample Network Using the Name Server

```
     ;  named.boot -- boot file for 4.3BSD name server
     ;  HostE is primary master server on Software subdomain
     ;
     ; Type    Domain                Source File or Host
     ; ----    ------                ----------------
     ;
①   domain   sw.rd.
     ;
②   primary  sw.rd.                 /etc/named.hosts
     ;
③   cache    .                      /etc/named.ca
     ;
④   primary  9.192.in-addr.arpa     /etc/named.rev
     ;
⑤   primary  127.in-addr.arpa       /etc/named.local
```

Figure C–2. A named.boot File for Primary Master Server

Item ① This line must be in every boot file. It designates the default domain for the server. When the server receives a query for a name that does not include a "." (that is, with no domain labels specified), the server appends the name in the second field of this line to the query name.

Item ② This line tells the server that it is a **primary** master server for zone sw.rd and that it should read the host data for this zone in the file **/etc/named.hosts.** See Section C.1.3 for more information about the **/etc/named.hosts** file.

Item ③ This line also must be in every boot file. It points to the file whose contents are used to prime the name server's cache with the address of the authoritative name server to which this server should send unresolved queries. A free standing "." in the name field stands for the current domain. See Section C.1.2 for more information about the **/etc/named.ca** file.

Item ④ This line tells the server that it is a **primary** master server for zone 9.192.in–addr.arpa and that it should read the host data for this zone in the file **/etc/named.rev.** Notice that the first two labels of this domain correspond to the internet address of the network as a whole. The domain "in–addr.arpa" is a special domain developed to allow inverse mapping (that is, address–to–domain name), since Internet addresses do not necessarily correspond to domain boundaries. See Section C.1.4 for more information about the file **/etc/named.rev.**

Item ⑤ This line tells the server that it is a **primary** master server for zone 127.in–addr.arpa and that it should read the host data for this zone in the file

/etc/named.local. This zone is defined for the local loopback interface, commonly known as **localhost** with the standard network address of 127.0.0.1. See Section C.1.5 for more information about the file **/etc/named.local.**

The following figure shows a sample named.boot file for a **secondary** master server.

```
   ;  named.boot -- boot file for 4.3BSD name server
   ;  HostD is secondary master server on Software subdomain
   ;
   ; Type        Domain                    Source File or Host
   ; ----        ------                    -------------------
   ;
①  domain      sw.rd.
   ;
②  cache       .                         /etc/named.ca
   ;
③  secondary   sw.rd.                    192.9.4.3
   ;
④  secondary   9.192.in-addr.arpa        192.9.4.3
   ;
⑤  primary     127.in-addr.arpa          /etc/named.local
```

Figure C-3. A named.boot File for Secondary Master Server

Item ① This line must be in every boot file. It designates the default domain for the server. When the server receives a query for a name that does not include a "." (that is, with no domain labels specified), the server appends the name in the second field of this line to the query name.

Item ② This line also must be in every boot file. It points to the file whose contents are used to prime the name server's cache with the address of the authoritative name server to which this server should send unresolved queries. A free standing "." in the name field stands for the current domain.

Item ③ This line tells the server that it is a **secondary** master server for zone sw.rd and that it should get the host data for this zone from the primary master server at address 192.9.4.3. Multiple primary master servers can be specified on this line and they will be tried in order by the secondary name server until it succeeds in obtaining the data or until it fails to find an available server. If it fails, the TCP/IP software will then use the **/etc/hosts** file for name/address information.

Item ④ This line tells the server that it is a **secondary** master server for zone 9.192.in-addr.arpa and that it should get the host data for this zone from the primary master server at address 192.9.4.3. Multiple primary master servers can be specified on this line.

Item ⑤ This line tells the server that it is a **primary** master server for zone 127.in-addr.arpa and that it should read the host data for this zone in the file **/etc/named.local.** This zone is defined for the local loopback interface, commonly known as **localhost** with the standard network address of 127.0.0.1. This file normally resides on every server.

The following table shows a sample **named.boot** file for a **caching only** server. The major difference between this file and the previous two is the lack of pointers to any host data files. Caching only servers do not need access to host data because they refer all queries to the authoritative servers listed in their **/etc/named.ca** file.

```
;  named.boot -- boot file for 4.3BSD name server

;  File for caching only server on Software subdomain
;
; Type   Domain                Source File or Host
; ----   ------                ----------------
;
domain   sw.rd.
;
cache    .                     /etc/named.ca
;
primary  127.in-addr.arpa      /etc/named.local
```

Figure C-4. Sample named.boot File for a Caching Only Server

C.1.2 The named.ca File

This boot file is pointed to in the **/etc/named.boot** file and contains information that tells the server where to send queries that it cannot resolve itself. The data in this file is added to the server's name/address cache when the server is started. For primary and secondary master servers, this file would contain information about the servers in the next higher zone of authority. For caching only servers, this file would contain information about the primary and secondary servers for the current zone.

Figure C–5 illustrates a cache file for a primary name server. A cache file for a secondary name server for the same zone of authority would be identical to this file.

```
    ;
    ;        name.ca   -- cache file for 4.3BSD name server
    ;        HostE is primary master server in sw.rd subdomain
    ;
    ;        Time to  Addr      Record    Record
    ;  Name   Live    Class      Type      Data
    ; -------------   ----      ------    ------
    ;
①  rd.        9999999   IN        NS       HostB.rd
    ;
②  ..         9999999   IN        NS       HostF
    ;
③  HostB.rd.  9999999   IN         A       192.9.3.1
    ;
④  HostG.     9999999   IN         A       192.9.1.1
```

Figure C–5. A named.ca File for Primary Master Server

Item ① This line tells the local name server that the authoritative name server for the rd subdomain is HostB.rd. The letters NS in the Record Type field indicate that this is a Name Server type of record. For a Name Server type of record, the first field specifies the domain that is serviced by the name server identified in the last field. See Section C.2, "Resource Record Format," for descriptions of the information in the other fields of this record.

Item ② This line tells the local name server that the authoritative name server for the root domain is HostF. The two dots in the Domain Name field indicate the root domain.

Item ③ This line indicates the internet address of HostB.rd. The letter A in the Record Type field indicates that this is an Address record. For an Address type of record, the first field specifies the domain name of the device whose address is given in the last field. See Section C.2 for descriptions of the information in the other fields of this record.

Item ④ This line indicates the internet address of HostG.

Figure C–6 illustrates a cache file for a caching only name server in the Software subdomain. Note that the authoritative name servers listed in this file are the primary and secondary servers for this subdomain.

```
;
;          name.ca   -- cache file for 4.3BSD name server
;          Cache file for caching only server in sw.rd subdomain
;
;                         Time to   Addr      Record    Record
;  Name                     Live    Class      Type      Data
; -------                 -------   ----      ------    ------
;
sw.rd.                    9999999    IN        NS       HostE.sw.rd
;
sw.rd.                    9999999    IN        NS       HostD.sw.rd
;
HostE.sw.rd.              9999999    IN         A       192.9.4.3
;
HostD.sw.rd.              9999999    IN         A       192.9.4.1
```

Figure C–6. A named.ca File for Caching Only Server

C.1.3 The named.hosts File

The **named.hosts** file contains name/address data about the hosts in a prticular zone of authority. The location of this file is pointed to in the name server's boot file. For primary name servers, this file must reside locally and its name is given in the boot file. For secondary name servers, the boot file specifies the internet address of the primary server on which this file resides.

Figure C–7 illustrates an example subdomain zone of authority. Figure C–8 illustrates the **named.hosts** file that resides on HostE, the primary master name server for the zone of authority shown in Figure C–7. The numbered items in the file are explained briefly below the figure. This file uses the Standard Resource Record Format described below in Section C.2. Other types of records that can be included in the **named.hosts** file, such as Host Information, Well Known Services, and several types of Mail records are also described in Section C.2.

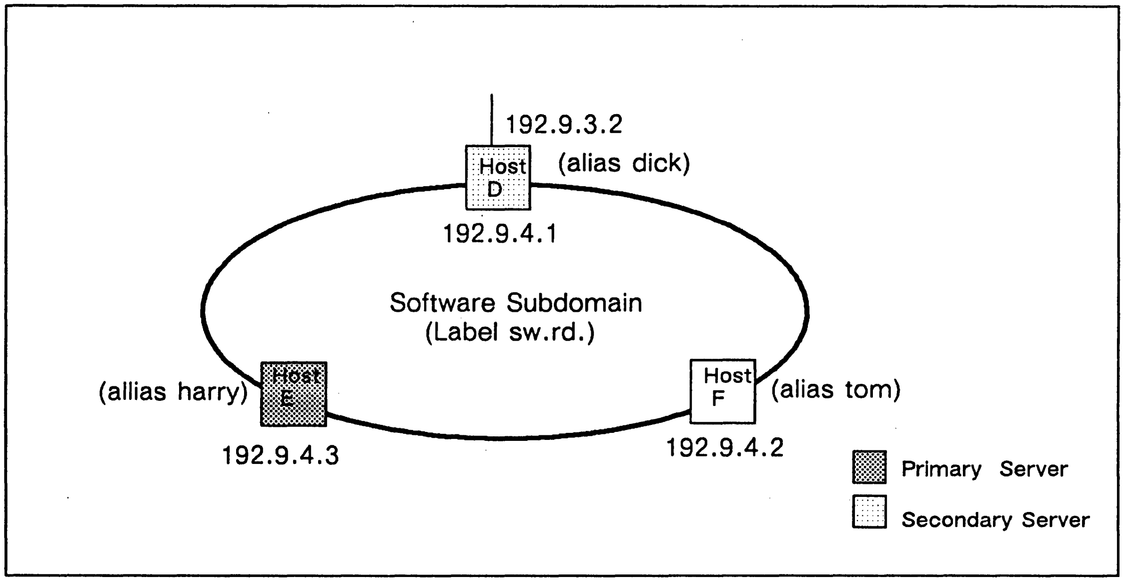


Figure C–7. Software Subdomain in Company XYZ Network

```
     ;        named.hosts file for sw.rd. subdomain
     ;        HostE is primary master server in sw.rd subdomain
     ;
     ;     Time to  Addr  Record   Record
     ;Name  Live   Class   Type     Data
     ;----  ----   -----  ------   ------
     ;
 ①  @                IN    SOA     HostE.sw.rd.   admin.HostE.sw.rd. (
                            1.1        ;serial
                            3600       ;refresh
                            300        ;retry
                            3600000    ;expire
                            3600 )     ;minimum
     ;
 ②                  IN    NS       HostE.sw.rd.
     ;
 ③  localhost        IN    A        127.0.0.1
     HostE            IN    A        192.9.4.3
     HostF            IN    A        192.9.4.2
     HostD            IN    A        192.9.4.1
                      IN    A        192.9.3.2
     ;
 ④  tom              IN    CNAME    HostF
     dick             IN    CNAME    HostD
     harry            IN    CNAME    HostE
     ;
 ⑤  admin            ANY   MB       HostE.sw.rd.
```

Figure C–8. The named.hosts File for HostE

Item ① This line is a Start of Authority record, which designates the start of a zone. The name of the zone of authority should be in the name field. In the example above, @ in the name field denotes the current origin; that is, the origin specified with the previous Origin record. The SOA data record fields indicate the name of the host on which this file resides (HostE.sw.rd.), the mailing address of the person responsible for the name server (admin.HostE.sw.rd.), and other information affecting server operation, such as how ofter a secondary server should refresh its cache from the primary server's data files (3600 seconds). Everything within the parentheses is part of the record. See Section C.2 for more information about the data fields of this record.

Item ② This line is a Name Server record that indicates the name server responsible for this domain, HostE. There should be one NS record for each primary master name server in the domain.

Item ③ This line is an Address record that relates the name and Internet address of the standard loopback interface. The lines immediately following are address records for every host address in the zone. In this example, we have one A record for HostE and HostF and two A records for HostD, which has two Internet addresses.

Item ④ These three lines are Canonical Name records that relate aliases to the canonical names of the hosts in the zone that have aliases.

Item ⑤ This line is a Mailbox record for the person responsible for the name server. The first field contains the user's login (admin); the last field contains the machine name where the user wants to receive the mail (HostE.sw.rd.).

C.1.4 The named.rev File

This file specifies the "in–addr.arpa" domain, which is a special domain for allowing address to name mapping. Since Internet addresses do not necessarily fall within domain boundaries, this domain was created to allow inverse (that is, Internet address to domain name) mapping.

The "in–addr.arpa" domain has four labels preceding it. Each label corresponds to an octet of an Internet address, in reverse order. All four octets must be specified, even if an octet is zero. For example, the Internet address of 192.9.0.1 is expressed as 1.0.9.192.in-addr.arpa.

