parsytec

Parsytec Helios Ethernet

Manual

Parsytec Helios Ethernet Version 1.2

October 1991

Copyright , Parsytec GmbH

Helios
Ethernet

Parsytec reserves the right to make changes in specifications at any time and without notice. The information furnished by Parsytec in this publication is believed to be accurate, however no responsibility is assumed for its use, nor for any infringement of patents or rights of third parties resulting from its use. No licence is granted under any patents, trademarks or other rights of Parsytec.

The Parsytec Helios Ethernet is Copyright (c) 1990-1991 by Parsytec

All rights reserved. This document may not, in whole or in part, be copied, photocopied, reproduced or reduced to any electronic medium or machine readable form without prior consent, in writing from Parsytec Computer GmbH, D-5100 Aachen, Juelicher Strasse 338, Germany.

Transputer is a trademark of the INMOS Group of Companies
HELIOS is a trademark of Perihelion Software Ltd.

The use of registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Content

1. Introduction

The *Parsytec Helios Ethernet Guide* describes the basic Helios network support package, so called **Parsytec Helios Ethernet**. This package provides the Helios user with a gateway from Helios to other networks with the required functionalities for establishing communication between Helios tasks and processes running somewhere in the external network. It is also possible to run well known programs like *rlogin* to login to remote systems and to transfer files quickly between different machines by using utilities like *ftp*.

Parsytec Helios Ethernet 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 Parsytec Helios Ethernet under Parsytec-Helios Rel.910701 or any other 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 Parsytec Helios Ethernet deals with "outgoing services" on the network, and indicates the interaction of the various support programs.

The preferable usage of Parsytec Helios Ethernet on Parsytec systems is for establishing "outgoing services"; especially if machines with electronic configuration facilities like Parsytec's SuperCluster and MultiCluster-2 series are involved. Typical applications may be programs which are exchanging data with processes running somewhere in the external network or applications where the client is modelled under Helios whereas the server is residing outside the Transputer world. An example for this may be an XWindow application with a Helios X-client and the X-server running somewhere in the network (1). For using "incoming services" like *rlogin* directly into the Helios-world, an *Appendix* supplies the required information for customization.

(1) : In this case, the Helios X11.R4 package should be used. It is available as a separate product.

Because Parsytec Helios Ethernet mostly 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 Parsytec Helios Ethernet 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 also consult the glossary 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.

2. Accessing Ethernet from the Helios world

There are several ways to get access to Ethernet from the Helios world, always based on TCP/IP - protocols. Three of them are relevant for practical usage and are outlined below:

1. Parsytec's Transputer based Ethernet board - TPM-ETN:

The most powerful gateway is built by adding a Transputer based Ethernet board - Parsytec's TPM-ETN - to your network. In this case, the maximal performance is achieved and the T800 or T805 Transputer on the TPM-ETN adds significant more processing power to an application.

2. Extended Helios SUN I/O-server with an /internet-service

Helios systems which are hosted by a SUN-4 machine can make usage of a gateway via the SUN I/O-server. With the latest version of the SUN I/O-server (3.88), a */internet* service is added. The SUN I/O-server 3.88 becomes part of the standard Parsytec-Helios release and is also added to the Helios Ethernet package. In this case no additional Ethernet hardware has to be supplied.

3. Extended Helios PC I/O-server with a /ether-device

Helios systems which are hosted by a PC can make usage of a gateway via the PC I/O-server. With the latest version of the PC I/O-server (3.85), a */ether* device is included. The PC I/O-server version 3.85 is part of Parsytec-Helios Rel.910701 and subsequent versions will also supply this interface. In this case, the PC has to be equipped with an ethernet-card (e.g. Western Digital).

3. The different operating modes

With some typical applications in mind, executed on a Transputer based parallel computer, there are some basic functionalities the user very often would like to have:

Data transfer:

Transferring data between a Helios task and a network process on the base of socket communication. Establishing the connection is done by the Helios task by using one of the three gateways as described above.

Standard networking utilities:

Using network utilities for file transfer like *ftp* or *rcp*, remote login feature (*rlogin*) or similar; called from a Helios session and accessing external network instances.

