Morning Star
Express Router

version 1.1.122

User Guide
and
Reference Manual

User Guide Revision
July 18, 1994

Published by
Morning Star Technologies
1760 Zollinger Rd
Columbus OH USA 43221-2856

+1 614 451 1883 (voice)
+1 800 558 7827 (voice)
+1 614 459 5054 (FAX)

`Marketing@MorningStar.Com` (e-mail for sales)
`Support@MorningStar.Com` (e-mail for technical help)
`ftp.MorningStar.Com:pub/Express/*` (anonymous FTP)
`http://www.MorningStar.Com:/` (WWW)

Contents

Chapter 1

General Information

1.1 Product Description

The Morning Star *Express* router provides economical, medium performance wide area internetworking connectivity. It transparently connects a local-area network, based on Ethernet hardware and the TCP/IP protocol suite, with other similar LANs, and with the global Internet.

The Morning Star *Express 2E* router provides economical, medium performance local area internetworking connectivity. It transparently connects local-area networks, based on Ethernet hardware and the TCP/IP protocol suite, with other similar LANs.

The Morning Star *Express PC* router provides all the above, plus the flexibility of customer configuration using popular off-the-shelf hardware components.

Since the members of the *Express* family adhere to the relevant protocol standards, they can be used in environments with other vendors' products that adhere to the same standards.

This document describes the *Express*, the *Express 2E*, and the *Express PC*. Where they are similar, only the *Express* is mentioned. Where they differ, the points of difference are described.

1.1.1 Features

The *Express* routes IP packets between

- One 10 Mbps Ethernet/IEEE 802.3 AUI interface

- Two synchronous or asynchronous RS-232 DB-25 DTE interfaces
  - Asynchronous speeds to 115,200 bits per second
  - Synchronous speeds to 128 Kb/s, externally clocked
  - Async RTS/CTS (“hardware”) or XON/XOFF (“software”) flow control
  - Full modem control (DSR, DTR, CD, RI)
  - Asynchronous SLIP
  - Synchronous or Asynchronous PPP
  - Frame Relay (optional)
- One synchronous V.35 DTE interface
  - Synchronous speeds to T1 (1.544 Mb/s) or E1 (2.048 Mb/s), internally or externally clocked
  - Internally or externally clocked
  - Synchronous PPP
  - Frame Relay (optional)

The *Express 2E* routes IP packets between

- Two 10 Mbps Ethernet/IEEE 802.3 AUI interfaces
- One synchronous or asynchronous RS-232 DB-25 DTE interface
  - Asynchronous speeds to 115,200 bits per second
  - Synchronous speeds to 128 Kb/s, externally clocked
  - Async RTS/CTS (“hardware”) or XON/XOFF (“software”) flow control
  - Full modem control (DSR, DTR, CD, RI)
  - Asynchronous SLIP
  - Synchronous or Asynchronous PPP

The *Express PC* routes IP packets between

- Up to four 10Mbps Ethernet/IEEE 802.3 interfaces
  - At least one NE2000-compatible Ethernet card from Morning Star Technologies
  - Up to three additional NE2000-compatible Ethernet cards

- Up to four asynchronous RS-232 DTE interfaces
  - Each port can be 8250, 16450, 16550, or a USR Sportster internal modem card
  - Asynchronous speeds to 38,400 bits per second (16550)
  - Async RTS/CTS ("hardware") or XON/XOFF ("software") flow control
  - Full modem control (DSR, DTR, CD, RI)
  - Asynchronous SLIP or PPP

The *Express* and *Express 2E* provide front-panel indicators for

- Electrical power (+5V, +12V, -12V)
- Ethernet activity (Carrier and Collision detect on *Express*, Carrier only on *Express 2E*)
- Router internal state and activity

The *Express* can be managed via

- Directly-attached async terminal console
- Telnet session console
- SNMP

1.1.2 Hardware Options and Requirements

The base-model *Express* and *Express 2E* contain 2 Megabytes of static RAM in SIMM packages. The base-model *Express PC* contains 4 Megabytes of static RAM in SIMM packages. This is sufficient memory to run `gated` in an environment that uses RIP for routing management, along with `frd` or `pppd` as required. Running `snmpd` will push a base-model router near its limit of memory utilization, but it should still run well.

If you need to run OSPF, BGP, or EGP, or `snmpd` along with several copies of `pppd` or `frd`, you'll need to use an *Express* that contains its maximum capacity of 8 Megabytes of static RAM. You'll also need a copy of the *Express* software that was built with support for those protocols, since they're left out of the standard distribution in order to save memory.

Contact the Marketing group at Morning Star Technologies about memory expansions for the *Express* router. We have had such mixed luck with different

brands of memory that we strongly recommend you purchase your expansion *Express* memory directly from us. We'll be able to give you a size, type, manufacturer, supplier, and brand that we know *for sure* works in our router.

1.1.3 Standards

The *Express* adheres to the relevant standards for each aspect of its operation:

Ethernet IEEE 802.3

TCP/IP over all media
RFC 1122 RFC 1191

Managing IP Routes
RIP RFC 1058

RIP-2 RFC 1387

OSPF RFC 1247

Wide-Area Networking over async or sync serial connections
SLIP RFC 1055 , RFC 1144

PPP RFC 1548 , RFC 1549 , RFC 1332 , RFC 1333 , RFC 1334 , RFC 1144

Frame Relay RFC 1490

Management using SNMP
MIB-II RFC 1213

PPP RFC 1471 , RFC 1472 , RFC 1473

1.2 Typographic Conventions

Throughout this document, output from commands and contents of files will be displayed `like this`, filenames `like this`, hostnames *like this*, commands `like this`, command arguments *like this*, and option names and other literals `like this`.

In the Index, page numbers in *italics* refer to the primary reference for that topic.

1.3 Warranty Disclaimer

The products and specifications, configurations, and other technical information regarding the products contained in this manual are subject to change without notice. All statements, technical information, and recommendations contained in this manual are believed to be accurate and reliable but are presented without warranty of any kind, express or implied, and users must take full responsibility for their application of any products specified in this manual. THIS MANUAL IS PROVIDED "AS IS" WITH ALL FAULTS. MORNING STAR TECHNOLOGIES DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING THOSE OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, OR ARISING FROM A COURSE OF DEALING, USAGE OR TRADE PRACTICE.

IN NO EVENT SHALL MORNING STAR TECHNOLOGIES BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL OR INCIDENTAL DAMAGES, INCLUDING WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF MORNING STAR TECHNOLOGIES HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Some states do not allow limitation of exclusion of liability for consequential or incidental damages or limitation on how long implied warranties last, so the above limitations or exclusions may not apply to you. This warranty gives Customers specific legal rights, and you may also have other rights that vary from state to state.

This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instruction manual, may cause interference to radio communications. This equipment has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J or Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

Notice of Restricted Rights:

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c) of the Commercial Computer Software - Restricted Rights clause at FAR § 52.27-19 and subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS § 252.227-7013. The information in this manual is subject to change without notice.

1.4 Credits and Copyright Information

Many parts of the networking software in the *Express* router are

ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The command line editing software in the *Express* router is

Copyright © 1991, 1992, 1993 Chris Thewalt <thewalt@ce.berkeley.edu>.

Permission to use, copy, modify, and distribute this software for any purpose and without fee is hereby granted, provided that the above copyright notices appear in all copies and that both the copyright notice and this permission notice appear in supporting documentation. This software is provided "as is" without express or implied warranty.

The Gate Daemon software is

Copyright © 1990, 1991, 1992, 1993 Cornell University. All rights reserved.

THIS SOFTWARE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Royalty-free licenses to redistribute GateD Release 3 in whole or in part may be obtained by writing to:

GateDaemon Project
Information Technologies/Network Resources
200 CCC
Cornell University
Ithaca, NY 14853-2601 USA

GateD is based on Kirton's EGP, UC Berkeley's routing daemon (routed), and DCN's HELLO routing Protocol. Development of GateD has been supported in part by the National Science Foundation.

Please forward bug fixes, enhancements and questions to the gated mailing list: <gated-people@gated.cornell.edu>.

Authors:

- Jeffrey C Honig <jch@gated.cornell.edu>

- Scott W Brim <swb@gated.cornell.edu>

The OSPF software in the *Express* router is

> Copyright © 1989, 1990, 1991 The University of Maryland, College Park, Maryland. All Rights Reserved.
>
> The University of Maryland College Park ("UMCP") is the owner of all right, title and interest in and to UMD OSPF (the "Software"). Permission to use, copy and modify the Software and its documentation solely for non-commercial purposes is granted subject to the following terms and conditions:
>
> 1. This copyright notice and these terms shall appear in all copies of the Software and its supporting documentation.
> 2. The Software shall not be distributed, sold or used in any way in a commercial product, without UMCP's prior written consent.
> 3. The origin of this software may not be misrepresented, either by explicit claim or by omission.
> 4. Modified or altered versions must be plainly marked as such, and must not be misrepresented as being the original software.
> 5. The Software is provided "AS IS". User acknowledges that the Software has been developed for research purposes only. User agrees that use of the Software is at user's own risk. UMCP disclaims all warrenties, express and implied, including but not limited to, the implied warranties of merchantability, and fitness for a particular purpose.
>
> Royalty-free licenses to redistribute UMD OSPF are available from The University Of Maryland, College Park. For details contact:
>
> Office of Technology Liaison
> 4312 Knox Road
> University Of Maryland
> College Park, Maryland 20742
> +1 301 405 4209 (voice)
> +1 301 314 9871 (FAX)
>
> This software was written by Rob Coltun <rcoltun@ni.umd.edu>

The DES encryption software in the *Express* router is

The MD5 Message-Digest Algorithm used in the *Express* router is

The *Express* router's Frame Relay software includes parts that are

Parts of the scheduler and telnet software in the *Express* router are

Chapter 2

Installation

2.1 Environmental Requirements

The *Express* is intended to work at ambient temperatures of 10-40 degrees C, with non-condensing humidity.

2.2 Preparing the Hardware

First decide what sort of connections you'll need, and collect appropriate cables. Then decide if you can use the default jumper settings. If not, you must remove the lid, change the jumpers, and replace the lid.

2.2.1 Removing the Lid

(Not applicable to the *Express PC*)

The lid is held on by a single Phillips head screw at the back. After the screw is removed, the lid may be removed by gently prying it up from the back.

2.2.2 Setting the Jumpers

Setting Jumpers on 1-Ethernet *Express*

The jumpers shown in Figure 2.1 have these meanings:

J0 Run memory diagnostic before booting (default: **in**)

J1 Debugging mode (default: **in**)

If this jumper is removed, the router will halt for debugging before finishing initialization. This jumper should only be removed at the instruction of Morning Star Technologies support staff.

J2 Disable security (default: **in**)

If this jumper is removed, a 1-Ethernet *Express* router will boot with the following configuration:

```
console tty1
console tty2
version
ifconfig lo0 127.1
ifconfig enet0 192.0.2.1
inetd
getty tty1 9600 nowait respawn
getty tty2 9600 nowait respawn
ifconfig enet0 rarp
```

Additionally, password checking will be disabled for the *root* account.

This jumper makes it possible to recover from severe misconfiguration.

J3 Make `tty1` the debugging port (default: **in**)

If removed, this jumper will make the debugging port `tty1`. With **J3** installed, the *Express* will use UDP for debugging. No debugging can be done at all unless **J1** is installed or if the `gdb` command has been issued.

J4 Print stack trace (default: **in**)

If removed, the router will print a stack dump to the console after a crash.

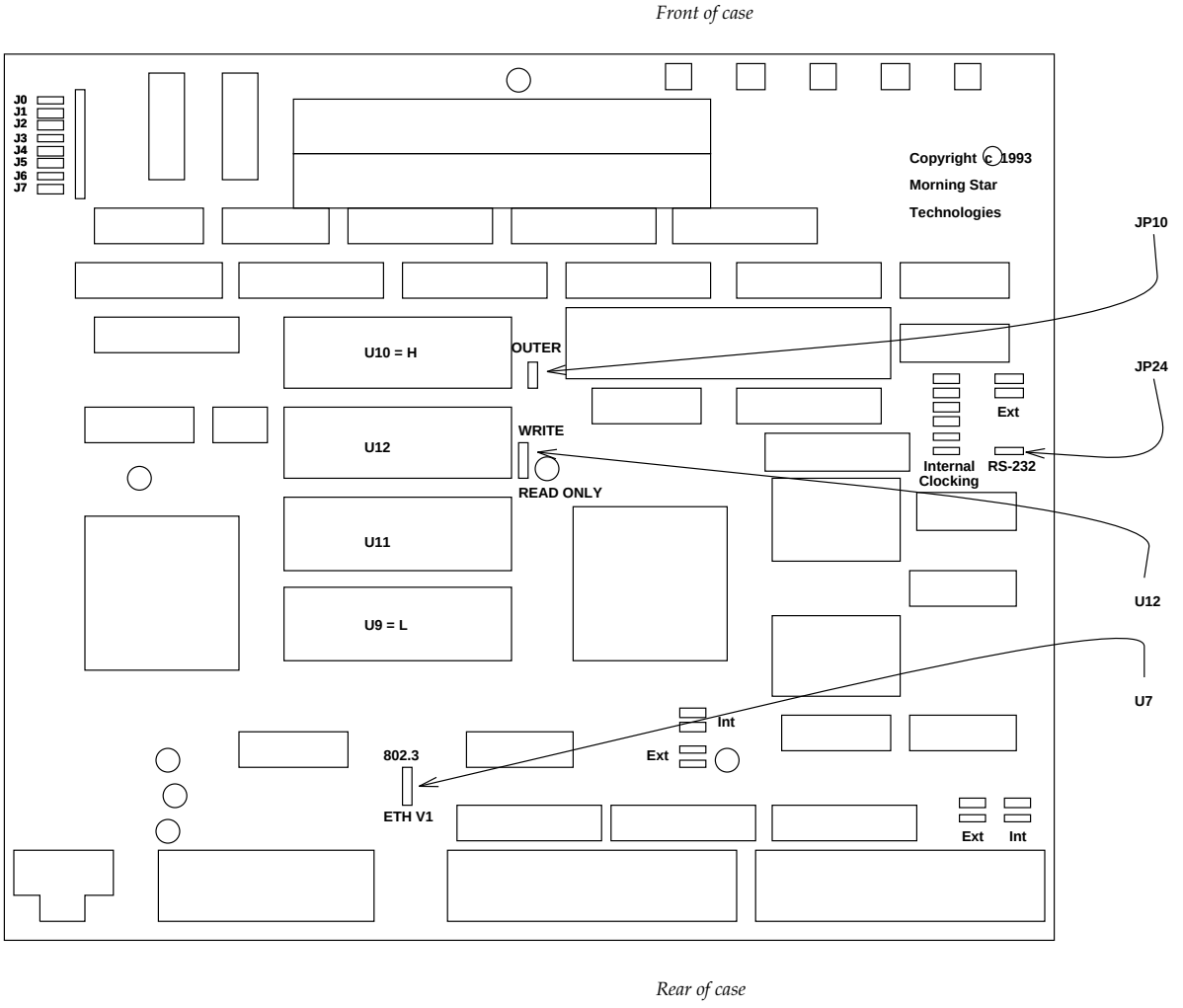

Figure 2.1: Jumper Locations for *Express*

J5 Not used

J6 Not used

J7 Not used

JP1 Factory test (default: **in**)

This jumper is removed only during factory testing.

JP10 Boot from outer flash sockets (default: **in**)

This jumper selects which pair of flash sockets contain the boot code. It is unlikely that you will want to change this except after receiving updated software in Flash from MST.

JP24 Make `tty0` RS-232. (default: **out**)

If installed, this jumper will cause `tty0` to use RS-232 signaling instead of V.35 signaling. Since a special cable is required for RS-232, this jumper is normally left out.

U7 Ethernet type (default: **802.3**)

This jumper may be set to either **802.3** or **Ethernet Version 1**. Most modern Ethernet networks are **802.3**.

U12 Write protect for flash memory (default: **WRITE**)

This jumper is normally in the **WRITE** position. If moved to the **READ-ONLY** position, it will not be possible to change the saved configuration files.

U25 Factory setting only (default: **nearest card edge**)

Do not move this jumper from its factory setting.

Int Internal clocking (default: **out**)

If in, these jumpers cause the serial port nearest to them to use internal clocking. They should not be in if the **Ext** jumpers are also in.

Ext External clocking (default: **in**)

If in, these jumpers cause the serial port nearest to them to use external clocking. They should not be in if the **Int** jumpers are also in.

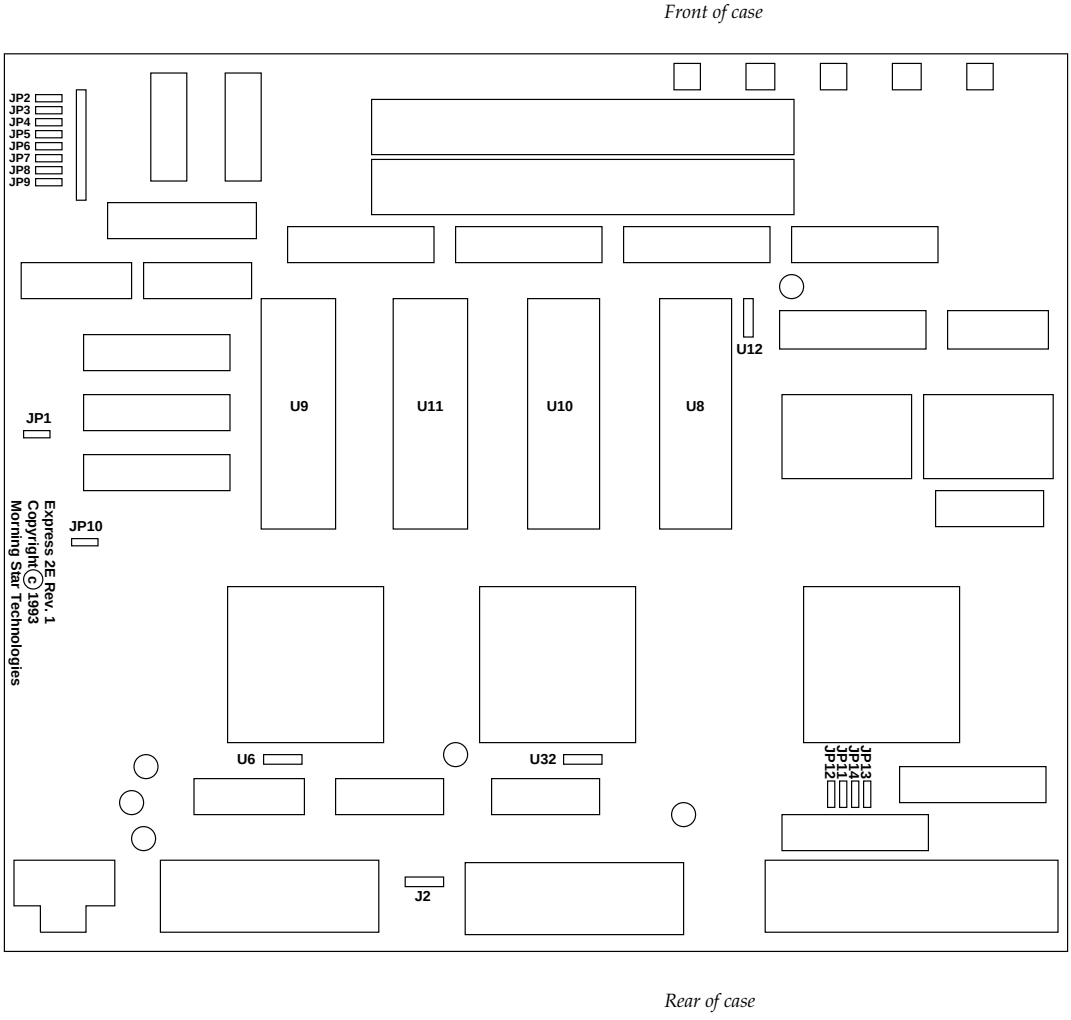

Figure 2.2: Jumper Locations for *Express 2E*

Setting Jumpers on 2-Ethernet *Express 2E*

The jumpers shown in Figure 2.2 have these meanings:

JP1 Factory test (default: **in**)

This jumper is removed only during factory testing.

JP2 Run memory diagnostic before booting (default: **in**)

JP3 Debugging mode (default: **in**)

If this jumper is removed, the router will halt for debugging before finishing initialization. This jumper should only be removed at the instruction of Morning Star Technologies support staff.

JP4 Disable security (default: **in**)

If this jumper is removed, a 2-Ethernet *Express 2E* router will boot with the following configuration:

```
console tty0
version
ifconfig lo0 127.1
ifconfig enet0 192.0.2.1
ifconfig enet1 192.0.3.1
inetd
getty tty0 9600 nowait respawn
ifconfig enet0 rarp
ifconfig enet1 rarp
```

Additionally, password checking will be disabled for the *root* account.

This jumper makes it possible to recover from severe misconfiguration.

JP5 Make `tty1` the debugging port (default: **in**)

If removed, this jumper will make the debugging port `tty1`. With **J3** installed, the *Express* will use UDP for debugging. No debugging can be done at all unless **J1** is installed or if the `gdb` command has been issued.

JP6 Print stack trace (default: **in**)

If removed, the router will print a stack dump to the console after a crash.

JP7 Not used

JP8 Not used

JP9 Not used

JP10 Boot from outer flash sockets (default: **in**)

U6 If in the position closest to the "U6" legend, enet0 conforms to Ethernet Version 1. If in the position farthest from the "U6" legend, enet0 conforms to IEEE 802.3. (default: **802.3**)

U32 If in the position closest to the "U32" legend, enet1 conforms to Ethernet Version 1. If in the position farthest from the "U32" legend, enet1 conforms to IEEE 802.3. (default: **802.3**)

J2 If jumpers are placed on any of these pins, the *Express 2E* will either reset itself or interrupt itself at IPL7. Jumpers should never be placed on any of these pins.

JP12 and JP11
If in, these jumpers cause tty0 use external clocking when running synchronous PPP. They should not be in if the **JP14 and JP13** jumpers are also in. (default: **in**)

JP14 and JP13
If in, these jumpers cause tty0 use internal clocking when running synchronous PPP. They should not be in if the **JP12 and JP11** jumpers are also in. (default: **out**)

U12 Write protect for flash memory This 3-pin jumper block is normally in the **WRITE** position, with the jumper on the two pins closest to the **U12** legend. If moved to the **READ-ONLY** position on the two pins farther from the **U12** legend, it will not be possible to change the saved configuration files. (default: **WRITE**)

Setting "Jumpers" on *Express PC*

Since the generic PC hardware used in the *Express PC* has no jumpers in the same sense as the custom hardware in the *Express* and the *Express 2E*, the *Express PC* software instead gets its configuration "jumper" information from a file on its floppy disc. The file named `jumpers` contains a string of `0` and `1` digits, numbered from

right to left with the bit corresponding to J0 on the right and the bit corresponding to J7 on the left. In each case, if a digit is `1` it indicates that the "jumper" is "in", and if a digit is `0` it indicates that the "jumper" is "out".

Like other files on the *Express PC*, `jumpers` can be edited on the *Express PC* using the router's native editor, or via FTP from another system, or by using a text editor on any MS-DOS system. The *Express PC* sees the file as being named `jumpers`, but a MS-DOS system will see it as `JUMPERS.` Note the trailing period (`'.'`), because of the limitations of filenames on MS-DOS.

The "jumpers" have these meanings:

J0 Not used

J1 Not used

J2 Disable security (default: `1`) If this jumper is `0`, an *Express PC* router will boot with the following configuration:

```
console tty0
version
ifconfig lo0 127.1
ifconfig ed0 192.0.2.1
inetd
getty tty0 9600 nowait respawn
ifconfig ed0 rarp
```

Additionally, password checking will be disabled for the *root* account.

This jumper makes it possible to recover from severe misconfiguration.

J3 Not used

J4 Print stack trace (default: `1`)

If `0`, the router will print a stack dump to the console after a crash.

J5 Not used

J6 Not used

J7 Not used

2.2.3 Replacing the Lid

(Not applicable to the *Express PC*)

Set the lid of the router front down on a flat surface. Gently insert the base into the lid, pressing down on the back to force the back panel under the lid's rear lip. Replace the Phillips screw to hold the top securely.

2.3 Cabling

You *must* use shielded cabling with the Morning Star *Express* in order to comply with the United States Code of Federal Regulations 47 Ch. 1 (10-1-92 Edition) §15.19(c):

> This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference than may cause undesired operation.

2.3.1 Synchronous Serial Cabling

(Not applicable to the *Express PC*)

RS-232

RS-232 cables require the following pins:

1	Frame ground
2	Transmit Data
3	Receive Data
4	Request to Send [1] [2]
5	Clear to Send [1] [2]
6	Data Set Ready [1] [3]
7	Signal Ground
8	Received Line Signal Detect/Carrier Detect [1] [4]
15	Transmit Clock [1]
17	Receive Clock [1]
20	Data Terminal Ready [4]
22	Ring Indicator [3]

V.35

V.35 cables require the following pins:

A	Chassis Ground
B	Signal Ground
C	Request To Send
D	Clear To Send
E	Data Set Ready
F	Receive Line Signal Detect
H	Data Terminal Ready
P	Transmitted Data
R	Received Data
S	Transmitted Data
T	Received Data
V	Receive Timing
X	Receive Timing
Y	Transmit Timing
AA	Transmit Timing

[1] Required in synchronous circuits
[2] Required for hardware flow control
[3] One of these two is required for dialup operation
[4] Required for dialup operation

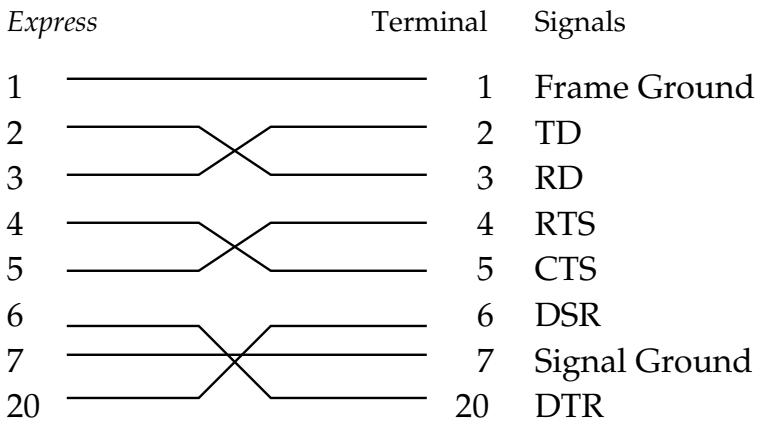

Figure 2.3: Null-Modem Cable

2.3.2 Configuring the Console

Express* and *Express 2E

The default console configuration for `tty1` and `tty2` is 9600 baud, 8 bits, no parity, 1 stop bit.

Express PC

The default console configuration for `tty0` is 9600 baud, 8 bits, no parity, 1 stop bit. `tty0` will be associated with the keyboard and monitor if one is present, or on the first serial port found on the boards COM1 through COM4.

Null-Modem Async Serial Console Cable

To connect the console to a terminal you will need a null-modem adapter that crosses wires as described in figure 2.3 on page 27. Pin 6 (DSR) is crossed with pin 20 (DTR), and pin 7 (Signal Ground) connects straight through, connecting with neither pin 6 nor pin 20.

2.3.3 Ethernet

Express* and *Express 2E

The Ethernet connector, `enet0` (also `enet1` on the *Express 2E*), provides a standard AUI interface. Connect either an AUI cable or a miniature transceiver, then secure it with the slide lock.

Express PC

The Ethernet interfaces, `ed0` through `ed3`, can present AUI, BNC, or Twisted Pair physical interfaces to the network, depending upon the type of board installed.

2.4 Applying Power

2.4.1 *Express* and *Express 2E*

Press the power supply plug into the round socket marked *Power*. The plug and socket are keyed so that the plug cannot be inserted incorrectly.

When power is applied, the +5V, +12V, and -12V green LEDs should light. If they do not light or if they flash, remove power and return to the factory.

During the boot process, the 7-segment LED displays go through a series of different patterns. They will display a representation of **bo** with a flashing period while booting, then **cl** while clearing memory, followed by **rc** while running the `rc.boot` file. If you see any other patterns in the display, see section B.2 on page 211 for keys to their meanings.

After the router software is running, the period flashes every 4 seconds, with the vertical segments showing activity on (left to right, respectively) `enet0`, `tty0`, `tty1`, and `tty2`.

2.4.2 *Express PC*

Connect a standard power cable to the socket on the *Express PC*.

There are no status indicator lights on the *Express PC* other than those provided with the chassis, and they should indicate that the *Express PC* is receiving power from its power cable.

During the boot process, the floppy drive should be active for about 30 seconds. Its status light should indicate activity.