Figure C–9 illustrates the **/etc/named.rev** file that resides on HostE, the primary server for the domain sw.rd.

```
    ;
    ;         named.rev file for sw.rd. subdomain
    ;         HostE is primary master server in sw.rd subdomain
    ;
    ;      Time to   Addr    Record      Record
    ;Name   Live     Class   Type        Data
    ;----  -------   -----   ------       ------
①  @                  IN      SOA     HostE.sw.rd.   admin.HostE.sw.rd. (
                               1.1            ;serial
                               3600           ;refresh
                               300            ;retry
                               3600000        ;expire
                               3600 )         ;minimum
    ;
②                     IN      NS       HostE.sw.rd.
    ;
③  $ORIGIN 4.9.192.in-addr.arpa.
    ;
④  1                  IN      PTR     HostD.sw.rd.
    2                  IN      PTR     HostF.sw.rd.
    3                  IN      PTR     HostE.sw.rd.
    ;
    $ORIGIN 3.9.192.in-addr.arpa.
    2                  IN      PTR     HostD.sw.rd
```

Figure C–9. The named.rev File for HostE

Item ① This line is a Start of Authority record, which designates the start of a zone. The name of the zone of authority should be in the name field. In the example above, @ in the name field denotes the current origin. The SOA data record fields indicate the name of the host on which this file resides, the mailing address of the person responsible for the name server, and other information affecting server operation.

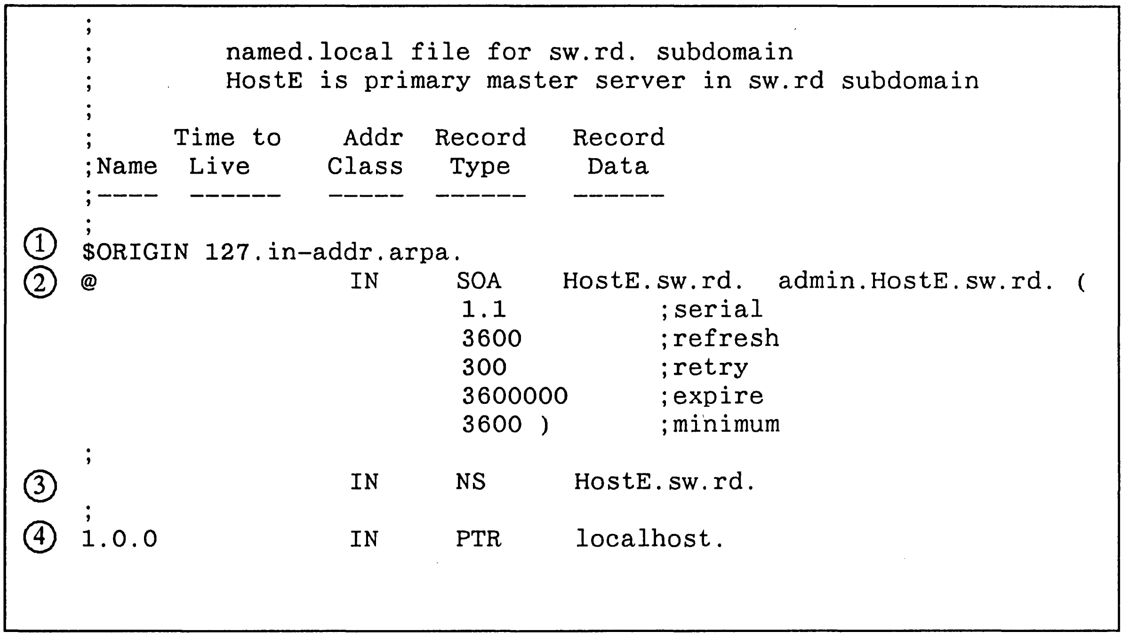
Item ② This line is a Name Server record that indicates the name server responsible for this domain, HostE. There should be one NS record for each primary master name server in the domain.

Item ③ This line is an Origin command that specifies the domain that is described in the following records.

Item ④ This line and the ones following are domain name pointer records that allow special names to point to some other location in the domain. In this case, these pointer records are reverse pointers for the "in–addr.arpa" domain. Pointer names should be unique to the zone.

C.1.5 The named.local File

The **/etc/named.local** file specifies the address for the local loopback interface, known as **localhost**, with the standard network address of 127.0.0.1. Figure C–10 illustrates the **named.local** file for HostE.



```
    ;
    ;        named.local file for sw.rd. subdomain
    ;        HostE is primary master server in sw.rd subdomain
    ;
    ;     Time to    Addr  Record   Record
    ;Name  Live     Class   Type     Data
    ;----  ------   -----  ------   ------
    ;
①  $ORIGIN 127.in-addr.arpa.
②  @                IN     SOA     HostE.sw.rd.  admin.HostE.sw.rd. (
                            1.1          ;serial
                            3600         ;refresh
                            300          ;retry
                            3600000      ;expire
                            3600 )       ;minimum
    ;
③                   IN     NS       HostE.sw.rd.
    ;
④  1.0.0            IN     PTR      localhost.
```

Figure C–10. Sample named.local File

Item ① This line is an Origin command that specifies the domain that is described in the following records.

Item ② This line is a Start of Authority record, which designates the start of a zone. See Section C.2 for a detailed description of this record type.

Item ③ This line is a Name Server record that indicates the name server responsible for this domain, HostE. There should be one NS record for each primary master name server in the domain.

Item ④ This line is a domain name pointer record that points to the "in–addr.arpa" domain. Pointer names should be unique to the zone.

C.2 Resource Record Format

Records in the name server data files are called resource records and have the following standard format:

```
name    ttl    addr-class    record type    record specific data
```

This format is specified in RFC882 and RFC973. The record fields are defined as:

`name` The name of the domain record. In some resource records, the name may be left blank. In that case, the record takes on the name of the previous record.

`ttl` Time to live. This optional field specifies how long this data will be stored in the data base. By leaving this field blank, the default time to live is specified in the Start of Authority resource record.

`addr-class` Address class. Specifies the type of address used in the record. Currently, there are two classes: IN for Internet addresses, and ANY for all address classes.

`record type` Specifies the type of this record. Types include Start of Authority, Name Server, and Address, among others.

`data` Content of the fields that follow the record type depend on the type of record.

Case is preserved in names and data fields when loaded into the name server. All comparisons and lookups in the name server data base are case insensitive.

C.2.1 Special Characters

The following characters have special meanings in resource records:

. 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.

\X Quotes character X, where X is any character other than a digit (0–9), so that its special meaning does not apply. For example, \. can be used to place a dot character in a label.

\DDD Is the octet corresponding to the decimal number described by digits DDD. Resulting octet is assumed to be text and is not checked for special meaning.

( ) Parentheses are used to group data that crosses a line. In effect, line terminations are not recognized within parentheses.

; Semicolon starts a comment. The remainder of the line is ignored.

* An asterisk signifies wildcarding.

Most resource records will have the current origin appended to names if they are not terminated by a dot. This is useful for appending the current domain name to the data, such as maching names, but may cause problems where you do not want this to happen. We suggest that if the name is not in the domain for which you are creating the data file, end the name with a dot.

C.2.2 Record Types

C.2.2.1 Start of Authority - SOA

The Start of Authority, SOA, record designates the start of a zone. There should only be one SOA record per zone. The format of this record follows the standard format illustrated above; a major difference from other records, however, is the amount of record specific data included. The following example illustrates an SOA record. The fields in the record specific data are explained after the example.

```
name  {ttl}  addr-class  rec-type    Origin          Person in charge
@            IN          SOA         HostE.sw.rd.    admin.HostE.sw.rd. (
                         1.1         ; Serial
                         3600        ; Refresh
                         300         ; Retry
                         3600000     ; Expire
                         3600 )      ; Minimum
```

Origin The name of the host on which this data file resides.

Person in charge The mailing address for the person responsible for the name server.

Serial number Version number of this data file. Increment this number whenever you change the data. The name server cannot handle numbers over 9999 after the decimal point.

Refresh Indicates how often, in seconds, a secondary name servers is to check with the primary name server to see if an update is needed.

Retry Indicates how long, in seconds, a secondary server is to retry after a failure to check for a refresh.

Expire The upper limit, in seconds, that a secondary name server is to use the data before it expires for lack of getting a refresh.

Minimum The default number of seconds to be used for the time to live field on resource records.

C.2.2.2 Name Server - NS

The Name Server record, NS, lists a name server responsible for a given domain. The first name field lists the domain that is serviced by the listed name server. There should be one NS record for each Primary Master server for the domain.

```
{name}   {ttl}    addr-class  rec-type    Name servers name
@                 IN          NS          HostE.sw.rd.
```

C.2.2.3 Address - A

The Address record, A, lists the address for a given machine. The name field is the machine name and the address is the network address. There should be one A record for each address of the machine.

```
{name}     {ttl}    addr-class  rec-type    address
HostD               IN          A           192.9.4.1
                    IN          A           192.9.3.2
```

C.2.2.4 Host Information - HINFO

The Host Information resource record, HINFO, is for host specific data. This record lists the hardware and operating system that are running at the listed host. Note that only a single space separates the hardware information and the operating system information. If you want to include a space in the machine name, you must quote the name. Host information is not specific to any address class, so ANY may be used for the address class. There should be one HINFO record for each host.

```
{name}     {ttl}    addr-class  rec-type    Hardware     OS
                    ANY         HINFO       VAX-11/780   UNIX
```

C.2.2.5 Well Known Services - WKS

The Well Known Services record, WKS, describes the well known services supported by a particular protocol at a specified address. The list of services and port numbers come from the list of services specified in /etc/services. There should be only one WKS record per protocol per address.

```
{name}  {ttl} addr-class rec-type  address    proto   list of services
              IN         WKS       192.9.4.1  UDP     who route timed
              IN         WKS       192.9.4.1  TCP     domain(echo telnet
                                                      discard unrpc sftp
                                                      uucp-pathsystat
                                                      daytime netstat
                                                      domain nameserver)
```

C.2.2.6 Canonical Name - CNAME

The Canonical Name resource record, CNAME, specifies an alias for a canonical name. An alias should be unique and all other resource records should be associated with the canonical name and not with the alias. Do not create an alias and then use it in other resource records.

aliases	{ttl}	addr-class	rec-type	Canonical name
harry		IN	CNAME	HostE

C.2.2.7 Domain Name Pointer - PTR

A Domain Name Pointer record, PTR, allows special names to point to some other location in the domain. The example below of a PTR record is used in setting up reverse pointers for the special IN-ADDR.ARPA domain. This example record is from a hosts.rev file. PTR names should be unique to the zone.

{name}	{ttl}	addr-class	rec-type	real name
1.0.0		IN	PTR	localhost

C.2.2.8 Mailbox - MB

MB is the Mailbox record. This record lists the machine where a user wants to receive mail. The name field is the users login; the machine field denotes the machine to which mail is to be delivered. Mail Box names should be unique to the zone.

{name}	{ttl}	addr-class	rec-type	Machine
admin		IN	MB	HostE.sw.rd.

C.2.2.9 Mail Rename - MR

Mail Rename, MR, can be used to list aliases for a user. The name field lists the alias for the name listed in the fourth field, which should have a corresponding MB record.

{name}	{ttl}	addr-class	rec-type	corresponding MB
Postmistress		IN	MR	admin

C.2.2.10 Mailbox Information - MINFO

The Mail Information record, MINFO, creates a mail group for a mailing list. This resource record is usually associated with a Mail Group record, but may be used with a Mail Box record. The name specifies the name of the mailbox. The requests field is where requests to be added to a mail group should be sent. The maintainer is a mailbox that should receive error messages. This is particularly appropriate for mailing lists when errors in members names should be reported to a person other than the sender.

{name}	{ttl}	addr-class	rec-type	requests	maintainer
BIND		IN	MINFO	BIND-REQUEST	kjd.Berkeley.Edu.

C.2.2.11 Mail Group Member - MG

The Mail Group record, MG, lists members of a mail group.

{mail group name}	{ttl}	addr-class	rec-type	member name
		IN	MG	Bloom

The following is an example for setting up a mailing list.

```
Bind        IN     MINFO    Bind-Request            kjd.Berkeley.Edu.
            IN     MG       Ralph.Berkeley.Edu.
            IN     MG       Zhou.Berkeley.Edu.
            IN     MG       Painter.Berkeley.Edu.
            IN     MG       Riggle.Berkeley.Edu.
            IN     MG       Terry.pa.Xerox.Com.
```

C.2.2.12 Mail Exchanger - MX

Main Exchanger records, MX, are used to specify a machine that knows how to deliver mail to a machine that is not directly connected to the network. In the first example below, Seismo.CSS.GOV. is a mail gateway that knows how to deliver mail to Munari.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 different transport medium. 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 RFC974 for more detailed information.

{name}	{ttl}	addr-class	rec-type	pref value	mailer exchanger
Munnari.OZ.AU.		IN	MX	0	Seismo.CSS.GOV.
*.IL.		IN	MX	0	RELAY.CS.NET.

Wildcard names containing the character "*" may be used for mail routing with MX records. There are likely to be servers on the network that simply state that any mail to a domain is to be routed through a relay. In the second line above, all mail to hosts in the domain IL is routed through RELAY.CS.NET. This is done by creating a wildcard resource record, which states that *.IL has an MX of RELAY.CS.NET.

C.2.3 Commands

In addition to resource records, **named** data files also may contain two types of command lines -- $INCLUDE and $ORIGIN.

C.2.3.1 $INCLUDE

An include line begins with $INCLUDE, starting in column 1, which is followed by a file name. This feature is particularly useful for separating different types of data into multiple files.

