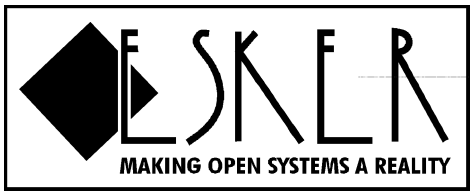
USER MANUAL

Tun KERNEL

TCP/IP Stacks for MS-DOS

Version 8.00



In continuing efforts to improve our products, keep up with current technology, and incorporate user's suggestions, ESKER may change the look and the characteristics of the product described in this manual at any time.

ESKER makes no warranty of any kind, expressed or implied, with regard to the contents of this manual. This is a purely technical document and does not represent a contractual obligation of any kind. ESKER shall not be held liable for incidental or consequential damages in connection with the use of the programs described herein.

PREFACE

Tun KERNEL for MS-DOS is a set of resident (TSR) programs implementing the TCP/IP protocol, it also incorporates the necessary tools for the configuration and testing of the protocol.

Tun KERNEL for MS-DOS is one of a range of complementary packages including Tun EMUL, Tun TCP, Tun MAIL and Tun SQL (see table below).

	WINDOWS	MS-DOS
Tun KERNEL	TCP/IP protocol stacks for Windows	TCP/IP protocol stacks for MS-DOS (TSR)
Tun TCP	TCP/IP applications for Windows (NFS, LPR, LPD, FTP, TELNET, TAR ...)	TCP/IP applications for MS-DOS (NFS, Printer Sharing, FTP, TELNET, TAR ...)
Tun EMUL	Comprehensive terminal emulator for Windows (asynchronous emulation, IBM3270, IBM5250)	Comprehensive terminal emulator for MS-DOS (asynchronous emulation)
Tun MAIL	Comprehensive TCP/IP E-Mail	N/A
Tun SQL	ODBC drivers for the Client-Server mode under TCP/IP (Oracle, Informix, Sybase RDBMS)	N/A

Tun KERNEL for MS-DOS is delivered as standard with the package Tun PLUS which incorporates all the above software. It is also delivered with the Tun TCP and Tun EMUL packages when they are purchased separately.

Tun KERNEL for MS-DOS can be installed independently of these other software packages, however the installation program incorporated in Tun PLUS, Tun TCP, Tun EMUL, Tun MAIL and Tun SQL includes the option to install Tun KERNEL itself.

TABLE OF CONTENTS

CHAPTER 1 - INSTALLATION

PACKAGE CONTENTS

Please make sure your Tun KERNEL for MS-DOS package contains the following:

- Tun KERNEL for MS-DOS**user manual**
- 2 **program disks**: 3.5" (1.44 Kb) labeled"**TCP/IP for DOS**"
- User license
- Sealed envelope containing a serial number, activation key, and a key disk (if applicable)
- Miscellaneous technical bulletins (if applicable)
- Registration card

> **Note:** Opening the sealed envelope indicates that you have accepted the terms and conditions of the User license (indicated on the sealed envelope) for using Tun KERNEL.

HARDWARE REQUIREMENTS

To integrate a PC running Tun KERNEL into a TCP/IP network, you need the following equipment :

- a 100% PC AT 286 compatible micro-computer or higher (ISA, EISA, or MCA bus), with an 84, 101, or 102-key keyboard
- 640 Kb RAM minimum (4 Mb on Windows PCs)
- a Network Interface Card
- a hard disk with approximately 3 Mb free space
- MS-DOS 3.30 or higher

BEFORE YOU BEGIN

Before installing Tun KERNEL, check to see if either the environment or the Network Card you intend to use require a special configuration procedure.

The **Appendices** in this manual address some of the circumstances which may require special attention. In particular, procedures are described for the installation of Tun KERNEL in multi-protocol environments:

- TCP/IP with Netware 3.12 and higher
- TCP/IP with LAN Manager
- TCP/IP with Windows for Workgroups and NetWare

PRODUCT INSTALLATION

It is not necessary to follow the instructions below if you have obtained Tun KERNEL as a part of the Tun PLUS, Tun TCP, or Tun EMUL packages, as the installation procedure for these packages will handle the installation of Tun KERNEL.

To install Tun KERNEL on a PC's hard disk, follow these steps:

1. Load the mouse driver if you would like to be able to navigate within the installation menus with a mouse.

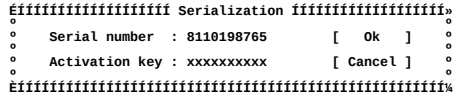
2. To install the Tun KERNEL programs and files, insert the floppy disk "**TCP/IP for DOS 1**" into the **A:** drive and type:

   ```
   A:\> INSTALL
   ```

3. Move through the fields using the mouse, arrow keys, or the <Tab> key. Press the space bar to place an "X" in the field for the appropriate drive.

- Choose the floppy drive containing the program
- Change the destination directory if necessary
- Press the <Enter> key or click on <OK> to continue

4. Next, enter the product's serial number and activation key:



The serial number and activation key can be found in the sealed envelope of your Tun KERNEL package. Click on <OK> or press <Enter> to begin installing the program files.

5. The installation program will prompt you to insert the program floppy disks. Click on <OK> or press <Enter> after inserting each disk. During installation, a window will display the names and directories of the files as they are extracted and installed.

6. The installation is now complete. Click on <OK> to return to the DOS prompt.

TUN KERNEL CONFIGURATION

Tun KERNEL for MS-DOS is delivered with a an interactive MS-DOS program (**TUNKRNL.EXE**) for configuring the product. After installing the program files, TUNKRNL.EXE must be run in order to set up the various program features.

DIRECTORY STRUCTURE AND INSTALLED FILES

The default installation procedure creates the directory \TUN\KERNELD, and installs the following files:

Tun KERNEL Supervisor program	
TUNKRNL.EXE	Tun KERNEL Supervisor Program
*.TLG	Language files used by the Supervisor
ACTIF.TCP	Current Supervisor settings
DTALOGO.DAT	Data file for graphical screen display
TCP/IP kernel	
ETHTCP.EXE	TCP/IP kernel for Ethernet
TOKTCP.EXE	TCP/IP kernel for Token Ring (using ODI drivers)
BOOTP.EXE	Startup utility
SNMPD.EXE	SNMP Agent (TSR)
ESTCPIP.SYS	Driver containing IP parameters
ESTCPIF.SYS	Driver containing interface parameters
PING.EXE	Network test utility
IPCONF.EXE	Utility for dynamically modifying IP parameters
IFCONF.EXE	Utility for dynamically modifying interface parameters
TELL.EXE	Utility for dynamically modifying the IP address
TCPSTOP.EXE	Unloads the TCP/IP kernel
TERMIN.COM	Unloads the packet driver
HOSTTAB	Host table
SERVICES	Services table
AUTOTCP.BAT	Batch file for starting the TCP/IP kernel
PACKET.*	Packet driver database
NDIS.*	NDIS driver database
ODI.*	ODI driver database
CLARKSON*.*	Clarkson packet drivers
DRIVERS*.*	Drivers for dual-protocol coexistence

CHAPTER 2 - INTRODUCTION TO Tun KERNEL

WHAT IS TCP/IP?

TCP/IP is a protocol suite used for connecting computer networks, and for exchanging data between host machines. **TCP** stands for **Transmission Control Protocol**; **IP** stands for **Internet Protocol**. TCP/IP is a standard that is described in its entirety in a set of documents called **RFC**s (Requests For Comment).

Today, TCP/IP is the most widely used protocol for interconnecting heterogeneous machines, both locally and across great distances.

Many large networks use the TCP/IP protocol, notably the **DARPA Internet** (Defense Advance Research Projects Agency Internet). Universities, businesses, and public agencies regularly use TCP/IP to access the gigantic **INTERNET** network.

The strength of TCP/IP lies in the **standardization** of information transmission, as well as the **specification** of numerous powerful application tools: Terminal emulation, File transfer, Messaging, Remote command execution, Network administration...). The TCP/IP protocol suite includes a complete set of advanced API calls. The set of utilities associated with TCP/IP is often referred to as **ARPA Utilities**.

