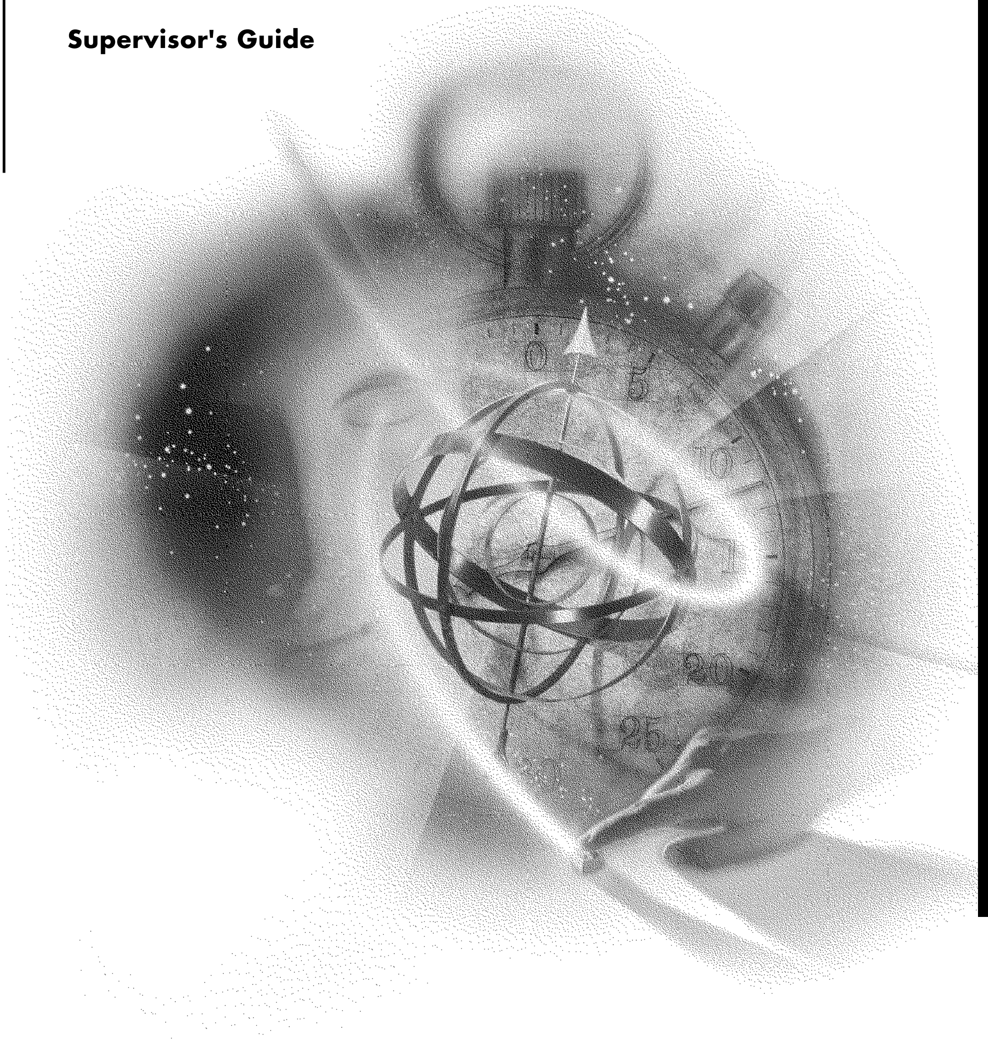
TCP/IP Transport

Supervisor's Guide



NetWare® 3.12

NETWORKING SOFTWARE

Novell®

Legal Notices

Novell, Inc. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

This product may require export authorization from the U.S. Department of Commerce prior to exporting from the U.S. or Canada.

Novell, Inc.
122 East 1700 South
Provo, UT 84606
U.S.A.

www.novell.com

Btrieve Installation and Operation
May 1993

Online Documentation: To access the online documentation for this and other Novell products, and to get updates, see www.novell.com/documentation.

Novell Trademarks

Third-Party Trademarks

Contents

About This Guide

This guide is written for network supervisors. It describes how to configure TCP/IP (Transmission Control Protocol/Internet Protocol) software on a NetWare v4.0 server using the Novell InterNetwork Configuration (INETCFG) utility, or the manual configuration method.

It also explains how to use the TCP/IP Console (TCPCON) NetWare Loadable Module™ (NLM™) to monitor and control TCP/IP.

Organization

Refer to the following chapters for the indicated information:

- Introduction for general information about TCP/IP.
- Configuring TCP/IP for information about configuring TCP/IP with the INETCFG utility.
- Managing Network Database Files for information about the sample Internet database files.
- Managing TCP/IP Networks for information about the TCP/IP Console services provided by the SNMP, SNMPLOG, and TCPCON NLM files.
- Appendix A, "IP Tunnel LAN Driver," on page 75 for information about IPX communication over an IP internet using IPTUNNEL.LAN.
- Appendix B, "TCP/IP Protocol Suite," on page 85 for more information about TCP/IP.
- Appendix C, "Manually Configuring and Loading TCP/IP," on page 111 for information about manually configuring and loading TCP/IP using the AUTOEXEC.NCF file (instead of using the INETCFG utility).

- Appendix D, “Manually Configured TCP/IP Examples,” on page 125 for information about several manually configured case examples.
- Appendix E, “Error Messages,” on page 133 for TCP/IP error message you can receive.

Related Publications

This guide describes administrative tasks and provides reference material for network supervisors. Use this guide with the *NetWare System Administration* manual.

The following Internet documents define and provide background information about the standard TCP/IP software:

- Case, J. and others, *A Simple Network Management Protocol*, RFC 1098, University of Tennessee at Knoxville, April 1989.
- Hedrick, C., *Routing Information Protocol*, RFC 1058, Rutgers University, June 1988.
- McCloghrie, K. and M. Rose, *Management Information Base for Network Management of TCP/IP-Based Internets*, RFC 1066, The Wollongong Group, August 1988.
- Plummer, D.C., *Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48-bit Addresses for Transmission on Ethernet Hardware*, RFC 826, Massachusetts Institute of Technology, November 1982.
- Postel, J. B. and Reynolds, J. K., *Standard for the Transmission of IP Datagrams over IEEE 802 Networks*, RFC 1042, USC/Information Sciences Institute, February 1988.
- Postel, J. B., *Internet Control Message Protocol-DARPA Internet Program Protocol Specification*, RFC 792, USC/Information Sciences Institute, September 1981.
- Postel, J. B., *Internet Protocol*, RFC 791, USC/Information Sciences Institute, September 1981.
- Postel, J. B., *Internet Control Message Protocol-DARPA Internet Program Protocol Specification*, RFC 793, USC/Information Sciences Institute, September 1981.
- Postel, J. B., *User Datagram Protocol*, RFC 768, USC/Information Sciences Institute, August 1980.

The preceding protocol specifications, as well as other protocol-related information, are gathered in the DDN Protocol Handbook, which is available from the following centers:

DDN Network Information Center SRI International 333 Ravenswood Avenue, Room EJ291 Menlo Park, CA 94025, USA

Defense Technical Information Center Cameron Station Alexandria, VA 22314, USA

This guide describes the TCP/IP NLM, which is required by all NetWare applications that use TCP/IP. NetWare is described in the following publications:

- *NetWare v4.0 Installation*, Part Number 100-000582-001, Novell, Inc., 1991.
- *NetWare v4.0 System Administration*, Part Number 100-000585-001, Novell, Inc., 1991.

The preceding publications are available from any authorized Novell dealer.

The following publications are NetWare-related:

- *NetWare NFS Supervisor's Guide* Part Number 100-000955-001, Novell, Inc., 1991.
- *Network C for NLMs* Software Developer's Kit Part Number 884-000011-003, Novell, Inc., 1991.

The following publications provide background information about computer networks and the principles, protocols, and architecture of TCP/IP:

- Comer, Douglas, *Internetworking with TCP/IP*, Volume I, Prentice-Hall, 1991.
- Tanenbaum, Andrew S., *Computer Networks*, Prentice-Hall, 1988.

Documentation Conventions

This guide uses special documentation conventions to distinguish descriptive text from other items. Specifically, these conventions are used when referring to the following:

- new terms
- command lines and command-line components
- system responses
- syntax statements

The characters < >, [ ], ( ), |, and " are used as metacharacters and, as such, you must not type these as part of user input.

Regular (nonbold, nonitalicized) characters represent system prompts or responses; for example:

:

Bold character strings represent user input, which must be entered exactly as shown. You can use any combination of uppercase and lowercase letters; for example:

`LOAD INETCFG <Enter>`

or

`Load inetcfg <Enter>`

Italicized character strings are used to show descriptive names for items that you must be replaced with appropriate values. For example, in the NetWare console command

`UNBIND IP FROM` ***`<driver>`*** **`<Enter>`**

You must replace <*driver*> with a driver name.

Normal Italic type is used for emphasis to introduce new terms.

< > (angle brackets) enclose a descriptive name for any value that is provided as output by a computer; for example:

`Error: <IP address> is illegal. Must be class A, B, or C.`

[ ] (square brackets) enclose an option item; for example:

```
BIND IP TO board_name ADDR=ip_addr [MASK=mask_addr]
  <Enter>
```

If a command, such as the preceding one, is too long to fit on one line, it wraps to the next line; such a command must be entered after the console prompt (:) but before the end-of-line (EOL) character.

{ } (braces) enclose multiple items separated by the `|' character; you must specify only one of the items. For example:

```
LOAD TCPIP [FORWARD={YES|NO}]<Enter>
```

| (vertical bar) separates items in a list of alternatives. If the list is enclosed in [ ], you can use any one or none of the items. If the list is enclosed in { }, you must use any one, but only one, of the items.

Numeric values are specified in binary, decimal, or hexadecimal bases. Binary numbers have the suffix bin (for example, 10bin). Decimal numbers have no prefixes or suffixes. Hexadecimal numbers have a 0x (zero and the letter x) prefix. For example, 0xA024 represents a hexadecimal number. Hexadecimal numbers have no prefixes or suffixes when they are used literally in an example or when referring to such an example.

NOTE: Notes are emphasized with a different font and the "Note" icon shown on the left).

Procedures are emphasized with the icon shown on the left.

IMPORTANT: Important notes are emphasized with a different font and the "Important" icon shown on the left.

1 Introduction

TCP/IP (Transmission Control Protocol/Internet Protocol) is a popular suite of standard networking protocols. These protocols are widely used, enabling dissimilar nodes in a heterogeneous environment to communicate with one another.

The general concept of connecting a network of dissimilar computers arose from research conducted by the Defense Advanced Research Projects Agency (DARPA). Within the framework of that research, DARPA developed the TCP/IP suite of protocols to communicate among networks, and implemented an internetwork called the *ARPAnet*, which later evolved into the *Internet*. The TCP/IP suite of protocols defines formats and rules for the transmission and receipt of information independently of any given network organization or computer hardware. Although the protocols were developed for the Internet, they are also applicable to other cases where networks must be connected.

The network, as conceived by DARPA and implemented with the TCP/IP suite of protocols, is a *packet-switched* network. A packet-switched network transmits information on the network in small segments, called *packets*. If one computer transmits a lengthy file to another computer, for example, the file is divided into many packets at the origin and then reassembled at the destination. The TCP/IP protocols define the format of these packets, including the origin of the packet, the destination of the packet, the length of the packet, and the type of packet, as well as the way computers on the networks are to receive and retransmit packets as necessary.

TCP/IP is a transport subsystem that brings TCP/IP connectivity to NetWare. This subsystem includes a collection of NLM (NetWare Loadable Module) files for NetWare designed to support applications requiring TCP/IP connectivity, such as the NetWare NFS™ (Network File System) product.

TCP/IP routing capabilities allow forwarding of IP traffic from one network to another, similar to the way many NetWare servers connect IPX

(Internetwork Packet Exchange) networks. TCP/IP uses the Routing Information Protocol (RIP) to communicate with other routers. This lets all routers in the internet discover the internet configuration without human intervention.

TCP/IP provides for communication between NetWare (IPX) networks across an IP internet that does not directly support IPX. This is known as IPX/IP tunneling (refer to Appendix A, "IP Tunnel LAN Driver," for introductory and configuration information).

In addition to routing services, TCP/IP provides a transport interface for the use of higher-level network services. This interface is used by NFS and by third-party applications written for either the 4.3BSD UNIX socket interface or the AT&T Streams Transport Layer Interface (TLI).

TCP/IP supports Ethernet, Token Ring, and ARCnet networks through the Novell Open Data-Link Interface (ODI). Therefore, it works with any network adapters of these types supported by a NetWare LAN driver certified for v4.0 or later.

NOTE: For more information about TCP/IP, the TCP/IP suite of protocols, TCP/IP addressing schemes, and TCP/IP subnetting, refer to Appendix B, "TCP/IP Protocol Suite."

TCP/IP Contents

The NetWare TCP/IP software includes the following NetWare NLM and database files:

- NetWare TCP/IP NLM (TCPIP.NLM)
- Simple Network Management Protocol NLM (SNMP.NLM)
- SNMP event logger NLM (SNMPLOG.NLM)
- TCP/IP Console NLM (TCPCON.NLM)
- IP configuration NLM (IPCONFIG.NLM)
- IPX/IP Tunnel module (IPTUNNEL.LAN)
- Sample Internet database files (GATEWAYS, HOSTS, NETWORKS, PROTOCOL, and SERVICES)

The TCP/IP NLM files are installed in the SYS:SYSTEM directory. The Internet database files are installed in the SYS:ETC\SAMPLES directory.

TCP/IP Subsystem

The TCP/IP subsystem provides underlying transport services designed to support service-level applications. TCP/IP, for example, supports implementations of the following applications:

- Line Printer Daemon (LPD)
- NFS server

The line printer daemon and an NFS server are part of NetWare NFS. The BSD socket and TLI libraries provide access to the transport services of TCP/IP. Figure 1 shows a sample TCP/IP subsystem.

Figure 1 NetWare TCP/IP Subsystem Overview

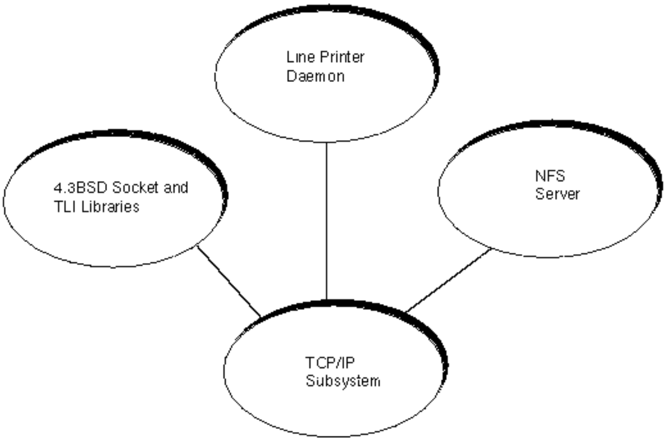
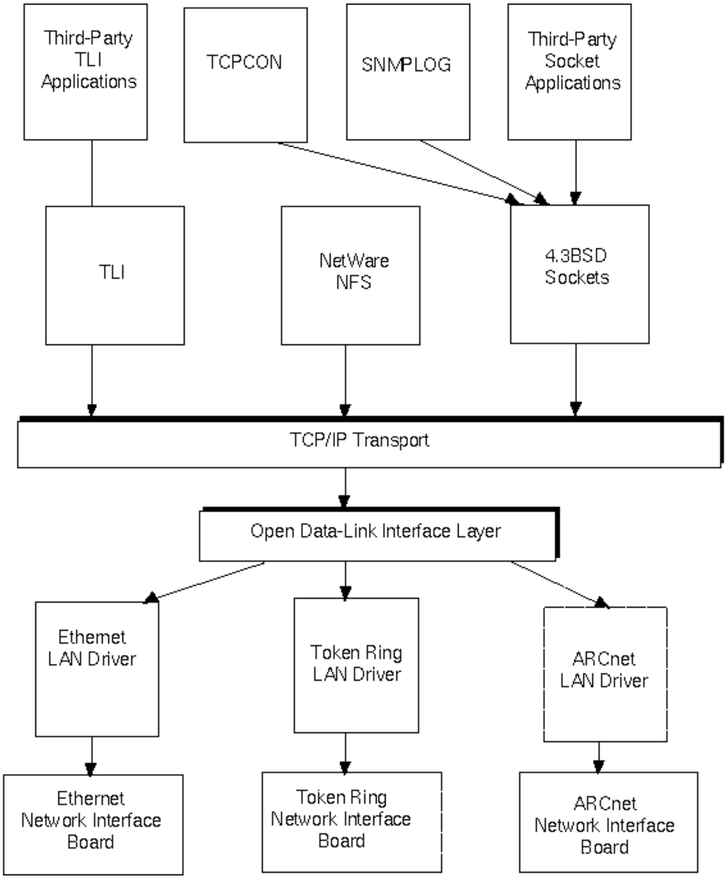


Figure 1-2 illustrates the architectural relationships of the TCP/IP subsystem.

Figure 2 TCP/IP Architectural Relationships


Third-Party TLI Applications
TCPCON
SNMPLOG
Third-Party Socket Applications
TLI
NetWare NFS
4.3BSD Sockets
TCP/IP Transport
Open Data-Link Interface Layer
Ethernet LAN Driver
Token Ring LAN Driver
ARCnet LAN Driver
Ethernet Network Interface Board
Token Ring Network Interface Board
ARCnet Network Interface Board

2 Configuring TCP/IP

Prerequisites for Running TCP/IP

Before running the TCP/IP software, your system must meet the following prerequisites:

- Computer. An 80386-based or 80486-based PC running NetWare 3.12 or later.
- Memory. Your system should have a minimum of 4 MB of memory to operate efficiently. Memory usage is summarized in Table 2-1.
- Network adapter. A network adapter (Ethernet, Token Ring, ARCnet, or FDDI) and the corresponding LAN driver must be installed in your computer.
- LAN driver. Your LAN driver must be certified for NetWare v4.0. Some third-party LAN drivers support the IPX protocol but do not support TCP/IP.

Table 1 TCP/IP Memory Requirements

Module	Memory	Usage
TCPIP.NLM	180 KB	Required
SNMP.NLM	35 KB	Required
TCPCON.NLM	100 KB	Optional*
SNMPLOG.NLM	8 KB	Optional *
IPTUNNEL.LAN	4 KB	Optional *
IPCONFIG.NLM	5 KB	Transient +

	* Consumes memory only when running.	+ Occupies memory only for a few seconds when running.

Configuration Overview

NetWare provides two methods for configuring the server:

1. Using the Internetworking Configuration (INETCFG) utility, which creates most of the LOAD and BIND commands automatically in a configuration file, after you configure the server.
2. Manually entering LOAD and BIND commands into the AUTOEXEC.NCF file.

To configure boards or protocols, you must choose only one of these configuration methods; you cannot use both. This chapter describes only the first method (INETCFG). Refer to Appendix C, "Manually Configuring and Loading TCP/IP," for information about the manual use of LOAD and BIND commands.

NOTE: Other services (such as AFP, ATPS, NFS, and so on) continue to be configured manually in the AUTOEXEC.NCF file.

Using INETCFG

INETCFG is installed during the installation of NetWare v4.0 and supersedes the manual use of most LOAD and BIND commands in the AUTOEXEC.NCF file. Board and protocol configurations are automatically entered into a configuration file that is directly referenced by the AUTOEXEC.NCF file at initialization time.

INETCFG supersedes the manual use of LOAD and BIND commands used with NetWare v3.*x*. When you save new configuration information using INETCFG, it updates its databases and automatically creates LOAD and BIND instructions, as appropriate, to be executed the next time the server or router initializes.

INETCFG supports the use of the most popular LAN media (Ethernet, Token Ring, ARCnet, LocalTalk, and FDDI) and the most popular network protocols (IPX, TCP/IP, and AppleTalk). However, since AppleTalk is documented separately, this chapter explains only the configuration of IPX and TCP/IP.

If this is the first time that INETCFG is being used on the server, the configuration program automatically reads the existing AUTOEXEC.NCF file and transfers protocol and interface configurations, LOADs, and BINDs to its own configuration file. The transferred entries in the AUTOEXEC.NCF file are then automatically converted (if you want) to commented lines. This eliminates the "manual reconfiguration" of an existing server. However, you should check all the configurations within INETCFG after the automatic transfer of configuration information from AUTOEXEC.NCF.

Although interfaces and protocols can be configured using INETCFG, there are a few services that can only be configured using LOAD commands (such as LOAD AFP, LOAD ATPS, LOAD NFS, and so on).

About INETCFG

INETCFG uses a series of menus and submenus to present a variety of parameter screens. This configuration interface allows you to enter names of new configurations, specify parameters, and override defaults without having to use the command line.

The items in the main menu correspond to the basic steps used to configure the network. These items should be configured in the same order as presented in the following configuration procedure. The last item (General Node Information) is optional.

The INETCFG main menu items are explained below:

- Boards: Configures the hardware parameters for the boards you have installed in your computer. The hardware parameters are dependent on the driver that each board uses, but usually include the interrupt and I/O address. Each board configuration and its network interface is given a unique name.
- Protocols: Configures specific parameters for transport and network layer communications through LAN connections. Depending on the protocol being configured, you may be asked for parameters that affect routing, checksums, and so on.
- Bindings: Allows a particular protocol to be bound to and routed across a particular network interface. INETCFG determines the frame type (encapsulation) that is used by the bound network interface, based on the protocol that is bound and the media type. In the case of IPX, the frame type can be changed if the default is not sufficient.
- General Node Information: Allows you to specify SNMP information such as the SNMP device name, hardware description, physical location, and human contact for the server/router. For troubleshooting purposes, it

also allows you to view and edit the AUTOEXEC.NCF file, view the INETCFG configuration file, and view the messages generated during system initialization.

Interacting with INETCFG

The bottom two lines of each INETCFG display show command keys and context specific "fast help."

The <Esc> key exits the current configuration window and returns you to the previous screen. Press <Esc> from the main screen to exit the program and save your configuration changes.

The <Enter> key views or modifies the selected option's configuration.

The <Ins> key adds a new configuration.

The <Del> key deletes or removes a configured option.

The <F1> key provides context-sensitive help messages. When you press <F1> a second time, it provides more general help. Press <Esc> to exit the levels of help screens.

Arrow keys move the current selection, indicated by a highlight bar, up and down through the choices.

Other keys may be meaningful in specific screens. If so, these command keys and their functions are displayed at the bottom of that particular screen.

To modify an existing configuration, select it from the appropriate menu and proceed through the menu hierarchy until you are displaying the screen on which the change is made. Make the change, press <Esc>, and then select SAVE.

INETCFG provides three levels of help in most of its menu levels (some menus have only two levels). Press <Esc>> to exit the help facility.

- Context-specific help is displayed within a single "fast help" line across the bottom of a menu or parameter screen. It is specific to the currently selected menu option or parameter line. It tells you the significance of a given menu option or gives you help in specifying a given parameter.
- Full context-specific help is displayed when you press <F1> while a menu or parameter screen is displayed. It provides more comprehensive help about the currently selected menu option or parameter line.

- More general screen-oriented help is displayed when you press <F1> a second time while viewing the full context-specific help. This level of help explains the overall purpose of the menu or parameter screen from which help was requested.

Configuration Names and Forming Associations

As a part of configuring network boards and their interfaces, you enter a mnemonic that identifies that configuration (some boards may provide several network interfaces). These network interface names are then used to bind a protocol configuration. This association is made by browsing through a list and selecting the desired network interface name.

When Configuration Changes Take Effect

You can run INETCFG while the server is operational. Most of the changes do not take effect until after the system is brought down and restarted. For example, you might enable a disabled board, disable an enabled board, change parameters on an enabled board, configure a new protocol, change the configuration of an existing protocol, bind a protocol to a network interface, and so on.

All these changes take effect *after* you type DOWN and EXIT at the server prompt, and then type SERVER to restart the server.

Configuration Procedure

The following configuration procedure is described in detail in this chapter.

1. Load INETCFG.NLM
2. Edit the AUTOEXEC.NCF File (Optional)
3. Configure LAN Boards
4. Configure Protocol Parameters
5. Bind Protocols to Network Interface
6. Configure General Node Information (Optional)
7. Reinitialize the Server

Perform the following steps after the NetWare v4.0 files have been installed on your system.

Load INETCFG.NLM

Complete the following steps to load INETCFG.NLM on your system:

1 Type the following command at the server prompt:

```
: LOAD INETCFG <Enter>
```

The INETCFG NLM loads. If this is the first time INETCFG has been run on this server, the program prompts whether you want to automatically transfer existing LOAD and BIND command lines from the AUTOEXEC.NCF file to the INETCFG configuration system and comment out the transferred lines from AUTOEXEC.NCF.

1a Select "Yes" to have INETCFG read the AUTOEXEC.NCF file and transfer the appropriate LOAD and BIND commands for protocols and interfaces to INETCFG's configuration file.

A backup file of the current AUTOEXEC.NCF file is created (SYS:SYSTEM\AUTOEXEC.BAK) to save your old configuration, in case you need to reconfigure your server to its original state.

The following lines are added to the AUTOEXEC.NCF file after the lines that set the internal IPX net number, the file server name, and other NetWare operating parameters:

```
LOAD CONLOG LOAD SNMP INITIALIZE SYSTEM ....
UNLOAD CONLOG
```

Table 2 AUTOEXEC.NCF Inserted Commands

Command	Action
LOAD CONLOG command	Loads a utility that captures the console messages generated during system initialization and writes the messages into the file ETC\CONSOLE.LOG. These messages can then be viewed at a later date and used to help troubleshoot your server installation.
LOAD SNMP and INITIALIZE SYSTEM commands	Initializes your system with the server configuration information entered in the INETCFG utility.
UNLOAD CONLOG command	(placed at the end of the file) Stops the logging of console messages when the startup is complete. If you want to log more than just the messages generated during startup, omit this step. Although this step is optional, its use is suggested to save disk space. There is no limit to the amount of disk space the log file can use.

If you have existing LOAD commands for CLIB, IPXS, TLI, AFTER311, or STREAMS (or any combination), the INETCFG program moves those lines above the LOAD SNMP command line that is automatically inserted. SNMP depends directly on these NLM files (indirectly on STREAMS) and must be located *after* these NLM files are loaded. INETCFG displays an informational message letting you know that these command lines have been moved.

INETCFG then displays its top-level menu of options:

```
Boards Protocols Bindings General Node
Information
```

You have now completed the only action in this step. However, if you are unfamiliar with INETCFG, you should read the remaining background information so that you understand what is requested of you during the remaining steps in this chapter.

1b Select "No" to edit your AUTOEXEC.NCF file manually.

If you select "No", you must comment out any BIND commands and most LOAD commands, and reconfigure your server using the INETCFG utility. Refer to the next step (Edit AUTOEXEC.NCF File) for more information about manually editing the AUTOEXEC.NCF file.

Edit AUTOEXEC.NCF File (Optional)

Perform this step *only* if you selected "No" to INETCFG's transfer process when the program prompted whether you wanted to transfer AUTOEXEC.NCF configuration information to the INETCFG configuration file (in Step 1). Complete these steps to edit your AUTOEXEC.NCF file:

1 Load the INETCFG NLM at the server prompt:

```
: LOAD INETCFG <Enter>
```

2 Select "General Node Information."

3 Select "View Configuration Information."

4 Select "AUTOEXEC.NCF File."

5 Ensure that the following INETCFG-related lines are present in the AUTOEXEC.NCF file:

```
LOAD CONLOGLOAD SNMPINITIALIZE SYSTEM
```

If these lines are not present, carefully insert these lines early in the file, ahead of any LOAD or BIND lines, but after the lines that set the file server name, IPX internal net number, and any other NetWare operating parameters.

The LOAD CONLOG command loads a utility that captures the console messages generated during system initialization and writes the messages to the ETC/CONSOLE.LOG file. These messages can then be viewed at a later date and used to help troubleshoot your server installation.

The LOAD SNMP and INITIALIZE SYSTEM commands are needed so your system is initialized with the server configuration information you entered using the INETCFG utility.

If you are adding the INETCFG software to an existing NetWare file server, you must remove or comment out the following commands if they exist. Preferably, you should comment out these commands by placing a # symbol at the beginning of each line.

IMPORTANT: Take notes about these commands since you must enter the parameter information again using the menu-driven INETCFG utility.

- LOAD commands for any network boards, drivers (for example, NE2000, 3C523, DL2000, etc.).
- LOAD commands for the following protocols and associated modules: TCPIP, TCONFIG, IPCONFIG.
- Note that AppleTalk has its own LOAD command lines that must be commented out of AUTOEXEC.NCF, if you use the NetWare for Macintosh product. Refer to the NetWare for Macintosh documentation for more information.
- All BIND commands to network interfaces.

  IMPORTANT: LOAD commands for other services (for example, LOAD AFP, LOAD ATPS, LOAD NFS, etc.) should *not* be removed or commented out of the AUTOEXEC.NCF file.

  IMPORTANT: Although interfaces and protocols can be configured using INETCFG, the preceding services can only be configured using manual LOAD commands.

  IMPORTANT: If your original AUTOEXEC.NCF file had a command to load ROUTE.NLM, you must enter a LOAD command for the new version of this NLM called SRCROUTE. The LOAD command for SRCROUTE uses the same parameters as ROUTE.NLM does, except that SRCROUTE must use a board name instead of a board number. SRCROUTE must be loaded after the INITIALIZE SYSTEM command.

6 At the end of the AUTOEXEC.NCF file, enter the following command:

```
UNLOAD CONLOG
```

This command stops the logging of console messages when startup is complete. If you want to log more than just the messages generated during startup, omit this step. Although this step is optional, its use is suggested to save disk space. There is no limit to the amount of disk space the log file can use.

7 Press <Esc> and select "Yes" to confirm that you want to save your changes to the AUTOEXEC.NCF file.

Configure LAN Boards

To configure LAN boards, complete the following steps:

1 Select the "Boards" option from the INETCFG main menu.

The "Configured Boards" screen is blank if you have not configured any network boards. This screen presents a table showing the board name, network interface name, driver type, hardware parameters, slot numbers, and status of each board you have configured.

2 Press <Ins> to add a new board.

3 Select the driver type that should be used with the board you are configuring.

The drivers listed on the screen represent the driver files (*.LAN files) that exist in the SYS:\SYSTEM directory. If the appropriate driver does not display in the list, you must install that driver file in the directory.

4 Enter an up-to-10 character alphanumeric name for this LAN board configuration.

5 Specify the hardware parameters (interrupt number, base I/O address, memory map address, and so on) for this board.

Based on the driver installation information file (*.LDI), INETCFG displays the hardware parameters specific to that driver. INETCFG uses the .LDI file to create screens of options and setting for the parameters, along with displaying the defaults specified for that board. Refer to the manual that came with the board for the specific hardware parameter settings.

If the specified driver does not have a driver installation information file, these parameters must be entered in text format and can be up to 50 characters long. For example:

```
LOAD NE2 SLOT=3 NAME=BACKBONE
```

Whether values are required for this field or not is determined by the driver being used.

6 You can optionally enter any comment (up to 51 alphanumeric characters) you have for this board.

7 Press <Esc> and select "Yes" if you are satisfied with your configuration choices.

The appropriate LOAD LAN Driver command is generated by INETCFG for use when you restart the server.

8 Repeat steps 2 through 7 for each network board.

At a later point, board configurations can be modified or deleted by using the function keys defined at the bottom of the screen. For example, to change the interrupt value for a board, simply select the appropriate board configuration and reenter the "Board Parameters" field. Deleting a board also deletes any ports or bindings configured for that board.

Configure Protocol Parameters

Complete the following steps for each protocol:

1 Select the "Protocols" option from the main menu.

A list of the available protocols is displayed. The status of each protocol is listed as enabled, disabled, or unconfigured. "Unconfigured" means that no parameter values have been saved for that protocol.

2 Select the TCP/IP protocol menu option.

Select TCP/IP from the "Protocols Supported" menu and configure the parameters that are displayed. The TCP/IP protocol parameters are explained in detail in Table 2-3.

NOTE: INETCFG can also be used to configure AppleTalk. For information about configuring AppleTalk, refer to the documentation provided with the AppleTalk support package.

The IPX protocol does not require any additional configuration under this option. IPX is, however, bound to the network interface in "Bind Protocols to Network Interface" section later on.

The two parameters for WAN Call Destinations (in the first TCP/IP screen) cannot be configured for NetWare v4.0 at this time.

Use of the Comment field is optional; it is provided for informational purposes only.

