ChorusOS 4.0 Network Administration Guide

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900
U.S.A.

Part Number 806-3715
November 1999

Contents

Figures

Code Examples

Preface

The *ChorusOS 4.0 Network Administration Guide* describes what you need to know to perform network administration tasks on ChorusOS™ systems.

The *ChorusOS 4.0 Network Administration Guide* does not describe network programming with ChorusOS 4.0.

Who Should Use This Guide

This document is written for both users and system administrators of the ChorusOS 4.0 product.

Before You Read This Guide

You must be familiar with the concepts explained in the *ChorusOS 4.0 Introduction*.

You must also have an operational ChorusOS 4.0 system including at least a target and a host. See the *ChorusOS 4.0 Installation Guide for Solaris Hosts* or the *ChorusOS 4.0 Installation Guide for Windows NT Hosts* for details about installing the ChorusOS 4.0 product.

How This Guide Is Organized

Chapter 1 explains where network administration fits into overall ChorusOS system administration.

Chapter 2 describes the network components provided in the ChorusOS 4.0 product.

Chapter 3 shows you how to set up Ethernet on a ChorusOS system.

Chapter 4 shows you how to set up PPP on a ChorusOS system.

Chapter 5 shows you how to set up SLIP on a ChorusOS system.

Chapter 6 discusses the network administration commands available in the ChorusOS 4.0 product.

Chapter 7 discusses the network administration daemons provided in the ChorusOS 4.0 product.

Appendix A explains how to configure a portmaster for use with ChorusOS systems.

Appendix B summarizes how to configure a ChorusOS system image to support networking.

Related Reading

RECOMMEND at least ONE INTRO TO NETWORKING, and ONE NETWORK PROGRAMMING BOOK

Here is where you provide a list of recommended books, related to the tasks that you are describing in this book only. In other words, don't recommend the entire system administration set here:

- The *Point-to-Point Protocol (PPP), Network Working Group RFC 1548*, which describes the standard supported by ChorusOS 4.0
- *SunFoo Developer's Guide*
- *Programming With the SunFoo Language*

Ordering Sun Documents

Fatbrain.com, an Internet professional bookstore, stocks select product documentation from Sun Microsystems, Inc.

For a list of documents and how to order them, visit the Sun Documentation Center on Fatbrain.com at `http://www1.fatbrain.com/documentation/sun`.

Accessing Sun Documentation Online

The docs.sun.com[SM] Web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. The URL is `http://docs.sun.com`.

What Typographic Conventions Mean

The following table describes the typographic changes used in this book.

TABLE P–1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls –a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name%` **`su`** `Password:`

TABLE P–1 Typographic Conventions *(continued)*

Typeface or Symbol	Meaning	Example
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type **rm** *filename*.
AaBbCc123	Book titles, new words, or terms, or words to be emphasized.	Read Chapter 6 in *User's Guide*. These are called *class* options. You must be *root* to do this.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–2 Shell Prompts

Shell	Prompt
C shell prompt	`machine_name%`
C shell superuser prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
Bourne shell and Korn shell superuser prompt	#

CHAPTER 1

Introduction

This introduction briefly describes where network administration usually fits into the ChorusOS puzzle.

This guide provides only a brief description of each of these items in the context of the ChorusOS 4.0 product. If you prefer a gentler introduction or more complete discussion, see “Related Reading” on page for suggestions.

On ChorusOS systems, many network administration tasks are performed during system initialization. In order to better understand how you can intialize and configure a ChorusOS system, it helps to start with a clear view of how network initialization fits into overall system initialization.

In essence, the ChorusOS boot process involves loading the system image, initializing kernel structures and finally starting all boot actors. At the end of the boot process, network structures are not yet initialized and the target is “bare”.

Initializing network structures is the task of the `C_INIT`(1M) boot actor. You need not be familiar at this point with the details of what `C_INIT` does during system initialization, but you do need to know that the `C_INIT` actor loads a special system initialization resource file, `sysadm.ini`(4CC), and executes the commands it contains sequentially.

Thus, `sysadm.ini` should contain all commands needed to set up networking on the ChorusOS target system. The rest of this document focuses on the commands related to networking setup and on the networking capabilities such commands enable.

Basically, `sysadm.ini` allows you to:

- Create network devices and network interfaces
- Execute network adminstration commands to configure network devices and network interfaces
- Start network administration daemons to manage network interfaces.

CHAPTER 2

Network Components

This chapter describes the network interfaces, devices and protocols provided in the ChorusOS 4.0 product. The goal is to help you better understand what is available in the ChorusOS 4.0 product before you begin setting up networking on a ChorusOS system. This chapter does not describe how to set up network components.

Supported Protocols

Network protocols define how messages are formatted and packaged for transmission over the network. They provide support for sockets, which are endpoints for communication described in `socket`(2POSIX).

ChorusOS systems provide support for the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP) over the Internet Protocol (IP). TCP is a high-level, reliable, connection-oriented protocol. In other words, it verifies that messages it sends get to their destinations and resends them if necessary. In order to do so, TCP relies on connections between the sender and the receiver. In contrast, UDP is also high-level, but is unreliable. It sends messages without verifying whether they arrive or not, making UDP faster and less resource-hungry than TCP. Both TCP and UDP sit atop the lower-lever IP, which is a transport protocol. For details about the ChorusOS implementations of these protocols, see `ip`(7P), `tcp`(7P), and `udp`(7P).

ChorusOS systems also support remote inter-process communication (remote IPC), by allowing you to create an IPC stack in the `IOM` system actor and attach the stack to an Ethernet device. For details, see `ethIpcStackAttach`(2K) and `IPC`(5FEA).

Finally, ChorusOS systems provide a mechanism to support Open Systems Interconnect (OSI), by allowing you to attach an OSI stack that you provide to an Ethernet device. For details, see `ethOsiStackAttach`(2K).

Supported Interfaces

ChorusOS systems provide support for Ethernet, loopback, PPP and SLIP network interfaces.

Network interfaces are lower-level than network protocols, providing the link between network hardware and network protocols. Figure 2–1 illustrates how the `ifnet` interface does this for Ethernet connections. Figure 2–2 does the same for the `ppp` and `sl` interfaces for PPP and SLIP connections, respectively.

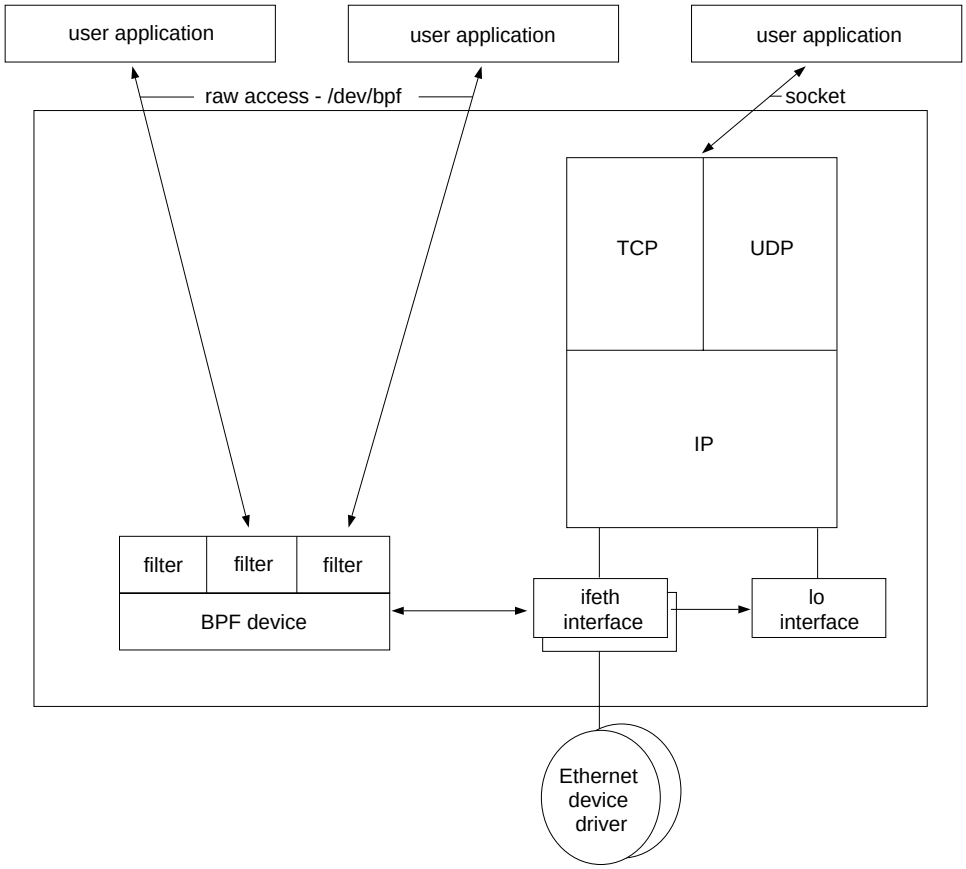

Figure 2–1 Ethernet Networking

The following list enumerates and describes the network interfaces provided:

ifeth*N* The ChorusOS Ethernet interface provides the standard solution for high-speed network connections with other systems.

The Ethernet interface name is built from `ifeth` concatenated with the unit number, *N*.

lo*N*

The loopback interface allows a system to communicate with itself through IP without sending packets over the network. It is a software–only interface and does not depend on network transport hardware.

The loopback interface name is built from `lo` concatenated with a number, *N*, usually `0` unless for some reason you require more than one loopback interface.

ppp*N*

A PPP interface allows the ChorusOS system to connect to another system using a direct serial line or modem connection in the same way as an Ethernet connection.

The PPP interface name is built from `ppp` concatenated with a unit number, *N*.

sl*N*

A SLIP interface also allows the ChorusOS system to connect to another system using a direct serial line or modem connection in the same way as an Ethernet connection.

The SLIP interface name is built from `sl` concatenated with a unit number, *N*.