Today, with a widely-installed and commercialized base, TCP/IP is in a position to replace the OSI (Open Systems Interconnection) reference model as the "standard" for network communications.

WHAT IS TUN KERNEL?

Tun KERNEL for MS-DOS provides a complete TCP/IP implementation under MS-DOS, including the following protocol layers:

- TCP (Transmission Control Protocol)
- UDP (User Datagram Protocol)
- ICMP (Internet Message Protocol)
- IP (Internet Protocol)
- ARP (Address Resolution Protocol)

The communication layers in Tun KERNEL are contained in a MS-DOS Terminate-and-Stay-Resident (TSR) program of about 70 Kb. The TSR redirects MS-DOS software interrupts between 0x60 and 0x69, offering an interface for programmers who wish to integrate the TCP/IP kernel with their programs.

Tun KERNEL's TCP/IP kernel is capable of up to 14 simultaneous TCP connections.

In addition, the TCP/IP kernel is able to resolve destination addresses through the use of either a **name server** or local **host table** (i.e. C:\TUN\KERNELD\HOSTTAB).

> **Note**: Tun KERNEL includes a standard WinSock DLL for using MS-DOS resident TCP/IP from within MS Windows. Any third-party Windows application written to the WinSock standard can access the TCP/IP services provided by Tun KERNEL.

NETWORK INTERFACE CARDS

In order to be able to use the vast majority of **Network Interface Cards** available on the market today, Tun KERNEL's TCP/IP kernel uses the *de facto* **packet driver** standard. Loaded before the kernel, packet drivers are small resident programs specific to the type of network card in the PC. The packet driver handles the interface between the TCP/IP kernel and the physical network (LAN) card.

The advantage to this system is that one kernel can interface with many different types of LAN cards. Since it is based on an accepted standard, the product is adaptable to new LAN cards not foreseen at the time of purchase.

The packet drivers delivered standard with Tun KERNEL are **freeware**, developed by Clarkson University (USA), and now distributed by **Crynwr Software**. In addition, network card manufacturers often supply packet drivers with their cards.

In order to use **ODI** drivers (Novell Netware) and **NDIS** (Microsoft), Tun KERNEL provides converters to adapt these environments to the standard needed by Tun KERNEL. These converters may be used to set up configurations with multiple protocols over a single LAN card.

USING THE TUN KERNEL SUPERVISOR

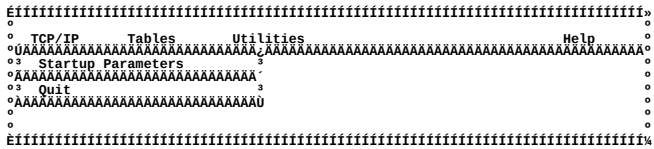
Tun KERNEL for MS-DOS is configured and maintained through the **Supervisor** program, TUNKRNL.EXE. Following a series of menus, users are able to set the parameters for their network interface card, TCP/IP kernel, and resource sharing of printers and Network File Systems.

The Supervisor provides a front-end for configuring the batch files used to load the resident programs in MS-DOS, or automatically in the AUTOEXEC.BAT.

To use the TCP/IP Supervisor program, type the following command at the MS-DOS prompt:

```
C:\TUN\KERNELD> TUNKRNL
```

The following figure shows the main menu of the Tun KERNEL Supervisor.



Most options in the main menu contain a series of sub-menus used to configure all the elements of TCP/IP network resource sharing.

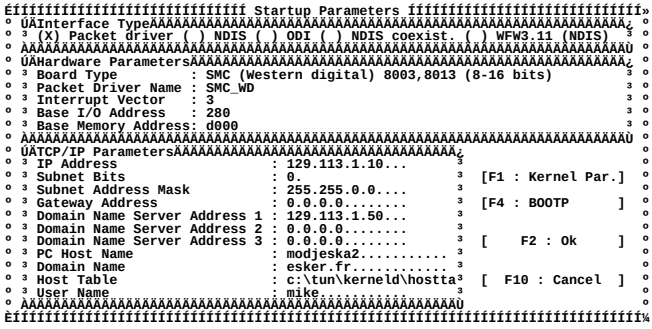
> **Note**: Navigation through all Tun KERNEL menus may be performed using a mouse or other pointing device, as long as the appropriate driver has been loaded before entering the program.

CHAPTER 3 - CONFIGURING Tun KERNEL

TCP/IP CONFIGURATION ON A PC

Run the program **TUNKRNL.EXE** to configure the TCP/IP kernel.

To begin configuring the TCP/IP kernel, select **Startup Parameters**. The following screen will be displayed:



The Startup Parameters menu contains three windows that display the current settings for the **Network Interface Type**, **Hardware Parameters**, and **IP** information. Parameters relating to TCP/IP kernel startup l may be viewed and modified by clicking the button **[F1 : Kernel Par.]** or by pressing **<F1>**.

Below is a description of the fields shown in the figure above.

Interface Type

The first field presents various types of networks protocols that may be used to manage the communication between network cards and the higher levels of applications software.

The **Interface Type** depends on both the network card and any potential coexistence between Tun KERNEL and other network protocols.

Driver Type	Comments
Packet Driver	Used when no other networking software will be running on the PC (single protocol installation). A packet driver must exist for the intended LAN card (either supplied with Tun KERNEL or available from the manufacturer)
NDIS	Used when no other networking software will be running on the PC (single protocol installation). May be used when no packet driver is available (almost all network cards support the NDIS protocol)
ODI	Used when the PC is already configured to run Novell Netware (multiple protocol installation).
NDIS w/ coexistence	Used when the PC is already configured for another network product using NDIS (such as LAN Manager)
WFWG	Used when the PC is already running Windows for Workgroups and the network card has been configured.

Note: In general, the most desirable solution in terms of simplicity, speed and memory use is to use packet drivers

Select one of the previous interface types using either the mouse or the arrow keys, and confirm your choice by pressing <Enter>.

Hardware Parameters

The fields given in the Hardware Parameters section will vary depending on your choice of Network Interface.

Board Type

This first field contains the name of the LAN board used for network connection. Scroll through and select the name of the driver for your board by clicking in the field with the mouse or using the space bar.

When you select the **Board Type**, suggested default values for the remaining fields are filled in automatically. The driver databases contain the list of supported LAN cards and their default parameters.

> **Note**: For proprietary reasons, not all of the packet drivers listed in the PACKET.DBA are delivered with Tun KERNEL. These drivers are delivered by the LAN card manufacturer, and should be copied into the CLARKSON directory.
>
> Likewise, the NDIS.DBA and ODI.DBA files contain information about drivers not delivered with Tun KERNEL. Copy these drivers from the manufacturer's floppy disk into the drivers subdirectory.

The table below lists the Packet Drivers delivered with Tun KERNEL:

Manufacturer	Network Interface Card
3COM	3C501, 3C503, 3C505, 3C507, 3C509, 3C523
Allied-Telesis	AT1500, AT1700
AQUILA Comm	Arlan 450
ARCNET	ARCNET
AT&T	starLAN 1/10
BICC Data Networks	ISOLAN 4110
Cabletron	DNI Exxxx series
D-LINK	DE600
David Systems	Ether-T PC/AT
DEC	DE100, DE200, DEPCA Rev E
EAGLE (NOVELL)	NE1000, NE2000, NE2100, NE2
FUJITSU	EtherCoupler 86965
HP	EtherTwist, Starlan , Thinlan, 27247B, 27252A
IBM	Token Ring
ICL (Nokia Data)	EtherTeam 16
INTEL	EtherExpress
KODIAK	Raven 8, Raven 16, Kombo
MULTITEC	EN-301
MYCRODYNE	EXOS2x5T
Mylex	LNE-390B
NCR	ET-105
NTI 16	NTI 16
RACAL INTERLAN	NI5010, NI5210, NI6510, NI9210, ES3210
SCHNEIDER & KOCH	SK-NET
SMC (WD)	Elite Ultra, WD8003, WD8013
THOMAS-CONRAD	Tcnet
TIARA	LANCARD/E
UNGERMANN BASS	NICPC, NICPS2
XIRCOM	Credit card adapter
ZDS	Z-note / IBM Thinkpad

Hardware settings

You may need to adjust the hardware settings such as **Interrupt vector (IRQ)**, **Base I/O Address**, and **Base Memory Address** to match your board.