2.4.3 The First Boot

When delivered, your *Express* will configure itself with this `rc.boot` script:

```
console tty1
console tty2
version
ifconfig lo0 127.1
ifconfig enet0 192.0.2.1
inetd
getty tty1 9600 nowait respawn
getty tty2 9600 nowait respawn
ifconfig enet0 rarp
```

Wnen delivered, your *Express 2E* will configure itself with this `rc.boot` script:

```
console tty0
version
ifconfig lo0 127.1
ifconfig enet0 192.0.2.1
ifconfig enet1 192.0.3.1
inetd
getty tty0 9600 nowait respawn
ifconfig enet0 rarp
ifconfig enet1 rarp
```

When delivered, your *Express PC* will configure itself with this `rc.boot` script:

```
console tty0
version
ifconfig lo0 127.1
ifconfig ed0 192.0.2.1
inetd
getty tty0 9600 nowait respawn
ifconfig ed0 rarp
```

A serial terminal (or, on the *Express PC*, a display and keyboard) attached to a device named in a `console` command will act as a console, with which you can proceed through the configuration process.

If no terminal or display is available for use as a console, you can attach the router to an Ethernet LAN, and `telnet` to it from some other system already installed on the LAN. You can either

- Configure the IP address of the other host's Ethernet interface to be on the 192.0.2.0 network, or
- Configure a RARP server on the LAN to respond to the *Express* router's RARP queries by furnishing an IP address that's on the same network number as the rest of the hosts on the LAN. The router's Ethernet MAC address is printed on the bottom of its enclosure, and consists of six hexadecimal numbers separated by colons. A Sun's `/etc/ethers` might include a line like

  ```
  0:3:c6:0:0:42 express
  ```

  The Sun's `/etc/hosts` file would need a line like

  ```
  192.5.58.18 express
  ```

  Use an IP address that's appropriate for your network.

2.5 Troubleshooting

Here are a few initial troubleshooting suggestions, if the hardware doesn't behave as expected.

no lights
: Make sure the power supply is plugged into a wall outlet. Make sure the wall outlet is providing power.

  Return to factory.

no `login:` prompt
: Check the speed setting on the terminal.

  Make sure that the router appears to be booting. If the router does not appear to be booting (indicated by **bo** in the 7-segment LED displays and a slowly flashing period), contact Morning Star support for assistance.

  Remove jumper **J2** and reboot. Log in as `root`, then examine the `rc.boot` file to see what is incorrect.

displays o9 while booting
: Your *Express* router is inverted. Roll the enclosure about its longitudinal axis so that the rubber feet are on the table top, and the faceplate is legible without spilling your coffee.

displays other codes

See section B.2 on page 211 for a list of the different diagnostic codes that the *Express* and *Express 2E* might present in their 7-segment displays.

Chapter 3

Configuration

3.1 Hardware Options and Requirements

The base-model *Express* and *Express 2E* contain 2 Megabytes of static RAM in SIMM packages. The base-model *Express PC* contains 4 Megabytes of static RAM in SIMM packages. This is sufficient memory to run `gated` in an environment that uses RIP for routing management, along with `frd` or `pppd` as required. Running `snmpd` will push a base-model router near its limit of memory utilization, but it should still run well.

If you need to run OSPF, BGP, or EGP, or `snmpd` along with several copies of `pppd` or `frd`, you'll need to use an *Express* that contains its maximum capacity of 8 Megabytes of static RAM. You'll also need a copy of the *Express* software that was built with support for those protocols, since they're left out of the standard distribution in order to save memory.

Contact the Marketing group at Morning Star Technologies about memory expansions for the *Express* router. We have had such mixed luck with different brands of memory that we strongly recommend you purchase your expansion *Express* memory directly from us. We'll be able to give you a size, type, manufacturer, supplier, and brand that we know *for sure* works in our router.

3.2 *Express PC* Hardware Configuration

3.2.1 *Express PC* Hardware Requirements

- 80386, 80486, or compatible processor
- Minimum of 4MB of RAM
- 1.44MB 3.5 inch diskette drive
- At least one NE2000-compatible Ethernet card from Morning Star Technologies
- Up to 3 additional NE2000-compatible Ethernet cards
- Up to 4 serial ports (8250, 16450, or 16550 (*16550 recommended*), including USR Sportster internal modem cards

3.2.2 *Express PC* Hardware Configuration Techniques

Insert the Morning Star Technologies *Express PC* diskette in drive A and reset or turn on the PC.

Though the *Express PC* normally runs without any display or input device, you will need to attach a keyboard and monitor/video card long enough to properly set the BIOS parameters. Unless they will be used with the PC-Express permanently, the keyboard and video parameters should be configured as "Not installed". All ROM and video shadow options should be configured as "Disabled".

3.2.3 *Express PC* Hardware Addresses

Ethernet cards

1st card	IRQ10, I/O address 0x300
2nd card	IRQ11, I/O address 0x320
3rd card	IRQ12, I/O address 0x340
4th card	IRQ13, I/O address 0x360

Serial ports

COM1	IRQ4, I/O address 0x3F8
COM2	IRQ3, I/O address 0x2F8
COM3	IRQ2, I/O address 0x3E8
COM4	IRQ5, I/O address 0x2E8

3.3 Interacting With the Command Shell

After login, you are presented with the shell prompt, `#`. The shell has the commands described in Appendix A, beginning on page 77. The shell has an editing mode for commands, which is described on page 133. You can also save sequences of commands into a file which can then be executed by typing its name, for example the commands:

```
netstat -ina
netstat -rn
ifconfig -a
```

if saved into a file called `status` could then be executed with the command `status`. Remember to save any files you want around in the long term into the flash with the `save` command.

3.4 Logging In for the First Time

- Connect a terminal to a console device (`tty1` or `tty2` on a *Express*, `tty0` on an *Express 2E* or *Express PC*) using a null-modem cable. Set the terminal speed to 9600 baud.

  If your *Express PC* has no serial port but instead has a keyboard and video display, that display will be used for `tty0`.

- Login as `root`. There is no password yet; just type a *return*.
- Change the root password immediately with the `passwd` command.
- Save the new password file with the command `save passwd`.

3.5 A Simple Sample Configuration

3.5.1 *Express*

Using a local terminal attached to `tty1` for the console, we assign an IP address to the Ethernet interface (using the default values `netmask 0xffffff00 broadcast 192.0.2.255`), set up outbound demand-dialed PPP using the Telebit T3000 modem on `tty2`, and frame relay running at T1 on `tty0`, the V.35 connector. Here are the commands we put in the `rc.boot` file:

```
console tty1
hostname thisone
version
ifconfig lo0 127.1
ifconfig enet0 192.0.2.1
frd tty0 addr 192.0.76.144
pppd 192.0.2.1:192.0.2.2 auto idle 120
getty tty1 9600 nowait respawn
getty tty2 57600 respawn
```

and in the `Systems` file:

```
192.0.2.2 Any ACU 57600 5551212 in: PPP word: \qsecret
```

and in the `Devices` file:

```
T3000 tty2 57600 rtscts
```

3.5.2 *Express 2E*

Using a local terminal attached to `tty0` for the console, we assign an IP address to each Ethernet interface, using the default values for network mask and broadcast address as if the attached networks were not subnetted, and we apply a Filter to one of the Ethernet interfaces. Here are the commands we put in the `rc.boot` file:

```
console tty0
hostname thatone
version
ifconfig lo0 127.1
ifconfig enet0 38.2.72.3 filter Filter
ifconfig enet1 192.0.2.1
getty tty0 9600 nowait respawn
```

3.5.3 *Express PC*

Using a local terminal attached to `COM1/tty0` for the console, we assign an IP address to the Ethernet interface (using the default values `netmask 0xffffff00 broadcast 192.0.2.255`), set up outbound demand-dialed PPP using the Telebit T1600 modem on `tty1`. Here are the commands we put in the `rc.boot` file:

```
console tty0
hostname theother
version
ifconfig lo0 127.1
ifconfig ed0 192.0.2.1
pppd 192.0.2.1:192.0.2.2 auto idle 120
getty tty0 9600 nowait respawn
getty tty1 9600 respawn
```

and in the `Systems` file:

```
192.0.2.2 Any ACU 9600 5551212 in: PPP word: \qsecret
```

and in the `Devices` file:

```
T1600 tty1 9600 rtscts
```

3.6 Configuring Modems

Use the `cu` command to connect to an attached modem, and then send it appropriate initialization commands, making sure to save them into the modem's long term memory (if any). See the default `Dialers` file for recommended initialization strings for various modems.

If your modem isn't among those described in `Dialers`, you can make up your own configuration from these general recommendations.

3.6.1 Flow Control

If your modem system supports it, use out-of-band 'hardware' (RTS/CTS) flow control between your router and your modem, or between two routers over a null-modem cable. Specify `rtscts` either on the `pppd` command line, or in the Optional Parameters field of the line in `Devices`.

If, for whatever reason, you are forced to use in-band 'software' (XON/XOFF, ^Q/^S) flow control, then specify `xonxoff` on the `pppd` command line or in `Devices`.

`Pppd`'s default behavior is to use no flow control.

3.6.2 General Recommendations

You should connect your modem to your computer at the highest asynchronous serial speed that the modem can support (typically 38400 baud, but sometimes 57600 or 115200 baud), and allow the modem's internal carrier speed matching capability to make a usable connection with whatever type of modem is in use at the other end. This will allow you to take maximum advantage of the modem's data compression capabilities.

If your modem supports error correction, you should enable its use unless your tests show your application has better performance without it (this is unusual). This will allow line noise problems to be corrected at as low a level in the protocol stack as possible. If the modem has both MNP level 4 and CCITT V.42, choose the latter so that you can also use CCITT V.42bis data compression.

If your modem supports data compression, you should enable its use unless, again, your tests show your application has better performance without it. This will allow you to pass as much information across the PPP line as possible. If the modem has both MNP5 and CCITT V.42bis, choose the latter because it has a higher maximum compression ratio and because it behaves better with precompressed data streams.

The most flexible async PPP installation allows inbound and outbound connections over any available modem. To allow bidirectional use of the modem, configure the modem as follows:

- Answer after one ring. For many modems that use the AT command set, set `S0=1`.

- Lock the DTE interface to the selected speed. The modem will use flow control and buffering as needed to accommodate any carrier speed. Many modems don't have an explicit register for the speed, but simply store the value that was in use when the `&w` command was issued.

- Configure flow control appropriately. If your modem can use hardware (RTS/CTS) flow control, then use it. If not, configure the modem for XON/XOFF flow control and make sure both ends of the PPP connection have the 0x000A0000 bits turned on in their Async Control Character Maps (this is the default for Morning Star PPP). If you use hardware flow control, make sure that the cable carries the RTS and CTS signals. Regardless of which flow control scheme you choose, make sure the computer is configured to use the same type of flow control as the modem. This may mean using a special tty device to enable hardware flow control, or using the `rtscts` or `xonxoff`

option on the `pppd` command line or in the `Devices` file. A computer and modem using software flow control can talk to a computer and modem using hardware flow control, but both ends must have the asyncmap set to accommodate software flow control.

- Disable printing of result codes by the modem when processing an incoming call. On some modems this is only possible by also disabling result codes for outgoing calls. The dialer for such a modem must start with `"" ATE1` or its equivalent rather than the more common `"" AT`.
- Better dialer performance is possible if the modem is set up to print extended result codes such as `BUSY` when dialing out. This can be done in the `Dialers` file entry if necessary.
- Configure the modem to assert the Carrier Detect (CD, also DCD) signal only when carrier is established with another modem.
- Configure the modem to disconnect the phone call and restore the modem parameters to their saved values when the DTR (Data Terminal Ready) signal is deasserted by the computer.

If your async PPP connection travels over a dedicated analog leased line, you may want to amend the suggestions above:

- Always assert the Carrier Detect signal, regardless of whether a carrer has been established with the far modem.
- Configure the modem to ignore the Data Terminal Ready signal, and always attempt to connect with the far modem.

3.6.3 Example

A Telebit T3000 modem with default parameters is attached to `tty1` with a straight-through DB-25 cable. The `Dialers` file contains this information about the T3000:

```
# Set up a Telebit T3000 by giving it the following command:
#
#    at&f9 s0=1 s7=120 s48=1 s51=6 s58=2 s59=15 s61=0 s63=1 \
       tq2x12 &c1&d3&s1 &w
#
```

So we do that with `cu`, after killing any `getty` or `console` commands on that port, exiting `cu` by typing *carriage-return tilde period* (‘*return* `~ .`’).

```
# cu -l tty2 -s 57600 -h
Connected
at
OK
at&f9s0=1s7=120s48=1s51=7s58=2s59=15s61=0s63=1tq2x12&c1&d3&s1&w
~Closed connection.
#
```

(We set S51=7 for 57600 baud, rather than the suggested S51=6 for 38400, so that we can take advantage of the modem’s full speed capability.)

Complete the configuration by adding the appropriate entry to the `Devices` file, doing a `save`, and then rebooting:

```
# edit Devices
Devices: New file
edit> a
T3000 tty1 57600 rtscts
.
edit> w
Devices: wrote 1 line, 24 characters
edit> q
# save
save Devices? y
#
```

Chapter 4

Management

4.1 Paradigm

The Morning Star *Express* operating system resembles that of a UNIX-like environment, with files, processes, and interfaces. Files may be listed with `ls`, viewed with `cat`, and are stored in flash memory for persistence across reboots and power outages. Process status may be viewed with `ps`, and processes may be manipulated with `kill`.

The router's activity is largely controlled by the contents of its configuration files, which are ASCII text and may be altered with text editors, either with the `edit` command on the *Express* or with an editor on another host. After editing on another host, the file may be moved onto the *Express* using `ftp` from the host or using `tftp` on the *Express*. The working configuration file then remains in volatile RAM until the `save` command is used to replace the copy stored in flash.

A reconfiguration process might look like

1. Notice that changes are needed
2. View the configuration file with `cat`
3. Change the configuration file with `edit`
4. Write the configuration file to volatile RAM
5. Restart or use `kill` to signal the relevant daemon, as appropriate, such as `pppd`, `frd`, `gated`, or `inetd`.

6. Verify that the new behavior is correct
7. Store the volatile copy to flash with the `save` command

4.2 Configuration Files

Each of these files is described in detail later, but here is an overview of each configuration file and its purpose.

4.2.1 System

`passwd` Associates passwords and commands with login accounts

`rc.boot`
: Shell script that configures the router when it boots

`tz` Provides information about the local time zone

`tzposixrules`
: Provides time zone information for the `tz` command to use when interpreting POSIX time zone strings.

See also section A.2 on page 152.

4.2.2 IP Networking

`hosts` Associates hostnames with IP addresses if the router isn't using the Domain Name System

`inetd.conf`
: Defines which processes will be started in response to incoming UDP or TCP connection requests

`protocols`
: Defines IP protocols like ICMP, UDP, and TCP

`resolv.conf`
: Configures the router's use of the Domain Name System

`services`
: Associates protocol names with TCP and UDP connection ports

See also section A.4 on page 184.

4.2.3 PPP/SLIP

`Auth` Contains peers' secrets for link-level authentication by CHAP or PAP.

`Devices`
Associates device names with speeds, attached modem types, and other attributes

`Dialers`
Contains command strings for dialing various modem types

`Filter` Defines packet filtering behavior

`Keys` Contains DES encryption keys for secured network exchanges

`Login` When named in `passwd`, invokes `pppd` for an incoming dialup IP connection

`Systems`
Associates devices, dialing information, and other parameters with each SLIP or PPP peer

See also section A.3 on page 160.

4.2.4 Frame Relay

`Filter` Defines packet filtering behavior

`Keys` Contains DES encryption keys for secured network exchanges

See also section A.3 on page 160.

4.2.5 Routing Protocols

`gated.conf`
Defines the behavior of `gated`. You must send a running `gated` a SIGHUP with the command `kill -HUP` *pid* after editing this file to have it use the new configuration.

See also section A.4 on page 184.

4.2.6 Network Management

`acl.parties`
: SNMP access control list

`smp.parties`
: Access information for SMP entities

`snmpd.config`
: SNMP agent configuration

See also section A.5 on page 208.

4.3 Serial Line Encapsulation

4.3.1 SLIP

The Idle Timer

The idle timer, specified on the `pppd` command line, is a management tool for dial-up links only. It has the same usage described below under PPP.

4.3.2 PPP

The Idle Timer

The idle timer, specified on the `pppd` command line, is a management tool for dial-up links only. When the specified number of seconds have elapsed with no activity as specified in the `keepup` clause of the `Filter` file, the link is shut down.

Link Quality Monitoring

Link quality monitoring is used to ensure the quality of the line does not degrade below a specified point. By default, the minimum threshold is set at 1 LQM frame in 5, meaning the circuit will be dropped if more than 4 out of 5 LQM frames are lost.

4.3.3 Frame Relay

ARP options

The `arp` command can be used to add entries to the ARP cache for frame relay circuits. This is not necessary if the remote supports *Inverse ARP*. If the remote

does not support Inverse ARP, then you must add entries to the ARP cache for each *DLCI* in use. Your network provider can tell you which DLCI is connected to which IP address. You should place `arp` commands for frame relay circuits into the `rc.boot` file, either before or after starting the `frd` frame relay daemon.

4.4 Special Link Management Issues

This section addresses how to manage the *Express* router's connections in situations other than on-demand async dialups using the PPP protocol. We'll also discuss the finer points of configuring and using PPP to achieve the network results you need.

4.4.1 Active vs. Passive

`Pppd` can be used on either end of a connection, either as the calling system or as the answering system. `Pppd`'s default behavior is to immediately begin sending PPP messages as soon as the line is connected, anticipating that another PPP implementation will be waiting there, ready to start negotiating. When in *auto* mode, this will happen as soon as the `Systems` chat script completes. When not in `auto` mode, messages are sent immediately when the daemon starts.

This default behavior will work correctly in almost all situations. When calling another system with a dialup modem, however, it is sometimes better to let the answering end speak first, and there exist PPP implementations that get confused if they hear their peer speak first. In these situations, give `pppd` the `passive` option, and it will not try to start communicating until the other end does.

4.4.2 The Idle Timer

The `idle` option may be used for both the calling and the answering invocation of `pppd`. Whichever end specifies the shorter interval will shut down the line first. But how should idle timer values be selected, and should they ever not be used?

When `pppd` is talking to a system running PPP software that does not implement on-demand auto-dialing, it should not take the link down and expect the sessions to stay intact, as they would if the other end were a Morning Star *Express*, or a UNIX system running Morning Star PPP. These systems include those running most free PPP implementations, plus MS-DOS PCs running FTP Software's PC/TCP (see below for configuration hints).

Idle timer values should be selected based on the scarcity of resources and the cost of maintaining a connection. The calling system should set a relatively

brief idle timeout if the connection carries a monetary cost per minute – e.g., a long-distance telephone call. On the other hand, if the telephone billing scheme is based on the number of calls rather than their duration, the calling system would probably prefer to use a longer idle timer, or none at all.

The answering system should set a relatively brief timeout if it can accommodate a limited number of incoming connections because of e.g. too few modems to service all the incoming requests. The answering system might also use a shorter idle-timeout value if it were acting as a hub and providing its services via a toll-free WATS (800) number. On the other hand, if it were providing its services via a 976 or 900 number scheme, it may not want to specify any timeout interval at all, since each unit of the calling system's connect time would put funds in the answering system's pocket.

4.4.3 SLIP framing rather than PPP

If you must establish a link with a peer host that is unable to use the Point-to-Point Protocol, the Morning Star *Express* `pppd` is able to optionally use the non-standard but popular Serial Line Internet Protocol (SLIP). SLIP is really only a framing convention for arranging IP packets on a link, and it provides few of PPP's advanced facilities. By default, `pppd` running with the `slip` option does not use RFC 1144 'VJ' TCP header compression, although it will start sending VJ-compressed packets if it first receives VJ-compressed messages from its peer. Specifying `vjcomp` will cause `pppd` to always use header compression with the default 16 slots (useful for talking to 'CSLIP'), or header compression can be completely disabled with the `novjcomp` option. SLIP's default MRU and MTU (maximum receive unit and maximum transmission unit) are 1006. Automatic dialing, idle line hangup, packet filtering, and most other management facilities are all available for use with SLIP.

When `pppd` is invoked with the `slip` option on its command line, it performs no LCP, PAP, CHAP, or IPCP negotiation. `pppd` can provide no Link Quality Monitoring services when running with SLIP framing. SLIP does not support the Asynchronous Control Character Map feature or the `escape` option, and therefore can only be used over connections that are completely transparent to all 8-bit character values. You must either use hardware flow control (specify `rtscts`) or no flow control at all. SLIP can be used only on asynchronous links that provide a clear 8-bit data path and don't interpret the passing data as flow control.

You can use `pppd` in its SLIP framing mode to dial into a terminal server made by cisco Systems. Since the SLIP protocol supports no option negotiations, cisco terminal servers print the incoming system's IP address in text on the serial line, before beginning the SLIP protocol. `pppd` can parse this text by use of the '

A' token in the 'expect' phase of the Systems file chat script. The daemon will then assign the assigned address to the *Express* end of the point-to-point networking interface. See Systems.ex for an example use of this facility.

4.4.4 Synchronous PPP

The *Express* router (except for the *Express PC*) can run PPP in synchronous mode over any of its serial ports.

For `pppd` to connect at T1 speeds to another host or router using port `tty0` use a `Systems` entry like this:

```
peeradddr Any;5 tty0 1536000 0
```

The `Devices` entry looks like this:

```
Direct  tty0 1536000 sync
```

Invoke `pppd` in `rc.boot` like this:

```
pppd hostaddr:peeraddr auto dedicated
```

You may want to provide automatic failover (to dialup facilities) for your synchronous or dedicated asynchronous connection, so that user services will continue even if the dedicated line becomes unavailable. Put a second entry in `Systems` referencing a dialup modem (`Any ACU 57600`), after the entry for the dedicated link. `Pppd` will by default ask the peer to send Link-Quality-Report messages. If the dedicated line fails, `pppd` will either stop receiving the reports or will note that CD has dropped, and will terminate the connection when enough reports have been lost to drive the measured link quality below the configured threshold (see the description of LQM in section 4.4.5 on page 48 for an explanation of how to set the threshold and message interval). After unsuccessfully attempting to reestablish the connection on the same line, `pppd` will automatically fail over to the second entry in `Systems`, using the modem to dial up and reestablish IP traffic. Users will notice about 30 seconds of interruption, and they'll notice that the link is slower once it's re-established, but the link will at least still be available. When the dedicated connection is restored, manually cause the dialup modem to hangup the line, and `pppd` will attempt to reconnect using the next entry in `Systems`, or wrap around to the first entry in the file, which describes the dedicated connection.

4.4.5 Link Quality Monitoring

Link Quality Monitoring is an optional facility defined within the PPP protocol specification, and provided by some PPP implementations, which allows PPP to make policy decisions based the observed quality of the link between peers.

LQM is particularly useful in the detection and appropriate response to total link failures, providing users with failover protection, as described in the section above. Most such total failures (e.g. a backhoe digging through the telephone company's cable) result in the communications equipment (CSU/DSU or modem) deasserting the Carrier Detect signal, which `pppd` observes as a hangup event. But some failure modes (e.g. misconfigured flow control, or over-reliance on in-band XON/XOFF flow control) can leave the modems connected while the PPP peers are unable to communicate. In this situation, the peers will observe a LQM failure and take appropriate action, usually a disconnection and redial.

Another use for LQM might be if an intercontinental telephone call is using a circuit of such poor quality that significant numbers of packets are being damaged in transit. The PPP implementations on each end can decide, based on LQM measurements, to hang up and redial in hopes of getting a better connection. This is a very rare occurrence if your modems provide V.42 or MNP4 error correction, but it does occasionally happen.

Whatever the cause, the disconnection and redial operation can happen without user intervention or application awareness, because even if PPP frames are damaged or lost, the upper protocol layers will arrange for retransmission as needed to provide the user with a complete data stream. The user will simply experience a pause while the modems reestablish the connection.

LQM works by asking the peer to send periodic status packets. `pppd` will request that the peer send Link-Quality-Report packets as frequently as specified in the `lqrinterval` command line argument, or every ten seconds if `lqrinterval` is not specified. If the link goes down or is degraded, many or all LQR packets (along with user data packets) will be lost. `pppd` counts the arriving LQRs, and if too many have been lost, `pppd` will close the connection. The `lqthreshold` argument specifies the minimum acceptable link quality, which defaults to requiring at least one LQR out of every five to successfully make it through. So (with the default values) unless at least one LQR has arrived from the peer in the past minute (counting timing slop), `pppd` will shut down the link.

The default LQM behavior is fairly permissive. If you want `pppd` to be more demanding about line quality, specify a higher value for `lqthreshold`. For example, to demand that no more than one LQR be dropped per minute, use 'lqthreshold 5/6'. This way, if two LQRs in a row are dropped, the line will be

shut down. `pppd` will discover line failures more quickly (at the expense of greater monitoring traffic) by decreasing the `lqrinterval`, but remember to also consider the `lqthreshold`. For example, you can demand that at least one LQR arrive per minute by specifying 'lqrinterval 5 lqthreshold 1/12', and you can demand that no more than one LQR be lost each minute by specifying 'lqrinterval 5 lqthreshold 11/12'.

If, during LCP option negotiation, the peer refuses to send Link-Quality-Reports, `pppd` will instead begin sending LCP Echo-Request messages at the requested `lqrinterval` and use the arriving LCP Echo-Response messages to make the link quality decision. If the peer doesn't correctly use a Protocol-Reject message to tell the *Express* router's `pppd` to switch to LCP Echo-Requests, `pppd` can be given either the `echolqm` argument, to dispense with the discovery phase and begin directly with Echo-Requests, or the `nolqm` argument, to disable link quality monitoring completely.

At `pppd`'s debugging verbosity level 4, the log file will receive summary messages like this:

```
5/7-13:45:27-1514 LQM: Pkt: 1/1 Oct: 53/53 LQRs: 5/5
```

This means that, during the last testing interval, this system transmitted one packet and received one packet. 53 octets crossed the link in each direction. And this system has received responses to all five of the most recent five Link Quality Reports it sent.

The LQR packet is 36 octets long, and the default `lqrinterval` of ten seconds will cause the additional traffic to be unnoticed on most connections. However, if the application is very sensitive to speed and requires absolutely every bit of available line bandwidth, Link Quality Monitoring can be disabled with the `nolqm` argument.

Link Quality Monitoring packets, since they are part of the Point-to-Point Protocol rather than user data, do not count against the idle line disconnection timer.

4.4.6 Dedicated Lines

Some PPP installations use asynchronous serial connections that are always available, often using high-speed asynchronous short-haul modems over a building or campus wiring plant. In this situation, it makes sense to keep the PPP connection up at all times, and reestablish connectivity as soon as possible if one end goes down. For this purpose, use the `dedicated` argument on the `pppd` command

line. `dedicated` instructs `pppd` to never give up on the connection; that is, if the peer tells `pppd` to disconnect, it will continuously attempt to reconnect. Fatal disconnects, though (LQM failures, loss of Carrier Detect), will still cause `pppd` to close the device, go back to the `Systems` file looking for another matching entry, and then go through the call retry delay if none are available.

Normally, on a dedicated line, no `getty` or `sgetty` process is run on the device and both ends of the circuit actively try to connect to their peer. Each machine's `rc.boot` script should contain a line like

```
pppd local:remote auto dedicated
```

The `Systems` file should specify the device name (such as `tty1`) in the `device` field (rather than `ACU`), and the `Devices` file should contain a line like

```
Direct cua 38400 rtscts
```

The Dialers file is ignored when `Direct` is found in the `dialer` field.

See the discussion above regarding line failovers and using an auto-dial modem as a backup link.

4.4.7 Constantly-Open Telephone Calls

Some PPP installations choose to have their connections up at all times, rather than depending upon `pppd`'s on-demand dialing capability to reestablish a formerly-idle link when traffic warrants. This is different from a `dedicated` line because a modem must be dialed or a login negotiated before PPP frames can be exchanged. In this situation, use the `up` argument on `pppd`'s command line. `up` instructs `pppd` to make every effort to keep the connection up. For example, when the connection goes down, `pppd` will immediately redial the modem, rather than waiting for traffic demand. (It would not be sensible to use the `up` argument along with the `idle` argument.)

4.4.8 Switched Synchronous PPP Connections

The *Express* router (except the *Express PC*) is useful for carrying IP traffic over switched data networks, such as ISDN or Switched-56K services. The *Express* can serve on either the initiating (outbound) or the answering (inbound) end of the call.