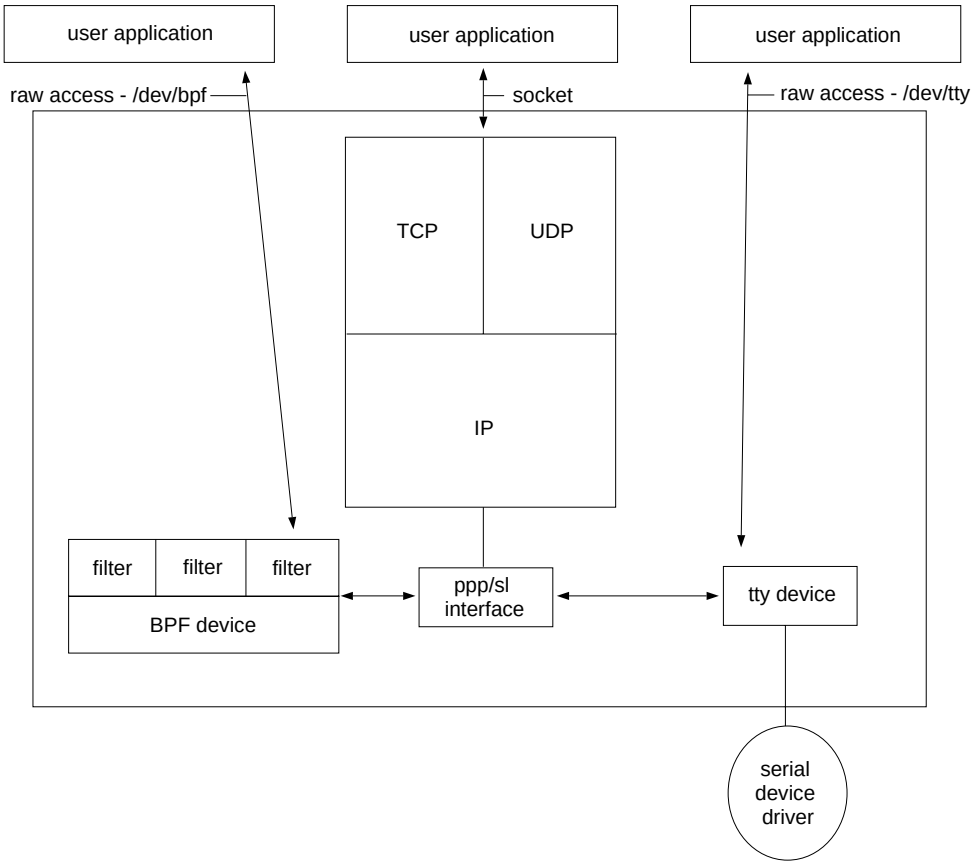

Figure 2–2 PPP/SLIP Networking

Supported Devices

Network devices are low-level drivers that make it easier for applications and interfaces to work closely with network hardware.

ChorusOS systems provide two important types of network devices, Berkeley Packet Filters (BPF) and Teletype (tty, ptyp, ttyp) devices. Each network device described below requires both a device structure and a special file in order to function.

Special files provide access either to peripheral devices, such as serial lines, or to logical “devices”, such as Berkeley packet filters. Each special file:

- Refers to either a block or a raw device. All special files used for networking a raw (character) devices.

- Has a *major* number. Major numbers are used by the system to select the corresponding device driver when several devices are configured.
- Has a *minor* number. Minor numbers are not used directly by the system, but by the selected device driver.

By convention, special files are located in the `/dev` directory. Special files are created, usually during system initialization, using the `mknod`(1M) command.

The following list enumerates and describes the network devices provided:

bpf*N*

A Berkeley Packet Filter is a special character device that allows applications to access all network packets directly, independently of network protocols. It is used in ChorusOS systems by networking utilities such as `dhclient`(1M) and `rarp`(1M).

The BPF device name is built from `bpf` concatenated with a unit number, *N*.

The BPF special file name takes the form `/dev/bpf`*N*.

tty*N*

Teletype character devices date from the time when systems actually wrote output on teletypewriters. ChorusOS systems use tty character devices for serial line communications. The two pseudo-tty devices, ptyp (master) and ttyp(slave), are used by PPP interfaces to detect when IP traffic occurs in order to configure tty devices for dial-up on demand.

The tty device name is built from `tty` concatenated with a unit number, *N*.

The tty special file names takes the form `/dev/tty0`*M*, `/dev/ptyp0`*M*, or `/dev/ttyp0`*M*, where *M* is `1`, `2`, `3`....

CHAPTER 3

Setting Up Ethernet

This chapter takes you through the process of setting up an Ethernet connection for a ChorusOS system. The ChorusOS Ethernet interface provides the standard solution for high-speed network connections with other systems. However, if your system does not support Ethernet, or if you do not plan to use Ethernet with your system, you may omit this chapter.

Hardware Configuration

ChorusOS systems may support multiple Ethernet devices that all have their own Ethernet interface. See the appropriate guide in the *ChorusOS 4.0 Target Family Documentation Collection* for the list of supported Ethernet devices for your particular target system hardware.

If you are unsure which Ethernet device your system uses, yet you are able to boot the system, you can use the `dtree`(1M) utility to display devices listed in the target system device tree:

```
$ rsh target dtree
```

A ChorusOS system often uses the Ethernet interface in conjunction with the Dynamic Host Configuration Protocol (DHCP) client or Reverse Address Resolution Protocol (RARP) to obtain an IP address at boot time. As the system runs, it uses the Address Resolution Protocol (ARP) to find the Ethernet addresses of other systems, based on their IP addresses.

Interface Creation

As described in Chapter 1, many network administration tasks are performed during system initialization when the commands in `sysadm.ini`(4CC) are executed by the `C_INIT` system actor. Interface creation is no exception. In order to create the network interfaces that the system needs to communicate through Ethernet devices, you use the `mkdev`(1M) utility to create one interface for each Ethernet device on your target, and one loopback device for your target as well. The command to create an Ethernet device thus takes the form:

`mkdev ifeth` *unit* [*pathname*]

where *unit* is the number that makes the interface unique and *pathname* is as shown in the output from the `dtree`(1M) utility. If you do not include *pathname* in the command, the ChorusOS system simply attaches the interface to the first free device it finds.

Note - At this point, the network interface is bound to the Ethernet low-level driver, but Ethernet is *not* configured for use.

CODE EXAMPLE 3–1 Ethernet Interface Creation

The following example `sysadm.ini` fragment creates an interface for an Ethernet device and the loopback interface needed for IP communication.

```
#
# Set the file creation mask to 0 during system configuration
#
umask 0

#
# Create an Ethernet interface for the first available device
#
# Since no dtree *pathname* is provided, the system uses the first
# device it finds
#
mkdev ifeth 0

#
# Create a loopback interface
#
# Note that the loopback device is not attached to a device in
# the dtree, so no *pathname* argument is provided here, either
#
mkdev lo 0
```

Device Creation

Ethernet connections do not *require* that you create any devices specifically for them, *unless* you want to configure an Ethernet interface using either the `dhclient`(1M) or `rarp`(1M) utility. Both of these utilities require a Berkeley Packet Filter.

It is also possible that your applications require a BPF device for raw access to network packets.

In order to create the BPF device, you can include commands of the following form in the `sysadm.ini` file that you build into the system image for your target:

```
# Create the BPF device
mkdev bpf unit
# Create the associated special file
mknod /dev/bpfunit c 23 unit
```

where *unit* is the number that makes the interface unique. Note that `mknod`(1M) is used to create special files. The `c` refers to the fact that BPF devices are character devices, and `23` is, by convention, the major number used for BPF devices.

CODE EXAMPLE 3–2 BPF Device Creation

The following example `sysadm.ini` fragment creates an interface for an Ethernet device and the loopback interface needed for IP communication. It then creates a BPF device needed by `dhclient`(1M) and `rarp`(1M).

```
#
# Set the file creation mask to 0 during system configuration
#
umask 0

#
# Create an Ethernet interface for the first available device
#
# Since no dtree *pathname* is provided, the system uses the first
# device it finds
#
mkdev ifeth 0

#
# Create a loopback interface
#
# Note that the loopback device is not attached to a device in
# the dtree, so no *pathname* argument is provided here, either
#
mkdev lo 0

#
# Create a Berkeley Packet Filter device and special file
#
# The BPF device is used by either the rarp or dhclient command to
```

```
# configure the Ethernet interface
#
mkdev bpf 0
mknod /dev/bpf c 23 0
```

Interface Configuration

It it not enough to create an Ethernet interface for Ethernet to work on a ChorusOS system. Once an interface has been created, you must configure it. Configuring the interface means providing an IP address, a subnet mask and a broadcast mask, and setting the status of the interface to `up` or `down`, depending on whether you want it to be operational.

Three utilities are provided to configure Ethernet interfaces on ChorusOS systems:

ifconfig `ifconfig`(1M) can be used to configure as many Ethernet interfaces as necessary as long as the IP addresses for those interfaces are fixed.

rarp `rarp`(1M) allows the ChorusOS system to configure an Ethernet interface with an IP address based on the Ethernet device address using the default subnet mask and broadcast address. This utility is specific to the ChorusOS 4.0 product. It requires a BPF device on the ChorusOS system and a RARP server on the local network.

dhclient `dhclient`(1M) allows the ChorusOS system to configure all parameters of the Ethernet interface, and also configure the routing tables. It requires a BPF device on the ChorusOS system and a DHCP server on the local network. It also requires that you build the `dhclient.r` actor into the ChorusOS system image for your target.

The three examples that follow demonstrate how to use the utilities described above.