When configuring your LAN card, it is important to avoid address and interrupt conflicts with other boards installed in your PC. If you select an interrupt vector that is used by another board, the kernel will most likely not be able to load.

IRQ 5 is a good interrupt to use for 8-bit LAN cards, as it corresponds to a rarely-used second parallel port. For 16-bit LAN cards, IRQ values above 9 are usually good to use.

You may scroll through the possible values in the Base I/O and Base Memory Address fields by clicking in the field with the mouse or using the space bar.

TCP/IP Parameters

The TCP/IP parameters section contains all the information necessary for integrating a PC into a TCP/IP network.

IP Address

This field contains the IP address you wish to assign to the PC, and therefore must not be used by another machine on the network.

An IP address is composed of four fields of numbers, separated by a period (.). For example: *124.131.120.111, 124.131.120.10*.

In a same network, IP addresses must be set in "harmony". As a general rule, the first three fields are the same in small networks, with the last field unique to each host. For example, a UNIX server could have the address 124.131.124.**100**, and a PC in the same network could use 124.131.124.**101**.

In a large network, it is preferable to obtain an IP address from the system administrator. (See Appendix IP Addresses for more details.)

Subnet Bits

Used for dividing the host portion of an IP address to indicate the presence of a subnetwork. For small networks (without subnetworks), this value is usually 0; on a larger network, this information should be given by the network administrator.

Subnet Address Mask

This field is filled in automatically according to the value given for Subnet Bits.

Gateway Address

The Gateway Address indicates the IP address of a gateway machine used to forward packets to other networks or subnetworks. The IP address format for this machine is the same as for any other host. If your network does not contain a gateway, enter **0.0.0.0** in this field.

Domain Name Server Address

On larger networks, one or more **Name Servers** may be used to store host names and their corresponding IP addresses. Instead of having host tables on every PC, which can be hard to manage, a **Name Servers** centralizes the information for everyone to share. The IP address format for this machine is the same as for any other host.

Local Hostname

An alias name by which the PC is know to other machines on the network.

Local Domain

Used if the network is divided into Domains, or is part of a larger Domain. This name should be supplied by the network administrator.

Local Host Table

This field specifies the *location* of a **Host Table** on the PC. A Host Table is a list of the names of the machines on the network with their corresponding IP addresses. (It is easier to remember the *name* of a host than its IP address.)

By default the Local Host Table is called "HOSTTAB", and is located in the directory where Tun KERNEL was installed.

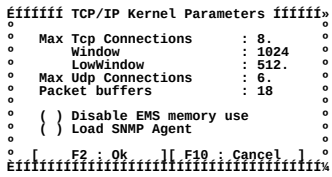
Host Tables may be edited using the Supervisor program, with the **Tables** option.

User Name

Applications such as RXPRINT, RSH and TAR use this field in order to obtain access rights on UNIX servers. The name given here must correspond to a real user name on the UNIX servers that will be used for these types of programs.

Kernel Customization

The **Kernel Parameters** screen is used to customize the TCP/IP resident itself. Press <F1> to modify connections and buffer information:



Max TCP Connections

This field determines the total number of process connections (sockets) that may be established by the PC. For example, each terminal emulation session requires one socket, each X11 window requires one socket, each remote printer requires one socket, and so on. Simply add up the anticipated number of processes, and enter that number into the **Max TCP Connections** field. Of course, the higher the number of connections, the more memory required for the kernel.

Max UDP Connections

This field specifies the maximum number of simultaneous logical UDP links usable by the TCP/IP kernel. In fact, only NFS and *address lookup* on a name server require this type of connection. The higher the value, the more memory needed by the kernel.

Window and LowWindow

These two options regulate the flow of data received by the TCP/IP kernel. Increasing their values may help make network performance more regular, but not necessarily faster. In general, the optimal values are approximately 2048 for **Window**, and 512 for **LowWindow**.

Packet Buffers

This value refers to the number of packet buffers allocated within the TCP/IP kernel. The higher the number of packets, the better the flow of data. The optimal value is 1.5 times the quantity of sockets. The higher the number of packets, the more memory used by the kernel.

Disable EMS memory use EMS

When this option is selected, the kernel will no longer use EMS memory to allocate buffers, thereby resolving possible memory conflicts. On the other hand, the kernel uses more conventional memory.

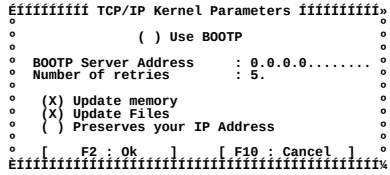
Load SNMP Agent

When this option is selected, an SNMP agent (MIB II-compliant) will be loaded as a TSR after the kernel has been started. This TSR allows remote administration of the PC by an SNMP supervisor. It uses approximately 36 Kb of conventional memory.

BOOTP

BOOTP is a utility that retrieves IP parameters, including IP address and gateway address, from a remote server rather than reading them locally. BOOTP is often used as a tool to help administrators in large networks manage PC's TCP/IP from a central location.

By pressing <F4> or clicking on the button **[F4 : BOOTP]**, you may configure Tun KERNEL to run BOOTP.EXE before loading the TCP/IP kernel.



Use BOOTP

An "X" in this box indicates whether or not BOOTP should be run before starting the TCP/IP kernel.

BOOTP Server Address

This field specifies the IP address of the BOOTP server. The address is optional however: if no address is given (default 0.0.0.0), then BOOTP emits a general broadcast message and waits for a response.

Number of retries

This parameter specifies how many times the BOOTP program should broadcast its request if it receives no reply from a server.

Update memory

This option causes the BOOTP response to update the IP parameters currently in memory. Any configuration performed using the Tun KERNEL Supervisor program remains unchanged, and the PC will use its previous parameters next time it is rebooted.

Update files

This option will allow BOOTP to update the configuration files (ESTCPIP.SYS & ESTCPIF.SYS) as configured by the Tun KERNEL Supervisor.

Preserve your IP address

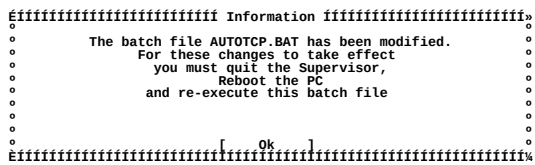
When this option is selected, the IP address currently in memory (as read from ESTCPIP.SYS) remains unchanged, even if the BOOTP server sends a different one. In this case, a message stating the differences will be displayed.

Note:	More details on configuring BOOTP may be found in the next chapter .

Saving changes

At any time, you may save the changes made to the Startup Parameters screen by clicking on <OK> with the mouse or by pressing <F2>. If you wish to cancel the operation without saving changes, click on <Cancel> or press <F10>. In either case, you will be returned to the previous menu.

If you save changes to the startup parameters, you will receive the message:



Any changes you made will be recorded in the files AUTOTCP.BAT, CONFIG.SYS, ESTCPIP.SYS, and ESTCPIF.SYS.

> **Note**: At this point, you should quit the Supervisor program and reboot the PC.

LOADING THE TCP/IP KERNEL

The procedures described in the previous sections modify a file called **AUTOTCP.BAT**, as well as the system files **ESTCPIF.SYS** and **ESTCPIP.SYS**.

The file **AUTOTCP.BAT** is a batch file that loads the packet driver and the TCP/IP kernel, either automatically in the AUTOEXEC.BAT file, or directly from the MS-DOS prompt.

In order to be able to use other options in the Supervisor program, **AUTOTCP.BAT** must first be run from MS-DOS in order to load the TCP/IP kernel.

> **Note**: To unload the TCP/IP resident programs from memory, run this batch file with the option "**stop**".