An example of an **$INCLUDE** command line is:

```
$INCLUDE /usr/named/data/mailboxes
```

The line would be interpreted as a request to load the file **/usr/named/data/mailboxes.** The $INCLUDE command does not cause data to be loaded into a different zone or tree. This feature allows data for a given zone to be organized in separate files. For example, mailbox data might be kept separately from host data using this mechanism.

C.2.3.2 $ORIGIN

An origin line is a way of changing the origin in a data file. The line starts with $ORIGIN in column 1, which is followed by a domain origin. This is useful for putting more than one domain in a data file.

Appendix D

TCP/IP System Administration Commands and Utilities

This appendix describes the TCP/IP system administration commands and utilities mentioned in this book. It is intended for users who operate in the Aegis environment. Reference material about these commands for the BSD and SysV environments is available on–line and in the *Managing System Software* manuals for those environments.

The command descriptions in this appendix may contain references to other commands in the form of "foo(nx)," where n is a number from 1 to 8 and x is a letter. More information about the referenced commands can be found in the *BSD Command Reference* (n = 1, 6, or 7), *BSD Programmer's Reference* (n = 2, 3, 4, or 5), or *Managing BSD System Software* (n = 8).

NAME

arp – address resolution display and control

SYNOPSIS

arp *hostname*
arp –a
arp –d *hostname*
arp –s *hostname ether_addr* [**temp**] [**pub**] [**trail**]
arp –f *filename*

DESCRIPTION

The **arp** program displays and modifies the Internet-to-ETHERNET* address translation tables used by the address resolution protocol (**arp**(4P)).

With no flags, the program displays the current ARP entry for *hostname*. You may specify the host by name or by number, using Internet dot notation.

OPTIONS

–a Display all of the current ARP entries in the internal file.

–d *hostname* A super-user may delete an entry for the host called *hostname*.

–s *hostname ether_addr* [**temp**] [**pub**] [**trail**]
Create an ARP entry for the host called *hostname* with the ETHERNET address *ether_addr*. The ETHERNET address is given as six hex bytes separated by colons. The entry will be permanent unless you specify the word **temp** in the command. If you specify the word **pub**, the entry will be ''published''; that is, this system will act as an ARP server, responding to requests for *hostname* even though the host address is not its own. The word **trail** indicates that trailer encapsulations may be sent to this host.

–f *filename* Read the file *filename* and set multiple entries in the ARP tables. Entries in the file should be of the form

hostname ether_addr [**temp**] [**pub**] [**trail**]

with argument meanings as given above.

NOTES

ETHERNET is a registered trademark of Xerox Corporation.

SEE ALSO

inet(3N), arp(4P), ifconfig(8C);
Configuring and Managing TCP/IP.

NAME

dtcb – dump contents of tcp control blocks

SYNOPSIS

/etc/dtcb [–f] [[–t] [<*tcb addr*> | **–a**] | **–u** [<*ucb_addr*> | **–a**]]

DESCRIPTION

The command **dtcb** dumps the contents of the **tcp** control blocks associated with a particular tcp connection. The address of the **tcb** to be dumped may be obtained using the **netstat** program. Two control blocks are dumped: the **ucb** (user control block) which contains the send and receive queues and user-related flags, and the **tcb** (tcp control block) which contains the connection sequence numbers, state, flags, and out-of-sequence queues.

OPTIONS

–f	Force output if tcpd not running.
–t <*tcb_addr*>	Hexadecimal address of a **tcb** or, if not supplied, all **tcbs**.
–u <*ucb_addr*>	Hexadecimal address of a **ucb** or, if not supplied, all **ucbs**.
–a	All (both **tcb**'s and **ucb**'s for each socket).

EXAMPLES

The dump of a tcp control block for a listening ftp connection might look like this:

```
$ /etc/dtcb -t 1A9A50
ucb at 0x1A99C4:
local 0.0.0.0  lport 21
host  0.0.0.0  fport 0
uc_snd 8192  uc_ssize 0  uc_rcv 8192  uc_rsize 0
uc_shead 0  uc_stail 0  uc_rhead 0  uc_rtail 0
xflag:
UREUSEADDR UCANACCEPT
iostate:

status:
UCLOSED
flags:
UTCP
oobmark 0  oobcnt 0
```

```
TCB at 0x1A9A50:
lport 0x0  fport 0x0
t_state LISTEN
irs 00000000 rcv_urp 00000000 rcv_urg 00000000 rcv_nxt 00000000
     rcv_end 00000000
iss 1E3202D4 seq_fin 1E3202D4 snd_end 1E3202D4
       snd_urp 00000000 snd_lst 00000000

snd_nxt 1E3202D4 snd_una 1E3202D4 snd_wl 00000000
                                  snd_hi 1E3202D4

rex_val 00000000 rtl_val 00000000 xmt_val 00000000
flags:
snd_wnd 0  maxseg 0  xmtime 2  rxtct 0
timers:
INIT 0  REXMT 0  REXMTTL 0  PERSIST 0  FINACK 0
t_rcv_next 1A9A50  t_rcv_prev 1A9A50
```

NAME

ftpd – DARPA Internet File Transfer Protocol server

SYNOPSIS

/etc/**ftpd** [**–d**] [**–l**] [–t*timeout*]

DESCRIPTION

ftpd is the DARPA Internet File Transfer Protocol server process. The server uses the TCP protocol and listens at the port specified in the ''ftp'' service specification; see **services**(5).

OPTIONS

–d Write debugging information to the syslog.

–l Log each **ftp** session in the syslog.

–t*timeout* Set the inactivity timeout period to *timeout*. Without this option, the **ftp** server will timeout an inactive session after 15 minutes.

FTP REQUESTS

The **ftp** server currently supports the following **ftp** requests; case is not distinguished.

Request	Description
ABOR	Abort previous command
ACCT	Specify account (ignored)
ALLO	Allocate storage (vacuously)
APPE	append to a file
CDUP	Change to parent of current working directory
CWD	Change working directory
DELE	Delete a file
HELP	Give help information
LIST	List files in a directory (''**ls -lg**'')
MKD	Make a directory
MODE	Specify data transfer *mode*
NLST	Give name list of files in directory (''**ls**'')
NOOP	Do nothing
PASS	Specify password
PASV	Prepare for server-to-server transfer
PORT	Specify data connection port
PWD	Print the current working directory
QUIT	Terminate session
RETR	Retrieve a file
RMD	Remove a directory
RNFR	Specify rename-from filename
RNTO	Specify rename-to filename
STOR	Store a file
STOU	Store a file with a unique name
STRU	Specify data transfer *structure*

TYPE	Specify data transfer *type*
USER	Specify username
XCUP	Change to parent of current working directory
XCWD	Change working directory
XMKD	Make a directory
XPWD	Print the current working directory
XRMD	Remove a directory

The remaining **ftp** requests specified in Internet RFC 959 are recognized, but not implemented.

The **ftp** server will abort an active file transfer only when the ABOR command is preceded by a Telnet "Interrupt Process" (IP) signal and a Telnet "Synch" signal in the command Telnet stream, as described in Internet RFC 959.

ftpd interprets filenames according to the ‘‘globbing’’ conventions used by **csh**(1). This allows users to utilize the metacharacters ‘‘*?[]{}~’’.

ftpd authenticates users according to four rules.

1) The username must be in the password data base, /etc/**passwd**, and not have a null password. In this case a password must be provided by the client before any file operations may be performed.

2) The username must not appear in the file /etc/**ftpusers**.

3) The user must have a standard shell returned by **getusershell**(3).

4) If the username is ‘‘anonymous’’ or ‘‘ftp’’, an anonymous ftp account must be present in the password file (user ‘‘ftp’’). In this case the user is allowed to log in by specifying any password (by convention this is given as the client host’s name).

In the last case, **ftpd** takes special measures to restrict the client’s access privileges. The server performs a **chroot**(2) command to the home directory of the ‘‘ftp’’ user. In order that system security is not breached, we recommend that the ‘‘ftp’’ subtree be constructed with care; the following rules are recommended.

~ftp Make the home directory owned by ‘‘ftp’’ and unwritable by anyone.

~ftp/bin Make this directory owned by the super-user and unwritable by anyone. The program **ls**(1) must be present to support the list commands. This program should have mode 111.

~ftp/etc Make this directory owned by the super-user and unwritable by anyone. The files **passwd**(5) and **group**(5) must be present for the **ls** command to work properly. These files should be mode 444.

~ftp/pub Make this directory mode 777 and owned by ‘‘ftp’’. Users should then place files which are to be accessible via the anonymous account in this directory.

CAUTIONS

The anonymous account is inherently dangerous and should be avoided when possible.

The server must run as the super-user to create sockets with privileged port numbers. It maintains an effective user id of the logged in user, reverting to the super-user only when binding addresses to sockets. The possible security holes have been extensively scrutinized, but are possibly incomplete.

NAME

gettable – get NIC format host tables from a host

SYNOPSIS

/etc/gettable [–v] *host* [*outfile*]

DESCRIPTION

gettable is a simple program used to obtain the NIC standard host tables from a ''nicname'' server. The indicated *host* is queried for the tables. The tables, if retrieved, are placed in the file *outfile* or by default, **hosts.txt**.

gettable operates by opening a TCP connection to the port indicated in the service specification for ''nicname''. A request is then made for ''ALL'' names and the resultant information is placed in the output file.

gettable is best used in conjunction with the **htable**(8) program which converts the NIC standard file format to that used by the network library lookup routines.

OPTIONS

–v Get just the version number instead of the complete host table, and puts the output in the file *outfile* or by default, **hosts.ver**.

outfile Place the tables in the specified *outfile*.

CAUTIONS

If the name-domain system provided network name mapping well as host name mapping, **gettable** would no longer be needed.

SEE ALSO

intro(3N), htable(8), named(8);
Configuring and Managing TCP/IP.

NAME

hostns – convert host table files to resource record format

SYNOPSIS

hostns [**–f** *file*] [**–d** *domain*] [**–h** *host*] [**–n** *subnet_id*] [–s *major.minor*]
[**–u** *user*]

DESCRIPTION

The **hostns** command converts host address information obtained from an existing /etc/**hosts** format file (see **hosts**(5)) into Standard Resource Record Format for use by **named**(8). It is intended to serve as an aid in bringing up the name server on an existing network. By default, **hostns** uses host address mapping information contained in the file /etc/**hosts.**

OPTIONS

–f *file* Specify an alternate host address mapping file, named *file*.

–d *domain* Use *domain* as the domain name for the local zone. Normally if the current machine's hostname contains any dots, the portion of the hostname after the first dot is taken to be the domain name for the local zone. If there are no dots in the hostname, the domain name defaults to .MY.DOMAIN.

–h *host* Build files for a different host. Normally **hostns** assumes **named**(8) will be run on the local machine and generates nameserver records.

–n *subnet_id* Limit the conversion to the specific subnet. Normally, **hostns** converts all of the entries in the specified /etc/**hosts** file. *subnet_id* may be specified in hexadecimal or in the standard Internet "dotted address" format. The **–n** option may be specified, with a different *subnet_id*, up to five times.

–s *major.minor* Set the serial number in the Start of Authority (SOA) record to *major.minor* instead of 1.1. The serial number is used to detect updates in a running network.

–u *user* Specify *user* as the responsible administrator. Normally, **hostns** assumes the user responsible for network administration is the user running the program.

FILES

The following files are produced:

named.boot	boot file
named.ca	initial cache data
named.local	localhost reverse pointer
named.hosts	hosts file
named.rev	hosts file reverse pointers

SEE ALSO

hosts(5), named(8), nshost(8);
Configuring and Managing TCP/IP.

NAME

htable – convert NIC standard format host tables

SYNOPSIS

/etc/htable [**–c** *connected-nets*] [**–l** *local-nets*] *file*

DESCRIPTION

The **htable** command is used to convert host files in the format specified in Internet RFC 810 to the format used by the network library routines. Three files are created as a result of running **htable**: **hosts**, **networks**, and **gateways**. The **hosts** file may be used by the **gethostbyname**(3N) routines in mapping host names to addresses if the nameserver, **named**(8), is not used. The **networks** file is used by the **getnetent**(3N) routines in mapping network names to numbers. The **gateways** file may be used by the routing daemon in identifying "passive" Internet gateways; see **routed**(8c) for an explanation.

If any of the files **localhosts**, **localnetworks**, or **localgateways** are present in the current directory, the file's contents are prepended to the output file. Of these, only the gateways file is interpreted. This feature allows sites to maintain local aliases and entries which are not normally present in the master database. Only one gateway to each network will be placed in the gateways file; a gateway listed in the **localgateways** file will override any in the input file.

OPTIONS

–c *connected-nets*

If the gateways file is to be used, use this flag to specify a list of networks to which the host is directly connected. The networks, separated by commas, may be given by name or in Internet-standard dot notation, for example,

–c **arpanet,128.32,local-ether-net**.

htable only includes gateways that are directly connected to one of the specified networks, or that can be reached from another gateway on a connected net.

–l *local-nets* The argument is a comma-separated list of networks to be treated as "local". List format rules are the same as for –**c**. Information about hosts on local networks is taken only from the **localhosts** file. Entries for local hosts from the main database will be omitted. This allows the **localhosts** file to completely override any entries in the input file.

htable is best used in conjunction with the **gettable**(8C) program which retrieves the NIC database from a host. If the name-domain system provided network name mapping well as host name mapping, **htable** would no longer be needed.