CODE EXAMPLE 3–3 Ethernet Configuration with `ifconfig`

The following example `sysadm.ini` fragment creates an interface for an Ethernet device and the loopback interface needed for IP communication. It then configures the Ethernet interface.

```
#
# Set the file creation mask to 0 during system configuration
#
umask 0
```

```
#
# Create an Ethernet interface for the first available device
#
# Since no dtree *pathname* is provided, the system uses the first
# device it finds
#
mkdev ifeth 0

#
# Create a loopback interface
#
# Note that the loopback device is not attached to a device in
# the dtree, so no *pathname* argument is provided here, either
#
mkdev lo 0

#
# Set the file creation mask back to the default value
#
umask 22

#
# Configure the Ethernet interface using ifconfig
#
# Note that ADMIN_IFCONFIG must be set to true in order for this to
# work
#
ifconfig ifeth0 129.157.197.88 netmask 0xffffff00 broadcast 129.157.197.255 up

#
# Configure the loopback interface as well
#
ifconfig lo0 127.0.0.1 up
```

CODE EXAMPLE 3–4 Ethernet Configuration with `rarp`

The following example `sysadm.ini` fragment creates an interface for an Ethernet device and the loopback interface needed for IP communication. It then creates a BPF device needed by `rarp`(1M) and configures the interface.

```
#
# Set the file creation mask to 0 during system configuration
#
umask 0

#
# Create an Ethernet interface for the first available device
#
# Since no dtree *pathname* is provided, the system uses the first
# device it finds
#
mkdev ifeth 0

#
# Create a loopback interface
```

```
#
# Note that the loopback device is not attached to a device in
# the dtree, so no *pathname* argument is provided here, either
#
mkdev lo 0

#
# Create a Berkeley Packet Filter device and special file
#
# The BPF device is used by either the rarp or dhclient command to
# configure the Ethernet interface
#
mkdev bpf 0
mknod /dev/bpf c 23 0

#
# Set the file creation mask back to the default value
#
umask 22

#
# Configure the Ethernet interface using rarp
#
# Note that BPF and ADMIN_RARP must be set to true in order for this
# to work
#
# rarp requires a BPF device
#
rarp ifeth0

#
# Configure the loopback interface as well
#
ifconfig lo0 127.0.0.1 up
```

CODE EXAMPLE 3–5 Ethernet Configuration with `dhclient`

The following example `sysadm.ini` fragment creates an interface for an Ethernet device and the loopback interface needed for IP communication. It then creates a BPF device needed by `dhclient`(1M) and configures the interface.

```
#
# Set the file creation mask to 0 during system configuration
#
umask 0

#
# Create an Ethernet interface for the first available device
#
# Since no dtree *pathname* is provided, the system uses the first
# device it finds
#
mkdev ifeth 0

#
# Create a loopback interface
```

```
#
# Note that the loopback device is not attached to a device in
# the dtree, so no *pathname* argument is provided here, either
#
mkdev lo 0

#
# Create a Berkeley Packet Filter device and special file
#
# The BPF device is used by either the rarp or dhclient command to
# configure the Ethernet interface
#
mkdev bpf 0
mknod /dev/bpf c 23 0

#
# Set the file creation mask back to the default value
#
umask 22

#
# Configure the Ethernet interface using dhclient
#
# Note that BPF must be set to true in order for this to work
#
# dhclient requires a BPF device
#
# dhclient also requires that the dhclient.r actor be included in the
# system image
#
arun /image/dhclient ifeth0 &

#
# dhclient does not return until the server has responded
#
# Wait until the Ethernet interface is up to avoid confusing the system
#
ifwait ifeth

#
# Configure the loopback interface as well
#
ifconfig lo0 127.0.0.1 up
```

Trying Out the Interface

Once you think the interface is properly configured, verify that the Ethernet connection is working:

```
$ rsh target ping host
```

CHAPTER 4

Setting Up PPP

This chapter takes you through the process of setting up a Point-to-Point Protocol (PPP) interface on a ChorusOS system. PPP allows the target system to connect to another system using a direct serial line or modem connection in the same way as an Ethernet connection. If your system does not support serial connections, or if you do not plan to use PPP with your system, you may omit this chapter.

This chapter also describes how to enable PPP on a host workstation for use with a ChorusOS system during application development.

Hardware Configuration

ChorusOS systems may support multiple PPP interfaces, each of which is identified by a unique name such as `ppp0`. Depending on the number of serial lines physically available on the ChorusOS system, the binary distribution of the ChorusOS 4.0 product provides support for up to two PPP lines. (If you have the source distribution, you can increase the number of lines possible by modifying the value of `NPPP` in the `ppp.h` header file.) See the appropriate guide in the *ChorusOS 4.0 Target Family Documentation Collection* to verify serial line support for your particular target system hardware.

Note - The first serial line on the ChorusOS system is reserved for system debug, and for console access using the host workstation `tip`(1) utility. It cannot be used for PPP.

If you are unsure what serial lines your system uses, yet you are able to boot the system, you can use the `dtree`(1M) utility to display devices listed in the target system device tree:

```
% rsh target dtree
```

ChorusOS systems may be used with modems. The `chat`(1M) utility can help you handle modem connection and configuration. Note that this document does *not* provide information about modem configuration commands.

Interface Creation

As described in “Interface Creation” on page 24, interface creation is usually performed by including commands in the `sysadm.ini` file that you build into the ChorusOS system image for your target. In order to create a PPP interface, use the `mkdev`(1M) utility as follows:

`mkdev ppp` *unit*

where *unit* is the number that makes the interface unique. In order to allow the system to use IP, you must also create a loopback interface.

CODE EXAMPLE 4–1 PPP Interface Creation

The following example `sysadm.ini` fragment creates a PPP interface and the loopback interface needed for IP communication.

```
#
# Set the file creation mask to 0 during system configuration
#
umask 0

#
# Create a PPP interface
#
mkdev ppp 0

#
# Create a loopback interface
#
mkdev lo 0
```

Note that the above commands simply create the interface; nothing is configured for use, yet.

Device Creation

PPP interfaces rely on tty devices to communicate directly with the serial hardware. You must therefore create at least one tty device for each PPP line you intend to open If the ChorusOS system is configured to “dial on demand”, you must also create a pseudo-tty master (ptyp) and a pseudo-tty slave (ttyp) device for each PPP line.

It is also possible that your applications require a BPF device for raw access to network packets.

CODE EXAMPLE 4–2 PPP Device Creation

The following example `sysadm.ini` fragment creates a PPP interface and the loopback interface needed for IP communication. It then creates the tty devices needed to open a PPP line that may be configured for “dial on demand”. Finally, it creates a BPF device for any applications that require one.

```
#
# Set the file creation mask to 0 during system configuration
#
umask 0

#
# Create a PPP interface
#
mkdev ppp 0

#
# Create a loopback interface
#
mkdev lo 0

#
# Create a tty device for the second serial port
#
# The first serial port is reserved for system debug and console access
# through the tip(1) utility on the host workstation
#
# If you do not know the device tree pathname to the second serial port,
# note that /pci/pci-isa/ns16650-2 is visible in the output of dtree(1M)
#
# Note that major number 0 is conventionally reserved for ttys
#
mkdev tty 0 /pci/pci-isa/ns16650-2
mknod /dev/tty01 c 0 0

#
# Create pseudo-tty devices for on-demand dialing
#
mknod /dev/ptyp0 c 5 0 # Master
mknod /dev/ttyp0 c 6 0 # Slave

#
```

```
# Create a Berkeley Packet Filter device and special file
#
mkdev bpf 0
mknod /dev/bpf c 23 0
```

PPP Configuration

This section explains how to configure Point-to-Point Protocol (PPP) with the ChorusOS 4.0 product standard interface and drivers. It assumes the serial line or modem is already physically connected. On some platforms, a unique serial line is used both for the console (accessed through `tip`), and for PPP. At boot time, the serial line connector must be connected to the host `tip` line.

Note - Systems often have two IP addresses: one for Ethernet, one for PPP.

This document differentiates between Ethernet Internet addresses and PPP addresses. As you read, be aware that *hostname* refers to the Ethernet hostname corresponding to the Ethernet IP address, and that *hostname_PPP* refers to the hostname corresponding to the PPP address.

PPP as implemented in ChorusOS 4.0 handles IP traffic differently than previous releases.

- First, you run `pppstart`, a daemon that listens for requests for connection.
- Next, you use the `pppd` command to enable a connection to a specific serial line.
- When finished with a PPP connection, you may close it using the `pppclose` command.
- Finally, you can disable PPP services altogether using the `pppstop` command.

The process therefore remains the same whether you open one PPP line or multiple lines.

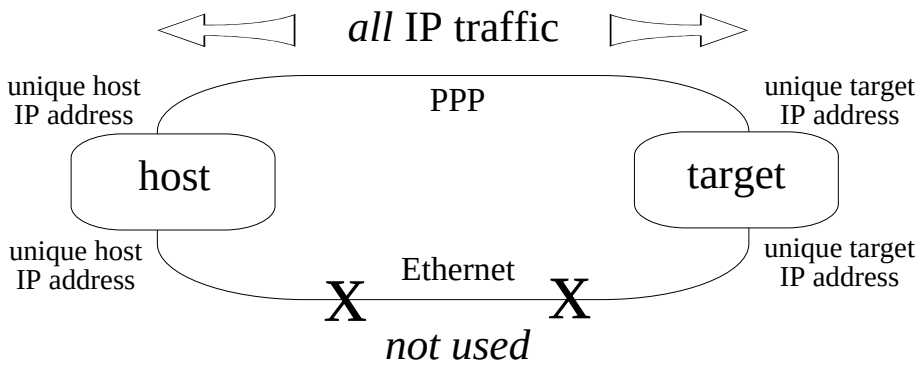

Figure 4–1 IP Traffic—PPP Only

In order to make sure the Ethernet interface can still be used for IP traffic, use different IP addresses for Ethernet and IP.

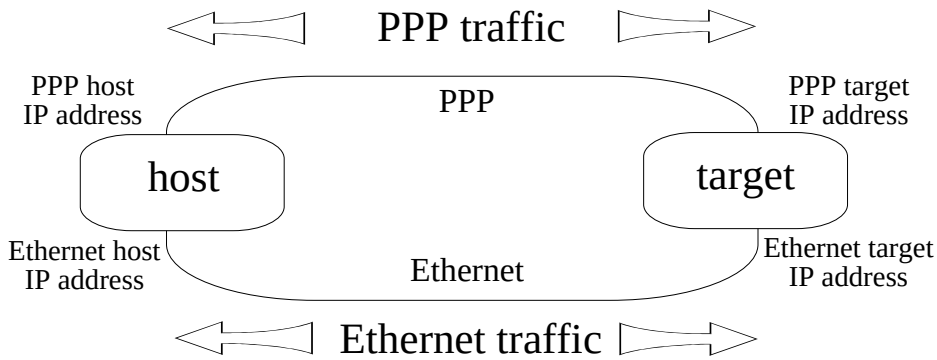

Figure 4–2 IP Traffic—Both Ethernet and PPP

It is *strongly recommended* for this release that you use different IP addresses for PPP and Ethernet.