Here is a sample AUTOTCP.BAT file generated by the Supervisor program:

```
@echo off
if "%1" = "" goto start
if "%1" = "START" goto start
if "%1" = "start" goto start
if "%1" = "STOP" goto stop
if "%1" = "stop" goto stop
:error
echo "Usage : %0 START|STOP
goto end
:start
C:\TUN\KERNELD\CLARKSON\SMC_WD 0x62 5 300
C:\TUN\KERNELD\ETHTCP -t 8 -u 3  -p 12
goto end
:stop
C:\TUN\KERNELD\TCPSTOP.EXE
C:\TUN\KERNELD\TERMIN 0x62
:end
```

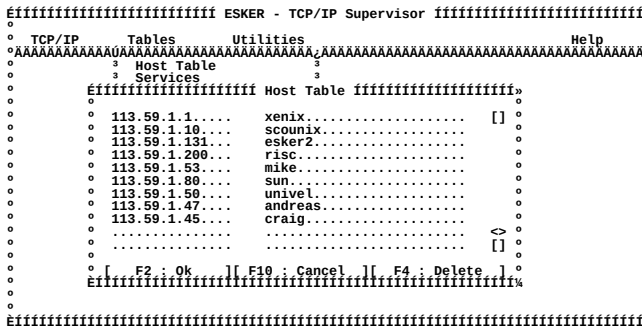
For more information on using ETHTCP.EXE, see the **Reference Guide** chapter.

TCP/IP TABLES

Editing a Host Table

Host tables are used to assign alias names to the hosts on your network. The IP address of a particular host is listed in the left column, and the alias name that you wish to assign is in the right column.

Figure 3-5 shows the Host Table editing screen, (which is displayed after selecting **Host table** from the **Tables** sub-menu):



After assigning an alias name to a host, you may then refer to that host by its name (rather than its IP address) when using terminal emulation, network utilities such as **ping**, or assigning NFS drives.

Editing the Services table

The **Services** table lists software port numbers for well-known TCP services. When a client process such as **TNVT52** or **FTP** requests a server process, it looks up in the services table to know what software port number to address on the server.

Usually this table does not need to be modified, but rather serves as a reference to programmers who wish to interface their programs with known services. It is also possible to add new services to the table.

TESTING NETWORK CONNECTIONS

The **Utilities** option provides menu access to network testing and dynamic management tools.

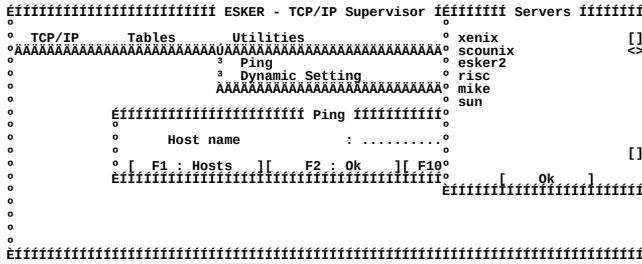
> **Note**: These functions can only work if the TCP/IP kernel has already been loaded on the PC.

Testing network access with Ping

The **Ping** utility is used to test network connections between two machines by sending an *ICMP_ECHO* request.

Ping displays a window in which you enter the name of the host you wish to test. You may enter a host name as entered in the *Host Table* or an IP address.

To obtain a list of the host names defined in the Host table (as shown in Figure 3-6 below), click on <Hosts> or press <F1>. You may use the mouse or the up/down arrows to select a host.



After a host name or IP address has been entered into the Host Name field, click on <OK> with the mouse, or press <Enter> or <F2> to send the request.

If the TCP/IP kernel and the network are functioning correctly, you should receive the message "**Host responding**"; if connection was not successful, the message "**Host unreachable**" is displayed.

In case you receive the error "Host unreachable", there are several places to begin looking for the cause of the failure:

1. Check to see that the TCP/IP kernel loads correctly by watching for error messages during startup. Also make sure that a **hard-coded** Ethernet or Token Ring address is displayed when the packet driver is loaded.

2. Make sure that there are no address or IRQ conflicts with the Network Interface Card.

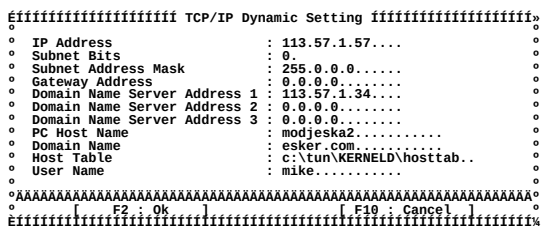
3. Verify that the network cable is properly connected and terminated.

4. Make sure that TCP/IP is running correctly on the destination host (for UNIX servers, try a local "telnet").

Dynamically changing IP settings

The **Dynamic settings** option lets you modify IP parameters while the TCP/IP kernel is loaded without having to reboot the PC after each change. This can be very practical for experimenting with different IP addresses during configuration.

> **Note:** Changes made using this menu option are not permanent, and will be lost when the PC is rebooted.



The fields presented in this window are described in the section **TCP/IP** earlier in this chapter.

Any changes you make in this screen will take effect as soon as you click on <Ok> or press <F2>. To cancel without changing anything, click on <Cancel> or press <F10>.

CHAPTER 4 - USING BOOTP

INTRODUCTION TO BOOTP

BOOTP is a utility for reading TCP/IP configuration information for PCs from a remote server, including parameters such as IP address, Gateways and Name servers. The values supplied by the BOOTP server are then used by the PC's TCP/IP kernel when it is started, thereby being "guaranteed" reliable information. BOOTP may be used for configuring and updating machines with or without hard disks.

An indispensable tool for large networks, BOOTP allows administrators to centralize IP information for PCs on a UNIX server, instead of having each PC contain its own configuration files. Network changes, such as new gateways and name servers, can be easily added to a single file on the BOOTP server. PCs can be guaranteed reliable and up-to-date information because BOOTP is run before TCP/IP.

Here is how BOOTP functions:

1. The BOOTP client (PC) creates a **BootRequest** packet that it broadcasts on the network.

2. A BOOTP server responds with a **BootReply** that contains IP information from **/etc/bootptab.** In addition to the client's IP address, responding server name and configuration file to transfer, a **Bootp** packet also contains a zone called the "Vendor Specific Area" which contains additional information in a particular format, such as Subnet mask, list of Time servers, Gateways and Domain name servers.

3. Configuration files on the PC will be updated by the BOOTP reply if necessary.

SERVER CONFIGURATION

The BOOTP database is contained in the file **/etc/bootptab** on the UNIX host. Follow these steps to configure the BOOTP server:

1. **Edit /etc/bootptab** and add one entry into **/etc/bootptab** for each *bootp* client (for each PC) as in the following example:

```
carter:ip=128.2.11.157:ht=1:ha=0000C0915C2A:sm=255.255.0.0
:\
:ds=128.2.11.200 128.2.11.80:\
:gw=128.2.11.200:\
:ns=128.2.11.80:\
:ts=128.2.11.200:\
:hn:vm=rfc1048:
```

The syntax used in the **/etc/bootptab** file is similar to that used in **/etc/termcap**. The name of the PC is always given as the first variable, other variables are listed below (with mandatory options marked with an asterisk).

VALUE	*	DESCRIPTION
bf		Name of boot file
cs		Cookie Server address
ds		Domain Name Server address
gw		Gateway address
ht	*	Hardware type (Ethernet, Token Ring...). Must be specified before **ha**
ha	*	Physical address of the client's LAN card. This is the number that the BOOTP server uses to respond to a client request
hd		Home Directory
hn		Indicates that the server should send the client's "host name"
im	*	Print server
ip	*	IP address for the client (PC)
lg		Log server
lp		LPR server
ns		IEN 116 Name Server
rl		Resource Location Protocol Server
sm		Subnet mask
tc		Template host (points to another entry in **/etc/bootptab** in order to avoid repeating information
to		Time Offset
ts		Time Server
vm	*	Include the "vendor specific area" according to RFC 1048

Some rules that apply when writing **/etc/bootptab** :

- the hostname (PC) is always the first field
- the **hardware address type** (ht) must be declared before the **hardware address** itself (hd)
- the gateway address must be a bootp server, otherwise there is no response
- specify only the NECESSARY fields (i.e., if using BOOTP with Tun KERNEL only, the home directory and boot file entries are not needed)

2. **Edit /etc/services** and make sure that the *bootps* service is entered in the **/etc/services** file, as shown here:

```
bootps      67/udp  #Boot Server Port
bootpc      68/udp  #Boot Client Port
```

3. **Edit /etc/inetd.conf**. The bootp server also needs to be entered in **/etc/inetd.conf**:

```
bootps  dgram   udp  wait  root  /etc/bootpd bootpd
```

4. The bootp server is automatically started every time a request is received on port 67, and **/etc/bootptab** is read.

USING BOOTP

On the PC, BOOTP exists in the form of an executable, which must be run before starting the TC/IP kernel (either ethtcp.exe or toktcp.exe).

