The Ethernet I Guide

PERIHELION SOFTWARE LTD

March 1991

Copyright

This manual is Edition 1.04, March 1991.

Printed in the UK.
PDF generated in AT (Vienna).

Acknowledgements

The Ethernet I Guide was written by members of the Helios group at Perihelion Software Limited. Helios software is available for multi-processor systems hosted by a wide range of computer types. Information on how to obtain copies of Helios software is available from Distributed Software Limited, The Maltings, Charlton Road, Shepton Mallet, Somerset BA4 5QE, UK. (Telephone: 0749 344345.)

Contents

Chapter 1

About this guide

The *Ethernet I Guide* describes the basic Helios network support package, **Ethernet I**. This package provides the Helios user with a gateway from Helios to other networks. With it the user can run well known programs to login to remote systems and to transfer files quickly between different machines. To enable the user to do this, the Ethernet I package conforms to the standard protocols for Ethernet, includes a TCP/IP server and features many of the commonly associated tools.

This guide is a systems administrators guide to setting up Ethernet I under Helios 1.2.1, or under any subsequent release of Helios. It covers how to install and customise the software for your own particular hardware configuration. It provides an overview of how Ethernet I deals with incoming and outgoing services on the network, and indicates the interaction of the various support programs. Because Ethernet I conforms to the BSD standard, the reference section simply consists of a quick summary of each of the commands provided. A full description of these commands can be found in any standard BSD documentation.

The *Ethernet I Guide*, however, does not attempt to be a detailed introduction to interprocess communication on Ethernet. This means that to get the best out of this guide, you should have some experience of interprocess communication, and a general understanding of the principles of TCP/IP. Readers without such a background should first consult the glossary and bibliography for further information.

There should be no need to read through this guide from cover to cover, although the order of topics is such that it takes you from setting up the software to testing and running it; all the reference material is at the back of the guide.

- **Checking things out** covers how to check out what is in your Ethernet I package.
- **Overview** gives an overview of the software and includes diagrams which summarise the information provided.
- **Installing Ethernet I** explains how to load the software described in the previous section, how to customise it for your network environment and then how test that the installed network interface is working correctly.

- **Examples** contains example code to help illustrate the use of Ethernet I.
- **User's reference** comprises a summary of each of the commands, arranged in alphabetical order.
- **Glossary** contains a useful alphabetical listing of common interprocess communication terms and concepts, together with a brief description of their use or meaning.
- **Bibliography** lists essential works of reference for background reading.

Chapter 2

Checking things out

First of all, before you do anything else, check that you are not running an early version of Helios. Ethernet I will not run on any version earlier than Helios 1.2.1. If you do not have the right version of Helios, contact your Helios distributor at once to arrange for an upgrade; without it you will not be able to to continue.

Check that you have sufficient memory available to run Ethernet I. To run the TCP/IP server, you should have a system processor with at least 1Mbyte of memory. The internet daemon also needs a lot of memory; you should allow a minimum of 1Mbyte for a single user and 2Mbytes or more for multi-user use. The internet daemon needs the extra memory because it creates daemons on the same processor and therefore it must have sufficient room to do so.

If you have a PC, you can use Ethernet I with a Western Digital EtherCard Plus WDLAN-EPR(F001) or a DLINK card. Instructions on how to set up and install the hardware are given in the section entitled **Installing the hardware**.

Now check your Ethernet I package; it should include the following:

- Two floppy discs (part number: M5110038)
- An *Ethernet I Guide* (part number: DM5042)

Notice that the package contains a choice of two PC-format floppy discs, 3.5 inch 720 Kbyte and 5.25 inch 1.2 Mbyte. Examine the contents of the disc that suits your system. Apart from the BSD documentation sources, it should contain the following files:

`/helios/lib/tcpip`	The TCP/IP server
`/helios/lib/ttyserv`	The PTY server
`/helios/lib/pc-ether.d`	The PC Ethernet driver
`/helios/lib/sq-ether.d`	The Parsytec Ethernet driver
`/helios/lib/in-ether.d`	The Inmos Ethernet driver
`/helios/lib/b407.b2h`	The Inmos B407 firmware
`/helios/lib/tpseudo.d`	The pseudo terminal (PTY) driver
`/helios/lib/tserial.d`	The serial line driver
`/helios/lib/inetd`	The Internet services daemon
`/helios/lib/ftpd`	The file transfer protocol daemon
`/helios/lib/telnetd`	TCP/IP TELNET protocol daemon
`/helios/lib/rlogind`	The remote login daemon
`/helios/lib/rshd`	The remote shell daemon
`/helios/lib/rexecd`	The remote execution daemon
`/helios/bin/ftp`	The file transfer program
`/helios/bin/telnet`	Remote system access
`/helios/bin/rcp`	Remote copy
`/helios/bin/rlogin`	Remote login
`/helios/bin/rsh`	Remote shell
`/helios/bin/ping`	Network maintenance tool
`/helios/bin/setterm`	Terminal type control
`/helios/etc/hosts`	Database files
`/helios/etc/services`	
`/helios/etc/networks`	
`/helios/etc/protocol`	
`/helios/etc/termcap`	
`/helios/etc/hosts.equiv`	
`/helios/etc/inetd.conf`	Network configuration files
`/helios/etc/socket.conf`	
`/helios/etc/devinfo.net`	
`/helios/include/netdb.h`	Include file
`/helios/include/net/*`	Include directories
`/helios/include/netinet/*`	
`/helios/include/arpa/*`	

`helios/local/tcpip/pc-ether/netdev.c`	Driver sources:
`helios/local/tcpip/pc-ether/devs.a`	example code to
`helios/local/tcpip/pc-ether/modend.a`	help you write your
`helios/local/tcpip/pc-ether/makedfile`	own hardware driver.
`helios/local/tcpip/pc-ether/example.c`	Example socket program.

Chapter 3

Overview

This section gives an overview of Ethernet I and shows how its constituent programs fit together. Because it is sometimes easier to understand things when they are displayed diagrammatically, a series of diagrams illustrating the following descriptions is included at the end of the section.

3.1 Helios /ether server

The Helios IO Server contains code for a server called `/ether`. This server allows Helios programs to access your PC's ethernet card in the same way as the `/serial` device allows Helios programs to access the PC's serial ports.

The `/ether` server provides low-level access to the actual ethernet hardware in the form: read a packet, write a packet, get an ethernet address. The transport medium at the `/ether` level is unreliable (that is, a packet that is written in *not* guaranteed to reach its destination).

The recommended way to use Ethernet I to communicate between machines is not at this low level. You should use the higher level commands, such as `ftp` (see below under **Commands**), which communicates with the TCP/IP server to call functions in the `ether.d` driver device (see below under **Drivers**. This device then communicates directly with the `/ether` server in the IO Server, which in turn interacts with the hardware of the ethernet card.