Table 3 TCP/IP Protocol Parameters

Expert Configuration Options	For some installations, you need to select this option to configure the following "expert-level" parameters. For most installations, these parameters can be left at the default values.
IP Packet Forwarding	This parameter determines whether the machine is used as an IP router or as an end node only. The default value is *Forward IP packets ("Router")*.
RIP Routing Protocol	This parameter allows the NetWare router/server to use the Routing Information Protocol (RIP) to exchange routing information with other routers on the network and maintain a dynamic routing table. If disabled, the router/server does not run RIP and is limited to using a static routing table. The default value is *Enabled*.
Static Routing Configuration	This parameter determines whether a static routing information database is loaded automatically during system initialization. If disabled, the static routing information database is ignored. The default value is *Disabled*.
Static Routing Tables	This parameter allows you to enter information into the static routing table. Each entry must include the destination network or host address and the address of the next hop. The parameters contained in this database (the SYS:\ETC\GATEWAYS database) are described below. Inserting or deleting an item here brings up the following menu items.
Route to Network or Host	This parameter determines whether the database entry routes datagrams addressed to a specific "host" or routes datagrams addressed to any host on a given "network." The default value is *Network*.
IP Address of Network or Host	This parameter specifies the IP address of the network or host to which data is routed. In the case of network addresses, enter only the network portion of the address, without trailing zeros. For example, enter 128.1 to route to network 128.1.0.0.
Next Hop Router on this Route	This parameter specifies the address of the closest router along the route. This router must have the same number for the network portion of the address as your local network interface.

Metric for this Route	This parameter represents the number of hops (routers) that the data must go through to reach its destination. Routes with lower values for the metric are chosen first when the NetWare router/server determines the best path. Enter a number between 1 and 16 to specify preference for this route. The default value is *1*.
Type of Route	This parameter specifies whether the route is *active* or *passive*. Active routes are continually updated by RIP, indicating their current status. Active routes can become *Invalid*. Passive routes are permanent entries that are loaded into IP during initialization. The default value is *Passive*.
SNMP Manager Table	This parameter allows you to enter the addresses of SNMP management stations that receive any SNMP traps from this server. The default is to send SNMP traps to the loopback address of 127.0.0.1. Refer to Chapter 4, "Managing TCP/IP Networks," for instructions about how to optionally log these loopback traps to a local file.

Bind Protocols to Network Interface

Binding protocols to network interfaces specifies that the selected protocol can be sent and received through the selected LAN interface card. Before you can accomplish this step, all affected protocols and boards must be already configured.

To bind protocols to the appropriate network interface, complete the following steps:

1 Select the "Bindings" option from the main menu.

2 Press <Ins> to add a new binding configuration.

3 Select a protocol to be bound.

A list of the available protocols is displayed. Select either IPX or TCP/IP from the list of configured protocols.

NOTE: INETCFG can also be used to configure AppleTalk. Refer to the documentation provided with the NetWare for Macintosh package for information about the relevant configuration parameters.

4 Select a network interface to which to bind.

Select one of your configurations.

5 Configure the protocol-specific parameters as required.

The IPX and TCP/IP bind parameters are explained in detail in Tables 2-4 and 2-5, respectively.

Table 4 IPX Bind Parameters

Network Interface	This field is informational only. It displays the name that you assigned to the associated network interface.
IPX Network Number	This parameter specifies the eight-digit hexadecimal IPX network number to be associated with the network bound to this interface. The number must be in the range of one to FFFFFFFE.
Frame Type	INETCFG assigns a default frame type according to the LAN media being used. This frame type can be changed only in the case of Ethernet and Token-Ring LANs. The default frame types are ETHERNET_802.3 and TOKEN-RING_SNAP, respectively. If you want to choose a different frame type, press <Return> to display the available options and select the desired option from the list. In rare cases where you need to bind IPX to more than one frame type, you can bind IPX to the network interface again and choose another frame type.

Table 5 TCP/IP Bind Parameters

Network Interface	This field is informational only. It displays the name that you assigned to the associated network interface.
Local IP Address of Interface	This parameter specifies the node's local IP address to be associated with the network bounded to this interface. The address must be entered as four numbers separated by dots (for example, 192.89.4.1). Each IP address on an IP internetwork must be unique, and is usually assigned by the network administrator. This parameter can be entered in decimal or hexadecimal form.
Subnetwork Mask of Connected Network	This parameter specifies the subnet(work) mask of the network bound to this interface. This setting must match the mask used by the other nodes on the network. In the mask, each bit of the address network number is set to one and each bit of the address host number is set to zero. If you do not specify the mask, IP assumes that the network is not divided into subnets. This parameter can be entered in decimal or hexadecimal form.
Expert Configuration Options	For some installations, this field selects the following "expert-level" parameters. For most installations, these parameters can be left at the default values.
Network Interface	This field is informational only. It displays the name that you assigned to the associated network interface.

Default Router	This parameter specifies the address of the router to which all packets with no specific route are sent. Normally, since RIP supplies this information, this field can be left unspecified. If you set this option, use caution. If two routers are configured with the other as a default router, a routing loop occurs, reducing network performance and potentially causing loss of connectivity.
Default Broadcast Address	This parameter determines which address the NetWare router/server uses for broadcasting on this network. This address should match the broadcast address of all the other machines on your network. The default value of FF.FF.FF.FF is recommended.
Multicast Override IP Address	This parameter determines the IP address to which IP should direct IP multicast packets, overriding IP's standard multicast handling.
Originate Default Route	This parameter determines whether the NetWare router/server announces itself as a default router for this network through RIP. The default value is *Disabled*.
Cost of Interface	This parameter specifies a number to indicate the preference for this interface. Use a higher number to discourage the use of the interface. For RIP, the maximum value is 15. The default value is *1*.
Use "Poison Reverse"	This parameter determines whether RIP uses Poison Reverse in RIP updates when this node is acting as a router. If disabled, RIP traffic on the network is reduced, at a small cost in stability. For more information about RIP and Poison Reverse, refer to RFC 1058. The default value is *Disabled*.

NOTE: The TCP/IP Permanent WAN Call Destination parameter cannot be configured for NetWare v4.0 at this time.

1. Press <Esc> and select "Yes" to save changes.
2. Repeat Steps 2 through 7 for each protocol combination you require.

   At a later point, binding configurations can be modified or deleted by using the function keys defined at the bottom of the screen. For example, to change the IP address of a network interface, select the appropriate binding configuration and reenter the "IP Address" field. Deleting a binding does not effect the configuration of the associated network interfaces.

Configure General Node Information (Optional)

To configure information that is used to assign values to the corresponding SNMP MIB II variables, complete the following steps:

1. Select the "General Node Information" option from the main menu.
2. Select the "Configure SNMP Information" option and configure each of the options with the appropriate information.

   Enter the information as you would in any text file. You are limited to 255 characters (carriage returns require two characters).

To display configuration information that can be used to troubleshoot your server installation, complete the following steps:

1. Select the "General Node Information" option from the main menu.
2. Select the "View Configuration Information" option.
3. Select the appropriate option to display the contents of the AUTOEXEC.NCF file, the information in the INETCFG configuration file, or the Console Log file messages generated during a system initialization.

   You can view INETCFG-generated LOAD and BIND commands, all at once or conveniently grouped according to Boards, Protocols, and Protocol bindings. All displays are read-only. To modify the parameters, you must use the appropriate INETCFG menu. The AUTOEXEC.NCF file can be edited directly within the INETCFG utility.

   Assuming the CONLOG utility has been loaded (usually through the AUTOEXEC.NCF file), the captured console messages generated during a system initialization can be displayed. These messages indicate any errors that occur during system initialization and can be helpful in localizing problems with your server installation and configuration. The file contains all the messages saved from the console.

   Only the last 8 KB of the file are shown. If the file is larger than 8 KB, other NetWare utilities, such as "Edit," can be used to display the entire file (/SYS:/ETC/CONSOLE.LOG).

Reinitialize the Server

Type the following command to bring down the server and reinitialize it:

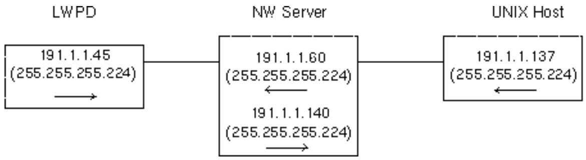
```
Down <Enter>

Exit <Enter>

Server <Enter>
```

You must reinitialize the server for the new configuration to take effect.

Testing TCP/IP

Once TCP/IP has been configured and the IP addressing specified, you can test the configuration to determine whether IP packets can be routed between the two networks. The easiest way to do this is with the PING utility. Testing can be initiated from either the workstation or the host.

For purposes of illustration, assume that the network configuration being tested consists of a DOS workstation running LAN WorkPlace for DOS (LWPD) on a physical network, which is attempting to connect to a UNIX host on another network, with a NetWare server routing IP packets between the two networks, as shown in the illustration below.



LWPD191.1.1.45 subnet mask 255.255.255.224 (IP network 191.1.1.32)

NW (board 1)191.1.1.60 subnet mask 255.255.255.224 (IP network 191.1.1.32)

NW (board 2)191.1.1.140 subnet mask 255.255.255.224 (IP network 191.1.1.128)

UNIX191.1.1.137 subnet mask 255.255.255.224 (IP network 191.1.1.128)

Test 1: Verify Connectivity to Server

To test the IP connection between the workstation and the server, enter the following command from the LWPD workstation:

```
PING 191.1.1.60 <Enter>
```

If the response returned is "`191.1.1.60 is alive`", the IP packet was successfully sent to the remote address, and the remote host (the server) was able to return the packet successfully.

If you do not receive the confirmation that the IP packet was returned correctly, there is a problem. At this point, the board on the NetWare server should appear as another host on the local network, so there should not be any routing issues.

Things to check include the following:

1. The network portion of the IP addresses on both the workstation and the server should be the same.
2. The workstation and the server should use the same subnet mask, which can be the default mask.
3. The physical connection between the two hosts exists and is a good connection.

Test 2: Verify Server Routing

The second step is to ping from the LWPD workstation through the server to the second board that is connected to the remote network, using the following command at the LWPD workstation:

```
PING 191.1.1.140 <Enter>
```

If the test is not successful, check the following:

1. Using the INETCFG "View Configuration Information" option under the "General Node Information" menu, verify that the following LOAD command has been correctly configured:

   ```
   LOAD TCPIP FORWARD=YES
   ```

   For more infomration about using the INETCFG, refer to the "Configure General Node Information" section in this chapter. If the server was not configured using INETCFG, the AUTOEXEC.NCF file on the server should contain this line.

2. Verify the TCP/IP configuration on the server by entering the following command at the console prompt:

   ```
   CONFIG <Enter>
   ```

   Verify that the IP addresses and the subnet masks of both boards are recognized as the values you configured for TCP/IP.

3. Verify that the LWPD workstation has the following entries in the NET.CFG file, indicating that the server (191.1.1.60) is the router to route packets from network 191.1.1.32 to network 191.1.1.128:

   ```
   IP_ROUTER 191.1.1.60

   IP_NETMASK 255.255.255.224
   ```

Once you are successfully able to ping the second board in the server, the routing on the LWPD workstation (the local TCP/IP host) is set up correctly and the NetWare server is routing packets correctly.

Test 3: Verify Routers to End System

The final step in testing the configuration is to ping from the LWPD workstation to the UNIX TCP/IP host on the remote network using the command:

```
PING 191.1.1.137 <Enter>
```

If the test is not successful:

1. Ping from the UNIX host to the NetWare server using the following command:

   ```
   PING 191.1.1.140 <Enter>
   ```

   If this does not work, check the items described in Test 1 to resolve the problem. After the UNIX host has successfully pinged 191.1.140, continue with the next test.

2. Ping from the UNIX host to the board in the NetWare server that is attached to network 191.1.1.32 using the following command:

   ```
   PING 191.1.1.60 <Enter>
   ```

   If this does not work, proceed to the next test below.

3. Verify that the routing is configured correctly on the UNIX host ensuring that IP packets can be routed to the NetWare server, that the NetWare server at 191.1.1.140 is recognized as the router to route packets from network 191.1.1.128 to network 191.1.1.32.

On a UNIX system, the general command is as follows:

```
ROUTE ADD <destination network> <gateway address>
  <metric>
```

For the UNIX host, check the manual pages for the exact syntax for adding routes. In this example, the command is as follows:

```
ROUTE ADD 191.1.1.32 191.1.1.140 1
```

This ROUTE command specifies that if there is a packet to be routed to network 191.1.1.32, send it to the router at address 191.1.1.140, which is only one hop away (the packet must be forwarded through one router to get to its remote network destination).

To assign the NetWare server ((IP address of 191.1.1.140) as the default router for the UNIX system, to route packets destined for any network other than the local, the command syntax is as follows:

```
ROUTE ADD 0.0.0.0 191.1.1.140 1
```

If there are other non-NetWare TCP/IP routers on the network, verify that those routers support RIP. The TCP/IP router supports only RIP. If the other routers do not support RIP, then static routes have to be added at all routers. Configuring static routes for the TCP/IP router is explained in Table 2-3 of this chapter, or in Chapter 4, "Managing TCP/IP Networks," in the section entitled "Configuring Static Routes Using the IPCONFIG NLM."

NOTE: If routing is configured correctly on one network, but not on the other, packets may be sent to a host on the remote network but the remote host will not be able to route the packets back. In this example, the LWPD workstation may have the correct routing entries in the NET.CFG file (resulting in the ping packet being correctly sent to the UNIX host on the remote network), but the UNIX host does not know how to route the packet back because it does not have a ROUTE ADD command set up.

After the preceding steps have been completed, and routing has been configured correctly on both networks, the hosts on one network should be able to ping hosts on the remote network.

3 Managing Network Database Files

TCP/IP uses four database files to convert internal data, such as IP addresses, into more identifiable and workable names. The user interface for TCPCON and other NLM files uses these database files. You can inform TCP/IP of names and addresses of local nodes and networks by adding that information to these files. The name files (HOSTS, NETWORKS, PROTOCOL, and SERVICES) are cached in memory to avoid disk access during lookup. Because of this, TCP/IP takes up more memory. If this is not desirable, keep the size of the database down, or simply delete the files.

TCP/IP finds these files in the SYS:ETC\ directory:

- HOSTS—maps host names to IP addresses
- NETWORKS—maps network names to network addresses
- PROTOCOL—maps protocol names to IP protocol numbers
- SERVICES—maps service names to transport protocol/well-known port pairs

If you are configuring TCP/IP for the first time, it is recommended that you start by copying the sample database files from SYS:ETC\SAMPLES to SYS:ETC. This provides you with some examples to refer to as you add your own entries, and also provides TCP/IP the PROTOCOL and SERVICES files, which you do not need to modify.

You can modify these files using a standard text editor from any NetWare client, or you can use the EDIT NLM from the NetWare system's console. The following sections describe the formats of the files, which are compatible with the same files on standard BSD UNIX systems. The examples in the sample files can also help you create your own entries.

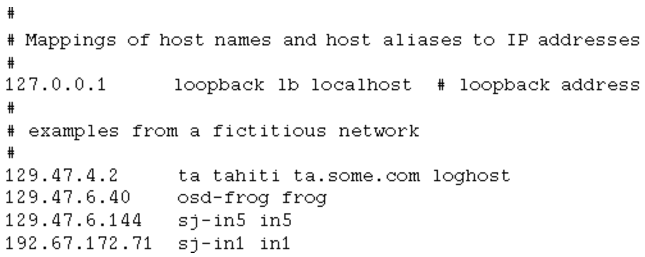
Each database file describes a table. Each line of the file describes a separate table entry. Blank lines and comments are ignored. Comments begin with a number sign (#) anywhere in a line and include the number sign and any characters following it on the same line.

NOTE: Do not use the sample addresses provided in the database files if you are connected to the Internet; these addresses are for example only.

HOSTS File

The file SYS:ETC\HOSTS contains information about the known hosts on the IP internet. Typically, it is centrally administered and distributed to all local hosts. Its format, as shown in Figure 3-1, is identical to /etc/hosts on UNIX systems. Each entry provides information about a single host. An entry cannot extend beyond one line.

Figure 3 Sample HOSTS File



```
#
# Mappings of host names and host aliases to IP addresses
#
127.0.0.1      loopback lb localhost  # loopback address
#
# examples from a fictitious network
#
129.47.4.2     ta tahiti ta.some.com loghost
129.47.6.40    osd-frog frog
129.47.6.144   sj-in5 in5
192.67.172.71  sj-in1 in1
```

The HOSTS file entry has the following format:

IP_address host_name [*alias* [...]]

The *IP_address* is a 4-byte address in standard dotted decimal notation. Each byte is a decima, hexadecimal, or octal value and is separated by a period. Hexadecimal numbers must start with the character pair 0x or 0X; octal numbers must start with 0.

The *host_name* is the name of the system associated with this internet address. The name cannot contain a space, tab, number sign (#), or end-of-line character. Each host name must be unique.

The *alias* is another name for the same system. Typically, this is a shorter name. A single host can have from one to ten aliases. For example, the host "sales" could have the following address and aliases:

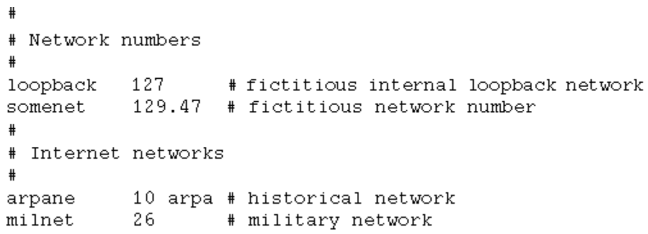
```
129.0.9.5 sales sa localhost
```

The sample file SYS:ETC\SAMPLES\HOSTS is included with the TCP/IP software. When you are configuring TCP/IP for the first time, copy the sample HOSTS file from SYS:ETC\SAMPLES to SYS:ETC. You then edit the SYS:ETC\HOSTS file. You can change your configuration at any time by editing your existing SYS:ETC\HOSTS file.

NETWORKS File

The file SYS:ETC\NETWORKS contains information about the networks in your internetwork. Each entry provides information about one network. An entry cannot extend beyond one line. Figure 3-2 shows a sample NETWORKS file.

Figure 4 Sample NETWORKS File



```
#
# Network numbers
#
loopback   127      # fictitious internal loopback network
somenet    129.47   # fictitious network number
#
# Internet networks
#
arpane     10 arpa # historical network
milnet     26      # military network
```

The NETWORKS file entry has the following format:

`network_name network_number``[/network_mask] [`*`alias`*`[...]]`

The *network_name* is the name of the network associated with this network number. The name cannot contain a space, tab, number sign (#), or end-of-line character. The network name must be unique.

The *network_number* is the number of the network. Hexadecimal numbers must start with the character pair 0x or 0X. The *network_number* can be specified with or without trailing zeros. For example, the addresses 130.57 and 130.57.0.0 denote the same IP network.

The *network_mask* is the subnet mask of the network. Like IP addresses, it can be specified in octal, decimal, or hexadecimal notation. This field is optional. If not specified, the subnet mask is deduced so that it excludes all trailing zeros. For example, 130.57.1.0 implies the mask 255.255.255.0.

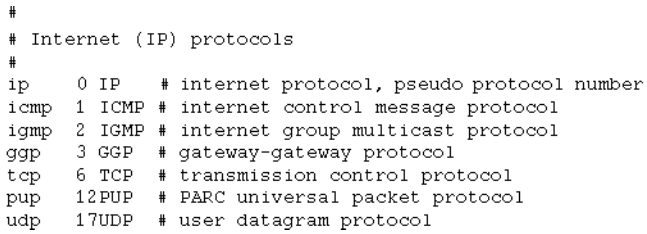
The *alias* is another name for the same network; you can specify up to ten aliases for a network.

The sample file SYS:ETC\SAMPLES\NETWORKS is included with the TCP/IP software. When you are configuring TCP/IP for the first time, copy the sample NETWORKS file from SYS:ETC\SAMPLES to SYS:ETC. You then edit the SYS:ETC\NETWORKS file. You can change your configuration at any time by editing your existing SYS:ETC\NETWORKS file.

PROTOCOL File

The file SYS:ETC\PROTOCOL, as shown in Figure 3-3, contains information about the known protocols used on the Internet. Each line provides information about one protocol. An entry cannot extend beyond one line.

Figure 5 Sample PROTOCOL File



```
#
# Internet (IP) protocols
#
ip    0 IP   # internet protocol, pseudo protocol number
icmp  1 ICMP # internet control message protocol
igmp  2 IGMP # internet group multicast protocol
ggp   3 GGP  # gateway-gateway protocol
tcp   6 TCP  # transmission control protocol
pup   12PUP  # PARC universal packet protocol
udp   17UDP  # user datagram protocol
```

The PROTOCOL file entry has the following format:

protocol_name protocol_number [*alias* [...]]

The *protocol_name* is the name of the Internet protocol associated with this protocol number. The name cannot contain a space, tab, number sign (#), or end-of-line character.

The *protocol_number* is the number of the Internet protocol.

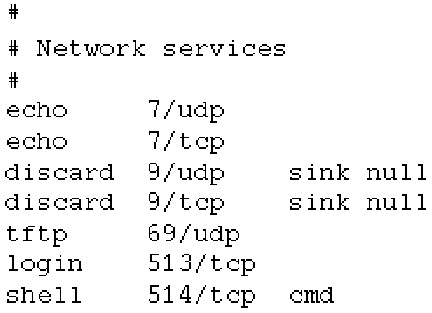
The *alias* is an alternate name for the protocol.

The sample file SYS:ETC\SAMPLES\PROTOCOL is included with the TCP/IP software. When you are configuring TCP/IP for the first time, copy the sample PROTOCOL file from SYS:ETC\SAMPLES to SYS:ETC. You then edit the SYS:ETC\PROTOCOL file. You can change your configuration at any time by editing your existing SYS:ETC\PROTOCOL file.

SERVICES File

The file SYS:ETC\SERVICES, as shown in Figure 3-4, contains information about the known services used on the IP internetwork. Each entry provides information about one service. An entry cannot extend beyond one line.

Figure 6 Sample SERVICES File



```
#
# Network services
#
echo     7/udp
echo     7/tcp
discard  9/udp    sink null
discard  9/tcp    sink null
tftp     69/udp
login    513/tcp
shell    514/tcp  cmd
```

The SERVICES file entry has the following format:

service_name port_number/protocol_name [*alias* [...]]

The *service_name* is the name of the service associated with this port number and protocol name. The name cannot contain a space, tab, number sign (#), or end-of-line character. These are generally application, presentation, or session-level services, such as TFTP, FTP, SMTP, and TELNET.

The *port_number* is the number of the Internet port used by the service.

The *protocol_name* is the protocol with which the service is associated. This is generally a transport or network-level protocol, such as TCP or UDP. You must put a slash between the port number and the protocol name (for example, SMTP 25/TCP MAIL).

The *alias* is an alternate name for the service.

The sample file SYS:ETC\SAMPLES\SERVICES is included with the TCP/IP software. When you are configuring TCP/IP for the first time, you should copy the sample SERVICES file from SYS:ETC\SAMPLES to SYS:ETC. You then edit the SYS:ETC\SERVICES file. You can change your configuration at any time by editing your existing SYS:ETC\SERVICES file.

4 Managing TCP/IP Networks

This chapter describes TCP/IP network management using the SNMP, SNMPLOG, and TCPCON NLM files.

Simple Network Management Protocol (SNMP) is the most popular network management protocol in the TCP/IP protocol suite. SNMP lets TCP/IP-based network management clients exchange information about the configuration and status of nodes on a TCP/IP-based internet. The information available is defined by a set of *managed objects* referred to as the SNMP Management Information Base (MIB). The subset of managed objects that make up the TCP/IP portion of the MIB is maintained by each TCP/IP node. SNMP also generates trap messages used to report significant TCP/IP events asynchronously to interested clients.

TCP/IP supports SNMP access to the complete TCP/IP portion of the MIB, as maintained by the protocol stack. The TCP/IP network management services are provided by three NLM files:

- SNMP
- SNMPLOG
- TCPCON

The SNMP NLM acts as an SNMP Agent for remote clients. It presents the standard TCP/IP portion of the MIB and generates trap messages required by the TCP/IP protocol stack. SNMPLOG and TCPCON are clients of the SNMP Agent NLM. They use SNMP to communicate with the TCP/IP protocol stack.

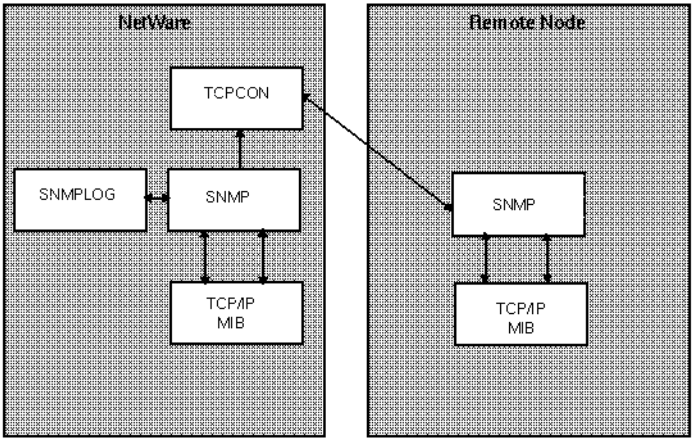
The SNMPLOG NLM processes SNMP trap messages directed to the local TCP/IP node and writes them to the log file

```
SYS:ETC\SNMP$LOG.BIN
```

This includes trap messages generated locally by the SNMP NLM for reporting local events and trap messages sent to this node from remote nodes for reporting remote events.

The TCPCON NLM, which presents a menu-driven console interface, lets you use SNMP to access the local TCP/IP protocol stack MIB as well as the MIB of any remote protocol stack supporting the TCP/IP MIB. It also provides access to the trap log file maintained by SNMPLOG. Normally, the local TCP/IP node is TCPCON's target, but it can also access any other TCP/IP node in the internet that supports SNMP. This includes TCP/IP protocol stacks provided by other vendors. Figure 4-1 shows the local and remote access methods.

Figure 7 Local and Remote TCP/IP Access



SNMP

The SNMP NLM performs the SNMP Agent services required by the TCP/IP module. It provides SNMP managers, such as TCPCON, access to TCP/IP protocol stack managed objects. It also generates SNMP trap messages used to report significant TCP/IP events.

The SNMP NLM provides load time parameters for establishing the community name used during SNMP trap message generation. The SNMP NLM also provides acceptable community names for read-only and read/write SNMP request message reception. Community names are used by the SNMP Agent for message authentication. The community name contained in a request message from an SNMP manager client must match the name established by the agent. The agent discards any request message that contains an invalid community name, without generating a response to the managed client.

TCP/IP always requests BIND information from SNMP when loading. If you are using the INETCFG utility to configure your server, the SNMP Agent returns any information in the INETCFG configuration file pertaining to IP BINDs. If any Bind commands exist, IP will autobind according to the parameters received.

Loading SNMP

The SNMP NLM is automatically loaded by the TCP/IP module if the SNMP Agent is not already running. To establish SNMP Agent community names, other than the default values described in the following section, you should explicitly start the SNMP NLM with the LOAD command and all appropriate option values.

If you used the INETCFG utility when you configured your server, the SNMP LOAD command is automatically written into the INETCFG configuration file. If you are not using INETCFG to configure your server, the SNMP LOAD command can be issued at the command line or inserted into the AUTOEXEC.NCF file.

The command line has the following format:

```
LOAD SNMP [options] <Enter>
```

SNMP Community Name Options

The LOAD command line accepts three SNMP community name options:

- MonitorCommunity
- ControlCommunity
- TrapCommunity

Each option sets the community name for the indicated community. Community names are used to authenticate SNMP request messages received at the agent. The community name in a message requesting a given access type must match the name defined for that access type by one of the SNMP community options.

Community names are arbitrary ASCII strings up to 32 characters in length. They can include any characters *except* space, tab, open square bracket ([), equal sign (=), colon (:), semicolon (;), or number sign (#).

If an option name (for example, MonitorCommunity) is followed only by an equal sign with no argument, *any* community name offered by the remote SNMP console is used for that community. Community names are case-sensitive; however, option names are case-insensitive. When specifying option parameters, you only have to enter the first character of the option name, although complete or partial names are also accepted.

NOTE: If you enter a community option without following it with an equal sign, the access defined for the specified community is disabled.

If you enter the option name with an equal sign and a community name value, the community name offered by the remote SNMP console must match the value specified.

The `MonitorCommunity` option sets the community name for read-only access. The default value is public. The command-line format is:

```
LOAD SNMP MonitorCommunity=<community name <Enter>
```

The **`ControlCommunity`** option sets the community name for read/write access. By default, this community is disabled. If you disable ControlCommunity, you disable all write access. Any community name established for read/write access is also valid for read-only access. The command-line format is:

```
LOAD SNMP ControlCommunity=<community name <Enter>
```

The **`TrapCommunity`** option sets the community name to be included in SNMP trap messages issued by the SNMP Agent. The default value is *public*.

If you disable the trap community, all trap message generation is suppressed. The command-line format is:

```
LOAD SNMP TrapCommunity=<community name <Enter>
```

LOAD Command-Line Examples

To set the read-only community name to secret:

```
LOAD SNMP MonitorCommunity=secret <Enter>>
```

To disable any read/write access:

```
LOAD SNMP ControlCommunity <Enter>
```

To allow any community name to be used to get read/write access:

```
LOAD SNMP ControlCommunity= <Enter>
```

To allow any community name to have read-only access and set the read/write community name to private (note the abbreviated option name Monitor and Control for the community options):

```
LOAD SNMP Monitor= Control=private <Enter>
```

SNMPLOG

To log SNMP trap messages received by the local NetWare TCP/IP node, load the SNMPLOG NLM after loading the TCPIP NLM. Once running, the SNMPLOG utility writes all arriving SNMP trap messages to disk. SNMPLOG is a background process that does not provide a console interface. If you do not require trap message logging, or if trap messages are being directed to another TCP/IP node, you do not need to load the SNMPLOG NLM.

Loading SNMPLOG

Use the LOAD command to link the SNMPLOG NLM with the operating system. This command starts the utility and initiates trap message logging.

The command line has the following format:

```
LOAD SNMPLOG <Enter>
```

The command line requires no parameters. Logging can be stopped at any time by unloading SNMPLOG using the UNLOAD command:

```
UNLOAD SNMPLOG <Enter>
```

SNMPLOG Operation

Trap message information is written in binary form to the file SYS:ETC\SNMP$LOG.BIN. No size restriction is placed on the log file. To avoid consuming excessive disk space, you should delete the file periodically. If the file does not exist when a trap message is received, SNMPLOG will create it. The log file contents are not human-readable; however, the TCPCON NLM can be used to view the file contents or delete the file.

The TCPIP NLM sends trap messages to the local node unless it is loaded with the *trap* parameter. In other words, unless otherwise instructed, each node logs its own traps. The *trap* parameter on the TCPIP LOAD command can redirect traps to some other node. (Refer to Appendix C, "Manually Configuring and Loading TCP/IP," for more details on the TCP/IP *trap* parameter.)

The trap address could be set to the same server on all nodes. In this way, all trap messages from all SNMP agents can be logged at a single node.

TCP/IP Console (TCPCON)

To monitor TCP/IP operations, load the TCPCON NLM (TCPIP Console) after the TCPIP NLM has been loaded. TCPCON provides a menu-driven interface for probing the TCP/IP MIB and lets the console operator look at the local trap log maintained by SNMPLOG. Although most useful for monitoring the local node's TCP/IP stack, TCPCON can also be directed to monitor a remote TCP/IP stack. This can be done by specifying the target host as an argument when TCPCON is loaded or by choosing the Change Host menu item from the TCPCON main menu.

NOTE: Only the local node trap log information is available, even when TCPCON is connected to a remote node.

Loading TCPCON

Use the LOAD command to link TCPCON with the operating system. This command starts the utility and initiates communication with a target host system.

The command line has the following format:

```
LOAD TCPCON [target_host] [options...] <Enter>
```

The LOAD command line takes the desired *target_host* as an optional argument. The target host is specified as either a symbolic name or an IP address in dotted notation. Symbolic host names are maintained in the file SYS:ETC\HOSTS. For more information about the HOSTS file, refer to Chapter 3, "Managing Network Database Files." For more information about IP addresses and dotted notation, refer to Appendix B, "TCP/IP Protocol Suite."

TCPCON Options

The LOAD command line also accepts a number of optional parameters controlling the operation of the utility. These parameters, with the exception of help={yes | no}, are also accessible from the TCPCON options menu. When specifying option parameters at the LOAD command line, you only have to specify the first character of the option name (for example "c" for "community"), complete or partial option names are also accepted. The following paragraphs describe each option. Table 4-1 summarizes the options.

Table 6 TCPCON Options

Option	Values	Explanation
community	Text string of 1 to 32 characters	Specifies the community name included in each SNMP request message.
poll	Integer from 0 to 900	Specifies the SNMP request message poll rate in seconds.
retry	Integer from 1 to 100	Specifies the maximum unanswered SNMP requests before displaying the HOST UNAVAILABLE message.
sort	Yes or No	Enables or disables sorting the symbolic host and network name lists.
timeout	Integer from 0 to 120	Specifies the time period in seconds to wait for an SNMP response.

Community option

The `community=`<*community name*> option specifies the community name included in each SNMP request message. This name is used by SNMP Agents to determine the type of MIB access that should be granted. By default, the SNMP Agent grants read-only MIB access to requests bearing the community name *public* and prohibits write access. Other SNMP Agents may require different community names. The community name must be between 1 and 32 characters. The default name is *public*.