By default, this program makes a BOOTP packet which it broadcasts on the network. If a server recognizes the physical address of the network card that sent the packet, then the information contained in **/etc/bootptab** is returned to the PC. The BOOTP program then updates the PC's TCP/IP configuration (in file and memory).

Details on the specifics for using BOOTP are given in the next chapter.

CHAPTER 5 - REFERENCE GUIDE

INDEX OF FILES AND PROGRAMS

BOOTP.EXE	Startup utility that retrieves IP information for a PC from a remote host.
ETHTCP.EXE	TCP/IP Kernel Terminate-and-Stay-Resident program (Ethernet II frames).
HOSTTAB	Host table.
IFCONF.EXE	Modifies IP parameters (ESTCPIF.SYS).
IPCONF.EXE	Modifies general parameters (ESTCPIP.SYS).
NDIS.CTL	Secondary database for referenced NDIS drivers.
NDIS.DBA	Database containing the list and syntax of the NDIS drivers referenced within Tun KERNEL
ODI.CTL	Secondary database for referenced ODI drivers.
ODI.DBA	Database containing the list and syntax of the ODI drivers referenced within Tun KERNEL.
PACKET.CTL	Secondary database for supported packet drivers.
PACKET.DBA	Database containing the list and syntax of the packet drivers referenced within Tun KERNEL.
PING.EXE	Sends an ICMP ECHO_REQUEST to a host.
SERVICES	TCP Services table.
SNMPD.EXE	SNMP agent (MIB II compliant TSR).
TCPSTOP.EXE	Unloads the TCP/IP kernel from memory.
TELL.EXE	Updates the TCP/IP resident after an IP address change.
TERMIN.COM	Unloads the Clarkson packet drivers.

TOKTCP.EXE TCP/IP Kernel Terminate-and-Stay-Resident program (Token Ring Frames).

TUNKRNL.EXE Tun KERNEL Supervisor / Configuration.

BOOTP BOOTP

Startup utility

Syntax

```
bootp [-m]  [-d] [-v] [-p] [-h ip_address] [-r n]
```

Description

BOOTP is a utility for reading TCP/IP configuration information from a remote server, including parameters such as a PC's IP address, Gateways and Name servers. The values supplied by the BOOTP server are then used by the PC's TCP/IP kernel when it is started, thereby being "guaranteed" reliable information. BOOTP may be used to configure and update machines with or without hard disks.

The advantage to using BOOTP is that IP information for an entire network may be centralized, instead of relying on different configuration files in every PC.

> Note: BOOTP must always be run before the TCP/IP kernel in the PC.

Options for BOOTP that may be given on the command line are given below:

-m Only the parameters in the PC's MEMORY will be updated by the server's response. Any changes will be lost when the PC is rebooted.

-d Only the parameters in the PC's configuration file will be updated by the server's response. Because these changes are stored in a file on disk, they will be kept for future use.

-v (verbose) The parameter values sent by the server will be displayed clearly on the screen.

-n Neither the configuration file nor the parameters in memory are updated. This option is often used in conjunction with **-v**.

-p The BOOTP client (the PC) keeps its current IP address. The currently in use by the PC will not be affected by the address sent by the server.

-h ip_address Specifies the IP address of the BOOTP server. In this case, BOOTP requests will not be treated by any other server.

-r nb_retry Specifies the number of times that a BOOTP request should be re-sent before acknowledging that the server is not responding.

ETHTCP ETHTCP

TCP/IP Terminate-and-Stay-Resident program (Ethernet II frames).

Syntax

ETHTCP [-i 0xnn] [-t nbtcp] [-p nbpacket] [-u nbudp] [-l]

Description

The resident software implements the TCP/IP layers (TCP, IP, ICMP, UDP, ARP...).
ETHTCP must be loaded before other TCP/IP programs, and just after the packet driver. The command TCPSTOP may be used to unload ETHTCP from memory.

If the driver EMM386 has been loaded, ETHTCP allocates is buffers in EMS memory by default. In this case, the size of the resident in conventional memory will always be the same, regardless of how many TCP sockets and packet buffers have been declared.

This TCP/IP application is designed to process ETHERNET II physical frame types. For coexistence with Novell Netware LANs in a Token Ring environment running ODI drivers, use TOKTCP.EXE (the TCP/IP kernel for Token-Ring packets).

The startup options for the kernel are as follows:

-i 0xnn Specifies the software interrupt vector used by applications in order to communicate with the TCP/IP kernel. The value "nn" must be between 61 and 69, and is 0x61 by default.

-t nb_tcp Specifies the number of simultaneous TCP connections. The larger the number, the more memory required for the TSR. The default value is 8.

-p nb_packet Specifies the number of communications buffers within the kernel. The value of "nb_packet" should be at least 1.5 times the number of TCP connections, plus the number of UDP links.

Data flow is smoother when this number is high, but more memory is needed for the TSR.

The default value for this parameter is 20.

> **Note**: In Token Ring networks, raising the number of buffers may help considerably.

-u nb_udp Specifies the number of simultaneous UDP links.

The higher the value for "nb_udp", the more memory required for the TSR. The default value is 5.

> **Note**: NFS uses several UDP sockets, regardless of the number of virtual drives.

-l Forces the TCP/IP resident to allocate resources in conventional memory, even if EMM386 is loaded on the PC.

See also

TOKTCP.EXE, TCPSTOP.EXE

HOSTTAB HOSTTAB

Local host table.

Description

The host table is used to assign alias names for network hosts based on their IP addresses. In practice, it is easier to refer to a host by name than by address.

Each line contains a server and its address. If your network contains a name server, it is probably not necessary to use a local host table.

Example

```
121.131.118.1          xenix
121.131.118.10      scounix
121.131.118.80      sun
121.131.118.200     risc
121.131.118.42      me
```

See also

SERVICES

IFCONF IFCONF

Modifies IP parameters.

Syntax

```
IFCONF interface option [option [...]]

IFCONF /?
```

Description

This utility may be used to change IP parameters used by the TCP/IP kernel, either temporarily or permanently.

Give one of the two following values for the *interface* parameter:

1. **ifcust0** changes certain parameters currently in memory
2. **PATH\estcpif.sys** changes the parameters to the system file ESTCPIF.SYS.

option gives the parameters to change, as well as the new values:

/ip internet_address	Changes the IP address
/subnu number	Changes the number of subnet bits
/submask internet_mask	Changes the subnet mask.
/show	Displays all active parameters
/?	Displays command parameters

> **Note**: After having changed IP parameters, use the TELL command to update the TCP/IP kernel in memory.

See also

IFCONF.EXE, TELL.EXE

IPCONF IPCONF

Changes general TCP/IP kernel parameters.

Syntax

```
IPCONF interface option [option [...]]

IPCONF /?
```

Description

This utility may be used to change a variety of different parameters used by the TCP/IP kernel, either temporarily or permanently.

The following values may be given for **interface:**

1. **$ipcust** dynamically changes parameters currently loaded into memory.
2. **PATH\estcpip.sys** changes the contents of the system file ESTCPIP.SYS.

Option is used to specify which parameters to change, as well as the new values:

/gw internet_address Modifies the IP address of the *gateway*.

/ns internet_address Modifies the IP address of the *name server*.

/ds internet_address Modifies the IP address of the *domain name server*.

/hostname string Changes the network name of the local host.

/dn string	Changes the name of the domain to which the host belongs.
/hosttable string	Changes the location of the default host table.
/username string	Changes the name of the workstation's*user name.*
/window value	Changes the size of the packet transmission and reception window.
/lowwindow value	Changes the minimum size of the transmission and reception window.
/tzname string	Changes the name of the current time zone (default is GMT)
/tzoffset value	Changes the offset with respect to the defined time zone. This value is expressed in minutes, and may be positive or negative (the default is 0).
/show	Displays all active parameters
/?	Displays command parameters

See also

IPCONF.EXE, TELL.EXE

NDIS.CTL NDIS.CTL

Secondary database for the NDIS drivers described in the NDIS.DBA.