Warning: Do not attempt to access the `/ether` server while the TCP/IP software is running as the results are undefined.

3.2 Servers

Each of these programs provide a given service: `tcpip` runs the TCP/IP server, which provides the controlling interface for communication between hosts on the network; `ttyserv` controls the pseudo terminal driver. The `tcpip` service must be active in order for you to access the network; the `ttyserv` service is started on demand.

`ttyserv` is responsible for translating the Helios screen control escape sequences to whatever is needed for your hardware. For example, if `telnet` is used to login to a Helios machine from a VT100 then `ttyserv` will translate the Helios escape sequences into their VT100 equivalents. Furthermore, the ttyserver provides multiple windows with a hotkey-switching mechanism, like the PC I/O server.

3.3 Drivers

Drivers provide an interface between the Ethernet software and the hardware: they take ethernet packets from the physical network cable and interpret them for the host, and the other way about. On your distribution disc you will find a number of possible drivers: `pc-ether.d`, `sq-ether.d`, and `in-ether.d`. The TCP/IP server selects its driver according to an entry in the

```
/helios/etc/devinfo
```

file. This is described in detail in the installation section, later in this guide. If your hardware is not represented by any of the above, you will need to write your own driver. Should this be the case, turn to the **Examples** section for further advice.

The drivers `tpseudo.d` and `tserial.d` are for the pseudo terminal server `ttyserv`. The driver `tpseudo.d` implements a pseudo-tty interface, which is functionally equivalent to that provided by Unix. The ttyserver is used by `telnetd` and `rlogind` in this package and by `xterm` in the Helios *X Window* package. The driver `tserial.d` allows the ttyserver to use a serial line for directly attached terminals.

3.4 Daemons

3.4.1 Standard daemons

The application layer TCP/IP protocols include a support daemon on the remote receiving host for each Ethernet I command that can be invoked from the local source host. These support daemons do all the background work. The standard naming convention is *command name* followed by the letter *d*; for example, the daemon for `ftp` is called `ftpd`. The Ethernet I package supports the following standard TCP/IP daemons: `ftpd`, `rexecd`, `rlogind`, `rshd`, and `telnetd`. (See **Commands** for further information.)

3.4.2 inetd

The daemon known as `inetd` must be active for a remote host to access your local host. This daemon, which is actually a special sort of server, controls all the other daemons. It consults its supporting configuration file `inetd.conf` to establish which daemons are required. For example, it will only run `rshd` if a remote user requests a shell on the local machine; similarly, it will only run `rlogind` if a remote user attempts to login to the local host, and so on. The system can get clogged if too many daemons are run unnecessarily.

3.5 Commands

These are the standard tools that enable the user to login to a remote host, run a remote session, transfer a file from one host to another one, and so on.

3.5.1 ftp

The application layer File Transfer Protocol comprises the `ftp` program, which runs on the local host machine (source), and the `ftpd` daemon, which runs on the remote host machine (destination). (Notice that, for remote hosts to access the local host, a corresponding fptd should also be available to run on the local host.) The remote `ftpd` handles any requests from the local `ftp`.

3.5.2 telnet

The Telnet protocol allows terminals to communicate over a network supporting TCP/IP. This application layer protocol consists of the `telnet` program, which runs on the local machine (source), and `telnetd`, a daemon which runs on the remote machine (destination). (Again, for remote terminals to communicate with the local one, a corresponding telnetd should be available to run locally.) On running `telnet`, you can invoke a number of commands, a quick description of these commands can be found in the summary for `telnet` in the reference section, a fuller description can be found in the BSD reference documentation on disc.

3.5.3 rlogin

The rlogin protocol is an application layer protocol for handling remote login. It consists of a local `rlogin` program (source) and a remote `rlogind` daemon (destination).

3.5.4 rsh

The rsh protocol is an application layer protocol which is concerned with starting up a shell on a remote machine. It consists of a local `rsh` program (source) and a remote `rshd` daemon (destination).

3.5.5 rcp

`rcp` is an application layer protocol. It copies files over the network: from local machine to remote host, from remote host to local machine, and from remote host to remote host. Given the optional flag `-r`, it allows files and subdirectories to be copied recursively from a given root. Multiple copies in this way must go to a directory. This feature makes `rcp` particularly useful for backing up over the network, faithfully copying your file hierarchy from one machine to your filespace on another one. `rcp` will always assume the remote userid to be the same as the current local userid, unless you give a remote username with the remote hostname as part of the complete remote file identification.

3.5.6 setterm

`setterm`, if run from a window supported by the ttyserver, alters the termcap file entry the server is using for translating ANSI escape codes.

3.6 Database and configuration files

These files include the communication configuration files and the database files, which are administrative files used by each host and service to keep track of what machines, subnetworks, protocols or services are available. The ones supplied with Ethernet I are summarised briefly below. Notice that other database and configuration files used by the package will already have been supplied with the operating system, such as the password database `/helios/etc/passwd` and the configuration file `host.con`. The files outlined here should be treated in the following order:

1. If a file pre-exists your purchase of Ethernet I, use that file in preference to the one supplied.
2. If you do not have these files already, take the ones supplied and adapt them to suit your network.

You should not need to write any of these files from scratch. In some cases a database file may be copied directly from an existing Unix system.

3.6.1 hosts

`hosts` is a simple database file containing the names of all the hosts known to the network. There should be a separate `hosts` database file on every host machine on your network. This file is used by programs such as `ftp`, `telnet`, `rlogin` etc and their respective daemons to find a given host.

The file should contain several one-line entries, one for each known host, and must include a one-line entry for the machine on which it resides as well as an entry for `localhost`, which must not be removed. The `localhost` entry is used by programs to locate and use services on the same local host as the one from which they were initiated.

The host names and their corresponding network numbers should be copied by the network administrator from an existing `hosts` file if at all possible to tie in with the rest of the network. Host names should be unique to the network on which they reside. One or more aliases are allowed for each host. These aliases should follow immediately after the official host name to which they refer. Each entry starts with the host's network number. This number should comprise the standard four 8-bit octets of an IP address number (see Glossary). Each octet contains a decimal number in the range 0-255 and is separated from the next by a decimal point. The entire address must be unique in order to locate the correct machine over the network.

3.6.2 services

This file consists of a simple database containing the names of the services that the network knows about. Each entry in this file consists of a line containing the official name of the service, its port number and protocol name (divided by a forward slash), optionally followed by any aliases the service may have. Standard services in Ethernet I include ftp and telnet. The network administrator can alter this file to add additional services (for further details, see Configuring the files in the section on installing the Ethernet I software).