SEE ALSO

intro(3N), gettable(8C), named(8);
Configuring and Managing TCP/IP.

NAME

ifconfig – configure network interface parameters

SYNOPSIS

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

DESCRIPTION

ifconfig is used to assign an address to a network interface and/or configure network interface parameters. **ifconfig** must be used at boot time to define the network address of each interface present on a machine. It may also be used at a later time to redefine an interface's address or other operating parameters. The *interface* parameter is a string of the form *name unit*, for example, **eth0**.

Since an interface may receive transmissions in differing protocols, each of which may require separate naming schemes, it is necessary to specify the *address_family*, which may change the interpretation of the remaining parameters. The only address family currently supported by Apollo is **inet**.

For the DARPA-Internet family, the address is either a host name present in the host name data base, **hosts**(5), or a DARPA Internet address expressed in the Internet standard ''dot notation''.

PARAMETERS

The following parameters may be set with **ifconfig**:

up Mark an interface ''up''. This may be used to enable an interface after an ''ifconfig down.'' It happens automatically when setting the first address on an interface. If the interface was reset when previously marked down, the hardware will be re-initialized.

down Mark an interface ''down''. When an interface is marked ''down'', the system will not attempt to transmit messages through that interface. If possible, the interface will be reset to disable reception as well. This action does not automatically disable routes using the interface.

trailers Request the use of a ''trailer'' link level encapsulation when sending (default). If a network interface supports **trailers**, the system will, when possible, encapsulate outgoing messages in a manner which minimizes the number of memory to memory copy operations performed by the receiver. On networks that support the Address Resolution Protocol (see **arp**(4P); currently, only 10 MB ETHERNET), this flag indicates that the system should request that other systems use trailers when sending to this host. Similarly, trailer encapsulations will be sent to other hosts that have made such requests. Currently used by Internet protocols only.

–trailers Disable the use of a ''trailer'' link level encapsulation.

arp Enable the use of the Address Resolution Protocol in mapping between network level addresses and link level addresses (default). This is currently implemented for mapping between DARPA Internet addresses and 10MB ETHERNET addresses.

–arp Disable the use of the Address Resolution Protocol.

metric *n* Set the routing metric of the interface to *n*, default 0. The routing metric is used by the routing protocol (**routed**(8c)). Higher metrics have the effect of making a route less favorable; metrics are counted as addition hops to the destination network or host.

debug Enable driver dependent debugging code; usually, this turns on extra console error logging.

–debug Disable driver dependent debugging code.

netmask *mask* (Inet only) Specify how much of the address to reserve for subdividing networks into sub-networks. The mask includes the network part of the local address and the subnet part, which is taken from the host field of the address. The mask can be specified as a single hexadecimal number with a leading 0x, with a dot-notation Internet address, or with a pseudo-network name listed in the network table **networks**(5). The mask contains 1's for the bit positions in the 32-bit address which are to be used for the network and subnet parts, and 0's for the host part. The mask should contain at least the standard network portion, and the subnet field should be contiguous with the network portion.

dstaddr Specify the address of the correspondent on the other end of a point to point link.

broadcast (Inet only) Specify the address to use to represent broadcasts to the network. The default broadcast address is the address with a host part of all 1's.

ifconfig displays the current configuration for a network interface when no optional parameters are supplied. If a protocol family is specified, **ifconfig** will report only the details specific to that protocol family.

Only the super-user may modify the configuration of a network interface.

DIAGNOSTICS

Messages indicating the specified interface does not exit, the requested address is unknown, or the user is not privileged and tried to alter an interface's configuration.

SEE ALSO

intro(4N), netstat(1), rc(8);
Configuring and Managing TCP/IP.

NAME

inetd – internet ''super–server''

SYNOPSIS

/etc/**inetd** [**–d**] [*configuration file*]

DESCRIPTION

The **inetd** server should be run at boot time by /etc/**rc.local**. It then listens for connections on certain internet sockets. When a connection is found on one of its sockets, it decides what service the socket corresponds to, and invokes a program to service the request. After the program is finished, it continues to listen on the socket (except in some cases which will be described below). Essentially, **inetd** allows running one daemon to invoke several others, reducing load on the system.

Upon execution, **inetd** reads its configuration information from a configuration file which, by default, is /etc/**inetd.conf**. There must be an entry for each field of the configuration file, with entries for each field separated by a tab or a space. Comments are denoted by a ''#'' at the beginning of a line. The fields of the configuration file are as follows:

service name
socket type
protocol
wait/nowait
user
server program
server program arguments

The *service name* entry is the name of a valid service in the file /etc/**services**. For ''internal'' services (discussed below), the service name *must* be the official name of the service (that is, the first entry in /etc/**services**).

The *socket type* should be one of **stream**, **dgram**, **qraw**, **rdm**, or **seqpacket**, depending on whether the socket is a stream, datagram, raw, reliably delivered message, or sequenced packet socket.

The *protocol* must be a valid protocol as given in /etc/**protocols**. Examples might be **tcp** or **udp**.

The *wait/nowait* entry is applicable to datagram sockets only (other sockets should have a *nowait* entry in this space). If a datagram server connects to its peer, freeing the socket so **inetd** can received further messages on the socket, it is said to be a ''multi-threaded'' server, and should use the *nowait* entry. For datagram servers which process all incoming datagrams on a socket and eventually time out, the server is said to be ''single-threaded'' and should use a *wait* entry. **comsat** (**biff**) and **talk** are both examples of the latter type of datagram server. **tftpd** is an exception; it is a datagram server that establishes pseudo-connections. It must be listed as *wait* in order to avoid a race; the server reads the first packet, creates a new socket, and then forks and exits to allow **inetd** to check for new service requests to spawn new servers.

The *user* entry should contain the user name of the user as whom the server should run. This allows for servers to be given less permission than root. The *server program* entry should contain the pathname of the program which is to be executed by **inetd** when a request is found on its socket. If **inetd** provides this service internally, this entry should be **internal**.

The arguments to the server program should be just as they normally are, starting with argv[0], which is the name of the program. If the service is provided internally, the word **internal** should take the place of this entry.

inetd provides several ''trivial'' services internally by use of routines within itself. These services are **echo**, **discard**, **chargen** (character generator), **daytime** (human readable time), and **time** (machine readable time, in the form of the number of seconds since midnight, January 1, 1900). All of these services are tcp based. For details of these services, consult the appropriate RFC from the Network Information Center.

inetd rereads its configuration file when it receives a hangup signal, SIGHUP. Services may be added, deleted or modified when the configuration file is reread.

SEE ALSO

comsat(8C), ftpd(8C), rexecd(8C), rlogind(8C), rshd(8C), telnetd(8C), tftpd(8C); *Configuring and Managing TCP/IP*.

NAME

mbd – dump usage info on tcp buffer pool

SYNOPSIS

/etc/mbd [**–f**] [**–k**]

DESCRIPTION

The **mbd** command dumps usage information about the tcp memory buffer pools. Usage statistics on tcp memory buffers may be obtained by using the **-m** option of the **netstat** command; **mbd** is intended for analyzing cases where buffers are being lost. It scans the entire buffer pool, finding all the chains of in-use buffers; it then prints each chain of buffers. This information may help you in discovering reasons why buffers are being lost.

OPTIONS

–f Dump the free pools as well as the chains of in-use buffers. This produces a lot of output.

–k Don't try to lock the mutex on the buffer pools before doing the dump. This is useful mostly when the **tcpd** has crashed with the buffer pool mutex locked.

EXAMPLES

A dump of the buffer pools of a basically idle tcp might look like this:

```
$ /etc/mbd
Offset 0x000035cc  size  1520 type 1 off   24 len 1512 refcnt 1 pool 1
Offset 0x000041cc  size  1520 type 1 off   24 len 1512 refcnt 1 pool 1
Offset 0x00003bcc  size  1520 type 1 off   24 len 1512 refcnt 1 pool 1
Offset 0x0000a7cc  size  1520 type 1 off   24 len 1512 refcnt 1 pool 1
Offset 0x00004dcc  size  1520 type 1 off   24 len 1512 refcnt 1 pool 1
Offset 0x00007dcc  size  1520 type 1 off   24 len 1512 refcnt 1 pool 1
```

Here there are 6 large (1520-byte) buffers in use, all on a single chain.

NAME

mkhosts – generate hashed host table

SYNOPSIS

/etc/**mkhosts** [–v] [*hostfile*]

DESCRIPTION

The **mkhosts** command is used to generate the hashed host database used by one version of the library routines **gethostbyaddr**(3N) and **gethostbyname**(3N). It is not used if host name translation is performed by **named**(8).

The new database is build in a set of temporary files and only replaces the real database if the new one is built without errors. **mkhosts** will exit with a non-zero exit code if any errors are detected.

OPTIONS

–v List each host as it is added. The default *hostfile* is **/etc/hosts**. If another file is specified, it must be in the format of /etc/**hosts** (see **hosts**(5)). **mkhosts** will generate database files named *hostfile*.**pag** and *hostfile*.**dir**.

FILES

hostfile.**pag**	- real database filenames
hostfile.**dir**	
hostfile.**new.pag**	- temporary database filenames
hostfile.**new.dir**	

SEE ALSO

gethostbyname(3), gettable(8), hosts(5), htable(8), named(8);
Configuring and Managing TCP/IP.

NAME

named – internet domain name server

SYNOPSIS

named [**–d** *debuglevel*] [**–p** *port#*] [*bootfile*]

DESCRIPTION

named is the internet domain name server (see RFC883 for more details). Without any arguments, **named** reads the default boot file /etc/**named.boot**, reads any initial data, and listens for queries.

OPTIONS

–d *debuglevel* Print debugging information. A number after the **d** determines the level of messages printed.

–p *port#* Use a different port number. The default is the standard port number as listed in /etc/**services**.

bootfile Any additional argument is taken as the name of the boot file. The boot file contains information about where the name server is to get its initial data.

EXAMPLE

The following example shows a boot file:

```
;
;     boot file for name server
;
; type        domain       source file or host
;
domain       berkeley.edu
primary            berkeley.edu    named.db
secondary    cc.berkeley.edu 10.2.0.78 128.32.0.10
cache                        named.ca
```

The first line specifies that **berkeley.edu** is the domain for which the server is authoritative. The second line states that the file **named.db** contains authoritative data for the domain **berkeley.edu**. The file **named.db** contains data in the master file format described in RFC883, except that all domain names are relative to the origin; in this case, **berkeley.edu** (see below for a more detailed description).

The third line specifies that all authoritative data under **cc.berkeley.edu** is to be transferred from the name server at 10.2.0.78. If the transfer fails it will try 128.32.0.10 and continue trying the address, up to 10, listed on this line. The secondary copy is also authoritative for the specified domain.

The fourth line specifies data in **named.ca** is to be placed in the cache (i.e., well known data such as locations of root domain servers). The file **named.ca** is in the same format as **named.db**.

MASTER FILE FORMAT

The master file consists of entries of the form:

$INCLUDE <*filename*>
$ORIGIN <*domain*>
<*domain*> <*opt_ttl*> <*opt_class*> <*type*> <*resource_record_data*>

where *domain* is dot ‘‘ .’’ for root, ‘‘@’’ for the current origin, or a standard domain name. If *domain* is a standard domain name that does not end with ‘‘.’’, the current origin is appended to the domain. Domain names ending with ‘‘.’’ are unmodified. The *opt_ttl* field is an optional integer number for the time-to-live field. It defaults to zero. The *opt_class* field is the object address type; currently only one type is supported, **IN**, for objects connected to the DARPA internet. The *type* field is one of the following tokens; the data expected in the *resource_record_data* field is in parentheses.

A	A host address (dotted quad)
NS	An authoritative name server (domain)
MX	A mail exchanger (domain)
CNAME	The canonical name for an alias (domain)
SOA	Marks the start of a zone of authority (5 numbers (see RFC883))
MB	A mailbox domain name (domain)
MG	A mail group member (domain)
MR	A mail rename domain name (domain)
NULL	A null resource record (no format or data)
WKS	A well-known service description (not implemented yet)
PTR	A domain name pointer (domain)
HINFO	Host information (cpu_type OS_type)
MINFO	Mailbox or mail list information (request_domain error_domain)

NOTES

The following signals have the specified effect when sent to the server process using the **kill**(1) command.

SIGHUP	Causes server to read **named.boot** and reload database.
SIGINT	Dumps current data base and cache to **/usr/tmp/named_dump.db**
SIGUSR1	Turns on debugging; each SIGUSR1 increments debug level.
SIGUSR2	Turns off debugging completely.