Description

This file contains the list of translated-type parameters used in the NDIS.DBA. You may add items to this ASCII file as necessary.

Each line represents a "type", made up of variable number of fields separated by the character "|". The order and content of the fields is a follows:

- Label of the translated type (given in the NDIS.DBA)
- Label of the first parameter
- Translated value of the first parameter
- Label of the second parameter
- ...

Note:

The label of a translated type cannot begin with the letter A, I, M, m, or H.

See also

NDIS.DBA

NDIS.DBA NDIS.DBA

Database for the NDIS drivers referenced in Tun KERNEL.

Description

This file contains the list and syntax of the NDIS drivers used by Tun KERNEL. You may add items to this ASCII file as necessary.

Each line represents a variable number of fields separated by the character "|". The order and content of the fields is as follows:

- Name of the NDIS driver
- List of network cards supported by the driver
- Execution syntax
- Symbolic name of the NDIS driver used in the PROTOCOL.INI
- Name of the NDIS driver loaded into memory
- Type of TCP/IP kernel to use (ETH= Ethernet, TOK=Token Ring). If not specified, the Ethernet kernel will be used
- Number of parameters to be presented on the Tun KERNEL Supervisor screen (maximum 5)
- Label of the first parameter displayed on the Supervisor screen
- Type of first parameter: (I=Integer, H=Integer in hexadecimal notation, A=I/O Address, M=Long memory-type address, m=Short memory-type address, XXX=translated type defined in the NDIS.CTL file.
- Default value of the first parameter
- Keyword for recognizing the first parameter in the PROTOCOL.INI
- Write format (C language) of the first parameter in the PROTOCOL.INI
- not used
- Label of the second parameter
- Second parameter type
- Default value for the second parameter
- ...

Note

The label of a translated type cannot begin with the letter A, I, M, m, or H.

See also

NDIS.CTL

ODI.CTL ODI.CTL

Secondary database for the ODI drivers described in the ODI.DBA.

Description

This file contains the list of translated-type parameters used in the ODI.DBA. You may add items to this ASCII file as necessary.

Each line represents a "type", made up of variable number of fields separated by the character "|". The order and content of the fields is a follows:

- Label of the translated type (given in the ODI.DBA)
- Label of the first parameter
- Translated value of the first parameter
- Label of the second parameter
- ...

Note

The label of a translated type cannot begin with the letter A, I, M, m, or H.

See also

ODI.DBA

ODI.DBA ODI.DBA

Database for the ODI drivers referenced within Tun KERNEL.

Description

This file contains the list and syntax of the ODI drivers used by Tun KERNEL. You may add items to this ASCII file as necessary.

Each line represents a variable number of fields separated by the character "|". The order and content of the fields is as follows:

- Name of the ODI driver
- List of network cards supported by the driver
- not used
- not used
- not used
- Type of TCP/IP kernel to use (ETH= Ethernet, TOK=Token Ring). If not specified, the Ethernet kernel will be used
- Number of parameters to be presented on the Tun KERNEL Supervisor screen (maximum 5)

- Label of the first parameter displayed on the Supervisor screen
- Type of first parameter: (I=Integer, H=Integer in hexadecimal notation, A=I/O Address, M=Long memory-type address, m=Short memory-type address, XXX=translated type defined in the ODI.CTL file.
- Default value of the first parameter
- Keyword for recognizing the first parameter in the NET.CFG
- Write format (C language) of the first parameter in the NET.CFG

- - Label of the second parameter
- - Second parameter type
- - Default value for the second parameter
- - ...

Note

The label of a translated type cannot begin with the letter A, I, M, m, or H.

See also

ODI.CTL

PACKET.CTL PACKET.CTL

Secondary database of the packet drivers described in the file PACKET.DBA.

Description

This file contains a list of the translated parameters used in the PACKET.DBA. Since it is an ASCII file, new information may be easily added.

Each line represents a **type**, with a variable number of fields separated by the character " | ". The order of the fields is as follows:

- Label of the translated **type** (used in PACKET.DBA)
- Name of the first parameter
- Translated value of the first parameter
- Name of the second parameter
- ...

Note

The label of a translated **type** may not begin with the letters A, I, M, m, or H.

See also

PACKET.DBA

PACKET.DBA PACKET.DBA

Database containing the packet drivers referenced within Tun KERNEL.

Description

This file contains the list and syntax of all the packet drivers referenced in the Tun KERNEL menus. Since it is an ASCII file, new packet drivers may be easily added.

Each line represents a packet driver, with a variable number of fields separated by the character " | ". The order of the fields is as follows:

- Name of the packet driver
- List of network cards supported by the packet driver
- Syntax for packet driver startup (C language format)
- Syntax for unloading the packet driver
- Type of TCP/IP kernel to use (ETH= Ethernet, TOK=Token Ring). If not specified, the Ethernet kernel will be used
- not used
- The number of parameters that are asked for in the supervisor program (maximum 5).

- The label for the first parameter, which will be displayed on the configuration screen of the Supervisor
- Data type of the first parameter (I:Integer, H:Integer in hexadecimal notation, A: I/O-type address, M:Long memory-type address, m: Short memory-type address, XXX: translated type defined in PACKET.CTL).
- Default value for the first parameter
- Not used
- Not used
- Not used
- Label of the 2nd parameter
- Type of the 2nd parameter
- Default value for the second parameter
- etc...

See also

TERMIN.COM

PING PING

Sends an ICMP ECHO_REQUEST towards a TCP/IP host.

Syntax

PING ip_address | hostname

Description

PING is sends an ICMP ECHO_REQUEST to a TCP/IP host on the network, and then waits for a response. This utility is used to test the physical and logical connection between a PC and a host server, and therefore can only function after the TCP/IP kernel has been loaded into memory on the PC.

There are two messages generated by the PINGprogram:

1. "**Host Responding**" indicates that the connection is working correctly,
2. "**Host Unreachable**" indicates that connection has not been successful.

Note

There are numerous reasons why a connection might fail. Here are three of the most common:

1. The Network Interface Card is incorrectly installed or configured,
2. The IP address of the PC is incompatible with those of the server or network,
3. Cabling is faulty.

See also

ETHTCP.EXE

SERVICES SERVICES

TCP/IP services table.

Description

The SERVICES file contains a list of the most frequently used services used in a TCP/IP environment.

Each line in the file describes a single service using the following syntax:

```
name        number/protocol        alias
```

where:

name is the official name of the service (i.e. telnet)

number is the software port number used by the service

protocol is the name of the TCP/IP sub-protocol used by the service (TCP, UDP, ICMP...)

alias is a list of alternative names, separated by spaces.

See also

HOSTTAB

SNMPD SNMPD

SNMP Agent.

Syntax

```
SNMPD [-u]
```

Description

As an SNMP agent for TCP/IP, SNMPD may be run after the TCP/IP kernel has been loaded.(either ETHTCP or TOKTCP). SNMPD enables network administrators to manage PCs from a distance.

Tun KERNEL's SNMPD agent implements MIB II.

-u May be used to remove the SNMPD agent from memory

TCPSTOP TCPSTOP

Unloads the TCP/IP kernel from memory.

Syntax

```
TCPSTOP
```

Description

This utility unloads the TCP/IP kernel (ETHTCP or TOKTCP) from the PC's memory. Before attempting to unload the kernel, all other TSR programs that are loaded after the TCP/IP must first be unloaded.

See also

TERMIN.COM

TELL TELL

Updates the resident TCP/IP kernel after IP parameters have been changed.

Syntax

TELL

Description

After having altered certain TCP/IP parameters (IP address, subnet bits, etc...), TELL may be used to update the TCP/IP kernel currently in memory.

See also

IPCONF.EXE, IFCONF.EXE

TERMIN TERMIN

Utility for unloading Crynwr (Clarkson) packet drivers.

Syntax

```
TERMIN 0xnn
```

Description

TERMIN unloads the Crynwr (Clarkson) packet drivers currently loaded in memory. The parameter **nn** is the software interrupt used by the packet driver (usually 62). This command may be used if the TCP/IP kernel has already been unloaded.