Outbound on-demand

The *Express* router can initiate PPP connections on demand through a synchronous communications channel over a switched data circuit (e.g. an ISDN Terminal Adaptor or a Switched-56K synchronous modem/CSU/DSU). Here's how:

rc.boot pppd my-addr:peer-addr auto idle 10

We invoke `pppd` in its "auto-call" mode, ready to dial `peer-addr` when a packet needs to cross the link. We specify a very brief *idle* timer, because switched digital networks typically provide very cheap and quick call-establishment, but they bill users by the number of seconds they're connected. An *idle* timer of 5 or 10 seconds ensures that a typist's connection won't be terminated between keystrokes, while minimizing the billed idle connection time. Experiment with different *idle* timers until you find one that suits the needs of your applications and your budget.

Systems peer-addr Any tty0 64000

In this example, a synchronous DCE (ISDN TA or modem or CSU/DSU) is attached to the V.35 interface on tty0. This example also assumes that no login "chat script" process will be required after the circuit is connected – the peer will immediately begin LCP negotiations.

Devices Direct tty0 64000 sync

In this example, a synchronous DCE (ISDN TA or modem or CSU/DSU) is attached to the V.35 interface on tty0. This example also assumes that the DCE will provide the synchronous clocking signal, which is almost always true.

Since the *Express router* cannot execute a chat script over a synchronous connection, the DCE (ISDN TA or modem or CSU/DSU) must perform the dialing task. Pre-configure the DCE with the "phone number" it must "dial" to reach the link peer. Then when `pppd` needs to initiate a connection, it will raise the DTR signal. The DCE should initiate the switched circuit connection when it observes the rise in DTR, and it should hang up when DTR drops.

This DTR-dialing technique limits the *Express* router to one demand-dial peer destination, since only one phone number can be configured for use when DTR rises. Still, this is sufficient for most needs. Future versions of the *Express* software may have synchronous chat script processing, in which case the `Dialers` file can contain a V.25bis dialing command string, and the DCE can be instructed to dial an unlimited number of peers on demand.

Inbound

The *Express* router can answer incoming intermittent connections that expect to use the PPP protocol.

rc.boot sgetty tty0 respawn cdwait command pppd my-addr: idle 10 requirechap tty0 sync nodetach

Bidirectional On-demand

Bidirectional on-demand switched synchronous PPP connections require a combination of the ideas from sections 4.4.8 and 4.4.8 above.

rc.boot pppd my-addr:peer-addr auto idle 10
sgetty tty0 respawn cdwait command pppd my-addr: idle 10 requirechap tty0 sync nodetach

Systems peer-addr Any tty0 64000

Devices Direct tty0 64000 sync

4.5 Security Techniques

In this section, we will discuss facilities and techniques that can lead to greater security within networks that are connected via Morning Star *Express* routers, or UNIX systems running Morning Star PPP. When external network connectivity is desired, but it's impractical to impose thorough security practices on each internal host, these techniques can offer some peace of mind.

In most cases, multiple security techniques can be used to manage a single connection. For example, a host might dial its peer only during certain hours, the peer's modem might dial back to the originator, the peer might also employ SecureID to restrict logins, the PPPs might authenticate each other using CHAP, and their packet filters might still only allow certain types of traffic to cross the link.

4.5.1 Time-To-Call Restrictions

The second field on each line in the `Systems` file specifies what times of day `pppd` is allowed to attempt to establish a connection. This can be used to control telephone charges, or to ensure that connections are attempted only when personnel are

present on each end to monitor the link. See `Systems` in section A.3.2 on page 161 for details.

4.5.2 Dial-Back

An answering modem set up for dial-back applications will typically challenge incoming callers with a prompt string, accept the input which probably identifies the caller, then hangup the call. It will then call some preconfigured number, usually associated with the caller's identification, and re-establish a carrier. The original answering modem might then issue another prompt and demand the same identification before allowing data to flow to the system connected to its serial interface.

The originating system's `pppd` must be prepared for the brief period of disconnection, and the temporary lack of a Carrier Detect signal from its modem. To avoid receiving a SIGHUP, `pppd` must instruct the *Express* router's serial port not to deliver the signal - to temporarily treat the serial interface as if it were connected to a local device like a terminal or printer, instead of a modem. `pppd` does this by specifying `\M` in the 'send' phase of a `Systems` chat script. After the disconnection period, `pppd` must instruct the system's serial drivers to respect the modem's full variety of control signals, by specifying `\m` in the 'send' phase of a `Systems` chat script.

For example, to dial into a system protected by a dial-back modem, the `Systems` chat script might look like

```
#
# This connects to a system that's protected by a
# callback modem (in this case, a Telebit T3000 with S46=2).
#
server Any ACU 38400 19071234567 TIMEOUT 60 \\
ENTER\\sPASSWORD: my_modem_password\\M \\
ENTER\\SPASSWORD: my_modem_password\\m \\
ogin: my_login_name ssword: my_login_password
```

See `Systems` in section A.3.2 on page 161 for details.

4.5.3 Packet Filtering

The *Express* router can selectively pass or block any packet to or from a serial link (SLIP, PPP, or Frame Relay) or Ethernet interface based on a wide variety of criteria:

- Source or Destination host or network, by name or address
- Session originated locally or remotely
- Inbound or outbound packet
- Protocol (e-mail, interactive terminal, file transfer, etc.)
- IP options (source route, record route, etc.)

When a packet is blocked, the *Express* can respond to the sender with an appropriate ICMP Destination Unreachable message (e.g. Host Unreachable, Source Route Failed, Network Access Administratively Prohibited).

The administrator may record the passage of any packet, but it is particularly easy and helpful to log those that attempted but failed to breach the firewall.

Be careful about the use of host names in the `Filter` file. If you use the Domain Name Service (see `resolv.conf` in section A.2.7 on page 155), remember that no name servers will likely be accessible during the router's boot sequence because the router itself probably provides the link by which it must reach those name servers. DNS users must specify hosts and networks in `Filter` by their literal IP addresses. However, if there's no `resolv.conf` in place and the *Express* relies upon its `hosts` file instead, host names will work as expected.

Here's an example complex Filter configuration, appropriate for a system used as a router between a private network (192.0.2.0) and the Internet, interspersed with commentary about its functions:

```
default
```

The 'default' filter's specs will be applied to any packet crossing the point-to-point link connecting this host to the peer.

```
bringup !ntp !3/icmp !5/icmp !11/icmp !who !route !nntp
```

If the link is configured for dialup connections on demand, the 'bringup' clause describes those packets that will cause a call to be placed and a connection initiated. In this case, it's simpler to name the types of 'nuisance' packets that should not be allowed to bring up the link. The Network Time Protocol, the Network News Transport Protocol, broadcasts from `rwhod` and `routed`, and various ICMP messages (pings and error codes) are put in the list of traffic types that aren't interesting enough to dial the modem. Any other sort of traffic not named here will initiate a dial connection.

```
pass !recv/ip-opt=srcrt/unreach=srcfail
```

Don't allow any incoming packets with the Source Route option set in the IP header. Respond with an ICMP Destination Unreachable message with the Source Route Failed code value.

```
!192.0.2.0/recv/src/unreach=net
!192.0.2.0/send/dst/unreach=net
```

Block any incoming packets that claim to be from our net, and block any outgoing packets that claim to be destined for our net. Respond with an ICMP Destination Unreachable message with the Bad Net code value.

```
nntp/128.146.110.50 nntp/128.146.8.50 nntp/150.252.1.1
nntp/131.236.20.19 nntp/129.22.8.64 nntp/128.146.111.36
!nntp/unreach=host
```

Allow Network News (Usenet) exchanges with only our known news neighbors. Block any other NNTP traffic, and respond with an ICMP Destination Unreachable message with the Bad Host code value.

```
domain/tcp/syn/recv/dst domain/tcp/syn/send/dst
```

Allow DNS secondary dumps in both directions, but only if our end of the stream is really being handled by our domain name server.

```
smtp/syn/recv/dst smtp/syn/send/dst
```

Allow SMTP (electronic mail) in both directions, but only if our end of the stream is really being handled by our mail server.

```
ftp/syn/recv/dst/192.0.2.7 !ftp/syn/recv/unreach=host
```

Allow incoming FTP (file transfer) traffic that uses our anonymous-FTP server system, but block any other incoming FTP requests. Respond with an ICMP Destination Unreachable message with the Bad Host code value.

```
!6000/tcp/syn/send/unreach=host
```

Allow no X11 clients running on hosts within our network to display on X11 servers elsewhere. Respond with an ICMP Destination Unreachable message with the Bad Host code value.

```
1024-65535/tcp/dst/recv
```

```
!0-1023/tcp/src/syn/recv/unreach=host
```

Incoming TCP connections to ports numbered above 1023 are OK, because those are "non-privileged" ports. Above this line, we should list only services where a daemon sends traffic from a privileged port.

```
!domain/tcp/syn domain
```

Allow no DNS zone transfer traffic except that named above (connecting with our own domain name server), and allow any other DNS traffic to pass.

```
!smtp/syn smtp
```

Allow no SMTP traffic except that handled by our mail server.

```
!ident/dst/recv/unreach=host
!ident/src/send/unreach=host
```

We don't use the RFC 1413 identification services here, so we might as well bounce the queries at the gateway instead of having inetd refuse the connection.

```
ftp ftp-data
```

After blocking the traffic specified above, allow both FTP command streams and FTP data streams to cross the link, both inbound and outbound.

```
www/syn/recv/192.0.2.11/dst
!www/syn/recv/unreach=host www
```

Allow incoming World Wide Web traffic to reach our WWW server, allow no other incoming WWW traffic, and allow ourselves unlimited outbound WWW access.

```
!login/syn/recv/unreach=host
!shell/syn/recv/unreach=host
!supdup/unreach=host
!exec/unreach=host
!uucp/unreach=host
!chargen/unreach=host
```

Block incoming rlogin, rsh, and supdup connections. Block access to our system's rexecd. Since we do no UUCP over TCP, block any attempts to appear to be UUCP traffic. Block access to our "character generator" port because that's only a testing facility and could be used by someone outside to swamp our link's bandwidth with unwanted but otherwise harmless traffic.

```
!finger/syn/recv/unreach=host finger
whois login shell gopher daytime
```

Block incoming finger probes until we install a safe finger daemon. Allow several other sorts of traffic that we know to be safe, within the constraints of the previous filters.

```
33410-33515/udp talk ntalk ntp
!1-65535/udp/unreach=port
```

The `traceroute` tool probes high-numbered UDP ports, and that's so useful that we should let it through. We allow both old and new versions of the `talk` command, and we use the Network Time Protocol to keep our clocks in sync.

```
!icmp/192.0.2.198 icmp
```

Don't allow anyone to ping a particular testing host here, but allow other ICMP messages to pass freely.

```
telnet/syn/recv/128.146.8.0/255.255.255.0
!telnet/syn/recv/unreach=host telnet
```

Allow incoming telnet connections from hosts on a subnet we know to be secure, allow no other inbound telnet connections, and allow outbound telnet connections from our network to anywhere else.

```
!all/unreach=host
```

Allow no other traffic besides that which we've explicitly specified as allowed through the firewall, and respond with an ICMP Destination Unreachable (Bad Host) message.

```
keepup !send !ntp !3/icmp !5/icmp !11/icmp !who !route
```

The link will be considered active (non-idle) if any packet passes that's not specified on this line as being blocked. Since there are certain link failure modes that will allow our system to continue sending even though the peer is unresponsive, no outbound traffic counts against the idle timer.

```
log !3/icmp rejected
```

Log any packet blocked by the 'pass' filter above, except ICMP Destination Unreachable messages.

4.5.4 Link-Level Authentication

When using PPP encapsulation, the *Express* implements both the Password Authentication Protocol (PAP) and the Challenge Handshake Authentication Protocol (CHAP). If `pppd` is invoked with either the `requireauth` or the `requirechap` option, it will demand that the peer (either calling or called) authenticate itself.

The `Auth` file contains pairs of either *names* and *secrets* for CHAP negotiation, or *usernames* and *passwords* for PAP negotiation. If a peer provides a name or username, its secret or password must match that found in `Auth` or the authentication phase fails and the connection is terminated.

Each *name/secret* pair in `Auth` may be followed by address patterns restricting the peer's negotiated IP address. If an address restriction is specified for a particular *name* and the peer's negotiated IP address does not match the restriction address patterns, `pppd` will terminate the connection.

4.5.5 Tunneling

The *Express* router's `pppd` can run the PPP protocol over async serial terminal ports, over the SnapLink's high-speed synchronous serial ports, or over TCP streams between two systems that are already connected by an existing IP network. The advantage of using PPP to tunnel a virtual network through an existing network lies in the ability to maintain the inaccessibility of hosts on that virtual network: if the virtual network uses IP addresses that are not known to the rest of the real Internet, hosts on that network are inaccessible except from other hosts on the virtual network.

Tunneling PPP Over Telnet

Suppose an *Express* router named frobozz has lines in `services` like

```
ppp 57/tcp
ppptelnet 59/tcp
```

and lines in `inetd.conf` like

```
ppp stream tcp nowait root pppd pppd nogob:
ppptelnet stream tcp nowait root pppd pppd bogon: telnet
```

Then some other host (with similar lines in `services`) could invoke `pppd` as

```
pppd blat:bogon auto
```

with a `Systems` line like

```
bogon Any telnet/frobozz/ppptelnet
```

This would connect the two UNIX systems via a PPP connection running over a TCP stream through an existing IP network, and the TCP stream would use telnet-style option negotiations for 8-bit transparency and telnet escape processing.

Alternatively, suppose some remote system dials into a terminal server that can only connect with frobozz via a telnet session, but it has the option of connecting to a port other than the usual 23. By invoking the 'answering' `pppd` from `inetd.conf`, rather than via the normal telnetd/login/passwd/Login/pppd mechanism, the overhead of `telnetd` and its pty processing can be avoided. The presence of `telnet` on the answering `pppd`'s command line causes that `pppd` to negotiate the telnet binary option, and understand telnet escape processing - the stuff that's normally done for you by telnetd.

Tunneling PPP over TCP

As a more general example of tunneling PPP over TCP through an existing IP network, some other host (with similar lines in /etc/services) could invoke `pppd` as

```
pppd snorf:nogob auto
```

with a `Systems` line like

```
nogob Any tcp/frobozz/ppp
```

All these `pppd` invocations can be seasoned to taste with appropriate options, like *asyncmap 0x0 nolqm netmask 0xffffff00 requirechap*, or whatever else is needed. If you're invoking `pppd` from `inetd.conf`, `requirechap` is a very good idea.

For more discussion of the idea of tunneling and virtual networks, please see RFC 1241 and research.att.com:dist/smb/pnet.ext.ps.Z or ftp.MorningStar.Com:pub/papers/pnet.ext.ps.Z .

4.5.6 Selective Gateway Encryption

Encryption is not available in products exported from the USA, except in special circumstances.

If `pppd` or `frd` is invoked with the `gw-crypt` option, the next word on the command line will be used as the name of a file containing keys to be used for DES

encryption and decryption. When `pppd` or `frd` is about to transmit a packet, it compares the packet's source and destination addresses with the endpoint values specified on each line of the key file. If the packet's addresses match the endpoints in a line in the keys file, the packet's contents are passed through a DES encrypter, using the key specified on the keys file line. When `pppd` or `frd` receives a packet with IP addresses matching an entry in the key file, the associated key is used to decrypt the packet.

In the keys file, the endpoint addresses are specified as either names or literal (numeric) addresses. The addresses may be individual hosts, entire networks, or to subnets with network masks specified after a delimiting slash ('/'). This way, all traffic between particular LANs can be encrypted during its traversal of an intervening insecure internet, but traffic between hosts on those LANs and hosts on any other connected LAN will be transmitted in the clear.

The *Express* router's packet encryption leaves the packet's IP header unchanged, so the packet can be routed through an intervening IP internet just as it would without encryption. The operation encrypts the TCP, UDP, or ICMP header, along with the user data segment of the packet. Snoopers will be able to count packets passing between the gateway encrypter endpoints, but will be unable to observe their content type, since other information (such as port numbers or the TCP SYN bit) is contained in the TCP header.

4.6 Monitoring Connection Progress

`Pppd` and `frd` record their state in two places: the argument vector and via the `syslog` facility.

4.6.1 The Argument Vector

As `pppd` or `frd` runs, it updates its argument vector. You can see its progress by typing `ps`. The connection status is described by several comma-separated fields inside a pair of brackets ('[' and ']'). Only the first field will always appear, but when other information is available, the fields appear in the following order:

- The state of the connection: **dialing**, **connecting**, **disconnecting**, **up**, or **off**, optionally including a description of the reason for a connection failure.
- The name of the network interface used by this connection, such as `du0`. (`pppd` only, in auto mode only)

- The name of the serial interface device in use (e.g. `tty1`).
- The Internet address of the peer. (`pppd` only)
- The current reception and transmission rates, separated by a slash, in bits per second.
- The current percentages of transmitted data successfully received at each end of the link. The two rates, separated by a slash, are the percentage of data our peer transmitted that we received and the percentage of data we transmitted that was actually received by our peer. This field will only be displayed if both ends of the link agree to do LQM and some loss has occurred recently. (`pppd` only)
- The idle timer value: the number of seconds that have passed since the link last carried a packet described in the `keepup` line. (`pppd` only)

The fields following the trailing ']' are the actual arguments with which `pppd` or `frd` was invoked.

4.6.2 Syslog

`Pppd` and `frd` send information to the system log that also contains reports about the daemon's progress. Higher debugging levels provide more detailed logging output. At extremely high debugging levels such as 8 or 9 (for `pppd`) or 5 (for `frd`), the log file will contain a record of each frame or even each byte sent or received. Such extreme detail should be avoided for normal operations, since the log file can quickly consume prodigious amounts of disk space, and since the very act of logging can occasionally cause protocol timeout problems that only Heisenberg could solve. Lower debugging levels can show progress through PPP's finite state machine, or allow observation of `pppd`'s progress through the chat scripts in `Systems` and `Dialers`.

Each line in the log begins with a date and time stamp, followed by the process ID of the daemon, followed a description of the event that caused the log message.

Status information is sent to the system log by each running `pppd` and `frd`. Each line printed consists of a message preceded by the date, the time, and the process ID number of the daemon writing the message. The quantity and verbosity of messages are controlled with the `debug` option and with the `log` filter (see `Filter` in section A.3.5 on page 172).

Each packet that brings up a PPP or SLIP link (at debug level 1 or more), each packet that matches the `log` filter (at any debug level), or any packet when

the debug level is 6 or more (for `pppd`) or 4 or more (for `frd`) sends a one-line description of the packet to the system log. The first item of the message is the protocol (tcp, udp, icmp, or a numeric protocol value). For ICMP packets, the keyword `icmp` is followed by the ICMP message type and sub code, separated by slashes. After the protocol comes an IP address and optionally a TCP or UDP port number, followed by an arrow indicating whether the packet was sent (`->`) or received (`<-`), followed by another address and port number, followed by the length of the packet in bytes before VJ TCP header compression, followed by zero or more keywords. For transmitted packets, the first IP address is the source address, while for received packets, the first IP address is the destination address. Well known TCP and UCP port numbers will be replaced by the equivalent name from the `services` file. The keywords and their meanings are:

`frag` the packet is a middle or later part of a fragmented IP frame

`syn` the packet has the TCP SYN bit set and the ACK bit off

`fin` the packet has the TCP FIN bit set

`bringup`
: the transmitted packet matches the bringup filter and is bringing up the link

`!keepup`
: the packet has been rejected by the keepup filter

`!pass` the packet has been rejected by the pass filter

`dial failed`
: the packet was dropped because `pppd` is waiting for the call retry timer to expire

`(c)` the received packet is VJ TCP header compressed

`(u)` the received packet is VJ TCP header uncompressed

`(e)` the sent or received packet was encrypted or decrypted

For example, the following system log message

```
9/6-14:06:26-17 tcp 63.1.6.3/1050 -> 17.9.1.9/ftp 44 syn
```

indicates that at 2:06:26 PM on September 6, process ID 17 sent a 44-byte TCP packet with the SYN bit set from port 1050 on 63.1.6.3 to the FTP port on 17.9.1.9.

Chapter 5

Techniques, Hints and Tips

5.1 Routing

In contrast to IP traffic over a broadcast network medium, PPP and SLIP links appear as point-to-point network connections between two known addresses. If neither endpoint resides on an IP-based local area network (LAN), packets will simply flow in both directions as soon as the PPP connection is established. When the PPP link connects a remote host to a LAN, or provides a connection between two LANs, or connects a LAN to the worldwide Internet, the systems involved must be concerned with routing.

In the examples below, we'll describe how to establish static routes when the machines boot. This can be cumbersome, because any topology change requires that all the machines even remotely involved be reconfigured by editing their boot-time command scripts. An alternative to static routes would be to use some automated system for updating all the hosts' routing tables. This is usually implemented as a daemon running on each host. Many networks use the RIP[1] protocol and run `routed` or `in.routed` on their UNIX systems. If your network's UNIX hosts and other routers use RIP and `routed` to propagate routing information, the *Express* router's `gated` must be configured for RIP as well.

Some sites prefer other routing protocols such as OSPF, and they use the Gate Daemon[2], the same software included in the *Express* router. If you use `gated` on

[1] Hedrick, C.L., 'Routing Information Protocol', RFC 1058, Rutgers, June 1988

[2] gated.cornell.edu:pub/gated/gated-R3_0_1.tar.Z, also available via anonymous PPP/FTP from Morning Star Technologies.

the *Express*, you should invoke `pppd` with the `netmask` argument set to the same value as is used on the LAN, if that subnet mask is different from the default for the class of your network.

5.1.1 Connecting a Host to a LAN

There are two conventional approaches for connecting a set of standalone hosts to a LAN via PPP.

Separate Network

The method that will likely cause the least confusion is to assign a distinct network or subnet for use by all the remote machines. For example, suppose the organization's class B network number is 134.19, and it is subnetted with a class C-sized network mask of 0xffffff00. The departmental LAN is subnet 134.19.5.0 with broadcast address 134.19.5.255, and it is populated by an *Express* named *alpha* (134.19.5.22), and a system named *bravo* (134.19.5.41). A modem is attached to one of *alpha*'s serial ports, and *alpha*'s `Login` shell script is the generic one described below. A laptop SPARCstation clone named *nomad* wishes to connect to the network by dialing into the *Express*.

Please see Figure 5.1 on page 65 for a diagram of a network configured this way.

Designate subnet 6 for remote machines. Assign *nomad* the IP address 134.19.6.17. The Morning Star PPP daemon on *nomad* would be invoked in the `/etc/ppp/Startup` script (called from `/etc/rc.local`) as

```
pppd 134.19.6.17:134.19.5.22 auto idle 180
```

or, if all the names can be found in the local `/etc/hosts` (or resolved via the Domain Name System without first needing to bring up the PPP link), it could look like

```
pppd nomad:alpha auto idle 180
```

The next line in *nomad*'s `/etc/ppp/Startup` would point an IP route through the dial-in gateway:

```
route add -net 134.19.0.0 134.19.5.22 1
```

Alternatively, and necessarily if the LAN were also connected to the Internet,

```
route add default alpha 1
```

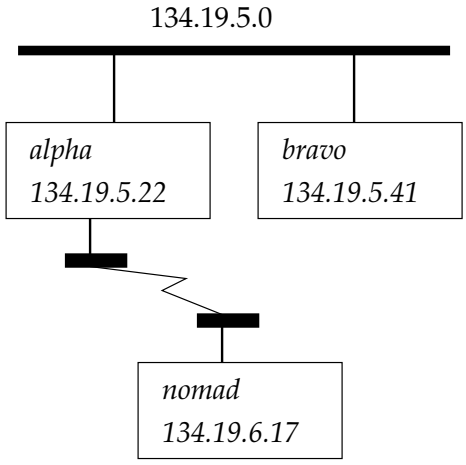

Figure 5.1: Separate Network

Similarly, the `Login` script on *alpha* would look like

```
pppd alpha:nomad auto idle 180
```

and *bravo*'s `/etc/rc.local` would contain a line like

```
route add -net 134.19.6.0 alpha 1
```

ARP table manipulation

If a separate subnet number is unavailable for use by remote-access machines, it is possible to assign them addresses on the same subnet number as the departmental LAN. This is accomplished by having the router answer ARP[3] requests for the remote host's IP address with its own LAN MAC address.

As above, suppose the organization's class B network number is 134.19, and they are subnetting with a class C-sized network mask of 0xffffff00. The departmental LAN is subnet 134.19.5.0, populated by an *Express* named *alpha* (134.19.5.22) with Ethernet address 0:3:c6:9:f1:27 and a workstation named *bravo* (134.19.5.41). A modem is attached to one of *alpha*'s serial ports, and *alpha*'s `Login` shell script

[3] Plummer, D.C., 'Ethernet Address Resolution Protocol', RFC 826, Symbolics, November 1982.

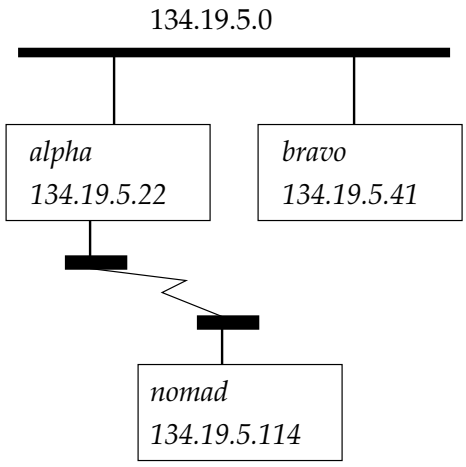

Figure 5.2: ARP Table Manipulation

is the generic one described above. A laptop SPARCstation clone named *nomad* wishes to connect to the network (figure 5.2).

Assign *nomad* the IP address 134.19.5.114. The Morning Star PPP daemon on *nomad* would be started in the `/etc/ppp/Startup` script (called from `/etc/rc.local`) as

```
pppd 134.19.5.114:134.19.5.22 auto idle 180
```

or, if all the names can be found in the local `/etc/hosts`,

```
pppd nomad:alpha auto idle 180
```

The next line in *nomad*'s `/etc/ppp/Startup` would point an IP route through the dial-in gateway:

```
route add -net 134.19.0.0 134.19.5.22 1
```

Alternatively, and necessarily if the LAN were also connected to the Internet,

```
route add default alpha 1
```

Similarly, the PPP daemon on *alpha* would be started in `Login` as

```
pppd alpha:nomad auto idle 180
```

Alpha's `rc.boot` file would contain a line like

```
arp -s nomad 0:3:c6:9:f1:27 pub
```

This would add a permanent entry to *alpha*'s ARP table and cause it to be provided to other systems on the local Ethernet upon request. Any time a host on that LAN tries to find the Ethernet address corresponding to *nomad*'s IP address, the request will be answered with instructions to forward the packets to *alpha*. Since *nomad* appears to be directly connected to the LAN, no hosts on the LAN, nor on any other IP-connected network, would require routing table modifications.

5.1.2 Connecting two LANs

Suppose an *Express* router named *exp-a* (192.0.2.63) and a workstation named *ws-a* (192.0.2.66) are on one LAN, with *exp-a* connected to a modem or CSU/DSU. Also suppose *exp-b* (134.19.14.29) and *ws-b* (134.19.14.102) are on another LAN across town, with *exp-b* connected to a modem or CSU/DSU (figure 5.3).

The PPP daemon on *exp-b* should be started as

```
pppd exp-b:exp-a auto idle 130
```

and *exp-b* should have a route like

```
route add -net 192.0.2.0 exp-a
```

and *ws-b* should have a route like

```
route add -net 192.0.2.0 exp-b 1
```

Similarly, *exp-a*'s PPP daemon should be started as

```
pppd exp-a:exp-b auto idle 130
```

and *exp-a* should have a route like

```
route add -net 134.19.0.0 exp-b
```

and *ws-a* should have a route like

```
route add -net 134.19.0.0 exp-a 1
```

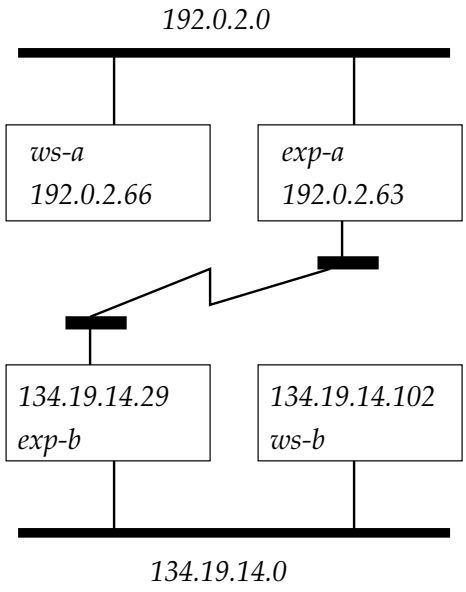

Figure 5.3: Connecting Two LANs

5.1.3 Connecting a LAN to the Internet

If your LAN is connected to the Internet, or if you have arranged an account at a Point Of Presence (POP) of a PPP- or SLIP-talking Internet connectivity vendor (say, foo.net), then you should arrange for your default route to point through the gateway at the other end of the PPP connection. If an *Express* router named *frobozz* were connected to a modem with which it were to call a POP, it would start its PPP daemon like

```
pppd frobozz:pop.foo.net auto idle 240
```

and would set up its default route with

```
route add default pop.foo.net
```

Any hosts on the LAN 'behind' *frobozz* would point their default routes through the IP address of *frobozz*'s LAN interface, like

```
route add default frobozz 1
```

A machine's default route should point to the next machine along its path 'outward' to the Internet. If *frobozz* were a remote machine dialing into one machine in a complex corporate internet, its default route should point to that hub machine, expecting that the hub will deal with the issues of routing *frobozz*'s packets to their destination, whether on the LAN or elsewhere on the corporate internet or onto the Internet.