3.6.3 networks

This database file contains information on known networks or subnetworks in your TCP/IP network. It matches names or aliases to network numbers. Each one-line entry is of the form: *network name*, *network number*, optionally followed by one or more aliases. Any text after a `#` symbol up to the end-of-line is treated as a comment. You may make local changes to this file to add unofficial aliases or unknown subnetworks (see **Configuring the files**).

3.6.4 protocols

This database file contains information on the known TCP/IP protocols used in the DARPA Internet. For instance, this file is one of the files consulted by the `ping` program. The form for each one-line entry in this file is as follows: official protocol name, protocol number, alias(es). For example:

```
tcp 6 TCP  # transmission control protocol
```

You do not have to maintain this file.

3.6.5 termcap

This file contains a database of terminal capabilities. It is used by the pseudo terminal server. The format is compatible with that used by BSD.

3.6.6 hosts.equiv

Each host on the network will have its own `hosts.equiv` file. On PCs, the filename of this file is truncated to `hosts.equ`. This file is used by `rlogind`. It gives permission for given certain remote hosts and their users to access the local host machine on which it resides without a password, by indicating which hosts and users can be trusted to access the local machine over the network. You can rlogin even if you are not in `hosts.equiv`, but you will have to type in your password again.

The single-line file entries can contain the following: just the name of a network host, in which case any user of that host can be trusted to access the local machine; or the name of the host and that of a known user (separated by a space), in which case that host can only be trusted if that user is attempting to access the local machine. If a remote user or host is not listed as trusted in the `hosts.equiv` file of the local machine then they will not be allowed to access that machine. See also `.rhosts` in **User files**.

3.6.7 inetd.conf

This configuration support file is used by the inet daemon, `inetd`. The entries in this file specify the incoming communication requirements for the File Transport Protocol and the TELNET protocol: program name (for example, `ftp`), stream or datagram, transport layer protocol (for instance, `tcp`), `wait` or `nowait`, local host name, full pathname for the relevant daemon, and daemon name. The comments appear after the usual hash `#` symbol. On PCs, the filename is truncated to `inetd.con`.

3.6.8 socket.conf

This is the socket configuration file. It is used to translate the arguments to the `socket()` call into an appropriate server name. The entries in this file are of the form: domain, type, protocol and server. On PCs, the filename is truncated to `socket.con`.

3.6.9 devinfo

This is a device information database. It is compiled from a text source file by the `gdi` program. A `devinfo` and `devinfo.src` file are distributed with the Helios file server, a `devinfo.net` file is distributed with the Ethernet I product. See the section on configuring for further information on merging these files.

3.6.10 Include files

These are the files you need to include if you wish to port your own programs.

3.6.11 Library support

All the necessary library support required for the Ethernet I package is included in the BSD compatibility library `/helios/lib/bsd.lib`,distributed with with Helios version 1.2.1. The following routines are defined:

```
inet_netof  inet_lnaof    inet_makeaddr  inet_addr
inet_ntoa   inet_network  rcmd rexec     ruserpass
```

In addition, all the appropriate `ioctl` operations are implemented, with the exception of `SIOCGIFCONF`.

3.6.12 User files

There are two user files that should be kept in each network user's own home directory: `.rhosts` and `.netrc`.

.rhosts

This file is similar to the `hosts.equiv` file in the `/etc` directory of the local host machine, except that it represents hosts and users which are considered to be trusted by the local user, and not just by the local host machine. It is consulted after `hosts.equiv` to further weed out untrusted hosts and users before granting them access to the local machine. It is up to each individual to ensure that this file represents their own wishes. The entries in the `.rhosts` file of the local user of the local machine have precedence over the entries in the `hosts.equiv` file of the same machine.

.netrc

This file provides information for auto-login. It is used, in particular, by `ftp`, and may contain one or more of the following, which may be separated by spaces, newlines, or tabs:

`machine` *name*

This represents a machine with which a connection can be opened. All further entries in the file refer to this machine until end-of-file, or until another `machine` or a `default` is encountered.

`default`

This is like `machine` *name*, except that it matches any name.

`login` *name*

Where *name* is the userid to be used on login if the remote destination server requires a userid.

`password` *string*

Where *string* is the password to be used for login. This string is supplied automatically if the remote destination server requires a password for login.

`macdef` *name*

Enables macros to be defined. All subsequent lines are included in the macro until a blank line is given.

Chapter 4

Installing the hardware

This section makes reference to the installation of a Western Digital EtherCard Plus, WDLAN-EPR(F001), or a DLINK card in a PC-based system. If you have an Inmos B407 or B431 ethernet TRAM or a Parsytec ETN card, refer to the manufacturers' documentation for hardware installation. Before installing your network card, check that it does not conflict with any other boards in the PC. Please see your manufacturer's installation guide for details.

4.1 Configuring the host.con file

As the IO Server needs to know where to find the board, you must set the following values in the `host.con` file.

`ethernet`

Tells the IO Server to provide a `/ether` server.

`ethertype`

Set to WD8003E or DLINK, depending on which card is provided.

`ethermem`

Set to the base memory address in hexadecimal: 0xNNNNNNNN.

`etherbase`

Set to the base I/O address in hexadecimal: 0xNNN.

`etherrcr`

Sets the receive configuration register on the network card to accept broadcast packets. If it is set to any value other than 4 then the action of the TCP software will be undefined. The default is 0x04. *You are strongly advised not to alter this setting.*

4.1.1 EtherCard Plus

These are the important items to consider:

- Base I/O address
- Base memory address

So, if the base I/O address is 0x280 and you want the memory at 0xD0000000, add the following lines to the `host.con` file:

```
ethernet
ethertype = WD8003E
etherbase = 0x280
ethermem  = 0xD0000000
```

`etherrcr` should be left with its default setting.

Note: There is no jumper on this Western Digital card to set the base memory address as it is set by software.

4.1.2 DLINK

These are the important items to consider:

- Base I/O address
- Base memory address

So, if the base I/O address is 0x280 and you want the memory at 0xD0000000, add the following lines to the `host.con` file:

```
ethernet
ethertype = DLINK
ethermem  = 0xD0000000
```

You do not have to set the `etherbase` base address here as the software can calculate it by looking at the base memory of the board. Again, `etherrcr` should be left with the default setting.

Note: For a base address of 0xD0000000, you would need to set the jumpers on the board for the D0000h setting.

4.2 Booting Helios

Having included the correct entries in the `host.con` file, you can go on to boot Helios. If you have made any mistakes in setting up the board or in configuring `host.con`, you may get one of the following error messages from the IO Server:

```
I/O Server  :  unknown ethernet board ABCDE
            :  supported boards are WD8003E and DLINK
```

You might get this if you have got the `ethertype` entry incorrect in the `host.con` file. The next error message indicates that the IO Server could not find the ethernet board check that the base address and memory address correspond with the jumpers set on the board:

I/O Server	:	/ether device not found
I/O Server	:	base 280, mem D0000000, etherrcr 4, level 1, type 1

If you get any error messages, you should check the board and `host.con` file and then reboot Helios.