As shown above, if each system has only one IP address, when an ifnet interface is set up between *target_PPP* and *host_PPP*, *all* traffic to *host_PPP* is routed through *target_PPP*. In other words, all IP traffic passes through the serial line.

How to Enable PPP Services

Enabling PPP services involves using the `pppstart` actor.

▼ Enabling PPP at Boot Time by Including pppstart in the System Image

1. **Change to the directory where you build system images:**

```
host% cd build_dir
```

2. **Ensure that both serial lines and network interfaces are available for PPP by including the necessary commands in the conf/sysadm.ini that you build into the system image. For example:**

```
mkdev tty 0 /pci/pci-isa/ns16550-2      # Create a tty interface
                                        # using the second serial port
                                        # because the first is reserved
                                        # for the tip line

mkdev ppp 0                             # Create a PPP interface that is
                                        # not bound to the tty yet.

mknod /dev/tty01 c 6 0                  # Create a tty special file

#mknod /dev/ttyp0 c 18 0                # Pseudo tty devices, needed
#mknod /dev/ptyp0 c 19 0                # for dialup on demand.

# Enable PPP
# Requires pppstart.r in system image
#
arun /image/sys_bank/pppstart &

pppd /dev/tty01                         # Open a PPP line.

# Wait for the interface to be up.
#
ifwait ppp0
```

3. **Start the ews configuration utility:**

```
host% ews conf/ChorusOS.xml &
```

4. **Use the hints in the table below to set system image features and tunables:**

Set...	Comments
VTTY=true	PPP requires virtual ttys.
PPP=true	
iom.nfs.rsize=1024	Optimizing NFS read and write buffer sizes for use with PPP by setting them to a maximum of 1024
iom.nfs.wsize=1024	

5. **Include the** `pppstart.r` **actor in the system image.**

6. **Rebuild the system image.**

7. **Copy the system image to the boot server:**

```
host% rcp system_image_name boot_server:/tftpboot
```

8. **Reboot the target system:**

```
host% rsh target reboot
```

▼ Enabling PPP Manually

1. **Ensure that both serial lines and network interfaces are available for** PPP**.**

2. **Run the actor:**

```
host% rsh target arun /bin/pppstart &
```

How to Open PPP Lines

Opening an available PPP line involves the built-in `C_INIT` command, pppd.

▼ Opening a PPP Line Manually

1. Run the built-in `C_INIT` command, `pppd`, on the target system:

```
host% rsh target pppd device
pppd device:info: Using interface ppp0
pppd device:notice: Connect: ppp0 <--> device
pppd device:notice: local  IP address local_addr
pppd device:notice: remote IP address remote_addr
```

where *device* is either `/dev/tty01` or `/dev/tty02`.

See `pppd`(1M) for details.

How to Close PPP Lines

Closing an open PPP line involves using the `pppclose` command.

▼ Closing a PPP Line Manually

1. Run the built-in `C_INIT` command, `pppclose`, on the target system:

```
host% rsh target pppclose device
pppd device:info: Terminating on signal 15.
pppd device:notice: Connection terminated, connected for 2 minutes
pppd device:info: Exit.
```

where *device* is either `/dev/tty01` or `/dev/tty02`.

How to Disable PPP Services

Disabling PPP services involves using the `pppstop` command.

▼ Disabling PPP Services

♦ **Run the built-in** `C_INIT` **command,** `pppstop`**, on the target system:**

```
host% rsh target pppstop
```

PPP on a Solaris™ Host Workstation

This section describes how to configure PPP on the Solaris system to work with a ChorusOS 4.0 system.

Note - You must be root to install and configure PPP. If you either do not know the root password for the workstation or do not feel comfortable with the tasks required, ask your system administrator to perform the tasks below.

Checking for Required Packages

Before configuring PPP, check that the following packages are installed on the host workstation:

- SUNWapppr
- SUNWapppu
- SUNWbnur
- SUNWbnuu
- SUNWpppk.

```
host% pkginfo | egrep '.*(bnu|ppp).*'
system      SUNWapppr      PPP/
IP Asynchronous PPP daemon configuration files
system      SUNWapppu      PPP/
IP Asynchronous PPP daemon and PPP login service
system      SUNWbnur       Networking UUCP Utilities, (Root)
system      SUNWbnuu       Networking UUCP Utilities, (Usr)
```

(continued)

(Continuation)

```
system      SUNWpppk       PPP/IP and IPdialup Device Drivers
```

If the packages are not installed, you must install them before you continue.

Configuration Files

This section describes all files which must be created or modified to configure PPP on a Solaris workstation.

▼ Configuring the Network

1. **Make sure** `/etc/hosts` **or the** NIS **configuration files include** PPP **addresses for both the host and target systems:**

```
# If you are using /etc/hosts, then include the following lines.
# Host PPP address          Host PPP hostname
host_PPP                     host_name
# Target PPP address        Target PPP hostname
target_PPP                   target_name
```

▼ Configuring UUCP

1. **Add the system name to** `/etc/uucp/Systems`:

```
# Name          Connect Hour    Device to Use   Connection speed
target          Any             Direct          9600
```

2. **Add the device to** `/etc/uucp/Devices`:

```
# Name     Physical device      Unused     Connection speed    Direct Dialer
#                                                             (no modem)
Direct     cua/a                -          9600                direct
```

3. **Make sure that the dialer called** `direct` **exists in** `/etc/uucp/Dialers`:

```
host% grep direct /etc/uucp/Dialers
direct
```

▼ Configuring PPP

1. **Update the PPP configuration file `/etc/asppp.cf`:**

```
#ident  "@(#)asppp.cf   1.10    93/07/07 SMI"
#
# Copyright (c) 1993 by Sun Microsystems, Inc.
#
# Sample asynchronous PPP /etc/asppp.cf file
#
#
#ifconfig ipdptp0 plumb mojave gobi up
#
#path
#       inactivity_timeout 120     # Approx. 2 minutes
#       interface ipdptp0
#       peer_system_name Pgobi     # The name we log in with (also in
#                                  # /etc/uucp/Systems

ifconfig ipdptp0 plumb host_PPP target_PPP up

path
       interface ipdptp0
       peer_system_name .zsmon
       ipcp_async_map 0xa000
       inactivity_timeout 1000000
```

Terminal Configuration

This section describes how to modify the terminal configuration to allow a PPP connection without login.

▼ Procedure

1. **Start `admintool`.**

2. **Select `Serial Ports` from the `Browse` menu.**

3. **Double-click the port to use for** PPP**.**

4. **Click the** `Expert` **radio button in the** `Detail` **field.**

5. **Use the hints in the table below to set options for the port.**

Field	State/Value
Service Enable	Selected
Baud Rate	`9600`
Initialize Only	Not selected
Bidirectional	Selected
Software Carrier	Selected
Port Monitor Tag	`zsmon`
Connect on Carrier	Selected
Service	`/usr/sbin/aspppls`
Streams Modules	`ldterm,ttcompat`
Timeout (secs)	`Never`

Starting and Stopping PPP

Note - Once properly configured, PPP starts automatically at boot time.

▼ Starting PPP on the Host

1. **Become superuser:**

```
host% su
Password: root password
```

2. **Start** PPP**:**

```
# /etc/init.d/asppp start
```

3. **Make sure** PPP **started successfully:**

```
# netstat -i
Name    Mtu  Net Dest   Address    Ipkts  Ierrs Opkts  Oerrs Collis Queue
lo0     8232 loopback   localhost  1386   0     1386   0     0      0
hme0    1500 host_name  host_name  93612  1714  73065  0     860    0
ipdptp0 8232 target_PPP host_PPP   0      0     0      0     0      0
# tail /etc/log/asppp.log
12:31:34 Link manager (2099) started 11/27/99
12:31:34 parse_config_file: Successful configuration
12:31:40 000531 ipdptp0 SEND PPP ASYNC 29 Octets LCP Config-
Req  ID=6b LEN=24
MRU=1500 ACCM=000a0000 MAG#=cdf1e77f ProtFCOMP AddrCCOMP
```

▼ Stopping PPP on the Host

1. **Become superuser:**

```
host% su
Password: root password
```

2. **Stop** PPP**:**

```
# /etc/init.d/asppp stop
```

Making the PPP Target Visible to the Entire Network

If you want the PPP target to be visible from the whole network, you must enable routing on your workstation.

▼ Enabling Routing

1. **Enable routing:**

```
# ndd -set /dev/ip ip_forwarding 1
```

2. **Use the proxy-arp command to declare the PPP target as having the same Ethernet address as the workstation:**

```
# ypcat ethers | grep host
Ethernet_address host
# arp -s target_PPP Ethernet_address pub
```

PPP on a Windows NT 4.0 Host Workstation

When PPP is used, Windows NT 4.0 allows the ChorusOS target to be fully connected to the enterprise LAN, making host-target communications possible with any network node, not only with the Windows NT 4.0 workstation.

PPP Connection Setup

This section explains how to set up the basic PPP connection between the host workstation and the ChorusOS target system.

▼ Installing the Remote Access Service

Remote Access Service (RAS) and Dial-Up Networking (DUN) provide the basic serial line and modem connection services.

1. **Double-click the** `My Computer` **icon on the desktop.**
2. **Double-click the** `Dial-Up Networking` **icon.**
3. **Install Dial-Up Networking with** `ChorusOS Cable` **as the modem type. The** `ChorusOS Cable` **modem type is created during the initial installation of the host workstation environment.**

▼ Configuring the Serial Cable Connection

1. **Select** `Start | Settings | Control Panel`**.**
2. **Double-click the** `Modems` **icon.**
3. **Check** `Don't detect my modem; I will select it from a list.`**, and then click the** `Next` **button.**
4. **Select** `ChorusOS Cable` **from the list.**
5. **Click the** `Properties...` **button.**
6. **Select** `9600` **in the** `Maximum Speed` **combo box.**
7. **Click the** `Connection` **tab.**
8. **Click the** `Advanced...` **button.**
9. **Check** `Record Log File`**.**
10. **Confirm your selections, closing each dialog box by clicking the** `OK` **or** `Close` **button.**

▼ Configuring the Remote Access Service

Remote Access Service should be configured after you change the modem configuration.