5.2 Software Updates

5.2.1 Failure Recovery from EPROM

If your Morning Star *Express* was delivered with two EPROM devices and two flash devices, you can recover from catastrophic failures (e.g. flash corruption) this way:

1. Remove power cord
2. Remove the lid as described in section 2.2.1 on page 17.
3. With the lid off:
   (a) Place the *Express* router on the table so that the front of the case is farthest from you, and the power and network connectors are toward you.

(b) Remove jumper **JP10**, as shown in Figure 2.1 or Figure 2.2

(c) Insert power cord, and the router will reboot with factory defaults

(d) Login as `root`, with no password

(e) Type `duplicate-flash`
This will copy the contents of EPROM to your flash. You now have flash which contains the software version with which your router shipped form the factory, and the default configuration files.

(f) Remove power cord

(g) Replace **JP10**

4. Replace lid

5. Insert power cord

6. Proceed with configuration

5.2.2 Software via the Internet

To get a software update for your Morning Star *Express* via the Internet,

1. FTP to *ftp.MorningStar.Com*, which is at IP address *137.175.2.7* right now.

2. Once there, you should find the latest `mse` file in the directory `/pub/Express/mse-`*version*. Copy that file to a host on your network, being sure that you perform the FTP transfer in binary mode.

   Though you'll be connecting anonymously with our FTP server and the documentation is freely accessible, you'll need to get an FTP password from the Morning Star Technologies technical support staff before you can get new software. Here's a typical session:

```
% ftp -v ftp.morningstar.com
Connected to remora.MorningStar.Com.
220 remora FTP server (Version 6.14
        Thu Dec 26 11:24:28 EST 1991) ready.
Name (ftp.morningstar.com:bob): anonymous
331 Guest login ok, send e-mail address as password.
Password:
230-Welcome to the Morning Star Technologies support
```

```
        server at Thu Nov 11 15:30:37 1993
230-
230 Guest login ok, access restrictions apply.
ftp> bin
200 Type set to I.
ftp> hash
Hash mark printing on (8192 bytes/hash mark).
ftp> prompt
Interactive mode off.
ftp> cd pub/Express
250 CWD command successful.
ftp>
ftp> dir MST*
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
-rw-rw-r-- 1 bob    4666 Feb 15 MST-Express-At-A-Glance
-rw-rw-r-- 2 karl 239891 Feb 21 MST-Express-User-Guide.ps.Z
-rw-rw-r-- 1 bob  188475 Feb 21 MST-Express-User-Guide.ps.gz
#
226 Transfer complete.
remote: MST*
162 bytes received in 0.17 seconds (0.91 Kbytes/s)
ftp> get MST-Express-User-Guide.ps.Z
200 PORT command successful.
150 Opening BINARY mode data connection for
        MST-Express-User-Guide.ps.Z (239891 bytes).
##################################################
226 Transfer complete.
local: MST-Express-User-Guide.ps.Z
        remote: MST-Express-User-Guide.ps.Z
239891 bytes received in 2.4 seconds (98 Kbytes/s)
ftp>
ftp> dir mse*
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
ftp> dir mse*
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
```

```
-rw-r----- 1 root 358562 Feb 19 mse-1.1.1
-rw-r----- 1 root 353250 Feb 19 mse-export-1.1.1
-rw-r----- 1 root 443975 Feb 19 mse-export-ospf-1.1.1
-rw-r----- 1 root 376099 Feb 19 mse-fr-1.1.1
-rw-r----- 1 root 368547 Feb 19 mse-fr-export-1.1.1
-rw-r----- 1 root 458976 Feb 19 mse-fr-export-ospf-1.1.1
-rw-r----- 1 root 466601 Feb 19 mse-fr-ospf-1.1.1
-rw-r----- 1 root 448852 Feb 19 mse-ospf-1.1.1
#
226 Transfer complete.
remote: mse*
68 bytes received in 0.17 seconds (0.38 Kbytes/s)
ftp> get mse-1.1.1
200 PORT command successful.
550 mse-1.1.1: Permission denied.
ftp> quote site group mse9310
200 Request for access to group mse9310 accepted.
ftp> quote site gpass eMYi15vX
200 Group access enabled.
ftp> get mse-1.1.1
200 PORT command successful.
150 Opening BINARY mode data connection for
        mse-1.1.1 (358562 bytes).
##########################################
##########################################
226 Transfer complete.
local: mse-1.1.1 remote: mse-1.1.1
358562 bytes received in 1.7 seconds
        (2.1e+02 Kbytes/s)
ftp>
ftp> dir *sum*
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
-rw-rw-r-- 1 root 244 Feb 23 sums-1.1.1
#
226 Transfer complete.
remote: *sum*
66 bytes received in 0.28 seconds (0.23 Kbytes/s)
```

```
ftp> mget *sum*
200 PORT command successful.
150 Opening BINARY mode data connection for
        sums-1.1.1 (244 bytes).
#
226 Transfer complete.
local: sums-1.1.1 remote: sums-1.1.1
244 bytes received in 0.059 seconds (4.1 Kbytes/s)
ftp>
ftp> quit
221 Goodbye.
%
```

3. If your host system has a `sum` command, use it to find the checksum of the file you have just received via FTP.

4. Compare the checksum you just computed, with the one listed with your software (`mse-1.1.1` in this example) in the `sums-1.1.1` file.

5. Transfer this file to your router with FTP (again, in binary mode), putting it on the router under the name `mse`. This copy will be resident in RAM for now.

6. Use the `cksum -o1` command on your router, and note the checksum of the new `mse` you have just placed there. If it differs from the checksum you noted on your host system and the one in the `sums` file you FTPd from Morning Star, transfer `mse` from your host again.

   If your host runs SysV, use `cksum -o2` on the router instead of `cksum -o1`.

7. To be sure that the new software is undamaged, say `reboot mse`. If the router can't boot the new `mse` from RAM, it will boot the old `mse` from flash memory instead, next time the router is reset.

8. Transfer the file to your router again with a binary FTP, again putting it on the router under the name `mse`.

9. Use the `cksum -o1` command on your router, and note the checksum of the new `mse` you have just placed there. If it differs from the checksum you noted on your host system and the one in the `sums` file you FTPd from Morning Star, transfer `mse` from your host again.

   If your host runs SysV, use `cksum -o2` on the router instead of `cksum -o1`.

10. You must then use the command `repack-flash` in order to save it in the flash memory, ready for the next time the router boots.

11. To run the newly arrived and newly `saved` software from the flash memory, use the `reboot` command. This last step is actually unnecessary, since you will already be running the new software from RAM since the previous `reboot` above, but it's always nice to try it out just to be sure.

5.2.3 Software via Courier

If the technical support staff at Morning Star Technologies determines that you need updated software, and you're unable to use the Internet to get it yourself electronically, we'll send it to you in new EPROM memory via courier service (DHL, UPS, Federal Express, etc.) When you receive the package, here's how to install the new EPROM memory parts:

1. If your router is running, use FTP to save your *Express* router's configuration files on some other system on your network.

2. Take your *Express* router to a work area where you can observe appropriate static precautions.

3. Remove power cord

4. Remove the lid as described in section 2.2.1 on page 17.

5. Place the *Express* router on the table so that the front of the case is farthest from you, and the power and network connectors are toward you.

6. With the lid off, taking appropriate static precautions:

   (a) Remove jumper **JP10**, as shown in Figure 2.1 or Figure 2.2

   (b)
   - 1-Ethernet *Express*
     Install the EPROM part labeled "L" in socket U11 (nearest you), and install the EPROM part labeled "H" in socket U12 (farthest from you). Pin 1 of each EPROM has been removed, but that end of the part should be to the right. The sockets are labeled on the circuit board, and in Figure 2.1 on page 19.
   - 2-Ethernet *Express 2E*
     Install the EPROM part labeled "L" in socket U10 (to the right), and install the EPROM part labeled "H" in socket U11 (to the left). Pin 1

of each EPROM has been removed, but that end of the part should be farthest from you. The sockets are labeled on the circuit board, and in Figure 2.2 on page 21.

(c) Insert the power cord, and the router will reboot with factory defaults

(d) Login as `root`, with no password

(e) Type `duplicate-flash`

(f) Remove the power cord

(g) Replace jumper **JP10**

(h) Replace the lid as described in section 2.2.3 on page 25.

7. Replace the power cord

8. Proceed with installing and configuring your router.

5.3 Booting From a TFTP Server

To make the router boot a (possibly experimental) version of code from a TFTP server, put the following into a file called `tftp-commands`:

```
c hostname-or-IP-address
b
g mse mse
q
```

and put the following into the `rc.boot` file:

```
ifconfig enet0 your-Ethernet-IP-address netmask your-netmask ...
tftp < tftp-commands
reboot mse
```

When the router comes back up, instead of executing the `rc.boot` file, it will execute `rc.test`. Put your normal configuration commands in `rc.test`, including the `ifconfig` commands for the Ethernet port, since it will be uninitialized by the reboot. Make sure to save all of these files in flash memory with the `save` command.

5.4 Using RARP to Set the Ethernet IP Address

RARP (the Reverse Address Resolution Protocol) can be used to set the IP address of the Ethernet interface. Begin by configuring your local RARP server to associate the appropriate IP address to the Ethernet MAC address of your *Express*. Then, use the `rarp` option of the `ifconfig` command to retrieve the IP address from the RARP server. See the `ifconfig` command in section A.1.24 on page 102 for more details.

5.5 Forgotten Password

5.5.1 *Express* or *Express 2E*

If you've forgotten your router's `root` password, or configured your router so that you can no longer change its configuration, temporarily remove jumper **J2**.

The router will boot with the default configuration described in section 2.2.2 on page 18.

5.5.2 *Express PC*

On an *Express PC*, "remove" the **J2** "jumper" by editing the `JUMPERS.` file on a MS-DOS system, changing it to look like

```
11111011
```

The router will boot with the default configuration described in section 2.2.2 on page 24.

Appendix A

Reference Manual

The Reference Manual is divided into these sections, beginning on the indicated pages:

A.1 Commands

A.1.1 ?—Print a list of commands

Synopsis:

```
?
```

Print a list of valid commands in an abbreviated multicolumn format.
See also `help` in section A.1.21 on page 97.

A.1.2 arp—Address resolution display and control

Synopsis:

`arp` *hostname*
`arp -a[n]`
`arp -d` *hostname*
`arp -s` *hostname hardware-addr* [`temp`] [`pub`]
`arp -f` *filename*

The `arp` program displays and modifies the Internet-to-hardware address translation tables used by the address resolution protocol (ARP). With no flags, the program displays the current ARP entry for *hostname*. The *hostname* may be specified by name or by IP address, using Internet "dot notation".

Options are:

`-a` The program displays all of the current ARP entries.

`-n` Display hosts' IP addresses numerically, without resolving the addresses into fully qualified domain names.

`-d` Delete the entry for the host called *hostname*.

`-s` *hostname hardware-addr*
Create an ARP entry for the host called *hostname* with the hardware address *hardware-addr*. For Ethernet devices, the hardware address (MAC address) is given as six hexadecimal bytes separated by colons. For frame relay devices, the hardware address is the tty device name followed by a colon and the DLCI number in decimal. The entry will be permanent unless the word `temp` is given in the command. If the word `pub` is given, the entry will be "published"; i.e., this system

will act as an ARP server, responding to requests for *hostname* even though the host address is not its own.

`-f` Causes the file *filename* to be read and multiple entries to be set in the ARP tables. Entries in the file should be of the form

hostname hardware-addr [`temp`] [`pub`]

with argument meanings as given above.

In the special case of the command

`arp -s 0.0.0.0` *hardware-addr* `pub`

where *hardware-addr* connects to a Frame Relay network, all packets destined for IP addresses with no ARP table entry will be transmitted over the indicated DLCI. This is useful when constructing a network using Frame Relay as the underlying carrier, but when the DLCI link topology is constrained to be point-to-point (usually with a central hub) rather than the fully-connected mesh subnet that is more common with IP-over-Frame Relay networks.

See also `ifconfig` in section A.1.24 on page 102.

A.1.3 cat—Concatenate and display files

Synopsis:

`cat` [*files*]

The `cat` utility reads *files* sequentially in command line order, writing them to the standard output. There are no options.

A.1.4 cksum—Calculate a checksum for a file

Synopsis:

`cksum` [`-o1`|`-o2`] *file*

The `cksum` command calculates and displays a checksum for the named file, and also displays the size of the file. It is typically used to validate a file after it has been transferred over a network, by comparing its checksum after the transfer with the checksum of the original file.

Options are:

`-o1` Behaves the same as BSD's `sum` *file* and SysV's `sum -r` *file*. *file*'s size is displayed in 512-byte blocks.

`-o2` Behaves the same as SysV's `sum` *file*. *file*'s size is displayed in 1024-byte blocks.

default Print a 32-bit CRC compatible with the POSIX 2 `cksum`; it is the same as the CCITT-1 polynomial used on HDLC links using 32-bit FCS. *file*'s size is displayed in bytes.

A.1.5 console—Send console messages to a tty

Synopsis:

`console` [+|−][*ttyname*]

The `console` command adds or deletes a terminal from the list of tty devices receiving system log messages. If used with a + or − alone, it adds or deletes the current terminal. If a *ttyname* is also given, it is added or deleted instead. If a *ttyname* is given without the + or −, the tty is opened and made a console, and the `console` command detaches and runs as a daemon.

A.1.6 cu—TTY access program

Synopsis:

`cu -l` *device* [`-s` *speed*] [`-d`] [`-h`] [`-x`]

The `cu` command establishes a full-duplex terminal connection to a tty device. A tilde ('`~`') appearing as the first character of a line is an escape signal which directs `cu` to perform a special action. The only special action is tilde period ('`~.`'), which drops the connection and exits. A tilde can be sent at the beginning of a line by typing two tildes.

Options:

`-l` *device* The tty port to connect to

`-s` *speed* The tty port speed

`-d` Increment the debugging verbosity

`-h` Enable hardware (RTS/CTS) flow control

`-x` Enable software (XON/XOFF) flow control

A.1.7 date—Set or display the time and date

Synopsis:

`date` [`-r` *seconds*] [*+format*] [[*yy*[*mm*[*dd*[*hh*]]]]*mm*[`.`*ss*]]

`Date` displays the current date and time when invoked without arguments. Providing arguments will either format and print the date and time in a user-defined way or will set the date.

The options are as follows:

`-r` Print out the date and time for *seconds* from the Epoch (Thu Jan 1 00:00:00 GMT 1970).

An operand with a leading plus ("+") sign signals a user-defined format string which specifies the format in which to display the date and time. The format string may contain any of the following conversion specifications, as well as any arbitrary text.

%% same as %

%a day of week, using locale's abbreviated weekday names

%A day of week, using locale's full weekday names

%b month, using locale's abbreviated month names

%h month, using locale's abbreviated month names

%B month, using locale's full month names

%c date and time as %x %X

%C date and time, in locale's long-format date and time representation

%d day of month (01-31)

%D date as %m/%d/%y

%e day of month (1-31; single digits are preceded by a blank)

%H hour (00-23)

%I hour (00-12)

%j day number of year (001-366)

%k	hour (0-23; single digits are preceded by a blank)
%l	hour (1-12; single digits are preceded by a blank)
%m	month number (01-12)
%M	minute (00-59)
%n	same as \n
%p	locale's equivalent of AM or PM, whichever is appropriate
%r	time as %I:%M:%S %p
%R	time as %H:%M
%S	seconds (00-59)
%t	same as \t
%T	time as %H:%M:%S
%U	week number of year (01-52), Sunday is the first day of the week
%w	day of week; Sunday is day 0
%W	week number of year (01-52), Monday is the first day of the week
%x	date, using locale's date format
%X	time, using locale's time format
%y	year within century (00-99)
%Y	year, including century (for example, 1988)
%Z	time zone abbreviation

The difference between *%U* and *%W* lies in which day is counted as the first day of the week. Week number 01 is the first week with four or more January days in it.

The format string for the default display is:

```
"%a %b %e %H:%M:%S %Z %Y\,%n"
```

If an operand does not have a leading plus sign, it is interpreted as a value for setting the system's notion of the current date and time. The canonical representation for setting the date and time:

yy	Year in abbreviated form (.e.g 89 for 1989).
mm	Numeric month. A number from 1 to 12.
dd	Day, a number from 1 to 31.
hh	Hour, a number from 0 to 23.
mm	Minutes, a number from 0 to 59.
ss	Seconds, a number from 0 to 59.

Everything but the minutes are optional.

Time changes for Daylight Saving and Standard time and leap seconds and years are handled automatically.

Examples

The command:

```
date "+DATE: %m/%d/%y%nTIME: %H:%M:%n"
```

will display:

```
DATE: 11/21/87
TIME: 13:36:16
```

The command:

```
date 8506131627
```

sets the date to "June 13 1985, 4:27 PM". The command:

```
date 1432
```

sets the time to 2:32 PM, without modifying the date.

See also `rdate` in section A.1.38 on page 126, and `tz` in section A.1.56 on page 148.

A.1.8 dmesg—Display recent system log messages

Synopsis:

```
dmesg
```

This command displays the last 4000 bytes or so of system log messages.

See also `syslogd` in section A.1.50 on page 134, and `syslog.conf` in section A.2.9 on page 157.

A.1.9 eb—Examine bytes in memory

Synopsis:

`eb` *hex-address*

Bytes are displayed with their address one at a time. Typing a period (‘`.`’) terminates the command.

See also `ew` in section A.1.15 on page 87, `el` in section A.1.14 on page 87, `eiob` in section A.1.12 on page 86, `eiow` in section A.1.13 on page 87, `inb` in section A.1.25 on page 104, and `outb` in section A.1.33 on page 110.

A.1.10 echo—Echo arguments

Synopsis:

`echo` [`-n`] *arguments*

This command writes the list of arguments followed by a newline to its standard output. The `-n` argument prevents the final newline from being printed.

A.1.11 edit—Edit a text file

Synopsis:

`edit` *filename*

`Edit` is an interactive line-oriented text editor. Commands are a single character, possibly followed by arguments. Commands are:

`a` *line number*
: Enter append mode starting after the named line

`i` *line number*
: Enter append mode starting before the named line

`e` *line number*
: Enter edit mode on the named line

`d` *range* Delete the named lines

`p` *range* Print the named lines

range Print the named lines

carriage return
: Print the next line

`-` : Print the previous line

`.` : Print the number of the current line

`w` : Write the file

`q` : Exit editor, return to command shell

`?` : Print a help message

Leave append mode with *Control-D* or a dot ('.') on a line by itself. Leave edit mode with a carriage return. A line number can be a number, a dot (current line), a minus sign ('`-`', previous line), a plus sign ('+', next line), or a dollar sign ('`$`', last line). A range is either a line number or two line numbers separated by a '`-`'. `Edit` will not write the edited file unless explicitly told to do so with the '`w`' command.

When in editing mode, the following characters are used (C-*X* means *Control-X*; hold down the *Control* key and type the *X* character):

newline : Terminate emacs editing mode

return : Terminate emacs editing mode

C-`A` : Position the cursor to the first character on the line

C-`B` : Position the cursor to the previous character on the line ("back")

C-`C` : Cancel editing this line

C-`D` : Delete the character under the cursor

C-`E` : Position the cursor after the last character on the line ("end")

C-`F` : Position the cursor to the next character on the line ("forward")

C-`H` : Delete the previous character

delete : Delete the previous character

tab : Insert a tab

C-`K` : Delete from the cursor to the end of the line ("kill")

C-`L` : redraw the current line

C-N	Go to the next line
C-O	Toggle Overwrite-mode (*not* like Emacs!)
C-P	Go to the previous line
C-R	Reverse incremental search
C-S	Forward incremental search
C-T	Transpose characters
C-U	Delete from the cursor to the beginning of the line (*not* like Emacs!)
C-W	Delete from the cursor to the beginning of the word
C-Y	Yank
esc-B	Move the cursor to the beginning of the word
esc-D	Delete the next word
esc-F	Move the cursor to the beginning of the following word

A.1.12 eiob—Examine bytes in I/O space

(Available only on *Express PC*)

Synopsis:

`eiob` *hex-address*

Bytes from I/O space are displayed with their address one at a time. Typing a period (‘`.`’) terminates the command.

See also `eb` in section A.1.9 on page 84, `ew` in section A.1.15 on page 87, `el` in section A.1.14 on page 87, `eiow` in section A.1.13 on page 87, `inb` in section A.1.25 on page 104, and `outb` in section A.1.33 on page 110.

A.1.13 eiow—Examine words in I/O space

(Available only on *Express PC*)
Synopsis:

`eiow` *even-hex-address*

Words from I/O space are displayed with their address one at a time. Typing a period (‘`.`’) terminates the command.

See also `eb` in section A.1.9 on page 84, `ew` in section A.1.15 on page 87, `el` in section A.1.14 on page 87, `eiob` in section A.1.12 on page 86, `inb` in section A.1.25 on page 104, and `outb` in section A.1.33 on page 110.

A.1.14 el—Examine longwords in memory

Synopsis:

`el` *even-hex-address*

Longwords are displayed with their address one at a time. Typing a period (‘`.`’) terminates the command.

See also `eb` in section A.1.9 on page 84, `ew` in section A.1.15 on page 87, `eiob` in section A.1.12 on page 86, and `eiow` in section A.1.13 on page 87.

A.1.15 ew—Examine words in memory

Synopsis:

`ew` *even-hex-address*

Short words are displayed in hexadecimal along with their address one at a time. Typing a period (‘`.`’) terminates the command.

See also `eb` in section A.1.9 on page 84, `el` in section A.1.14 on page 87, `eiob` in section A.1.12 on page 86, and `eiow` in section A.1.13 on page 87.

A.1.16 frd—Frame relay daemon

Frame Relay is optional on the *Express* router, and is unavailable on the *Express 2E Express PC* routers.
Synopsis:

`frd` [*options*]

Optional arguments:

`debug` *n* Set the debugging level to *n* (default 1).

`addr` *loc* Set the local interface address to *loc*.

`netmask` *n*
Set interface subnet mask to *n* (default is by network class).

`mtu` *n* Set interface MTU value to *n* (default 1500).

`filter` *file*
Set the name of the filter file to *file* (default if `filter` *filter-file* is not specified: `Filter`)

`gw-crypt` *key-file*
Encrypt traffic between the pairs of hosts or networks specified in the designated *key-file*.

Encryption is not available in products exported from the USA.

`nodetach`
Don't detach from the controlling tty.

`ignore-cd`
Don't pay attention to CD (Carrier Detect). This is useful when the cabling or communications equipment (DCE) don't properly support CD.

`internal-clocking`
Supply TxCLK and RxCLK on the serial connector. A speed must also be specified. The **Internal clocking** jumpers corresponding to the appropriate serial port must also be set; see *Setting the Jumpers* in section 2.2.2 on page 18.

`external-clocking`
Expect the modem, CSU/DSU, or modem eliminator to provide the synchronous clock signal, TxCLK and RxCLK, on the serial connector. This is the default behavior. The **External clocking** jumpers corresponding to the appropriate serial port must also be set; see section 2.2.2, *Setting the Jumpers*, on page 18.

speed Set the communications data rate to *speed*.

device Communicate over tty *device*.

`lmi` *type* Use LMI frames of type *type* to communicate with the Frame Relay switch. *type* can be one of *default*, *annex-d*, *stratacom*, or *none* (default *default*).

`N1` *frequency*
One out of every *frequency* LMI status enquiries will request a full LMI status response (default 1, meaning every LMI status enquiry requests a full LMI status response).

`N2` *threshold*
The number of LMI errors out of the last `N3` that must occur before an error is reported (default 2). `N2` must not exceed `N3`.

`N3` *out-of* The measurement interval for `N2` (default 4 LMI polls).

`T1` *interval*
The LMI status enquiry polling interval (default 10 seconds).

Signal Handling

Upon reception of the following signals, `frd` re-reads the `Filter` and `Key` files, then takes the indicated actions:

SIGHUP
Try to re-establish the link.

SIGINT Shut the link down gracefully, then try to re-establish the link.

SIGTERM
Shut the link down and exit.

SIGALRM
No further action.

SIGUSR1
Increments debugging verbosity level.

SIGUSR2
Resets the debugging verbosity level to the "base level", 1 by default.

Debugging Verbosity Levels

`Frd` runs by default at debugging verbosity level 1. Higher verbosity levels provide more detailed information. Debugging levels above 1 may be specified on the `frd` command line, or the verbosity of an active daemon may be adjusted by sending it SIGUSR1 and SIGUSR2 signals. The various debugging levels provide the following information:

1	Daemon startup, link status, important messages
2	Framing and other errors
3	IP interface changes and LMI message summaries
4	IP message summaries
5	All messages (without framing), unrecognized packet type errors
6	Procedure call traces
7	Internal timers

Each debugging verbosity level also provides the information of all the lower-numbered levels.

Messages at debugging levels 0 and 1 use `syslog` facility `daemon.info`. Messages at debugging levels 2 through 7 use `syslog` facility `daemon.debug`. Fatal errors use `syslog` facility `daemon.crit`.

LMI

Unless it is invoked with `lmi` *none*, `frd` sends one LMI frame each second until it receives a response. When the first LMI response arrives, `frd` recognizes the link as being up, and ready to carry network datagrams. Thereafter, `frd` sends one LMI every ten seconds. If LMI decides that the link is down (probably because LMI responses stop arriving), then `frd` also regards the link as down, and unavailable to carry network datagrams. `frd` begins sending one LMI frame per second until the link is up again.

If *frd* is invoked with `lmi` *none*, `frd` regards the link as being up and ready for IP traffic as soon as the Carrier Detect signal is received from the DCE. If Carrier Detect disappears, `frd` regards the link as down, and unavailable to carry network datagrams.

The *Express* works with Cascade Frame Relay switches when `frd` is invoked with no explicit `lmi` specification, or with either `lmi` *default* or `lmi` *stratacom*.

The *Express* works with Northern Telecom DMS-100 switches when `frd` is invoked with `lmi` *default*, but not with `lmi` *annex-d*.

See Also

See also `arp` in section A.1.2 on page 78, `Filter` in section A.3.5 on page 172, the Filter discussion in section 4.5.3 on page 53, RFC 1490 , and ANSI T1.617 Annex D. (CCITT Q.933 Annex A seems to be compatible with ANSI T1.617 Annex D.)

A.1.17 ftpd—File Transfer Protocol server

Synopsis:

`ftpd` [`-d`] [`-l`] [`-t` *timeout*] [`-T` *maxtimeout*]

`Ftpd` is the Internet File Transfer Protocol server process. The server uses the TCP protocol and listens at the port specified in the `ftp` service specification; see `services` on in section A.2.8 on page 156.

Available options:

`-d` Debugging information is written to the system log.

`-l` Each `ftp` session is sent to the system log.

`-t` The inactivity timeout period is set to *timeout* seconds (the default is 15 minutes).

`-T` A client may also request a different timeout period; the maximum period allowed may be set to *maxtimeout* seconds with the `-T` option. The default limit is 2 hours.

The ftp server currently supports the following ftp requests; case is not distinguished.

ABOR abort previous command

ACCT specify account (ignored)

ALLO allocate storage (vacuously)

APPE append to a file

DELE delete a file

HELP give help information

LIST give a list of files (“ls -lgA”)

MDTM show last modification time of file

MODE specify data transfer mode

NLST give name list of files

NOOP do nothing

PASS specify password

PASV prepare for server-to-server transfer

PORT specify data connection port

QUIT terminate session

REST restart incomplete transfer

RETR retrieve a file

RNFR specify rename-from file name

RNTO specify rename-to file name

SITE non-standard commands (see below)

SIZE return size of file

STAT return status of server

STOR store a file

STOU store a file with a unique name

STRU specify data transfer structure

SYST show system type of server

TYPE specify data transfer type

USER specify user name

The following non-standard commands are supported by the SITE request.