4.3 Finding out if /ether exists

To find out if `/ether` exists, type

```
ls /ether
```

to the shell. This should locate the device. If you fail to locate it, check the following:

- Have you have mistyped the word `ethernet` in the `host.con` entry?
- Are there any error messages from the I/O Server? Check by swapping to the Error Window with the ALT-F1 key combination.
- Have you connected your Helios system to a busy ethernet site? You should still be able to check that your system is working by typing `dump /ether` to read any broadcast packets from your ethernet and then display them to the screen. Notice that this will only work if you have other machines on the network sending broadcast ethernet packets.

Now follow the instructions in the next section of this manual to install the rest of the software.

Chapter 5

Installing Ethernet I

Installation can be broken down into the following steps:

1. Backing up the original distribution disc
2. Loading the package
3. Configuring the distribution files
4. Setting up subnetworks (this is optional and is not covered here)
5. Starting up
6. Checking the network

5.1 Backing up the distribution disc

Do not forget to back up the original distribution disc. Copy the contents of the disc to a safe storage device, preferably to another floppy disc, which should then be write disabled. This simple precaution should not be overlooked.

5.2 Loading the distribution files

To install Ethernet I, insert the distribution disc in your host machine and type:

```
loadpac
```

The `loadpac` program prompts you for the required information as it runs.

1. It will ask you to indicate which disc drive contains the package you wish to load. The default is `/a`. If you wish to select another drive, type `1`. `loadpac` will then display a new screen, giving you instructions on how to continue.
2. Once `loadpac` knows which drive to use, you can proceed with the installation. Select menu item 4 to install new software.

3. `loadpac` will now display a menu screen containing a range of packages that can be loaded. Select **Ethernet** by typing `5`.
4. All the standard files will now be copied from the distribution disc on the selected drive to the correct directories on your machine's hard disc. If you have inserted the wrong disc, or selected the wrong drive by mistake, `loadpac` will inform you that this has happened and will allow you to quit or amend your mistake. To quit `loadpac`, type `q`.
5. It will ask you if you wish to use the database and configuration files provided, or if you wish to retain previous versions. See *Configuring the files* below for further details.

Once you have loaded the package, you can proceed to configuring the files to reflect your own network environment.

5.3 Configuring the files

This section is about taking the default files provided on the distribution disc and amending them where necessary to reflect your network. For example, all the database files must contain entries that correspond to the hosts, services, and so on that are present on your network. Note that these files should normally be copied directly from existing machines. The network administrator for the whole site should be responsible for doing this.

5.3.1 services

You should only alter this database file if you have any additional services available on your network. Each service should have a one-line entry, consisting of the official name of the service, its port number and protocol name (divided by a forward slash), optionally followed by any alias the service may have. The first item may not have any leading blanks before it; otherwise, there may be any number of blank spaces in between each item. The port number and protocol name are treated together, hence they are simply divided by an intervening slash. The `#` symbol introduces a comment; routines calling this file will ignore all subsequent characters up to the end of the line.

5.3.2 hosts

The `hosts` database file contains the names of all the hosts known to the network. If you already have other Unix 5.4, BSD or SUNOS-based hosts, you may already have access to a standard /etc/hosts file. In which case, copy that file and adapt it according to your present requirements. Otherwise, take the `hosts` file provided and adapt that one instead.

For your Helios system to become part of your local network it must have both a unique name and a unique internet address. In most networks these are allocated by some central authority. This may be as informal as saying that the `hosts` database on a particular machine is the master copy. Alternatively, in very large systems, you may have to apply to a particular person who ensures that names and addresses are unique, the network administrator. Either way, the name and address of your machine must be installed both in the local `hosts` file and in the `hosts` files of all machines with which you wish to communicate. `hosts` is regularly consulted by `ftp`, `telnet`, etc and their respective daemons, to find a given host. There should therefore be a `hosts` database file on every host machine on the network to provide an index to the others. Each one-line entry in the file refers to a single host. The format is as follows:

inetaddr name [alias] [`#`*comment]*

The names and aliases must be unique to the network on which they reside. The *inetaddr* is a number allocated to the host which acts as its address on the network. It should follow the standard four 8-bit octets of an IP address number (see Glossary). Each octet contains a decimal number in the range 0-255. The first three octets represent the network and indicate its classification. The last octet indicates the host.

Warning: Do not remove the existing entry for `localhost` in the `hosts` database:

```
127.0.0.1   localhost
```

This entry must be present for programs to find services on the same local host as the one from which they were initiated. To add another host to `hosts`, follow these steps:

1. Unless the host is new to the whole network, insert into the file the existing address for that host, which you can find elsewhere (in another `hosts` database on another host machine, for instance). Otherwise, if the host is completely new to the network, take the entry with the highest `inetaddr`, copy it and increase the last octet (the host number) in the copied entry by 1. Remember that the maximum number is 255.

2. Alter the host *name* of the copied entry to match the name of the host to be added. The maximum length for a host name is 32 printable ASCII characters. Host names may not include the characters newline, hash (`#`) or slash (/).

3. Add or alter the *alias*; this may not be necessary because aliases are optional and are not always present. The same alias should not be used for two hosts on the same network as aliases and names should be unique, but two or more aliases can be used for the same host. Maximum length and character rules for aliases are the same as for *names.*

4. Insert a new comment after the `#` symbol. For example:

```
                # This is Bill's PC
```

5.3.3 networks

The `networks` database file contains information on known networks or subnetworks in your TCP/IP network. This file is used to match names or aliases to network numbers. You should only need to make local changes to this file to add unofficial aliases or unknown subnetworks. (Official network names are ones known externally to all internet networks; these names are maintained by the *Network Information Service*. If you reuse an official name, you may find a few interesting problems arise from the confusion.) Each one-line entry in this file is of the form:

> *name number [alias][comment]*

where *name* is the name by which the network is known, *number* is the IP network number, the optional *alias* refers to one or more extra unofficial names by which the network may be known, and the *comment*, which is also optional, is any supplementary text starting with a `#` symbol through to end-of-line. Entries may not start with any blank spaces; otherwise there may be any number of blank spaces or tab characters between each item. Lines starting with a `#` symbol will be treated as comments (that is, routines calling this file will ignore all subsequent characters up to the end of the line). To add new entry for each additional network or subnetwork, follow these steps:

1. Take the entry with the highest network number, copy it and increase the last digit of the last octet in the copy by 1. Network numbers should be specified with the usual octet and '.' notation used for IP network numbers.