1. **Select** Start | Settings | Control Panel**.**

2. **Double-click the** Network **icon.**

3. **Click the** Services **tab.**

4. **Select** Remote Access Services **from the list.**

5. **Click the** Properties... **button.**

6. **Select** COMx ChorusOS Serial Cable **from the list.**

7. **Click the** Configure... **button.**

8. **Select** Dial-out and receive calls **in the** Port Usage **group, and then click the** OK **button.**

9. **Click the** Network... **button.**

10. **Make sure that only** TCP/IP **is selected in both the** Dial-Out Protocols **and** Server Settings **groups.**

11. **Select** Allow any authentication including clear text **in the** Encryption settings **group.**

12. **Click the** Configure... **button next to the** TCP/IP **checkbox.**

13. **Select** Entire network **in the** Allow remote TCP/IP clients to access **group.**

14. **Select** Use static address pool**.**

15. **Enter the range of** IP **addresses that may be used for** PPP**.**

    The range must contain at least two numbers, as the first number is used by the host workstation and the second by the ChorusOS target system. It is recommended that you use IP addresses on the same network as the Ethernet.

16. **Confirm your selections, closing each dialog box by clicking the** OK **or** Close **button.**

    Windows NT 4.0 reconfigures its network settings and asks to reboot the system. Do *not* reboot the system yet.

▼ Starting the Remote Access Service at Boot Time

1. **Select** `Start | Settings | Control Panel`**.**
2. **Double-click the** `Services` **icon.**
3. **Select** `Remote Access Server` **from the list.**
4. **Click the** `Startup...` **button.**
5. **Select** `Automatic` **in the** `Startup Type` **group.**
6. **Click the** `OK` **button to close the** `Service` **dialog box.**
7. **Click the** `Close` **button to close the** `Services` **dialog box.**

▼ Adjusting the Auto-Disconnect Timeout

By default, the Remote Access Service disconnects incoming connections after twenty minutes of inactivity, which may not be suitable with a ChorusOS system. You can modify the default behavior by editing the Registry.

1. **Select** `Start | Run`**.**
2. **Enter** `regedit` **in the field provided to start the** Registry Editor**.**
3. **Select**

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RemoteAccess\Parameters\Autodisconnect`

in the Registry tree view.

4. **Change the value from** `0x14` **to** `0x0`**.**
5. **Disable the timeout.**
6. **Exit the** Registry Editor**.**
7. **Reboot the Windows NT 4.0 system.**

▼ Checking the Remote Access Service

After rebooting, Windows NT 4.0 listens for incoming PPP calls on the serial line. You can run the Remote Access Admin administration tool to see the state of the port.

1. **Select**
   `Start|Programs|Administrative Tools (Common)|Remote Access Admin`
   to start the application.

2. **If Remote Access Service is not started, select**
   `Server | Start Remote Access Service....`

Authentication

The Windows NT 4.0 workstation PPP host requires that the client provide a valid username and password in order to connect. Both the username and password must be authorized for dial-in access in the Windows NT 4.0 user database.

Currently, the only authentication protocol shared by both ChorusOS systems and Windows NT 4.0 is the plain text Password Authentication Protocol (PAP). Therefore, you must select PAP for logging in. The username and password are the same as the ones used for logging in to the system.

▼ Enabling Automated Target-to-Host Connections

In order to allow automated connections of diskless ChorusOS systems to the Windows NT 4.0 host at boot time, `pppd` supports a ChorusOS specific option, `userpass` *username password*, and its functional equivalent `+ua` *filename*, which makes it possible to obtain the same information from a file.

1. **Edit the** `options` **file to negociate** PAP**, rather than** CHAP **authentication:**

```
# refuse CHAP authentication and require PAP
   refuse-chap
   require-pap
```

2. **Edit the** `pap.scr` **file to include the username and password needed for authentication:**

```
   # pap.scr example
   # username * password
   myuser * pa55word
```

▼ Granting Dial-In Permission on the Host

Use the User Manager to authorize *user* to access the host workstation.

1. **Select**
   `Start|Programs|Administrative Tools (Common)|User Manager`**.**

2. **Double-click** *user* **in the** `Username` **list.**

3. **Click the** `Dialin` **button.**

4. **Select** `Grant dialin permission to user`**.**

5. **Confirm your selections, closing each dialog box by clicking the** `OK` **button.**

Troubleshooting the PPP Connection

The following detailed logs can help you solve connection problems.

%SystemRoot%\ModemLog_ChorusOS_Cable.txt

The modem log traces what happens during the modem connection before PPP communications are established. The modem log is enabled when you select `Record Log File` as part of the modem configuration procedure.

%SystemRoot%\system32\ras\ppp.log

The PPP log traces setup of PPP communications. The PPP log must be explicitly enabled in the Registry, which you can do using the Registry Editor utility.

To start `Registry Editor`, select `Start | Run`, and then run `regedit`.

To enable the PPP log, set the

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RasMan\PPP\Logging`

key to `1`.

▼ On the Host

In order to make the ChorusOS system visible on the local Ethernet network, you must enable routing on the host workstation.

1. **Select** `Start | Settings | Control Panel`**.**
2. **Double-click the** `Networks` **icon.**
3. **Click the** `Protocols` **tab.**
4. **Select** `TCP/IP Protocol` **from the list.**
5. **Click the** `Properties...` **button.**
6. **Click the** `Routing` **tab.**
7. **Check** `Enable IP Forwarding`**.**
8. **Confirm your selections, closing each dialog box by clicking the** `OK` **button.**
9. **Reboot the host system.**

▼ On the ChorusOS Target

In order to route target IP traffic through the host workstation, you must add a default route to the IP address of the Windows NT 4.0 system *after* the connection from the target has been set up using the `defaultroute` option with `pppd`.

♦ **Use the** `route` **command to enable a default route:**

```
$ rsh target route add default IP_address
```

where *IP_address* **is the** IP **address to which** IP **traffic is routed by default.**

Note - You can remove existing default routes using the delete option with the `route` command:

```
$ rsh target route delete default IP_address
```

Sharing IP Addresses with the LAN

When you specify a subset of the addresses used on the local area network for serial line connections between the host workstation and one or more target systems, Windows NT 4.0 activates proxy-ARP for the connected serial clients automatically and behaves like a bridge. Windows NT 4.0 makes it possible, by default, to reach the PPP client from other systems on the Ethernet.

Setting the Connection Speed

In order to set the line speed to `115200` bits per second follow the procedure below.

1. **Set the baud rate when you connect the serial line on the target system:**

```
$ rsh target pppd /dev/tty01 115200
```

2. **Select** `Start | Settings | Control Panel`**.**
3. **Click the** `Modems` **icon.**
4. **Set the** `Maximum Speed` **to** `115200`**.**
5. **Confirm your selections, closing each dialog box by clicking the** `OK` **or** `Close` **button.**
6. **Select** `Start|Programs|Administrative Tools (Common)|Remote Access Admin`**.**
7. **Select** `Server | Stop Remote Access Service...`.
8. **Select** `Server | Start Remote Access Service...`.

9. **Boot the rebuilt ChorusOS system image on the target. It should be immediately visible on the network at the new speed.**

CHAPTER 5

Setting Up SLIP

This chapter takes you through the process of setting up a Serial Line Internet Protocol (SLIP) interface on a ChorusOS system. SLIP allows the target system to connect to another system using a direct serial line or modem connection in the same way as an Ethernet connection. If your system does not support serial connections, or if you do not plan to use SLIP with your system, you may omit this chapter.

This chapter also describes how to enable SLIP on a host workstation for use with a ChorusOS system during application development.

Hardware Configuration

ChorusOS systems may support multiple SLIP interfaces, each of which is identified by a unique name such as `sl0`. The binary distribution of the ChorusOS 4.0 product provides support for only one SLIP line. (If you have the source distribution, you can increase the number of lines possible by modifying the value of `NSL` in the `sl.h` header file.) See the appropriate guide in the *ChorusOS 4.0 Target Family Documentation Collection* to verify serial line support for your particular target system hardware.

Note - The first serial line on the ChorusOS system is reserved for system debug, and for console access using the host workstation `tip`(1) utility. It cannot be used for SLIP.

If you are unsure what serial lines your system uses, yet you are able to boot the system, you can use the `dtree`(1M) utility to display devices listed in the target system device tree:

```
% rsh target dtree
```

ChorusOS systems may be used with modems. The `chat`(1M) utility can help you handle modem connection and configuration. Note that this document does *not* provide information about modem configuration commands.

Interface Creation

As described in “Interface Creation” on page 24, interface creation is usually performed by including commands in the `sysadm.ini` file that you build into the ChorusOS system image for your target. In order to create a SLIP interface, use the `mkdev`(1M) utility as follows:

```
mkdev sl unit
```

where *unit* is the number that makes the interface unique. In order to allow the system to use IP, you must also create a loopback interface.

CODE EXAMPLE 5–1 SLIP Interface Creation

The following example `sysadm.ini` fragment creates a SLIP interface and the loopback interface needed for IP communication.

```
#
# Set the file creation mask to 0 during system configuration
#
umask 0

#
# Create a SLIP interface
#
mkdev sl 0

#
# Create a loopback interface
#
mkdev lo 0
```

Note that the above commands simply create the interface; nothing is configured for use, yet.

Device Creation

SLIP interfaces rely on tty devices to communicate directly with the serial hardware. You must therefore create at least one tty device for each SLIP line you intend to open.

It is also possible that your applications require a BPF device for raw access to network packets.