Network services:

Access to remote servers like an X-Server under XWindow 11.R4. In this case, the client part (X-Client)is modelled as a Helios task and executed within the Transputer world, whereas the server (X-Server) is somewhere residing in the network.

Each of these three application examples can be realized by using the Parsytec Helios Ethernet package as supplied with this release. XWindow 11.R4 is available under Helios as a separate product.

4. The Parsytec Helios Ethernet release

First of all, before you do anything else, check that you are not running an early version of Helios like Parsytec-Helios Rel.900301. Parsytec-Helios Ethernet will not run on any version earlier than Parsytec Helios Rel. 910701.

If you do not have a suitable version of Helios, contact your Helios distributor at once to arrange for an upgrade; without it you will not be able to continue.

The Parsytec Helios Ethernet release contains the following files:

/helios/lib/tcpip	The TCP/IP server
/helios/lib/pc-ether.d	The PC Ethernet driver
/helios/lib/sq-ether.d	The Parsytec Ethernet driver
/helios/lib/tpseudo.d	The pseudo terminal (PTY) driver
/helios/bin/ftp	The file transfer program
/helios/bin/ping	Network maintainance tool
/helios/bin/rcp	Remote copy
/helios/bin/rlogin	Remote login
/helios/bin/rsh	Remote shell
/helios/bin/telnet	Remote system access
/helios/etc/hosts	Database files
/helios/etc/services	
/helios/etc/networks	
/helios/etc/protocol	
/helios/etc/hosts.equiv	
/helios/etc/inetd.conf	Network configuration files
/helios/etc/socket.conf	
/helios/etc/devinfo.net	
/helios/include/net/*	Include directories
/helios/include/netinet/*	
/helios/include/arpa/*	
	Device driver example source, if someone wants to write his own device driver.

```
/helios/users/guest/examples/pc-ether/netdev.c
/helios/users/guest/examples/pc-ether/devs.a
/helios/users/guest/examples/pc-ether/modend.a
/helios/users/guest/examples/pc-ether/makefile
```

Within a separate tar-archive, the following daemons can be found, which are required, if someone wants to make explicitely usage of the "incoming services".

/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

In the following, the basic installation steps are described. Specific details about customization the software for proper usage are given in chapter 6 ("Customizing the system").

4.1 Installation on a SUN

For a proper installation of the Parsytec Helios Ethernet package on a SUN, you have to get *superuser* permissions. The first thing you have to do afterwards, is reading the tape and copying its content into the Helios root directory. You can perform this for example in the following way:

```
cd /helios <enter>
tar xvpf /dev/rst0 <enter>
```

After reading the distribution tape, you have to change to the Ethernet subdirectory by performing

```
cd ethernet <enter>
```

and afterwards performing the installation by calling

```
make install <enter>
```

Note: If you want to make usage of the */internet* service embedded into the Helios SUN-server, you should keep in mind that this facility is only supported on SUN-4 machines. In this case you have to check whether you are using an I/O-server version below 3.88. This makes it essential to replace that SUN I/O-server with the version 3.88 as supplied with this release!

To install the server version 3.88 type the following command:

```
make server <enter>
```

4.2 Installation on a PC

The PC installation is simplified by using the *loadpac* utility, which allows an interactive installation process.

Running under Helios, you can load additional software packages like this release of Parsytec Helios Ethernet by using the *loadpac* utility. *Loadpac* is a menu driven installation and maintenance program and you will be guided by the instructions that appear on the screen.

Just start installation by typing

```
% loadpac <enter>
```

After using *loadpac* you should first call *rehash* before making usage of the software to rebuild the internal hash-tables of the shell to assure that the new objects can be found using the standard search pathes.

5. Overview on Parsytec Helios Ethernet

In the following, informations are given on various topics directly related to the Parsytec Helios Ethernet package like the different host server gateways, the set of configuration files and so on.

Note

With regard to general requirements like reliability of communication and portability of generated code, it is highly recommended to use high level commands like *ftp* or POSIX based *socket* calls based on the TCP/IP layer, instead of accessing the correspondig hardware on device driver level directly.