2. Alter the network name and alias (if one exists) to match the name of the network or subnetwork you are adding. The names in this file may include any printable ASCII character except slash (/), newline, and hash (`#`), and may have a maximum length of 32 characters.

3. Add a comment to indicate to others what you have done.

5.3.4 devinfo

The TCP/IP server selects the ethernet device driver it is to use by consulting this file. The `devinfo` file is compiled by the `gdi` program that is distributed with Helios 1.2.1.

If you have the Helios File Server then you should already have `devinfo`, `devinfo.src` and `gdi`. You must use the new version of `gdi` distributed with Helios V1.2.1 and not that distributed with the File Server. This new version announces itself as Version 1.2 and is backwards compatible with the older version. To merge the existing devinfo files, simply concatenate the file `devinfo.net` onto the end of `devinfo.src` and proceed to edit it as described later.

If you do not have the Helios File Server, simply rename `devinfo.net` as `devinfo.src`. If you subsequently install the File Server, ensure that you rename this file back to `devinfo.net` beforehand and proceed as described above to merge the files.

If you are using the Inmos B407 or B431 ethernet TRAM you will need to edit the `devinfo.src` file to set up the link number and ethernet address. The link number is indicated by the `controller` field of the `netdevice in_ether` entry. This must be the link of the processor running the TCP/IP server to which the B407 or B431 is attached. This type of ethernet controller also needs to be told what its ethernet address is, valid values for this should be included with the hardware documentation.

The TCP/IP server takes a command line option to define the `netdevice` entry it will use. If no option is given then the name ‘ether’ is looked for. You may either change the name of the `netdevice` entry you want to use to ‘ether’ or supply it name to the server. Once the `devinfo.src` file has been edited to your satisfaction, save it and compile it with the following command line:

```
gdi /helios/etc/devinfo.src /helios/etc/devinfo
```

The file is now ready for use.

5.4 Starting up

Having configured all the files, the ethernet software is started up by invoking the TCP/IP server. The command line syntax for this is:

```
/helios/lib/tcpip name inetaddr [-s mask] [-e device]
```

The *name* and *inetaddr* must be the name and address allocated to your machine and should match the entries in all the `hosts` databases in the network. The *-s* option introduces a subnet mask in normal ‘dot’ notation. You will be informed by your network administrator whether you need to supply this, and what value should be given. The *-e* option introduces the name of the `netdevice` entry in the `devinfo` file which should be used. If no *-e* option is given the entry name ‘ether’ is used. If you wish other users to access the services supplied by your machine you will also need to run the internet daemon. As an example, a shell script to start the ethernet software would look something like this:

```
/helios/lib/tcpip Zaphod 42.0.0.42 -e in_ether &
/helios/lib/inetd &
```

Alternatively you can add the following lines to your `initrc` file:

```
run -e /helios/lib/tcpip tcpip Zaphod 42.0.0.42 -e in_ether
run -e /helios/lib/inetd inetd
```

If the processor to which your ethernet hardware is attached is not your root processor, you can use `remote` from the shell or `initrc`. Alternatively, you can run tcpip and inetd from the resource map (see the chapter on Networks in the latest *Helios Operating System* manual).

5.5 Checking the network

Once you have loaded Ethernet I and configured it for your own network, you may wish to check that the network interface is up and running correctly. To help you do this, Ethernet I includes the `ping` command. `ping` sends an echo request between two hosts on the network and watches for a response. Using it, you can isolate any inter-network problems that may occur. Because of the wide range of network hardware and the possible complexity of gateway interconnections, pinpointing a problem in the hardware or software can sometimes be a problem in itself.

`ping` acts by sending an ICMP/IP echo request datagram (a *ping*) to other network hosts to provoke an echo response from a host or gateway. Each of the ECHO_REQUEST datagrams has an IP and ICMP header, a `timeval` datastructure, and lastly a number of bytes of padding to fill out the rest of the packet. (The default length for a datagram is 64 bytes, but you may wish to change this with the *packetsize* option; see below.) The format for `ping` is as follows:

`ping` *[*`-r`*] [* `-v` *] host [packetsize] [count]*

The `-r` option causes the usual routing tables to be bypassed, so that the *ping* is sent directly to a host on an attached network. If the target host is not on a network that is attached directly to the originating host's network, an error occurs. The `-v` option causes any output to be *verbose*: it lists any ICMP packets other than ECHO_RESPONSE that it receives. *host* represents the target host. It can consist of an Internet address or a character-string that matches one of the known host names listed in the `hosts` file. *packetsize*, as mentioned earlier, allows you to specify a different byte size for the datagram packet (the default is 64). *count* is optional, it represents the number of times you wish the request to be sent.

`ping` sends one echo request datagram per second, and then returns one line of output for each corresponding response it receives. You can specify *count* number of requests. In which case, as `ping` only produces output if it gets a response from a request, you should get exactly *count* number of responses and *count* number of lines of output if all is going well. `ping` continues until it has

- Received all the responses it expects
- Timed out
- Terminated after receiving an interrupt signal

Each successful response provokes one line of output: for example,

```
64 bytes from 89.0.0.0 icmp_seq=0.time=12.ms
```

`ping` also works out the round-trip times for each *ping*, plus any packet loss statistics, and displays a brief summary. If a *ping* fails, one of the following error messages may be displayed:

```
Host unreachable
Network unreachable
Bad response from server
Timeout
Out of resources
The intial echo request packet could not be sent
```

To isolate a fault:

1. First run `ping` on your local host. This way you can check that the local network software is up and running correctly.

2. Next, in turn, *ping* each successive host or gateway away from your local host. Continue until you locate the problem.

If a remote host fails to respond to a network request, it means that there is a cable break at some point between your local host and the remote host, the host is down, or that host does not support the service you require. The purpose of `ping` is to help you work out which of these has caused the failure. If you can *ping* other remote hosts on the same network successfully, then it is likely that the original target host is down or not listening to the network. If you cannot *ping* any host on the same network successfully, then it is likely that the trouble is somewhere en route between your local host and the target remote host. You should then work along the route from your local host until you stop getting the expected response.

Datagrams are by definition unreliable: their delivery cannot be guaranteed. It is therefore quite possible for an echo request to be lost if, for example, the network is overloaded. This means that you should not assume that there is a problem on the network unless your *pings* consistently fail. Nevertheless, if most *pings* succeed up to a certain point on the network and then consistently fail beyond that point, you have cause to be suspicious.

Note: Overuse of `ping` places a great load on the network.