IDLE set idle-timer. E.g. SITE IDLE 60

HELP give help information. E.g. SITE HELP

EXEC run the specified command

The remaining ftp requests specified in Internet RFC 959 are recognized, but not implemented. MDTM and SIZE are not specified in RFC 959 , but will appear in the next updated FTP RFC.

The ftp server will abort an active file transfer only when the ABOR command is preceded by a Telnet "Interrupt Process" (IP) signal and a Telnet "Synch" signal in the command Telnet stream, as described in Internet RFC 959 . If a STAT command is received during a data transfer, preceded by a Telnet IP and Synch, transfer status will be returned.

The user name must be in the password data base, `passwd`, and must not have a null password. A password must be provided by the client before any file operations may be performed.

You can use the `SITE EXEC` feature from your FTP client (running on your workstation) like this:

```
put Filter
quote site exec save Filter
```

or maybe

```
put mse-ospf mse
quote site exec cksum -o1 mse
quote site exec reboot mse
```

This can save the trouble of telnetting to the *Express* router and logging in, just to save the changed file to Flash, or to check its integrity.

See also `inetd` in section A.1.26 on page 104, and `syslogd` in section A.1.50 on page 134.

A.1.18 gated—Start gateway routing daemon

Synopsis:

`gated` [`-c`] [`-C`] [`-n`] [`-N`] [`-t` *trace-options*] [`-f` *config-file*] [*trace-file*]

`Gated` is a routing daemon that handles multiple routing protocols. `Gated` can handle the RIP, BGP, EGP, HELLO, and OSPF routing protocols. The `gated` process can be configured to perform all routing protocols or any combination of the five. In order to conserve memory, `gated` on the *Express* router is usually compiled to support only a subset of these protocols; contact Morning Star Technologies for more information.

`Gated` may be run only once during any *Express* boot session. If you must restart *gated* after it has already been run since the last time your *Express* was booted, you must `reboot` your *Express*.

The command-line options are:

`-c` Specifies that the configuration file will be parsed for syntax errors and then `gated` will exit. The `-c` option implies `-tgeneral,kernel,nostamp`. All `traceoption` and `tracefile` clauses in the configuration file will be ignored.

`-C` Specifies that the configuration file will just be parsed for syntax errors. The `-C` option implies `-tnostamp`.

`-n` Specifies that `gated` will not modify the kernel's routing table. This is used for testing `gated` configurations with actual routing data.

`-N` Specifies that `gated` will not daemonize. Normally, if tracing to *stderr* is not specified, `gated` will daemonize if the parent process ID is not 1.

`-t` Specifies a comma separated list of trace options to be enabled on startup. If no flags are specified, `general` is assumed. No space is allowed between this option and its arguments.

This option must be used to trace events that take place before the config file is parsed, such as determining the interface configuration and reading routes from the kernel.

The trace options are explained in greater detail in the section on `gated.conf`.

`-f` Use an alternate config file. By default, `gated` uses `gated.conf`.

If on the command line a trace file is specified, or no trace flags are specified, `gated` detaches from the terminal and runs in the background. If trace flags are specified without specifying a trace file, `gated` assumes that tracing is desired to stderr and remains in the foreground.

Signal Handling

`Gated` catches the following signals and does special processing.

SIGHUP

Re-read configuration.

A SIGHUP causes `gated` to reread the configuration file. `Gated` first performs a clean-up of all allocated policy structures. All BGP and EGP peers are flagged for deletion and the configuration file is reparsed.

If the re-parse is successful, any BGP and EGP peers that are no longer in the configuration are shut down, and new peers are started. `Gated` attempts to determine if changes to existing peers require a shutdown and restart.

It should also be possible to enable/disable any protocol without restarting `gated`.

Reconfiguration is currently disabled when OSPF is enabled; this will hopefully be fixed in a future release.

SIGINT Snap-shot of current state.

The current state of all `gated` tasks, timers, protocols and tables are written to `gated.dump`.

`Gated` immediately processes the dump, which may impact gated's routing functions.

SIGTERM

Graceful shutdown.

On receipt of a SIGTERM, `gated` attempts a graceful shutdown. All tasks and protocols are asked to shut down. Most will terminate immediately, the exception being EGP peers which wait for confirmation. It may be necessary to repeat the SIGTERM once or twice if it this process takes too long.

All exterior routes (BGP and EGP) are removed from the kernel's routing table on receipt of a SIGTERM. Interior routes (all others) remain. To terminate `gated` with the exterior routes intact, use SIGKILL.

SIGUSR1

Toggle tracing.

On receipt of a SIGUSR1, `gated` will close the trace file. A subsequent SIGUSR1 will cause it to be reopened. This allows the file to be moved.

It is not possible to use SIGUSR1 if a trace file has not been specified, or if tracing is being performed to stderr.

SIGUSR2

Check for interface changes.

On receipt of a SIGUSR2, `gated` will rescan the kernel interface list looking for changes.

Authors

Mark Fedor <fedor@psi.com>, Jeffrey C. Honig <jch@gated.cornell.edu>, Rob Coltun <rcoltun@ni.umd.edu> and Dennis Ferguson <dennis@ans.net>

See also `gated.conf` in section A.4.1 on page 184, `arp` in section A.1.2 on page 78, `ifconfig` in section A.1.24 on page 102, `netstat` in section A.1.31 on page 107, RFC 891 (DCN Local-Network Protocols (HELLO)), RFC 904 (Exterior Gateway Protocol Formal Specification), RFC 1058 (Routing Information Protocol), RFC 1163 (A Border Gateway Protocol (BGP)), RFC 1164 (Application of the Border Gateway Protocol in the Internet), and RFC 1247 (OSPF Specification, Version 2).

A.1.19 gdb—Transfer control to the gdb debugger

This command enables an internal debugging client and waits to be contacted by a remote debugger.

Do not run this command except under the direction of Morning Star Technologies support staff.

A.1.20 getty—Enable logins on a serial port

Synopsis:

`getty` *tty-port speed* [*options*]

Options are:

`riwait` Wait for RI to rise before raising DTR (default)

`cdwait` Wait for CD to rise before raising DTR

`nowait` Don't wait for RI or CD

`respawn`
After the child (login) process exits, start again

`p_even` Print login and password prompts in even parity (this is the default)

`p_zero` Print login and password prompts in 8 bits, no parity

`timeout` *n*
Require login to complete within *n* seconds (default is 60, or 0 if `nowait` is specified)

`debug` Print debugging messages to the system log

`Getty` monitors the modem control signals on a *tty-port* (depending on which of the `riwait`, `cdwait`, or `nowait` options are given), opens and initializes the device, prints a login prompt and reads a login name, reads and verifies the password against the `passwd` file, and then runs the command found in the `passwd` file entry. If the `respawn` option is not specified, `getty` then exits; otherwise, it waits for the spawned command to exit and then repeats the entire process.

If the passwd file contains an entry with username `PPP` and a null password, such as

```
PPP:::pppd myhostname:  rtscts idle 60 requirechap
```

then if `getty` sees an incoming PPP frame, it will respond by invoking the specified command. In this example, the shell is `pppd` with the indicated arguments. This relieves the calling system of the need to negotiate through a login/password chat script before commencing LCP negotiations. For security, be sure to employ link-level authentication!

See also `passwd` in section A.2.4 on page 154, and `sgetty` in section A.1.46 on page 132.

A.1.21 help—Get help on commands

Synopsis:

```
help
```

Print a one-line description of each available command. See also `?` in section A.1.1 on page 78.

A.1.22 host—look up host names using domain server

Synopsis:

`host` [*-w*] [*-v*] [*-r*] [*-d*] [*-t* `querytype`] [*-c* `class`] [*-a*] *host* [*server*]

`Host` looks for information about Internet hosts. It gets this information from a set of interconnected servers that are spread across the country. By default, it simply converts between host names and Internet addresses. However with the *-t* or *-a* options, it can be used to find all of the information about this host that is maintained by the domain server.

The arguments can be either host names or host numbers. The program first attempts to interpret them as host numbers. If this fails, it will treat them as host names. A host number consists of first decimal numbers separated by dots, e.g. 128.6.4.194 A host name consists of names separated by dots, e.g. topaz.rutgers.edu. Unless the name ends in a dot, the local domain is automatically tacked on the end. Thus a Rutgers user can say "host topaz", and it will actually look up "topaz.rutgers.edu". If this fails, the name is tried unchanged (in this case, "topaz"). This same convention is used for mail and other network utilities. The actual suffix to tack on the end is obtained by looking at the results of a "hostname" call, and using everything starting at the first dot. (See below for a description of how to customize the host name lookup.)

The first argument is the host name you want to look up. If this is a number, an "inverse query" is done, i.e. the domain system looks in a separate set of databases used to convert numbers to names.

The second argument is optional. It allows you to specify a particular server to query. If you don't specify this argument, the default server (normally the local machine) is used. The query proceeds according to the rules specified in `resolv.conf`.

If a name is specified, you may see output of three different kinds. Here is an example that shows all of them:

```
# host sun4
sun4.rutgers.edu is a nickname for
ATHOS.RUTGERS.EDU
ATHOS.RUTGERS.EDU has address 128.6.5.46
ATHOS.RUTGERS.EDU has address 128.6.4.4
ATHOS.RUTGERS.EDU mail is handled by
ARAMIS.RUTGERS.EDU #
```

The user has typed the command "host sun4". The first line indicates that the name "sun4.rutgers.edu" is actually a nickname. The official host name is

"ATHOS.RUTGERS.EDU". The next two lines show the address. If a system has more than one network interface, there will be a separate address for each. The last line indicates that ATHOS.RUTGERS.EDU does not receive its own mail. Mail for it is taken by ARAMIS.RUTGERS.EDU. There may be more than one such line, since some systems have more than one other system that will handle mail for them. Technically, every system that can receive mail is supposed to have an entry of this kind. If the system receives its own mail, there should be an entry the mentions the system itself, for example "XXX mail is handled by XXX". However many systems that receive their own mail do not bother to mention that fact. If a system has a "mail is handled by" entry, but no address, this indicates that it is not really part of the Internet, but a system that is on the network will forward mail to it. Systems on Usenet, Bitnet, and a number of other networks have entries of this kind.

There are a number of options that can be used before the host name. Most of these options are meaningful only to the staff who have to maintain the domain database. The available options are:

-w Wait forever until reply

-v Verbose output

-r Disable recursive processing

-d Turn on debugging output

-t `querytype`
Look for a specific type of information

-c `class` Class to look for non-Internet data

-a Same as *-v -t* *

host Domain or host name to ask about

server Domain name server to query

The option `-w` causes host to wait forever for a response. Normally it will time out after around a minute.

The option `-v` causes printout to be in a "verbose" format. This is the official domain master file format, which is documented in the UNIX man page for "named". Without this option, output still follows this format in general terms, but some attempt is made to make it more intelligible to normal users. Without `-v`, "a", "mx", and "cname" records are written out as "has address", "mail is handled by", and "is a nickname for", and TTL and class fields are not shown.

The option `-r` causes recursion to be turned off in the request. This means that the name server will return only data it has in its own database. It will not ask other servers for more information.

The option `-d` turns on debugging. Network transactions are shown in detail.

The option -t allows you to specify a particular type of information to be looked up. Query types include

`afsdb`	Andrew File System
`a`	a host address
`cname`	the canonical name for an alias
`gid`	
`hinfo`	host information
`isdn`	ISDN address
`mb`	a mailbox domain name (experimental)
`md`	a mail destination (Obsolete - use MX)
`mf`	a mail forwarder (Obsolete - use MX)
`mg`	a mail group member (experimental)
`minfo`	mailbox or mail list information
`mr`	a mail rename domain name (experimental)
`mx`	mail exchange
`ns`	an authoritative name server
`null`	any data (experimental)
`ptr`	a domain name pointer
`rp`	responsible person
`rt`	route through
`soa`	marks the start of a zone of authority
`txt`	descriptive text strings

`uid`

`uinfo`

`unspec`

`wks` a well known service description

`x25` PSDN address

`any` wildcard

`*` wildcard

The default is to look first for "a", and then "mx", except that if the verbose option is turned on, the default is only "a".

The option `-a` (for "all") is equivalent to `-v -t` *any*.

The option `-c` allows you to specify a particular class of information to be looked up. Query classes include

in the Internet

cs the CSNET class (Obsolete - used only for examples in some obsolete RFCs)

ch the CHAOS class

hs Hesiod

See also `resolv.conf` in section A.2.7 on page 155, RFC 1035, and RFC 1183.

A.1.23 hostname—Set or print the name of the router

Synopsis:

`hostname` [*nameofhost*]

`Hostname` prints the name of the current host. The system's hostname can be changed by supplying an argument; this should usually be done in the initialization script `rc.boot`, normally run at boot time.

A.1.24 ifconfig—Configure network interfaces

Synopsis:

```
ifconfig -a
ifconfig interface [address [dest-address]] [parameters]
```

`Ifconfig` is used to assign an address to a network interface and/or configure network interface parameters. `Ifconfig` can be used at boot time to define the network address of interfaces present on a machine; it may also be used at a later time to redefine an interface's address or other operating parameters. Run with an *interface* name and no other arguments, `ifconfig` will display the current parameters for that *interface*. Run with the `-a` option, `ifconfig` will display the current parameters for all configured interfaces.

Available operands for `ifconfig`:

interface The *interface* parameter is a string of the form *name unit*, for example, `enet0`

address The *address* is either a host name as translated with the Domain Name System (DNS) or an Internet address expressed in the Internet standard "dot notation".

The following parameters may be set with ifconfig:

`alias` *address*
Establish an additional network address for this interface. This is sometimes useful when changing network numbers, and one wishes to accept packets addressed to the old interface.

`broadcast` *address*
Specify the address to use to represent broadcasts to the network. The default broadcast address is the address with a host part of all ones.[1]

`delete` *address*
Remove the network address specified. This would be used if you incorrectly specified an alias, or it was no longer needed.

[1] Some older systems, and networks configured around those systems, used a broadcast address of all zeroes. If other hosts on your network use the old all-zeroes broadcast address, your *Express* router will need to be configured that way also. For more discussion, see section 3.3.6 of RFC 1122 .

dest-address address
Specify the address of the peer on the other end of a point to point link.

`down` Mark an interface "down". When an interface is marked "down", the system will not attempt to transmit messages through that interface. If possible, the interface will be reset to disable reception as well. This action does not automatically disable routes using the interface.

`metric` *n*
Set the routing metric of the interface to n, default 0. The routing metric is used by the routing protocol (see `gated`). Higher metrics have the effect of making a route less favorable; metrics are counted as additional hops to the destination network or host.

`netmask` *mask*
Specify how much of the address to reserve for subdividing networks into sub-networks. The mask includes the network part of the local address and the subnet part, which is taken from the host field of the address. The mask can be specified as a single hexadecimal number with a leading 0x, with a dot-notation Internet address, or with a pseudo-network name translated with the Domain Name System. The mask contains ones for the bit positions in the 32-bit address which are to be used for the network and subnet parts, and zeros for the host part. The mask should contain at least the standard network portion, and the subnet field should be contiguous with the network portion. If none is specified, the default network mask is determined from the class of the IP address of the interface (the local address of a point-to-point link), as if the network were not subnetted.

`up` Mark an interface "up". This may be used to enable an interface after an `ifconfig down`. It happens automatically when setting the first address on an interface. If the interface was reset when previously marked down, the hardware will be re-initialized.

`ether` Transmit packets on an Ethernet interface using Ethertypes. This is the default.

`snap` Transmit packets on an Ethernet interface using SNAP encapsulation. This is the opposite of `ether`.

`rarp` Use the Reverse Address Resolution Protocol (RARP) to configure the address of this *interface*. This option may only be used on Ethernet interfaces. If invoked with `rarp`, `ifconfig` detaches and runs in the background.

`filter` *filter-file*
Look in *filter-file* for packet filtering information.

`gw-crypt` *key-file*
Encrypt traffic between the pairs of hosts or networks specified in the designated *key-file*.

Encryption is not available in products exported from the USA.

`Ifconfig` displays the current configuration for a network interface when no optional parameters are supplied. If a protocol family is specified, `ifconfig` will report only the details specific to that protocol family.

See also `netstat` in section A.1.31 on page 107, `gated` in section A.1.18 on page 93, `pppd` in section A.1.36 on page 87, `frd` in section A.1.16 on page 87, and the Filter discussion in section 4.5.3 on page 53.

A.1.25 inb–Read a byte in I/O space

(Available only on *Express PC*)

Synopsis:

`inb` *hex-address*

Bytes in I/O address space are displayed with their address one at a time. Typing a period (‘`.`’) terminates the command.

See also `eb` in section A.1.9 on page 84, `ew` in section A.1.15 on page 87, `el` in section A.1.14 on page 87, `eiob` in section A.1.12 on page 86, `eiow` in section A.1.13 on page 87, and `outb` in section A.1.33 on page 110.

A.1.26 inetd—Internet services daemon

Synopsis:

`inetd` [`-d`] [*configuration-file*]

`Inetd` should be run at boot time by `rc.boot`. It then listens for connections on certain internet sockets. When a connection is found on one of its sockets, it

decides what service the socket corresponds to, and invokes a program to service the request. After the program is finished, it continues to listen on the socket (except in some cases which will be described below). Essentially, `inetd` allows running one daemon to invoke several others, reducing load on the system.

The one option available for `inetd`, `-d`, turns on debugging messages.

Upon execution, `inetd` reads its configuration information from a configuration file which, by default, is `inetd.conf`. All fields must be present in each non-empty line of the configuration file, with entries for each field separated by a tab or a space. Comments are denoted by a "#" at the beginning of a line. There must be an entry for each field. The fields of the configuration file are as follows:

service-name socket-type protocol `wait|nowait` *user program arguments*

The *service-name* entry is the name of a valid service in the file `services`. For "internal" services (discussed below), the *service-name* must be the official name of the service (that is, the first entry of that name in `services`).

The *socket-type* should be one of `stream`, `dgram`, `raw`, `rdm`, or `seqpacket`, depending on whether the socket is a stream, datagram, raw, reliably delivered message, or sequenced packet socket.

The protocol must be a valid protocol as given in the `protocols` file. Examples might be `tcp` or `udp`.

`Wait` and `nowait` are applicable to datagram sockets only (other sockets should have a `nowait` entry in this space). If a datagram server connects to its peer, freeing the socket so `inetd` can receive further messages on the socket, it is said to be a *multi-threaded* server, and should use the `nowait` entry. For datagram servers which process all incoming datagrams on a socket and eventually time out, the server is said to be *single-threaded* and should use a `wait` entry.

The *user* entry is ignored; it should be set to `root`.

The *program* entry should contain the name of the program which is to be executed by `inetd` when a request is found on its socket. If inetd provides this service internally, this entry should be `internal`.

The *arguments* should be just as command arguments normally are, starting with the name of the program. If the service is provided internally, the word `internal` should take the place of this entry.

`Inetd` provides several "trivial" services internally by use of routines within itself. These services are `echo`, `discard`, `chargen` (character generator), `daytime` (human readable time), and `time` (machine readable time, in the form of the number of seconds since midnight, January 1, 1900). All of these services are tcp based. For details of these services, consult the appropriate RFC from the Network Information Center.

`Inetd` rereads its configuration file when it receives a SIGHUP signal. Services may be added, deleted or modified when the configuration file is reread.

See also `ftpd` in section A.1.17 on page 91, and `telnetd` in section A.1.52 on page 141.

A.1.27 kill—Send a signal to a process

Synopsis:

`kill` [−*signal*] *pids*

Send a signal to one or more running programs. *Signal* is one of

SIGHUP
: Hangup

SIGINT Interrupt

SIGIO I/O activity

SIGKILL
: Terminate abruptly

SIGALRM
: Alarm

SIGCHLD
: Child process terminated

SIGUSR1
: Increase debugging verbosity level

SIGUSR2
: Decrease debugging verbosity level

SIGTERM
: Terminate gracefully—default if none specified

Saying `kill` −*XXX pid* will send a SIG*XXX* signal to the program with process ID *pid*.

By default, the SIGHUP, SIGINT, SIGIO, SIGKILL, SIGALRM, and SIGTERM signals cause a process to terminate, while SIGCHLD, SIGUSR1, and SIGUSR2 are ignored. Any of these, except for SIGKILL, can be caught by a program and some other action taken. For example, SIGUSR1 and SIGUSR2 are frequently used

to increase and decrease log message verbosity levels, and SIGHUP is sometimes used to cause a daemon to reread its configuration file. See the descriptions of each program for specific actions, if any.

A.1.28 ls—List files in memory or flash or on floppy

Synopsis:

`ls` [`-f`] [`-S`] [*files*]

`Ls` lists the named *files* in the memory-resident filesystem, or, if the `-f` option is given, the named *files* saved in flash memory or on floppy. If no filenames are specified, all files will be listed.

If the `-S` option is given, the named *files* will be listed in a short format.

A.1.29 memory—Print memory use statistics

Synopsis:

`memory`

`Memory` prints memory usage information by size and by function. It also displays the amount of flash memory in use.

A.1.30 mv—Rename a file in memory

Synopsis:

`mv` *old-filename new-filename*

`Mv` changes the name of *old-filename* to be *new-filename*.

A.1.31 netstat—Show network status

Synopsis:

`netstat` [`-Aan`] [`-f` *address-family*]
`netstat` [`-dimnqrst`] [`-f` *address-family*]
`netstat` [`-n`] [`-I` [*interface*]] [`-w` *wait*]
`netstat` [`-p` *protocol*]

The `netstat` command symbolically displays the contents of various network-related data structures. There are a number of output formats, depending on the options for the information presented. The first form of the command displays a list of active sockets for each protocol. The second form presents the contents of one of the other network data structures according to the option selected. Using the third form, with a *wait* interval specified, `netstat` will continuously display the information regarding packet traffic on the configured network interfaces. The fourth form displays statistics about the named protocol.

The options have the following meaning:

`-A` With the default display, show the address of any protocol control blocks associated with sockets; used for debugging.

`-I` *interface*
Show information only about this interface; used with a wait interval as described below.

`-a` With the default display, show the state of all sockets; normally sockets used by server processes are not shown.

`-d` With either interface display (option `-i` or an interval, as described below), show the number of dropped packets.

`-f` *address-family*
Limit statistics or address control block reports to those of the specified address family. The only address family recognized is `inet`.

`-i` Show the state of interfaces which have been configured.

`-m` Show statistics recorded by the memory management routines (the network manages a private pool of memory buffers).

`-n` Show network addresses as numbers (normally netstat interprets addresses and attempts to display them symbolically). This option may be used with any of the display formats.

`-p` *protocol*
Show statistics about *protocol*, which is either a well-known name for a protocol or an alias for it. Some protocol names and aliases are listed in the file `protocols`. A null response typically means that there are no interesting numbers to report. The program will complain if protocol is unknown or if there is no statistics routine for it.

`-q` Print queue lengths.

`-r` Show the routing tables. When `-s` is also present, show routing statistics instead.

`-s` Show per-protocol statistics.

`-t` Show timer information for each interface.

The default display, for active sockets, shows the local and remote addresses, send and receive queue sizes (in bytes), protocol, and the internal state of the protocol. Address formats are of the form *host.port* or *network.port* if a socket's address specifies a network but no specific host address. When known, the host and network addresses are displayed symbolically using the domain name system (DNS). If a symbolic name for an address is unknown, or if the `-n` option is specified, the address is printed numerically. Unspecified, or *wild-card*, addresses and ports appear as '*'.

The interface display provides a table of cumulative statistics regarding packets transferred, errors, and collisions. The network addresses of the interface and the maximum transmission unit ("MTU") are also displayed.

The routing table display indicates the available routes and their status. Each route consists of a destination host or network and a gateway to use in forwarding packets. The flags field shows the state of the route (`U` if up), whether the route is to a gateway (`G`), whether the route was created dynamically by a redirect (`D`), and whether the route has been modified by a redirect (`M`). Direct routes are created for each interface attached to the local host; the *gateway* field for such entries shows the address of the outgoing interface. The *refcnt* field gives the current number of active uses of the route. Connection oriented protocols normally hold on to a single route for the duration of a connection while connectionless protocols obtain a route while sending to the same destination. The *use* field provides a count of the number of packets sent using that route. The *interface* entry indicates the network interface utilized for the route.

When netstat is invoked with a *wait* interval argument, it displays a running count of statistics related to network interfaces. This display consists of a column for the primary interface (the first interface configured at boot time) and a column summarizing information for all interfaces. The primary interface may be replaced with another interface with the `-I` option. The first line of each screen of information contains a summary since the system was last rebooted. Subsequent lines of output show values accumulated over the preceding interval.

See also `ps` in section A.1.37 on page 125, `protocols` in section A.2.5 on page 154, and `services` in section A.2.8 on page 156.

A.1.32 od—Hexadecimal dump

Synopsis:

`od` *file*

`Od` prints the contents of *file* in hexadecimal notation.

A.1.33 outb–Write a byte in I/O space

(Available only on *Express PC*)
Synopsis:

`outb` *hex-address*

Bytes in I/O address space (e.g. device registers) are changed one at a time. Typing a period (‘`.`’) terminates the command.

See also `eb` in section A.1.9 on page 84, `ew` in section A.1.15 on page 87, `el` in section A.1.14 on page 87, `eiob` in section A.1.12 on page 86, `eiow` in section A.1.13 on page 87, and `inb` in section A.1.25 on page 104.

A.1.34 passwd—Change passwords

Synopsis:

`passwd` [*user*]

`Passwd` changes the password of the administrator (`root`) or of the named *user*. First, the user is prompted for the current password. If the current password is correctly typed, a new password is requested. The new password must be entered the same way twice to forestall mistakes.

The new password should be at least six characters long and not purely alphabetic. Its total length must be less than 128 characters. Numbers, upper case letters and punctuation characters are encouraged.

Once the password has been verified, `passwd` writes a disguised (digestified with the MD5 message digest algorithm) version of it to the `passwd` file.

After running `passwd`, the `save` command must be used to store the `passwd` file in flash or on floppy if the new password is to remain in effect across reboot and power outages.

See also `passwd` in section A.2.4 on page 154, `getty` in section A.1.20 on page 96, and `save` in section A.1.45 on page 131.

A.1.35 ping—Probe network hosts

Synopsis:

> `ping` [`-Rdfnqrv`] [`-c` *count*] [`-i` *wait*] [`-l` *preload*] [`-p` *pattern*] [`-s` *packetsize*] *hostname*

`Ping` uses the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway. ECHO_REQUEST datagrams ("pings") have an IP and ICMP header, followed by a "struct timeval" and then an arbitrary number of "pad" bytes used to fill out the packet. The options are as follows:

`-c` *count* Stop after sending (and receiving) *count* ECHO_RESPONSE packets.

`-d` Set the SO_DEBUG option on the socket being used.

`-f` Flood ping. Outputs packets as fast as they come back or one hundred times per second, whichever is more. For every ECHO_REQUEST sent a period ('`.`') is printed, while for every ECHO_REPLY received a backspace is printed. This provides a rapid display of how many packets are being dropped. This can be very hard on a network and should be used with caution.

`-i` *wait* Wait wait seconds between sending each packet. The default is to wait for one second between each packet. This option is incompatible with the `-f` option.

`-l` *preload*
If preload is specified, ping sends that many packets as fast as possible before falling into its normal mode of behavior.

`-n` Numeric output only. No attempt will be made to lookup symbolic names for host addresses.

`-p` *pattern*
You may specify up to 16 "pad" bytes to fill out the packet you send. This is useful for diagnosing data-dependent problems in a network. For example, `-p ff` will cause the sent packet to be filled with all ones.

`-q` Quiet output. Nothing is displayed except the summary lines at startup time and when finished.

`-R` Record route. Includes the RECORD_ROUTE option in the ECHO_REQUEST packet and displays the route buffer on returned packets. Note that the IP header is only large enough for nine such routes. Many hosts ignore or discard this option.

`-r` Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-attached network, an error is returned. This option can be used to ping a local host through an interface that has no route through it (e.g., after the interface was dropped by `gated`).

`-s` *packetsize*
Specifies the number of data bytes to be sent. The default is 56, which translates into 64 ICMP data bytes when combined with the 8 bytes of ICMP header data.

`-v` Verbose output. ICMP packets other than ECHO_RESPONSE that are received are listed.

When using `ping` for fault isolation, it should first be run on the local host, to verify that the local network interface is up and running. Then, hosts and gateways further and further away should be "pinged". Round-trip times and packet loss statistics are computed. If duplicate packets are received, they are not included in the packet loss calculation although the round trip time of these packets is used in calculating the minimum/average/maximum round-trip time numbers. When the specified number of packets have been sent (and received) or if the program is terminated with Control-C, a brief summary is displayed.