CODE EXAMPLE 5–2 SLIP Device Creation

The following example `sysadm.ini` fragment creates a SLIP interface and the loopback interface needed for IP communication. Next, it creates a BPF device for any applications that require one.

```
#
# Set the file creation mask to 0 during system configuration
#
umask 0

#
# Create a SLIP interface
#
mkdev sl 0

#
# Create a loopback interface
#
mkdev lo 0

#
# Create a tty device for the second serial port
#
# The first serial port is reserved for system debug and console access
# through the tip(1) utility on the host workstation
#
# If you do not know the device tree pathname to the second serial port,
# note that /pci/pci-isa/ns16650-2 is visible in the output of dtree(1M)
#
# Note that major number 0 is conventionally reserved for ttys
#
mkdev tty 0 /pci/pci-isa/ns16650-2
mknod /dev/tty01 c 0 0

#
# Create a Berkeley Packet Filter device and special file
#
mkdev bpf 0
mknod /dev/bpf c 23 0
```

SLIP Configuration

This section explains how to configure Serial Line Internet Protocol (SLIP) with the ChorusOS 4.0 product standard interface and drivers. It assumes the serial line or modem is already physically connected. On some platforms, a unique serial line is used both for the console (accessed through `tip`), and for SLIP. At boot time, the serial line connector must be connected to the host `tip` line.

Note - Systems often have two IP addresses: one for Ethernet, one for SLIP.

This document differentiates between Ethernet Internet addresses and SLIP addresses. As you read, be aware that *hostname* refers to the Ethernet hostname corresponding to the Ethernet IP address, and that *hostname_SLIP* refers to the hostname corresponding to the SLIP address.

Before starting SLIP, make sure that the target system SLIP interface is correctly registered on the local network with the correct IP address, in particular for NFS access.

This section also assumes that you have configured your ChorusOS system image to support the SLIP connection and create the SLIP interface at boot time, even if your system image does not include everything necessary to *establish a SLIP connection* at boot time.

SLIP as implemented in ChorusOS 4.0 handles attaching the SLIP interface to the serial line using the `slattach` command, and the `chat` command to handle modem connections and authentication.

▼ Enabling SLIP at Boot Time by Including `slattach` in the System Image

1. **Change to the directory where you build system images:**

```
host% cd build_dir
```

2. **Ensure that both serial lines and network interfaces are available for** SLIP **by including the necessary commands in the** `conf/sysadm.ini` **that you build into the system image. For example:**

```
mkdev tty 0 /pci/pci-isa/ns16550-2      # Create a tty interface
                                        # using the second serial port
```

```
                                        # because the first is reserved
                                        # for the tip line.

mkdev sl 0                              # Create a SLIP interface that is
                                        # not bound to the tty yet.

mknod /dev/tty01 c 6 0                  # Create a tty special file

# Simple initialization of direct serial line connection
# Requires slattach in system image
#
arun /image/sys_bank/slattach -s 38400 -l /dev/tty01 &

# Initialization using chat to connect through a modem
# Requires slattach.r, chat.r and chat.cmd in system image
#
#arun /image/sys_bank/slattach -z -r "'/image/sys_bank/chat \
#-f /image/sys_bank/chat.cmd'" /dev/tty01 &

# Configure the SLIP interface
# Syntax: ifconfig *interface* *target_SLIP* *host_SLIP* netmask *NETMASK*
#
ifconfig sl0 129.157.197.88 129.157.197.144 netmask 255.255.255.0
```

3. Start the `ews` configuration utility:

```
host% ews conf/ChorusOS.xml &
```

4. Use the hints in the table below to set system image features and tunables:

Set...	Comments
VTTY=true	SLIP requires virtual ttys.
SLIP=true	
iom.nfs.rsize=512	Optimizing NFS read and write buffer sizes for use with SLIP by setting them to a maximum of 512 (must be less than the Maximum Transfer Unit size).
iom.nfs.wsize=512	

5. Include the `slattach.r` actor in the system image.

6. If you need `chat`, include the `chat.r` actor in the system image.

7. **If you need** `chat` **and it needs a chat script, modify the** `conf/chat.cmd` **file, and include it in the system image. An example chat script follows:**

```
# Abort if we cannot contact the remote system
ABORT            BUSY
ABORT            'NO CARRIER'
# Wait seven seconds max. for the remote system to reply
TIMEOUT          7
# Reset the modem, enable RTS/CTS
''               ATZ
OK               AT&K3
# Dial the number 1234567890
OK               ATDT1234567890
# Wait one minute max. for the connection
TIMEOUT          60
CONNECT
ECHO             OFF
# Wait for the reply
TIMEOUT          7
# Login as chatuser with pa55worD
ogin:-\\r-ogin:  chatuser
ssword:          pa55worD
SAY              "Logged in successfully."
```

8. **Rebuild the system image.**

9. **Copy the system image to the boot server:**

```
host% rcp system_image_name boot_server:/tftpboot
```

10. **Reboot the target system:**

```
host% rsh target reboot
```

▼ Starting SLIP from the Command Line

♦ **Start** SLIP **by running** `slattach` **as an actor:**

```
host% rsh target_SLIP arun /bin/slattach [options]
```

SLIP on a Solaris Host Workstation

This section describes how to configure SLIP on a Solaris host workstation for use with a ChorusOS system.

Note - You must be root to install and configure SLIP. If you either do not know the root password for the workstation or do not feel comfortable with the tasks required, ask your system administrator to perform the tasks below.

▼ Checking for Required Packages

1. **Before attempting to configure** SLIP**, check that the package** `SUNWpcnfs` **is installed on the workstation:**

```
host% pkginfo | grep SUNWpcnfs
networking  SUNWpcnfs     PC-NFS Daemons
```

2. **If the** `SUNWpcnfs` **package is not installed, get the Solstice NFS Client™ 3.1 product, and install the package.**

3. **During installation, reply** `y` **("yes") to the question, "Do you want to install the PC-NFS SLIP driver?", and** `n` **to all other questions.**

▼ Configuring the Network

1. **Make sure** `/etc/hosts` **or the** NIS **configuration files include** SLIP **addresses for both the host and target systems:**

```
# If you are using /etc/hosts, then include the following lines.
# Host SLIP address          Host SLIP hostname
host_SLIP                    host_name
# Target SLIP address        Target SLIP hostname
target_SLIP                  target_name
```

▼ Configuring the Terminal

1. **Start** `admintool`**.**

```
host% admintool &
```

2. **Select** `Go to expert mode` **from serial port configuration menu.**

3. **Unset the** `Service Enable` **flag.**

▼ Starting SLIP on the Host

1. **Start** SLIP**:**

```
# slattach ttya host target baud_rate
```

2. **Make sure the interface is operational:**

```
# netstat –i
Name  Mtu  Net/Dest    Address     Ipkts  Ierrs Opkts  Oerrs Collis Queue
lo0   8232 loopback    localhost   307    0     307    0     0      0
hme0  1500 host        host        14720  268   10827  0     507    0
sl0   1006 target_SLIP host_SLIP   0      0     0      0     0      0
```

▼ Stopping SLIP on the Host

1. **Stop** SLIP**:**

```
# sldetach ttya
```

▼ Routing

If you want the SLIP target to be visible from the whole network, you must enable routing on your workstation.

1. **Enable routing:**

```
# ndd –set /dev/ip ip_forwarding 1
```

2. **Use the proxy-arp command to declare the** SLIP **target as having the same Ethernet address as the workstation:**

```
# ypcat ethers | grep host
Ethernet_address host
# arp –s target_SLIP Ethernet_address pub
```

SLIP on a Windows NT 4.0 Host Workstation

This section describes how to configure SLIP on a Windows NT 4.0 host workstation for use with a ChorusOS system.

Before You Start

Setting up Windows NT 4.0 as a host for SLIP communications is not easy, because Windows NT 4.0 is officially only able to act as a SLIP client.

Establishing communication between the Windows NT 4.0 system and the ChorusOS system on the serial line is relatively straightforward. However, making the ChorusOS system visible on the LAN to which the Windows NT 4.0 host is connected is difficult and is not currently supported.

Please use PPP for communications if you want full routing from the target to the LAN through the Windows NT 4.0 host. The setup of the Windows NT 4.0 host is only explained in order to allow communication with both the ChorusOS target and the LAN, without doing routing between them.

▼ Installing RAS

To install the Remote Access Service (RAS), also called Dial-Up Networking (DUN):

1. **Double-click the** `My Computer` **icon on the desktop.**
2. **Double-click the** `Dial-Up Networking` **icon.**
3. **If the** DUN **is not yet installed, Windows NT 4.0 proposes to install it. Click the** `Install` **button and follow the directions. Install** `ChorusOS Cable` **as the modem type. This modem was created during the installation of the ChorusOS environment.**
4. **If the** DUN **is already installed, click** `Close` **to close the dialog box.**

▼ Configuring the Serial Connection

To configure the serial cable connection:

1. **Open the** `Control Panel`**.**
2. **Double-click** `Modems`**.**
3. **Select** `ChorusOS cable` **in the Modem list-box.**
4. **Click the** `Properties` **button.**
5. **Select** `9600` **in the** `Maximum Speed` **combo box.**
6. **Click the** `Connection` **tab.**
7. **Click the** `Advanced...` **button.**
8. **Check the** `Record Log File` **checkbox.**
9. **Click** `OK`**.**
10. **Click** `OK`**.**

11. Click `Close`**.**

▼ Reconfiguring DUN

Windows NT 4.0 will suggest reconfiguring the RAS because you have changed the modem. If it does not, for example because the DUN was installed earlier, do the following:

1. **Double-click the** `Network` **icon in the** `Control Panel`**.**
2. **Select the** `Services` **tab.**
3. **Select the** `Remote Access Service` **entry in the** `Network Services` **listbox.**
4. **Click the** `Properties` **button.**
5. **Select the** `COMx ChorusOS Cable` **line from the list.**
6. **Click** `Configure...`**.**
7. **Click the** `Dial-out` **radio button in the** `Port usage` **group.**
8. **Click the** `Network...` **button.**
9. **Click** `OK`**.**
10. **Clear every check-box except** `TCP/IP` **in the** `Dial-out Protocols` **group.**
11. **Click** `Continue`**.**
12. **Click** `OK`**.**
13. **Click** `Close`**.**

Windows NT 4.0 will reconfigure its network settings and suggest rebooting the machine. Do so.