Poll option

The `poll=`<*rate*> option specifies the rate, in seconds, at which the specified TCP/IP target host is sent SNMP request messages to update management statistic displays. The poll rate must be between 0 and 900 seconds. A value of zero results in continuous polling. The default poll rate is 1 second.

Retry option

The `retry=`<*count*> option specifies the number of consecutive unanswered requests to be issued before indicating that the selected TCP/IP host is unreachable. When this count is exceeded, the message HOST UNAVAILABLE is shown in the lower-right corner of the summary display. The retry count must be between 1 and 100. The default value is 3 retries.

Sort option

The `sort={yes | no}` option controls the presentation of symbolic host and network names when selecting a name from a list. When sorting is enabled, the symbolic names are presented in alphabetic order. When sorting is disabled, the names are presented in the same order as contained in the HOSTS and NETWORKS database files. The default setting is `yes`.

Timeout option

The `timeout=`<*count*> option specifies the time period, in seconds, to wait for a response from the selected TCP/IP host before retrying. The default value is 1 second.

TCPCON Error Reporting

TCPCON error reports take a number of forms depending on the type and severity of the error. This section describes the various error reports.

Nonrecoverable Errors

Nonrecoverable load time initialization errors can occur that cause TCPCON to abort. Reports of these errors are displayed on the TCPCON screen. The reports remain displayed until the user acknowledges the error by pressing a key, at which point TCPCON unloads itself.

Specifying invalid option names or values on the LOAD command line create nonrecoverable errors resulting in an error display that identifies the offending option name or value. Invalid command-line error reports also display the proper LOAD command-line syntax, as shown in the following example:

```
bad option: <option name>

format: LOAD TCPCON [target_host] [option...]
```

where the options are as follows:

```
community=name string

poll=interval in seconds

retry=count

sort=yes/no

timeout=interval in seconds
```

TCPCON internal initialization errors are also nonrecoverable and cause the utility to abort. These errors typically occur as a result of some system resource depletion such as free memory, or because of a failure in TCP/IP.

Possible initialization errors are as follows:

```
TCPCON help screen access failed.

TCPCON help screen initialization failed.

TCPCON user interface initialization failed.

UDP socket allocation failed.

UDP socket bind failed.

UDP socket connect failed.

UDP socket ioctl failed.
```

Recoverable Errors

TCPCON reports recoverable runtime errors that are generally the result of invalid user input, target host system failures, or some transient network failures. Runtime errors are reported when TCPCON fails to complete a user-requested function, such as display of a specific statistic screen. They are displayed until the user acknowledges the error by pressing a key. Once the error is acknowledged, TCPCON resumes normal operation. Typical recoverable errors are as follows:

```
Invalid hex string, please use xx-xx-xx-xx-xx-xx format.

Invalid target host, please use symbolic name or dotted
  notation.

IP Routing table entry edit failed due to: NO HOST RESPONSE

IP Routing table access failed due to: UNKNOWN OBJECT

TCPCON help was disabled at load time (help=no).

UDP receive operation failed.

UDP send operation failed.
```

TCPCON also presents warning messages in the main screen summary display. These messages report errors that occur during normal polling of the selected target host system. For example, a target host that occasionally fails to respond to SNMP requests for summary information causes the HOST UNAVAILABLE warning to be displayed from time to time. Warning messages, which do not require acknowledgment from the user, are discussed in the TCPCON Console main screen section of this chapter.

TCPCON Control Keys

TCPCON presents a standard NetWare utility user interface. The following paragraphs describe the control keys used by TCPCON and Table 4-2 gives a brief summary of these keys.

The *arrow* keys move the highlight through the menu items. The arrow keys can also be used to scroll through a table one line at a time. You can use the Up and Down arrow keys to highlight a menu item, table entries, and statistics displays. The highlight must be positioned on an item before the item can be selected.

Use the <Delete> key to remove an existing entry from a writable table. You can also use the <Delete> key to select and clear any host name, hexadecimal, or text type field.

Use the <Enter> key to select the currently highlighted menu item, table entry, or statistical object. Selecting a table entry causes additional information associated with the entry to be displayed when available.

The <Esc> key returns you to the previous menu level or ends a field selection.

The <F1> key displays online help for the current display. Help screens are available for individual statistical objects, as well as for each table or menu display. To access statistical objects help, first select the desired object, as described later in this section.

Use the <Insert> key to add a new entry to a writable table. Table entries are created using a form containing default values for each table entry field. Each writable field can be edited before saving the new entry.

You can also use the <Insert> key to select a host name display field and choose from the list of host names contained in the SYS:ETC\HOSTS database file.

Use the <PageUp> and <PageDown> keys to scroll through table entries one screen at a time. You can press <Ctrl-PageDown> to get to the bottom of a table or <Ctrl-PageUp> to get to the top of a table.

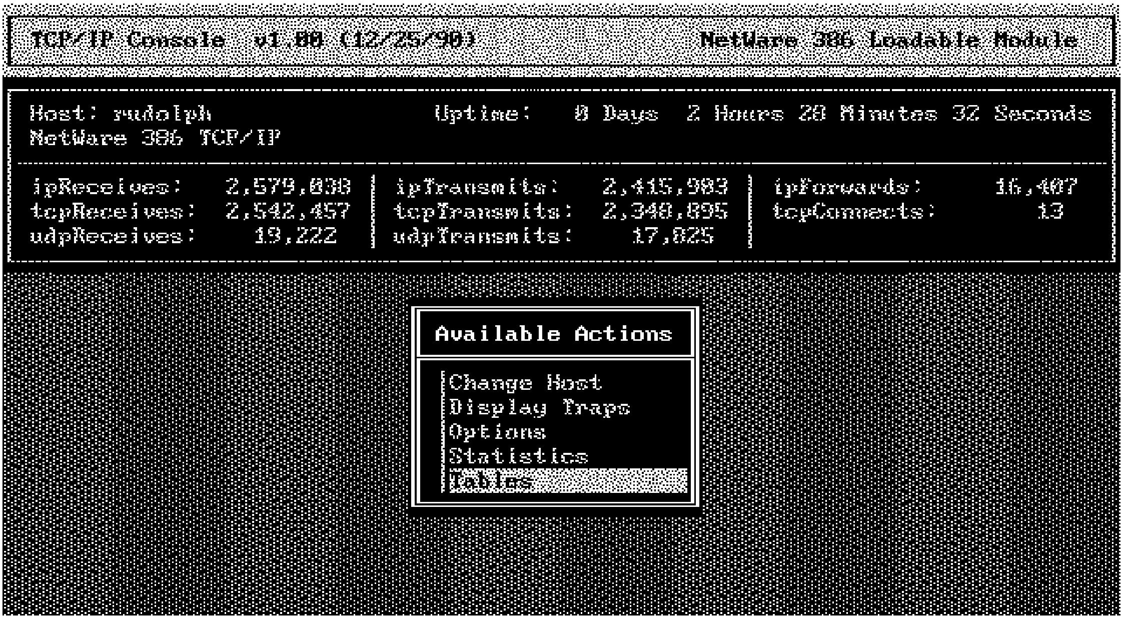
Use the <Tab> key to toggle host displays between symbolic name and dotted decimal IP address representation. Hosts that do not have a symbolic name in the SYS:ETC\HOSTS file are always displayed as IP addresses.

Table 7 TCPCON Control Keys

Key	Description
Arrow keys	Move the highlight through menu items.
<Delete>	Removes an existing entry from a writable table or selects and clears a host name field.
<Enter>	Selects the currently highlighted menu item or selects a host name from the list.
<Esc>	Returns to the previous menu level.
<F1>	Accesses online help.
<Insert>	Adds a new entry to a writable table.
<PageUp>/ <PageDown>	Scrolls through table entries
<Tab>	Toggles between the host name and the IP address display

TCP/IP Console Main Screen

The TCP/IP Console main screen displays a summary of information for the selected TCP/IP target host. The display, which is located at the top of the screen and remains active throughout TCP/IP Console (TCPCON) operation, presents various statistics from the TCP/IP portion of the MIB.



```
TCP/IP Console  v1.00 (12/25/90)                    NetWare 386 Loadable Module

Host: rudolph                    Uptime:   0 Days  2 Hours 28 Minutes 32 Seconds
NetWare 386 TCP/IP

ipReceives:    2,579,038 | ipTransmits:    2,415,983 | ipForwards:      16,407
tcpReceives:   2,542,457 | tcpTransmits:   2,340,895 | tcpConnects:         13
udpReceives:      19,222 | udpTransmits:      17,825 |
```

The summary display includes the following information:

- Target host name or IP address
- Target host system description
- Length of time host protocol stack has been running
- IP transmit and receive datagrams
- Number of IP packets forwarded
- TCP transmit and receive segments
- Number of TCP connections
- UDP transmit and receive datagrams
- Warning message field

The TCPCON warning field, located in the lower-right corner of the summary display, is normally blank. When an SNMP-related runtime error is detected, a short blinking message is displayed in this field. Warning messages are displayed until the associated error condition is corrected. Possible warning messages and their meanings are as follows:

```
HOST UNAVAILABLE
```

The specified target host system is not responding to SNMP request messages issued by TCPCON. Either the SNMP Agent at the target system is not active, or the community name in the TCPCON generated request messages does not match the community name established at the target system.

`READ ONLY ACCESS`

An attempt was made to modify a read-only object. Either the target host system does not support write access to a normally writable MIB object, or the community name in the TCPCON-generated request message does not match the read/write community name established at the target system.

`MEMORY SHORTAGE`

An attempt to allocate memory from the NetWare operating system failed. This can occur during heavy network load or when multiple NLM files are running concurrently.

`MSG BUILD FAILURE`

An attempt to build SNMP request message failed. This is a diagnostic message that reports an error condition that should never occur.

`MSG SEND TRUNCATED`

The UDP transport truncated a TCPCON-generated request message. This is a diagnostic message that reports an error condition that should never occur.

`MSG ID MISMATCH`

The SNMP ID field of the response message from the target host system does not match the request message. This error indicates a problem in the target host system.

`MSG NAME MISMATCH`

The SNMP community name field of the response message from the target host system does not match the request message.

`MSG PARSE FAILURE`

The SNMP response message is corrupted.

`REPLY MSG TOO BIG`

The SNMP response message from the target host system is too long.

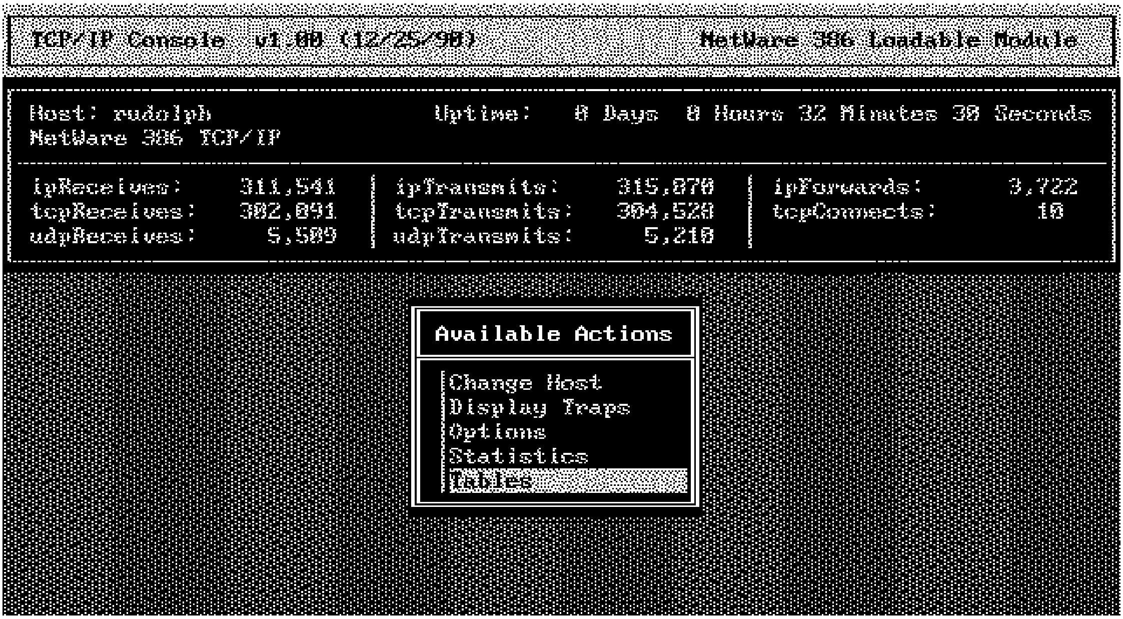
`UNKNOWN OBJECT`

The target host system does not support one of the Internet standard MIB objects.

```
BAD OBJECT VALUE
```

The SNMP response message from the target host system contain an object value that is invalid for the specified object type.

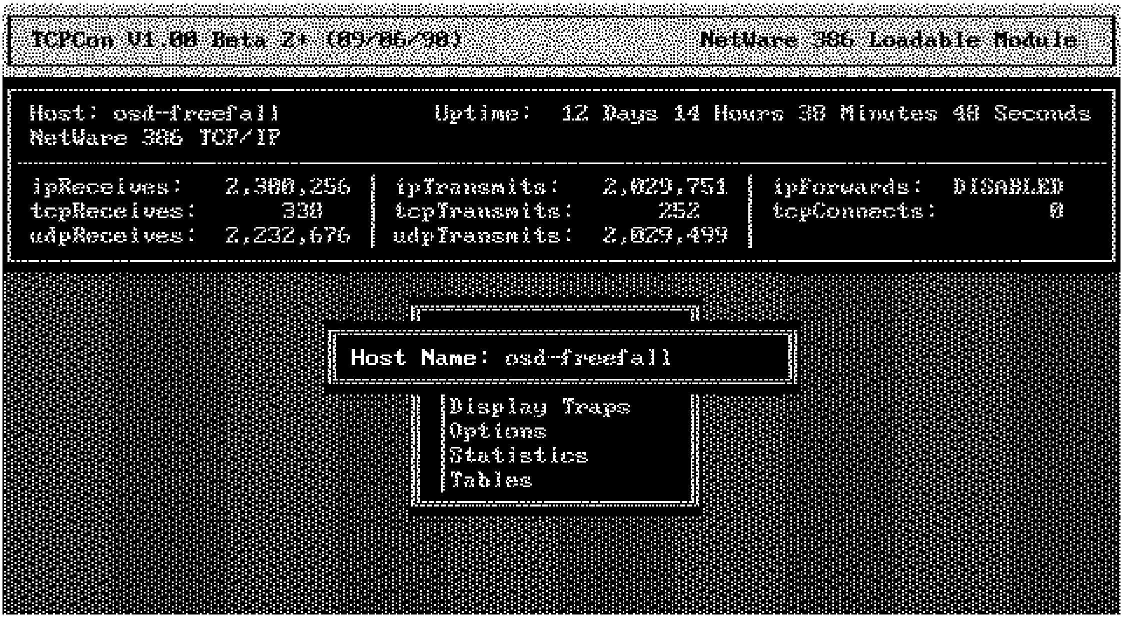
The TCPCON main screen also displays an Available Actions menu. This menu allows access to additional features of the utility as described in the following paragraphs.



```
TCP/IP Console  v1.00 (12/25/90)                    NetWare 386 Loadable Module

Host: rudolph                   Uptime:   0 Days  0 Hours 32 Minutes 30 Seconds
NetWare 386 TCP/IP
-------------------------------------------------------------------------------
ipReceives:     311,541 | ipTransmits:    315,070 | ipForwards:       3,722
tcpReceives:    302,091 | tcpTransmits:   304,520 | tcpConnects:         10
udpReceives:      5,509 | udpTransmits:     5,210 |
```

Change Host

Lets you select a new target host node by entering a new host name on the Host Name screen. The host specification supports either symbolic names or dotted notation (similar to the LOAD command line). Use the <Insert> key to display a list of host names contained in the SYS:ETC\HOSTS database.

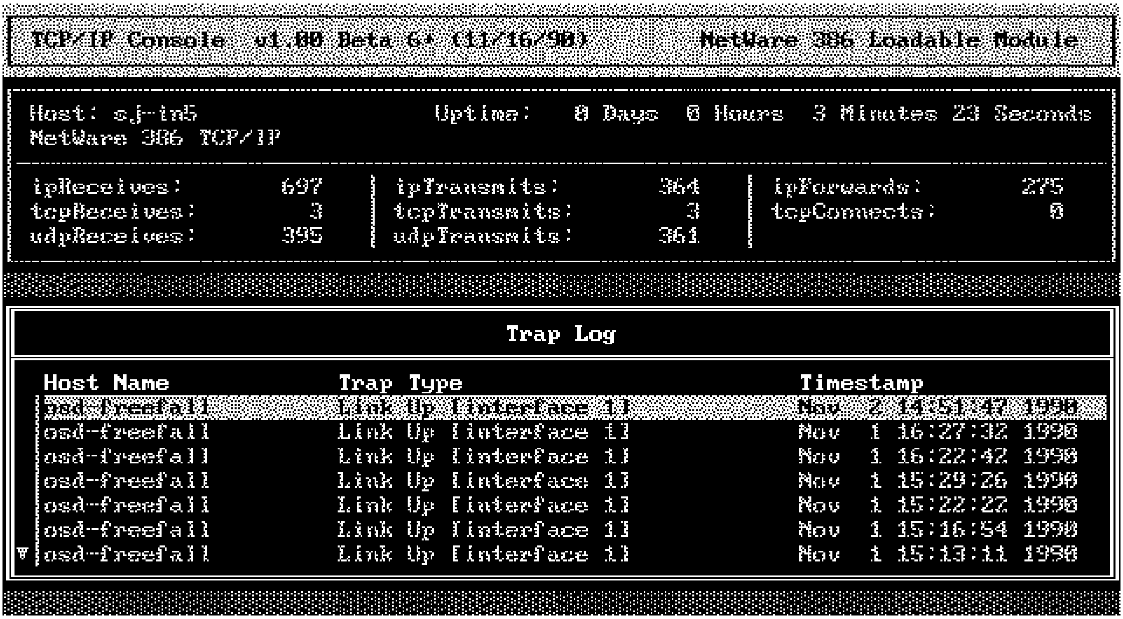


```
TCPCon V1.00 Beta 2+ (09/06/90)                    NetWare 386 Loadable Module

Host: osd-freefall           Uptime:  12 Days 14 Hours 30 Minutes 40 Seconds
NetWare 386 TCP/IP

ipReceives:   2,300,256 | ipTransmits:   2,029,751 | ipForwards:  DISABLED
tcpReceives:        330 | tcpTransmits:        252 | tcpConnects:        0
udpReceives:  2,232,676 | udpTransmits:  2,029,499 |

Host Name: osd-freefall
```

For more information about the HOSTS file, refer to Chapter 3, "Managing Network Database Files." For more information about IP addresses and dotted notation, refer to Appendix B, "TCP/IP Protocol Suite."

Display Traps

Allows access to the SNMP trap log. SNMP trap messages report significant TCP/IP events. The SNMPLOG NLM processes trap messages sent to the local TCP/IP node and writes them to disk. The Trap Log display presents those trap messages, with the most recent traps listed first. When you exit the Trap Log display, the system prompts you to either delete or keep the trap log file.



```
TCP/IP Console  v1.00 Beta 6* (11/16/90)          NetWare 386 Loadable Module

Host: sj-in5                     Uptime:   0 Days  0 Hours  3 Minutes 23 Seconds
NetWare 386 TCP/IP

ipReceives:        697  | ipTransmits:        364  | ipForwards:        275
tcpReceives:         3  | tcpTransmits:         3  | tcpConnects:         0
udpReceives:       395  | udpTransmits:       361  |

                                  Trap Log

  Host Name             Trap Type                        Timestamp
 osd-freefall           Link Up [interface 1]            Nov  2 14:51:47 1990
 osd-freefall           Link Up [interface 1]            Nov  1 16:27:32 1990
 osd-freefall           Link Up [interface 1]            Nov  1 16:22:42 1990
 osd-freefall           Link Up [interface 1]            Nov  1 15:29:26 1990
 osd-freefall           Link Up [interface 1]            Nov  1 15:22:27 1990
 osd-freefall           Link Up [interface 1]            Nov  1 15:16:54 1990
▼osd-freefall           Link Up [interface 1]            Nov  1 15:13:11 1990
```

The following fields are displayed for each trap message:

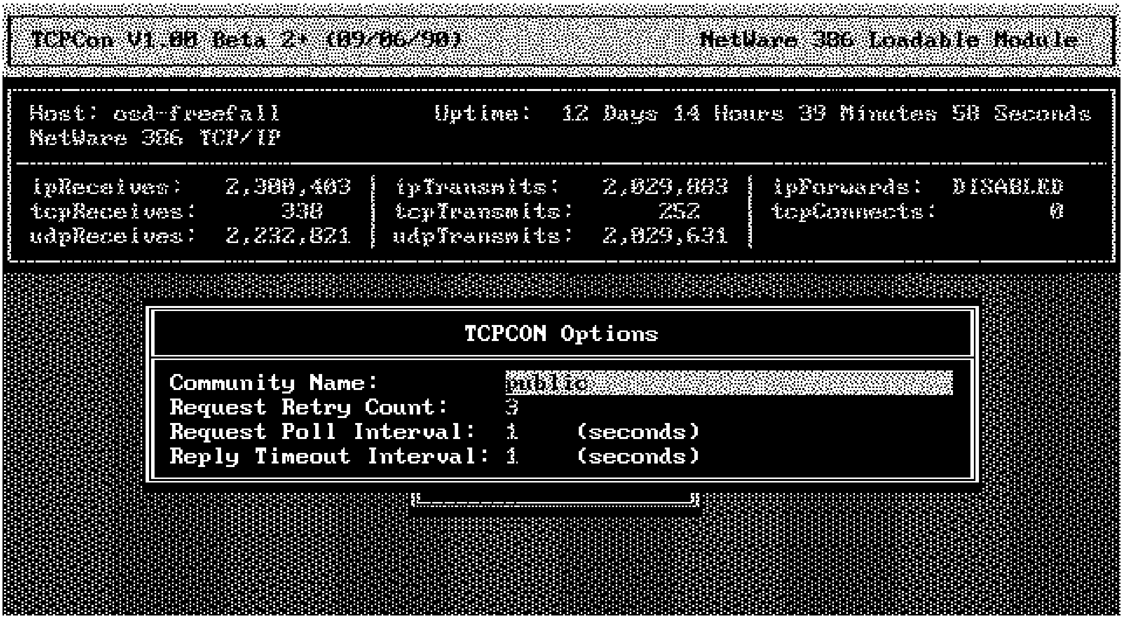
- Host Name. Indicates the host name or IP address of the TCP/IP node that issued each trap message. Symbolic host names are in the SYS:ETC\HOSTS file. IP addresses are displayed in dotted notation. The tab key alternates the display between symbolic name and the IP address.
- Trap Type. Indicates the event type triggering each trap message. Six types of trap messages are defined to report the following TCP/IP conditions: coldstart, warmstart, link down, link up, request authentication failure, and routing neighbor loss. A seventh type of trap messages is used to report enterprise- (vendor-) specific events. Enterprise-specific event codes are displayed following the trap type indication. They have the following format:

```
Enterprise Specific [<decimal number>]
```

- Timestamp. Indicates the date and time each trap message was received and processed by the local TCP/IP node.

Options

The TCPCON Options screen allows access to optional parameters controlling the operation of TCPCON. The parameters are also accessible from the LOAD command line, as previously described in this section.



```
TCPCon V1.00 Beta 2* (09/06/90)                    NetWare 386 Loadable Module

Host: osd-freefall            Uptime:  12 Days 14 Hours 39 Minutes 58 Seconds
NetWare 386 TCP/IP

ipReceives:    2,300,403 | ipTransmits:    2,029,003 | ipForwards:  DISABLED
tcpReceives:         338 | tcpTransmits:         252 | tcpConnects:        0
udpReceives:   2,232,821 | udpTransmits:   2,029,631 |

                            TCPCON Options

        Community Name:           public
        Request Retry Count:      3
        Request Poll Interval:    1    (seconds)
        Reply Timeout Interval:   1    (seconds)
```

The Options screen allows the community name to be changed from the default value of public. All statistics object displays are dynamically updated at a configurable interval (Request Poll Interval), also included in the Options screen.

The community name must match the name established for the SNMP Agent at the selected target host. A "MonitorCommunity" (read access) name mismatch causes the remote SNMP Agent at the target host to discard TCPCON read request messages. When this occurs, the summary display shows "UNKNOWN" in all statistics fields and HOST UNAVAILABLE in the lower-right corner.

Statistics

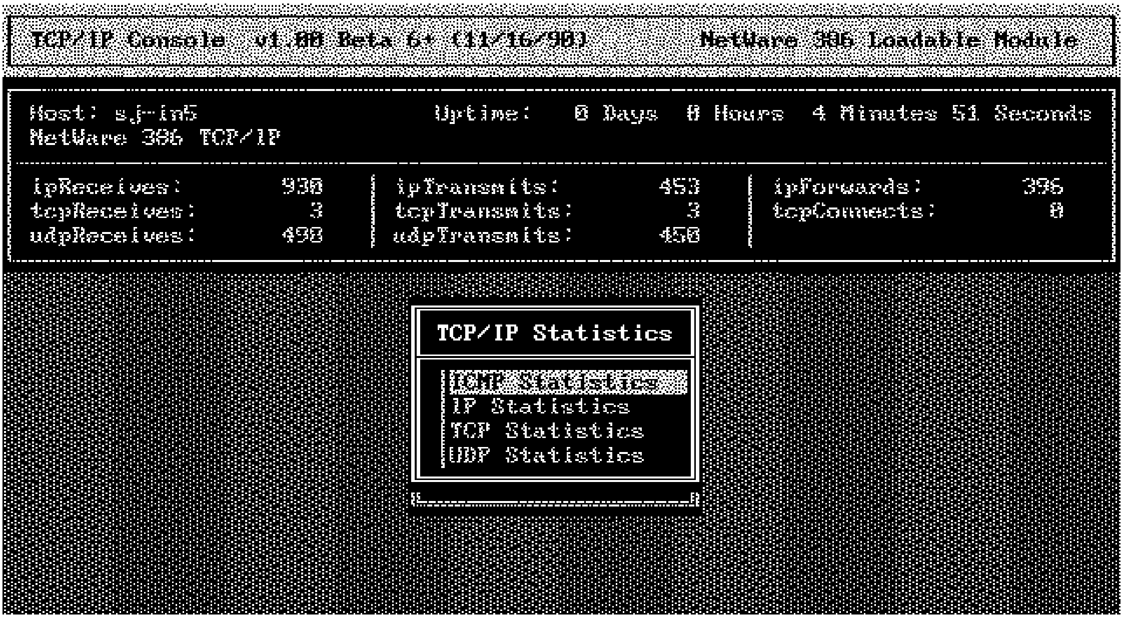
Allows access to the TCP/IP MIB statistics group displays. These displays, described in a later section, provide detailed information about the operation of the major protocols within the TCP/IP protocol stack.

Tables

Allows access to the TCP/IP MIB information table displays. These displays, described in a later section, provide detailed information about your network's physical address translation, physical interfaces, IP addresses, IP routing, and TCP connections.

TCP/IP Statistics Screen

The TCP/IP Statistics screen displays a menu of the four types of statistics maintained by the TCP/IP protocol stack.



```
TCP/IP Console  v1.00 Beta 6+ (11/16/90)        NetWare 386 Loadable Module

Host: sj-in5                    Uptime:   0 Days  0 Hours  4 Minutes 51 Seconds
NetWare 386 TCP/IP

ipReceives:         930 | ipTransmits:        453 | ipForwards:         396
tcpReceives:          3 | tcpTransmits:         3 | tcpConnects:          0
udpReceives:        498 | udpTransmits:       450 |

                          TCP/IP Statistics

                          ICMP Statistics
                          IP Statistics
                          TCP Statistics
                          UDP Statistics
```

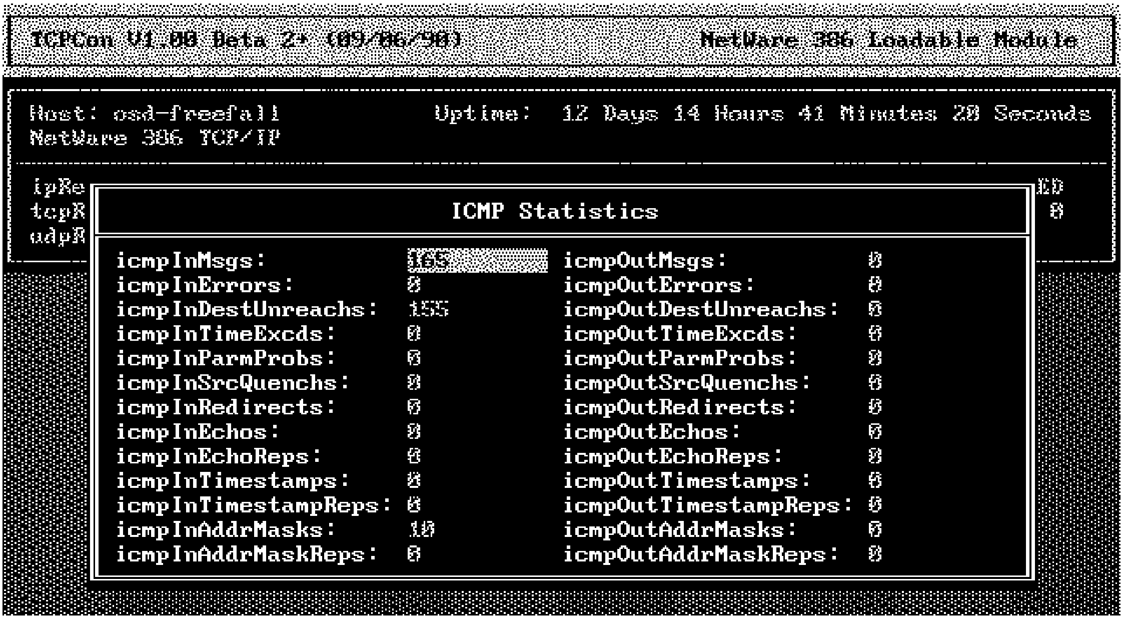
Selecting a menu item displays all statistics of that protocol type. Most statistical objects are read-only, and are identified by a colon (:) following the object field name. Read/write fields (for example, ipForwarding from the IP Statistics screen) are identified by a colon-period (:.) combination.

Read or write access to a statistics object may require redefinition of the TCPCON SNMP community name. The community name can be changed in the Options screen or on the TCPCON LOAD command line, as described earlier in this section.

Online help is available for each type of statistics display, as well as for individual statistics objects. Press <F1> to access general help. To access help for a specific statistical object, first select the object using the <Enter> key, then press <F1> to display the help screen.