5.1 The Transputer based Ethernet board - TPM-ETN

If you are using Parsytec's TPM-ETN board, setting up the system is quite simple: The first thing you have to do is to write a resource map which integrates the TPM-ETN node, compile it and boot the network. During a second step, the Parsytec Helios Ethernet software, mainly the tcpip-server have to be run on this node. Chapter 6, "Customizing the system", gives all the required informations to do this properly.

Please take a closer look at your TPM-ETN hardware documentation for details about the hardware installation procedure. For further details about software customization, see below.

A special note for TPM-ETN users:

With the TPM-ETN embedded into a SuperCluster or MultiCluster-2 system, a Berg - to - D-Sub connector is supplied. The correct pin-layout is given below:

Berg (10 pins)			D-Sub (15 pins)	
	PIN		PIN	
COLL_P	1		2	COLL_P
COLL_M	2		9	COLL_M
RX_P	3		5	RX_P
RX_M	4		12	RX_M
TX_M	7		10	TX_M
TX_P	8		3	TX_P
		GND	6	GND
		+12 V	13	POWER

There may be systems with a pin 4 to pin 4 connection instead of a pin 4 to pin 12 connection as shown above. If you recognize a different pin interconnection layout, please call our product support departement to arrange an update.

5.2 The Helios SUN /internet service

The latest version of the Helios SUN I/O-server (version 3.88 and subsequent versions) includes extensions to allow users of a Helios system hosted by a SUN-4 to access the SUN's ethernet via a */internet* service. In this case, no additional Ethernet hardware is required. In essence the */internet*-service replaces the *tcpip*-server as supplied with the Helios Ethernet package. The */internet*-service provides full support for the Helios socket calls by translating them into SUN socket system calls.

To enable this service, add the entry

```
internet
```

to the host.con file. The */internet* service will then start automatically when the I/O-server boots up.

Note

As far as the rest of the external network is concerned, the Helios world is indistinguishable from the host machine. As a consequence: Assuming that the networking daemons (e.g. *inetd*) are running on the host machine, they cannot run concurrently under Helios, as the Helios world responds to the same internet address (and port numbers) as the host machine. In other words: Incoming internet traffic will 'intercepted' by the daemons of the host machine.

In general certain network commands allow the selection of the port number (check the host system documentation for more details). One such command is *telnet*, when used thus

```
telnet   host   port
```

e.g.

```
telnet   luna   7800
```

The example above requests *telnet* to connect to host 'luna' using the non-standard port 7800. To reflect this you have to install a new service under this port number (See section about configuration files.)

Another problem with using the host machine's operating system to provide a */internet* service is that certain system calls are restricted under UNIX to the super-user. In particular, creating a raw socket and binding to a reserved port are super-user only calls! Since most of the Parsytec Helios Ethernet commands do, indeed, bind to reserved ports (e.g. *rlogin*) or create raw sockets (e.g. *ping*), the I/O-server now has to run as super-user if these calls are to succeed. (Without super-user status for the server, *telnet* and *ftp* can still be used.)

However, the code has been written so that the I/O-server only assumes super-user status for the duration of these calls, reverting to the uid of the program initiator at all other times. If security is a major issue - and this is no point of discussion for larger multi-user systems - the I/O-server should not be setuid root. Otherwise the following UNIX commands will enable this feature:

```
chown   root   server
chmod   04755   server
```

After these slightly modifications, the SUN I/O-server gateway to ethernet is ready to use. To find out whether the /internet-service is active, just type

```
ls -l /internet
```

from a Helios shell. This should locate the /internet-service within your SUN I/O-server and give at least the message '*0 entries*'. If this fails, please check your installation again and take especially a look on the following:

- Have you mistyped the word **internet** in the host.con entry ?
- Are there any error messages given by the I/O Server ?

Check this by turning to the Serverwindow.

5.3 The Helios PC /ether server

Similar to the Helios SUN I/O-server, the Helios I/O-server for the PC (version 3.85 and upper) contains also code for building a gateway to ethernet. In this case, it is a server called */ether* embedded into the PC I/O-server. This server allows Helios programs to access the PC's ethernet card in the same way as the */serial* device allows Helios programs to access the PC's serial ports. Warning: Do not attempt to access the /ether server directly while the TCP/IP software is running as the results are undefined.