This program is intended for use in network testing, measurement and management. Because of the load it can impose on the network, it is unwise to use `ping` during normal operations or from automated scripts.

ICMP Packet Details

An IP header without options is 20 bytes. An ICMP ECHO_REQUEST packet contains an additional 8 bytes worth of ICMP header followed by an arbitrary amount of data. When a *packetsize* is given, this indicated the size of this extra piece of data (the default is 56). Thus the amount of data received inside of an IP packet of type ICMP ECHO_REPLY will always be 8 bytes more than the requested data space (the ICMP header).

If the data space is at least eight bytes large, `ping` uses the first eight bytes of this space to include a timestamp which it uses in the computation of round trip times. If less than eight bytes of pad are specified, no round trip times are given.

Duplicate and Damaged Packets

`Ping` will report duplicate and damaged packets. Duplicate packets should never occur, and seem to be caused by inappropriate link-level retransmissions. Duplicates may occur in many situations and are rarely (if ever) a good sign, although the presence of low levels of duplicates may not always be cause for alarm.

Damaged packets are obviously serious cause for alarm and often indicate broken hardware somewhere in the ping packet's path (in the network or in the hosts).

Trying Different Data Patterns

The (inter)network layer should never treat packets differently depending on the data contained in the data portion. Unfortunately, data-dependent problems have been known to sneak into networks and remain undetected for long periods of time. In many cases the particular pattern that will have problems is something that doesn't have sufficient "transitions", such as all ones or all zeros, or a pattern right at the edge, such as almost all zeros. It isn't necessarily enough to specify a data pattern of all zeros (for example) on the command line because the pattern that is of interest is at the data link level, and the relationship between what you type and what the controllers transmit can be complicated.

This means that if you have a data-dependent problem you will probably have to do a lot of testing to find it. If you are lucky, you may manage to find a file that either can't be sent across your network or that takes much longer to transfer than other similar length files. You can then examine this file for repeated patterns that you can test using the `-p` option of `ping`.

TTL Details

The TTL value of an IP packet represents the maximum number of IP routers that the packet can go through before being thrown away. In current practice you can expect each router in the Internet to decrement the TTL field by exactly one.

The TCP/IP specification states that the TTL field for TCP packets should be set to 60, but many systems use smaller values (4.3 BSD uses 30, 4.2 used 15).

The maximum possible value of this field is 255, and many systems set the TTL field of ICMP ECHO_REQUEST packets to 255. This is why you will find you can `ping` some hosts, but not reach them with `telnet` or `ftp`.

In normal operation `ping` prints the ttl value from the packet it receives. When a remote system receives a ping packet, it can do one of three things with the TTL field in its response:

- Not change it; this is what Berkeley Unix systems did before the 4.3BSD-Tahoe release. In this case the TTL value in the received packet will be 255 minus the number of routers in the round-trip path.
- Set it to 255; this is what current Berkeley Unix systems do. In this case the TTL value in the received packet will be 255 minus the number of routers in the path from the remote system to the pinging host.
- Set it to some other value. Some machines use the same value for ICMP packets that they use for TCP packets, for example either 30 or 60. Others may use completely wild values.

Bugs

Many hosts and gateways ignore the RECORD_ROUTE option.

The maximum IP header length is too small for options like RECORD_ROUTE to be completely useful. There's not much that that can be done about this, however.

Flood pinging is not recommended in general, and flood pinging the broadcast address should only be done under very controlled conditions.

See also `netstat`, `ifconfig`, and `gated`.

A.1.36 pppd—Point-to-Point Protocol daemon

Synopsis:

`pppd` [*options*]

`Pppd` is a daemon process used to manage connections to other hosts using PPP, the Point to Point Protocol, or SLIP, the Serial Line Internet Protocol.

Daemon management options

`auto` Start in 'autocall' mode and detach from the controlling terminal to run as a daemon. Initiate a connection in response to a packet

specified in the 'bringup' category in filter-file. Requires the remote address.

`up` When used with auto, bring the link up immediately rather than waiting for traffic. If the link goes down, attempt to restart it (after the call retry delay timer expires) without waiting for an outbound packet.

`dedicated`
Treat the connection as a dedicated line rather than a demand-dial connection. This option tells `pppd` to never give up on the connection; that is, if the peer tries to shut down the link, go ahead and do so, but then immediately try to reestablish the connection. Similarly, when first trying to connect, `pppd` will not give up after sending a fixed number of Configure-Request messages. Hangup events (LQM failures, loss of Carrier Detect) will still cause the device to be closed, just as with dial-up connections, and the Systems file will then be checked for alternate entries. If none are available, the connection will be reestablished after the call retry delay timer expires. Use a short call retry delay timer on dedicated circuits; something like Any;5-30 should work well. Implies `up`.

`nodetach`
Don't detach from the controlling terminal in 'autocall' mode.

`acct` Print session accounting messages into the system log.

`filter` *filter-file*
Look in *filter-file* for packet filtering and link management information (default if `filter` *filter-file* is not specified: `Filter`).

`debug` *debug-level*
Set the logging verbosity to *debug-level*, chosen from the following table:

0	Daemon start messages
1	Link status messages, calling attempts (the default)
2	Chat script processing, input framing errors
3	LCP, IPCP, PAP and CHAP negotiation
4	LQM status summaries, CHAP re-authentication

5	IP interface changes
6	IP message summary
7	Full LQM messages
8	All PPP messages (without framing)
9	Characters read or written
10	Procedure call messages
11	Internal timers

Each debugging verbosity level also provides the information of all the lower-numbered levels.

Messages at debugging levels 0 and 1 use `syslog` facility `daemon.info`. Messages at debugging levels 2 through 11 use `syslog` facility `daemon.debug`. Fatal errors use `syslog` facility `daemon.crit`.

Communications options

`asyncmap` *async-map*

Set the desired Async Control Character Map to *async-map*, expressed in C-style hexadecimal notation (default 0xA0000).

`noasyncmap`

Disable LCP Async Control Character Map negotiation.

`escape` *odd-character*

In addition to those characters specified in the PPP Async Control Character Map (which can include only 0x00 through 0x1F), also apply the escaping algorithm when transmitting *odd-character*. The value of *odd-character* must be between 0x00 and 0xFF, and cannot be any of 0x5E, 0x7D or 0x7E.

Odd-character can be specified as a decimal number, in C-style hexadecimal notation, or as an ASCII character with optional '`^`' control-character notation. For example, the XON character could be specified as 17, 0x11, or `^Q`.

If a character specified with the escape argument, when transformed into its escaped form, would be the same as a character contained in the peer's negotiated Async Control Character Map, a warning

will be printed in the system log and the character specified on the command line will not be escaped.

If a character specified with the escape argument, when transformed into its escaped form, would be the same as a character specified in another escape argument on the daemon's command line, `pppd` will print an error message and exit.

device Communicate over the named device (default is the controlling tty).

comm-speed
Set communications rate to *comm-speed* bits per second.

`sync` (Not available on the *Express PC*)

Run the serial port in synchronous mode, rather than the default asynchronous mode. Can also be specified in `Devices`, as described in section A.3.3 on page 168.

`ignore-cd`
Ignore the state of the CD (Carrier Detect, also called DCD, Data Carrier Detect) signal. This is useful when the cabling or communications equipment (DCE) don't properly support CD.

`rtscts` Set the async line to use out-of-band EIA RS-232-D 'hardware' (RTS/CTS) flow control. (The default is to use no flow control.) For an outbound connection, this may be specified either in `Devices` or on the `pppd` command line.

`crtscts`
A synonym for `rtscts`.

`xonxoff`
Set the async line to use in-band ('software') flow control, using the characters DC3 (`^S`, XOFF, ASCII 0x13) to stop the flow and DC1 (`^Q`, XON, ASCII 0x11) to resume. (The default is to use no flow control.) For an outbound connection, this may be specified either in `Devices` or on the `pppd` command line.

Link Management Options

`nooptions`
Disable all LCP and IPCP options

`noaccomp`
Disable HDLC Address and Control Field compression.

`noprotcomp`
Disable LCP Protocol Field Compression.

`slip` Use RFC 1055 SLIP packet framing rather than PPP packet framing. Disables all option negotiation, and implies `noasyncmap`, `noipaddress`, `vjslots 16`, `novjcid`, `nomagic`, `nomru`, and `mru 1006`. Implies `vjcomp` if peer sends a header-compressed TCP packet.

`extra-slip-end`
When running in SLIP mode, prepend a SLIP packet framing character (0xC0) to each frame before transmission, even if this frame immediately follows the previous frame. This is useful if the peer's SLIP implements only the optional framing style suggested in the second paragraph in the PROTOCOL section on the second page of RFC 1055 , and cannot receive adjacent frames separated by only one END character (0xC0). This behavior has only been observed in the Hewlett-Packard HP 4995A LanProbe II/Ethernet. By default, `pppd` transmits only one framing character between adjacent SLIP frames.

`nomagic`
Disable LCP Magic Number negotiation.

`mru` *mru-size*
Set LCP Maximum Receive Unit value to *mru-size* for negotiation. The default is 1500 for PPP and 1006 for SLIP.

`nomru` Disable LCP Maximum Receive Unit negotiation, and use 1500 for our interface.

`active` Begin LCP negotiations as soon as the hardware layer is ready (the default).

`passive`
Begin LCP negotiations only after receiving a well-formed HDLC packet.

`timeout` *restart-time*
Set the LCP, IPCP, CCP, PAP, and CHAP option negotiation restart timers to *restart-time* (default 3 seconds).

`max-configure` *configure-attempts*
: Set the number of Configure-Request packets to be sent without receiving a valid Configure-Ack, Configure-Nak or Configure-Reject, before assuming that the peer is unable to respond. (default 10)

`max-terminate` *terminate-attempts*
: Set the number of Terminate-Request packets to be sent without receiving a Terminate-Ack before assuming that the peer is unable to respond. (default 2)

`max-failure` *failure-attempts*
: Set the number of Configure-Nak packets to be sent without sending a Configure-Ack before assuming that configuration is not converging. Any further Configure-Nak packets are converted to Configure-Reject packets. (default 10)

`lqrinterval` *time*
: Send Link-Quality-Reports or Echo-Requests every *time* seconds (default 10 seconds). If the peer responds with a Protocol-Reject, send LCP Echo-Requests every *time* seconds instead, and use the received LCP Echo-Replies for link status policy decisions.

`lqthreshold` *min/per*
: Set a minimum standard for link quality by considering the connection to have failed if fewer than *min* out of the last *per* LQRs we sent have been responded to by the peer (default `1/5`).

`echolqm`
: Use LCP Echo-Requests rather than standard Link-Quality-Report messages for link quality assessment and policy decisions. The peer can override this if it actively tries to configure Link Quality Monitoring unless the `nolqm` parameter is also specified.

`nolqm` Don't send or recognize Link-Quality-Report messages. If echolqm is also specified, Echo-Request messages will be used to detect link failures.

`idle` *idle-time*
: Shut down the link when *idle-time* seconds pass without receiving or transmitting a packet specified in the `keepup` category in the filter file (default is to never shut down).

IP Options

local **:** *remote*
The address of this machine, followed by the expected address for the remote machine. Can be specified either as symbolic names or as literal IP addresses, if their addresses cannot be discovered locally without using the PPP link.

Both addresses are optional, but a colon by itself is not valid, and the remote address is required when running as a daemon in 'autocall' mode. If only '*local* **:** ' is specified when receiving an incoming call, the remote address will normally be discovered during IPCP IP-Address negotiations.

`netmask` *subnet-mask*
Set the subnet mask of the interface to *subnet-mask*, expressed either in C-style hexadecimal (e.g. 0xffffff00) or in decimal dotted-quad notation (e.g. 255.255.255.0). The default subnet mask will be appropriate for the network (class A, B, or C), assuming no subnetting.

`noipaddress`
Disable IPCP IP-Address negotiation.

`vjcomp` Enable RFC 1144 'VJ' Van Jacobson TCP header compression negotiation with 16 slots and slot ID compression (this is the default with PPP framing). 'VJ' compression is enabled by default for async connections, and disabled by default for sync connections.

`novjcomp`
Disable RFC 1144 'VJ' Van Jacobson TCP header compression (this is the default with SLIP framing, until the peer sends a header-compressed TCP packet).

`vjslots` *vj-slots*
Set the number of VJ compression slots (min 3, max 32, default 16).

`novjcid`
Disable VJ compression slot ID compression (enabled by default).

`rfc1172-vj`
Backwards compatibility with older PPP implementations (4-byte VJ configuration option), but with the correct option negotiation value of 0x002d.

`rfc1172-typo-vj`
: Backwards compatibility with older PPP implementations (4-byte VJ configuration option) that conform to the typographical error in RFC 1172 section 5.2 (Compression-Type value 0x0037).

`rfc1172-addresses`
: Backwards compatibility with older PPP implementations that conform to RFC 1172 section 5.1 (IP-Addresses, IPCP configuration option 1) and not with the newer RFC 1332 (IP-Address, IPCP configuration option 3), but that respond with something besides a Configure-Reject when they receive an IPCP Configure-Request containing an option 3.

Authentication Options

`nopap`
: Don't allow PAP authentication (default: allow PAP but don't require it).

`nochap`
: Don't allow CHAP authentication (default: allow CHAP but don't require it).

`requireauth`
: Require either PAP or CHAP authentication.

`requirechap`
: Require CHAP authentication as described in RFC 1334 .

`oldchap`
: Interoperate with CHAP implementations (e.g. Morning Star PPP version 1.3) that conform with certain draft versions of the CHAP protocol specification.

`olderchap`
: Interoperate with CHAP implementations that conform with certain older draft versions of the CHAP protocol specification.

`rechap` *interval*
: Challenge the peer to re-authenticate itself every *interval* seconds.

`name` *identifier*
: Provide the identifier used during PAP or CHAP negotiation. This option is necessary if the PPP peer requires authentication. The default value is the value set with the `hostname` command.

Encryption Options

`gw-crypt` *key-file*
: Encrypt traffic between the pairs of hosts or networks specified in the designated *key-file*.

  Encryption is not available in products exported from the USA.

Compression Options

`compress`
: Offer all supported compression types when negotiating.

compress-pred1
: Accept any, but prefer Predictor-1 compression.

compress-stac
: Accept any, but prefer Stac LZS compression.

nopred1 Never use Predictor-1 compression.

nostac Never use Stac LZS compression

Status Messages

Status information is sent to the system log by each running `pppd` daemon. Each line printed consists of a message preceded by the date, the time, and the process ID number of the daemon writing the message. The quantity and verbosity of messages are controlled with the `debug` option and with the `log` filter (see `Filter`).

Messages at debug levels 0 and 1 use `syslog` facility `daemon.info`, and messages above debug level 1 use `syslog` facility `daemon.debug`. Fatal errors use `syslog` facility `daemon.crit`.

Each packet that brings up the link (at debug level 1 or more), each packet that matches the `log` filter (at any debug level), or any packet when the debug level is 6 or more sends a one-line description of the packet to the system log. The first item of the message is the protocol (tcp, udp, icmp, or a numeric protocol value). For ICMP packets, the keyword `icmp` is followed by the ICMP message type and sub code, separated by slashes. After the protocol comes an IP address and optionally a TCP or UDP port number, followed by an arrow indicating whether the packet was sent (`->`) or received (`<-`), followed by another address and port number, followed by the length of the packet in bytes before VJ TCP header compression, followed by

zero or more keywords. For transmitted packets, the first IP address is the source address, while for received packets, the first IP address is the destination address. Well known TCP and UCP port numbers will be replaced by the equivalent name from the `services` file. The keywords and their meanings are:

`frag` the packet is a middle or later part of a fragmented IP frame

`syn` the packet has the TCP SYN bit set and the ACK bit off

`fin` the packet has the TCP FIN bit set

`bringup`
: the transmitted packet matches the bringup filter and is bringing up the link

`!keepup`
: the packet has been rejected by the keepup filter

`!pass` the packet has been rejected by the pass filter

`dial failed`
: the packet was dropped because `pppd` is waiting for the call retry timer to expire

`(c)` the received packet is VJ TCP header compressed

`(u)` the received packet is VJ TCP header uncompressed

`(e)` the sent or received packet was encrypted or decrypted

For example, the following system log message

```
9/6-14:06:26-17 tcp 63.1.6.3/1050 -> 17.9.1.9/ftp 44 syn
```

indicates that at 2:06:26 PM on September 6, process ID 17 sent a 44-byte TCP packet with the SYN bit set from port 1050 on 63.1.6.3 to the FTP port on 17.9.1.9.

Signal Handling

Upon reception of the following signals, `pppd` re-reads the `Filter` and `Key` files then takes the indicated actions:

SIGKILL
: Don't use this. Since `pppd` won't be able to shut down gracefully, it could leave your serial and IP interfaces in some unknown state. Use SIGTERM instead so that `pppd` will shut down cleanly and leave the system in a well-defined state.

SIGINT Disconnect gracefully from an active session. If in 'autocall' mode, reset the call retry delay timer and call retry backoff interval. If 'up' was specified, attempt to re-establish the link. Exit if not in 'autocall' mode.

SIGHUP
Disconnect abruptly from an active session. If 'up' was specified, attempt to re-establish the link. Exit if not in 'autocall' mode.

SIGTERM
Disconnect gracefully from an active session, clean up the state of any serial and IP interfaces that are open, then exit.

SIGUSR1
Increment the verbosity level for debugging information.

SIGUSR2
Reset the debugging verbosity level to the base value (1 unless debug 0 was supplied on the command line).

SIGALRM
Take no action except to re-read the filter and key files.

Example

To run a pair of daemons on 'oursystem', one maintaining a constant link with 'backbonesystem' and the other prepared to initiate outbound calls to a neighboring machine named 'theirsystem', add the following to rc.boot:

```
pppd oursystem:backbonesystem auto up
pppd oursystem:theirsystem auto idle 120
```

To allow a PPP implementation running on 'theirsystem' to dial into 'oursystem', insert the following into `passwd` on 'oursystem':

```
Pthem::Their PPP:Login
```

where Login is a shell script that looks something like

`pppd` *hostname*`:`

and where *hostname* should be replaced by the hostname of the router.

Recommendations

Use host names when starting `pppd` only if they are known locally. If a PPP connection to a DNS server would be required to resolve a host name, use its literal IP address instead.

Standards Conformance

The `pppd` daemon implements the IETF Proposed Standard Point-to-Point Protocol and many of its options and extensions, in conformance with RFC 1548 , RFC 1549 , RFC 1332 , RFC 1333 , RFC 1334 , and RFC 1144 . It can be configured to be conformant with earlier specifications of the PPP protocol, as described in RFC 1134 , RFC 1171 , and RFC 1172 . It implements the nonstandard SLIP protocol as described in RFC 1055 and RFC 1144 .

See also `Auth` in section A.3.1 on page 160, `Devices` in section A.3.3 on page 167, `Dialers` in section A.3.4 on page 170, `Filter` in section A.3.5 on page 172, `Keys` in section A.3.7 on page 181, `Systems` in section A.3.2 on page 161, the Filter discussion in section 4.5.3 on page 53, RFC 1548 , RFC 1549 , RFC 1332 , RFC 1333 , RFC 1334 , RFC 1144 , RFC 1171 , RFC 1172 , RFC 1055 , and PPP Compression Draft .

A.1.37 ps—Display the status of current processes

Synopsis:

```
ps [-w] [-l]
```

`Ps` displays a one-line status report about each running process. If the `-w` option is given, the entire command line of each running program is printed; otherwise, each line will be cut off to fit within the 80 columns of a standard Hollerith punch card. The different columns printed are:

`pid` The unique process ID of each process.

`ppid` The pid of our parent process.

`pgrp` The process group to which this process belongs. Certain signals (such as SIGHUP) generated by physical or logical events are sent to an entire process group rather than to an individual process.

`st` The process run status; either `S` if the process is sleeping, `R` for a process that is running or waiting to be run, or `Z` for a process that is exiting.

`tty` The controlling tty of a process group, if any. The `tty` part is not included, so that `tty2` is displayed as `2`, and `ttyp0` is displayed as `p0`.

`args` The command name, optionally followed by command-specific status within brackets, followed by the command arguments.

If the `-l` option is given, the following additional columns will be included in each report:

`wkup` The number of times the process has been awakened by another process or by a kernel event.

`stack alloc`
The amount of space allocated for stack.

`stack used`
The amount of the stack ever used.

`mem alloc`
The amount of dynamic memory currently in use.

`wchan` The event or device for which a sleeping process is waiting.

A.1.38 rdate—Set system date from a remote host

Synopsis:

`rdate` [`-d`] *hostname* ...
`rdate -b` [`-t` *interval*] *hostname* …

`Rdate` sets the local date and time from the *hostname* given as an argument. If the `-d` option is specified, the time is not set, but only compared and the difference printed.

If the `-b` option is specified, `rdate` will detach and run in the background, polling the named hosts every *interval* seconds and setting the time. If `-b` is specified but not `-t`, `rdate` will run in the background until the time has been set, and then exit.

If more than one *hostname* is given, `rdate` will poll them in sequence until it finds one that answers. It will stop looking for more hosts after a successful poll.

`Rdate` uses the UDP/Time protocol, as described in RFC 868 .

See also `tz` in section A.2.10 on page 158.

A.1.39 reboot—Restart the router

Synopsis:

`reboot` [`-f`] [*filename*]

If a *filename* is specified, load its contents into router memory and begin executing it. If no *filename* is given, use the `mse` file from flash memory or the floppy.

If any command is in the process of changing flash memory, `reboot` will print

```
Flash is in use; waiting...
```

and wait until the flash is no longer being written. If the `-f` option is given, `reboot` will not wait.

A.1.40 repack-flash—Rewrite all files in the flash

(Not available on *Express PC*)

Synopsis:

`repack-flash` [`-f`] [*files*]

Erase the flash memory, then write back the specified *files* and any others that were already in flash. Because of its size, this is often the only way to replace the saved `mse` file. If the `-f` option is not given, `repack-flash` will print the names of any changed files that will overwrite saved files, and ask whether they should be overwritten.

`Repack-flash` recovers any space lost by `unsave` commands.

See also `memory` and `unsave`.

A.1.41 restore—Restore files from flash or floppy

Synopsis:

`restore` [*files*]

`Restore` loads *files* from flash memory or floppy, overwriting the current memory-resident versions, if any.

A.1.42 ripquery—Query RIP gateways

Synopsis:

`ripquery` [`-1`] [`-2`] [`-n`] [`-p`] [`-r`] [`-v`] [`-w` *time*] *gateway* ...

`Ripquery` is used to request all routes known by a RIP gateway by sending a RIP request or POLL command. The routing information in any routing packets returned is displayed numerically and symbolically. `Ripquery` is intended to be used as a tool for debugging gateways, not for network management. SNMP is the preferred protocol for network management.

`Ripquery` by default uses the RIP POLL command, which is an undocumented extension to the RIP specification supported by routed on SunOS 3.x and later and by gated 1.4 and later. The RIP POLL command is preferred over the RIP REQUEST command because it is not subject to Split Horizon and/or Poisoned Reverse. See the RIP RFC (RFC 1058) for more information.
Options:

`-1` Send the query as a version 1 packet.

`-2` Send the query as a version 2 packet (default).

`-n` Prevents the address of the responding host from being looked up to determine the symbolic name.

`-p` Uses the RIP POLL command to request information from the routing table. This is the default, but is an undocumented extension supported only by some versions of SunOS 3.x and later versions of gated. If there is no response to the RIP POLL command, the RIP REQUEST command is tried.

`-r` Use the RIP REQUEST command to request information from the gateway's routing table. Unlike the RIP POLL command, all gateways should support the RIP REQUEST. If there is no response to the RIP REQUEST command, the RIP POLL command is tried.

`-v` Version information about `ripquery` is displayed before querying the gateways.

`-w` Specifies the time in seconds to wait for the initial response from a gateway. The default value is 5 seconds.

See also `gated` in section A.1.18 on page 93, `gated.conf` in section A.4.1 on page 184, and RFC 1058 (Routing Information Protocol).

A.1.43 rm—Remove files from memory

Synopsis:

`rm` *files*

Remove the named *files* from memory.

See also `unsave` in section A.1.57 on page 150 and `ls` in section A.1.28 on page A.1.28.

A.1.44 route—Manipulate the routing tables

Synopsis:

`route` [`-n`] [`-q`] [`-v`] *command* [[*modifiers*] *args*]

`Route` is used to manually manipulate the network routing tables. It is usually not needed if the system routing table management daemon, `gated`, is running. Options supported by route:

`-n` Prevent attempts to print host and network names symbolically when reporting actions.

`-v` (verbose) Print additional details.

`-q` Suppress all output.

Commands accepted by route:

`add` Add a route.

`flush` Remove all routes.

`delete` Delete a specific route.

`change` Change aspects of a route (such as its gateway).

`get` Lookup and display the route for a destination.

`monitor`
Continuously report any changes to the routing information base, routing lookup misses, or suspected network partionings.

The monitor command has the syntax

`route` [`-n`] `monitor`

The flush command has the syntax

`route` [`-n`] `flush`

The other commands have the following syntax:

`route` [`-n`] *command* [`-net`|`-host`] *destination gateway*

where *destination* is the destination host or network, or the special keyword *default*, and *gateway* is the next-hop gateway to which packets should be addressed. Routes to a particular host are distinguished from those to a network by interpreting the Internet address associated with destination. The optional keywords `-net` and `-host` force the destination to be interpreted as a network or a host, respectively. Otherwise, if the destination has a "local address part" of INADDR_ANY, or if the destination is the symbolic name of a network, then the route is assumed to be to a network; otherwise, it is presumed to be a route to a host.

For example, 128.32 is interpreted as -host 128.0.0.32; 128.32.130 is interpreted as `-host` 128.32.0.130; `-net` 128.32 is interpreted as 128.32.0.0; and `-net` 128.32.130 is interpreted as 128.32.130.0.

If the route is via an interface rather than via a gateway, the `-interface` modifier should be specified; the gateway given is the address of this host on the common network, indicating the *interface* to be used for transmission.

The optional `-netmask` qualifier is intended to achieve the effect of an OSI ESIS redirect with the netmask option. One specifies an additional ensuing address parameter (to be interpreted as a network mask). The implicit network mask can be overridden by making sure this option follows the destination parameter.

The optional modifiers `-rtt`, `-rttvar`, `-sendpipe`, `-recvpipe`, `-mtu`, `-hopcount`, `-expire`, and `-ssthresh` provide initial values to metrics maintained in the routing entry. These may be individually locked by preceding each such modifier to be locked by the `-lock` meta-modifier, or one can specify that all ensuing metrics may be locked by the `-lockrest` meta-modifier.

In a change or add command where the destination and gateway are not sufficient to specify the route, the `-ifp` or `-ifa` modifiers may be used to determine the interface or interface address.

All symbolic names specified for a destination or gateway are looked up as host names.

If the flush command is specified, route will "flush" the routing tables of all gateway entries.

Diagnostics

`add` [`host`|`network`] *destination*`:` `gateway` *gateway* `flags` *value*

The specified route is being added to the tables. If the gateway address used was not the primary address of the gateway, the gateway address is printed numerically as well as symbolically.

`delete` [`host`|`network`] *destination*`:` `gateway` *gateway* `flags` *value*

As above, but when deleting an entry.

destination gateway `done`

When the flush command is specified, each routing table entry deleted is indicated with a message of this form.

`Network is unreachable`

An attempt to add a route failed because the gateway listed was not on a directly-connected network. The next-hop gateway must be given.

`not in table`

A delete operation was attempted for an entry which wasn't present in the tables.

`routing table overflow`

An add operation was attempted, but the system was low on resources and was unable to allocate memory to create the new entry.

See also `gated` in section A.1.18 on page 93, and `netstat` in section A.1.31 on page 107.

A.1.45 save—Save files to flash or floppy

Synopsis:

`save` [*files*]

Save the named *files* to flash memory or floppy.

A.1.46 sgetty—Enable incoming synchronous calls

Synopsis:

`sgetty` *tty-port speed* [*options*] *command* `command...`

Options are:

`riwait` Wait for RI to rise before raising DTR (default)

`cdwait` Wait for CD to rise before raising DTR

`nowait` Don't wait for RI or CD

`respawn`
After the specified `command` exits, start again

`debug` Print debugging messages to the system log

command `command ...`
Execute `command` (the remainder of the command line) when CD rises (or immediately, if you also specified `nowait`), supplying the rest of the `sgetty` command line as arguments

`Sgetty` monitors the modem control signals on a *tty-port* (depending on which of the `riwait`, `cdwait`, or `nowait` options are given), opens and initializes the device, and then runs the `command` specified after the *command* argument, providing the specified arguments. If the `respawn` option is not specified, `sgetty` then exits; otherwise, it waits for the spawned command to exit and then repeats the entire process.