ICMP Statistics

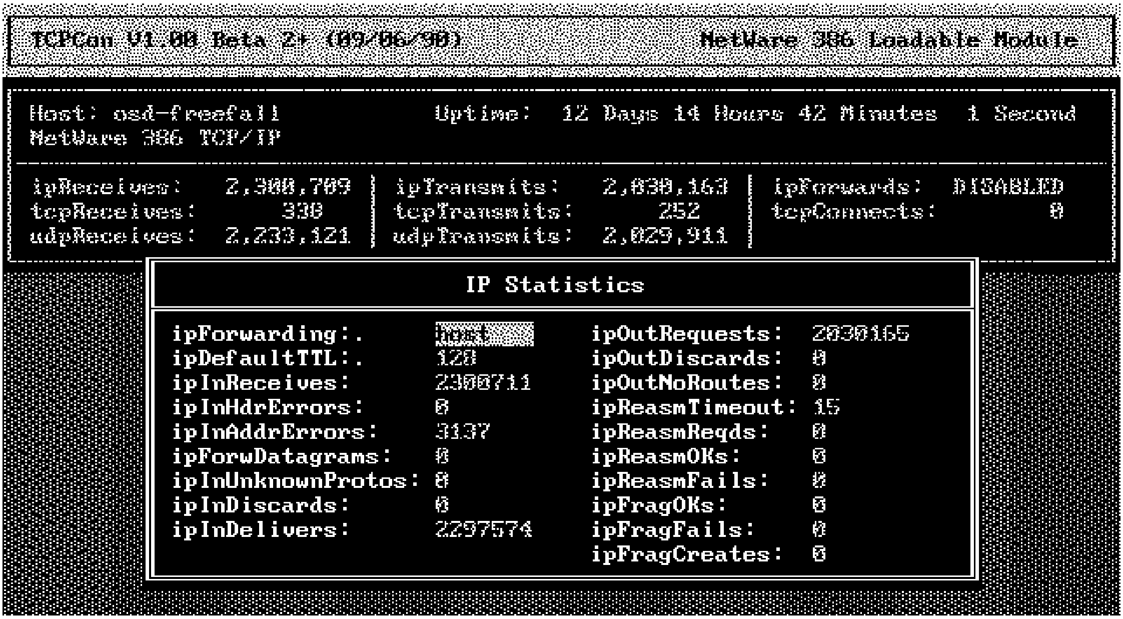
This display presents the Internet Control Message Protocol (ICMP) input and output statistics. All objects in this group are read-only.



```
TCPCon V1.00 Beta 2* (09/06/90)                NetWare 386 Loadable Module

Host: osd-freefall         Uptime:  12 Days 14 Hours 41 Minutes 20 Seconds
NetWare 386 TCP/IP

ipRe                                                                    ED
tcpR                          ICMP Statistics                            0
udpR
       icmpInMsgs:           165         icmpOutMsgs:           0
       icmpInErrors:         0           icmpOutErrors:         0
       icmpInDestUnreachs:   155         icmpOutDestUnreachs:   0
       icmpInTimeExcds:      0           icmpOutTimeExcds:      0
       icmpInParmProbs:      0           icmpOutParmProbs:      0
       icmpInSrcQuenchs:     0           icmpOutSrcQuenchs:     0
       icmpInRedirects:      0           icmpOutRedirects:      0
       icmpInEchos:          0           icmpOutEchos:          0
       icmpInEchoReps:       0           icmpOutEchoReps:       0
       icmpInTimestamps:     0           icmpOutTimestamps:     0
       icmpInTimestampReps:  0           icmpOutTimestampReps:  0
       icmpInAddrMasks:      10          icmpOutAddrMasks:      0
       icmpInAddrMaskReps:   0           icmpOutAddrMaskReps:   0
```

IP Statistics

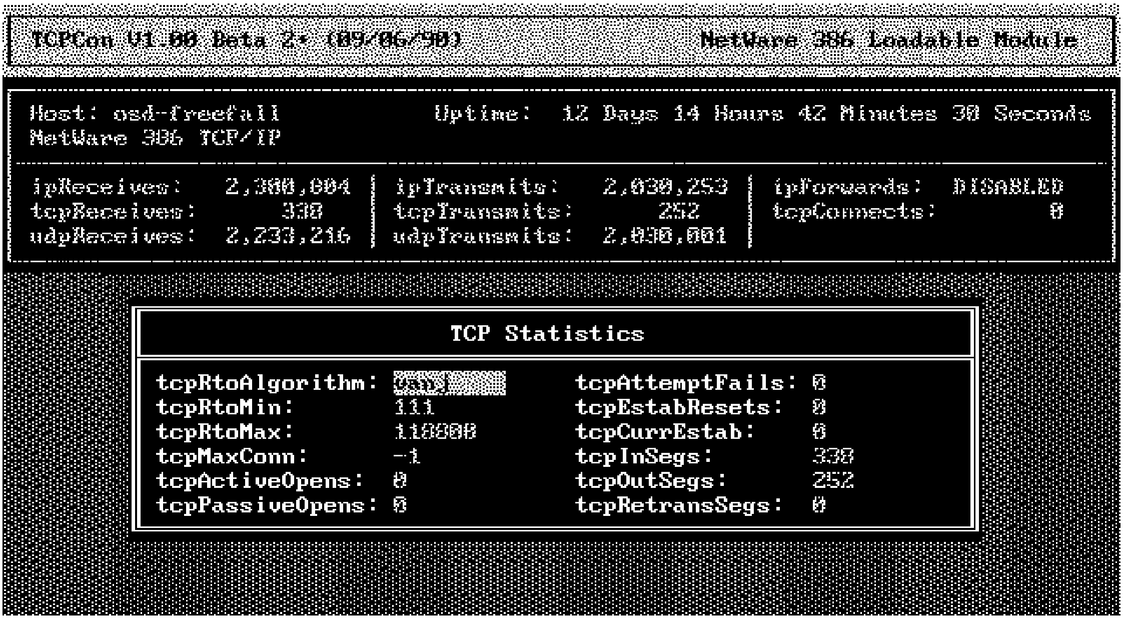
This display presents the Internet Protocol (IP) input and output statistics. It also contains the IP Forwarding and IP Default Time To Live (TTL) parameters. All statistics objects are read-only, while the Forwarding and Default TTL parameters are read/write.



```
TCPCon V1.00 Beta 2+ (09/06/90)                    NetWare 386 Loadable Module

Host: osd-freefall            Uptime:  12 Days 14 Hours 42 Minutes  1 Second
NetWare 386 TCP/IP

ipReceives:   2,300,709 | ipTransmits:   2,030,163 | ipForwards:  DISABLED
tcpReceives:        330 | tcpTransmits:        252 | tcpConnects:        0
udpReceives:  2,233,121 | udpTransmits:  2,029,911 |

                              IP Statistics

   ipForwarding:.      host        ipOutRequests:   2030165
   ipDefaultTTL:.      128         ipOutDiscards:   0
   ipInReceives:       2300711     ipOutNoRoutes:   0
   ipInHdrErrors:      0           ipReasmTimeout:  15
   ipInAddrErrors:     3137        ipReasmReqds:    0
   ipForwDatagrams:    0           ipReasmOKs:      0
   ipInUnknownProtos:  0           ipReasmFails:    0
   ipInDiscards:       0           ipFragOKs:       0
   ipInDelivers:       2297574     ipFragFails:     0
                                   ipFragCreates:   0
```

TCP Statistics

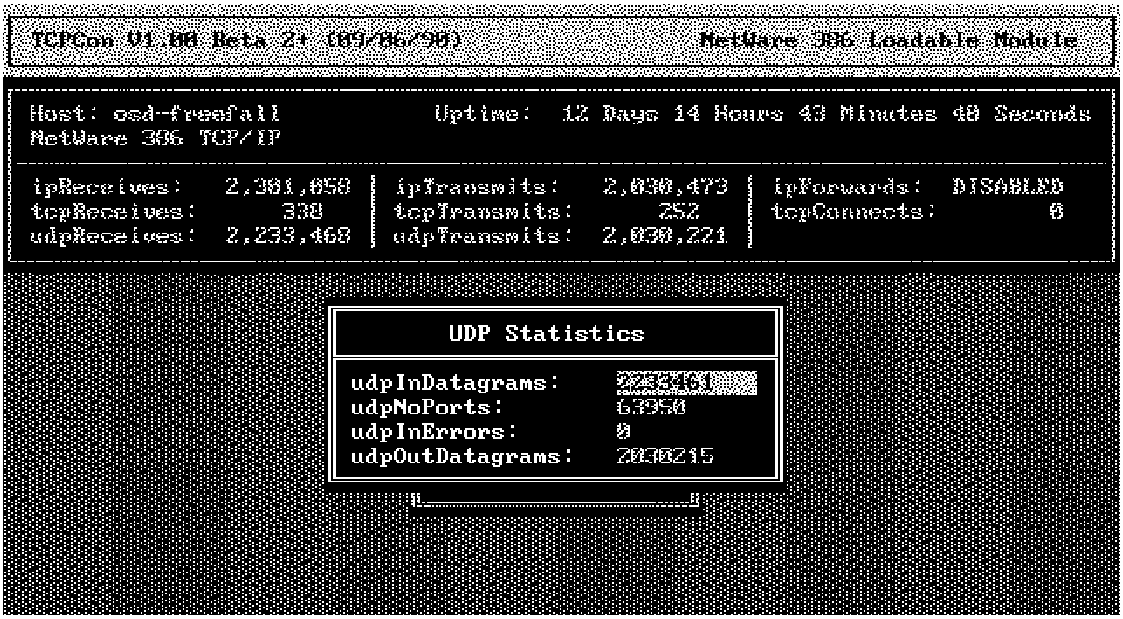
This display presents the TCP input and output statistics. All statistics objects are read-only.



```
TCPCon V1.00 Beta 2* (09/06/90)                    NetWare 386 Loadable Module

Host: osd-freefall              Uptime:  12 Days 14 Hours 42 Minutes 30 Seconds
NetWare 386 TCP/IP

ipReceives:    2,300,004 | ipTransmits:    2,030,253 | ipForwards:  DISABLED
tcpReceives:         330 | tcpTransmits:         252 | tcpConnects:        0
udpReceives:   2,233,216 | udpTransmits:   2,030,001 |

                            TCP Statistics

         tcpRtoAlgorithm: vanj          tcpAttemptFails: 0
         tcpRtoMin:       111           tcpEstabResets:  0
         tcpRtoMax:       110000        tcpCurrEstab:    0
         tcpMaxConn:      -1            tcpInSegs:       330
         tcpActiveOpens:  0             tcpOutSegs:      252
         tcpPassiveOpens: 0             tcpRetransSegs:  0
```

UDP Statistics

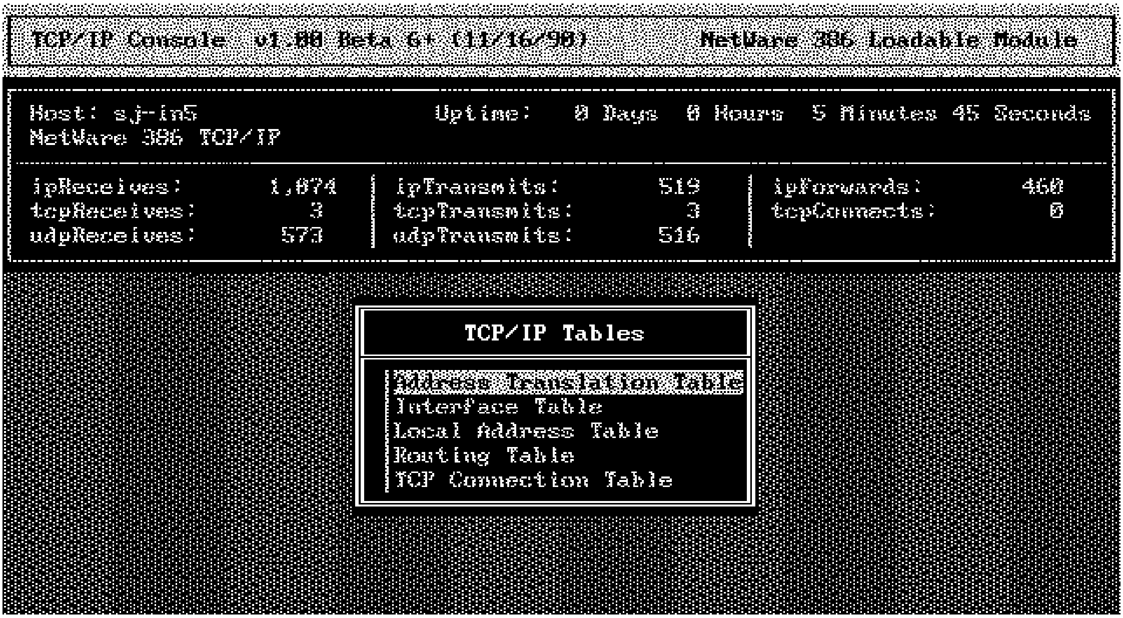
This display presents the User Datagram Protocol (UDP) input and output statistics. All statistics objects are read-only.



```
TCPCon V1.00 Beta 2+ (09/06/90)                    NetWare 386 Loadable Module

Host: osd-freefall              Uptime:  12 Days 14 Hours 43 Minutes 48 Seconds
NetWare 386 TCP/IP

ipReceives:   2,301,050 | ipTransmits:   2,030,473 | ipForwards:  DISABLED
tcpReceives:        330 | tcpTransmits:        252 | tcpConnects:        0
udpReceives:  2,233,468 | udpTransmits:  2,030,221 |

                        UDP Statistics

              udpInDatagrams:      2233461
              udpNoPorts:          63950
              udpInErrors:         0
              udpOutDatagrams:     2030215
```

TCP/IP Tables Screen

The TCP/IP Tables screen displays a menu of table types maintained by the TCP/IP protocol stack:



```
TCP/IP Console  v1.00 Beta 6+ (11/16/90)        NetWare 386 Loadable Module

Host: sj-in5                    Uptime:   0 Days  0 Hours  5 Minutes 45 Seconds
NetWare 386 TCP/IP

ipReceives:       1,074  | ipTransmits:        519  | ipForwards:        460
tcpReceives:          3  | tcpTransmits:         3  | tcpConnects:         0
udpReceives:        573  | udpTransmits:       516  |
```

You can select a menu item to display all entries for that table type. You can scroll through a table with the Up and Down arrows or the PageUp and PageDown keys. Table entries can be selected to allow a more detailed display of the desired entry. Read-only table entry fields are identified by a colon (:) following the field name. Read/write fields are identified by a colon-period (:.) combination.

Read or write access to table fields may require redefinition of the TCPCON SNMP community name. The community name can be changed in the Options screen or on the TCPCON LOAD command line, as described earlier in this section.

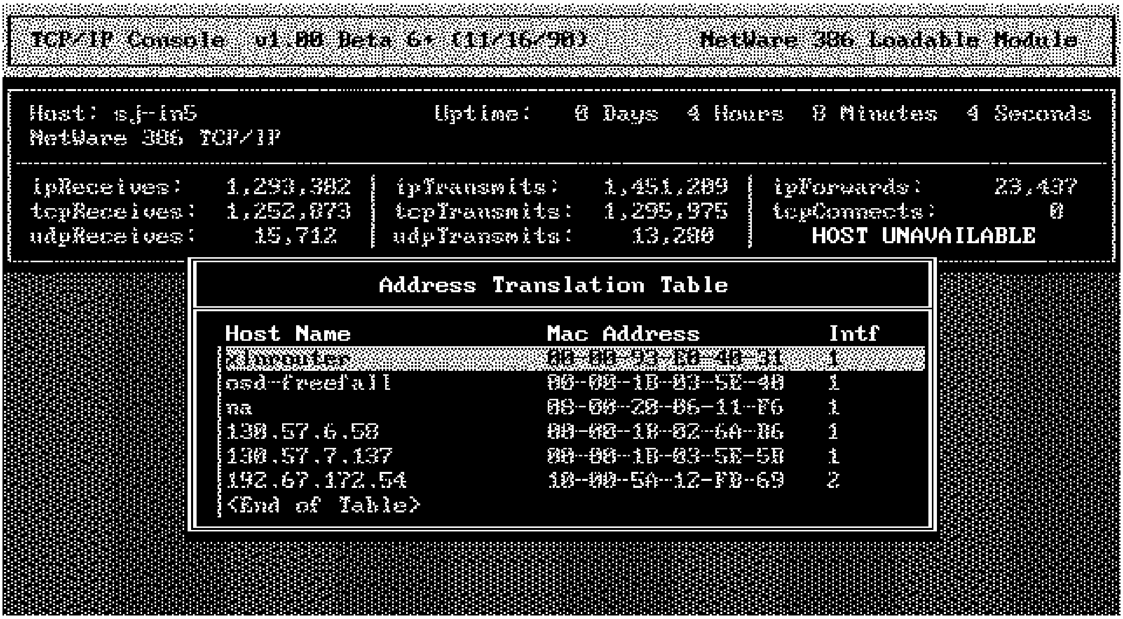
All table displays are static, representing a single snapshot of the table contents at the time the table was selected from the TCP/IP Table screen menu. To update a table display, press <Esc> for the TCP/IP Tables screen menu, then reselect the desired table.

Online help is available for each type of table display, as well as for individual table entries. Press <F1> to access help for a table type. To access help for a specific table entry field, first select the table entry using the <Enter> key, select the table entry field using the <Enter> key, then press <F1> to display the help screen.

Address Translation Table

The Address Translation Table contains the IP address to physical address equivalencies. For each IP address, a physical address and interface number are maintained. TCP/IP nodes use the Address Resolution Protocol (ARP) to obtain address translation information.

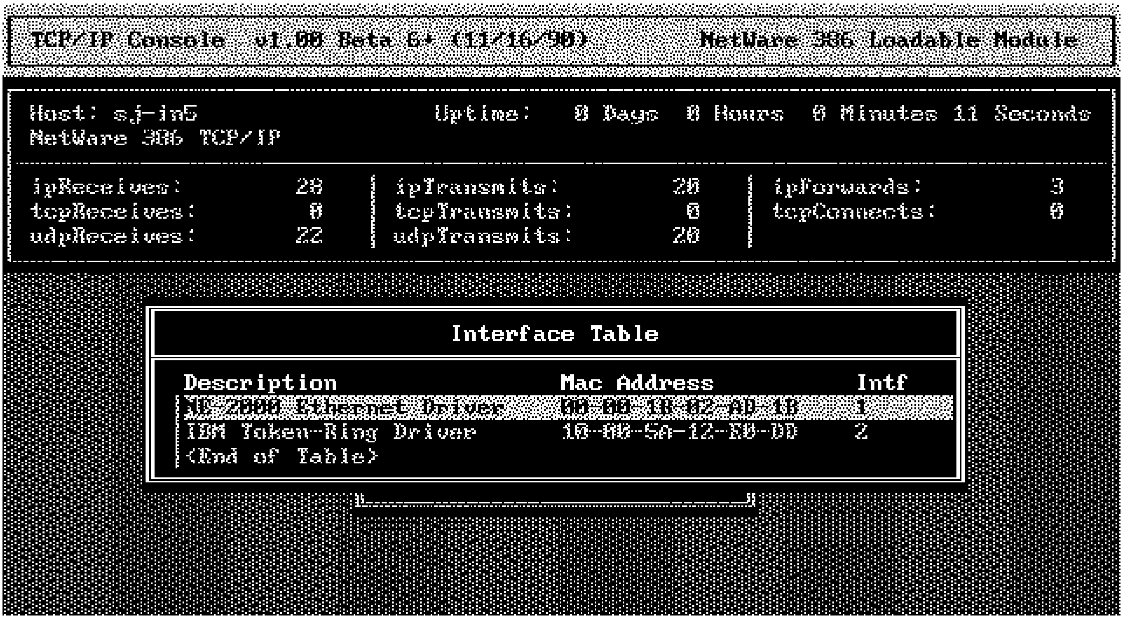
The <Insert and <Delete> keys allow creation and deletion of table entries. The <Enter> key allows modification of an existing table entry. The <Tab> key toggles the Host Name field between symbolic host name and dotted numeric address representation. Symbolic host names are in the SYS:ETC\HOSTS file.



```
TCP/IP Console  v1.00 Beta 6+ (11/16/90)          NetWare 386 Loadable Module

Host: sj-in5                      Uptime:   0 Days  4 Hours  8 Minutes  4 Seconds
NetWare 386 TCP/IP

ipReceives:    1,293,382 | ipTransmits:    1,451,209 | ipForwards:      23,437
tcpReceives:   1,252,873 | tcpTransmits:   1,295,975 | tcpConnects:          0
udpReceives:      15,712 | udpTransmits:      13,200 |    HOST UNAVAILABLE

                     Address Translation Table

         Host Name                   Mac Address          Intf
         xlnrouter                   00-00-93-B0-40-31    1
         osd-freefall                00-00-1B-03-5E-40    1
         na                          08-00-20-06-11-F6    1
         130.57.6.50                 00-00-1B-02-6A-B6    1
         130.57.7.137                00-00-1B-03-5E-5B    1
         192.67.172.54               10-00-5A-12-FB-69    2
         <End of Table>
```

Interface Table

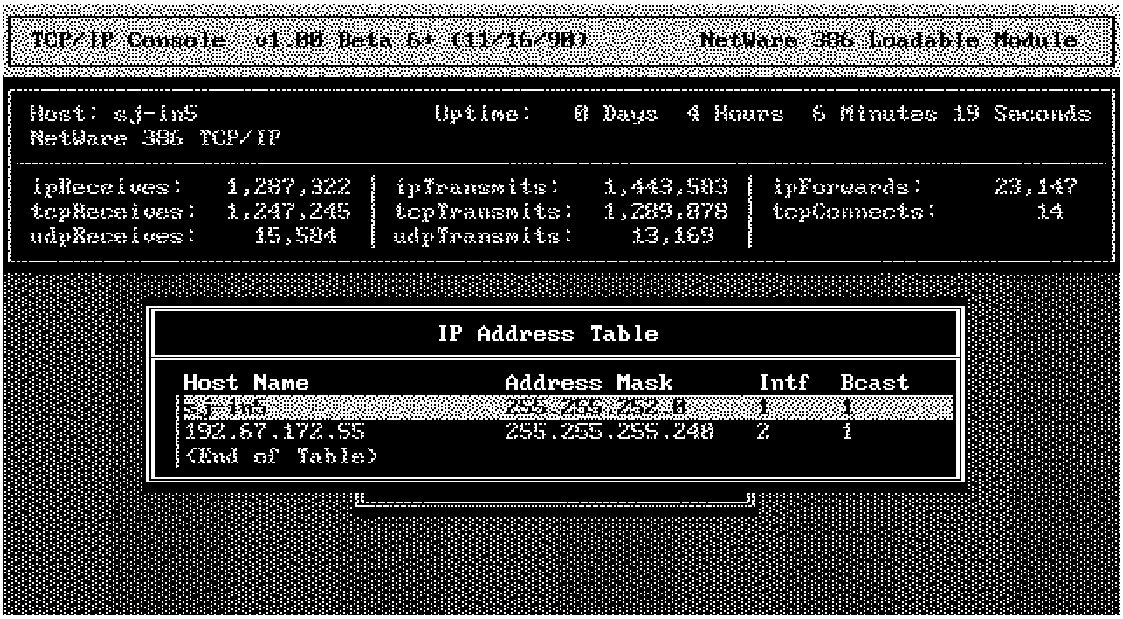
The Interface Table displays the network interfaces supported by the selected TCP/IP host system. Each entry contains a description of the interface type, the interface number, and the physical address. The <Enter> key allows examination of statistics objects associated with the highlighted interface. The Interface Table is read-only.



```
TCP/IP Console  v1.00 Beta 6+ (11/16/90)        NetWare 386 Loadable Module

Host: sj-in5                    Uptime:   0 Days  0 Hours  0 Minutes 11 Seconds
NetWare 386 TCP/IP

ipReceives:         20  | ipTransmits:        20  | ipForwards:          3
tcpReceives:         0  | tcpTransmits:        0  | tcpConnects:         0
udpReceives:        22  | udpTransmits:       20  |

                         Interface Table

   Description                  Mac Address          Intf
   NE-2000 Ethernet Driver      00-00-1B-02-AD-1B    1
   IBM Token-Ring Driver        10-00-5A-12-E0-DD    2
   <End of Table>
```

Local IP Address Table

The Local IP Address Table displays the local IP address(es) of the selected TCP/IP node. Each entry contains a host name or IP address, an address mask in dotted notation, the network interface number of the interface supporting the IP address, and an indication of the type of IP broadcast address supported by the node. The <Tab> key toggles the Host Name field between symbolic host name and dotted numeric address representation. All table objects are read-only.



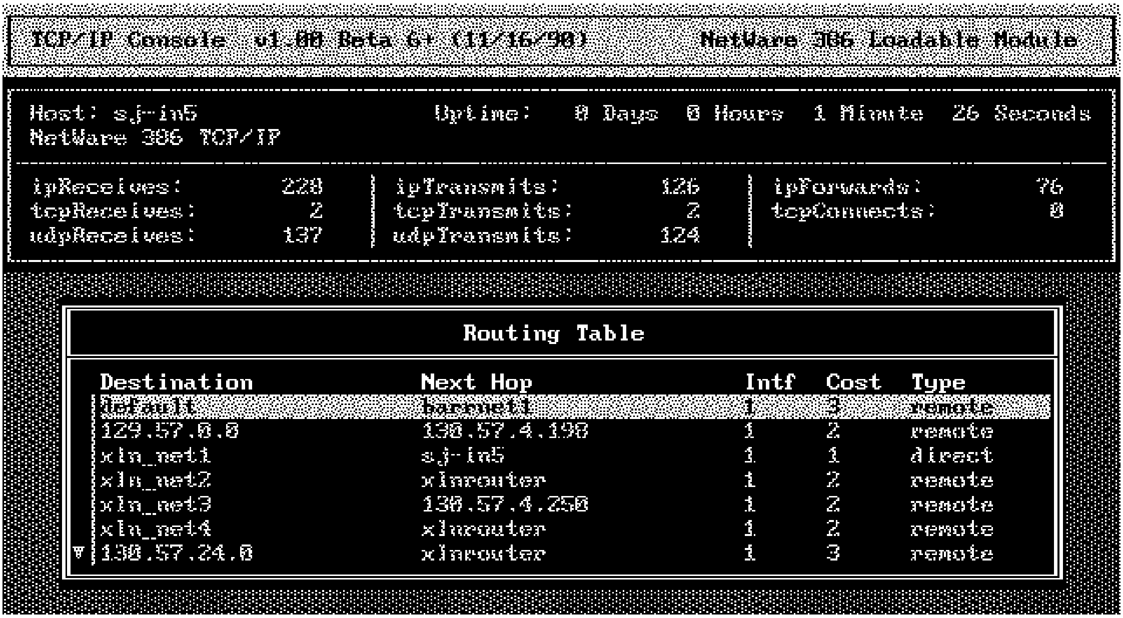
```
TCP/IP Console  v1.00 Beta 6+ (11/16/98)        NetWare 386 Loadable Module

Host: sj-in5                    Uptime:   0 Days  4 Hours  6 Minutes 19 Seconds
NetWare 386 TCP/IP

ipReceives:    1,287,322 | ipTransmits:   1,443,583 | ipForwards:      23,147
tcpReceives:  1,247,245 | tcpTransmits:  1,289,878 | tcpConnects:         14
udpReceives:     15,584 | udpTransmits:     13,169 |

                              IP Address Table

          Host Name                   Address Mask      Intf  Bcast
          sj-in5                      255.255.252.0     1     1
          192.67.172.55               255.255.255.240   2     1
          <End of Table>
```

Routing Table

The TCP/IP Console Routing Table displays an entry for each route known to the selected TCP/IP host system. Each entry contains the destination network name or IP address, the host name or IP address of the next hop, the network interface number associated with the next hop, the relative cost of the route, and the type of route.

Destination network names are in the SYS:\ETC\NETWORKS file. Next hop host names are in the SYS:ETC\HOSTS file. The <Tab> key toggles between symbolic name and dotted numeric address representation of the destination and next hop fields. The <Insert> and <Delete> keys allow creation and deletion of route entries. The <Enter> key allows examination and modification of additional information associated with the highlighted route entry.

The Routing Table screen displays the following fields for each table entry:



```
TCP/IP Console  v1.00 Beta 6+ (11/16/90)          NetWare 386 Loadable Module

Host: sj-in5                   Uptime:   0 Days  0 Hours  1 Minute  26 Seconds
NetWare 386 TCP/IP

ipReceives:        228  | ipTransmits:        126  | ipForwards:         76
tcpReceives:         2  | tcpTransmits:         2  | tcpConnects:         0
udpReceives:       137  | udpTransmits:       124  |

                              Routing Table

   Destination              Next Hop                 Intf  Cost  Type
  |default                  harnet1                  1     3     remote
  |129.57.0.0               130.57.4.190             1     2     remote
  |xln_net1                 sj-in5                   1     1     direct
  |xln_net2                 xlnrouter                1     2     remote
  |xln_net3                 130.57.4.250             1     2     remote
  |xln_net4                 xlnrouter                1     2     remote
 ▼|130.57.24.0              xlnrouter                1     3     remote
```

Destination. Specifies the name or IP address of the target network or host to which the route leads.

Next Hop. Specifies the name or IP address of the first gateway of the route. The local node sends all traffic to the destination through the next hop gateway.

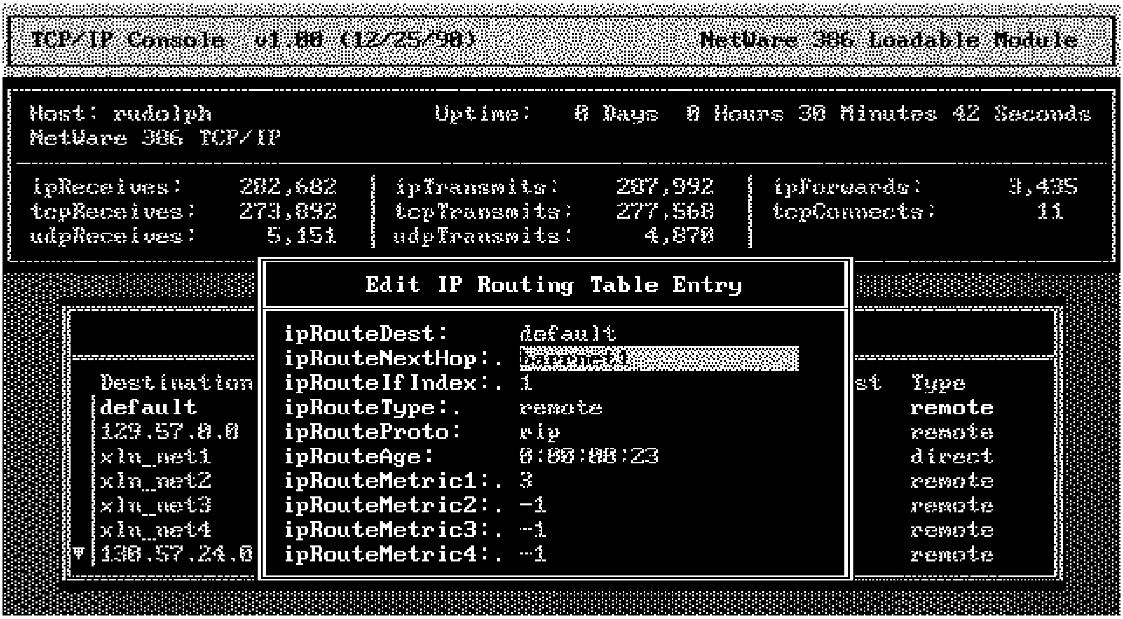
Intf. Specifies the number of interfaces through which the next hop gateway is reachable. It represents the locally attached network on which the next hop gateway is a neighbor.

Cost. Specifies the measure of expense associated with the route. It is usually the number of hops required within the route to get to the destination.

Type. Specifies whether the route is remote or direct. The remote route describes the route to get to a remote host or network. The direct route describes the route to a network to which the router is directly attached. The direct routes are permanent entries that are entered only by IP during initialization. The type field also specifies whether the route is invalid. Since only remote routes can become invalidated, all routes of invalid type are remote routes.

The TCPCON user can display and edit parameters of a particular route entry by placing the cursor on the desired entry and pressing the <Enter> key.

The following screen is an example of the Edit IP Routing Table Entry:



```
 TCP/IP Console  v1.00 (12/25/90)                    NetWare 386 Loadable Module

 Host: rudolph                 Uptime:   0 Days  0 Hours 30 Minutes 42 Seconds
 NetWare 386 TCP/IP

 ipReceives:     282,682 | ipTransmits:     287,992 | ipForwards:        3,435
 tcpReceives:    273,892 | tcpTransmits:    277,568 | tcpConnects:          11
 udpReceives:      5,151 | udpTransmits:      4,870 |

                      Edit IP Routing Table Entry

   Destination  ipRouteDest:      default
  default       ipRouteNextHop:.  barrnet1                   st  Type
  129.57.0.0    ipRouteIfIndex:.  1                              remote
  xln_net1      ipRouteType:.     remote                         remote
  xln_net2      ipRouteProto:     rip                            direct
  xln_net3      ipRouteAge:       0:00:00:23                     remote
  xln_net4      ipRouteMetric1:.  3                              remote
 ▼130.57.24.0   ipRouteMetric2:.  -1                             remote
                ipRouteMetric3:.  -1                             remote
                ipRouteMetric4:.  -1
```

The ipRouteDest, ipRouteNextHop, and ipRouteIfIndex fields correspond to the first three fields in the Routing Table display (Destination, Next Hop, and Intf). The ipRouteType field corresponds to the type field in the routing table display.

ipRouteProto. Specifies the protocol through which the route entry was created. It can be either local, NETMGMT, ICMP, or RIP. Local signifies manual entries usually created by IP itself. Entries created by SNMP are assigned according to the NETMGMT protocol.