In the following, some informations are given specific for the PC based system with an additional ethernet card used. Up from now, we refer to a PC, equipped with the *Western Digital EtherCard Plus, WDLAN-EPR(F001)*:

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.

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.

ethermem

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

etherbase

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

etherrcrm

> 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.

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.)

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.

For further details about software customization, see below.

5.4 The TCP/IP Server

The TCP/IP server is initiated by runnning */helios/lib/tcpip*. The TCP/IP server provides the controlling interface for communication between hosts on the network. All incomming socket calls for the internet domain will be handled by this server.

5.5 Drivers:

The drivers provide an interface between the Ethernet TCP/IP Server and the underlying hardware - or in other words: they take ethernet packets from the physical network cable and interpret them for the host, and the other way round.

On the distribution media you will find some possible drivers:

pc-ether.d	:	The PC Ethernet driver
sq-ether.d	:	The Parsytec Ethernet driver for the TPM-ETN

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.

5.6 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 the Parsytec Helios Ethernet package 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 host configuration file *host.con*. The files outlined here should be treated in the following order:

1. If a configuration file pre-exists before your purchase of the Parsytec Helios Ethernet Package, use that file in preference to the one supplied and make modifications, if necessary.

2. If you do not have these files already, take the ones supplied with this release 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 to be used under Helios.

hosts

The *hosts* file is a simple database 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, ping* etc and their respective daemons to find a given host. The file should contain several entries, one line for each known host, and must include a one-line entry for the machine on which it resides as well as an entry for the *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.

services

The file *services* 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 slash), optionally followed by any aliases the service may have. Standard services in Parsytec Helios Ethernet include *ftp* and *telnet*. The network administrator can alter this file to add additional services (for further details, see 'Customizing the software' in the section on installing the Parsytec Helios Ethernet software.)

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 following 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 again 'Customizing the software').

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
```

(Usually you do not have to maintain this file.)

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. (Note: Within a PC environment, the filename is truncated to *socket.con* .)

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 for example distributed with the Parsytec Filing System (PFS), a *devinfo.net* file is distributed with the this release of Parsytec Helios Ethernet. See the section on customizing for further information on merging these files.

Include files

These are the files you need to include if you wish to port your own programs. They are found in the */helios/include* directory and three subsequent directories (*net*, *netinet* and *arpa*).

Library support

All the necessary library support required for the Parsytec Helios Ethernet package is included in the BSD compatibility library */helios/lib/bsd.lib*, which is distributed with Parsytec-Helios Rel.910701. 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.

6. Customizing the system

Customizing can be broken down into the following steps:

- o Performing the basic installation
- o Configuring the distribution files
- o Starting up
- o Checking the network

The initial steps for basic installation were described in chapter four. In the following, configuration and the subsequent steps are outlined.

6.1 Configuring the files

This section is about taking the default files provided on the distribution media 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.

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 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.

hosts

The *hosts* database file contains the names of all the hosts known to the network. If you already have other Unix V.4, BSD- or SUN-OS-based hosts, you may already have access to a standard */etc/hosts* file. In this case, copy that file and adapt it according to your present requirements. Otherwise, take the *hosts* file provided and adapt this one instead.

To make 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, your 'network administrator'. Nevertheless which way you take, 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*, *ping* 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
```

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.

devinfo

The TCP/IP server selects the ethernet device driver it is to use by consulting this file. The *devinfo.src* file is compiled by the *gdi* program that is distributed with Parsytec-Helios Rel.910701.

If you are running the Parsytec Filing System (PFS), you should already have *devinfo*, *devinfo.src* and *gdi*. Please make sure that you are using the new version of *gdi* as distributed with Parsytec-Helios Rel.910701! This latest version identifies itself as *Version 1.2* and is backwards compatible with the older ones. 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 on.

If you are not running the Parsytec Filing System (PFS), simply rename *devinfo.net* as *devinfo.src*. If you subsequently want to install also the Parsytec Filing System, ensure that you rename this file back to *devinfo.net* firstly before you make further installation steps and proceed as described above to merge these files.

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 usage.

6.2 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.

As an example, a shell script to start the TCP/IP Server would look something like this:

```
/helios/lib/tcpip  Zaphod  42.0.0.42  -e in-ether  &
```

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
```

If the processor to which your ethernet hardware is attached is not your root processor - e.g. Parsytec's Transputer based Ethernet board TPM-ETN - you can use the remote command from the shell. In the following example we assume that the Ethernet-node is called "ETN" in the resource map and that the network was booted up successfully.