FILES

/etc/**named.boot**	name server configuration boot file
/etc/**named.conf**	configuration file
/etc/**named.pid**	the process id
/usr/**tmp/named.run**	debug output
/usr/**tmp/named_dump.db**	dump of the name servers database

Configuration files read by /etc/**named.boot**:
/etc/**named.ca**
/etc/**named.hosts**
/etc/**named.local**
/etc/**named.rev**

SEE ALSO

kill(1), gethostbyname(3N), signal(3c), resolver(3), resolver(5);
Configuring and Managing TCP/IP;
RFC882, RFC883, RFC973, RFC974, *Name Server Operations Guide for BIND.*

NAME

nshost – generate host tables from the name server

SYNOPSIS

nshost [**–f** *file* ...] [**–d** *domain* ...] [**–s** *server*] [**–o** *output-file*] [**–r**]

DESCRIPTION

The **nshost** command converts host and address information from name server configuration files (see **named**(8)) or from running name server(s) into the /etc/**hosts** format (see **hosts**(5)) or into the /etc/**named.rev** (reverse pointer records) format.

OPTIONS

–f *file* Search the *file* for Standard Resource Records specifying addresses (A) or aliases (CNAME); these are then written to the standard output or the specified *output-file*. You may specify more than one **–f** *file* option.

–d *domain* Query a running name server for the specified domain (determined by looking in the file /etc/**resolv.conf**) for this same information. You may specify more than one **–d** option. See the **–s** option.

–s *server* Specify alternate servers. You may specify up to three **–s** options per **–d** option.

–r Request the /etc/**named.rev** (reverse pointer records) format.

SEE ALSO

hosts(5), hostns(8), named(8);
Configuring and Managing TCP/IP.

NAME

ping – send ICMP ECHO_REQUEST packets to network hosts

SYNOPSIS

/etc/**ping** [**–d**] [**–r**] [–v] *host* [*packetsize*] [*count*]

DESCRIPTION

The DARPA Internet is a large and complex aggregation of network hardware, connected together by gateways. Tracking a single-point hardware or software failure can often be difficult. The **ping** program utilizes the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams (''pings'') have an IP and ICMP header, followed by a **struct timeval**, and then an arbitrary number of ''pad'' bytes used to fill out the packet. Default datagram length is 64 bytes, but this may be changed using the command-line option.

When using **ping** for fault isolation, you should first run it on the local host, to verify that the local network interface is up and running. Then, hosts and gateways further and further away should be ''pinged''. **ping** sends one datagram per second, and prints one line of output for every ECHO_RESPONSE returned. No output is produced if there is no response. If you specify an optional *count*, only that number of requests is sent. Round-trip times and packet loss statistics are computed. When all responses have been received or the program times out (with a *count* specified), or if the program is terminated with a SIGINT, **ping** displays a brief summary.

This program is intended for use in network testing, measurement and management. It should be used primarily for manual fault isolation. Because of the load it could impose on the network, it is unwise to use **ping** during normal operations or from automated scripts.

OPTIONS

Other options are:

–d Display debugging information.

–r Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly attached network, an error is returned. This option can be used to ping a local host through an interface that has no route through it (e.g., after the interface was dropped by **routed**(8C).

–v Verbose output. List ICMP packets other than ECHO RESPONSE that are received.

SEE ALSO

ifconfig(8C), netstat(1);
Configuring and Managing TCP/IP.

NAME

rexecd – remote execution server

SYNOPSIS

/etc/rexecd

DESCRIPTION

rexecd is the server for the **rexec**(3X) routine. The server provides remote execution facilities with authentication based on user names and passwords.

The **rexecd** server listens for service requests at the port indicated in the ''exec'' service specification; see **services**(5). When a service request is received the following protocol is initiated:

1) The server reads characters from the socket up to a null ('\0') byte. The resultant string is interpreted as an ASCII number, base 10.

2) If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the **stderr**. A second connection is then created to the specified port on the client's machine.

3) A null terminated user name of at most 16 characters is retrieved on the initial socket.

4) A null terminated, unencrypted password of at most 16 characters is retrieved on the initial socket.

5) A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.

6) **rexecd** then validates the user as is done at login time and, if the authentication was successful, changes to the user's home directory, and establishes the user and group protections of the user. If any of these steps fail the connection is aborted with a diagnostic message returned.

7) A null byte is returned on the initial socket and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by **rexecd**.

DIAGNOSTICS

Except for the last one listed below, all diagnostic messages are returned on the initial socket, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 7 above upon successful completion of all the steps prior to the command execution).

```
username too long
```

The name is longer than 16 characters.

```
password too long
```

The password is longer than 16 characters.

```
command too long
```

The command line passed exceeds the size of the argument last (as configured into the system).

```
Login incorrect
```

No password file entry for the user name existed.

```
Password incorrect
```

The wrong was password supplied.

```
No remote directory
```

The **chdir** command to the home directory failed.

```
Try again
```

A **fork** by the server failed.

```
<shellname>: ...
```

The user's login shell could not be started. This message is returned on the connection associated with the **stderr**, and is not preceded by a flag byte.

CAUTIONS

Indicating ''Login incorrect'' as opposed to ''Password incorrect'' is a security breach which allows people to probe a system for users with null passwords.

A facility to allow all data and password exchanges to be encrypted should be present.

SEE ALSO

rexec (3X)

NAME

rlogind – remote login server

SYNOPSIS

/etc/rlogind [–d]

DESCRIPTION

rlogind is the server for the **rlogin**(1C) program. The server provides a remote login facility with authentication based on privileged port numbers from trusted hosts.

rlogind listens for service requests at the port indicated in the "login" service specification; see **services**(5). When a service request is received the following protocol is initiated:

1) The server checks the client's source port. If the port is not in the range 0-1023, the server aborts the connection.

2) The server checks the client's source address and requests the corresponding host name (see **gethostbyaddr**(3N), **hosts**(5) and **named**(8)). If the hostname cannot be determined, the dot-notation representation of the host address is used.

Once the source port and address have been checked, **rlogind** allocates a pseudoterminal (see **pty**(4)), and manipulates file descriptors so that the slave half of the pseudoterminal becomes the **stdin**, **stdout**, and **stderr** for a login process. The login process is an instance of the **login**(1) program, invoked with the **–r** option. The login process then proceeds with the authentication process as described in **rshd**(8C), but if automatic authentication fails, it reprompts the user to log in as one finds on a standard terminal line.

The parent of the login process manipulates the master side of the pseudoterminal, operating as an intermediary between the login process and the client instance of the **rlogin** program. In normal operation, the packet protocol described in **pty**(4) is invoked to provide CTRL/S and CTRL/Q type facilities and propagate interrupt signals to the remote programs. The login process propagates the client terminal's baud rate and terminal type, as found in the environment variable, "TERM"; see **environ**(7). The screen or window size of the terminal is requested from the client, and window size changes from the client are propagated to the pseudoterminal.

DIAGNOSTICS

All diagnostic messages are returned on the connection associated with the **stderr**, after which any network connections are closed. An error is indicated by a leading byte with a value of 1.

```
Try again.
```

A **fork** by the server failed.

```
/bin/sh: ...
```

The user's login shell could not be started.

CAUTIONS

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an ''open'' environment.

A facility to allow all data exchanges to be encrypted should be present.

A more extensible protocol should be used.

NAME

route – manually manipulate the routing tables

SYNOPSIS

/etc/**route** [–f] [–n] [*command args*]

DESCRIPTION

route is a program used to manually manipulate the network routing tables. It normally is not needed, as the system routing table management daemon, **routed**(8C), should tend to this task.

route accepts three commands: **add**, to add a route; **addp**, to add a priority route; and **delete**, to delete a route.

The **addp** command adds a priority route. The TCP/IP server process will use priority routes before default routes or routes established by **routed**(8C). A route added with **addp** will appear first in the gateway table (displayed with the BSD command **netstat –r**(1)). You can only add priority routes with **addp**. Routes added manually by **route** cannot be deleted by **routed**(8C).

COMMAND SYNTAX

All commands have the following syntax:

/etc/**route** *command* [**net** | **host**] *destination gateway* [*metric*]

where *destination* is the destination host or network, *gateway* is the next-hop gateway to which packets should be addressed, and *metric* is a count indicating the number of hops to the *destination*. The metric is required for **add** and **addp** commands; it must be zero if the destination is on a directly-attached network, and nonzero if the route utilizes one or more gateways. If adding a route with metric 0, the gateway given is the address of this host on the common network, indicating the interface to be used for transmission.

Routes to a particular host are distinguished from those to a network by interpreting the Internet address associated with *destination*. The optional keywords **net** and **host** force the destination to be interpreted as a network or a host, respectively. If the *destination* has a ''local address part'' of INADDR_ANY, or if the *destination* is the symbolic name of a network, then the route is assumed to be to a network; otherwise, it is presumed to be a route to a host.

If the route is to a destination connected through a gateway, the *metric* should be greater than 0. All symbolic names specified for a *destination* or *gateway* are looked up first as a host name using **gethostbyname**(3N). If this lookup fails, **getnetbyname**(3N) is then used to interpret the name as that of a network.

You can add a default route as follows:

/etc/**route** *add default gateway_name* [*non-zero metric*]

TCP/IP software will use the default route when other routes occurring earlier in the routing table have failed, or when there are no other possible routes.

route uses a raw socket and the SIOCADDRT and SIOCDELRT **ioctl**'s(2) to do its work. As such, only the super-user may modify the routing tables.

OPTIONS

–f ''Flush'' the routing tables of all gateway entries. Using this option in conjunction with one of the commands described above flushes the tables prior to the command's application.

–n Suppress printing symbolic host and network names when reporting actions.

DIAGNOSTICS

```
add [ host | network ] %s: gateway %s flags %x
```

The specified route is being added to the tables. The values printed are from the routing table entry supplied in the **ioctl**(2) call. If the gateway address used was not the primary address of the gateway (the first one returned by **gethostbyname**(3N)), the gateway address is printed numerically as well as symbolically.

```
delete [ host | network ] %s: gateway %s flags %x
```

As above, but when deleting an entry.

```
%s %s done
```

When the –f flag is specified, each routing table entry deleted is indicated with a message of this form.

```
Network is unreachable
```

An attempt to add a route failed because the gateway listed was not on a directly-connected network. The next-hop gateway must be given.

```
not in table
```

A delete operation was attempted for an entry that wasn't present in the tables.

```
routing table overflow
```

An add operation was attempted, but the system was low on resources and was unable to allocate memory to create the new entry.

SEE ALSO

intro(4N), routed(8C);
Configuring and Managing TCP/IP.

NAME

routed – network routing daemon

SYNOPSIS

/etc/routed [**–g**] [**–s**] [**–q**] [**–t**] [**–n**] [**–f**] [**–h**] [*logfile*]

DESCRIPTION

The **routed** daemon is invoked at boot time to manage the network routing tables. The routing daemon uses a variant of the Xerox NS Routing Information Protocol in maintaining up to date kernel routing table entries. It uses a generalized protocol capable of use with multiple address types, but is currently used only for Internet routing within a cluster of networks.

In normal operation **routed** listens on the **udp**(4P) socket for the **route** service (see **services**(5)) for routing information packets. If the host is an internetwork router, it periodically supplies copies of its routing tables to any directly connected hosts and networks.

When **routed** is started, it uses the SIOCGIFCONF **ioctl**(2) to find those directly connected interfaces configured into the system and marked ''up'' (the software loopback interface is ignored). If multiple interfaces are present, it is assumed that the host will forward packets between networks. **routed** then transmits a *request* packet on each interface (using a broadcast packet if the interface supports it) and enters a loop, listening for *request* and *response* packets from other hosts.

When a *request* packet is received, **routed** formulates a reply based on the information maintained in its internal tables. The *response* packet generated contains a list of known routes, each marked with a ''hop count'' metric (a count of 16, or greater, is considered ''infinite''). The metric associated with each route returned provides a metric *relative to the sender*.

Response packets received by **routed** are used to update the routing tables if one of the following conditions is satisfied:

1) No routing table entry exists for the destination network or host, and the metric indicates the destination is ''reachable'' (i.e. the ''hop count'' is not infinite).

2) The source host of the packet is the same as the router in the existing routing table entry. That is, updated information is being received from the very internetwork router through which packets for the destination are being routed.

3) The existing entry in the routing table has not been updated for some time (defined to be 90 seconds) and the route is at least as cost effective as the current route.

4) The new route describes a shorter route to the destination than the one currently stored in the routing tables; to decide this, the metric of the new route is compared against the one stored in the table.

When an update is applied, **routed** records the change in its internal tables. The change is reflected in the next *response* packet sent.

In addition to processing incoming packets, **routed** also checks the routing table entries periodically. If an entry has not been updated for 3 minutes, the entry's metric is set to infinity and marked for deletion. Deletions are delayed an additional 60 seconds to ensure that the invalidation is propagated throughout the local internet.

Hosts acting as internetwork routers gratuitously supply their routing tables every 30 seconds to all directly connected hosts and networks. The response is sent to the broadcast address on nets capable of that function, to the destination address on point-to-point links, and to the router's own address on other networks. The normal routing tables are bypassed when sending gratuitous responses. The reception of responses on each network is used to determine that the network and interface are functioning correctly. If no response is received on an interface, another route may be chosen to route around the interface, or the route may be dropped if no alternative is available.

OPTIONS

routed supports several options:

–g This flag is used on internetwork routers to offer a route to the ''default'' destination. This option is typically used on a gateway to the Internet, or on a gateway that uses another routing protocol whose routes are not reported to other local routers.