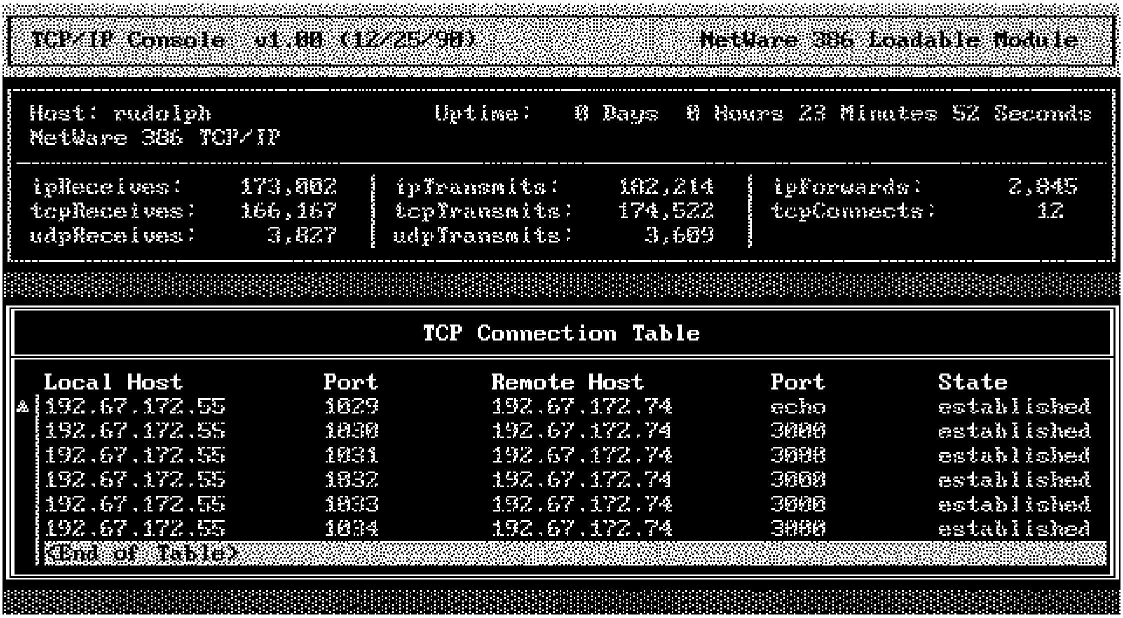
ipRouteAge. Specifies the time in seconds since a valid entry was updated, or since the entry has become invalidated if the entry is invalid.

ipRoutemetric1. Corresponds to the cost field in the routing table display. The ipRouteMetric2 through ipRouteMetric4 fields are not currently used.

The user can edit all the fields in the route entry except ipRouteProto and ipRouteAge. When a new entry is created through TCPCON, ipRouteProto is set to NETMGMT and ipRouteAge is initialized to 0.

TCP Connection Table

The TCP Connection Table displays an entry for each TCP connection maintained by the selected TCP/IP host system. Each entry contains the source host name or IP address, source port, destination host name or IP address, destination port, and the connection state. Symbolic host names are in the SYS:ETC\HOSTS file. The <Tab> key toggles between host name and dotted numeric address representation of the source and destination nodes. All TCP connection table fields are read-only.

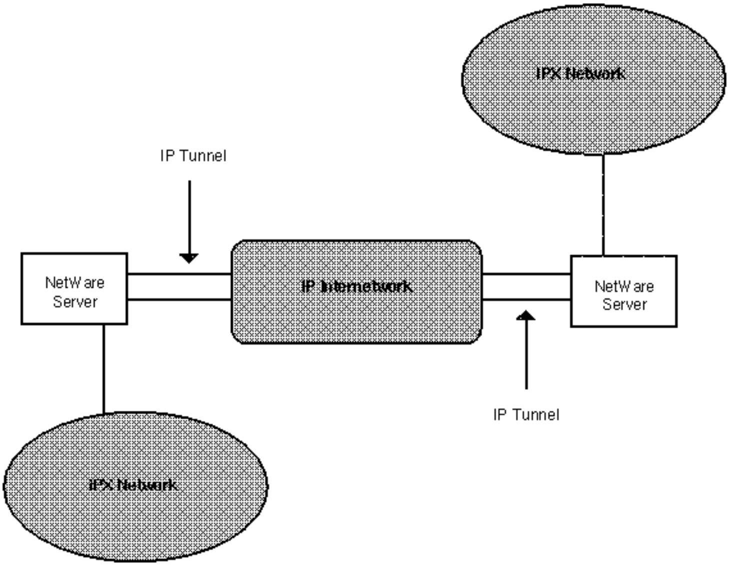


```
 TCP/IP Console  v1.00 (12/25/90)              NetWare 386 Loadable Module

 Host: rudolph                 Uptime:   0 Days  0 Hours 23 Minutes 52 Seconds
 NetWare 386 TCP/IP

 ipReceives:     173,002 | ipTransmits:     182,214 | ipForwards:       2,845
 tcpReceives:    166,167 | tcpTransmits:    174,522 | tcpConnects:         12
 udpReceives:      3,827 | udpTransmits:      3,609 |

                            TCP Connection Table

  Local Host            Port         Remote Host           Port         State
 192.67.172.55          1029         192.67.172.74         echo         established
 192.67.172.55          1030         192.67.172.74         3000         established
 192.67.172.55          1031         192.67.172.74         3000         established
 192.67.172.55          1032         192.67.172.74         3000         established
 192.67.172.55          1033         192.67.172.74         3000         established
 192.67.172.55          1034         192.67.172.74         3000         established
 <End of Table>
```

A IP Tunnel LAN Driver

The IP Tunnel LAN driver lets the NetWare Internetwork Packet Exchange (IPX) implementation use an IP internetwork to communicate with other IPX nodes. From the IPX protocol perspective, the IP Tunnel performs the same functions as the typical NetWare LAN driver. The IP network functions as if it were a hardware network, passing packets among the IPX nodes connected to it. Figure A-1 shows an IP Tunnel connecting two IPX networks.

Figure 8 IP Tunnel Connecting IPX Networks



IPX uses the Open Data-Link Interface (ODI) to pass packets through the IP Tunnel. The IP Tunnel sends each IPX packet across the IP internetwork by encapsulating it in a User Datagram Protocol (UDP) packet. The tunneling

LAN driver at the destination removes the UDP header from each arriving packet and passes it through the ODI to IPX.

The result is a straightforward method for connecting two or more IPX networks over an existing IP internetwork. To ensure data integrity, the IP Tunnel includes a UDP checksum in each packet that protects IPX packets from damage while traveling through the IP internetwork. Although data corruption is rare, it does occur occasionally in any internetwork.

The IP Tunnel can be used with the Schneider & Koch product SK-IPX/IP Gateway to connect NetWare nodes with NetWare 2.x nodes. (The SK-IPX/IP Gateway product provides the NetWare 2.x compatibility.)

The IP Tunnel also serves clients using either the Novell IP Tunnel client driver, a component of the LAN WorkPlace for DOS, or the Schneider & Koch DOS end-node product.

Operating with the IP Tunnel

The IP Tunnel presents a standard LAN driver interface to the NetWare system. You load it like any other LAN driver. You then bind IPX to it to instruct IPX to receive and route packets over the internetwork the tunnel is using.

The IP Tunnel handles normal IPX traffic like any other LAN driver. The IP internetwork is the *medium*. The IP address is the *immediate address*, performing the same function in the IP *medium* as the Media Access Control (MAC) address performs in the Ethernet medium.

To spread routing and service information between IPX routers, IPX depends on broadcasting some messages to every other NetWare server connected to the medium. Because broadcast facilities are limited in IP internets, the IP Tunnel must handle broadcast traffic differently.

When starting the IP Tunnel, you inform it of the internetwork IPX routers that you plan to communicate with. These other routers are known as *peers*. Whenever IPX broadcasts a packet, the IP Tunnel duplicates the packet and sends one copy directly to each peer.

It is recommended that you configure any connected group of peers so that all the servers in the group have the IP addresses of all other servers in the group. Other configurations are possible, but not advisable. They frequently create confusing and surprising IPX routing topologies.

LAN WorkPlace for DOS or SK-IPX/IP Gateway client PCs do not need to be in the peer list, because they do not need to see broadcast traffic on the medium. If the IP Tunnel server is the only IPX router connected to the internet, the *peer* parameter is not necessary.

Configuring a Server for the IP Tunnel

To configure a server to use the IP Tunnel, you must first load and bind TCP/IP to the interfaces you plan to use. The IP Tunnel requires local IP addressing information. The IP Tunnel can fail if TCP/IP has not been connected to a network. Refer to Configuring TCP/IP for automatic TCP/IP configuration procedures or Appendix D, "Manually Configured TCP/IP Examples," on page 125 for manually loading and binding TCP/IP.

LOAD IPTUNNEL Command Line

The LOAD IPTUNNEL command line has the following format:

```
LOAD IPTUNNEL [PEER=remote IP address] [CHKSUM={YES|NO}]
   [LOCAL=local IP address] [PORT=UDP port number]
   [SHOW={YES|NO}]<Enter>
```

If you must add more than one peer to a peer group, you can repeat the LOAD command to add other peers to the list.

LOAD IPTUNNEL Command Options

The LOAD IPTUNNEL command has five options:

- `chksum`
- `local`
- peer
- `port`
- `show`

These options let you disable UDP checksums, specify your local IP address, designate a peer server, set the UDP port, and display your current configuration. The `port` and `local` parameters can only be given in the first LOAD IPTUNNEL command. The `chksum` parameter can be placed on any LOAD IPTUNNEL command and overrides any previous setting of the checksum flag. The `show` parameter can be placed in any LOAD IPTUNNEL command.

The LOAD IPTUNNEL command parameters are summarized in Table A-1. The paragraphs that follow describe each parameter in detail.

Table 8 IPTUNNEL Command-line Parameters

Parameter	Values	Explanation
chksum	Yes or No	Enables or disables UDP checksums on transmitted packets. The default value is Yes.
local	IP address	Specifies a single local IP address for the IP tunnel. The default value is the IP address of the first interface TCP/IP was bound to.
peer	IP address	Adds an IP address to the peer list. If this parameter is not present, no peer is added.
port	Integer from 1 to 65535	Sets the UDP port the IP tunnel uses. The default value is port 213.
show	Yes or No	Requests configuration report. If you specify this parameter without a value, the default value is Yes.

chksum

Normally, the IP Tunnel puts a UDP checksum in each packet it transmits. Setting the *chksum* parameter to "No" disables UDP checksum calculations on all packets the IP Tunnel sends. Using UDP checksums improves data reliability; we recommend that you leave this parameter on.

local

For IPX routing to work correctly, the IP Tunnel must consistently use a single local IP address. If TCP/IP is bound to more than one interface, the *local* parameter lets you pick from among the local IP addresses. If this parameter is not present in the first LOAD IPTUNNEL command, the tunnel uses the IP address of the first interface to which TCP/IP was bound.

The IP Tunnel only uses the *local* parameter on the first LOAD command.

peer

The *peer* parameter is the most important parameter. With the `peer` parameter, you can add an additional IP address to a peer list. By repeating the LOAD IPTUNNEL command with different IP addresses given with the `peer` parameter, you can add as many peers as you want to the IP Tunnel's peer list. Remember, however, that whenever IPX broadcasts a packet, the IP Tunnel duplicates it and transmits it in a UDP packet to each peer in the list. For this reason it is best to keep the peer list small. It is recommend that you have no more than ten peers for any one node.

NOTE: If you have to add more than one peer to a peer group, you can repeat the LOAD IPTUNNEL command to add other peers to the list.

port

The *port* parameter sets the UDP port the IP Tunnel uses. If not present on the first LOAD IPTUNNEL command, the IP Tunnel uses UDP port 213, the officially assigned UDP port for IPX packets. Versions of the Schneider & Koch SK-IPX/IP, version 1.3 or later, products use UDP port 213. If you must communicate with nodes using those products prior to version 1.3, you can use port=59139. (Current versions of the Schneider & Koch SK-IPX/IP products default to UDP port 213.)

The IP Tunnel only uses the *port* parameter on the first LOAD command.

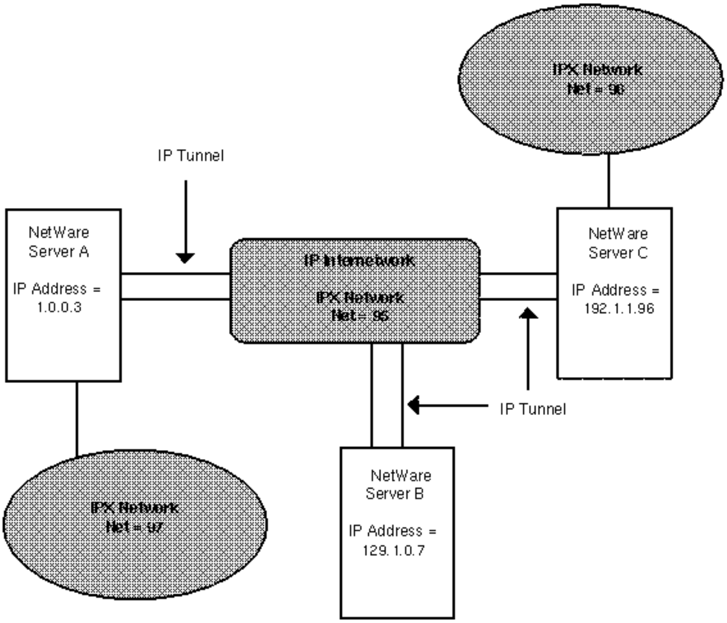
show

The *show* parameter controls display of the current configuration. Normally, a configuration report is not displayed. If you specify the show parameter without a value, the default is Yes. If `show`=yes, the IP Tunnel displays the local IP address, the port being used, the peer list, and whether checksums are enabled.

IP Tunnel Server Example

Figure A-2 shows how an IP Tunnel LAN driver lets NetWare IPX network 95 use an IP internetwork to communicate with other IPX nodes (servers A, B, and C). The IP internetwork can pass packets to or from any IPX nodes connected to it.

Figure 9 Sample IP Tunnel Server Configuration



The following commands are entered at Server A's console:

```
LOAD IPTUNNEL PEER=129.1.0.7 <Enter>
LOAD IPTUNNEL PEER=192.1.1.96 <Enter>
BIND IPX to IPTUNNEL  NET=95 <Enter>
```

The first two lines (the LOAD IPTUNNEL commands) load the IP Tunnel into the system and add peer entries on this node for IP addresses 129.1.0.7 and 192.1.1.96. The last line connects IPX to the IP Tunnel.

The following commands are entered at Server B's console:

```
LOAD IPTUNNEL PEER=1.0.0.3 <Enter>
LOAD IPTUNNEL PEER=192.1.1.96 <Enter>
BIND IPX to IPTUNNEL NET=95 <Enter>
```

The preceding commands are almost identical to the Server A configuration commands. These commands add peer entries for IP addresses 1.0.0.3 and 192.1.1.96.

The following commands are entered at Server C's console:

```
LOAD IPTUNNEL PEER=1.0.0.3 <Enter>
LOAD IPTUNNEL PEER=129.1.0.7 <Enter>
BIND IPX TO IPTUNNEL NET=95 <Enter>
```

The preceding commands are almost identical to the Server B configuration commands. These commands add peer entries for IP addresses 1.0.0.3 and 129.1.0.7. Once Servers A, B, and C are correctly configured, they are ready to send and receive IPX traffic.

Configuring DOS IP Tunnel for LAN WorkPlace Client

To use the DOS IP Tunnel on a LAN WorkPlace or third-party client system, you must first load and bind TCP/IP to the interfaces it uses. The typical loading sequence is as follows:

```
LSL
NE1000 (or appropriate driver)
TCPIP
IPTUNNEL
IPXODI
NET
```

The DOS IP Tunnel requires local IP addressing information when it comes up and fails if TCP/IP has not been loaded.

The DOS IP Tunnel utility is loaded during LAN WorkPlace for DOS installation. To start the DOS IP Tunnel, use the following command:

```
C:> IPTUNNEL <Enter>
```

DOS IP Tunnel is a terminate and stay resident utility (TSR) that can be removed from memory with the following command:

```
C:> IPTUNNEL U <Enter>
```

DOS IP Tunnel NET.CFG Parameters

The DOS IP Tunnel has the following configuration parameters in the NET.CFG file:

- gateway
- checksum
- port

These options let you specify the remote tunnel gateway, disable UDP checksums, and set the UDP port. Table A-2 provides a summary of each option.

Table 9 DOS IP Tunnel NET.CFG Parameters

Parameter	Values	Explanation
gateway	IP address	Lets you specify the remote tunnel gateway. The default value is the broadcast address 255.255.255.255.
checksum	Yes or No	Enables or disables UDP checksums on transmitted packets. The default value is Yes.
port	Integer from 1 to 65535	Sets the UDP port the tunnel uses. The default value is port 213.

gateway

The *gateway* parameter lets you specify the IP address of up to ten IP Tunnel gateway addresses. If you do not specify a gateway address, the default value is the broadcast address 255.255.255.255. Using the broadcast address enables the IPX client to find any number of IP Tunnel gateways on the local TCP/IP network.

Although a broadcast entry can be used to find all local gateways, a specific entry must be made for each remote gateway. This is required so that the DOS client can route packets correctly to any gateway when multiple servers are tunneling among themselves on an IP internetwork.

checksum

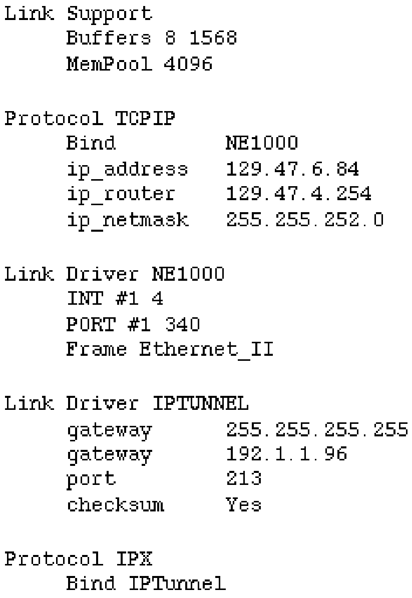
Normally, the IP Tunnel puts a UDP checksum in each packet it transmits. Setting the *checksum* parameter to No disables UDP checksum calculations on all packets the IP Tunnel sends. Using UDP checksums improves data reliability; it is recommended that you leave this parameter on.

port

The port parameter sets the UDP port the IP Tunnel uses. If not present in the NET.CFG file, the IP Tunnel uses UDP port 213, the officially assigned UDP port for IPX packets. Older versions of the Schneider & Koch SK-IPX/IP client and server products use UDP port 59139. If you must communicate with those products, you can use *port*=59139. (Current versions (version 1.3 or later) of Schneider & Koch's SK-IPX/IP products default to UDP port 213.)

Figure A-3 shows a sample NET.CFG file showing the DOS IP Tunnel parameters in the Link Driver IPTUNNEL section.

Figure 10 Sample NET.CFG File with DOS IPTUNNEL Parameters

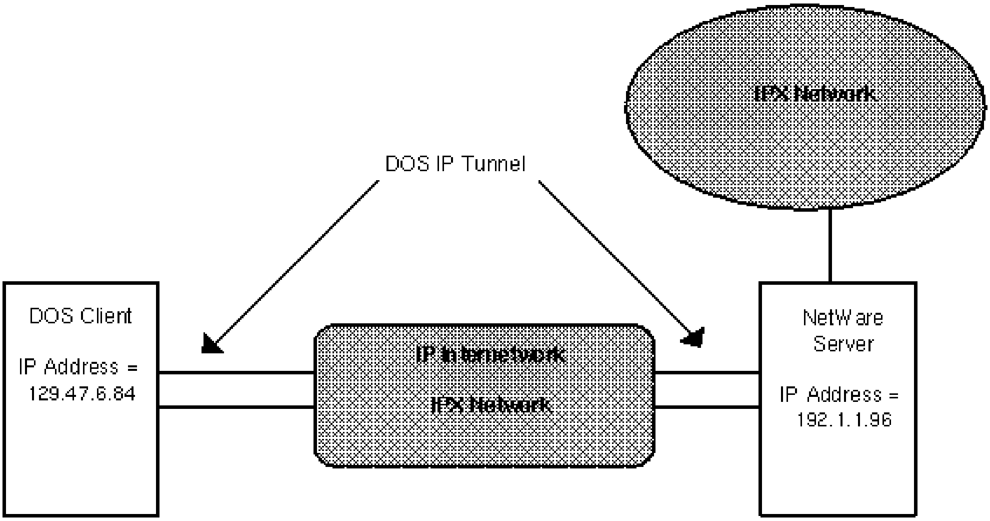


```
Link Support
     Buffers 8 1568
     MemPool 4096

Protocol TCPIP
     Bind         NE1000
     ip_address   129.47.6.84
     ip_router    129.47.4.254
     ip_netmask   255.255.252.0

Link Driver NE1000
     INT #1 4
     PORT #1 340
     Frame Ethernet_II

Link Driver IPTUNNEL
     gateway      255.255.255.255
     gateway      192.1.1.96
     port         213
     checksum     Yes

Protocol IPX
     Bind IPTunnel
```

DOS IP Tunnel Client Example

Figure A-4 shows how a DOS IP Tunnel LAN driver lets a NetWare client PC use an IP internet to communicate with a NetWare server on a remote IPX network. Figure A-3 provides the NET.CFG file for the DOS client PC at IP address 129.47.6.84 shown in Figure A-4.

Figure 11 Sample DOS IP Tunnel Client Configuration



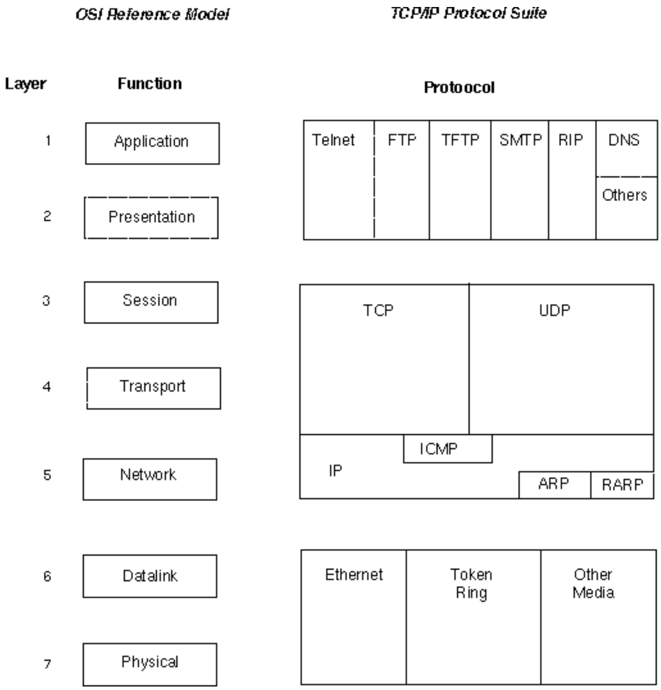
B TCP/IP Protocol Suite

This appendix contains an overview of the TCP/IP suite of protocols and its use for connecting dissimilar computers. It highlights the main features of TCP/IP for users familiar with network protocols or networking.

The TCP/IP Suite of Protocols

The protocols in the TCP/IP suite roughly correspond to a network communications model defined by the International Organization for Standardization (ISO). This model is called the Open Systems Interconnection (OSI) reference model. The OSI model describes an ideal computer network system in which communication on the network occurs between processes at discrete and identifiable layers. Each layer on a given host provides services to layers above it and receives services from the layers below it. Figure B-1 illustrates the seven layers of the OSI reference model as defined by ISO and the roughly corresponding layers of the TCP/IP protocol suite.

Figure 12 OSI Reference Model



The layering system lets developers concentrate their efforts on the functions in a given layer. It is not necessary for them to create all the mechanisms to send information across the network. They have to know only what services the software needs to provide to the layer above it, what services the layers below it can provide to the software, and which protocols in the suite provide those services.

Table B-1 lists some of the more common protocols in the TCP/IP suite, the services they provide, and the relationship between the protocols and the layers of the OSI reference model.

Table 10 TCP/IP Protocols

Protocol	Service
Internet Protocol (IP)	Provides packet delivery services between nodes.
Internet Control Message Protocol (ICMP)	Controls transmission of error and control messages between hosts and gateways.
Address Resolution Protocol (ARP)	Maps Internet addresses to physical addresses.
Reverse Address Resolution Protocol (RARP)	Maps physical addresses to Internet addresses.
Transmission Control Protocol (TCP)	Provides reliable stream delivery service between clients.
User Datagram Protocol (UDP)	Provides unreliable connectionless packet delivery service between clients.
File Transfer Protocol (FTP)	Provides application-level services for file transfer.
Telnet	Provides terminal emulation.
Routing Information Protocol (RIP)	Allows the exchange of routing information between routers.

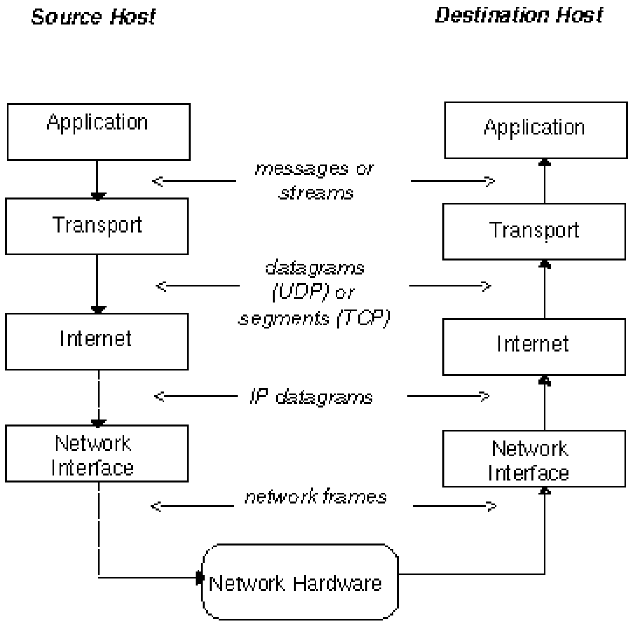
An Overview of TCP/IP Protocol Usage

Applications developed for TCP/IP generally use several of the protocols in the suite. The sum of the layers of the protocol suite is also known as the *protocol stack*. User applications communicate with the top layer of the protocol suite. The top-level protocol layer on the source computer passes information to the lower layers of the stack, which in turn pass it to the physical network. The physical network transfers the information to the destination computer. The lower layers of the protocol stack on the destination computer pass the information to higher layers, which in turn pass it to the destination application.

Each protocol layer within the TCP/IP suite has various functions; these functions are independent of the other layers. Each layer, however, expects to receive certain services from the layer beneath it, and each layer provides certain services to the layer above it.

Figure B-2 shows the layers of the TCP/IP suite. Each layer of the protocol stack on the source computer communicates with that same layer on the destination computer. The layers at the same level on the source and destination computers are *peers*. The application on the source computer and the application on the destination computer are also peers. From the perspective of the software developer or user, the transfer takes place as if the peer layers sent their packets directly to one another.

Figure 13 TCP/IP Protocol Layers



An application for transferring files using TCP, for instance, performs the following operations to send the file contents:

1. The application layer passes a stream of bytes to the transport layer on the source computer.
2. The transport layer divides the stream into TCP segments, adds a header with a sequence number for that segment, and passes the segment to the Internet (IP) layer.
3. The IP layer creates a packet with a data portion containing the TCP segment. The IP layer adds a packet header containing source and destination IP addresses. The IP layer also determines the physical address of the destination computer or intermediate computers on the way to the destination host. It passes the packet and the physical addresses to the datalink layer.
4. The datalink layer transmits the IP packet in the data portion of a datalink frame to the destination computer. This may include forwarding of the IP packet by intermediate systems.
5. At the destination computer, the datalink layer discards the datalink header and passes the IP packet to the IP layer.

6. The IP layer checks the IP packet header. If the checksum contained in the header does not match the checksum computed by the IP layer, it discards the packet.
7. If the checksums match, the IP layer discards the IP packet header and passes the TCP segment to the TCP layer. The TCP layer checks the sequence number to determine whether the segment is the correct segment in the sequence.
8. The TCP layer computes a checksum for the TCP header and data. If the computed checksum does not match the checksum transmitted in the header, the TCP layer discards the segment. If the checksum is correct and the segment is in the correct sequence, the TCP layer sends an acknowledgment to the source computer.
9. On the destination computer, the TCP layer discards the TCP header and passes the bytes in the segment just received to the application.
10. The application on the destination computer receives a stream of bytes, just as if it were directly connected to the application on the source computer.

The following sections describe these interactions in more detail:

- Physical addresses and Internet addresses
- Internet to physical address translation
- Dividing an Internet Network (subnetwork addressing)
- Unreliable packet delivery service
- Routing
- The Internet Protocol
- Error and control messages
- The transport layer protocols (UDP and TCP)

Physical Addresses and Internet Addresses

At the datalink layer, the nodes on a network communicate with other nodes on that network using addresses specific to that network. A node on a network can be a microcomputer, a file server, an intelligent printer, or any other device with its own TCP/IP implementation.

Each node has a *physical address* for the specific hardware device that connects it to a network. The physical addresses have different forms on different networks, and they are assigned in different ways. For instance, a physical address on an Ethernet network is a 6-byte numeric value, such as 08-00-14-57-69-69; it is assigned by the manufacturer of the Ethernet interface hardware. CCITT X.25 networks use the X.121 standard for physical addresses, which consist of 14-digit numbers. LocalTalk networks use 3-byte addresses, consisting of a 2-byte network number and a 1-byte node number. On a LocalTalk network, the network number is static, but the node number is assigned dynamically each time the node starts up.

NOTE: Throughout the rest of this appendix, all references to Media Access Control (MAC) addresses assume physical addresses on Ethernet, token ring, or ARCnet networks. Other network types use other algorithms for determining IP addresses.

The IP address for a node is a logical address and is independent of any particular hardware or network configuration; it has the same form regardless of the network type. It is a 4-byte (32-bit) numeric value that identifies both a network and a local host or node (computer or other device) on that network. The 4-byte IP address is usually represented in dotted decimal notation. Each byte is represented by a decimal number, and dots separate the bytes (for example, 129.47.6.17). In some contexts, IP addresses are represented as hexadecimal numbers (for example, 0x81.0x2F.0x6.0x11) or in octal (for example, 0201.057.06.021).

Nodes using the TCP/IP protocols translate the destination IP addresses to the physical addresses of MAC hardware to send packets to other nodes on the network. Each sending application sends its IP address in the packet as well. The receiving application can reply to the sender by using the sending IP address from the packet.

Because IP addresses are not dependent on any particular network type, they can be used to send packets from one network type to another network. On each network type, the TCP/IP software makes the correspondence between physical addresses and IP addresses on its network. If a packet is transmitted to another network, the TCP/IP software translates the destination IP address to a physical address appropriate for the network. The receiving application uses the sender IP address, which was included in the packet, to reply to the sender in the same manner.

Assigning Internet Addresses

An IP address is necessary for a node to communicate with other nodes using the TCP/IP protocol suite, including nodes on other private networks as well as those on the Internet. Your network address could be determined in one of the following ways:

- If you would like your network to join the Internet community, you must first get a registered Internet address from the following organization:

  DDN Network Information Center SRI International 333 Ravenswood Avenue, Room EJ291 Menlo Park, CA 94025 USA

- If your network is not part of the Internet community, you can choose an arbitrary IP network address. If you choose your IP network address this way, IP addresses for all the nodes on the network must meet the following criteria:
- The network portion of each address must agree with the network address (for example, all nodes on the 129.47 network must use the 129.47 network address).
- The IP address for each node must be unique within your network.

The rest of this section describes how to select an address class.

IP Address Classes

Each 4-byte IP address is divided into two parts:

- a network portion, which identifies the network
- a host portion, which identifies the node

IP addresses are differentiated into three classes based on the two most significant bits of the first of four bytes. This is done so that routers can efficiently extract the network portion of the address.

This division can fall at one of three locations within the 32-bit address. These divisions correspond to the three IP address classes: Class A, Class B, and Class C. Regardless of address class, all nodes on any single network share the same network portion; each node has a unique host portion.

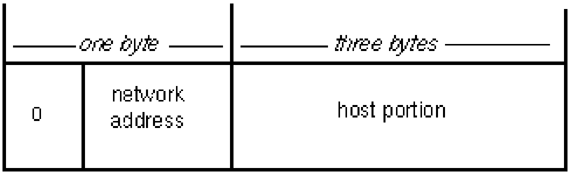
Class A Addresses

A Class A IP address, as illustrated below, consists of a 1-byte network portion followed by a 3-byte host portion. The highest-order bit of the network portion is always set to 0. Thus, within an internetwork, there can be a total of 126 Class A networks (1 through 126) with more than 16 million nodes in each (networks 0 and 127 are reserved).

For example (`n' = network address and `h' = host address):

```
Class A 0nnnnnnn.hhhhhhhh.hhhhhhhh.hhhhhhhh
```

(7 bits of network address, 24 bits of host address)



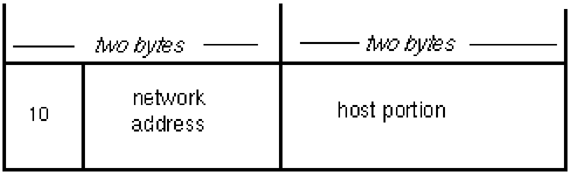
Class B Addresses

A Class B IP address, as illustrated below, consists of a 2-byte network portion followed by a 2-byte host portion. The two highest-order bits of the network portion are always set to 10. Thus, within a single internetwork there can be approximately 16 thousand Class B networks with more than 65 thousand nodes in each.

For example (`n' = network address and `h' = host address):

```
Class B 10nnnnnn.nnnnnnnn.hhhhhhhh.hhhhhhhh
```

(14 bits of network address, 16 bits of host address)



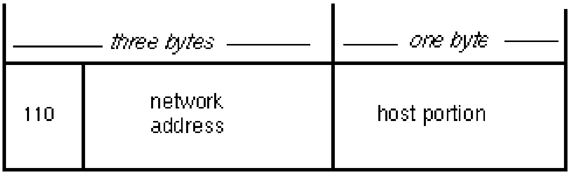
Class C Addresses

The Class C IP address, as illustrated below, consists of a 3-byte network portion followed by a 1-byte host portion. The three highest-order bits of the network portion are always set to 110. Thus, within a single internetwork, there can be approximately two million Class C networks with up to 254 nodes in each.