```
remote -d ETN /helios/lib/tcpip Zaphod 42.0.0.42 -e sq-ether
```

or within the */helios/etc/initrc* - file

```
run -e /helios/bin/remote remote -d ETN /helios/lib/tcpip \
Zaphod 42.0.0.42 -e sq-ether
```

6.3 Checking the network

Once you have loaded the Parsytec Helios Ethernet software and successfully configured it for your own network, you may wish to check that the network interface is up and running correctly. To assist you in doing this, Parsytec Helios Ethernet 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 (so called 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 or 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.

7. Command Reference

This section contains a quick summary of each of the commands. For a fuller description, refer to the BSD documentation sources.

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

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

Each successful response provokes one line of output.

rcp

Purpose: Copies files between machines

Format: *rcp filename1 filename2*

or

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).

-ec

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 *luna* 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: **% luna <command>** and you can log into *luna* with the simple command **% luna**.

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
```

?

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 BSD reference documentation or that of SUNOS or Unix V.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, escape, interrupt, quit, flushoutput, erase, kill* and *eof*.

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.

Appendix

This appendix deals with the "incoming network services". In contrast to applications running under Helios and accessing network instances outside the Transputer world - for example exchanging data between a Helios task and a process running somewhere in the network - "incoming services" allow for example a remote login into an existing Helios processor pool from an external network site.

Note

This operating mode makes especially sense, if small or medium sized systems have to be managed, which does not offer electronic configuration facilities like Parsytec's SuperCluster and MultiCluster-2 machines. Because of the fact that in this case several users share transparently a pool of processors, special care has to be taken to customise the system properly.

The usage of "incoming services" may become relevant for the following two types of operating environments:

- A multi-processor system with adjustable topology (e.g. Parsytec's MultiCluster-1) to be shared among several users.

- A separate partition within a SuperCluster or MultiCluster-2 machine to be shared among several users.

In addition to this, it may also be desirable to access an existing Helios network via ethernet by one user, maybe running a remote shell (*rsh*) or performing a *ftp*.

Sharing a pool of processors among a heterogenous group of users - logged in via ethernet - is an efficient way of using Transputer resources with several users. On the other side there are some implications to be mentioned, which have to be kept in mind when establishing "incoming services":

o Because of the lack of physical separation of different users, it is possible that one user may have influence on the behaviour of others. This may result in fatal situations, if this user for example crashes some of his processors.

o Running applications of different users on one physical processor pool may interfere in the way that communication is routed transparently through the network. Depending on the allocation of processors to users being active on the network, situations like the following may appear: Communication between two tasks of user *A* are routed via a processor used by user *B* and similar things.

o Having several users working simultaneously on a processor pool defines additional requirements for the reliability of the involved filing system. Please note that for example a MS-DOS file system does not fulfill these requirements because it is not designed to be used in a multi-user, multi-tasking environment. In this case, the use of a Transputer based mass storage controller board like Parsytec's MSC, running the Parsytec File System, is highly recommended.

o Another thing to be kept in mind is the question of security. Running Parsytec-Helios Rel.910701 allows to protect processors allocated by one user to be accessed explicitely by another user.

o Depending on the mass storage devices used, security cannot be guaranteed under all conditions: The MS-DOS filing system for example does not have any mechanisms for user protection. Embedded into a SUN-environment, all users looged into the system via ethernet are having the same access rights to the UNIX-file system.

As a consequence of these considerations, there are in general two variations of multi-user systems:

1.) The **shared processor pool** - accessed via Ethernet or dedicated hosts - gives a good hardware utilization and easy access to a limited set of processor resources to a number of users. The primarily usage of such a system is for evaluation, development and testing of parallel applications. Installation details for such an environment are given below.

2.) Systems with **physically separated partitions** like Parsytec's SuperCluster and MultiCluster-2 machines give the highest possible reliability and efficiency and are therefore primarily to be used for "production runs" (2). In this case, "incoming services" are directly built on host functionalities. Nevertheless it is possible to take a separate partition from such a system and use it as a processor pool, allowing development work with several users on it. The following chapters will also cover this aspect.