–s Forces **routed** to supply routing information whether it is acting as an internetwork router or not. This is the default if multiple network interfaces are present, or if a point-to-point link is in use.

–q This option is the opposite of the –s option.

–t If the **–t** option is specified, all packets sent or received are printed on the standard output. In addition, **routed** will not divorce itself from the controlling terminal, so that interrupts from the keyboard will kill the process.

–d Not supported by Domain/OS BSD.

Domain/OS BSD EXTENSIONS

–n Directs **routed** not to install changes into the local routing table. However, the **routed** process continues to receive broadcasts from other **routed** processes. The **–n** option is used for debugging purposes.

–f Directs **routed** at startup to ''flush'' (purge) all routes from the local routing table, except routes added manually with /etc/**route**(8C).

–h Exit, if not supplier, when routing table is stable. Use this switch on hosts only, not on gateways.

Any other argument supplied is interpreted as the name of the file in which **routed**'s actions should be logged. This log contains information about any changes to the routing tables and, if not tracing all packets, a history of recent messages sent and received which are related to the changed route.

In addition to the facilities described above, **routed** supports the notion of distant passive and active gateways. When **routed** is started up, it reads the file /etc/**gateways** to find gateways that may not be located using only information from the SIOGIFCONF **ioctl**(2). Gateways specified in this manner should be marked passive if they are not expected to exchange routing information, while gateways marked active should be willing to exchange routing information (that is, they should have a **routed** process running on the machine).

Passive gateways are maintained in the routing tables forever, and information regarding their existence is included in any routing information transmitted. Active gateways are treated equally to network interfaces. Routing information is distributed to the gateway and if no routing information is received for a period of the time, the associated route is deleted. External gateways are also passive, but they are not placed in the routing table nor are they included in routing updates The function of external entries is to inform **routed** that another routing process will install such a route, and that alternate routes to that destination should not be installed. Such entries are only required when both routers may learn of routes to the same destination.

The /etc/**gateways** file is comprised of a series of lines, each in the following format:

< **net** | **host** > *name1 gateway name2 metric value* < **passive** | **active** | **external** >

The **net** or **host** keyword indicates if the route is to a network or specific host.

Name1 The name of the destination network or host. This may be a symbolic name located in /etc/**networks** or /etc/**hosts** (or, if started after **named**(8), known to the name server), or an Internet address specified in ''dot'' notation; see **inet**(3n).

Name2 The name or address of the gateway to which messages should be forwarded.

Value A metric indicating the hop count to the destination host or network.

One of the keywords **passive**, **active** or **external** indicates if the gateway should be treated as passive or active (as described above), or whether the gateway is external to the scope of the **routed** protocol.

Internetwork routers that are directly attached to the ARPANET or Milnet should use the Exterior Gateway Protocol (EGP) to gather routing information rather then using a static routing table of passive gateways. EGP is required in order to provide routes for local networks to the rest of the Internet system. Sites needing assistance with such configurations should contact the Computer Systems Research Group at Berkeley.

For a node to run **routed**, it must be correctly configured to run BSD TCP/IP. See *Configuring and Managing TCP/IP* for more information about **routed**.

NOTES

The **routed** daemon is normally started on a node at boot time, from the **/etc/rc.local** file. We recommend that you run **routed** on each gateway to dynamically update the gateway's routing tables. You can also run **routed** on hosts so they receive the latest routing information.

FILES

/etc/gateways	for distant gateways

BUGS

The kernel's routing tables may not correspond to those of **routed** when redirects change or add routes. The only remedy for this is to place the routing process in the kernel.

routed does not incorporate other routing protocols, such as Xerox NS and EGP. Using separate processes for each requires configuration options to avoid redundant or competing routes.

routed does not currently listen to intelligent interfaces, such as an IMP, and to error protocols, such as ICMP, to gather more information. It does not always detect unidirectional failures in network interfaces (e.g., when the output side fails).

SEE ALSO

udp(4P), htable(8), route(8C), rc(8);
Configuring and Managing TCP/IP;
''Internet Transport Protocols'', XSIS 028112, Xerox System Integration Standard.

NAME

rshd – remote shell server

SYNOPSIS

/etc/rshd

DESCRIPTION

rshd is the server for the **rcmd**(3X) routine and, consequently, for the **rsh**(1C) program. The server provides remote execution facilities with authentication based on privileged port numbers from trusted hosts.

rshd listens for service requests at the port indicated in the "cmd" service specification; see **services**(5). When a service request is received the following protocol is initiated:

1) The server checks the client's source port. If the port is not in the range 0-1023, the server aborts the connection.

2) The server reads characters from the socket up to a null ('\0') byte. The resultant string is interpreted as an ASCII number, base 10.

3) If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the **stderr**. A second connection is then created to the specified port on the client's machine. The source port of this second connection is also in the range 0-1023.

4) The server checks the client's source address and requests the corresponding host name (see **gethostbyaddr**(3N), **hosts**(5) and **named**(8)). If the hostname cannot be determined, the dot-notation representation of the host address is used.

5) A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as the user identity on the client 's machine.

6) A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as a user identity to use on the server 's machine.

7) A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.

8) **rshd** then validates the user according to the following steps. The local (server-end) user name is looked up in the password file and a **chdir** is performed to the user's home directory. If either the lookup or **chdir** fail, the connection is terminated. If the user is not the super-user, (user id 0), the file **/etc/hosts.equiv** is consulted for a list of hosts considered "equivalent". If the client's host name is present in this file, the authentication is considered successful. If the lookup fails, or the user is the super-user, then the file **.rhosts** in the home directory of the remote user is checked for the machine name and

identity of the user on the client's machine. If this lookup fails, the connection is terminated.

9) A null byte is returned on the initial socket and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by **rshd**.

DIAGNOSTICS

Except for the last one listed below, all diagnostic messages are returned on the initial socket, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 9 above upon successful completion of all the steps prior to the execution of the login shell).

`locuser too long`

The name of the user on the client's machine is longer than 16 characters.

`remuser too long`

The name of the user on the remote machine is longer than 16 characters.

`command too long`

The command line passed exceeds the size of the argument list (as configured into the system).

`Login incorrect.`

No password file entry for the user name existed.

`No remote directory.`

The **chdir** command to the home directory failed.

`Permission denied.`

The authentication procedure described above failed.

`Can't make pipe.`

The pipe needed for the **stderr**, wasn't created.

`Try again.`

A **fork** by the server failed.

`<shellname>: ...`

The user's login shell could not be started. This message is returned on the connection associated with the **stderr**, and is not preceded by a flag byte.

CAUTIONS

The authentication procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an ''open'' environment.

SEE ALSO

rsh(1C), rcmd(3X);
Configuring and Managing TCP/IP.

NAME

rwhod – system status server

SYNOPSIS

/etc/**rwhod** [–w]

DESCRIPTION

The **rwhod** server maintains the database used by the **rwho**(1C) and **ruptime**(1C) programs. Its operation is predicated on the ability to broadcast messages on a network.

On Apollo networks, **rwhod** has been changed to take advantage of the Domain distributed file system and to reduce contention for the **rwho** directory. Since on Apollo networks, there is no need for every host on the network to maintain its own /**usr**/**spool**/**rwho** directory, a change has been made to allow you to specify one host per network who will write the information to a master /**usr**/**spool**/**rwho** directory. All the other hosts link to that master directory to access the information but do not write to the directory. Specifically, **rwhod** writes to /usr/spool/rwho only if that directory is local to the node, or if the optional –w switch is supplied.

rwhod both produces and consumes system status information. It periodically queries the state of the system and constructs status messages which are broadcast on a network, and it listens for other **rwhod** servers' status messages as well. When it receives a status message from another server, **rwhod** validates it and records it in a file located in the directory /**usr**/**spool**/**rwho**, if the directory is physically located on the host or if **rwhod** was started with the –w switch. On hosts where **rwhod** was started without the –w switch and where /**usr**/**spool**/**rwho** is a link to a remote host, **rwhod** does not write to the directory.

The **rwho** server transmits and receives messages at the port indicated in the "rwho" service specification. The messages sent and received, are of the form:

```
struct  outmp {
        char    out_line[8];/* tty name */
        char    out_name[8];/* user id */
        long    out_time;/* time on */
};

struct  whod {
        char    wd_vers;
        char    wd_type;
        char    wd_fill[2];
        int     wd_sendtime;
        int     wd_recvtime;
        char    wd_hostname[32];
        int     wd_loadav[3];
        int     wd_boottime;
        struct  whoent {
                struct  outmp we_utmp;
                int      we_idle;
        } wd_we[1024 / sizeof (struct whoent)];
};
```

All fields are converted to network byte order before transmission. The load averages represent averages over the 5, 10, and 15 minute intervals prior to a server's transmission. The host name included is that returned by the **gethostname**(2) system call. The array at the end of the message contains information about the users logged in to the sending machine. This information includes an entry for each non-idle terminal line and a value indicating the time since a character was last received on the terminal line.

Messages received by the **rwho** server are discarded unless they originated at an **rwho** server's port. In addition, if the host's name, as specified in the message, contains any unprintable ASCII characters, the message is discarded. Valid messages received by **rwhod** are placed in files named **whod.hostname** in the directory **/usr/spool/rwho**. These files contain only the most recent message, in the format described above.

OPTIONS

–w   Update the files in the **/usr/spool/rwho** directory, even if the directory is not local to the node.

NOTES

Note that **rwhod** will update the files in the **/usr/spool/rwho** directory in two cases:

1. If the **/usr/spool/rwho** directory is physically located on that host.

2. If **rwhod** was started with the –w switch.

To reduce contention for the rwho directory and provide updated information to hosts, we recommend that you

- Physically locate the **/usr/spool/rwho** directory on the network's TCP/IP administrative node. All other hosts on the network should be set up so that **/usr/spool/rwho** is a link to the administrative node.

- If you are running **rwhod** in a Domain internet, you can configure your network in one of two ways:

  1. Configure one node on each subnet to have a local **/usr/spool/rwho** and link all other nodes on that subnet to that node.

  2. Configure all nodes in the internet to link to one master TCP/IP administrative node containing the **/usr/spool/rwho** directory. This enables the **rwho** and **ruptime** utilities to see all nodes in the internet.

> In order to make this arrangement work, you must run **rwhod –w** on one node on each subnet. They will then write their subnet's information to the directory on the internet's master TCP/IP administrative node.

You also can run **rwhod** on a gateway to see broadcasts from two networks.

rwhod, like other BSD daemons, is invoked at boot time by the **/etc/rc.local** startup file. See *Configuring and Managing TCP/IP* for more information about **rwhod**.

NOTES

People often interpret the server's dying as a machine in the network going down.

SEE ALSO

rwho(1C), ruptime(1C), rc(8)

NAME

tcpd – TCP/IP protocol server

SYNOPSIS

/etc/**tcpd** [**–a**] [**–d**<*mask*>] [**–p**<*time*>] [**–t**<*ipttl*>] [**–w**<*window*>]

DESCRIPTION

Invoking **tcpd**, the TCP/IP protocol server or daemon, enables a node's TCP/IP socket-call interface and initializes several internal tables required for operation of the protocols. The **tcpd** daemon must be run on every node that uses TCP/IP. Normally, **tcpd** is invoked by the node's startup file, /etc/**rc.local**.

Options available with this command allow you to define certain parameters of the protocols.

OPTIONS

–a Suppress delayed packet acknowledgements (ACKs), the default condition. Delayed ACKs is a performance optimization feature for TCP which allows a receiver to delay until 33% of the maximum window size offered can be uncovered (but in no case to wait longer than 0.25 seconds) before it acknowledges received packets.

–p<*time*> Set gateway ping interval in seconds. The value must be expressed in hexadecimal and can range from 0 to A. Default value is 4 seconds. Setting the interval to 0 inhibits pinging. TCP issues an ICMP Echo Request (ping) to local hosts and gateways at the specified interval to verify their continued operation. Any gateway that fails to respond after three tries is moved to the end of the node's routing table.

–t<*ipttl*> Set the IP parameter, packet maximum time to live. The value must be expressed in hexadecimal and can range from 1 to FF. Default value is 1E.

–w<*window*> Set the receive window size (a TCP flow control parameter) in octets. The value must be expressed in HEX and can range from 1 to 239C. Default value is 239C.

–d<*mask*> Display debugging information as defined by the 16-bit mask. See the table below for a description of the debug information that can be requested. Add bit values to request several types of information.

Bit Value	Debug Information
0001 (default)	General information
0002	IP level information
0004	ARP information
0008	TCP information
0010	Data in TCP packets
0020	UDP information
0200	Broadcasts
1000	TCP finite state machine information
2000	Device level information
4000	Additional detail at any level
foff	All available debug information except broadcasts

SEE ALSO

netstat(1), ping(8);
Configuring and Managing TCP/IP.

NAME

telnetd – DARPA TELNET protocol server

SYNOPSIS

/etc/telnetd

DESCRIPTION

telnetd is a server which supports the DARPA standard **TELNET** virtual terminal protocol. **telnetd** is invoked by the internet server (see **inetd**(8)), normally for requests to connect to the **TELNET** port as indicated by the **/etc/services** file (see **services**(5)).

telnetd operates by allocating a pseudoterminal device (see **pty**(4)) for a client, then creating a login process which has the slave side of the pseudoterminal as **stdin**, **stdout**, and **stderr**. **telnetd** manipulates the master side of the pseudoterminal, implementing the **TELNET** protocol and passing characters between the remote client and the login process.