For example (`n' = network address and `h' = host address):

```
Class C 110nnnnn.nnnnnnnn.nnnnnnnn.hhhhhhhh
```

(21 bits of network address, 8 bits of host address)



Identifying Network Classes

When the first byte of an IP address fits in the range listed below, it identifies which of the three network classes that address belongs:

1 - 127 (1.h.h.h - 127.h.h.h) Class A

128-191 (128.n.h.h - 191.n.h.h)Class B

192 - 223 (192.n.n.h - 223.n.n.h)Class C

An IP address beginning with 154 would be a Class B address, with the first two bytes of the address representing the `n'etwork portion of the address and the last two representing the `h'ost portion. For example, an IP address of 154.1.0.3 means the IP network portion would be 154.1.0.0 and the host portion on that network would be #.#.0.3.

The network portion of an IP address should be the same for all nodes on that network. The network interface in the server that is connecting to network 89.0.0.0 must have a unique IP host address assigned to it, such as 89.0.0.254.

HINT: The key to selecting a number for the host portion of the IP address is to ensure that the number selected is unique, that no other host on the network has the same IP address.

Reserved IP Addresses

The IP addressing rules reserve the following types of IP addresses for special purposes:

- Network addresses. These are IP addresses in which the host portion is set to all zeros. 129.47.0.0, for example, is the network address (or network number) for a Class B network. These are addresses of networks rather than nodes on a network. By convention, no node is ever assigned a host portion consisting of all zeros.
- Broadcast addresses. These are addresses in which the host portion is set to all ones. A packet with a broadcast address is destined for every node on the network. By convention, no node is ever assigned a host portion consisting of all ones.
- Loopback addresses. 127.0.0.0 and 127.0.0.1 are reserved addresses.
- Addresses in which the network portion consists of all zeros or all ones are reserved.

Selecting an Appropriate Address Class

When selecting an IP address class, you must decide on both network numbers and host address portions. Because the first one, two, or three bits of the IP address determine how the entire address is to be interpreted, where the division between network address and host address portion is to occur, you should know the consequences of your choice. When deciding on a network class, you should consider the number of nodes to be supported on your TCP/IP network and the number of networks you plan to configure.

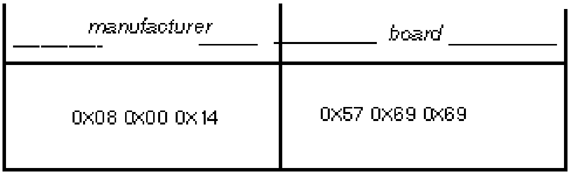
For example, if you use Class C addresses (the first three bits of the IP address are 110 binary), then you are restricted to 254 nodes on your network.

Internet to Physical Address Translation

Each different network type uses a particular form for the packets sent by its nodes. These forms include, among other things, the physical address of the destination node. Each physical medium has its own *physical* address for nodes on that medium. The physical addresses are also called Media Access Control (MAC) addresses. Ethernet or Token-Ring networks use 6-byte MAC addresses. ARCnet uses a 1-byte MAC address.

An example of the Ethernet form is shown in Figure B-3. The complete MAC address is always unique for each node. The first three bytes represent a unique vendor identification number, and the second three bytes, which can be duplicated across vendors, represent a vendor-assigned board identification number.

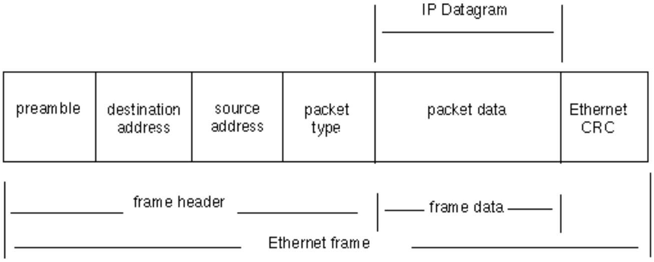
Figure 14 **Physical Ethernet Address**



manufacturer	*board*
0X08 0X00 0X14	0X57 0X69 0X69

IP addresses are independent of the hardware. When an IP packet is transmitted on the network, it is first *encapsulated* within the physical frame used by that network. Figure B-4 shows an IP packet encapsulated in an Ethernet frame. The IP packet contains an Internet address for a node, but the Ethernet frame must have a physical address for it to be delivered on the network. Therefore, the sending node must be able to determine which physical address on the network corresponds to the IP address contained in the IP packet.

Figure 15 **IP Datagram Encapsulation**



Mapping Internet Addresses to Physical Addresses

An IP address is mapped to a physical (or MAC) address using the Address Resolution Protocol (ARP) on broadcast networks such as Ethernet, Token Ring, or ARCnet. When a node sends a packet using IP, it must determine which physical address on the network corresponds to the destination IP address specified in the IP packet. To find the physical address, the node broadcasts an ARP packet containing the destination IP address. After this ARP packet seeks a response from the node with this destination IP address, the node with the destination IP address sends its physical address back to the requesting node.

Address Resolution Cache

To speed packet transmissions and reduce the number of broadcast requests, which must be examined by every node on the network, each node keeps an *address resolution cache*. Each time the node broadcasts an ARP request and receives a response, it creates an entry in its address resolution cache. The entry maps the IP address to the physical address.

When the node sends another IP packet, it looks up the IP address in its cache. If it finds that IP address, the node uses the corresponding physical address for its packet. The node broadcasts an ARP request only if the IP address is not in its cache.

Dividing an Internet Network

One Internet network (at a single Internet network address) can be divided into one or more smaller networks. Listed below are some reasons why you might divide your network:

- To use multiple media. It could be impossible, inconvenient, or too expensive to connect all nodes to a single network medium when these nodes are too far apart or already connected to different media.
- To reduce congestion. Traffic between nodes on a single network uses network bandwidth. As a result, more bandwidth is required when you have more nodes. Splitting nodes into separate networks reduces the number of nodes on a network. If nodes on a single, smaller network speak mostly to other nodes on the same network, congestion is reduced.
- To reduce CPU usage. Reducing CPU usage on connected nodes is similar to reducing congestion. More nodes on a network cause more

broadcasts on that network. Even if a broadcast is not sent to a particular node, each node on a network must react to every broadcast, before deciding to accept it or discard it.

- To isolate a network. By splitting a larger network into smaller networks, you limit the impact of one network's problems on another. Such problems can include network hardware failure (an open Ethernet tap), or software failures (broadcast storm).
- To improve security. On a broadcast network medium such as Ethernet, each node on a network has access to all packets sent on that network. By allowing sensitive network traffic only on one network, other network monitors can be prevented from accessing this sensitive traffic.

Methods for Dividing the Network

The following choices are available to divide your network:

- Create network numbers
- Obtain new network numbers
- Create subnetworks

Creating Network Numbers

If you are maintaining your own internetwork and can create your own network numbers, you can assign another number for each new network.

Obtaining New Network Numbers

If your network number has been assigned by a higher authority, you do not have the flexibility to create a network number. For example, if you are connected to the Internet, all network numbers must be allocated by the Network Information Center at SRI. However, you can request additional network numbers.

Creating a Subnetwork

If you have been assigned a Class A or Class B network, it is generally easier to divide your network into subnetworks than to request additional networks. You can use the network number you already have and split it into subnetworks.

Characteristics of a Subnetwork

Each subnetwork functions as if it were an independent network. To remote networks, however, the subnetworks appear to be a single, discrete network. This means that the local network needs only one Internet network address and that remote networks need not worry about the location of a particular node on a subnetwork.

Communication between a node on a local subnetwork and a node on a different subnetwork is like communication between nodes on two different networks. To a user, routing between subnetworks is transparent. Internally, however, the IP software recognizes any IP addresses that are destined for a subnetwork and sends those packets to the gateway to that subnetwork.

As in network-to-network communication, the gateway mapping information for subnetwork communication between subnetworks is maintained in the routing table (by the IP) for each node or gateway. In the case of subnetworks, however, the mapping information consists not only of the network address, but also the subnetwork address.

Therefore, the 4-byte IP address has the following portions:

```
<IP address>=<Network address> <Subnetwork address> <Host
  address>
```

When a network is divided into subnetworks, the host address portion of the IP address is divided into two parts, just as the IP address itself is divided into two parts. The host address portion specifies both the subnetwork at the Internet network and the node on that subnetwork.

For instance, if a network has the Class B IP network address portion 129.47, the remainder of the IP address can be divided into subnetwork addresses and host addresses. This division is controlled by the local network to allow the most flexibility for network development at the local site.

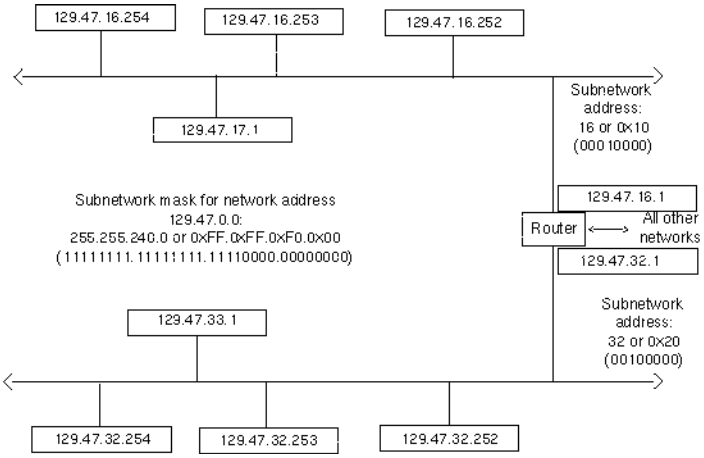
For example, the subnetwork address could comprise four bits of the remaining two bytes. This allows 14 subnetworks, each with 4094 nodes. In another example, the subnetwork address could comprise eight bits, allowing 254 subnetworks (a subnetwork address of all zeros or all ones is not valid), each with 254 nodes.

NOTE: A subnet field of all ones denotes all subnets of a particular network; therefore, all ones for a subnet cannot be used as a local IP address.

Figure B-5 shows a single IP network divided into two subnetworks. The router shown has physical devices and IP addresses on both subnetworks

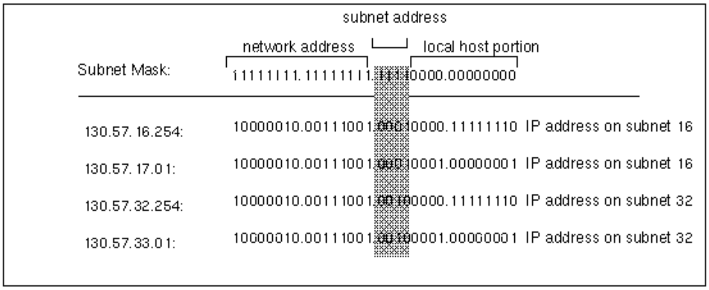
(129.47.16.1 and 129.47.32.1). It could also have physical devices and IP addresses (nn.nn.nn.nn) connecting it to other networks.

Figure 16 Network with Two Subnetworks



A subnetwork mask indicates how the host portion of the IP address is divided into subnetwork addresses and local host address portions. The network mask is a 32-bit number with all ones for all network and subnetwork address portions of the complete IP address, and all zeros for host address portions. With a Class B IP network address portion of 129.47 and a 4-bit subnetwork address, for instance, the subnetwork mask consists of 20 ones and 12 zeros. In essence, a subnetwork mask locally extends the network address portion of an IP address. Figure B-6 shows examples of IP network addresses, their relationship to the subnetwork mask, and the corresponding subnetwork.

Figure 17 Subnetwork Mask and IP Addresses


subnet address
network address
local host portion
Subnet Mask: 11111111.11111111.11110000.00000000
130.57. 16.254: 10000010.00111001.00010000.11111110 IP address on subnet 16
130.57. 17.01: 10000010.00111001.00010001.00000001 IP address on subnet 16
130.57.32.254: 10000010.00111001.00100000.11111110 IP address on subnet 32
130.57.33.01: 10000010.00111001.00100001.00000001 IP address on subnet 32

Subnet Addressing

The Network Information Center (NIC) assigns three classes of Internet IDs to different size networks, as shown in Table B-2.

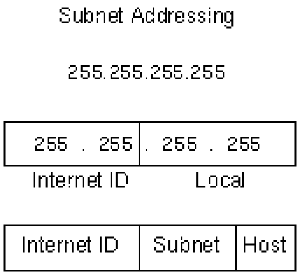
Table 11 Network Address Classes

Class	Length of Internet ID	Length of Local Portion	Size of Network
Class A	8 bits (1 octet)	24 bits (3 octets)	Largest networks
Class B	16 bits (2 octets)	16 bits (2 octets)	Medium-sized networks
Class C	24 bits (3 octets)	8 bits (1 octet)	Small networks

When the NIC assigns a longer Internet ID, there are fewer bits remaining in the local portion to uniquely identify each host within the network; shorter Internet IDs are normally assigned to larger networks.

Subnet addressing is a means of using a single Internet ID to identify a number of subnetworks within an organization. An organization that uses subnet addressing divides the local portion of each IP address into a subnet ID and a host ID. It can then divide its network into subnetworks and assign a unique subnet ID for each physical network. Figure B-7 illustrates the IP address and how it is divided to identify a subnet and host number.

Figure 18 Subnet Addressing



For example, with a Class B Internet ID, an organization can use the third octet in each host address to identify the subnet and use the fourth octet to uniquely identify each host on each subnetwork. With an Internet ID of 130.57, the first host on the first subnetwork could be assigned the address 130.57.1.1.

Subnet addressing allows the administrator of each subnetwork to easily assign unique host addresses. A subnet of all zeros, however, is not recommended by RFC 950 and is not supported by Novell.

The *subnet mask* is a 32-bit binary number that defines how many bits in the address identify the network and how many bits identify the host. A `1' indicates that a bit identifies the network; a `0' indicates that a bit identifies a host.

The subnet ID for the Class B network in the example above was one octet (8 bits) long; Class B Internet IDs are 16 bits. The following subnet mask would be used for this network:

```
255.255.255.0
```

or in binary:

```
11111111 11111111 11111111 00000000
```

The subnet mask used by an organization should be determined by the number of physical networks in the organization's internetwork and the maximum number of hosts on any network. it is not necessary to use an entire octet to represent the subnetwork.

Table B-3 illustrates the different subnet masks available for a Class B network, which has only one octet in the local portion

Table 12 Class B Subnet Masks

Max. No. Subnets	Max. No. Hosts/ Subnet	Subnet Mask	Subnet ID Length (# of bits)	Host ID Length (# of bits)
2	16382	255.255.192.0	2	14
6	8190	255.255.224.0	3	13
14	4094	255.255.240.0	4	12
30	2046	255.255.248.0	5	11
62	1022	255.255.252.0	6	10
126	510	255.255.254.0	7	9
254	254	255.255.255.0	8	8
510	126	255.255.255.128	9	7

1022	62	255.255.255.192	10	6
2046	30	255.255.255.224	11	5
4094	14	255.255.255.240	12	4
8190	6	255.255.255.248	13	3
16382	2	255.255.255.252	14	2

NOTE: A host address should not have a subnet or host ID field of all zeros or all ones; those values are reserved. The subnet mask must allow at least two bits for the subnet and host ID.

A Class C Internet ID uses three octets to identify the Internet ID. The fourth octet in each unique host address can be used to identify the subnet and the host. Table B-4 illustrates the different subnet masks available for a Class C network.

Table 13 **Class C Subnet Masks**

Max. No. Subnets	Max. No. Hosts/ Subnet	Subnet Mask	Subnet ID Length (# of bits)	Host ID Length (# of bits)
2	62	255.255.255.192	2	6
6	30	255.255.255.224	3	5
14	14	255.255.255.240	4	4
30	6	255.255.255.248	5	3
62	2	255.255.255.248	6	2

Unreliable Packet Delivery Service

In the TCP/IP protocol suite, all packets are delivered by the Internet Protocol *unreliable connectionless packet delivery* service. The service is *unreliable* because packet delivery is not guaranteed. A packet can be misdirected, duplicated, or lost on its way to its destination. The service is *connectionless* because all packets are transmitted independently of any other packets. This is in contrast to a telephone network, for instance, where a circuit is established and maintained, and the information traveling across the circuit is guaranteed of delivery.

TCP/IP applications using the unreliable connectionless packet delivery service can keep track of the status of delivery, however, by expecting to receive replies from the destination node, or by using one of the transport-layer protocols within the TCP/IP suite. Additionally, the routers on the Internet can send ICMP error messages to inform nodes of any problems that have developed.

Routing

The term *routing* refers to the transmission of a datagram from one node to another on the same or a different network. The term also refers to the paths that are chosen to transmit an IP datagram from its origin to its destination, based on the IP addresses contained in the datagram. The two basic kinds of routing are *direct* and *indirect*.

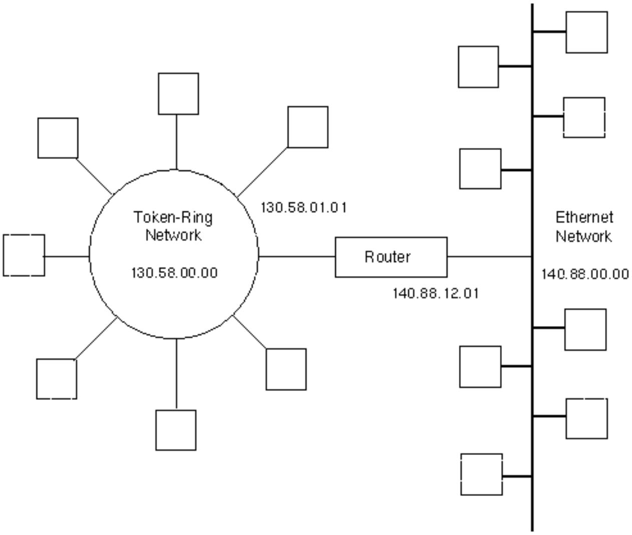
Direct Routing

Direct routing is the transmission of a datagram directly from one node to another within a single network. Within a network, a node sending an IP datagram can directly query all the other nodes on the network for the physical address corresponding to an IP address, encapsulate the IP datagram in a physical frame containing that physical address, and send it directly to the destination node's physical address on the network.

Indirect Routing

Indirect routing is the transmission of a datagram from one network to another through a node called a router. When a datagram is sent to a node on another network, the network portions of the originating IP address and the destination IP address are different. The sending node recognizes this difference, and sends the packet to the router that connects the originating network with other networks, as shown in Figure B-8. Two individual networks can only be connected if at least one computer is attached to both networks and is able to pass data in a form that is compatible with both networks.

Figure 19 Network Router

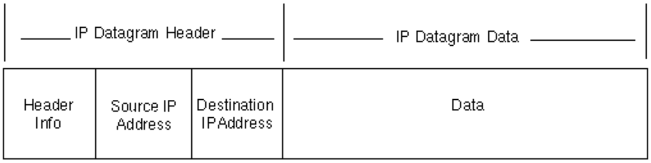


The sending node has a table of IP addresses for one or more computers on the network that serve as routers to other networks. It looks for the IP address of a router in its table and broadcasts an ARP request to the router for the physical address of the router. It then sends the packet containing the IP datagram to the router's physical address. When the router receives the IP datagram, it uses the IP address in the datagram to send to its final destination in a similar manner. If the IP address is on a network connected directly to the router, the router sends the IP datagram directly to the destination node. For all other network addresses, the router has only the address of another router that can route the packet to its destination.

The Internet Protocol

The Internet Protocol (IP) defines the form that packets must take and the ways to handle packets when they are transmitted or received. The form the packet takes is called an IP datagram. An IP datagram is analogous to a physical frame transmitted on a network. A datagram has a header section containing the sender and the receiver IP addresses among other information, and a data section. Figure B-9 shows the general form of an IP datagram. Each network type transmits IP packets in the data section of its physical frame.

Figure 20 IP Datagram Structure



Unlike a network frame, which has a physical length set by the technical requirements of the physical network characteristics, the length of a datagram is set by the networking software. When a datagram is transported by a network frame, it is encapsulated in the data area of the network frame. The IP software on a node creates a datagram that fits inside the network's physical transmission frame. In traveling to its destination, however, a datagram may pass across many different network types with many different physical frame lengths.

To handle this facet of packet transmission, the IP specifies a method for breaking datagrams into *fragments* at any node that must retransmit the datagrams, and a corresponding method for *reassembling fragments* at the destination. Therefore, a router receiving packets from one network with one physical frame size may need to fragment the IP packets it receives for retransmission on another network, if the second network has a smaller physical frame size than the first. Once packets are fragmented, they are not reassembled until they reach their ultimate destination.

Error and Control Messages

Another protocol in the TCP/IP suite is the Internet Control Message Protocol (ICMP). ICMP packets contain information about failures on the network: inoperative nodes and gateways, packet congestion at a gateway, and so on. The IP software, rather than the application, interprets an ICMP message. The IP software then takes the appropriate action with respect to the ICMP message independently of the application. Because an ICMP message may need to travel across several networks to reach its destination, it is encapsulated in the data portion of an IP datagram.

Transport Layer Protocols: UDP and TCP

The transport layer of the TCP/IP protocol suite consists of two protocols: the User Datagram Protocol (UDP) and the Transmission Control Protocol (TCP). UDP provides an unreliable connectionless delivery service to send and receive messages from specific processes on the sending and receiving nodes. TCP adds reliable stream-delivery services on top of the Internet Protocol's unreliable connectionless packet delivery service.

UDP

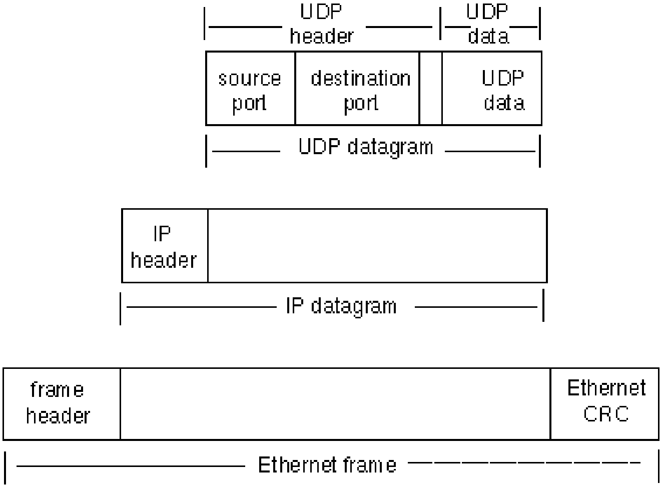
Within the TCP/IP suite, UDP allows applications to exchange individual packets of information over a network.

The UDP protocol defines a set of destinations as *protocol ports*. The protocol also defines two types of protocol ports: well-known port assignments and dynamically bound ports. In the case of well-known port assignments, TCP/IP reserves certain port numbers for certain applications. The ports numbered between 1 and 255 are well-known port numbers and are assigned to certain widely used applications. All TCP/IP applications use those same port numbers in the same way. In the case of dynamically bound ports, an application requesting services from a process must query the node first to identify which port the process uses. It can then direct UDP datagrams to that port.

UDP allows multiple clients to use the same port number and different IP addresses. This applies to both socket and TLI interfaces. The arriving UDP datagrams are delivered to the client that matches both the port number and destination address. If there is no matching client, the packet is delivered to the client whose address is 0.0.0.0; if no client exists with this address, the packet is dropped.

The UDP datagram is encapsulated in one or more IP datagrams that are in turn encapsulated in physical frames. Figure B-10 shows a UDP datagram encapsulated in an IP datagram, which is in turn encapsulated in an Ethernet frame. Figure B-10 also illustrates how the concept of layering, discussed at the start of this chapter, affects the construction of packets sent across the network.

Figure 21 UDP Datagram Encapsulation



In this example, the IP address directs the IP datagram to the correct node. At that destination, the IP software extracts the UDP datagram and delivers it to the UDP layer software. The UDP layer software delivers the UDP data and control information to the destination protocol port it specifies. The process at that port uses the data in the UDP datagram. The UDP datagram also contains a source protocol port so the destination process can reply correctly.

TCP

For applications that must send or receive large volumes of data, unreliable connectionless packet delivery may become burdensome. Application programmers may have to develop extensive error handling and status information modules to track the progress and state of data transfer for every application. The TCP/IP suite of protocols avoids this problem by using the Transmission Control Protocol (TCP). TCP is a *reliable stream-delivery protocol*. It establishes a *virtual circuit* between two applications, and sends a *stream* of bytes to a destination in exactly the same order that they left the source. Before transmission begins, the applications at both ends of transmission obtain a TCP *port* from their respective operating systems. These are analogous to the ports used by the UDP protocol. The application initiating the transfer, known as the *active* side, generally obtains a port dynamically. The application responding to the transfer request, known as the *passive* side, generally uses a well-known TCP port. The active side calls the well-known TCP port on the passive side.

Like the UDP datagrams, TCP *segments* are encapsulated in an IP datagram. TCP *buffers* the stream by waiting for enough data to fill a large datagram before sending a datagram. The stream is *unstructured*, which means that before transmission of data, both the sending and receiving applications must agree on the meaning of the contents of the stream. The TCP protocol uses *full-duplex* transmission. Full duplex means two data streams can flow in opposite directions simultaneously. Thus, the receiving application can send control information back to the sending application while the sending application continues to send data.

The TCP protocol gives each segment a sequence number. At the receiving end of the virtual circuit, the application checks successive sequence numbers to ensure that all the segments are received and processed in the order of the sequence numbers. When the receiving end gets the next segment in the sequence, it sends an *acknowledgment* to the sending node. When the sending node gets the acknowledgment, it indicates to the application that the last segment was successfully sent. If the sending node doesn't receive an acknowledgment for a segment within a certain time, it retransmits that segment. This scheme, called *positive acknowledgment with retransmission*, ensures that the stream delivery is reliable.

C Manually Configuring and Loading TCP/IP

This chapter describes how to load and configure the TCP/IP software.

These TCP/IP commands can be entered from the NetWare console prompt, but normally you put these commands in the file SYS:SYSTEM\AUTOEXEC.NCF so they are executed automatically when the system is loaded. Use the INSTALL NLM to edit this file.

Configuring TCP/IP

This section briefly describes how to configure NetWare TCP/IP. You need to perform the following operations when starting the software:

1. Use the CONFIG command, as described in the *NetWare Version 4.0 Supervising the Network* manual, to verify the LAN driver(s) you want TCP/IP to use.
2. Set up your network database files, as described in Chapter 3, "Managing Network Database Files."
3. Load TCPIP.NLM. This puts the software into the system and starts it, but does not connect it to any network interfaces. TCP/IP automatically loads the SNMP.NLM subsystem.
4. If you want to log SNMP trap messages, load SNMPLOG.NLM.
5. Bind IP to the LAN driver you want NetWare TCP/IP to use.
6. If you must initialize static routing entries, load IPCONFIG.NLM.
7. Load any applications that run on TCP/IP (for example, NFS and TCPCON).

The following example shows how to set up TCP/IP to use IP address 1.0.0.3 on an Ethernet network. This example includes a LOAD IPCONFIG command for setting up static routes and a LOAD SNMPLOG command for setting up an SNMP log.

```
LOAD NE2000 PORT=2E0 INT=2 FRAME=ETHERNET_II NAME=ENGR-EII
  <Enter>>
LOAD TCPIP <Enter>
LOAD SNMPLOG <Enter>>
BIND IP TO ENGR-EII ADDR=1.0.0.3 <Enter>
LOAD IPCONFIG <Enter>
```

NOTE: Always specify the frame type on the LOAD command for the board driver. If no frame type is specified, the default frame type of ETHERNET_802.2 is used.

Linking TCP/IP with Your System Using LOAD

Use the LOAD command to link the TCP/IP module with the operating system. This command starts TCP/IP, but does not connect it to any network interface. Once the TCP/IP module is loaded, it must be bound to a network driver with the BIND command before it can communicate with other nodes.

The LOAD TCPIP command, usually located in the AUTOEXEC.NCF file, determines whether TCP/IP on the server will be recognized as a TCP/IP host on the network or as an IP router between two or more networks (Ethernet, Token Ring, or ARCnet).

The LOAD command to make the server visible as a TCP/IP host on the network has the following form:

```
LOAD TCPIP
```

The LOAD command to make the server route IP packets has the following form:

```
LOAD TCPIP FORWARD=YES
```

The full LOAD command line and its parameters consists of the following:

```
LOAD TCPIP [FORWARD={YES|NO}] [RIP={YES|NO}]
  [TRAP=ip_address] <Enter>
```

The TCP/IP LOAD command parameters are summarized in Table C-1.

Table 14 TCP/IP LOAD Options

Parameter	Values	Explanation
forward	Yes or No	Enable or disable IP packet forwarding. The default is No.
rip	Yes or No	Enable or disable the routing information protocol (RIP) module. The default is Yes.
trap	IP address/Host name	Specifies an IP address or host name to receive SNMP trap messages. The default is to send traps to the local trap logger.

The following paragraphs describe each parameter in detail.

The forward parameter enables or disables IP packet forwarding. When enabled, the server functions as an IP router. A valid setting is either Yes or No. The default value is No.

NOTE: If you do not include the *forward* parameter on the LOAD command, your server does not act as a router.

The *rip* parameter enables or disables the RIP module. When disabled, IP does not look at RIP traffic and does not send any. When enabled, IP monitors RIP traffic for routing information and, if the server is functioning as an IP router (if *forward*=yes), IP broadcasts RIP updates indicating the other networks it can reach. RIP is enabled by default. A valid value is either Yes or No.

The *trap* parameter specifies the IP address or host name to which the local system sends SNMP trap messages. If this parameter is specified, all SNMP trap messages are transmitted to the given address. If not specified, TCP/IP sends trap messages to 127.0.0.1, the loopback address. This delivers trap messages to the local SNMP trap logger.

NOTE: Enter all IP addresses in dotted notation. Refer to Appendix B, "TCP/IP Protocol Suite," for information about dotted notation.

The following examples illustrate how to specify the various LOAD parameters:

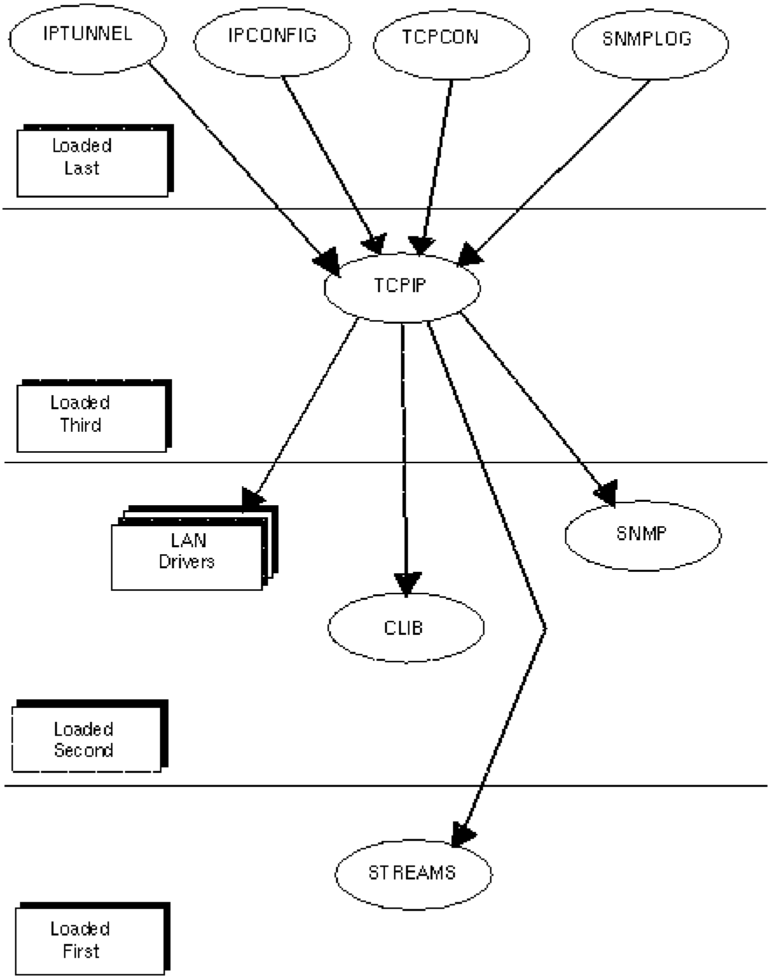
```
LOAD TCPIP <Enter>
LOAD TCPIP FORWARD=YES TRAP=1.0.0.2 <Enter>
LOAD TCPIP RIP=NO <Enter>
```

The first example loads TCPIP. The second example specifies the *forward* and *trap* parameters. The third example turns RIP off.