Chapter 6

Examples

6.1 Communication via sockets

While this document is not intended to be an introduction to the use of sockets for inter-process communication, a simple example program of using Internet sockets is included on the disc in

`/helios/local/tcpip/example/socket.c.`

Read the comments in this file for further information. For more information on the use of sockets, read the document available with any Unix system which supports it. The documentation supplied with SunOs is particularly recommended.

6.2 Writing a new driver

This can be found in the directory `/helios/local/tcpip/pc-ether` together with a makefile. The comments in the sources should be sufficient for you to be able to write a functionally equivalent driver. Note that in order to make drivers you need the latest release of the AMPP assembler macro pre-processor.

Chapter 7

User's reference

This section contains a quick summary of each of the commands. For a fuller description, see the 4BSD documentation sources on the distribution disc.

ftp

Purpose: Enables users to transfer files between network hosts.

Format: *ftp [-v] [-d] [-i] [-n] [-g] [host]*

Description:

`ftp` is the user interface to the ARPANET standard File Transfer Protocol. *host* is the name of the client host. If specified, `ftp` will open a connection to

the corresponding `ftpd` on that machine. If you omit *host*, `ftp` will start up the command interpreter to handle commands locally. You can find out what commands are available by giving the command `help` or by typing a question mark on a line by itself. Here is a quick list, similar to what would be displayed if you requested such general assistance.

```
!         $           ?           account   append
ascii     bell        binary      bye       case
cd        cdup        close       cr        delete
debug     dir         disconnect  form      get
glob      hash        help        image     lcd
ls        macdef      mdelete     mdir      mget
mkdir     mls         mode        modtime   mput
nlist     nmap        ntrans      open      prompt
proxy     put         pwd         quit      quote
recv      remotehelp  rename      reset     rmdir
rstatus   runique     send        sendport  size
status    struct      sunique     system    tenex
trace     type        user        verbose
```

To find out specific details about a particular command, specify the command's name as an argument to `help`. Most commands affect files or directories on the remote machine. For example, `cd` changes your directory to a directory on the remote machine to which you are connected. To change directory to a directory on your local machine, use `lcd`.

To terminate `ftp`, type `quit` or `bye`. To close a connection without terminating the session, use `close` or `disconnect`.

To abort a file transfer, press CTRL-C (terminal interrupt). If `ftp` is waiting for a remote reply, it will ignore the interrupt until it is ready. You may find in some circumstances, however, that you have to kill `ftp` instead.

To retrieve remote files, use `recv`, `get` or `mget`. To send a local file to a remote machine, use `send`, `put` or `mput`.

Notice that the `m` before the `put` or `get` commands in `mput` and `mget` means that `ftp` should expect multiple file transfers. Most commands expect to affect a single file; this may be confusing with familiar commands such as `rm`, which under `ftp` will only remove one remote file given as argument.

`ftp` processes filename arguments according to the following rules:

1. If you specify – in place of a filename, `ftp` takes input from *stdin* or sends output to *stdout*, depending on context.
2. If the first character of the filename is |, `ftp` interprets the remainder of the argument as a shell command to which the output is piped.
3. If 'globbing' is enabled, it expands local filenames according to `glob` shell command rules.
4. If the `recv`, `mget` and `get` commands are not given local filenames, the local filename is considered to be the same as the remote filename.
5. If the `send`, `mput` and `put` commands are not given remote filenames, the remote filename is considered to be the same as the local filename.

Parameters that affect file transfer are as follows:

type This may be ascii or image (binary). Ascii is the default type.

mode, *form*, and *struct* These take default values only: *stream* mode, *stream* structure and *file* format.

You may specify the following options at the command line, or to the command interpreter:

–v Turns verbose mode on. All responses from the remote server will be shown and data transfer statistics will be reported in full.

–n Stops auto-login on connection. If auto-login is enabled, `ftp` consults the user's `.netrc` for an entry for the remote machine. If it cannot find such an entry, it will prompt the user for a login id and, if necessary, for a password as well.

–i Stops interactive prompting.

–d Enables debugging.

–g Disables filename globbing.

ping

Purpose: A diagnostic tool that sends echo requests over the network.

Format: *ping [–r] [–v] host [packetsize] [count]*

Description:

`ping` is a network maintenance tool that can be used to isolate inter-network problems. A full description of its use can be found earlier in this guide, under the heading *Checking the network*.

`ping` acts by sending an ICMP/IP echo request datagram (a *ping*) to other network hosts to provoke an echo response from a host or gateway. Each ECHO_REQUEST datagram has an IP and ICMP header, a `timeval` datastructure, and lastly a number of bytes of padding to fill out the rest of the packet. (The default length for a datagram is 64 bytes, but it can be changed using the *packetsize* option.)

The arguments are as follows:

`-r` Bypasses the usual *routing* tables, sending the *ping* directly to a host on an attached network. An error occurs if the target host is not on a network that is attached directly to the originating host's network.

`-v` Provokes *verbose* output; that is, it lists any ICMP packets other than ECHO_RESPONSE that it receives.

host Represents the target host (IP address or host name string). The identification must match a known host in the `hosts` file.

packetsize Allows you to specify a different byte size for the datagram packet (the default is 64).

count Represents the number of times the request is to be sent (optional).

`ping` sends one echo request datagram per second, and then returns one line of output for each corresponding response it receives. You can specify *count* number of requests. In which case, as `ping` only produces output if it gets a response from a request, you should get exactly *count* number of responses and *count* number of lines of output if all is going well.

`ping` continues until it has

- Received all the responses it expects
- Timed out

- Terminated after receiving an interrupt signal

Each successful response provokes one line of output.

rcp

Purpose: Copies files between machines

Format: *rcp filename1 filename2*

Format: *rcp [–r] filename . . . dirname*

Description:

`rcp` stands for remote copy. It carries out remote file copying over the network in the same way that `cp` does within one machine. The arguments *filename*, *filename1*, *filename2* and *dirname* refer to remote or local filenames or directory names. Remote names are given as

`hostname:pathname`

or

`hostname.remoteusername:pathname.`

Local names are as usual, although they must not include a colon character so as not to be confused with a remote file description.

The optional flag `-r` indicates that the copying should be recursive, repeatedly copying the contents of any subdirectories below *filename.* If more than one file is being copied, its destination name must be that of a directory.

Within a remote filename description, *pathname* is usually assumed to be relative to your home login directory on the remote host *hostname.* Your local user id must match exactly with one on remote host *hostname* for the remote copy to take place, unless you specify *remoteusername* as being the name of the file owner on remote host *hostname.*

rlogin

Purpose: Attempts to login on a remote host.