When a **TELNET** session is started up, **telnetd** sends **TELNET** options to the client side indicating a willingness to do *remote echo* of characters, to *suppress go ahead*, and to receive *terminal type information* from the remote client. If the remote client is willing, the remote terminal type is propagated in the environment of the created login process. The pseudoterminal allocated to the client is configured to operate in "cooked" mode, and with XTABS and CRMOD enabled (see **tty**(4)).

telnetd is willing to *do*: *echo*, *binary*, *suppress go ahead*, and *timing mark*. **telnetd** is willing to have the remote client *do*: *binary*, *terminal type*, and *suppress go ahead*.

BUGS

Some **TELNET** commands are only partially implemented.

The **TELNET** protocol allows for the exchange of the number of lines and columns on the user's terminal, but **telnetd** doesn't make use of them.

Because of bugs in the original 4.2BSD **telnet**, **telnetd** performs some dubious protocol exchanges to try to discover if the remote client is, in fact, a 4.2BSD **telnet**.

Binary mode has no common interpretation except between similar operating systems (the UNIX system in this case).

The terminal type name received from the remote client is converted to lowercase.

The *packet* interface to the pseudoterminal (see **pty**(4)) should be used for more intelligent flushing of input and output queues.

telnetd never sends **TELNET** *go ahead* commands.

SEE ALSO

telnet(1C)

NAME

tftpd – tftp daemon

SYNOPSIS

/etc/tftpd

DESCRIPTION

tftpd is a daemon which runs the trivial file transfer protocol server for the BSD Internet software. It is called by **inetd**(8C) when an incoming datagram requests the **tftp**(1C) service. It then handles **tftp**(1C) file transfers in accordance with RFC783.

Note that **/etc/tftpd** must be setuid to a designated **tftp**(1C) user (usually a very non-privileged userid and never root) and that the **tftp**(1C) spool directory must be owned by that user.

NOTES

The Domain/OS BSD versions of **tftp**(1C) and **tftpd** are adaptations of the Massachusetts Institute of Technology implementations of the **tftp** protocol. Domain/OS BSD **tftp** will interface with any RFC783 compliant implementation. Note, however, that the 4.3BSD distribution version of **tftp** does not meet these restrictions.

WARNINGS

tftp(1C) and **tftpd** comprise an implementation of the Trivial File Transfer Protocol described in RFC783. They allow you to quickly copy files among hosts on an internet without regard to ownership or access restrictions. Therefore, the desired security of a system should be considered before allowing **tftp**(1C) transactions. In an inspired attempt to prevent accidental destruction of important files, **tftp**(1C) requires that remote file names be absolute pathnames (that is, beginning with /) containing the string "**/tftp/**", but not containing the string "/../".

SEE ALSO

tftp(1C), inetd(8C);
Configuring and Managing TCP/IP.

NAME

trpt – transliterate protocol trace

SYNOPSIS

trpt [**–a**] [**–c**] [**–d** *<PCB addr>*] [**–e**] [**–f**] [**–j**] [**–l**]
[**–s**] [**–t**] [**–w**] [**–p** *<PCB addr>*] [*<filename>*]

DESCRIPTION

trpt interrogates the buffer of TCP trace records created when a socket is marked for "debugging" (see **setsockopt**(2)), and prints a readable description of these records. When no options are supplied, **trpt** prints all the trace records found in the system, grouped according to TCP connection protocol control block (PCB). The following options may be used to alter this behavior.

OPTIONS

–a Print the values of the source and destination addresses for each packet recorded, in addition to the normal output.

–f Follow the trace as it occurs, waiting a short time for additional records each time the end of the log is reached.

–j Just give a list of the protocol control block addresses for which there are trace records.

–p *<PCB addr>* Show only trace records associated with the protocol control block, the address of which follows.

–s Print a detailed description of the packet sequencing information, in addition to the normal output. (Currently unimplemented)

–t Print the values for all timers at each point in the trace, in addition to the normal output. (Currently unimplemented)

Domain/OS BSD EXTENSIONS

–c Clear trace buffer.

–d *<PCB addr>* Toggle debug on a connection.

–e Exit on a bad trace record.

–l Print lapsed times, in addition to the normal output.

–w Warn on bad trace records.

NOTES

The recommended use of **trpt** is as follows. Isolate the problem and enable debugging on the socket(s) involved in the connection. Find the address of the protocol control blocks associated with the sockets using the –A option to **netstat**(1). Then run **trpt** with the **–p** option, supplying the associated protocol control block addresses. The **–f** option can be used to follow the trace log once the trace is located. If there are many sockets using the debugging option, the –j option may be useful in checking to see if any trace records are present for the socket in question.

If debugging is being performed on a file other than the default, which is **'node_data/systmp/tcp_data**, <*filename*> may be used to specify another file.

DIAGNOSTICS

```
no namelist
```

Printed when the system image doesn't contain the proper symbols to find the trace buffer.

Other diagnostics should be self explanatory.

CAUTIONS

trpt should print the data for each input or output, but this is not saved in the race record.

The output format is inscrutable.

FILES

'node_data/systmp/tcp_data

SEE ALSO

setsockopt(2), netstat(1);
Configuring and Managing TCP/IP.

Appendix E

TCP/IP File Formats

This appendix describes the formats of the TCP/IP files mentioned in this book. It is intended for users who operate in the Aegis environment. Reference material about these files for the BSD and SysV environments is available on–line and in the *Programmer's Reference* manuals for those environments.

The file descriptions in this appendix may contain references to commands in the form of "foo(nx)," where n is a number from 1 to 8 and x is a letter. More information about these commands can be found in the *BSD Command Reference* (n = 1, 6, or 7), *BSD Programmer's Reference* (n = 2, 3, 4, or 5), or *Managing BSD System Software* (n = 8).

NAME

hosts – host name data base

DESCRIPTION

The **hosts** file contains information regarding the known DARPA Internet hosts with which your Domain node can communicate (usually via TCP/IP). For each host, a single line should be present with the following information:

Internet address
official host name
aliases

Items are separated by any number of blanks and/or tab characters. A "#" indicates the beginning of a comment: characters up to the end of the line are not interpreted by routines which search the file.

When using the name server **named**(8), this file provides a backup when the name server is not running. For the name server, it is suggested that only a few addresses be included in this file. These include address for the local interfaces that **ifconfig**(8C) needs at boot time and a few machines on the local network.

This file may be created from the official host data base maintained at the Network Information Control Center (NIC), though local changes may be required to bring it up to date regarding unofficial aliases and/or unknown hosts. As the data base maintained at NIC is incomplete, use of the name server is recommend for sites on the DARPA Internet.

Network addresses are specified in the conventional "." notation using the **inet_addr**() routine from the Internet address manipulation library, **inet**(3N). Host names may contain any printable character other than a field delimiter, newline, or comment character.

FILES

/etc/hosts

SEE ALSO

gethostbyname(3N), ifconfig(8C), named(8)

NAME

inetd.conf – configuration file for **inetd**(8C)

DESCRIPTION

/etc/**inetd.conf** is a link to `/**node_data/etc/inetd.conf**. The Internet superdaemon, **inetd**, reads /etc/**inetd.conf** at boot time and, in some cases, after it gets a hangup signal.

The **inetd.conf** file is ''free format.'' All fields must be present in each entry, and must appear in the order shown below.

service name	Must be must present in /etc/**services.**
socket type	Must be one of stream, dgram, raw, rdm, or seqpacket.
protocol	Must be listed in /etc/**protocols.**
wait/nowait	Use *wait* for single-threaded servers (ones that simply take over the socket from **inetd**). Use *nowait* for multi-threaded servers (ones which connect directly to the peer, freeing up the socket for continued use by **inetd**.)
server program	The full pathname to this program (e.g., /etc/**ftpd**).
server program arguments	A maximum of MAXARGS (normally 5).

Continuation lines, if required, must begin with a space or tab. To allow comments, **inetd** ignores any line that has a pound sign (#) in column 1.

FILES

/etc/**services**	List of Internet services
/etc/**protocols**	List of Internet protocols
/etc/**inetd**	Internet superdaemon; reads **inetd.conf** for configuration data.
/etc/**ftpd**	FTP daemon
/etc/**rexecd**	Remote execution server
/etc/**rlogind**	Remote log-in daemon
/etc/**rshd**	Remote Shell server
/etc/**telnetd**	DARPA TELNET protocol server

SEE ALSO

inetd(8C), services(5), rexecd(8C), rlogind(8C), rshd(8C), telnetd(8C).

NAME

networks – network name data base

DESCRIPTION

The **/etc/networks** file contains information regarding the known networks which comprise the DARPA Internet. For each network a single line should be present with the following information:

official network name
network number
aliases

Items are separated by any number of blanks and/or tab characters. A "#" indicates the beginning of a comment: characters up to the end of the line are not interpreted by routines which search the file. This file is normally created from the official network data base maintained at the Network Information Control Center (NIC), though local changes may be required to bring it up to date regarding unofficial aliases and/or unknown networks.

Network numbers may be specified in the conventional "." notation using the **inet_network**() routine from the Internet address manipulation library, **inet**(3N). Network names may contain any printable character other than a field delimiter, newline, or comment character.

FILES

/etc/**networks**

SEE ALSO

getnetent(3N)

NOTE

A name server should be used instead of a static file.

NAME

protocols – protocol name data base

DESCRIPTION

The **protocols** file contains information regarding the known protocols used in the DARPA Internet. For each protocol a single line should be present with the following information:

official protocol name
protocol number
aliases

Items are separated by any number of blanks and/or tab characters. A ''#'' indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file.

Protocol names may contain any printable character other than a field delimiter, newline, or comment character.

FILES

/etc/**protocols**

SEE ALSO

getprotoent(3N)

CAUTIONS

A name server should be used instead of a static file.

NAME

resolver: resolv.conf – resolver configuration file

SYNOPSIS

/etc/resolv.conf

DESCRIPTION

The resolver configuration file contains information that is read by the resolver routines the first time they are invoked by a process. The file is designed to be human readable and contains a list of name-value pairs that provide various types of resolver information.

On a normally configured system this file should not be necessary. The only name server to be queried will be on the local machine and the domain name is retrieved from the system.

The different configuration options are:

nameserver followed by the Internet address (in dot notation) of a name server that the resolver should query. At least one name server should be listed. Up to MAXNS (currently 3) name servers may be listed, in that case the resolver library queries them in the order listed. If no **nameserver** entries are present, the default is to use the name server on the local machine. (The algorithm used is to try a name server, and if the query times out, try the next, until out of name servers, then repeat trying all the name servers until a maximum number of retries are made).

domain followed by a domain name, that is the default domain to append to names that do not have a dot in them. If no **domain** entries are present, the domain returned by **gethostname**(2) is used (everything after the first period (.)). Finally, if the host name does not contain a domain part, the root domain is assumed.

The name value pair must appear on a single line, and the keyword (e.g. **nameserver**) must start the line. The value follows the keyword, separated by white space.

FILES

/etc/resolv.conf

SEE ALSO

gethostbyname(3N), resolver(3), named(8)

NAME

services – service name data base

DESCRIPTION

The **services** file contains information regarding the known services available in the DARPA Internet. For each service a single line should be present with the following information:

official service name
port number
protocol name
aliases

Items are separated by any number of blanks and/or tab characters. The port number and protocol name are considered a single *item*; a "/" is used to separate the port and protocol (e.g. "512/tcp"). A "#" indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file.

Service names may contain any printable character other than a field delimiter, newline, or comment character.

FILES

/etc/services

SEE ALSO

getservent(3N)

CAUTIONS

A name server should be used instead of a static file.

Glossary

Address

A series of numbers that uniquely identifies a node on a network. Each network and protocol family has its own address format.

Address mapping table

A table that TCP/IP uses to convert **Internet addresses** to and from local addresses.

Address Resolution Protocol

(ARP) The protocol used by **TCP/IP** hosts and gateways to maintain their **address mapping tables.** ARP updates the table whenever a host cannot find the local address that corresponds to the Internet address of a host or gateway on its local network.

Administrative files

See **TCP/IP administrative files.**

/etc/hosts, /etc/gateways, /etc/networks, /etc/hosts.equiv.

Administrative node

See **TCP/IP administrative host.**

Alias

A secondary name for a TCP/IP host. You can use a TCP/IP host alias as if it were the host name. Aliases are listed in the **/etc/hosts** file after the host name.

Apollo Token Ring Network

A **local area network** that uses coaxial cable as its transmission medium and operates at 12 megabits per second.

ARP

See Address Resolution Protocol.

ARPANET

A nationwide network originally developed by the Department of Defense and Bolt, Baranek, and Newman (BBN). The ARPANET is one of the largest networks that uses TCP/IP protocols.

Authoritative name server

For networks using the DARPA Internet Name Resolution Protocol, a master name server that maintains all the name data for a particular **zone of authority**. Authoritative name servers either resolve queries or redirect them to another authoritative server.