Example

(This would be on one line in `rc.boot`, but it's broken onto several lines here because of the constraints of the printed page.)

```
sgetty tty0 respawn cdwait command pppd
192.0.2.1:192.0.2.2 tty0 sync debug 2 nodetach
requireauth
```

See also `getty` in section A.1.20 on page 96.

A.1.47 sh—Command interpreter

Synopsis:

```
sh
```

`Sh` is the command interpreter; it executes commands read from a terminal session or from a file. Each task running on the *Express* router is started from a command line. Each command line is parsed into several tokens separated by horizontal white space (spaces or tabs), the first token of which is the command name. The command name must either be one of the commands described in this manual, or it can be the name of a text file containing more commands.

When a command is run, the command line tokens are made into a list of arguments and given to the newly created task. Commands interpret the arguments as described in this manual; arguments passed to a text file of commands are ignored.

A token beginning with a '`#`' and everything following is considered a comment and is ignored. The meaning of special characters can be removed by preceding them with a backslash ('`\`') or by enclosing them in double quotes.

Each task has a *standard input*, a *standard output*, and a *standard error output*. When a command is run from a terminal session, keystrokes from the terminal are sent to the *standard input*, and data written to the *standard output* and the *standard error output* are sent to the terminal. Commands run at boot time from the `rc.boot` file have no *standard input*, and their outputs are sent to the system log.

These *input*, *output*, and *error* data streams are inherited from the parent task, but it is possible to direct them elsewhere by the use of special command line notations. One or more of the following may appear on a command line following the last argument:

< *filename*
: Read *standard input* from *filename*.

> *filename*
: Write *standard output* to *filename*.

>> *filename*
: Append *standard output* to the end of *filename*.

2> *filename*
: Write the *standard error output* to *filename*.

`2>&1` Send the *standard error output* to the same place as the *standard output*.

A.1.48 sleep—suspend execution for a specified interval

Synopsis:

`sleep` *seconds*

`sleep` suspends execution for a specified number of seconds. After *seconds* seconds have passed, the next command (e.g. in `rc.boot`) is executed.

A.1.49 snmpd—SNMP agent daemon

Synopsis:

`snmpd` [`-p` *port*] [`-d` *loglevel*]

`Snmpd` is an SNMP application which listens for and responds to network management queries and commands from logically remote network management stations. `Snmpd` accesses the supported variables in the instrumentation of the protocol layers and the kernel. These parameters are made accessible via the Simple Network Management Protocol (SNMP). The `-p` option allows specifying a UDP *port* on which `snmpd` will listen, overriding the default of `161`. If an alternate *port* is specified with `-p`, SNMP traps will be sent on port *port*+`1`; otherwise they will be sent on port `162`.

Debugging and log information is sent to the system log. The `-d` option is used to set the verbosity level; higher values of *loglevel* generate more information.

When `snmpd` is started, it reads the `snmpd.config` file, which contains descriptions of installation-dependent values plus SNMP *community* definitions.

A.1.50 syslogd—System message log daemon

Synopsis:

`syslogd` [`-f` *config-file*] [`-m` *mark-interval*] [`-d`]

`Syslogd` sends system log messages to log files and/or other machines as specified by its configuration file. The options are as follows:

`-f` Specify the pathname of an alternate configuration file; the default is `syslog.conf`.

`-m` Select the number of minutes between "mark" messages; the default is `20`.

`-d` Debugging: show progress parsing `syslog.conf`

`Syslogd` reads its configuration file when it starts up and whenever it receives a hangup signal. For information on the format of the configuration file, see `syslog.conf` in section A.2.9 on page 157.

`Syslogd` creates the file `syslog.pid`, and stores its process id there. This can be used to kill or reconfigure `syslogd`.

See also `syslog.conf` in section A.2.9 on page 157, and `services` in section A.2.8 on page 156.

A.1.51 telnet—User interface to the TELNET protocol

Synopsis:

`telnet` [`-l` *user*] [`-echar`] [`-b`] [*host-name* [*port*]]

Description:

The `telnet` command is used to communicate with another host using the TELNET protocol.

Options:

`-l` *user* When connecting to the remote system, if the remote system understands the `ENVIRON` option, then *user* will be sent to the remote system as the value for the variable `USER`. This option implies the `a` option. This option may also be used with the `open` command.

`-e`*char* Sets the initial `telnet` escape character to *char*. If *char* is ommitted, then the escape character defaults to `^]`.

`-b` Connect in binary (8-bit transparent) mode by enabling or disabling the `TELNET BINARY` option on both input and output.

host-name
Connect to *host-name*, which may be specified by name or IP address

port Connect to TCP port *port* on the specified *host-name*. The *port* may be specified by name, if the name is specified in the `services` file. If the name is not specified in the `services` file, the *port* must be specified numerically. If neither a number nor a name is specified on the `telnet` command line, the default `telnet` port (23) is used.

While connected to a remote host, `telnet` command mode may be entered by typing the `telnet` escape character (initially `^]`).

If `telnet` is invoked without the *host-hame* argument, it enters command mode, indicated by its prompt

```
telnet>
```

When in command mode, the normal terminal editing conventions are available. In this mode, it accepts and executes the commands listed below. If telnet is invoked with arguments, it performs an `open` command with those arguments.

The following `telnet` commands are available. Only enough of each command to uniquely identify it need be typed (this is also true for arguments to the `set`, `toggle`, `unset`, and `display` commands).

`?` Display commands

`close` Close a `TELNET` session and return to command mode.

`display` *argument ...*
Displays all, or some, of the `set` and `toggle` values (see below).

`open` *host user port*
Open a connection to the named host. If no port number is specified, `telnet` will attempt to contact a `TELNET` server at the default port. The host specification may be either a host name (see `hosts` in section A.2.1 on page 152, and `resolv.conf` in section A.2.7 on page 155) or an Internet address specified in the dot notation.

If no *user* is specified, `telnet` supplies no user name to the remote system. The `l` option may be used to specify the user name to be passed to the remote system via the `ENVIRON` option. When connecting to a non-standard port, `telnet` omits any automatic initiation of `TELNET` options. When the port number is preceeded by a minus sign, the inital option negotiation is done.

`send` *char ...*
Sends one or more special character sequences to the remote host. The following are the arguments which may be specified (more than one argument may be specified at a time):

Valid arguments are:

`abort` Sends the `TELNET ABORT` (ABORT processes) sequence.

`ao` `TELNET AO` (Abort Output) sequence, which should cause the remote system to flush all output *from* the remote system *to* the user's terminal.

`ayt` Sends the `TELNET AYT` (Are You There) sequence, to which the remote system may or may not choose to respond.

`brk` Sends the `TELNET BRK` (Break) sequence, which may have significance to the remote system.

`ec` Sends the `TELNET EC` (Erase Character) sequence, which should cause the remote system to erase the last character entered.

`el` Sends the `TELNET EL` (Erase Line) sequence, which should cause the remote system to erase the line currently being entered.

`ga` Sends the `TELNET EL` (Go Ahead) sequence.

`ip` Sends the `TELNET IP` (Interrupt Process) sequence.

`eof` Sends the `TELNET EOF` (End Of File) sequence.

`eor` Sends the `TELNET EOR` (End of Record) sequence.

`escape` Sends the current `TELNET` escape character (initially `^]`).

`ga` Sends the `TELNET GA` (Go Ahead) sequence, which likely has no significance to the remote system.

`getstatus`
If the remote side supports the `TELNET STATUS` command, `getstatus` will send the subnegotiation to request that the server send its current option status.

`ip` Sends the `TELNET IP` (Interrupt Process) sequence, which should cause the remote system to abort the currently running process.

`nop` Sends the `TELNET NOP` (No OPeration) sequence.

`susp` Sends the `TELNET SUSP` (SUSPend process) sequence.

`synch` Sends the `TELNET SYNCH` sequence. This sequence causes the remote system to discard all previously typed (but not yet read) input. This sequence is sent

as TCP urgent data (and may not work if the remote system is a 4.2 BSD system – if it doesn't work, a lower case `r` may be echoed on the terminal).

`?` display help information

`echo` This is the value (initially `^E`) which, when in *line by line* mode, toggles between doing local echoing of entered characters (for normal processing), and suppressing echoing of entered characters (for entering, say, a password).

`eof` A `TELNET EC` sequence (see `send` *ec* above) is sent to the remote system. The initial value for the erase character is taken to be the terminal's `erase` character.

`escape` This is the `TELNET` escape character (initially `^[`) which causes entry into `telnet` command mode (when connected to a remote system).

`quit` Exit `telnet`.

`set` *variable value ...*

The `set` command will set any one of a number of `TELNET` variables to a specific value or to TRUE. The special value `off` turns off the function associated with the variable, this is equivalent to using the `unset` command. The values of variables may be interrogated with the `display` command. The variables which may be set or unset, but not toggled, are listed here. In addition, any of the variables for the `toggle` command may be explicitly set or unset using the `set` and `unset` commands.

Valid arguments are:

escape character to escape back to telnet command mode

flushoutput char
character to cause an Abort Output (requires *localchars*)

interrupt char
character to cause an Interrupt Process (requires *localchars*)

quit char character to cause an Abort Process (requires *localchars*)

eof char character to cause an End Of File (requires *localchars*)

autoflush enable flushing of output when sending interrupt characters

autosynch enable automatic sending of interrupt characters in urgent mode

binary enable sending and receiving of binary data

inbinary enable receiving of binary data

outbinary enable sending of binary data

crlf enable sending carriage returns as telnet <CR><LF>

crmod enable mapping of received carriage returns

localchars enable local recognition of certain control characters

options enable viewing of options processing (debugging)

? display help information

`unset` *variable ...*

The `unset` command will disable or set to FALSE any of the specified functions. The values of variables may be interrogated with the `display` command. The variables which may be set or unset, but not toggled, are listed here. In addition, any of the variables for the `toggle` command may be explicitly set or unset using the `set` and `unset` commands.

Valid arguments are:

escape no character escapes back to telnet command mode

flushoutput no character causes an Abort Output

interrupt no character causes an Interrupt Process

quit no character causes an Abort Process

eof no character causes an End Of File

autoflush disable flushing of output when sending interrupt characters

autosynch disable automatic sending of interrupt characters in urgent mode

binary disable sending and receiving of binary data

inbinary	disable receiving of binary data
outbinary	disable sending of binary data
crlf	disable sending carriage returns as telnet <CR><LF>
crmod	disable mapping of received carriage returns
localchars	disable local recognition of certain control characters
options	disable viewing of options processing (debugging)
`?`	display help information

`status`

`toggle` *variable ...*

Toggle (between TRUE and FALSE) various flags that control how `telnet` responds to events. These flags may be set explicitly to TRUE or FALSE using the `set` and `unset` commands listed above. More than one argument may be specified. The state of these flags may be interrogated with the `display` command.

Valid arguments are:

autoflush	toggle flushing of output when sending interrupt characters
autosynch	toggle automatic sending of interrupt characters in urgent mode
binary	toggle sending and receiving of binary data
inbinary	toggle receiving of binary data
outbinary	toggle sending of binary data
crlf	toggle sending carriage returns as telnet <CR><LF>
crmod	toggle mapping of received carriage returns
localchars	toggle local recognition of certain control characters
options	toggle viewing of options processing (debugging)
?	display help information

See also `telnetd` in section A.1.52 on page 141, and `services` in section A.2.8 on page 156.

A.1.52 telnetd—TELNET protocol server

Synopsis:

`telnetd` [`-debug` [*port*]] [`-debug`] [`-h`] [`-n`] [`-D` *options*] [`-D` *report*] [`-D` *netdata*] [`-D` *ptydata*]

`Telnetd` is a server which supports the Internet standard TELNET virtual terminal protocol. `Telnetd` is invoked by `inetd`, normally for requests to connect to the TELNET port as indicated by the `services` file. The `-debug` option may be used to start up telnetd manually instead of through `inetd`. If started up this way, *port* may be specified to run `telnetd` on an alternate TCP port number.

The `-D` option may be used for debugging purposes. This allows telnet to print out debugging information to the connection, allowing the user to see what `telnetd` is doing. There are several modifiers: `options` prints information about the negotiation of TELNET options, `report` prints the options information, plus some additional information about what processing is going on, `netdata` displays the data stream received by `telnetd`, and `ptydata` displays data written to the pty.

The `-h` option prevents `telnetd` from printing the banner at connection time.

The `-n` option prevents keepalives from being sent on the socket.

`Telnetd` operates by allocating a pseudo-terminal device for a client, then creating a `getty` process which has the slave side of the pseudo-terminal as stdin, stdout, and stderr. `Telnetd` manipulates the master side of the pseudo-terminal, implementing the TELNET protocol and passing characters between the remote client and the `getty` process.

When a TELNET session is started up, `telnetd` sends TELNET options to the client side indicating a willingness to do remote echo of characters, to suppress go ahead, to do remote flow control, and to receive various information from the remote client.

`Telnetd` is willing to do: echo, binary, suppress go ahead, and timing mark. `Telnetd` is willing to have the remote client do: linemode, binary, toggle flow control, and suppress go ahead.

Bugs

Some TELNET commands are only partially implemented.

Because of bugs in the original 4.2 BSD telnet, `telnetd` performs some dubious protocol exchanges to try to discover if the remote client is, in fact, a 4.2 BSD telnet.

Binary mode has no common interpretation except between similar operating systems.

`Telnetd` never sends TELNET go ahead commands.

A.1.53 tftp—Trivial file transfer program

Synopsis:

`tftp` [*host*]

`Tftp` is the user interface to the Internet TFTP (Trivial File Transfer Protocol), which allows users to transfer files to and from a remote machine. The remote *host* may be specified on the command line, in which case `tftp` uses *host* as the default host for future transfers (see the `connect` command below).

Commands

Once `tftp` is running, it issues the prompt and recognizes the following commands:

`?` *command-name ...*
Print help information.

`ascii` Shorthand for `mode ascii`

`binary` Shorthand for `mode binary`

`connect`
host-name [*port*] Set the host (and optionally port) for transfers. Note that the TFTP protocol, unlike the FTP protocol, does not maintain connections between transfers; thus, the `connect` command does not actually create a connection, but merely remembers what host is to be used for transfers. You do not have to use the `connect` command; the remote host can be specified as part of the `get` or `put` commands.

`get` *filename*

`get` *remotename localname*

`get` *file1 file2 …fileN*
: Get a file or set of files from the specified sources. Source can be in one of two forms: a *filename* on the remote *host*, if the *host* has already been specified, or a string of the form *host*`:`*filename* to specify both a host and filename at the same time. If the latter form is used, the last *hostname* specified becomes the default for future transfers.

`mode` *transfer-mode*
: Set the mode for transfers; *transfer-mode* may be one of `ascii` or `binary`. The default is `ascii`.

`put` *file*

`put` *localfile remotefile*

`put` *file1 file2 ... fileN remote-directory*
: Put a file or set of files to the specified remote file or directory. The destination can be in one of two forms: a *filename* on the remote *host*, if the *host* has already been specified, or a string of the form *host*`:`*filename* to specify both a *host* and *filename* at the same time. If the latter form is used, the *hostname* specified becomes the default for future transfers. If the *remote-directory* form is used, the remote *host* is assumed to be a UNIX machine.

`quit`
: Exit `tftp`.

`rexmt` *retransmission-timeout*
: Set the per-packet retransmission timeout, in seconds.

`status`
: Show current status.

`timeout` *total-transmission-timeout*
: Set the total transmission timeout, in seconds.

`trace`
: Toggle packet tracing.

`verbose`
: Toggle verbose mode.

Bugs

Because there is no user-login or validation within the TFTP protocol, the remote site will probably have some sort of file-access restrictions in place. The exact methods are specific to each site and therefore difficult to document here.

A.1.54 tftp-dump–Register host:filename as crash dump recipient

Synopsis:

`tftp-dump` *host* **:** *filename*

`Tftp-dump` specifies that, in the event of a catastrophic failure, the *Express* will save the contents of its memory across the local Ethernet to a file named *filename* on the host named *host*. The location of that file in *host*'s filesystem will be determined by the configuration of the tftp service on *host*.

Contact the Morning Star Technologies Technical Support staff for instructions regarding what to do with the resulting dump file.

A.1.55 traceroute—Print the route to a network host

Synopsis:

`traceroute` [`-m` *max-ttl*] [`-n`] [`-p` *port*] [`-q` *nqueries*] [`-r`] [`-s` *src-addr*] [`-t` *tos*] [`-w` *waittime*] *host* [*packetsize*]

The Internet is a large and complex aggregation of network hardware, connected together by gateways. Tracking the route one's packets follow (or finding the miscreant gateway that's discarding your packets) can be difficult. `Traceroute` utilizes the IP protocol 'time to live' field and attempts to elicit an ICMP TIME_EXCEEDED response from each gateway along the path to some host.

The only mandatory parameter is the destination host name or IP number. The default probe datagram length is 38 bytes, but this may be increased by specifying a packet size (in bytes) after the destination host name.

Other options are:

`-m` *max-ttl*
: Set the max time-to-live (max number of hops) used in outgoing probe packets. The default is 30 hops (the same default used for TCP connections).

`-n`
: Print hop addresses numerically rather than symbolically and numerically (saves a nameserver address-to-name lookup for each gateway found on the path).

`-p` *port*
: Set the base UDP port number used in probes (default is 33434). `Traceroute` hopes that nothing is listening on UDP ports $base$ to $base + nhops - 1$ at the destination host (so an ICMP

PORT_UNREACHABLE message will be returned to terminate the route tracing). If something is listening on a port in the default range, this option can be used to pick an unused port range.

`-q` *nqueries*
Set the number of probes per "ttl" to nqueries (default is three probes).

`-r` Bypass the normal routing tables and send directly to a host on an attached network. If the host is not on a directly-attached network, an error is returned. This option can be used to ping a local host through an interface that has no route through it (e.g., after the interface was dropped by `gated`).

`-s` *src-addr*
Use the following IP address (which must be given as an IP number, not a hostname) as the source address in outgoing probe packets. On hosts with more than one IP address, this option can be used to force the source address to be something other than the IP address of the interface the probe packet is sent on. If the IP address is not one of this machine's interface addresses, an error is returned and nothing is sent.

`-t` *tos* Set the type-of-service in probe packets to the following value (default zero). The value must be a decimal integer in the range 0 to 255. This option can be used to see if different types-of-service result in different paths. Not all values of TOS are legal or meaningful—see the IP spec for definitions. Useful values are probably '`-t 16`' (low delay) and '`-t 8`' (high throughput).

`-v` Verbose output. Received ICMP packets other than TIME_EXCEEDED and UNREACHABLEs are listed.

`-w` Set the time (in seconds) to wait for a response to a probe (default 3 sec.).

This program attempts to trace the route an IP packet would follow to some internet host by launching UDP probe packets with a small ttl (time to live) then listening for an ICMP "time exceeded" reply from a gateway. We start our probes with a ttl of one and increase by one until we get an ICMP "port unreachable" (which means we got to "host") or hit a max (which defaults to 30 hops and can be changed with the `-m` flag). Three probes (changed with `-q` flag) are sent at each ttl setting and a line is printed showing the ttl, address of the gateway and round

trip time of each probe. If the probe answers come from different gateways, the address of each responding system will be printed. If there is no response within a 3 sec. timeout interval (changed with the `-w` flag), a "*" is printed for that probe.

We don't want the destination host to process the UDP probe packets so the destination port is set to an unlikely value (if someone on the destination is using that value, it can be changed with the `-p` flag).

A sample use and output might be:

```
# traceroute nis.nsf.net.
traceroute to nis.nsf.net (35.1.1.48), 30 hops max
1 helios.ee.lbl.gov (128.3.112.1) 19 ms 19 ms 0 ms
2 lilac-dmc.Berkeley.EDU (128.32.216.1) 39 ms 39 ms 19 ms
3 lilac-dmc.Berkeley.EDU (128.32.216.1) 39 ms 39 ms 19 ms
4 ccngw-ner-cc.Berkeley.EDU (128.32.136.23) 39 ms 40 ms 39 ms
5 ccn-nerif22.Berkeley.EDU (128.32.168.22) 39 ms 39 ms 39 ms
6 128.32.197.4 (128.32.197.4) 40 ms 59 ms 59 ms
7 131.119.2.5 (131.119.2.5) 59 ms 59 ms 59 ms
8 129.140.70.13 (129.140.70.13) 99 ms 99 ms 80 ms
9 129.140.71.6 (129.140.71.6) 139 ms 239 ms 319 ms
10 129.140.81.7 (129.140.81.7) 220 ms 199 ms 199 ms
11 nic.merit.edu (35.1.1.48) 239 ms 239 ms 239 ms
```

Note that lines 2 and 3 are the same. This is due to a buggy kernel on the 2nd hop system—lbl-csam.arpa—that forwards packets with a zero ttl (a bug in the distributed version of 4.3 BSD). Note that you have to guess what path the packets are taking cross-country since the NSFNet (129.140) doesn't supply address-to-name translations for its NSSes.

A more interesting example is:

```
# traceroute allspice.lcs.mit.edu.
traceroute to allspice.lcs.mit.edu (18.26.0.115), 30 hops max
1 helios.ee.lbl.gov (128.3.112.1) 0 ms 0 ms 0 ms
2 lilac-dmc.Berkeley.EDU (128.32.216.1) 19 ms 19 ms 19 ms
3 lilac-dmc.Berkeley.EDU (128.32.216.1) 39 ms 19 ms 19 ms
4 ccngw-ner-cc.Berkeley.EDU (128.32.136.23) 19 ms 39 ms 39 ms
5 ccn-nerif22.Berkeley.EDU (128.32.168.22) 20 ms 39 ms 39 ms
6 128.32.197.4 (128.32.197.4) 59 ms 119 ms 39 ms
7 131.119.2.5 (131.119.2.5) 59 ms 59 ms 39 ms
8 129.140.70.13 (129.140.70.13) 80 ms 79 ms 99 ms
9 129.140.71.6 (129.140.71.6) 139 ms 139 ms 159 ms
```

```
10 129.140.81.7 (129.140.81.7) 199 ms 180 ms 300 ms
11 129.140.72.17 (129.140.72.17) 300 ms 239 ms 239 ms
12 * * *
13 128.121.54.72 (128.121.54.72) 259 ms 499 ms 279 ms
14 * * *
15 * * *
16 * * *
17 * * *
18 ALLSPICE.LCS.MIT.EDU (18.26.0.115) 339 ms 279 ms 279 ms
```

Note that the gateways 12, 14, 15, 16 and 17 hops away either don't send ICMP "time exceeded" messages or they send them with a ttl too small to reach us. 14–17 are running the MIT C Gateway code that doesn't send "time exceeded" messages. It's anyone's guess what's going on with 12.

The silent gateway 12 in the above may be the result of a bug in the 4.[23] BSD network code (and its derivatives): 4.x (for $x \leq 3$) sends an unreachable message using whatever ttl remains in the original datagram. Since, for gateways, the remaining ttl is zero, the ICMP "time exceeded" is guaranteed to not make it back to us. The behavior of this bug is slightly more interesting when it appears on the destination system:

```
1 helios.ee.lbl.gov (128.3.112.1) 0 ms 0 ms 0 ms
2 lilac-dmc.Berkeley.EDU (128.32.216.1) 39 ms 19 ms 39 ms
3 lilac-dmc.Berkeley.EDU (128.32.216.1) 19 ms 39 ms 19 ms
4 ccngw-ner-cc.Berkeley.EDU (128.32.136.23) 39 ms 40 ms 19 ms
5 ccn-nerif35.Berkeley.EDU (128.32.168.35) 39 ms 39 ms 39 ms
6 csgw.Berkeley.EDU (128.32.133.254) 39 ms 59 ms 39 ms
7 * * *
8 * * *
9 * * *
10 * * *
11 * * *
12 * * *
13 rip.Berkeley.EDU (128.32.131.22) 59 ms ! 39 ms ! 39 ms !
```

Notice that there are 12 "gateways" (13 is the final destination) and exactly the last half of them are "missing". What's really happening is that RIP (a Sun-3 running SunOS 3.5) is using the ttl from our arriving datagram as the ttl in its ICMP reply. So, the reply will time out on the return path (with no notice sent to anyone since ICMP's aren't sent for ICMP's) until we probe with a ttl that's at least twice the

path length. I.e., rip is really only 7 hops away. A reply that returns with a ttl of 1 is a clue this problem exists. `Traceroute` prints a "!" after the time if the ttl is ≤ 1. Since vendors ship a lot of obsolete or non-standard software, expect to see this problem frequently and/or take care picking the target host of your probes. Other possible annotations after the time are !H, !N, !P (got a host, network or protocol unreachable, respectively), !S or !F (source route failed or fragmentation needed—neither of these should ever occur). If almost all the probes result in some kind of unreachable, `traceroute` will give up and exit.

This program is intended for use in network testing, measurement and management. It should be used primarily for manual fault isolation. Because of the load it could impose on the network, it is unwise to use `traceroute` during normal operations or from automated scripts.

See also `netstat` and `ping`.

A.1.56 tz—Set the time zone

Synopsis:

`tz` *timezone-file*
`tz` *timezone-string*

If `tz` is given a filename, or if the *Express* router is booted with the `tz` file in place, the time zone will be set as defined in the file. The format of time zone files is described in the section on `tz` on page 158.

If `tz` is given a time zone string and if the `tzposixrules` file is present, the time zone is set according to rules in the IEEE 1003.1 POSIX specification. The *timezone-string* can contain the following information:

- A string of characters designating the standard time zone name. None of the characters in the name be a digit, comma (‘,’), plus sign (‘+’), or minus sign (‘−’).
- The amount of time past Universal Coordinated Time (UTC) for standard time in the format

  [+|−]*hh*[:*mm*[:*ss*]]

  with *hh*, *mm*, and *ss* being hours, minutes, and seconds, respectively. Positive times are west of the prime meridian; negative numbers are east.
- A string of characters designating the daylight savings time zone name.

- The amount of time past GMT for daylight savings time in the same format as above.
- A comma (‘`,`’) or semicolon (‘`;`’) followed by a pair of date specifiers separated by a comma. The two date specifiers define the time and date that daylight savings time begins (the first specifier) and ends. Each specifier consists of a date, optionally followed by a slash (‘`/`’) and a time. The time is in the format

  hh[`:`*mm*[:*ss*]]

  with *hh*, *mm*, and *ss* as before. The date is in one of the following formats:

  `J`*n* A one-based Julian day ($1 \leq n \leq 365$).

  n A zero-based Julian day ($0 \leq n \leq 364$). In either of the two Julian day formats, leap days (February 29) are never counted; that is, February 28 (J59) is immediately followed by March 1 (J60) even in leap years.

  `W`*n*`.`*d* Day *d* ($0 \leq d \leq 6$) of week *n* ($1 \leq n \leq 53$). Sunday is the first day of the week (0). If *d* is omitted, Sunday is assumed. The fifty-third week of the year is always the last week containing day *d* whether there are actually fifty-three weeks containing day *d* or not.

  `M`*m*`.`*n*`.`*d* Day *d* of week *n* ($1 \leq n \leq 5$) of month *m* ($1 \leq m \leq 12$). The fifth week of the month is always the last week containing day *d* whether there are actually five weeks containing day *d* or not.

  Starting and ending times are relative to the alternate time zone. If the alternate time zone start and end dates and the time are not provided, the days for the United States that year will be used and the time will be 2 AM. If the start and end dates are provided but the time is not provided, the time will be midnight.

For example, the Eastern time zone in the U.S. can be set either with

```
# tz EST5EDT
```

or with

```
# tz EST5:00:00EDT4:00:00;117/2:00:00,299/2:00:00
```

A.1.57 unsave—Remove files from flash or floppy

Synopsis:

`unsave` [*files*]

`Unsave` removes a file from flash memory or floppy by marking it as being deleted. The space is not recovered; use `repack-flash` to recover space from `unsave`d files.

See also `repack-flash` in section A.1.40 on page 127, `rm` in section A.1.43 on page 129, and `ls` in section A.1.28 on page A.1.28.

A.1.58 uptime—Print information about the last boot

Synopsis:

`uptime`

`Uptime` prints the time and date the router was booted, as well as the elapsed time since booting.

A.1.59 version—Print version and other information

Synopsis:

`version`

Express and *Express 2E*

On the *Express* and *Express 2E*, the `version` command prints the following information:

- The version number of the running software and when it was built
- The MAC address of the Ethernet interface
- The CPU type and hardware version
- The Ethernet interface type(s), hardware version, and MAC addresses
- The amount of installed DRAM memory
- The amount and type of installed flash memory
- The reason and time of the last *panic* if the current boot followed a crash.