Format: *rlogin rhost [-ec] [-l user] [-8] [-L]*

Description:

rlogin connects your terminal to remote host *rhost.*

Each host has a file `/helios/etc/hosts.equiv` which contains a list of the names of trusted remote hosts and users. If you are listed as being trusted on the `hosts.equiv` file on the remote machine, you will not need to give your password in order to login. The `.rhosts` file is a private version of `hosts.equiv` and can override the entries given there to effect automatic login. If your username or *user* is not listed as being trusted, `rlogin` will send the login prompt and request a password.

The options accepted by `rlogin` are as follows:

`-8` Allows an eight-bit input data path.

`-L` *litout* mode (provided for compatibility with 4BSD).

`-e`*c* Sets up escape character *c.* There should be no space between option flag `-e` and character *c.*

rsh

Purpose: Runs a command in a remote shell.

Format: *rsh host [-l user] [[-n] command]*

Description:

`rsh` opens a connection to *host*, and executes *command*. The local stdin will then go directly to the remote command, the remote commands's stdout will go to the local stdout, and the stderr of the remote command will go to the local stderr. Any interrupts generated locally will go automatically to the remote shell.

host This argument must match a name given in `helios/etc/hosts`.

user This is only necessary if the name is different from the local username. Notice that, unlike `rlogin`, `rsh` will not check passwords.

command If you run `rsh` without giving *command*, `rsh` will act like `rlogin`.

Any quoted shell metacharacters are interpreted on the remote machine; unquoted shell metacharacters are interpreted on the local machine. If a link is made to `rsh` using the name of a machine, then the program will detect this and use this as the destination host name. On files systems which cannot support links, this can be simulated by copying `rsh` to files of the appropriate name at the expense of extra disc usage. Additionally if no command is given then `rsh` executes `rlogin`. These features may be combined to make remote execution a little more natural. For example, suppose you have a remote machine `sparky` which you use frequently. Put a link to, or copy of, `rsh`, called `sparky`, into one of your command directories. Now, commands can be executed on `sparky` by: `% sparky <command>` and you can log into *sparky* with the simple command `% sparky`

setterm

Purpose: Sets the tty server terminal type.

Format: *setterm ttytype*

Description:

This program tells the ttyserver to change to a new terminal type. The single argument must be a terminal name which will match an entry in the termcap file. The program must be run in a window supported by the tty server.

Under Helios all programs use ANSI standard escape sequences to interact with terminals. The IO server implements these codes directly, but serial lines and remote sessions may have some other form of terminal attached to them. In order for the tty server to present the effect of any ANSI sequence it must be informed what kind of terminal it is dealing with and translate the codes accordingly. In the case of remote sessions the tty type will by set automatically, but in the case of serial line sessions, the type must be set by the user.

telnet

Purpose: Provides a user interface to the TELNET protocol.

Format: *telnet [host [port]]*

Description:

`telnet` is used to communicate with another host using the TELNET protocol. If it is invoked without any arguments, `telnet` enters command mode and displays the prompt `telnet>`. If invoked with arguments it opens a connection to the given host in exactly the same way as it would if the internal command `open` (given below) was invoked.

Once a connection has been opened `telnet` enters input mode, which can be line by line or character by character, depending on what the remote host requires.

The internal commands are as follows:

```
?       close   display  mode
open    quit    send     set
status  toggle  z
```

`?` *[command]* Displays a summary of available commands.

`close` Closes a `telnet` session.

`display` *[arg. . .]* Displays *set* or *toggle* values.

`mode line` | `character` Sets the mode type to line-by-line or characters.

`open` *host [port]* Opens a connection to a named host via either a named port or the default one.

`quit` Closes a session and exits.

`send` *arg. . .* Sends one or more special character sequences to the remote host. A full description of all the accepted arguments to `send` can be found in the summary for `telnet` in the reference section or in the BSD reference documentation or that of SUNOS or Unix 5.4. However, they include:

`escape` Sends the current Escape character

`synch` Sends the SYNCH sequence

`brk` Sends the Break sequence

`ip` Sends the Interrupt Process sequence

`ao` Sends the Abort Output sequence

- `ayt` Sends the Are You There sequence
- `ec` Sends the Erase Character sequence
- `el` Sends the Erase Line sequence
- `ga` Sends the Go Ahead sequence
- `nop` Sends the No OPeration sequence

`set` *arg value* Sets a telnet variable to the given value. A full description can be found in the full formal documentation of `telnet`. Here is a quick list of the possible variables: `echo`, `eof`, `erase`, `escape`, `flushoutput`, `interrupt`, `kill` and `quit`.

`status` Shows the current status of `telnet`.

`toggle` *arg...* Toggles between TRUE and FALSE for the various arguments to control telnet's response to events. Here is a quick list of the possible arguments:

- `localchars` Initially TRUE for line-by-line and FALSE for character at a time mode.
- `autoflush` Initially TRUE.
- `autosynch` Initially FALSE.
- `crmod` (Carriage return mode) Initially FALSE.
- `debug` Initially FALSE.
- `options` Initially FALSE.
- `netdata` Initially FALSE.

`z` Suspends `telnet` from the shell.

ttyserv

Purpose: To provide terminal functions on a stream

Format: *ttyserv [-i path] [-o path] [-l path] [-n name] [-d device] [-w] [-s] [-p] [-u]*

Description:

This is a server which provides terminal emulation, including line editing, ANSI escape sequence interpretation and multiple full screen windows. The exact form of the supporting device is determined by the driver loaded into the tty server. Two drivers are currently supplied: `tpseudo.d` which emulates Unix pseudo terminals, and `tserial.d` which uses a serial stream.

The options are as follows:

`-i` *path*

Use the stream *path* for keyboard input.

`-o` *path*

Use the stream *path* for screen output.

`-l` *path*

Use the stream *path* for input and output.

`-n` *name*

Use *name* as the server name, if this option is absent the name used will be `tty.N` where `N` is the lowest integer not already in use. If the pseudo-tty device is being used, it will use the same server name with the first character changed to a 'p' (e.g. `pty.0` for `tty.0`).

`-t` *type*

Initialise the ttyserver to use the *type* entry from the `termcap` file. If this is not present then it will attempt to use the termcap entry defined by the TERMCAP environment variable.

`-d` *device*

Use device *device.*

`-w`

Write the server pathname onto the output at startup. This is used by some pseudo-terminal clients.

`-s`

Equivalent to `-d tserial.d`.

`-p`

Equivalent to `-d tpseudo.d -w`.

`-u`

Print a usage message.