▼ Establishing the Connection with the Target

After the machine has rebooted, you will be able to establish the connection with the ChorusOS 4.0 target:

1. **Double-click the** `My Computer` **icon on the desktop.**

2. **Double-click the** `Dial-Up Networking` **icon.**

3. **If** Windows NT 4.0 **displays a message box containing** `The phonebook is empty. Press OK to add an entry.`**, press** `OK`**. If it displays the** `Dial-Up Networking` **dialog box directly, skip the next step.**

4. **Follow the directions in the** `New Phonebook Entry Wizard`**. Leave all fields at their default values. At the end click** `Finish`**. This creates an entry named** `MyDialUpServer`**.**

5. **Select** `MyDialUpServer` **from the** `Phonebook entry to dial` **combo box.**

6. **Click** `More`**.**

7. **Select** `Edit entry and modem properties` **item from the popup menu.**

8. **In the** `Edit Phonebook Entry` **dialog box, select the** `Basic` **tab.**

9. **Make sure that** `ChorusOS Cable (COM1)` **is the value set for the** `Dial using` **field.**

10. **Click** `Configure`**.**

11. **Make sure the speed displayed in the** `Initial speed (bps)` **combo box is the same you have selected for the** `Modem`**.**

12. **Check the** `Enable hardware flow control` **checkbox.**

13. **Clear the** `Enable modem error control` **and** `Enable modem compression` **checkboxes.**

14. **Click** `OK` **to close the box.**

15. **Select the** `Server` **tab.**

16. **Select** `SLIP: Internet` **as the value for the** `Dial-up server type` **field.**

17. **Make sure** TCP/IP **is checked.**

18. **Click** TCP/IP Settings**.**

19. **Fill the** IP Address **field with the** IP **address you have selected for the serial interface on the** Windows NT 4.0 **host in the** SLIP TCP/IP Settings **dialog box. The** SLIP **address must be different from the** IP **address selected on the ChorusOS 4.0 target side. See below for more information on selecting the** IP **addresses.**

20. **Clear** Force IP header compression**. You will not be able to communicate with the ChorusOS 4.0 target if** Force IP header compression **remains checked.**

21. **Clear** Use default gateway on remote network**. If you leave it checked, you lose access to your** LAN **after the communication with the target is established.**

22. **Click** OK **twice. Doing so returns you to the** Dial-Up Networking **window.**

23. **Click** Dial**. Doing so opens the** Connect to MyDialUpServer **dialog box.**

24. **Check** Save password**.**

25. **Click** OK**.**

26. **Click** Do not display this message again **in the** Connection Complete **dialog box.**

27. **Click** OK**.**

You are now connected to the target at the low level. If you have selected addresses which belong to the LAN for your serial line, you must perform a route command, described below, before being able to reach the target from the host.

Note - If the RAS dialer hangs displaying the Connecting to MyDialUpServer message box with All devices connected and a Cancel button which does not do anything, kill both Dial-Up Networking windows from the Applicationstab of the Windows NT Task Manager (press Ctrl-Alt-Del to show it), run Dial-Up Networking again, click the Hang Up button, confirm and then click the Dial button again..

Note that the first IP packet sent to the target after the connection is lost because it is used internally to complete the connection.

If the target system is rebooted, and the line is therefore dropped, you will need to `Dial` again using the `Dial-Up Networking` window. Alternatively, you can run `rasdial MyDialUpServer` from the command line.

It is possible to see a detailed modem-level log, which traces what happens until SLIP communications are started. It is located in the *%SystemRoot%*`\ModemLog_ChorusOS_Cable.txt` file and is enabled during serial cable connection configuration. The log may be useful if problems are encountered.

CHAPTER 6

Network Administration Commands

This chapter describes the utilities you can use to perform network administration tasks on a ChorusOS system. The commands described below apply whether you are using an Ethernet, PPP or SLIP network interface.

arp

The `arp`(1M) utility lets you display and manipulate the tables used to translate IP addresses to Ethernet addresses according to the Address Resolution Protocol (ARP).

CODE EXAMPLE 6–1 Displaying the ARP Table

The following example displays the IP address/Ethernet address pairs known to a ChorusOS system that has no name service daemons operating:

```
$ rsh target arun /bin/arp -a
started aid = 22
? (129.157.197.144) at 8:0:20:a7:d6:f3
```

Note that the only system known to the ChorusOS system is the boot server, and that its hostname is not known.

You may also use a ChorusOS–specific utility called `rarp`(1M) that makes it possible to configure the IP address of the ChorusOS system during system initialization from a RARP server on the local network.

ifconfig

`ifconfig`(1M) allows you both to assign an IP address to a network interface, and to configure network interface parameters. It also allows you to check the interfaces you have configured.

CODE EXAMPLE 6–2 Configuring and Checking Interfaces with `ifconfig`

The following interactive example configures the primary Ethernet and loopback interfaces for *target*, then displays the result.

```
$ rsh target ifconfig ifeth0 129.157.197.88 netmask 0xffffff00 broadcast 129.157.197.255
$ rsh target ifconfig lo0 127.0.0.1 up
$ rsh target ifconfig -a
ifeth0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
 inet 129.157.197.88 netmask 0xffffff00 broadcast 129.157.197.255
 ether 00:e0:29:3c:6c:7f
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
 inet 127.0.0.1 netmask 0xff000000
```

Note that the example above uses the `ifconfig` command that is built into the `C_INIT`(1M) system actor. Thus, if you set the `ADMIN_IFCONFIG` feature to `true` for the ChorusOS system, you could easily adapt the above example to include the commands in the `sysadm.ini`(4CC) system initialization script.

`ifconfig` is also available as a stand-alone actor, `/bin/ifconfig.r`.

netstat

`netstat`(1CC) displays information about network-related data structures, such as network interfaces (use the `–i` option) and routing tables (use the `–r`) option. The utility is available both as a `C_INIT`(1M) built-in command, and as a standalone actor that supports a wider range of options.

CODE EXAMPLE 6–3 Displaying Network Statistics with `netstat`

The following example uses the built-in version of `netstat` to view information about network interfaces and routing tables.

```
$ rsh target netstat -i
Name  Mtu   Network       Address            Ipkts Ierrs    Opkts Oerrs  Coll
ifeth 1500  <Link>      00.e0.29.3c.6c.7f       17     0       10     0     0
ifeth 1500  129.157       129.157.197.88        17     0       10     0     0
lo0   16384 <Link>                              0     0        0     0     0
lo0   16384 127           127.0.0.1             0     0        0     0     0
$ rsh target netstat -r
Routing tables

Internet:
Destination        Gateway            Flags     Refs     Use     Netif Expire
127.0.0.1          127.0.0.1          UH          0        0       lo0
129.157            link#1             UC          0        0
129.157.197.144    8:0:20:a7:d6:f3    UHLW        3       17    ifeth0   1178
```

Note that in order for the built-in version to function, you must set the `ADMIN_NETSTAT` feature to `true` and rebuild the ChorusOS system image.

`netstat` is also available as a stand-alone actor, `/bin/netstat.r`.

nfsstat

The `nfsstat`(1CC) command displays statistics about NFS activity between the server and the client. The ChorusOS implementation of this utility supports only the `–w` option.

CODE EXAMPLE 6–4 Displaying NFS Activity

The following example displays statistics about NFS activity every second:

```
$ rsh target arun /bin/nfsstat -w 1
started aid = 22
         Getattr   Lookup Readlink     Read    Write   Rename   Access  Readdir
Client:      155       51        0      153        0        0      161        2
Server:        0        0        0        0        0        0        0        0
…
```

Note that you can use the `akill`(1M) utility to stop `nfsstat`, which has actor ID 22:

```
$ rsh target akill 22
```

ping

The ChorusOS implementation of `ping`(1M), a basic tool for checking quickly whether a network connection is working or not, requests an ICMP `ECHO_RESPONSE` from the specified host and simply displays "*host* is alive" if the host responds with in 20 seconds.

CODE EXAMPLE 6–5 Checking a Connection with `ping`

The following example using the `ping` utility to check the connection with the system having IP address `129.157.197.1`:

```
$ rsh target ping 129.157.197.1
129.157.197.1 is alive
```

The example below shows what happens when the host does not respond:

```
$ rsh target ping 129.157.197.44
no answer from 129.157.197.44
```

Note that `ping` does not support any options.

route

When a system using IP receives a network data packet, it uses the routing table, managed using `route`(1M), to determine where to send the packet. A properly configured routing table helps the system:

- Deliver packets locally if they are addressed to the system itself.
- Send packets directly if they are addressed to other systems whose IP addresses it knows and with which it has direct connections.
- Send other packets to the gateway system that allows communication with the rest of the Internet.
- Drop any packets to which the above cases do not apply.

IP forwarding allows the system to forward packets to other systems, such as the gateway. IP forwarding is enabled using the `sysctl`(1M) command as shown below.

CODE EXAMPLE 6–6 Routing with IP Forwarding

The following example `sysadm.ini` fragment uses `route` to configure the routing table to deliver packets addressed to the local system (`129.157.197.88`), and to forward other packets to the Ethernet:

```
#
# Enable IP forwarding (requires the sysctl.r actor)
#
arun /bin/sysctl -w IPCTL_FORWARDING=1
# arun /image/sys_bank/sysctl -w IPCTL_FORWARDING=1 # if built into system image

#
# Deliver packets addressed to the local system
#
route add -host 129.157.197.88 lo0

#
# Send other packets back out to the Ethernet
#
route add default -interface ifeth0
```

Note that the first `route` command is unnecessary if the ChorusOS system IP address is assigned dynamically.

`route` is also available as a stand-alone actor, `/bin/route.r`.

ypcat

If the ChorusOS system is bound to a NIS domain, you can use `ypcat`(1CC), `ypmatch`(1CC) and `ypwhich`(1CC) to obtain information from the NIS database.

CODE EXAMPLE 6–7 Reading NIS Information

The following example uses `ypcat` to find “demo” networks mentioned in the NIS database:

```
$ rsh target arun /bin/ypcat networks | grep demo
started aid = 22
demo                    129.157.171
demo2                   129.157.176
```

Note that `ypcat`, `ypmatch` and `ypwhich` require access to the NIS database in order to look up information. See “Name Services and `ypbind`” on page 73 for suggestions about binding to the NIS server for the domain.

CHAPTER 7

Network Adminstration Daemons

This chapter describes the daemons on a ChorusOS system that provide network services. Not all daemons are useful on each system.

Name Services and ypbind

Name services make it possible to convert between IP addresses and system names.

The most basic name service solution on a ChorusOS system consists of using the inetNShost(1CC) daemon that obtains information from the /etc/hosts file, or /etc/networks file.