See also

TCPSTOP.EXE

TOKTCP TOKTCP

Resident TCP/IP kernel (Token Ringframes).

Syntax

TOKTCP [-i 0xnn] [-t nbtcp] [-p nbpacket] [-u nbudp] [-l]

Description

TOKTCP is a complete implementation of the TCP/IP layers as described previously for **ETHTCP**. However, TOKTCP is designed to process TOKEN-RING type frames as used by ODI drivers in a TOKEN-RING network.

TCPSTOP may be used to unload the kernel from memory.

The startup parameters for TOKTCP are the same as those for ETHTCP:

Notes:

1. When running the packet driver IBMTOKEN.COM, which generates ETHERNET II packets, use the Ethernet TCP/IP kernel (ETHTCP.EXE).

2. It is recommended to substantially increase the number of **packets** for Token Ring networks (using either resident TCP/IP kernel).

See also

ETHTCP.EXE, TCPSTOP.EXE

TUNKRNL TUNKRNL

Configuration menu for Tun KERNEL (MS-DOS only).

Syntax

```
TUNKRNL
```

Description

TUNKRNL is an interactive program that is used to configure, test, and launch the various functions offered by Tun KERNEL.

This program is described in detail in earlier chapters of this manual.

APPENDIX A - NETWARE COEXISTENCE

TCP/IP is a versatile protocol that is able to function in many different network environments, and over different topologies such as Ethernet and Token Ring. As the interconnection of different types of networks becomes more popular, it is increasingly necessary for PCs to be able to run several communications protocols at the same time.

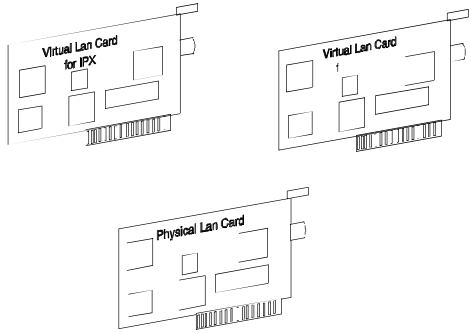
The Tun KERNEL Supervisor program automates installation of TCP/IP with Netware, by creating appropriate AUTOTCP.BAT and NET.CFG files. These files, as well as any differences in configuring Netware 3.11 and 3.12, are described in more detail in this chapter.

USING ODI WITH ETHERNET

Novell's ODI drivers

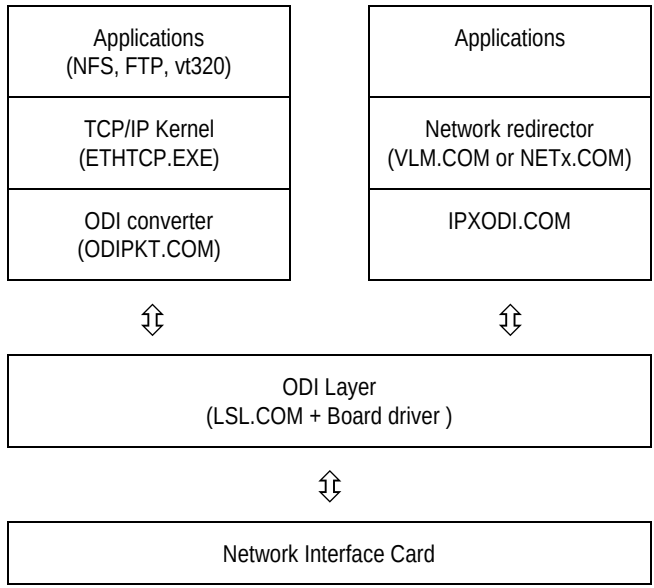
The most common way to configure network cards in PCs with Novell Netware is using ODI drivers (Open Datalink Interface). Since ODI has become a firmly-established standard, most LAN cards on the market are delivered with them.

ODI drivers are used to transform a single physical LAN card into two "virtual" LAN cards, each of which is configured to handle a single protocol. TCP/IP and IPX/SPX therefore each address their own virtual LAN cards for network communications.



ODI Protocol stack

In an **Ethernet** network, TCP/IP communicates using Ethernet II packets, and IPX/SPX uses Ethernet 802.3 packets.



In the above diagram:

1. The ODI layer is provided by Novell Netware's ODI drivers
2. Tun KERNEL includes a packet driver converter, which acts as an interface between the TCP/IP kernel and the ODI drivers

Configuration files

The configuration files necessary for loading ODI and several network protocols at the same time are:

NET.CFG Delivered by Netware, the card manufacturer, or created using the Tun KERNEL Supervisor

CONFIG.SYS Necessary for loading TCP/IP parameters into memory

AUTOTCP.BAT As configured in Tun KERNEL, this batch file may be used or may serve as an example of the programs that need to be loaded.

NET.CFG (Ethernet)

The virtual LAN cards are configured through a text file called NET.CFG. The necessary additions in this example for an NE2000 card are given in bold characters:

```
LINK SUPPORT
   buffers 6 1600
   Max board 4
LINK DRIVER NE2000
   int 5
   port 300
   Frame ETHERNET_802.3
   Frame ETHERNET_II
   Protocol IPX 0 ETHERNET_802.3
   Protocol ODIPKT 8137 ETHERNET_II
```

The NET.CFG file prepares the network card to use the different frame types required by Netware (802.3) and TCP/IP (Ethernet II).

CONFIG.SYS (Ethernet)

The Tun TCP Supervisor will setup the CONFIG.SYS as shown here, adding ESTCPIP.SYS and ESTCPIF.SYS to the existing file:

```
shell=c:\command.com /e:1024 /p
files=30
buffers=30
device=c:\tun\kerneld\estcpip.sys
device=c:\tun\kerneld\estcpif.sys
```

> **Note**: These two .SYS files are used by Tun KERNEL regardless of the LAN card driver type.

ODI Startup files (Ethernet)

Dual protocol configurations using ODI require different resident programs to be loaded than single protocol configurations.

For example, a standard **autotcp.bat** (for packet drivers only) might execute the following commands:

```
C:\TUN\KERNELD\CLARKSON\NE2000 0x62 5 300
C:\TUN\KERNELD\ETHTCP -p 4 -u 3 -t 6
```

In an ODI configuration this file would be replaced by

```
lsl
ne2000
ipxodi
odipkt 1
ethtcp -t 4 -u 3 -p 6
vlm (or netx)
```

This AUTOTCP.BAT file loads the following programs:

lsl(.com) Link Support Layer (also called the Link Services Layer); used for routing data packets between Network Interface Cards; also maintains LAN card, protocol, and packet buffer information. Furnished by Novell (in general, try to use the most recent version available).

ne2000(.com) The ODI driver for an NE2000 network card; also known as the MLID (Multiple Link Interface Driver); receives and copies packets from the Link Support Layer. Replace this with the driver for your LAN card. This driver is usually supplied by the board manufacturer.

As the ODI driver is loaded, a message indicating the presence of both protocols configured in the NET.CFG should be displayed.

For example:

```
Int 5, Port 300, Mem CA000, Node Address C01C2665
Max Frame 1514 bytes, Line Speed 10 Mbps
Board 1, Frame ETHERNET_802.3
Board 2, Frame ETHERNET_II
```

> **Note**: The messages given by the ODI driver are essential. Any difficulties encountered here indicate an incorrect net.cfg.

odipkt 1 The packet driver for the TCP/IP kernel (located in \KERNELD\DRIVERS). The parameter "1" links the odipkt driver to the virtual board #1 (which is really the second virtual card defined in the NET.CFG; the first is #0.)

ethtcp(.exe) The Esker TCP/IP kernel for Ethernet networks.

ipxodi(.com) Replaces ipx.com traditionally used for Novell-only networks. IPXODI does not need to be generated for a particular LAN card configuration, it addresses the virtual LAN card configured in the NET.CFG. Furnished by Novell (use versions more recent than 1.0).

vlm.com The Netware workstation shell, for establishing connection with Netware file servers. Replaces the older NETX.COM. Furnished by Novell.

For more information on ODI installations, please consult the Novell Netware User Manual.