Running the daemons

All network daemons are collected in a separate tar-archive and if someone wants to make explicitely usage of the "incoming services", the installation procedure is described briefly in the following:

Contents of the tar-archive:

/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

(2) : Imagine for example a system of 256 processors, residing in a computing center, with jobs running for several days or weeks.

Network daemon installation on a SUN:

After reading the distribution tape and installing the Parsytec Helios Ethernet package, you have to run

```
make   daemons   <enter>
```

inside the Ethernet subdirectory. Note: You need still to have super-user permission to do this. This will copy all the daemons into the right place of your Helios system.

Network daemon installation on a PC

The PC installation is simplified by using the *loadpac* utility, which allows an interactive installation process. During the installation process of this release of Parsytec Helios Ethernet you will be asked whether you want to install the daemons or not.

Starting up "Incoming services"

Besides the *tcpip* server which is assumed to run already at this point the daemon known as *inetd* must be active for a remote host to access a network service on 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 work out which daemons are required and to establish them.

For example, it will only run *rshd* if a remote user requests a shell on the local machine; similarly, it will only run *ftp* if a remote user attempts to establish a *ftp* connection to the local host, and so on. Please note that the system can get clogged if too many daemons are run unnecessarily.

To start up the internet daemon the type the following command line:

```
/helios/lib/inetd  &
```

or to start it on a dedicated processor in the Transputer network:

```
remote  -d  <processor>  /helios/lib/inetd
```

Alternatively you can start the *inetd* from your *initrc* file. In this case you have to add the following line:

```
run  -e  /helios/lib/inetd  inetd
```

or to start it on a dedicated processor in the Transputer network:

```
run  -e  /helios/bin/remote   remote  -d  <processor> \
/helios/lib/inetd
```

After performing this, the user should be able to contact all services which are configured in */helios/etc/inetd.conf* from a remote host.

Establishing the multi-user system

A multi-user system based on accesses via ethernet is an extension to the multi user / multi processor environment as described in **chapter 4.3.4** of the Helios system software documentation ("Helios Operating System Manual"). In contrast to the example given with the Helios manual, ethernet services are established and used instead of enabling links from dedicated hosts.

Some comments on the topology of the processor pool:

- In general it is desirable to have a high connectivity of the involved processors to make best usage of the communication bandwidth of the Transputers.

- With a system of adjustable topology like the MultiCluster-1, you should establish the target topology manually.

- Working on a machine with electronic configuration facilities implicits a slightly different proceeding: In this case, the processor pool forms a separate partition within the whole system, whereas the other partitions are still for exclusive usage by different users. Dedicated processors like the TPM-ETN and (optionally) the MSC should be configured and embedded as *special processors* into the network. (Please take a closer look at the *Network Configuration Manager* documentation for further details.)

- If a TPM-ETN or a MSC are involved, it is a good practice to put them somewhere on the "edges" of the processor pool to minimize interferences with running applications.

Some advices concerning configuration files:

The *nsrc*-file should be set-up in the way that only the *preload_netagent* and the *waitfor_network* options are enabled. It may look like this:

```
# single_user
# processor_protection
# no_taskforce_manager
# share_root_processor

preload_netagent
waitfor_network
```

The *initrc*-file includes in this case also the installation of the *tcpip*-server and the creation of the *inetd*-daemon as described in chapter 6.2 and this appendix.

If the Parsytec File System - running on a MSC - is involved, it is a good practice to keep the standard Helios directory-tree there. There is still the requirement for a host booting the network and initializing all relevant software instances, but afterwards it is possible to define an *alias* to search standard Helios objects up from now not on the root's filing system but on the MSC.

Assuming that the network is booted successfully and the file server is up and running, the following command would define the alias. (The mass storage node is called */MSC* and the file system's root there */fs*.)

```
/helios/lib/alias   helios   /MSC/fs &
```

Caution: Please be aware that both Helios directory trees, residing on the host and on the mass storage node, should be consistent to avoid problems!

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 the Parsytec 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.

Raw

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

RCP

Remote Copy Protocol.

RDM

Reliably Delivered Message.

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.

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.