Load-Order Dependencies

Most NLM files use services provided by other NLM files. An NLM that provides a service must be loaded before other NLM files using that service. Figure C-1 shows the NLM files that depend on TCP/IP and the NLM files that TCP/IP depends on. You must load the NLM files in the order shown. Arrows point from a service user to a service provider.

Figure 22 TCP/IP Load-Order Dependencies



NLM files toward the bottom of the figure must be loaded before those closer to the top of the figure. NLM files at the same level can be loaded in any order.

The TCPIP NLM is pictured as having a loose dependence on the LAN drivers. Although this NLM uses services provided by the LAN drivers, it accesses those services indirectly through the NetWare ODI interface.

NOTE: LAN drivers can be loaded in any order (before or after the TCP/IP NLM), but a LAN driver must be present for IP to be bound to it.

The dependencies between NLM files are built into the NLM files. In most cases, if an NLM requires another NLM that is not yet loaded, it automatically loads the required NLM. This means that the operating system will find the required NLM and load it before loading the NLM that depends on the service it provides. For example, if the TCPIP NLM is loaded before any other NLM, NetWare loads the CLIB NLM, the STREAMS NLM, and the SNMP NLM before actually loading the TCPIP NLM.

Because TCP/IP takes command-line arguments, you may not want it to be loaded automatically.

Loading a LAN Driver

The *NetWare 4.0 Supervising the Network* manual describes the basic elements of loading a LAN driver, and your system probably already loads a LAN driver for IPX. However, the TCP/IP protocol suite sometimes requires a different datalink framing method than IPX. This section describes how to load drivers with the correct framing type for TCP/IP.

Depending on how you configured your IPX network, you may find you must load a particular LAN driver twice with two different framing types for the same network interface board. It is strongly recommended that you use the *name* parameter when you load the driver. This lets you assign a name of your own choosing to each logical board in the system.

On Ethernet networks, the TCP/IP protocol suite uses the framing type known to NetWare LAN drivers as Ethernet_II. To get that framing type, the LAN driver must be loaded with the *frame* parameter set to Ethernet_II.

IMPORTANT: Always specify the frame type on the LOAD command for the board driver. If no frame type is specified, the default frame type of ETHERNET_802.3 is used.

When you reload a driver for the same physical board, be sure to use the same hardware settings (for example, the port and int parameters) as in the first LOAD command for that board.

To load an Ethernet LAN driver, the command-line format consists of the following:

```
LOAD <LAN driver> INT=<interrupt> PORT=<port> FRAME=<frame
   type> NAME=<board name> <Enter>
```

For example:

```
LOAD NE2000 INT=2 PORT=2E0 FRAME=ETHERNET_II NAME=ETHERNET
   <Enter>
```

On Token-Ring networks, the TCP/IP protocol suite uses the framing type known to NetWare LAN drivers as `Token-Ring_SNAP`.

Again, IPX normally uses a different framing technique, so you must load the Token-Ring LAN driver a second time as follows:

```
LOAD TOKEN FRAME=TOKEN-RING_SNAP NAME=TOKEN-SNAP <Enter>
```

Maximum Packet Receive Buffers

Depending on the TCP/IP applications and their use, you may need to increase the value of the SET command Maximum Packet Receive Buffers parameter. The default value is 100 buffers, which may be insufficient for some configurations (supported values are 50 through 2000). You can use MONITOR.NLM to observe the number of Packet Receive Buffers currently allocated, to determine whether the current limit is sufficient. If many TCP/IP applications are concurrently active, the buffering needs are greater.

You can use the SET command with the Maximum Packet Receive Buffers parameter at the NetWare console prompt, but if you put this command in the SYS:SYSTEM\AUTOEXEC.NCF file, this parameter is set automatically each time the system is loaded.

For example:

```
SET MAXIMUM PACKET RECEIVE BUFFERS=200 <Enter>
```

NOTE: Increase this value based on memory requirements. By increasing the Maximum Packet Receive Buffers, you are allowing NetWare to use memory for network packets at the expense of other memory needs. If you set this value too high, NetWare could use too much memory for packets, leaving insufficient memory to continue normal operations.

Refer to the *NetWare 4.0 Supervising the Network* manual for more information about this SET command parameter.

Connecting IP to a Network Interface Using BIND

Once you load TCP/IP into your server, you must use the BIND command to connect IP to a network interface before your server can use TCP/IP to communicate with other nodes on the network. The BIND command binds IP to a network driver that has been previously loaded.

The BIND command for IP has the following form:

```
BIND IP TO board_name [ADDR = ip_address] [MASK =
  mask_address] [BCAST = bcast_address] [GATE =
  gate_address] [DEFROUTE = {YES | NO}] [ARP = {YES | NO}]
  [COST = number_of_hops] [POISON = {YES | NO}] <Enter
```

For example:

```
BIND IP TO NE2000 ADDR=89.1.2.3 <Enter>
```

Table C-2 summarizes each BIND parameter.

Table 15 Parameters for TCP/IP binding options

Parameter	Explanation
addr	Specifies the node's local IP address or host name on the network connected to this interface. There is no default value.
mask	Indicates the IP network mask for the IP network connected to this interface. The default value is the standard IP network mask.
bcast	Provides the default IP address to use for broadcasting on this network. The default value is 0xFF.0xFF.0xFF.0xFF.
gate	Provides a default gateway on this IP network. If this parameter is not specified, IP gathers all routing information from RIP.
defroute	If forward=Yes on the TCP/IP load line, announces the node as a default gateway through RIP. The default value is No.
arp	Indicates whether the Address Resolution Protocol (ARP) should be used on this network to map IP addresses to hardware addresses. The default value is Yes.
cost	Number of hops of cost to assign to this interface. The default cost is 1.

Parameter	Explanation
poison	Enables or disables the use of RIP poison reverse for routing updates sent on this interface. The default setting is No.

The following paragraphs describe each parameter in detail.

The *addr* parameter specifies the node's local IP address or host name on the network connected to this interface. IP must have a different local IP address for each interface it is bound to. Each IP address in an IP internetwork must be unique. Normally, these addresses are assigned by a network administrator. There is no default value. If you do not provide an addr value, NetWare prompts you for one.

NOTE: Enter all IP addresses in dotted notation. Refer to Appendix B, "TCP/IP Protocol Suite," for information about dotted notation.

The *mask* parameter lets you specify a subnetwork mask for the IP network connected to this interface. All nodes on the network must use the same mask. In the mask, each bit of the address's network number is set to one and each bit of the address host number is set to zero. If you do not specify the *mask* parameter, IP assumes the network is not divided into subnetworks.

TCP/IP allows subnets of the same network to have different subnet masks, although there are several restrictions on what combinations of masks can be used. TCP/IP issues a warning message when it detects such a configuration to ensure that masks are intentionally different.

Novell does not recommend this type of configuration, as it requires IP expertise to set up the environment so that packets can be successfully routed through the IP internet. Furthermore, the RIP routing protocol cannot handle mixed subnet masks, so NetWare cannot express the topology accurately to the rest of the internet.

The *bcast* parameter provides the default IP address to use for broadcasting on this network. The default value is all ones (0xFF.0xFF.0xFF.0xFF). It is recommended that you use this address unless you have compatibility problems with existing networking programs that do not understand this address. (A UNIX node running 4.2BSD reacts unpredictably when it receives packets with this IP address, and many products derived from 4.2BSD still have this problem.)

The *gate* parameter provides a default gateway on this network. IP uses this gateway in the absence of more specific information. If this parameter is not specified, IP gathers all routing information from RIP. Novell does not recommend using the *gate* parameter when binding TCP/IP with `RIP=yes`. There is no default value for this parameter.

If your server is functioning as a gateway (that is, TCP/IP was loaded with *forward*=yes), when the *defroute* parameter is set to Yes, IP announces itself as a default route in RIP messages sent to this network. Otherwise, IP announces specific routing information only to the network. This parameter has a default value of No.

NOTE: Default routes are very dangerous and should be used with great care. Incorrect configuration of default routes can produce routing loops.

The *arp* parameter indicates whether ARP should be used on this network to map IP addresses to hardware addresses. When set to No, the host portion of the IP address is mapped directly to the local hardware address. It is strongly urged that you leave this set to its default value, which is Yes.

The *cost* parameter is the number of hops of cost to assign to this interface. The value of this parameter is a base 10 integer no smaller than 1 and no larger than 15. This indicates an abstract cost of sending packets on this interface. This cost is reflected in the routing database for all routes through this interface. In general, higher costs discourage IP from using this interface. The default cost is 1.

The *poison* parameter (with a value of Yes or No) enables or disables the use of RIP poison reverse for routing updates sent on this interface. This obscure parameter controls the use of poison reverse in RIP updates when this node is acting as a gateway. For more information about RIP and poison reverse, refer to RFC 1058, Routing Information Protocol. The default value, No, reduces RIP traffic on this network at a small cost in stability.

Examples:

```
BIND IP TO NE2000 ADDR=1.0.0.2 MASK=255.255.0.0 <Enter>
BIND IP TO ENGR-EII COST=3 ADDR=1.0.0.3 GATE=1.0.0.4
  <Enter>
BIND IP TO TRXNET MASK=FF.FF.FC.0 <Enter>
```

The first example binds IP to LAN driver NE2000 with IP address 1.0.0.2 with a subnet mask of 255.255.0.0 for the IP network. The second example binds IP to a board named ENGR-EII, specifying the *cost*, *addr*, and *gate* parameters. The third example binds IP to LAN driver TRXNET with a subnet mask of 0xFF.0xFF.0xFC.0x0. An IP address is not provided, so the system prompts for one.

Disconnecting IP from a Network Interface Using UNBIND

TCP/IP can be disconnected from a network interface by using the UNBIND command. This command has no parameters specific to TCP/IP.

The general form of the command is as follows:

```
UNBIND IP FROM board_name <Enter>
```

Refer to the *NetWare Version 4.0 Supervising the Network* manual for more information about the UNBIND command.

Unbinding causes IP to disassociate itself from the specified driver. It discards all routing entries that use interfaces supported by the specified driver and routes all future packets to another interface, if one is available to reach the destination. If the server is functioning as a gateway with RIP active, a RIP packet is sent on each interface, including the one being disconnected, informing all RIP peers of the configuration change.

Examples:

```
UNBIND IP FROM NE2000 <Enter>
UNBIND IP FROM MKTG-EII <Enter>
```

The first example disconnects IP from LAN driver NE2000. The second example disconnects IP from a board named MKTG-EII.

Removing TCP/IP from Memory Using UNLOAD

To stop the TCP/IP module and remove it from system memory, use the UNLOAD command, as follows:

```
UNLOAD TCPIP <Enter>
```

You cannot remove TCP/IP while other NLM files, such as NFS or TCPCON, are directly linked to the TCPIP NLM. You must remove the other NLM files first.

Unloading stops TCP/IP and notifies all clients that are using streams or sockets of an abrupt disruption of service. If IP is still bound to any network interfaces, it is unbound. Unloading the TCPIP NLM frees 135 kilobytes of memory. Refer to Table 2-1 for memory requirements of other NLM files.

If the TCPIP NLM is unloaded while any application is actively using the AT&T Streams interface for TCP, UDP, or IP communications, the system may fail. Currently, no Novell products use streams to access TCP/IP, so you should not encounter this problem unless you have installed add-on TCP/IP products on your system. If such an application is present, TCP/IP issues the following warning:

```
Some utilities have streams open to the TCP/IP protocols. TCP/
  IP service to those utilities will be terminated, and system
  integrity may be compromised.

Unload module anyway? n
```

Novell strongly discourages you from unloading TCP/IP until you have first unloaded the utilities using the streams interface.

Configuring Static Routes Using the IPCONFIG NLM

This section explains how to set static routes using the IP Configuration (IPCONFIG) NLM.

Setting Static Routes Using the IPCONFIG NLM

Typically, TCP/IP obtains its routing information from RIP. Gateways use RIP to inform each other of the internetwork topology. The RIP information is sufficient for most situations, but in some situations RIP may not provide enough information for TCP/IP to route packets correctly. As an alternative, a nearby router that cannot use RIP can be configured as a static route, with the IP configuration utility.

If you require static routes, you should indicate the static routes in the SYS:ETC\GATEWAYS file and then load the IPCONFIG NLM after you load the TCPIP NLM and bind IP. The IPCONFIG NLM reads the SYS:ETC\GATEWAYS file and places the indicated routes into the internetwork IP routing database. IP uses those routes to direct packets in addition to the routes derived from RIP. If the TCPIP NLM is functioning as an IP router, it also announces the static routes to other routers through RIP.

NOTE: In most cases, you should not need static routes; therefore, you do not need to create the SYS:ETC\GATEWAYS file, and you should not load IPCONFIG.

Format of SYS:ETC\GATEWAYS

If you are familiar with the UNIX IP utility routed, you will recognize that SYS:ETC\GATEWAYS has a format compatible with the `\ETC\GATEWAYS` file read by `routed`. Comments and blank lines are ignored. Comments begin at a number sign (#) and continue to the end of the line.

Each route is on its own line and has the following format:

```
{NET network_name |{network_number [/network_mask]} | HOST
  host} GATEWAY gateway [METRIC cost] [{ACTIVE | PASSIVE}]
```

where*:*

The *network_name* is the name of a network in SYS:ETC\NETWORKS.

The *network_number/network_mask* is the IP network address and its subnet mask; the subnet mask is optional. If the *network_mask* is not specified, the subnet mask is derived from the network number, excluding all trailing zeros. For example, if 130.57.1 is specified as the network number, the implied subnet mask is 255.255.255.0.

The *host* is the IP address of a host or the name of a host in SYS:ETC\HOSTS.

The *gateway* is the IP address of a gateway on a locally connected network or the name of such a gateway in SYS:ETC\HOSTS.

The *cost*, or measure of expense, is a value greater than or equal to 1 but less than or equal to 15.

The *active* or *passive* indicates the type of route for routing information.

For example:

```
NET 2.0.0.0 GATEWAY 129.1.0.3
NET 2.0.0.0 GATEWAY 129.1.0.3 HOST 129.1.0.3 GATEWAY
  193.1.1.1 PASSIVE
NET 0 GATEWAY 129.1.1.1 COST 4
NET 130.57.1.0/255.255.255.0 GATEWAY 193.1.1.1 PASSIVE
NET 0 GATEWAY 139.1.1.1 COST 4
```

Loading the IPCONFIG NLM

To add the static routes in SYS:ETC\GATEWAYS to the routing database, you must load the IPCONFIG NLM. This can be done by entering the LOAD IPCONFIG command at the system console. Normally, you place this command in your SYS:SYSTEM\AUTOEXEC.NCF file so that the static routes are added each time the system is loaded.

The command is as follows:

```
LOAD IPCONFIG <Enter>
```

NOTE: If you do not have any static routes to configure, you do not need to load IPCONFIG.

IPCONFIG reads SYS:ETC\GATEWAYS, places each route into the routing database, and then exits. While you are "debugging" your GATEWAYS file, problems in the file may cause IPCONFIG to print diagnostic messages.

For each route IPCONFIG finds in SYS:ETC\GATEWAYS, IPCONFIG adds a routing entry for the host or network indicated. The first hop in the new routing entry is the provided gateway and the metric is the given cost. If no metric is given, a cost of 1 is used.

Placing the keyword *active* on the line indicates that the route is maintained by the indicated gateway using normal RIP updates. Active routes are subject to normal RIP rules. They may be replaced by routes with lower metric values and they time out and disappear if no RIP updates are received for them.

The keyword *passive*, which is the default if neither *active* nor *passive* are given, indicates a route for which no RIP information will ever be available. Such routes never time out and IP ignores RIP information for them. They can be deleted or modified using TCPCON.

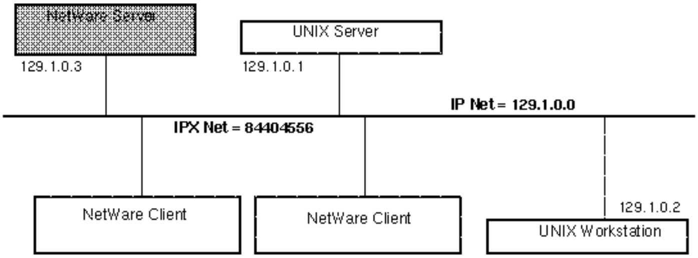
D Manually Configured TCP/IP Examples

This appendix shows how to configure NetWare servers as either IP end nodes or routers in an IP internet. It describes four network scenarios and explains how each case affects the server's IP configuration. Each scenario builds on the preceding scenario.

The examples use a number of specific IP addresses. Please do not use these addresses. You should follow your own site-determined network administration procedures to obtain unused addresses from the person responsible for your network IP address space.

Case 1: Single Network with IPX and IP Nodes

Figure 23 Single Network with IPX and IP Nodes



The Case 1 scenario consists of the following elements:

- Several UNIX and NetWare systems (clients and servers) share a single Ethernet network.

- The NetWare systems communicate only by using IPX on network number 84404556.
- The UNIX server and client systems communicate only by using IP on network number 129.1.0.0.
- The UNIX server and client addresses belong to the same Ethernet network and bear a unique host portion (the third and fourth bytes of the IP address). Refer to Appendix B, "TCP/IP Protocol Suite," for detailed information about IP address structure.

Before you can connect IPX and IP systems, you must assign the NetWare server a unique IP address.

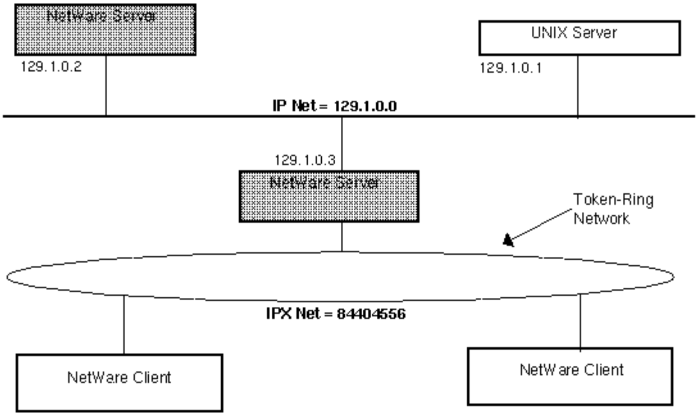
This configuration uses NetWare TCP/IP to permit the UNIX systems to communicate with the NetWare server. The NetWare server is connected to only one network, and therefore does no routing. The key configuration requirement is to assign an IP address. This address must belong to the same network and be unique. Assuming that Figure D-1 shows all IP nodes on the network, then 129.1.0.3 is a suitable IP address because no other node is assigned the same address.

The NetWare server AUTOEXEC.NCF file contains the following commands (PORT and INT values depend on hardware configuration):

```
LOAD NE2000 PORT=300 INT=3 FRAME=ETHERNET_802.3
  NAME=IPXNET
BIND IPX TO IPXNET NET= 84404556
LOAD NE2000 PORT=300 INT=3 FRAME=ETHERNET_II NAME=IPNET
LOAD TCPIP
BIND IP TO IPNET ADDR=129.1.0.3
```

Case 2: Separate IP and IPX Networks

Figure 24 Separate IP and IPX Networks



The Case 2 scenario consists of the following elements:

- Two networks share a NetWare server. The upper, Ethernet network uses only the IP protocol, while the lower, Token-Ring network uses only the IPX protocol.
- The UNIX client system has the same access to the NetWare server's resources that it did in Case 1.

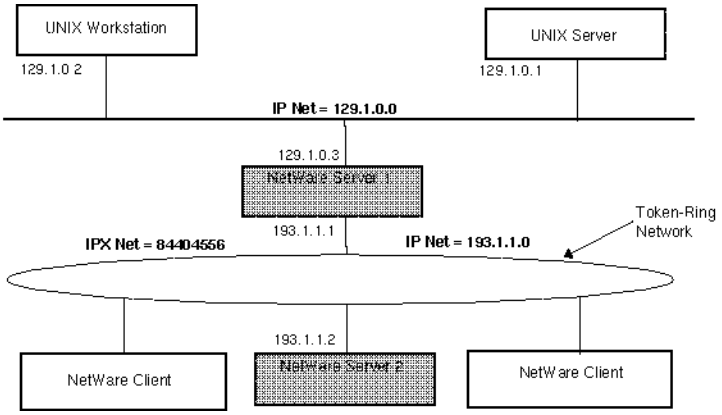
To connect the IPX and IP systems, follow the same steps as in Case 1, except you bind IP to a different interface than IPX.

The NetWare server AUTOEXEC.NCF file would contain the following commands:

```
LOAD TOKEN FRAME=TOKEN-RING NAME=IPXNET
BIND IPX TO IPXNET NET= 84404556
LOAD NE2000 PORT=300 INT=3 FRAME=ETHERNET_II NAME=IPNET
LOAD TCPIP
BIND IP TO IPNET ADDR=129.1.0.3
```

Case 3: Routing IP Packets through a NetWare Server

Figure 25 Routing IP Packets through a NetWare Server



The Case 3 scenario is similar to Case 2, except for the following elements:

- Case 3 provides access from the UNIX client system to a second NetWare server.
- The second NetWare server (193.1.1.2) is on the Token-Ring network.
- Server 193.1.1.2 does not have an interface to the Ethernet network.

In this case, the only means of access to the NetWare server 193.1.1.2 from a UNIX client on the Ethernet network is by routing IP packets through the first NetWare server, 129.1.0.3.

Take the following steps to configure your system to route packets to the Token-Ring network:

- Assign a new IP network number to the Token-Ring network.
- Assign a unique IP address to each NetWare server using IP on this network (193.1.1.1 and 193.1.1.2 are used in this example).

Unlike the previous examples, NetWare server 1 has two IP interfaces, each with its own address.

The AUTOEXEC.NCF configuration file might look like this:

```
LOAD TOKEN FRAME=TOKEN-RING NAME=IPX-TOKEN
BIND IPX TO IPX-TOKEN NET=84404556
```

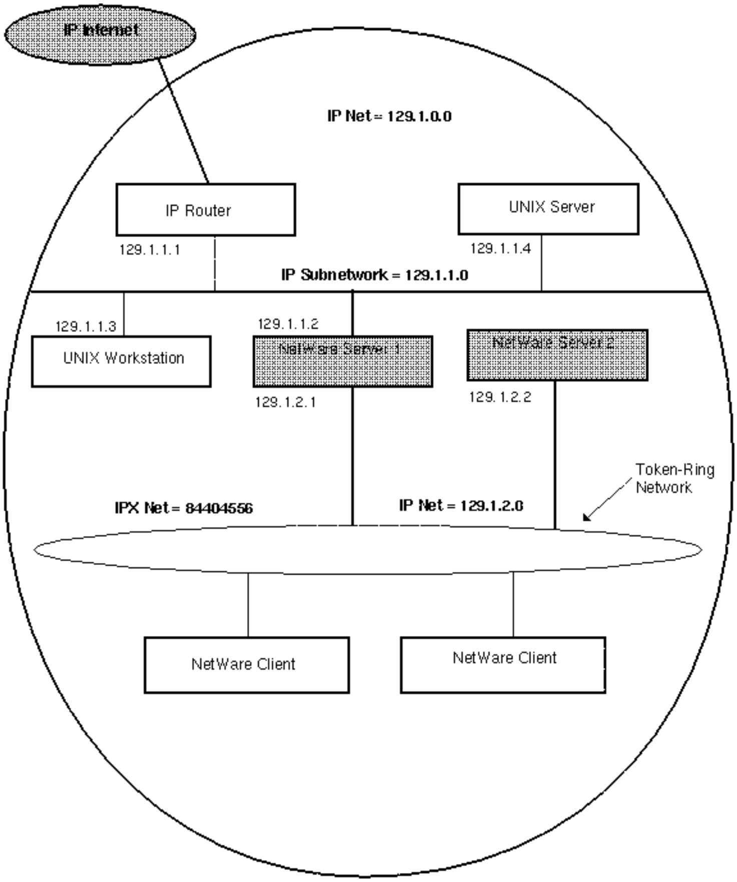
```
LOAD NE2000 PORT=300 INT=3 FRAME=ETHERNET_II NAME=IP-ETHER
LOAD TOKEN FRAME=TOKEN-RING_SNAP NAME=IP-TOKEN
LOAD TCPIP FORWARD=YES
BIND IP TO IP-ETHER ADDR=129.1.0.3
BIND IP TO IP-TOKEN ADDR=193.1.1.1
```

The Server 2 configuration is similar to other end-node configurations, in previous examples:

```
LOAD TOKEN FRAME=TOKEN-RING NAME=IPX-TOKEN
BIND IPX TO IPX-TOKEN NET=84404556
LOAD TOKEN FRAME=TOKEN-RING_SNAP NAME=IP-TOKEN
LOAD TCPIP
BIND IP TO IP-TOKEN ADDR=193.1.1.2
```

Case 4: Subnetwork Configuration

Figure 26 Subnetwork Configuration



The Case 4 scenario, shown in Figure D-4, is similar to Case 3, except the Ethernet network is attached by an IP router to a large IP internetwork, which is administered by another organization.

All network numbers are assigned by Internet administrators. In this example, the assigned network number is divided into subnetworks. Subnetwork numbers are then assigned to the internal networks.

Follow these steps to communicate with the IP Internet through a subnetwork (as shown in this scenario):

- Treat all local IP networks as subnetworks of the assigned network number, the class B address 129.1.0.0.
- Use the third byte of the address as a subnetwork field, assigning subnetwork 129.1.1.0 to the Ethernet network and 129.1.2.0 to the Token-Ring network.
- Use the fourth byte of each IP address to identify each host as a unique node on the subnetwork.
- Configure NetWare server 1 with a subnet mask of 0xFF.0xFF.0xFF.0x00 for each IP interface.

The AUTOEXEC.NCF file of NetWare server 1 would have the following commands:

```
LOAD TOKEN FRAME=TOKEN-RING NAME=IPX-TOKEN
BIND IPX TO IPX-TOKEN NET=84404556
LOAD NE2000 PORT=300 INT=3 FRAME=ETHERNET_II NAME=IP-ETHER
LOAD TCPIP FORWARD=YES
BIND IP TO IP-ETHER ADDR=129.1.1.2 MASK=FF.FF.FF.00
LOAD TOKEN FRAME=TOKEN-RING_SNAP NAME=IP-TOKEN
BIND IP TO IP-TOKEN ADDR=129.1.2.1 MASK=FF.FF.FF.00
```

The AUTOEXEC.NCF file of Server 2 uses the same subnet mask:

```
LOAD TOKEN FRAME=TOKEN-RING NAME=IPX-TOKEN
BIND IPX TO IPX-TOKEN NET=84404556
LOAD TOKEN FRAME=TOKEN-RING_SNAP NAME=IP-TOKEN
LOAD TCPIP
BIND IP TO IP-TOKEN ADDR=129.1.2.2 MASK=FF.FF.FF.00
```

E Error Messages

This appendix lists and explains the error messages that could display when you are using TCP/IP. These messages are classified as follows:

- Simple errors begin with the text Error:. These errors indicate that the NLM cannot complete the requested task as specified, but the NLM prompts the console operator for additional information. If the console operator provides better information, the task can be completed.
- Fatal errors begin with the text Fatal Error:. A fatal error is an error that prevents the NLM from accomplishing the requested task. For example, a fatal error on a bind IP command indicates that IP could not bind to the board.
- Warnings begin with the text Warning:. A Warning indicates that the NLM encountered something puzzling, but it can complete the requested task. A warning could indicate a lack of system resources to accomplish a nonessential part of the task, or it could indicate abnormal, possibly illegal, inputs from the command line.
- Informational messages begin with a string identifying the entity providing the information (for example, IP: for the IP protocol or IPTunnel: for the IP Tunnel NLM). Informational messages display status information at the system console. Typically, they provide feedback about command-line parameters, so the console operator can verify that the NLM completed the expected operation.
- Alerts begin with the date, the time, and the text 0.0.0 TCP/IP:. (For example, "1/4/91 `10:47am: 0.0.0 TCP/IP: Unbound from 129.47.6.40.`") TCP/IP sends system alerts to NetWare to inform the system administrator of events that are not associated with a particular console command. NetWare displays the alerts on the system console and, in some cases, writes them to the system error log file. Alerts do not necessarily indicate a problem.

The TCP/IP messages in this appendix are listed alphabetically. Other NetWare messages are described in the NetWare System Messages manual. If the explanations given in this appendix and throughout the rest of this guide do not solve or isolate a persistent problem, contact your Novell authorized reseller.

Errors

<*IP address*> is a local address. I cannot talk to myself.

Source: Issued by IPTUNNEL.LAN.

Explanation: A peer parameter has been given to the IP tunnel that indicates that a local address is an IP tunnel peer. This is not legal.

Action: Do not use a local address as an IP tunnel peer.

<*IP address*> is already being used by another interface. Each interface must have its own unique IP address.

Source: Issued during BIND IP command.

Explanation: The IP address being bound to this interface is already bound to some other interface in the system.

Action: Allocate a different IP address for this interface and use it in this BIND command line. Be sure the IP address has the correct network value for the interface's network.

<*IP address*> is illegal. Must be class A, B, or C.

Source: Issued during BIND IP command.

Explanation: The "addr" parameter carried an IP address that was not a legal class for a local address. The first byte of the IP address must be less than 224, decimal.

Action: Correct the IP address.

<IP address> is not currently a local IP address.

Source: Issued by IPTUNNEL.LAN.

Explanation: The IP tunnel saw a local parameter, but the indicated address is not a local IP address.

Action: Provide a local address for the local parameter, or bind IP to a LAN driver with the indicated address as the "addr" parameter before loading the IP tunnel.

Could not get memory to add <IP address> to the peer list.

Source: Issued by IPTUNNEL.LAN.

Explanation: The system is out of memory. The IP tunnel could not add the indicated peer to the peer list.

Action: Add more memory.

Found '<character>'. Each byte of an IP address must be separated by '.'

Source: Issued from any NLM.

Explanation: The command line contained an IP address in which two bytes were separated by the indicated character rather than the required dot.

Action: Correct the IP address.

The {first | second | third | fourth} byte of the address is illegal.

Source: Issued from any NLM.

Explanation: The command line contained an IP address that contained a syntactical problem in the indicated byte.

Action: Correct the IP address.

There are no network bits set in <IP address>.

Source: Issued during BIND IP command.

Explanation: The "addr" parameter carried an IP address that had no bits set in the network field.

Action: Correct the IP address.

Fatal Errors

<*IP address*> is a loopback network address.

Source: Issued during BIND IP command.

Explanation: The IP address you have specified is reserved for the loopback address.

Action: Obtain a legal IP address from the network administrator and use it for the interface.

<*IP address*> is illegal. Host field is zero or all ones.

Source: Issued during BIND IP command.

Explanation: Zero or all ones in a host part of the IP address is reserved and cannot be assigned to an interface.

Action: Obtain a valid IP address with valid host number from the network administrator.

<*IP address*> is illegal. Subnet field is zero or all ones.

Source: Issued during BIND IP command.

Explanation: Zero or all ones in the subnet number part of the IP address is reserved and cannot be assigned to an interface.

Action: Obtain a valid IP address with valid subnet number from the network administrator.

ARP could not allocate resource tag.

Source: Issued during LOAD TCPIP command.

Explanation: The system is extremely short of memory.

Action: Reconfigure the system to use less memory by loading fewer NLM files, reducing some settable parameters, or add more memory to the system.

Cannot support <*IP address*> with mask <*subnet mask*> because it conflicts with <*IP address 2*> with mask <*subnet mask 2*>, an address IP is already supporting on another interface.

Source: Issued during BIND IP command.

Explanation: The subnet mask that you have specified is illegal.

Action: Review your subnet configurations and rebind the interfaces involved with the correct subnet masks.

Could not determine ARP's protocol ID.

Source: Issued during BIND IP command.

Explanation: ARP cannot determine its protocol ID for the medium IP is being bound to.

Action: This should not normally happen. Contact your Novell authorized reseller.

Could not add interface to routing database.

Source: Issued during BIND IP command.

Explanation: This unlikely error probably indicates that the system is extremely short of memory.

Action: Reconfigure your system to use less memory by loading fewer NLM files, reducing some settable parameters, add more memory to the system, or contact your Novell authorized reseller.

Could not allocate a resource tag.

Source: Issued by IPTUNNEL.LAN.

Explanation: The system is out of memory. The IP tunnel cannot complete initialization.

Action: Reconfigure your system to use less memory by loading fewer NLM files, reducing some settable parameters, add more memory to the system, or contact your Novell authorized reseller..

Could not determine IP's protocol ID.

Source: Issued during BIND IP command.

Explanation: IP cannot determine its protocol ID for the medium it is being bound to.

Action: This should not normally happen. Contact your Novell authorized reseller.

Could not register ARP.

Source: Issued during LOAD TCPIP command.

Explanation: NetWare refused to recognize the ARP protocol module. This probably indicates that some other NLM running on the system has already claimed to be the protocol ARP.

Action: Remove the software that has claimed to be ARP.

Could not register IP (LSL error code=<*value*>).

Source: Issued during LOAD TCPIP command.