When the pseudo-tty driver is used, the tty server needs to be started from a program, this is what `rlogind` and `telnetd` do. A serial line server can be created from the command line by the following:

```
/helios/lib/ttyserv -l /rs232/default -s -t vt100 &
```

A shell, or any other command can now be run with:

```
shell </tty.0/Shell >&/tty.0/Shell &
```

To add an extra login terminal you can add the following lines to your initrc file:

```
run -e /helios/lib/ttyserv ttyserv -l /rs232/default -s -t vt100
waitfor /tty.0
console /tty.0 User1
run -e /helios/bin/newuser newuser
```

Any program which may be run on the IO server windows may be run in a tty server window. New full screen windows may be created in exactly the same way with `wsh` or `run`. For this to be useable the termcap file entry must contain definitions of the screen switching keys `%5` and `%8`.

Chapter 8

Glossary

This glossary is intended to act as a quick reminder to experienced readers and to help less experienced readers to follow and understand the description of inter process communication in this Helios Ethernet package.

Address family

Named groups, *domains*, using common address formats (`AF_HELIOS`, `AF_UNIX`, `AF_INET`, etc).

ARP

Address Resolution Protocol - resolves Internet addresses into Ethernet hardware addresses.

ARPA

Advanced Research Project Agency (part of US DoD) also known as DARPA, the Defense Advanced Research Project Agency.

ARPANET

Network of computers (1969-1988), since superseded by Internet, supported by DoD's ARPA agency and run internationally at universities and other research establishments.

Bandwidth

Data transfer rate of a device.

Broadcast

Sending messages through the network to all hosts.

BSD

Berkeley Software Distribution

Client

User or application requesting services from the network. The client, therefore, initiates a connection.

Collision detection

Detection of clashing message transmissions, where hosts attempt to transmit simultaneously over the same connection. If a host detects that such a collision has occurred, it must wait and then repeat the failed transmission.

Connection mode

Transfer mode whereby information is transmitted by way of an established connection in a reliable, sequenced manner. (*See also Sockets and Streams.*)

Connectionless mode

Transfer mode whereby information is divided into self-contained units and transmitted unreliably in unsequenced order. (*See also Datagram.*)

Daemon

Common Unix name for server.

DARPA

See ARPA.

Datagram

A unit of data transmitted between two tasks using the connectionless mode of communication. It is unreliable, unsequenced and it allows messages to be duplicated. It does, however, retain any internal record boundaries.

DoD

The United States of America's Department of Defense, the original source of funds for research into interprocessor communication.

Domain

A communications domain includes a common address structure and protocol for tasks that are communicating by way of sockets. HELIOS domain sockets have Helios pathnames. Sockets in the same domain can easily exchange data; sockets in different domains can only communicate if some translation process is implemented.

Ethernet

IEEE standard 802.3.

FTP

File Transfer Protocol

Host

A processor that requests services (*see Client*) or provides services. A transport user.

Internet

See also ARPANET, IP

IP

Internet Protocol

IP Address

Internet Protocol address. In order for packets to find their correct destination, the IP protocol on the source host attaches its address. This address is made up of three numbers: a network number, which is externally assigned by the official Network Information Center, and a subnetwork number and a host number, both of which are assigned locally by the network administrator. The total address is 32-bits wide, divided into four 8-bit fields, called *octets*. Each octet field is divided from the next by a decimal point. Each byte of the address can be represented by a decimal number, in the range 0-255.

IPC

Inter Process Communication

Layers

There are seven layers in the International Standards Division (ISO) Open Systems Interconnection (OSI) reference model, listed here from the lowest to the highest level: Layer 1, **Physical** (raw data transmission over some sort of data communications medium, such as an interface board or cable); Layer 2, **Data Link** (handles the exchange of data between the network layers, detecting and correcting errors in physical transmission); Layer 3, **Network** (manages the network, routing data exchanges for transport layer - IP works at this level); Layer 4, **Transport** (provides data transfer services for session layer by TCP); Layer 5, **Session** (provides services for presentation layer, helping with data exchange management); Layer 6, **Presentation** (manages information representation for applications layer); Layer 7, **Application** (serves communicating applications, handling their information exchange). Notice that each level provides services for the next level up. The level above therefore need not concern itself with the protocol used to provide its services.

OOB

Out Of Band.

OSI Reference Model

See Layers.

Ping

Command that is useful for testing and debugging networks: it sends a message to the specified host and then waits for a reply. It then reports back success or failure. A full description of `ping` can be found in the reference section of this manual.

Port

Transport user id (acts a bit like a phone number!) Certain port numbers are restricted:

Protocol

A formal set of rules and conventions that govern and regulate the exchange of information between communicating entities.

Protocol family

Named groups of protocols; for example, `PF_INET` for Internet protocol family.

Pseudo terminal

Raw

A raw socket provides access to the underlying communications protocols. They are of little interest to the general user.

RCP

Remote Copy Protocol. *See* `rcp` *in the command reference section of this manual.*

RDM

Reliably Delivered Message.

Sequenced packet stream

Server

Process supplying some form of service to the network. (Can also refer to opposite end of communication link from client.)

Socket

An endpoint of communication to which a name can be bound. Sockets can be: stream sockets, datagram sockets, raw sockets, etc. A pair of connected stream sockets look and act like a pipe. A datagram socket (type `SOCK_DGRAM`), unlike a stream socket (type `SOCK_STREAM`), is not sequenced or reliable. It may be duplicated and delivered in an order different from that which was originally sent. Unlike a stream socket, though, it does retain any record boundaries. Raw sockets (type `SOCK_RAW`) depend on the underlying protocol; they are not intended for the casual user. Sequenced packet sockets (type `SOCK_SEQPACKET`) are like stream sockets, except that they preserve record boundaries, whereas RDM sockets (type `SOCK_RDM`) are like datagram sockets, except that they undertake to deliver messages reliably, like stream sockets.

Stream

Sequenced data message, with no record boundaries, that flows reliably over an established inter-task connection. *See Socket.*

TCP

Transmission Control Protocol

TCP/IP

Transmission Control Protocol and Internet Protocol

Telnet

The standard TCP/IP remote login protocol. It allows you to use your terminal as if it were attached to a machine elsewhere on the network. *See* `telnet` *in the command reference section of this manual.*

Trailer

Method of sending information over Ethernet

Transport layer

The ISO layer that supports communication between users by carrying out any data transfer services: it receives data from the network layer, carries out necessary services and then passes the data on to the session layer. *See also Layer.*

Chapter 9

Bibliography

David D. Clark: *Internet Protocol Implementation Guide.* SRI International, August 1987.

J. E. McNamara: *Local Area Networks: An Introduction To The Technology.* Digital Press, 1985.

B. Nowicki: *NFS: Network File System Protocol Specification.*
Sun Microsystems, March 1989.

J. Postel, C. A. Sunshine, D. Cohen: *The ARPA Internet Protocol.* Computer Networks 5, no.4 (July 1981) pp 261-271.

RFC 791 Internet Protocol, DARPA Internet Program Protocol Specification.
Information Sciences Institute, University of Southern California,
September 1981.