ChorusOS systems usually rely on other systems to provide name services, however.

The inetNSdns(1M) daemon calls Domain Name Servers listed in /etc/resolv.conf to obtain the information it needs. You can also pass Domain Name server IP addresses to the inetNSdns daemon when you start it.

The inetNSnis(1M) daemon calls the Network Information Services Name Server for the domain you set using the domainname(1CC) command. The inetNSnis command also relies on the portmap(1M) and ypbind(1M) daemons.

CODE EXAMPLE 7–1 Binding to a NIS Server

The following example configures the NIS daemon for the fictitious an.example.COM domain:

```
$ rsh target arun /bin/domainname an.example.COM
started aid = 22
$ rsh target arun /etc/portmap&
```

(continued)

(Continuation)

```
started aid = 22
$ rsh target arun /bin/ypbind
started aid = 23
$ rsh target arun /bin/inetNSnis&
started aid = 23
```

Note that the actors in this example are normally found in a file system outside the system image, such as a root file system located on the host.

The `inetNSien116`(1M) daemon calls a UDP name server as specified in *IEN116* to obtain the information it needs. The IP address of the UDP name server is passed to `inetNSien116` when the daemon is started. Note that the name server causes `gethostbyaddr`(3STDC) to return a `NULL` value.

dhclient

The Dynamic Host Configuration Protocol utility , `dhclient`(1M), allows the ChorusOS system to obtain network information, such as a dynamically assigned IP address, or the IP addresses of the default router and name server, from a DHCP server at boot time.

`dhclient` reads `dhclient.cf`(4CC) and `dhcp.options`(4CC).

ftpd

`ftpd`(1M) provides File Transfer Protocol services on ChorusOS systems.

CODE EXAMPLE 7–2 Sample `ftp` Session

The following example starts the `ftpd_s` server on the ChorusOS system and tries it out:

```
$ rsh target arun /etc/ftpd_s&
started aid = 23
$ ftp target
Connected to target.
220- Welcome to ChorusOS 4.0!
```

(continued)

```
220  FTP server (Version 6.00) ready.
Name (target:user):
331- Password not checked
331 Login ok.
Password:
230- Logging in with home=/
230 User user logged in.
ftp> ls
ls
200 PORT command successful.
150 Opening ASCII mode data connection for 'file list'.
bin
dev
etc
lib
Makefile
226 Transfer complete
30 bytes received in 0.11 seconds (0.28 Kbytes/s)
ftp> bye
bye
221 Goodbye.
$
```

Note that *target* is running in non-secure mode, so no password is required.

nfsd

`nfsd`(1M) provides Network File Services to NFS clients on the network. See the *ChorusOS 4.0 File Systems User's Guide* for details about running an NFS server on a ChorusOS system.

portmap

In order to make RPC calls, the `portmap`(1M) daemon must be running on the ChorusOS system. `portmap` is required both by the `inetNSnis` name service daemon and by the `nfsd` NFS daemon.

When starting `portmap` note that it is located in the `/etc` directory by default:

```
$ rsh target arun /etc/portmap&
started aid = 24
```

pppstart

pppstart(1M) enables client PPP operations on the ChorusOS system. See Chapter 4 for details on configuring a ChorusOS system as a PPP client.

slattach

slattach(1M) attaches a SLIP interface to a serial line. See Chapter 5 for details on configuring a ChorusOS system as a SLIP client.

APPENDIX **A**

Configuring a Portmaster

This appendix briefly describes a configuration using a Livingston Portmaster PM2E.

It is assumed that the target is connected to a port (port S7, for example) of the Portmaster. The serial line must have an inverter.

To configure the Portmaster, ask a system administrator to open a window on the Portmaster, and to set the following values:

```
mode: standard
port type: dialnet
dialnet type: hardwired
protocol: either 'PPP' or 'SLIP'
MTU: 1006
baud rate: 9600 9600 9600
modem control: off
flow control: RTS/CTS
IP destination: either target_PPP or target_SLIP
netmask: 255.255.255.0
routing: none
compression: disabled
```

Press the `apply` button, and the `remote reset` button. If there are any problems, press the `remote reset` button again before starting PPP or SLIP.

APPENDIX **B**

System Image Configuration Summary

This appendix summarizes how to configure your system image to support networking.

TABLE B–1 Networking Features, Tunables, Actors and Scripts

If you want...	Set...	Include the following in the system image...
DHCP (client)	■ BPF=true	■ `dhclient.cf` script ■ `dhclient.r` actor ■ `sysadm.ini`
Ethernet	■ ADMIN_IFCONFIG=true	■ `sysadm.ini`
`ifconfig` (built-in `C_INIT`(1M) command)	■ ADMIN_IFCONFIG=true	
`mount` (built-in `C_INIT`(1M) command)	■ ADMIN_MOUNT=true	
`netstat` (built-in `C_INIT`(1M) command)	■ ADMIN_NETSTAT=true	

TABLE B–1 Networking Features, Tunables, Actors and Scripts *(continued)*

If you want...	Set...	Include the following in the system image...
PPP (client)	■ ADMIN_IFCONFIG=true ■ VTTY=true ■ PPP=true ■ iom.nfs.rsize=1024 ■ iom.nfs.wsize=1024	■ `chat.cmd` chat script (if needed to configure a modem) ■ `chat.r` actor (if needed to configure a modem) ■ `pppstart.r` actor ■ `sysadm.ini`
RARP (client)	■ BPF=true ■ ADMIN_IFCONFIG=true ■ ADMIN_RARP=true ■ BPF=true	■ `sysadm.ini` commands
`route` (built-in `C_INIT`(1M) command)	■ ADMIN_ROUTE=true	
SLIP (client)	■ ADMIN_IFCONFIG=true ■ VTTY=true ■ SLIP=true ■ iom.nfs.rsize=512 ■ iom.nfs.wsize=512	■ `chat.cmd` chat script (if needed to configure a modem) ■ `chat.r` actor (if needed to configure a modem) ■ `slattach.r` actor ■ `sysadm.ini`

TABLE B–2 Networking `sysadm.ini` Commands

If you want…	Adapt the following example `sysadm.ini` commands…

DHCP (client)	`# Create an Ethernet interface`
	`#`
	`mkdev ifeth 0 /pci/epic100`
	`#mkdev ifeth 1 /pci/pci-isa/smc1660-1`
	
	`# Create a Berkeley Packet Filter`
	`mkdev bpf 0`
	`mknod /dev/bpf0 c 20 0`
	
	`# Create a loopback interface`
	`#`
	`mkdev lo 0`
	
	`# Configure the Ethernet interface with DHCP`
	`arun /image/sys_bank/dhclient ifeth0 &`
	`ifwait ifeth0`
	
	`# Configure the loopback interface`
	`#`
	`ifconfig lo0 127.0.0.1 up`

Ethernet	`# Create an Ethernet interface`
	`#`
	`mkdev ifeth 0 /pci/epic100`
	`#mkdev ifeth 1 /pci/pci-isa/smc1660-1`
	
	`# Create a loopback interface`
	`#`
	`mkdev lo 0`
	
	`# Configure the interfaces using ifconfig`
	`# (IP address is known)`
	`#`
	`ifconfig ifeth0 129.157.197.88 netmask 0xffffff00 \`
	`broadcast 129.157.197.255`
	`ifconfig lo0 127.0.0.1 up`

TABLE B–2 Networking `sysadm.ini` Commands *(continued)*

If you want...	Adapt the following example `sysadm.ini` commands...

IP forwarding	`#`
	`# Enable IP forwarding`
	`#`
	`arun /bin/sysctl -w IPCTL_FORWARDING=1`
	
	`#`
	`# Deliver packets addressed to the local system`
	`#`
	`route add -host 129.157.197.88 lo0`
	
	`#`
	`# Send other packets back out to the Ethernet`
	`#`
	`route add default -interface ifeth0`

PPP (client)	`mkdev tty 0 /pci/pci-isa/ns16550-2  # Create a tty interface`
	`# using the second serial port`
	`# because the first is reserved`
	`# for the tip line`
	
	`mkdev ppp 0  # Create a PPP interface that is`
	`# not bound to the tty yet.`
	
	`mknod /dev/tty01 c 6 0  # Create a tty special file`
	
	`#mknod /dev/ttyp0 c 18 0  # Pseudo tty devices, needed`
	`#mknod /dev/ptyp0 c 19 0  # for dialup on demand.`
	
	`# Enable PPP`
	`# Requires pppstart.r in system image`
	`#`
	`arun /image/sys_bank/pppstart &`
	
	`pppd /dev/tty01  # Open a PPP line.`
	
	`# Wait for the interface to be up.`
	`#`
	`ifwait ppp0`

TABLE B–2 Networking `sysadm.ini` Commands *(continued)*

If you want...	Adapt the following example `sysadm.ini` commands...
RARP (client)	

```
# Create an Ethernet interface
#
mkdev ifeth 0 /pci/epic100
#mkdev ifeth 1 /pci/pci-isa/smc1660-1

# Create a loopback interface
#
mkdev lo 0

# Create a Berkeley Packet Filter for RARP
mkdev bpf 0
mknod /dev/bpf0 c 20 0

# Configure the Ethernet interface with RARP
rarp ifeth0

# Configure the loopback interface
#
ifconfig lo0 127.0.0.1 up
```

SLIP (client)

```
mkdev tty 0 /pci/pci-isa/ns16550-2    # Create a tty interface
                              # using the second serial port
                              # because the first is reserved
                              # for the tip line.

mkdev sl 0                       # Create a SLIP interface that is
                              # not bound to the tty yet.

mknod /dev/tty01 c 6 0            # Create a tty special file

# Simple initialization of direct serial line connection
# Requires slattach in system image
#
arun /image/sys_bank/slattach -s 38400 -l /dev/tty01 &

# Initialization using chat to connect through a modem
# Requires slattach.r, chat.r and chat.cmd in system image
#
#arun /image/sys_bank/slattach -z -r "'/image/sys_bank/chat \
#-f /image/sys_bank/chat.cmd'" /dev/tty01 &

# Configure the SLIP interface
#
ifconfig sl0 129.157.197.88 129.157.197.144 netmask 255.255.255.0
```