Explanation: NetWare refused to recognize the IP protocol module. This probably indicates that some other NLM running on the system has already claimed to be the protocol IP.

Action: Remove the software that has claimed to be IP.

Host part of <*IP address*> is all zeroes or all ones. IP bind failed because the local IP address was illegal.

Source: Issued during BIND IP command.

Explanation: The command attempted to associate an illegal IP address to the interface. The bits not masked by the network mask cannot be all zeros or all ones.

Action: Assign a legal IP address to the interface.

Interface allocation failed.

Source: Issued during BIND IP command.

Explanation: The system is extremely short of memory.

Action: Reconfigure your system to use less memory by loading fewer NLM files, reducing some settable parameters, or add more memory to the system.

This interface, <*IP address*>, is connected to the same network as the interface using <*IP address 2*>.

Source: Issued during BIND IP command.

Explanation: TCPIP prints this message if the network (or subnet, if subnetted) number that is being assigned to the interface <*IP address*> is already assigned to the interface <*IP address 2*>.

Action: Assign different network (subnet) number to each interface, the network/ subnet number is <*IP address*> bits masked with subnet mask of the interface <*IP address*>.

IP could not get enough memory to remember interface.

Source: Issued during BIND IP command.

Explanation: The system is extremely short of memory.

Action: Reconfigure your system to use less memory by loading fewer NLM files, reducing some settable parameters, or add more memory to the system.

IP could not get MLID configuration pointer.

Source: Issued during BIND IP command.

Explanation: IP asked the LAN driver for information about its configuration, but the driver refused.

Action: Do not use that LAN driver. It has not been certified by Novell.

IP could not get MLID control entry point.

Source: Issued during BIND IP command.

Explanation: IP asked NetWare for the control entry point of the LAN driver supporting the board IP has been bound to, but NetWare did not have it.

Action: This should not normally happen. Contact your Novell authorized reseller.

IP is already bound to that board.

Source: Issued during BIND IP command.

Explanation: This command attempts to bind IP to a board and frame type that IP is already bound to.

Action: Do not try to bind IP twice to the same board unless the frame type is different for each binding.

IP must be bound first: no local IP address found.

Source: Issued by IPTUNNEL.LAN.

Explanation: IP cannot find any local IP addresses because IP has not been bound to any drivers.

Action: Bind IP to a LAN driver before loading the IP tunnel.

It is illegal to bind ARP directly to a device.

Source: Issued during BIND ARP command.

Explanation: A BIND ARP console command was issued at the system console. It is not legal to bind ARP directly to a media device. ARP is bound automatically by the BIND IP console command.

Action: Do not try to bind ARP to a LAN driver.

No protocol ID for ARP on that media.

Source: Issued during BIND IP command.

Explanation: ARP could not connect to the medium that IP is being bound to because that medium has no protocol ID for ARP. Normally this error follows the message:

```
Warning: ARP does not recognize the media <name>.
Using defaults.
```

When ARP does not recognize a media type, the ARP protocol ID must be registered manually using the PROTOCOL REGISTER console command, but this has not been done.

Action: Follow directions given by your LAN driver documentation to register the correct protocol ID for ARP before binding IP to that board.

No protocol ID for IP on that medium.

Source: Issued during BIND IP command.

Explanation: IP could not connect to the indicated board because the board medium has no protocol ID for IP. Normally, this error follows the message:

```
Warning: IP does not recognize the media <name>.
Using defaults.
```

When IP does not recognize a medium, the IP protocol ID must be registered manually using the PROTOCOL REGISTER console command, but this has not been done.

Action: Follow directions given by your LAN driver documentation to register the correct protocol ID for IP before binding IP to the board.

System rejected ARP's bind request.

Source: Issued during BIND IP command.

Explanation: NetWare did not allow ARP to bind to the board that IP is being bound to.

Action: This could indicate an excessively complicated network configuration, but it should not normally happen. Contact your Novell authorized reseller.

System rejected IP's bind request.

Source: Issued during BIND IP command.

Explanation: NetWare did not allow IP to bind to the indicated board.

Action: This message could indicate an excessively complicated network configuration, but this message should not normally occur. Contact your Novell authorized reseller.

There are no network bits set in "<*IP address*>".

Source: Issued during BIND IP command.

Explanation: A zero network number cannot be assigned to an IP network.

Action: Use a valid IP address with valid network and host number from the network administrator.

UDP port <*port number*> (<*hex port number*>) is in use.

Source: Issued by IPTUNNEL.LAN.

Explanation: The IP tunnel tried to use the indicated UDP port for IP tunnel traffic, but the port was already in use by something else in the system.

Action: Specify the correct UDP port in the port parameter or unload the application that is using that UDP port.

UDP registration for port <*port number*> failed with error code <*error number*>.

Source: Issued by IPTUNNEL.LAN.

Explanation: The IP tunnel encountered an unexpected error while trying to register with the UDP port.

Action: Please contact your Novell authorized reseller, as this should never happen.

UDP rejected UDP interface version <*version number*>.

Source: Issued by IPTUNNEL.LAN.

Explanation: UDP no longer supports the UDP interface version that the IP Tunnel uses. This can happen only if your versions of IPTUNNEL.LAN and TCPIP.NLM are incompatible.

Action: Use IPTUNNEL.LAN and TCPIP.NLM from the same release of the NetWare operating system.

UDP rejected port <*port number*> (<*hex port number*>) as invalid.

Source: Issued by IPTUNNEL.LAN.

Explanation: UDP did not like the port number you gave the IP tunnel in the port parameter. Port numbers must be less than 65536.

Action: Correct the port number speci*fied in the port parameter so that it is in range.

IP Messages

ARP disabled.

Source: Issued during BIND IP command.

Explanation: The BIND command line contained arp=no.

Action: This is not an error. No action is necessary.

Bound to board <*number*>. IP address <*address*>, net mask <*address*>.

Source: Issued during BIND IP command.

Explanation: IP has been successfully bound to the given board as indicated.

Action: This is not an error. No action is necessary.

Configured as default router for this network.

Source: Issued during BIND IP command.

Explanation: The BIND command line contained `defroute=yes`.

Action: This is not an error. No action is necessary.

Configured for gateway operations.

Source: Issued during LOAD TCPIP command.

Explanation: The command line contained the parameter forward=yes.

Action: This is not an error. No action is necessary.

Interface cost set to <*number*>.

Source: Issued during BIND IP command.

Explanation: The BIND command line contained the cost parameter, setting the cost of the interface to the value indicated.

Action: This is not an error. No action is necessary.

RIP disabled.

Source: Issued during LOAD TCPIP command.

Explanation: The command line contained the parameter rip=no.

Action: This is not an error. No action is necessary.

SNMP traps will be sent to <*IP address*>.

Source: Issued during LOAD TCPIP command.

Explanation: The LOAD command line carried a trap parameter indicating that traps should be sent to a particular IP address.

Action: This is not an error. No action is necessary.

Subnetwork mask adjusted to agree with mask used with <*IP address*>.

Source: Issued during BIND IP command.

Explanation: No netmask parameter was given on the BIND command line, and the IP address used in this command has a network number that is the same as the network number of the IP address of another interface already bound. Subnetworks of the same network should have the same mask, so IP has used the other interface's subnetwork mask.

Action: This is not an error. No action is necessary.

User cancelled request.

Source: Issued during the BIND IP command.

Explanation: User pressed the <Esc> key in response to a request from IP for additional information.

Action: This is not an error. No action is necessary.

Using <*IP address*> as a default gateway.

Source: Issued during BIND IP command.

Explanation: A gate parameter was specified on the command line setting the IP address of the default gateway to use for that interface.

Action: This is not an error. No action is necessary.

Using 'poison reverse' in RIP's split horizon.

Source: Issued during BIND IP command.

Explanation: The BIND command line contained poison=yes.

Action: This is not an error. No action is necessary.

IPCONFIG Messages

Could not create a screen for error messages. <*NetWare error string*>

Source: Issued by IPCONFIG.

Explanation: IPCONFIG had some problems parsing the gateways file, but when it tried to report the problems, it could not create a screen. This normally indicates a chronic memory shortage.

Action: Fix any problems you can find in SYS:ETC\GATEWAYS. You could also reconfigure your server so there's more memory available.

Could not open sys:etc\gateways. <*NetWare error string*>. No static routes configured.

Source: Issued by IPCONFIG NLM.

Explanation: NetWare did not let IPCONFIG open the SYS:ETC\GATEWAYS file for the reason given. The most common reason is "No such file or directory," which simply indicates that you have forgotten to create SYS:ETC\GATEWAYS.

Action: Create SYS:ETC\GATEWAYS or take whatever action is warranted based on the NetWare error indicated.

Line <*line number*>: <*line contents*> Ignoring mask for host <*host name*>.

Source: Issued by IPCONFIG NLM.

Explanation: A mask is not required for a host static route. This route was added, the mask parameter was ignored.

Action: None, if the intent was to specify a static route to the host. If a route to a subnet was intended, replace "NET" with "HOST" on the indicated line, remove the host bits from the address, and reload IPCONFIG.

Line <*line number*>: <*line contents*> Warning: The network address <*net address*> is incorrect. Using address <*address*>.

Source: Issued by IPCONFIG NLM.

Explanation: Host bits were detected in a (sub)network static route. The route was added, the host bits were ignored.

Action: None, if the intent was to specify a static route to the (sub)network. If a route to a host was intended, replace "HOST" with "NET" on the indicated line and reload IPCONFIG.

IPTunnel Messages

Added <*IP address*> to the list of peers.

Source: Issued by IPTUNNEL.LAN.

Explanation: The IP tunnel saw a peer parameter and added the indicated peer to the list of peers.

Action: No action is necessary.

UDP checksums are {enabled|disabled} for transmission.

Source: Issued by IPTUNNEL.LAN

Explanation: The IP tunnel saw a chksum parameter.

Action: No action is necessary.

Using <*IP address*> as the local IP address for tunneling.

Source: Issued by IPTUNNEL.LAN.

Explanation: The IP tunnel saw a "local" parameter on the command line and is using the indicated local address for IP tunnel traffic.

Action: No action is necessary.

Using UDP port <*port number*> decimal (<*port number*> hex).

Source: Issued by IPTUNNEL.LAN.

Explanation: The IP tunnel saw a port parameter and is using the indicated UDP port for IP tunnel traffic.

Action: No action is necessary.

SNMP Messages

<community> is now disabled.

Source: Issued by SNMP NLM.

Explanation: You disabled the indicated community by specifying the community on the SNMP NLM command line without following it with an equal sign.

Action: If the intention was to disable the indicated community, no action is necessary.

<*community*> is now <*string*>.

Source: Issued by SNMP NLM.

Explanation: The SNMP NLM saw a parameter on the command line setting the indicated community to the indicated community name.

Action: No action is necessary.

<*community*> now accepts any community name.

Source: Issued by SNMP NLM.

Explanation: You opened the indicated community to allow it to accept any community name by specifying the community on the SNMP NLM command line followed by an equal sign but with no name following the equal sign.

Action: If your intention was to open the indicated community to any community name, no action is necessary.

SNMP Log Messages

Could not open /etc/snmp$log.bin. Reason: <*NetWare error string*>.

Source: This message could display on the system console any time after the SNMPLOG NLM has been loaded.

Explanation: SNMPLOG could not open the log file. The second line contains the reason NetWare gave for refusing the open request.

Action: This should not normally happen. The NetWare error string may give you some clues as to why this failed. The most likely cause ("No such file or directory") is that the directory SYS:ETC does not exist. NetWare installation creates this directory automatically, so this error indicates that a problem occurred during installation.

TCP/IP Alerts

<*media address*> is using our IP address: <*IP address*>.

Source: This message is a spontaneous system alert from the TCPIP NLM.

Explanation: The ARP protocol has detected another IP node on the network using the local node IP address. Either the local system or the system with the indicated 6-byte hardware address is using the wrong IP address.

Action: Correct the IP address.

Hardware Failure on board <*board number*>.

Source: This message is a TCPIP.NLM spontaneous system alert.

Explanation: The IP address was previously bound to a board in the system. The LAN driver supporting that board has detected a hardware failure for the board and has removed it from service. IP can no longer use that board.

Action: Fix the hardware, then rebind IP to the board.

Unbound from address.

Source: This is a spontaneous system alert issued by TCPIP.NLM.

Explanation: The indicated IP address was previously bound to a board in the system and it has now been unbound. This could be caused by either unbinding IP from the board using an UNBIND console command or unloading the LAN driver that is supporting the interface.

Action: Normally, this is not a problem because the action was taken intentionally by the console operator. Use the BIND command to rebind IP to the board, if desired.

Warning Messages

<*IP address*> not on the same network as <*IP address*>. Nevertheless, IP will assume <*IP address*> is on the local network and will use it as a default gateway.

Source: Issued during BIND IP command.

Explanation: A gate parameter was given on the BIND IP command line, but the gateway indicated did not have the same network number as the interface itself. IP expects to send traffic directly to the gateway; therefore, the gateway must be connected directly to the interface network.

Action: If the indicated IP address for the gateway is incorrect, supply the correct IP address for the gateway. In some unusual configurations, this may not be an error and, in that case, no action is required.

<*name*> was truncated to 17 characters.

Source: Issued by IP Tunnel.LAN

Explanation: The name supplied for the IP tunnel exceeded the maximum length.

Action: None.

'<*token*>' could not be understood as an IP address.

Source: Issued from any NLM.

Explanation: A token was encountered that should have been an IP address, but it does not have the correct format. Other error messages supply additional information to help you identify the nature of the problem.

Action: Correct the IP address.

'<*token*>' ignored following an IP address.

Source: Issued from any NLM.

Explanation: After command line processing completed parsing an IP address, it found some extra characters in the token that it did not understand, so it ignored them.

Action: Remove the extra characters.

'<*token*>' is a duplicate parameter. Overriding previous <name> parameter.

Source: Issued from any NLM.

Explanation: A parameter occurred twice on the command line. The command processor is discarding the previous parameter in favor of the new one.

Action: Remove the duplicate parameter.

'<*token*>' is an invalid interface cost.

Source: Issued during BIND IP command.

Explanation: The cost parameter carried an argument that cannot be understood as a cost value. The cost is a decimal number no less that 1 and no greater than 15.

Action: Correct the cost parameter.

'<*token*>' is an unrecognized parameter.

Source: Issued from any NLM.

Explanation: A parameter was encountered that is not recognized as a legal parameter in the current context. Usually the problem is that the parameter was misspelled.

Action: Correct the parameter.

`<*token*>' is not a valid port value.

Source: Issued from IPTUNNEL.LAN.

Explanation: The value given on the command line is not a legal UDP port value.

Action: The IP Tunnel will use the default port; if this is not desirable, correct the port value.

'<*token*>' is not recognized as either affirmative or negative. Please use 'Yes' or 'No' as the argument to the <*name*> parameter.

Source: Issued from any NLM.

Explanation: A parameter that requires a Yes or No argument had neither.

Action: Use 'Yes' or 'No' as the value of the parameter.

ARP does not recognize the media <*name*>. Using defaults.

Source: Issued during BIND IP command.

Explanation: The ARP module does not recognize the medium IP is being bound to, so it is treating it the same way it treats Ethernet. This could indicate a problem with the LAN driver for that medium or an old version of TCP/IP.

Action: Follow directions given by your LAN driver documentation to allow ARP and IP to use this medium.

Cannot disable ARP: media has no alternative mapping mechanism.

Source: Issued during BIND IP command.

Explanation: The BIND command line had the argument ARP=NO, but IP does not have an alternative address mapping mechanism for that medium.

Action: Do not use ARP=NO for that medium.

Could not add ARP's protocol ID (<*number*>).

Source: Issued during BIND IP command.

Explanation: ARP could not register its protocol ID for the medium IP is being bound to. The number in parentheses is an error code. In itself, this error is not fatal, and it might not be a problem unless the bind fails subsequently.

Action: This could indicate an excessively complicated network configuration, but it should not normally happen. Contact your Novell authorized reseller, if necessary.

Could not add IP's protocol ID (<*error code*>).

Source: Issued during BIND IP command.

Explanation: IP could not register its protocol ID for the medium it is being bound to. The number in parentheses is an error code. In itself, this error is not fatal, and it might not be a problem unless the bind fails subsequently.

Action: This could indicate an excessively complicated network configuration, but it should not normally happen. Contact your Novell authorized reseller, if necessary.

Could not load the streams device /dev/{tcp|udp|ip}.

Source: Issued during LOAD TCPIP command.

Explanation: This unlikely error probably indicates that the system is extremely short of memory. The indicated streams device is not available.

Action: Reconfigure your system to use less memory by loading fewer NLM files, reducing some settable parameters, adding more memory to the system, or contacting your Novell authorized reseller.

Could not start TCP Character Generator service on port 19, or Could not start TCP Discard service on port 9, or Could not start TCP Echo Generator service on port 7.

Source: Issued during LOAD TCPIP command.

Explanation: For some reason the indicated trivial TCP service cannot be started. This probably means that the system is extremely short of memory.

These services are not required for normal operations. The failure to start them is interesting mainly because it usually indicates a serious lack of system memory.

Action: Reconfigure your system to allow more memory for the TCPIP NLM; reconfigure the system to use less memory by loading fewer NLM files, reducing some settable parameters, adding more memory to the system, or contacting your novell authorized reseller.

Could not unload the streams device /dev/{tcp|udp|ip}.

Source: Issued during UNLOAD TCPIP command.

Explanation: The NetWare operating system streams support module did not let TCP delete one of its streams devices. This should not happen.

Action: Please contact your Novell authorized reseller, as this should never happen.

Failed to add IPX's protocol ID.

Source: Issued by IPTUNNEL.LAN.

Explanation: The IP tunnel tried to add IPX's protocol to the system's list of protocols, but it failed.

Action: Please contact your Novell authorized reseller, as this should never happen.

ICMP could not register with IP <*error code*>. Continuing.

Source: Issued during LOAD TCPIP command.

Explanation: The Internet Protocol ICMP tried to register with IP, but it could not.

Action: Please contact your Novell authorized reseller, as this should never happen. It should not affect TCP/IP operations.

Ignoring <*IP address*>: peer is already in the peer list.

Source: Issued by IPTUNNEL.LAN.

Explanation: The IP tunnel saw a peer parameter, but the indicated peer was already in the peer list.

Action: Avoid adding duplicate peers to the peer list.

Ignoring default gateway parameter <*token*> is illegal.

Source: Issued during IP BIND.

Explanation: The specified gateway token could not be interpreted as a valid IP address.

Action: Provide the IP address of the default gateway.

Ignoring the ambiguous parameter `<*token*>'.

Source: Issued from any NLM.

Explanation: A parameter was given on the command line that could be an abbreviation for either of two possible parameters.

Action: Use a longer abbreviation, or do not abbreviate.

IP does not recognize the media <*name*>. Using defaults.

Source: Issued during BIND IP command.

Explanation: IP does not recognize the medium it is being bound to, so it is treating it in much the same way it treats Ethernet. This could indicate a problem with the LAN driver for the medium or an old version of TCP/IP.

Action: Follow directions given by your LAN driver documentation to allow IP to use this medium.

IPLocalAddrCheck returned <*error number*> while checking for local peer.

Source: Issued by IPTUNNEL.LAN.

Explanation: The IP tunnel encountered an unexpected error while checking to determine whether a local address has been used (illegally) as a peer.

Action: Please contact your Novell authorized reseller, as this should never happen.

On <*media name*>, an IP packet size of <*number*> bytes is too small. (In STARTUP.NCF, set Maximum Physical Packet Size to <*number*>.)

Source: Issued during BIND IP command.

Explanation: The LAN driver reported a maximum packet size that IP knows is too small for the medium. Normally, this is because the Maximum Physical Receive Packet Size parameter has not been set large enough to handle packets for this medium, although it could also indicate a defective LAN driver.

Action: Set Maximum Physical Receive Packet Size in STARTUP.NCF to the value indicated. If it is already set to at least the indicated value, do not use this LAN driver, as it has not been certified by Novell.

Registration of UDP <*name*> service failed with error code <*value*>.

Source: Issued during LOAD TCPIP command.

Explanation: One of the trivial UDP test services cannot be set up, indicating an internal failure.

Action: This should not normally happen. Contact your Novell authorized reseller.

RIP not running: could not create polling process.

Source: Issued during LOAD TCPIP command.

Explanation: This unlikely error probably indicates that the system is extremely short of memory.

Action: Reconfigure system to use less memory by loading fewer NLM files, reducing some settable parameters, adding more memory to the system, or contacting your Novell authorized reseller.

RIP not running: could not open UDP port.

Source: Issued during LOAD TCPIP command.

Explanation: RIP could not open the RIP UDP port.

Action: This should not normally happen. Contact your Novell authorized reseller.

Routing module could not allocate resource tag.

Source: Issued during LOAD TCPIP command.

Explanation: This unlikely error probably indicates that the system is extremely short of memory.

Action: Reconfigure system to use less memory by loading fewer NLM files, reducing some settable parameters, adding more memory to the system, or contacting your Novell authorized reseller.

SNMP could not allocate a resource tag.

Source: Issued during LOAD TCPIP command.

Explanation: This unlikely error probably indicates that the system is extremely short of memory.

Action: Reconfigure your system to use less memory by loading fewer NLM files, reducing some settable parameters, adding more memory to the system, or contacting your Novell authorized reseller.

SNMP failed to register client with UDP.

Source: Issued during LOAD TCPIP command.

Explanation: The network management protocol SNMP tried to register with UDP, but it could not.

Action: Please contact your Novell authorized reseller, as this should never happen. It should not affect TCP/IP operations, although network management functions will not be available.

SNMP failed to register TCP/IP trap handler with agent.

Source: Issued during LOAD TCPIP command.

Explanation: This unlikely error probably indicates that the system is extremely short of memory.

Action: Reconfigure system to use less memory by loading fewer NLM files, reducing some settable parameters, adding more memory to the system, or contacting your Novell authorized reseller.

SNMP failed to register 1066 MIB with agent.

Source: Issued during LOAD TCPIP command.

Explanation: This unlikely error probably indicates that the system is extremely short of memory.

Action: Reconfigure your system to use less memory by loading fewer NLM files, reducing some settable parameters, adding more memory to the system, or contacting your Novell authorized reseller.

Some utilities have sockets open to the TCP/IP protocols. TCP/IP service to those utilities will be terminated.

Source: Issued during UNLOAD TCPIP command.

Explanation: TCP/IP is warning you that some utilities still have BSD sockets open to the TCP/IP protocol stack.

Action: You may want to abort the UNLOAD command if you want the utilities to continue to use TCP/IP. You can unload the NLM files that are using the TCP/IP protocol stack to allow them to exit gracefully before you unload the TCPIP.

Subnetwork mask disagrees with mask used with *<IP address>*.

Source: Issued during BIND IP command.

Explanation: The subnet mask that you specified is different from the mask specified for the interface <IP address> that is attached to the same IP network. Make sure that you are configuring subnets of different sizes within an IP network.

Action: If the subnet masks are different by mistake, you should unbind the interface with the wrong subnet mask and bind it again with the correct value.

The streams device /dev/{tcp|udp} is still being used. Some utilities have streams open to the TCP/IP protocols. TCP/IP service to those utilities will be terminated, and SYSTEM INTEGRITY MAY BE COMPROMISED.

Source: Issued during UNLOAD TCPIP command.

Explanation: TCP/IP is warning you that some utilities still have AT&T streams open to the TCP/IP streams devices indicated. If any of these utilities are actively transmitting data, the system could crash if you continue to unload the TCPIP NLM.

Action: You probably want to abort the UNLOAD command to preserve system integrity. You can unload the NLM files that are using the TCP/IP protocol stack to allow them to exit gracefully. Once you have done that, you can try again to unload the TCPIP NLM.

TCP not running: could not allocate timer process.

Source: Issued during LOAD TCPIP command.

Explanation: This unlikely error probably indicates that the system is extremely short of memory.

Action: Reconfigure your system to use less memory by loading fewer NLM files, reducing some settable parameters, adding more memory to the system, or contacting your Novell authorized reseller.

TCP trivial service could not be started.

Source: Issued during LOAD TCPIP command.

Explanation: This unlikely error probably indicates that the system is extremely short of memory. The trivial TCP test services is not available. This is normally not a serious problem in itself because these services are not typically used in normal operations.

Action: Reconfigure your system to use less memory by loading fewer NLM files, reducing some settable parameters, adding more memory to the system, or contacting your Novell authorized reseller.

Unable to register TCP adapter, or Unable to register UDP adapter Unable to register ICMP adapter

Source: Issued during LOAD TCPIP command.

Explanation: TCP or UDP or ICMP BSD socket support could not register with CLIB.

Action: Please contact your Novell authorized reseller, as this should never happen.

Unexpected failure from TCP registration: error code <*value*>.

Source: Issued during LOAD TCPIP command.

Explanation: Tried to register with IP, but the registration failed unexpectedly with the indicated error code.

Action: Please contact your Novell authorized reseller, as this should never happen.

GLOSSARY

This glossary includes a list of abbreviations. Refer to the *NetWare Concepts* manual for a more extensive glossary. This glosary contains terms used in this guide, and their definitions.

Address Resolution Protocol (ARP)

The Internet Protocol that dynamically identifies a high-level Internet address based on a known low-level physical hardware address. ARP broadcasts across a single physical network, but only on networks that support hardware broadcasts.

Agent

In the client-server model, the agent is the part of the system that performs information preparation and exchange on behalf of a client or server application.

Attached Resource Computer network (ARCnet)

A high-speed, multivendor network.

Bridge

A bridge usually functions at the physical and datalink layers (it forwards packets received from a remote system from one cable to another). It reconditions signals like a repeater, but a bridge is more intelligent. A bridge can look at a data frame and, by reading the physical address, determine whether the frame should be passed to the adjacent segment. This lets administrators isolate local traffic to their respective segments and still support communication between segments. A bridge stores and forwards complete packets while a repeater forwards electrical signals.

Defense Data Network (DDN)

DDN refers to the MILNET and ARPANET, and the TCP/IP protocols they use. More literally, it is the MILNET and associated parts of the Internet that connect military installations.

Domain Name System (DNS)

A DNS server has network addresses for each computer on a network. A PC must call a name server to get address information. A DNS server translates symbolic names (for example, Bronco) into IP addresses.

Encapsulate

To enclose one complete entity (for example, a UDP datagram or a TCP segment) in the data area of a lower-layer entity such as an IP packet.

File Transfer Protocol (FTP)

The Internet standard, application-level protocol for transferring files from one computer to another. Usually FTP is implemented in application programs using the Telnet and TCP protocols. The server side requires a client login and password before it honors requests.

Gateway

A special purpose computer that connects networks and routes packets from one network to another. A gateway translates all seven protocol layers between computers with different upper-layer protocols, operating systems, and network media types. "Gateway" can also be used as another name for a router.

Hop count

A measure of distance between two points in the Internet. Each hop count corresponds to one router separating a source from a destination (for example, a hop count of "3" indicates that three routers separate a source from a destination).

Host Name Database

Contains the names, aliases (optional), and network addresses of the computers on your network.

internet (not capitalized) or internetwork

Physically, a collection of packet-switching networks connected by routers along with protocols that let them function logically as a single, large, virtual network. Compare Internet.

Internet (capitalized)

Internet refers specifically to the DARPA Internet and the TCP/IP protocols it uses. The Internet is the collection of networks and routers (including the ARPANET, MILNET, and NFSnet) that uses the TCP/IP protocol suite and functions as a single, cooperative virtual network. The Internet provides universal connectivity and three levels of network services: unverified, connectionless packet delivery; reliable, full-duplex stream delivery; and application-level services such as electronic mail. Compare internet.

Internet Control Message Protocol (ICMP)

An integral part of the Internet Protocol (IP) that handles error and control messages. Specifically, gateways and hosts use ICMP to send reports of problems about datagrams back to the original datagram source. ICMP also includes an echo request/reply used to test whether a destination is reachable and responding.

Internet Protocol (IP)

The standard network-layerprotocol that provides connectionless, unreliable datagram delivery. It is "connectionless" because all prackets are transmitted independently of other packets. It is "unreliable" because packet delivery is not guaranteed.

IP address

The Internet Protocol address is a network level (level three of the OSI networking reference model) address assigned to each system in a TCP/IP network. It is four bytes long.

IP host address

An IP host address is a part of the four-byte IP address. The IP address can be divided into two logical parts: an IP network address and a local host address. The IP host address is unique for every node on a single network.

IP network address

An IP network address is a part of the four-byte IP address. The IP address can be divided into two logical parts: an IP network address and a local host address. The IP network address is the same for every node on a single network. The IP network address facilitates routing between Internet networks.

Link Services Layer (LSL)

Routes packets between LAN boards with their multiple link interface drivers (MLIDs) and protocol stacks. The LSL maintains LAN board, protocol stack, and packet buffer information.

Management Information Base (MIB)

SNMP lets TCP/IP-based network management clients exchange information about the configuration and status of nodes on a TCP/IP-based internetwork. The information available is defined by a set of "managed objects" such as TCP, IP, and ICMP statistics. The subset of managed objects making up the TCP/IP portion of the MIB is maintained by each TCP/IP node. The MIB is a virtual information store.

Media Access Control (MAC)

The MAC is the sublayer in between the physical and datalink layers for controlling the use of the network hardware.

Multiple Link Interface Driver (MLID)

Accepts multiple protocol packets. When an MLID device driver receives a packet, the MLID does not interpret the packet; it copies identification information and passes the packet to the Link Support Layer (LSL). MLIDs (LAN drivers) are either supplied by Novell, the network board manufacturer, or a third-party supplier (for example, a university).

Network Information Center (NIC)

NIC, located at SRI International, is the central authority that assigns all Internet addresses.

Open Data-Link Interface (ODI)

A set of specifications that defines the relationships between one or more protocol stacks, the Link Services Layer (LSL), and one or more Multiple Link Interface Drivers MLIDs. These specifications allow multiple communications protocols such as IPX/SPX, TCP/IP, and AppleTalk to share the same driver and adapter.

Packet InterNet Groper (PING)

A program used to test the accessibility of destinations by sending them an ICMP echo request and waiting for a reply.

Packet receive buffers

Packet receive buffers are areas in the file server memory set aside to temporarily hold data packets arriving from the various network stations. The file server holds these packets in buffers until the server is ready to process them and send them to their destination on the network.

Protocol element

A protocol-level command for performing FTP operations on the protocol level. After you have established an FTP session, you can enter REMOTEHELP from the FTP prompt. FTP then displays a list of FTP protocol elements recognized by the FTP server (for example, USER, PORT, XEXC). Most FTP servers support a subset of available protocol elements. Elements that begin with the letter X are experimental. Refer to RFC 959 for additional information about FTP protocol elements.

Protocol stack

The software modules that take data from an application and transform or encapsulate it for transmission across a network. The stack may have several layers of modules, as shown in Figure B-1. Each layer provides services to the layer above; each layer requests services from the layer below. Examples of protocol stacks are IPX, TCP/IP, and OSI.

Repeater

A repeater functions at the physical layer. It indiscriminately passes all signals from one segment to another. It reconditions each signal to extend the distance between two hosts beyond the maximum segment length for any particular hardware. The media type must be the same (for example, Ethernet to Ethernet).

Reverse Address Resolution Protocol (RARP)

The Internet Protocol used by a PC at startup to find its Internet address from its hardware address. The PC broadcasts a request that contains its physical hardware address and a server responds by sending the PC its Internet address.

Router

A router uses a packet's IP destination address to determine which network or network segment is its destination. A router can find the best way to transmit data between networks. This may involve hops across multiple networks. Routers use special protocols that let them talk to each other and advise their peers of available routes, hop counts, time to destination, and router maintenance. Upper-layer protocols must be the same.

Server

A node that provides services to other nodes on the network. A server can be a print server and provide print spooling and printer access for other nodes on the network.

Simple Mail Transfer Protocol (SMTP)

The standard protocol for sending and receiving electronic mail. SMTP provides specifications for mail system interaction and control message formats.

Simple Network Management Protocol (SNMP)

SNMP is the most popular network management protocol in the TCP/IP protocol suite. SNMP lets TCP/IP sites exchange information about the configuration and status of a node's TCP/IP protocol stack.

Transmission Control Protocol (TCP)

A virtual circuit connection-oriented protocol.

TCP/IP Transport software

Software residing in the operating system, including a set of device drivers. These drivers have an interface with the MLID/ODI drivers at the bottom layer, and support the Berkeley socket APIs at the top layer. .

Stream

A stream is a full-duplex connection between a user's task and a device. The stream can include an encapsulated processing module. The TCP/IP stream service does not support structured data streams.

Telnet

The Internet-standard terminal access application-level protocol. Telnet supports character terminals, block terminals (for example, DEC VT220), and graphics terminals. It is used for remote login on an internet network.

User Datagram Protocol (UDP)

An unverified (usually called unreliable) connectionless transport protocol.

Workstation

A node that requests services from other nodes on the network, but does not provide services to other nodes. Some nodes that provide services can be both workstations and servers, but in the context of providing services, they are servers.