Bridge

A device that physically links two or more networks using the same communications protocols. For example, a bridge can connect two Domain networks. In Domain internets, network controllers and their associated software perform bridge functions. See also, **router** and **routing node**.

Caching Only Server

A name server that queries the local **master name server** for name information and maintains a cache of naming information until the data expires. Caching only servers are not authoritative for any **zone of authority**.

Configuration

The arrangement of a computer system or network as defined by the nature, number, and chief characteristics of its functional units. More specifically, the term configuration can refer to a hardware configuration or a software configuration.

Daemon

A UNIX server process.

DARPA

The Defense Advanced Research Projects Agency of the U.S. Department of Defense.

DARPA Internet

A group of networks that can communicate using the TCP/IP protocols.

Destination address

The field in a packet that identifies the intended recipient of the packet.

Domain (Apollo) address

A Domain node ID is a 20–bit node address, expressed in hexadecimal numbers, for a single Domain network. In some contexts, a 32–bit Domain network number precedes the node ID. Domain network numbers are assigned by Apollo Computer Inc.

domain (UNIX)

1. For networks using the **DARPA** Internet Name Resolution Protocol, a group of resources that usually fall within the boundaries of one administrative unit and that share a common name server to which all unresolved queries are directed.

2. One of the official organizational divisions in the ARPANET, such as the *edu* domain for educational (university) networks and the *com* domain for commercial networks.

3. A member of one of the official domains. For example, the *edu* domain consists of smaller domains such as harvard.edu , mit.edu, stanford.edu, and other educational networks. The *com* domain consists of bbn.com, marble.com, and other company networks. Universities, commercial establishments, and members of other ARPANET domains choose their own domain names (subject to SRI–NIC approval).

domain name

All parts of a UNIX name server domain are given labels (for example, *edu, harvard, vax1*). The domain name of an object within a domain is the concatenation of all the labels from the domain's root to that object. For example, the domain name of a VAX computer at Harvard University would look like the following: vax1.harvard.edu.

domain name space

A tree–structured naming system used by networks using the DARPA Internet Name Resolution Protocol.

Domain Network

A network consisting of nodes that use Domain protocols. At present, Domain nodes can operate on an Apollo Token Ring network or on an ETHERNET IEEE 802.3 network.

ETHERNET

A **local area network** that uses coaxial cable as its communications medium and operates at 10 megabits per second.

ETHERNET address

A 48–bit number that identifies a device on an ETHERNET network. ETHERNET addresses are expressed as six hexadecimal octets. ETHERNET addresses are assigned by the Xerox Corporation.

File Transfer Protocol (FTP)

A protocol for transmitting files between host computers. FTP is defined by DARPA. FTP uses TCP and IP as underlying protocols.

FTP

See File Transfer Protocol.

ftpd

The UNIX **daemon** process that accepts incoming FTP requests.

Gateway

Usually, a device or set of devices that connects networks using differnet communication protocols by providing protocol translation. (In contrast, a bridge connects two networks that use the same protocols and so requires no protocol information). See also **TCP/IP Gateway**.

Hop

A packet's passage through a **routing node** on its way to its final destination. The number of hops in a route is the same as the number of gateways a packet passes through.

Host

A computer or workstation that communicates over a network. A host can both initiate communications and respond to communications that are addressed to it.

Host number

The portion of a TCP/IP **Internet address** that uniquely identifies the host on its local network.

hosts.txt file

A file containing information on the networks, gateways, and hosts that make up the **ARPANET** and other DARPA Internets that are supported by the **Network Information Center** (NIC).

inetd

The **daemon** process that listens for incoming requests for services listed in the **/etc/inetd.conf** file. When a request for a certain server listed in this file arrives, **inetd** forks the desired program.

Internet

1. Two or more connected networks that may or may not use the same communication protocol. The device that connects the networks may perform routing and/or gateway functions.

2. A TCP/IP Internet conforms to the Internet Protocol and the Transmission Control Protocol.

Internet address

1. An address that conforms to the **DARPA**-defined **Internet protocol**. A unique, four-byte number that identifies a host or gateway on the Internet, consisting of a **network number** followed by a **host number**. The host number can be further divided into a **subnet number**. Internet addresses are expressed as four decimal numbers, ranging between 0–255 and separated by periods.

2. An address that uniquely identifies a destination on an **internet**.

Internet gateway

See Gateway.

Internet Protocol (IP)

A protocol defined by the Defense Advanced Research Projects Agency (DARPA) for connecting networks through **gateways**.

IP

See Internet Protocol.

Label

The string associated with one component of a UNIX **domain**.

LAN

See Local Area Network.

Local address

An address that uniquely identifies a destination within a single network.

Local Area Network (LAN)

A communications network linking a number of devices that are located within a relatively short distance, typically less than a mile.

Local network

The network to which a node is directly attached.

Local node

The node executing the commands. For example, the processes created by the Domain **crp** command execute on the node specified in the **–on** option. For contrast, see **remote node**.

Master name server

An **authoritative name server** who has access to the most current information about hosts in the **zone of authority**.

named

The UNIX **daemon**, or server process, that responds to and resolves name queries.

Name server

A host running the name server daemon, **named**. Named resolves name queries about network objects from the local or remote hosts.

Network

Transmission media and software that links nodes and peripherals.

Network address

An Internet address created by appending zeros to the **network number**. For example, 205.3.1.0 is a **Type C** network address.

Network Information Center (NIC)

A centralized information resource managed by SRI International (Menlo Park, CA). The Network Information Center assigns DARPA Internet network numbers, and maintains the master **hosts.txt** table and copies of the DARPA Internet specifications.

Network number

The component of an **Internet address** on an internet that uniquely identifies the network. See **Internet address, Network address**.

NIC

See Network Information Center.

Node ID

The unique 20–bit identifier for a Domain node.

Node specification

An operating system identifier for a node. A node specification can consist of a node ID, the Domain address, Internet address or the node name.

Packet

A sequence of binary digits that is transmitted as a unit in a computer network. A packet usually contains control information plus data. Packets are transferred as a single unit over a **packet–switched network**.

Packet–switched network

A network that transmits data in the form of packets. Each **packet** is transmitted separately over the network; they are dynamically routed from source to destination. The DARPA Internet is a packet–switched network.

Physical layer

The lowest communications layer. It provides the mechanical, electrical, functional, and procedural means to provide communications over a physical medium.

Physical layer interface

The interface between **TCP/IP** and the **physical layer** software that sends messages over a particular transmission medium. Each physical layer interface identifier indicates a particular network to which the host is attached. For example, **dr0** specifies an Apollo Token Ring interface.

Primary master name server

An **authoritative name server** whose disk contains the name resolution data files for a particular zone. The files are maintained by the system administrator and contain the most current information about hosts in the **zone of authority**.

Prime gateway

A gateway that maintains up–to–date information about routes to destinations on the TCP/IP Internet. A prime gateway uses a protocol such as **RIP** to dynamically maintain its routing tables.

Protocol

A set of rules that governs the procedures used in exchanging information between two communication processes.

Remote network

A network not directly connected to a node. A node must send packets through a **router** or **gateway** to communicate on a network.

Remote node

A node other than the node executing commands.

Remote server

A UNIX name server running on a remote host that services requests from other hosts. This option is suitable for diskless hosts and hosts with limited memory or CPU cycles.

Request for Comment (RFC)

A specification or proposed specification that applies to the **DARPA Internet**. You can obtain copies of any RFC from the **Network Information Center**.

RFC

See Request for Comment.

RIP

See Routing Information Protocol.

Root

The top level object within a UNIX name server domain. The host that holds the database containing pointers to all master name servers in the domain.

Route

1. To determine the path by which a packet will reach its destination, when the packet is being transmitted through an internet.
2. The path a packet takes from its source to its destination.
3. A UNIX command.

routed

The **daemon** process that maintains the **routing table** on TCP/IP gateways.

Router

1. The software process that controls the transmission of packets between networks. A router manages data transfer across a bridge.
2. A node that runs routing software. See **Routing node**.

Routing Information Protocol (RIP)

A **protocol** used by **routed** to dynamically maintain the routing tables on gateways and, in some cases, hosts.

Routing node

A node that runs the routing process and transmits packets between different networks, especially through a bridge. A node that transmits packets between dissimilar networks is called a **gateway**.

Routing server

Same as **router**.

Routing table

A table maintained by hosts and gateways that indicates the next gateway in the route to a destination.

Secondary master name server

An **authoritative name server** that requests name resolution data files from the **primary master server**. During system boot, the secondary server attempts to establish a connection with the primary server and obtain a dump of the data files. Thereafter, the secondary server can answer requests of name resolution information. Secondary servers automatically update expired data from the files on the primary server.

Server

A process that is dedicated to managing a certain function. A variety of servers support TCP/IP communications. See also **daemon**.

Software loopback interface

A **physical layer interface** simulator within TCP/IP software that directs messages to be received within the node without transmitting them to the physical layer software.

Subdomain

An administrative division of a UNIX name server domain.

Subnet number

The portion of the **Internet address** that identifies networks within an internet. A **network number** identifies a single **internet** while subnet numbers identify networks within that internet.

TCP

See Transmission Control Protocol.

TCP/IP

Transmission Control Protocol/Internet Protocol. An acronym used to refer to the TCP and IP protocols and related Internet protocols, such as FTP and Telnet, defined by the Defense Advanced Projects Agency.

TCP/IP administrative files

The following files are the TCP/IP administrative files: **/etc/hosts**, **/etc/networks**, **/etc/gateways**, **/etc/hosts.equiv**, and **/etc/resolv.conf**. They reside on the TCP/IP Administrative Node for the network.

TCP/IP administrative host

A node on which the TCP/IP administrative files reside.

TCP/IP gateway

A gateway that routes information on a TCP/IP internet. A TCP/IP gateway usually translates protocols for unlike networks. However, TCP/IP gateways are required on Domain routing nodes to maintain TCP/IP services across the physical link between two Domain networks.

tcpd

The **server** process that manages TCP/IP communications on all Domain nodes.

Telnet

A remote terminal emulation protocol defined by the Defense Advanced Research Projects Agency for internetwork communications. Telnet uses TCP and IP as underlying protocols.

telnetd

The **daemon** process that accepts incoming Telnet requests.

Topology

The arrangement of networks and systems on those networks.

Transmission Control Protocol (TCP)

A protocol for sending datagrams from one network to another. It was defined by the Defense Advanced Research Projects Agency for internetwork communications.

Type A address

An **Internet address** where the network number is represented by a single byte with a leftmost 0 bit, and the local address consists of three bytes.

Type B address

An **Internet address** where the network number is represented by two bytes with the leftmost two bits having the value 10, and the local address consisting of two bytes.

Type C address

An **Internet address** where the network number is represented by three bytes with the leftmost three bits having the value 110, and the local address consisting of one byte.

Zone of authority

The area within a UNIX name server domain for which a name server is authoritative. Authoritative name servers can only resolve queries for their zone.

Index

Entries in color indicate task-oriented information.

Reader's Response

Please take a few minutes to send us the information we need to revise and improve our manuals from your point of view.

Document Title: *Configuring and Managing TCP/IP*
Order No.: 008543-A00
Date of Publication: July, 1988

What type of user are you?

_____ System programmer; language ____________________
_____ Applications programmer; language ____________________
_____ System maintenance person
_____ System Administrator
_____ Student Programmer
_____ Other

_____ Manager/Professional
_____ Technical Professional
_____ Novice

How often do you use the Apollo system?____________________

What parts of the manual are especially useful for the job you are doing?____________________

What additional information would you like the manual to include?____________________

Please list any errors, omissions, or problem areas in the manual. (Identify errors by page, section, figure, or table number wherever possible. Specify additional index entries.)____________________

Your Name Date

Organization

Street Address

City State Zip

No postage necessary if mailed in the U.S.

cut or fold along dotted line

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 78 CHELMSFORD, MA 01824

POSTAGE WILL BE PAID BY ADDRESSEE

APOLLO COMPUTER INC.
Technical Publications
P.O. Box 451
Chelmsford, MA 01824

FOLD

Reader's Response

Please take a few minutes to send us the information we need to revise and improve our manuals from your point of view.

Document Title: *Configuring and Managing TCP/IP*
Order No.: 008543-A00
Date of Publication: July, 1988

What type of user are you?

_____ System programmer; language ________________________

_____ Applications programmer; language ____________________

_____ System maintenance person

_____ System Administrator

_____ Student Programmer

_____ Other

_____ Manager/Professional

_____ Technical Professional

_____ Novice

How often do you use the Apollo system?____

What parts of the manual are especially useful for the job you are doing?______________________
____
____
____

What additional information would you like the manual to include?____________________________
____
____
____

Please list any errors, omissions, or problem areas in the manual. (Identify errors by page, section, figure, or table number wherever possible. Specify additional index entries.)_________________________
____
____
____
____
____
____

____
Your Name Date

____
Organization

____
Street Address

____
City State Zip

No postage necessary if mailed in the U.S.

cut or fold along dotted line

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 78 CHELMSFORD, MA 01824

POSTAGE WILL BE PAID BY ADDRESSEE

APOLLO COMPUTER INC.
Technical Publications
P.O. Box 451
Chelmsford, MA 01824

FOLD