USING ODI WITH TOKEN RING

The instructions and principles given in the previous section apply to Token Ring environments as well as Ethernet, with one or two minor differences.

Token Ring networks use Token-Ring 802.5 frame types (as opposed to 802.3 for Ethernet); TCP/IP over Token Ring uses Token-Ring_Snap frame types.

NET.CFG (Token Ring)

The NET.CFG file for a Token Ring installation needs to modified in order to take into account the name of the Token Ring ODI driver:

```
LINK SUPPORT
   buffers 6 1600
   Max board 4
LINK DRIVER LANSUP
   Frame TOKEN-RING
   Frame TOKEN-RING_SNAP
   Protocol IPX E0 TOKEN-RING
   Protocol ODIPKT E0 TOKEN-RING_SNAP
```

ODI STARTUP FILES (TOKEN RING)

The AUTOTCP.BAT file is similar to that given in the previous section; however, two lines need to be changed:

```
lsl
lansup (or ibmtoken.com)
odipkt 1
toktcp -p 4 -u 3 -t 6
ipxodi
vlm or netx
```

lansup(.com) ODI driver for an IBM Token Ring LAN card; used in conjunction with DXMA0MOD.SYS and DXMC0MOD.SYS (described below). Furnished by card manufacturer. In the NET.CFG, the line **LINK DRIVER LANSUP** refers to lansup.com.

toktcp(.exe) The Esker TCP/IP kernel used for Token Ring installations on ODI.

CONFIG.SYS (TOKEN RING)

As mentioned earlier, the Tun KERNEL Supervisor program creates and updates two system files: ESTCPIP.SYS, ESTCPIF.SYS. These files contain IP address information (including Hostname, Gateway Address, Name server, etc.).

In an ODI installation on Token Ring, the **config.sys file** also needs to include special Token Ring drivers. In addition, we recommend using the DOS **stacks** command as shown in figure A-7.

For example:

```
shell=c:\command.com /e:1024 /p
files=30
buffers=30
device=c:\xxx\DXMA0MOD.SYS
device=c:\xxx\DXMC0MOD.SYS
device=c:\tun\kerneld\estcpip.sys
device=c:\tun\kerneld\estcpif.sys
stacks=9,512
```

> Note: The drivers DXMA0MOD.SYS and DXMC0MOD.SYS are furnished by the LAN board manufacturer, and must be present when using LANSUP.COM to access a Token Ring card.
>
> You may also use the ODI driver IBMTOKEN.COM, which does **not** require the use of DXMA0MOD.SYS and DXMC0MOD.SYS to activate a Token Ringcard.

TUN KERNEL, NETWARE & WFWG 3.11

Running Tun KERNEL, Novell Netware and Windows for Workgroups at the same requires a configuration with three network protocols: TCP/IP, IPX/SPX and NDIS.

Before installing Tun KERNEL, please make sure that the WFWG and Novell functions are installed and running correctly. If this is the case, then adding a third protocol is very simple. Usually it is best to install Netware first, and then install Windows for Workgroups - declaring Netware (IPX) as an already-existing protocol in the PC.

After installing Tun KERNEL, use the Supervisor to select a card configuration using **NDIS Coexistence** in order not to create an AUTOTCP.BAT file, and not to load a packet driver. Usually, the network programs are configured by WFWG to load automatically in the AUTOEXEC.BAT file.

NET.CFG (Triple protocol)

If Netware and WFWG are properly running, add the two lines shown in bold type below to the existing NET.CFG file:

```
LINK SUPPORT
   buffers 6 1600
   Max board 4
LINK DRIVER SMC8000
   int 5
   port 300
   Frame ETHERNET_802.3
   Frame ETHERNET_II
   Protocol IPX 0 ETHERNET_802.3
   Protocol ODIPKT 8137 ETHERNET_II
   Frame Ethernet_802.2
   Frame Ethernet_SNAP
```

Startup files (Triple protocol)

The order of program execution should then be as follows:

```
c:\windows\net start
cd \windows
lsl
smc8000 (ODI driver for an SMC card)
c:\tun\kerneld\drivers\odipkt 1
c:\tun\kerneld\ethtcp -t 8 -u 6 - p 18
c:\windows\odihlp.exe
ipxodi
vlm (or netx)
```

> **Note**: Do not run a packet driver from the TUN\KERNELD\CLARKSON directory when using this type of configuration

KERNEL INSTALLATION ON A NETWARE SERVER

Tun TCP network licenses for use on a Novell Netware file server only need to be installed once. The procedure is the same as for a standard installation, except that you will specify a network drive as the destination instead of a local drive.

The configuration files for each PC in the network are customized using the copy installed on the file server. The files **ESTCPIP.SYS** and **ESTCPIF.SYS** must be customized for each PC, since they contain the IP address and other information specific to the PC.

Setting up each PC

1. Install Tun KERNEL on a network drive,

2. Run the TCP/IP configuration utility for the first PC,

3. Copy the system files ESTCPIP.SYS and ESTCPIF.SYS from the Tun KERNEL network directory to the PC's local hard drive, and include them in the config.sys file (for example, **device=c:\tun\kerneld\estcpif.sys** and **device=c:\tun\kerneld\estcpip.sys**). These files contain customized IP settings such as address, host name, etc.

4. Copy the appropriate packet driver to the PC's hard drive,

5. Copy the appropriate TCP/IP kernel to the PC's hard drive,

6. Copy the AUTOTCP.BAT to the PC's local hard drive (edit it as necessary to indicate the correct paths of the executable files),

7. Reboot and test TCP/IP network access from the PC (i.e. using PING.EXE),

8. Configure all following PCs the same way.

APPENDIX B - IP ADDRESSES

OVERVIEW

IP addresses are logical numbers used to identify each host in a TCP/IP network: every IP Address must therefore be unique.

IP addresses may be numbers up to 32 bits long, separated into four (4) 8-bit fields by periods (.). Many millions of possible combinations exist: 128.127.126.10, 60.0.0.1, 200.45.30.1 are examples of valid IP addresses.

In simple networks, selection of IP addresses can be made almost at random (as long as no two nodes are the same). However, there are rules and conventions for organizing larger and more complex networks, including those that will be connected to an "outside world" network such as the Internet or the ARPAnet Careful assignment of IP addresses can help divide large networks into smaller segments, thereby reducing total network traffic by keeping data local to each segment most of the time.

COMPOSITION OF AN IP ADDRESS

There are two parts to an IP address: the **Host portion** that identifies the machine itself, and the **Network portion** that must be the same for all nodes in the same network.

For example:

60.0.0.1 60.0.0.2 60.0.0.10	These addresses identify hosts **0.0.1**, **0.0.2**, and **0.0.10** on network **60**.
128.127.0.1 128.127.0.2 128.127.0.10	These addresses identify hosts **0.1**, **0.2**, and **0.10** on network **128.127**.
200.50.50.1 200.50.50.2 200.50.50.10	These addresses identify hosts **1**, **2**, and **10** on network **200.50.50**.

The **network portion** of the IP address can take 1, 2, or 3 bytes of the total 4 byte address, with the **host portion** taking the remainder. Potential network size (the total number of nodes) depends on how many bytes are available to identify the host.

The number in the first field on the left (specifically the first three bits of that number) indicates the **network class**, and therefore how many fields of the IP address are considered to make up the **network portion**. The table below summarizes network classes and the amount of hosts that may be assigned for each:

First field	Network Class	Example	Network Portion	Host Portion	Number of Hosts
1-126	A	40.1.2.3	40	1.2.3	254*254*254 (app. 16 million)
128-191	B	129.1.2.3	129.1	2.3	254*254 (app. 64,000)
192-223	C	210.1.2.3	210.1.2	3	254

SUBNET BITS AND SUBNET MASK

To further isolate network segments, it is possible to divide large networks into **subnetworks**. A subnetwork address uses part of the host portion of the IP address; the network portion stays the same. The number of **subnet bits** determines how much of the host portion will be used to identify a subnetwork, and therefore what possible addresses remain for the host portion.

The advantage to using subnetworks is that to the "outside world", the whole network is considered as a single physical network, but within the structure of the network, there are several different segments. This can help organization, and may be used to reduce overall network traffic.

INDEX

U

W