Express PC

On the *Express PC*, the `version` command prints the following information:

- The version number of the running software and when it was built
- The MAC address of the Ethernet interface
- The CPU type and hardware version
- The Ethernet interface name(s) and MAC address(es)
- The amount of installed DRAM memory
- The hardware configuration, including each device's
  - hardware class (e.g. tty, diskette, Ethernet)
  - configuration name (e.g. tty0, fd0, ed0)
  - hardware type or capacity (e.g. 16550A, 1.44M, N32000 16 bit)
  - bus address (e.g. 0x3f8, 0x3f0, 0x300)
  - irq (e.g. 4, 2, 10)
  - bus type (e.g. isa)
- The reason and time of the last *panic* if the current boot followed a crash.

A.2 System Configuration File Formats

A.2.1 hosts

The `hosts` file contains information regarding the known hosts on the TCP/IP internet. For each host a single line should be present with the following information:

```
Internet-address official-host-name aliases
```

Items are separated by any number of blanks and/or TAB characters. A "`#`" indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file. This file is normally created from the official host data base maintained at the Network Information Center (NIC), though local changes may be required to bring it up to date regarding unofficial aliases and/or unknown hosts.

Network addresses are specified in the conventional "`.`" notation using the *inet_addr*() routine from the Internet address manipulation library, *inet*(3N). Host names may contain any printable character other than an upper case character, a field delimiter, NEWLINE, or comment character. Here is a typical line from the `hosts` file:

```
192.9.1.20        gaia                          # John Smith
```

See also `resolv.conf` in section A.2.7 on page 155.

A.2.2 inetd.conf

The `inetd.conf` file, when read by the `inetd` daemon, defines actions to take in response to incoming UDP or TCP connection requests. The file format is described in `inetd` in section A.1.26 on page 104.

A.2.3 jumpers

(Available only on *Express PC*)

Since the generic PC hardware used in the *Express PC* has no jumpers in the same sense as the custom hardware in the *Express* and the *Express 2E*, the *Express PC* software instead gets its configuration "jumper" information from a file on its floppy disc. The file named `jumpers` contains a string of `0` and `1` digits,

numbered from right to left with the bit corresponding to J0 on the right and the bit corresponding to J7 on the left. In each case, if a digit is `1` it indicates that the "jumper" is "in", and if a digit is `0` it indicates that the "jumper" is "out".

Like other files on the *Express PC*, `jumpers` can be edited on the *Express PC* using the router's native editor, or via FTP from another system, or by using a text editor on any MS-DOS system. The *Express PC* sees the file as being named `jumpers`, but a MS-DOS system will see it as `JUMPERS.` Note the trailing period ('`.`'), because of the limitations of filenames on MS-DOS.

The "jumpers" have these meanings:

J0 Not used

J1 Not used

J2 Disable security (default: `1`) If this jumper is `0`, an *Express PC* router will boot with the following configuration:

```
console tty0
version
ifconfig lo0 127.1
ifconfig ed0 192.0.2.1
inetd
getty tty0 9600 nowait respawn
ifconfig ed0 rarp
```

Additionally, password checking will be disabled for the *root* account.

This jumper makes it possible to recover from severe misconfiguration.

J3 Not used

J4 Print stack trace (default: `1`)

If `0`, the router will print a stack dump to the console after a crash.

J5 Not used

J6 Not used

J7 Not used

A.2.4 passwd

The `passwd` file associates each login account with its password and with a command to be run upon successful login. Each line in the file contains four fields separated by colons (‘`:`’):

username
: The login account name.

password
: The password after being digestified by the MD5 message digest algorithm.

comment
: Ignored

command
: The command to execute after being successfully authenticated by `getty`

If the passwd file contains an entry with username `PPP` and a null password, such as

```
PPP:::pppd myhostname:  rtscts idle 60 requirechap
```

then if `getty` sees an incoming PPP frame, it will respond by invoking the specified command. In this example, the shell is `pppd` with the indicated arguments. This relieves the calling system of the need to negotiate through a login/password chat script before commencing LCP negotiations. For security, be sure to employ link-level authentication!

See also `getty` in section A.1.20 on page 96, and the `passwd` command in section A.1.34 on page 110.

A.2.5 protocols

The `protocols` file contains information regarding the known protocols used in the Internet. For each protocol a single line should be present in the following format:

official-protocol-name protocol-number aliases

Items are separated by any number of blanks and/or tab characters. A “`#`” indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file.

Protocol names may contain any printable character other than a field delimiter, newline, or comment character.

A.2.6 rc.boot

Normally, commands from the `rc.boot` file are run when the *Express* router is booted. This is the proper place to put commands that start daemons, set the time, add static routes or ARP entries, or set the hostname. There are several other similar files, however, that are run under special conditions:

```
if jumper J2 is in
        create and run rc.default
 else
        if rebooting from a downloaded file and rc.test exists
                run rc.test
        else
                if rebooting from flash and rc.flash exists
                        run rc.flash
                if rc.boot exists
                        run rc.boot
                if we haven't run an rc file yet
                        create and run rc.default
```

See also `reboot` in section A.1.39 on page 127.

A.2.7 resolv.conf

The `resolv.conf` resolver configuration file contains information that is read by the Domain Name System name resolution routines the first time they are invoked by a process. The file is designed to be human readable and contains a list of keywords with values that provide various types of resolver information.

The different configuration options are:

nameserver

Internet address (in dot notation) of a name server that the resolver should query. Up to 3 name servers may be listed, one per keyword. If there are multiple servers, the resolver library queries them in the order listed. The algorithm used is to try a name server, and if the query times out, try the next, until out of name servers, then repeat trying all the name servers until a maximum number of retries are made.

domain Local domain name. Most queries for names within this domain can use short names relative to the local domain.

search Search list for host name lookup. The search list is normally determined from the local domain name; by default, it begins with the local domain name, then successive parent domains that have at least two components in their names. This may be changed by listing the desired domain search path following the search keyword with spaces or tabs separating the names. Most resolver queries will be attempted using each component of the search path in turn until a match is found. Note that this process may be slow and will generate a lot of network traffic if the servers for the listed domains are not local, and that queries will time out if no server is available for one of the domains.

The search list is currently limited to six domains with a total of 256 characters.

The domain and search keywords are mutually exclusive. If more than one instance of these keywords is present, the last instance will override.

The keyword and value must appear on a single line, and the keyword (e.g. `nameserver`) must start the line. The value follows the keyword, separated by white space.

In the absence of `resolv.conf`, the *Express* router will consult the static `hosts` file instead.

See also `hosts` in section A.2.1 on page 152, and `host` in section A.1.22 on page 98.

A.2.8 services

The `services` file contains information regarding the known services available on Internet-connected hosts. For each service a single line should be present in the following format:

official-service-name port-number/protocol-name aliases

Items are separated by any number of blanks and/or tab characters. The port number and protocol name are considered a single item; a "`/`" is used to separate the port and protocol (e.g. "`512/tcp`"). A "`#`" indicates the beginning of a comment; subsequent characters up to the end of the line are not interpreted by the routines which search the file.

Service names may contain any printable character other than a field delimiter, newline, or comment character.

See also `inetd` in section A.1.26 on page 104.

A.2.9 syslog.conf

The `syslog.conf` file is the configuration file for the `syslogd` daemon, which controls the distribution of system log messages. Each line has two fields: the *selector* field which specifies the types of messages and priorities to which the line applies, and an *action* field which specifies the action to be taken if a message `syslogd` receives matches the selection criteria. The *selector* field is separated from the *action* field by one or more tab characters.

Selectors are encoded as a *facility*, a period (‘`.`’), and a *level*, with no intervening white space. Both the *facility* and the *level* are case insensitive.

The *facility* describes the part of the system generating the message, and is one of the following keywords: `auth`, `authpriv`, `cron`, `daemon`, `kern`, `mark`, `syslog`, `user`, and `local0` through `local7`. The *Express* router itself emits messages with *facility* `kern`, and all its programs (e.g. `pppd`, `frd`, etc.) emit messages with *facility* `daemon`.

The *level* describes the severity of the message, and is a keyword from the following ordered list (higher to lower): `emerg`, `alert`, `crit`, `err`, `warning`, `notice`, and `debug`.

If a received message matches the specified facility and is of the specified level (or a higher level), the action specified in the action field will be taken.

Multiple selectors may be specified for a single action by separating them with semicolon (‘`;`’) characters. It is important to note, however, that each selector can modify the ones preceding it.

Multiple facilities may be specified for a single level by separating them with comma (‘`,`’) characters.

An asterisk (‘`*`’) can be used to specify all facilities or all levels.

The special facility `mark` receives a message at priority `info` every 20 minutes (see `syslogd`). This is not enabled by a facility field containing an asterisk.

The special level `none` disables a particular facility.

The *action* field of each line specifies the action to be taken when the *selector* field selects a message. There are two forms:

- A pathname (beginning with a leading slash). Selected messages are appended to the file. This is not recommended, since all files are held in router

memory, and a continuously growing file could use up all available router memory.

- A hostname (preceded by an at (‘@’) sign). Selected messages are forwarded to the `syslogd` program on the named host.

Blank lines and lines whose first non-blank character is a hash (‘#’) character are ignored.

Here is a very simple configuration that logs all system log messages to another host:

```
# Log all messages to server
*.* @137.175.2.11
```

Be careful! The *selector* field is separated from the *action* field by one or more *tab* characters, not by *space* characters.

Bugs

The effects of multiple selectors are sometimes not intuitive. For example, `daemon.crit,*.err` will select `daemon` facility messages at the level of `err` or higher, not at the level of `crit` or higher.

See also `syslogd` in section A.1.50 on page 134.

A.2.10 tz

The `tz` and `tzposixrules` time zone information files begin with several bytes reserved for future use, followed by four four-byte values written in a “standard” byte order (the high-order byte of the value is written first). These values are, in order:

tzh_ttisstdcnt
: The number of standard/wall indicators stored in the file.

tzh_leapcnt
: The number of leap seconds for which data is stored in the file.

tzh_timecnt
: The number of “transition times” for which data is stored in the file.

tzh_typecnt
: The number of “local time types” for which data is stored in the file (must not be zero).

tzh_charcnt
The number of characters of "time zone abbreviation strings" stored in the file.

The above header is followed by *tzh_timecnt* four-byte values, sorted in ascending order. These values are written in "standard" byte order. Each is used as a transition time at which the rules for computing local time change. Next come *tzh_timecnt* one-byte values of; each one tells which of the different types of "local time" types described in the file is associated with the same-indexed transition time. These values serve as indices into an array of *ttinfo* structures that appears next in the file; these structures are defined as follows:

```
struct ttinfo {
        long tt_gmtoff;
        int tt_isdst;
        unsigned int tt_abbrind;
};
```

Each structure is written as a four-byte value for *tt_gmtoff* of type long, in a standard byte order, followed by a one-byte value for *tt_isdst* and a one-byte value for *tt_abbrind*. In each structure, *tt_gmtoff* gives the number of seconds to be added to GMT, *tt_isdst* tells whether *tm_isdst* should be set and *tt_abbrind* serves as an index into the array of time zone abbreviation characters that follow the ttinfo structure(s) in the file.

Then there are *tzh_leapcnt* pairs of four-byte values, written in standard byte order; the first value of each pair gives the time at which a leap second occurs; the second gives the total number of leap seconds to be applied after the given time. The pairs of values are sorted in ascending order by time.

Finally there are *tzh_ttisstdcnt* standard/wall indicators, each stored as a one-byte value; they tell whether the transition times associated with local time types were specified as standard time or wall clock time, and are used when a time zone file is used in handling POSIX-style time zone environment variables.

The first standard-time *ttinfo* structure in the file (or simply the first *ttinfo* structure in the absence of a standard-time structure) is used if either *tzh_timecnt* is zero or the current time is less than the first transition time recorded in the file.

Preconfigured `tz` files for a wide variety of time zones are available from the anonymous ftp server at *ftp.morningstar.com*. If the `tzposixrules` file is present, the `tz` command can be used to set time zone information using POSIX-style rules.

A.3 Link Management and Configuration File Formats

A.3.1 Auth

The `Auth` file contains values used by `ppp`'s implementation of the link-level authentication protocols, CHAP (Challenge Handshake Authentication Protocol) and PAP (Password Authentication Protocol). This implementation of both CHAP and PAP conforms to RFC 1334. Interoperability with draft versions of CHAP is provided through the use of `pppd`'s `oldchap` and `olderchap` command line options.

CHAP is a stronger authentication mechanism and should be used whenever possible, in preference over PAP.

Format

Each authentication specification is on its own single line of up to 1023 characters. Longer lines are truncated. Comments begin with a '`#`' and extend to the end of the line; blank lines, or lines beginning with a '`#`', are ignored. Fields are separated by horizontal white space (blanks or tabs).

If `pppd` is using CHAP authentication, the first word on the `Auth` line must match the peer's *Name* as received in a CHAP Challenge or Response packet and the second word is used for the *Secret*. If `pppd` is using PAP authentication, the first word on the line must match the *Peer-ID* in a transmitted or received PAP Authenticate-Request packet and the second word is used for the *Password*. The default value used for the *Name* in transmitted CHAP packets or for the *Peer-ID* in transmitted PAP packets is the `hostname` of the *Express* router.

In the midst of the *Name*/*Peer-ID* and *Secret*/*Password* strings, `^`*x* is translated into the appropriate control character before matching, and `^`*xxx* represents the character corresponding to the octal number *xxx*. Other special sequences are:

`\s` Matches a space character (ASCII 0x20)

`\t` Matches a horizontal tab character (ASCII 0x09)

`\n` Matches a line feed character (ASCII 0x0a)

`\r` Matches a carriage return character (ASCII 0x0d)

The fields have the following meaning:

name The Name field of a sent or received CHAP Challenge or Response message, or the Peer-ID field of a sent or received PAP Authenticate-Request message. For transmitted packets, this is the hostname unless overridden by the `pppd name` option.

secret The secret word that the peer also knows.

optional address restrictions
A set of zero or more patterns restricting the addresses that we will allow to be used with the named peer. Patterns are separated by spaces or tabs and are parsed from left to right. Each pattern may begin with an exclamation mark to indicate that the following pattern should not be allowed. The rest of the pattern consists of digits and periods, and optionally a leading or trailing asterisk, which will match anything. If none of the patterns match, then the address will be allowed if the last pattern began with an exclamation point, and will be disallowed otherwise.

Example

The following `Auth` provides `pppd` with a secret for use when a peer claims to be other-host, robin, or 'Jack's machine'.

```
#
#  Auth - PPP authentication name/secret file
#  Format:
#name              secret      optional address restrictions
other-host         secret-key  !137.175.9.2 137.175.9.*/0xffffff00
robin              dK3ig8G8hs  137.175.11.4
Jack's\smachine  I\sam\sa\sjelly\sdonut.
```

A.3.2 Systems

The `Systems` file tells `pppd` how to connect with neighboring systems via PPP or SLIP.

Format

Entries are one to a line; blank lines are ignored. Comments begin with a '#' and extend to the end of the line. Upper/lower case distinctions are ignored in

hostname specifications, but are significant elsewhere. Fields on a line are separated by horizontal white space (blanks or tabs). If a chat script ends with a backslash ('\'), the next line is considered a continuation of the chat script. Continuations may only occur in the midst of a chat script. Lines longer than 1023 characters are truncated.

Each entry must contain six fields, in the following order:

name The hostname or IP address of the destination machine, which should be resolvable locally.

when A string that indicates the days of the week and the times of day when the system can be called (for example, `MoTuTh0800-1740`). The day portion may be a list containing any of `Su`, `Mo`, `Tu`, `We`, `Th`, `Fr` or `Sa`. The day may also be `Wk` for any weekday (same as `MoTuWeThFr`) or `Any` for any day (same as `SuMoTuWeThFrSa`).

You can indicate hours in a range (for example, `0800-1230`). If you do not specify a time, calls will be allowed at any time.

Note that a time range that spans `0000` is permitted. For example, `0800-0600` means that all times are allowed except times between 6 AM and 8 AM.

Multiple day specifications that are separated by a vertical bar (|) are allowed. For example, `Any0100-0600|Sa|Su` means that the system can be called any day between 1 AM and 6 AM or any time on Saturday and Sunday.

The entire (sequence of) days and times may be followed by a semicolon (';') and up to three decimal numbers separated by hyphens:

one If only one number follows the semicolon, it is used as the redial delay, which is the initial time (in seconds) before a failed call will be retried. For example, `Any;60` means call any time, but wait at least 60 seconds after a failure has occurred before trying to call again. If a call retry fails, `pppd` will double the delay before trying again. If no initial retry delay is specified, 10 seconds is assumed.

two If two numbers follow the semicolon, the second number is used as the maximum redial delay, which is the maximum time (in seconds) to delay before retrying a

call. The retry time will double with each unsuccessful call until it reaches this value, after which the call will be retried every time the maximum number of seconds passes. If no maximum retry delay is specified, 3600 seconds is assumed.

three If three numbers follow the semicolon, the first is used as the callback delay, the second as the redial delay, and the third as the maximum redial delay. The callback delay is the time (in seconds) to wait before attempting to re-establish a previously active connection that ended because of an abrupt line disconnection (a Hangup or SIGHUP event in the log file). The default is to not delay before calling back.

During the delay following an unsuccessful call, any level 7 debugging messages will have the message 'dial failed' appended.

device If set to `ACU`, any device in the `Devices` file with a matching speed may be used. The device's dialer chat script will be executed first, followed by the chat script in the `Systems` file.

If set to the name of a device name (`tty0`, `tty1`, etc.), then there may be an optional corresponding `Direct` entry in the `Devices` file, `Dialers` will not be consulted, and only the `Systems` chat script will be executed.

If set to `tcp`, then it must be followed by a slash, then the hostname or IP address of the system that will serve as the destination of the PPP link, then another slash, then the socket number on which to contact the remote PPP daemon.

speed The speed of the connection. If the device field is `ACU`, the speed field will be string matched against entries in `Devices`. Speeds must either be valid speed numbers or must begin with them (`38400`, `57600`, `19200-PEP`, etc.). If the device field is '`tcp`...', the speed field is optional and is ignored if present.

If your *Express* router is running synchronous PPP with the default external clocking, the modem or CSU/DSU is expected to provide the clock signal and the *speed* field is ignored, except to ensure the match between the corresponding lines in `Systems` and `Devices`.

phone number
: The value to replace the `\T` escape sequence in the dialer script. If the device field names an actual device, the phone number field is optional. If the device field is '`tcp`…', the phone number field is optional, and is ignored if present.

chat script
: A description of the conversation that `pppd` holds with the remote machine.

Chat Script Particulars

A chat script takes the form of a word to expect the remote end to send, followed by a word to send in response. Unless a *send* string ends with `\c`, `pppd` will send a following carriage return character (ASCII 0x0d).

Chat scripts are '*expect send expect send* …' or '*expect-send-expect send* …', where the *send* following the hyphen is executed if the preceding *expect* fails to match received text.

Certain special words may be used in chat script *send* strings to control the behavior of `pppd` as it attempts to dial. Both `ABORT` and `TIMEOUT` must be in the *expect* phase of the chat script.

`ABORT` *abort-string*
: If `pppd` sees *abort-string* while executing the remainder of the chat script, abort the dialing attempt and note the failure in a log message.

`TIMEOUT`
timeout-time
: While executing the current chat script, wait *timeout-time* seconds for an expected response before regarding the dialing attempt as having failed. Writes have a fixed 60-second timeout.

The *expect-send* couplet of `""` `P_`*WORD* sets the line parity accordingly:

`P_ZERO` Transmit characters with the parity bit set to zero (no parity).

`P_ONE` Transmit characters with the parity bit set to one.

`P_EVEN` Transmit characters with even parity. This is the default.

`P_ODD` Transmit characters with odd parity.

In the midst of either an *expect* string or a *send* string, ^*x* gets translated into the appropriate control character, and *x* gets translated into *x*. Other special sequences are:

`\s` Send or receive a space character (ASCII 0x20)

`\t` Send or receive a horizontal tab character (ASCII 0x09)

`\n` Send or receive a line feed character (ASCII 0x0a)

`\r` Send or receive a carriage return character (ASCII 0x0d)

`\\` Send or receive a backslash character (ASCII 0x5c)

`\^` Send or receive a caret character (ASCII 0x5e)

`\`*character*
Send or receive the single character Ctrl-*character* (ASCII 0x00 through 0x1f)

`\`*ddd* Send or receive a character, specified in octal *d*igits

`\p` Pause for .25 second before proceeding (send only)

`\d` Delay for two seconds before proceeding (send only)

`\K` Send a break (.25 second of zero bits)

`\M` Disable hangups (sets the POSIX CLOCAL tty mode bit)

`\m` enable hangups (unsets the CLOCAL bit) (the default)

`\c` Don't append a carriage return character after sending the preceding string (send only)

`\q` Don't print following send strings (e.g. a password) in any debugging or logging output. Subsequent `\q` sequences toggle 'quiet' mode.

`\A` Parse the incoming string as an IP address, written as four decimal numbers separated by periods, and use it for the local end of the point-to-point connection (receive only)

Example

In the example below, we call host *everyone* using a V.32bis/V.42/V.42bis modem with its DTE interface set at 57600 bps, and we are connected to host *someone* via a direct null-modem cable attached to `tty1`, running asynchronous PPP at 38400. We talk to *anyone* via a T1 CSU/DSU attached to `tty0`, the V.35 connector. If that T1 link fails, we will attempt to establish a dialup connection if the modem is available. And we connect with *pseudo-one* via a PPP connection tunneled across a TCP stream to port 77 on *realone.somewhere.com*.

If we are unsuccessful at connecting with *someone* we will try again in two seconds. If that attempt fails, we will wait four seconds before the next attempt; then eight, then sixteen, then thirty-two, then forty seconds. We will continue attempting to contact *someone* every forty seconds. Our retry intervals and maximum backoff values for *everyone* and *anyone* are the default `10-3600`.

The notation `""  ""` means to *expect* nothing, then *send* nothing (followed by a carriage return). The implicit carriage return is often useful for eliciting a response from a remote system.

```
#
#       Systems - PPP systems file
#
everyone Any ACU 57600 5551212 in:--in: Pwe word: \qfoObar
someone Any;2-40 tty2 38400 0 in:--in: Pthem word: \qmeumBle
anyone Any tty0 1536000
anyone Any ACU 57600 5553434 "" "" in:--in: Pno word: \qfroBozZ
pseudo-one Any;2-2 tcp/realone.somewhere.com/57
```

Recommendations

The default retry time and backoff (i.e. `Any;10-3600`) are appropriate for use with dialup connections where the PPP connection must be reestablished as quickly as possible after an interruption but where it is not desirable to continuously redial a host that may be down. A much shorter maximum would be appropriate for a dedicated line between two systems, or where call attempts cost nothing. Moderate call retry times, such as 60 seconds, work well on systems that can establish connections in either direction using dialup modems, to avoid deadlocks waiting for telephone busy signals from each calling the other at the same time. Because of the difference between the behaviors of originating and answering modems, the 60-second clocks will usually start ticking at different times, allowing one side to call the other without interference. Alternatively, different call retry

times may be specified at either end of a link to help keep the two systems from calling each other simultaneously.

If you specify host names, be sure that their addresses are available locally, even with the connection down. If you find that you must bring up a connection to resolve a domain name, consider using that host's IP address (decimal numbers separated by periods) in both `Filter` and `Systems` entries instead.

Automatic failover recovery can be arranged between systems that each have multiple modems, or multiple connection methods. If two systems are connected via a dedicated line (sync or async), that entry should be first in `Systems`, followed by another entry describing an on-demand dial-up connection.

See also `pppd`, `Dialers`, and `Filter`.

A.3.3 Devices

The `Devices` file tells `pppd` what sort of communications facilities are available for its connections via PPP or SLIP. `Pppd` examines it when placing a call to a neighboring machine. If no suitable speed is found, or if all devices associated with that speed are busy, `pppd` will try again later.

Format

Entries are one to a line; blank lines are ignored. Comments begin with a '`#`' and extend to the end of the line. Upper/lower case distinctions are significant. Fields on a line are separated by horizontal white space (blanks or tabs).

Each entry must contain three or more fields, in this order:

dialer Either the string '`Direct`', or the name of the modem dialing chat script (found in `Dialers`) to use with this device.

device The name of the device, chosen from

`tty0`	The V.35 connector
`tty1`	The first DB-25 RS-232 connector
`tty2`	The second DB-25 RS-232 connector

speed The baud rate of the synchronous connection, or a string to be matched against the speed field of entries in `Systems` when the `Systems` device field is set to ACU. Speeds must either be valid async baud-rate numbers (`9600`, `19200`, `38400`, `57600`, and `115200` are most common), or must begin with them (`38400`, `19200-PEP`, etc.),

or must be speeds of which the synchronous hardware is capable (`9600`, `56000`, `64000`, `1536000`, etc.)

If your *Express* router is running synchronous PPP with the default external clocking, the modem or CSU/DSU is expected to provide the clock signal and the *speed* field is ignored, except to ensure the match between the corresponding lines in `Systems` and `Devices`.

optional parameters

Any special handling for this device. Currently supported values include

- asynchronous parameters

  `rtscts` Specifies that the async line be conditioned for out-of-band EIA RS-232-D 'hardware' (RTS/CTS) flow control. The default is to use no flow control. For an outbound connection, this may be specified either in `Devices` or on the `pppd` command line.

  `crtscts`
  a synonym for rtscts

  `xonxoff`
  Specifies that the line be conditioned for in-band ('software') flow control, using the characters DC3 (`^S`, XOFF, ASCII 0x13) to stop the flow and DC1 (`^Q`, XON, ASCII 0x11) to resume. The default is to use no flow control. For an outbound connection, this may be specified either in `Devices` or on the `pppd` command line.

  `ignore-cd`
  Ignore the state of the CD (Carrier Detect, also called DCD, Data Carrier Detect) signal. This is useful when the cabling or communications equipment (DCE) don't properly support CD.

- synchronous parameters

  `sync` Indicates that `pppd` should talk synchronous PPP over this port. Lacking `sync`, `pppd` will talk asynchronous PPP over this port.

  `internal-clocking`
  The *Express* will provide the synchronous clock signal, TxCLK and RxCLK, on the serial connector.

By default, it expects the modem, CSU/DSU or modem eliminator to provide the clock signal. A speed must also be specified. The **Internal clocking** jumpers corresponding to the appropriate device must also be set; see section 2.2.2, *Setting the Jumpers*, on page 18.

`external-clocking`
: The *Express* will expect the modem, CSU/DSU or modem eliminator to provide the synchronous clock signal, TxCLK and RxCLK, on the serial connector. This is the default behavior. The **External clocking** jumpers corresponding to the appropriate device must also be set; see section 2.2.2, *Setting the Jumpers*, on page 18.

`32-bit-fcs`
: The *Express* will calculate 32-bit FCS values for transmitted frames, and check received frames with 32-bit FCS calculations. This is not negotiable at connection establishment time. The 32-bit FCS option is only available with the `sync` option.

`min-flags=minflags`
: The number of additional HDLC flag characters the *Express* should insert between data frames. The default and minimum is 2; the maximum is 16.

`ignore-cd`
: Ignore the state of the CD (Carrier Detect, also called DCD, Data Carrier Detect) signal. This is useful when the cabling or communications equipment (DCE) don't properly support CD.

Example

```
#
#       Devices - PPP devices file
#
#Dialer        device  speed    Optional parameters
USR-Sportster  tty1    57600    rtscts
Direct         tty2    38400    rtscts
Direct         tty0    1536000  sync
```

This *Express* has a US Robotics Sportster 14,400 modem attached to its `tty1` port, a direct async serial line on its `tty2`, and a T1 CSU/DSU attached to the V.35 connector on `tty0`; the CSU/DSU provides the synchronous clock signal.

A.3.4 Dialers

The `Dialers` file describes how to dial each type of modem attached to the *Express* that is to be made available for outbound PPP or SLIP calls. `pppd` examines it when placing a call to a neighboring machine.

When `pppd` selects a line from `Systems` it uses the *speed* field to select an entry in `Devices` from which it uses the *dialer* field to select an entry in `Dialers`. `Pppd` then interprets the 'chat script' field from that dialer description.

Format

Entries are one to a line; blank lines are ignored. Comments begin with a '`#`' and extend to the end of the line. Upper/lower case distinctions in the *dialer* field are significant for matching purposes, as are strings in the chat script. Fields on a line are separated by horizontal white space (blanks or tabs). If a chat script ends with a backslash ('`\`'), the next line is considered a continuation of the chat script. Continuations may only occur in the midst of a chat script. Lines longer than 1023 characters are truncated.

Each entry must contain these fields, in this order:

dialer The name of this dialer, to be matched against the *dialer* field in `Devices`

chat-script
A description of the conversation that `pppd` holds with the modem.

Chat Script Particulars

A chat script takes the form of a space-separated list of *expect-send* pairs. Each pair consists (at minimum) of a field to expect the 'remote' end to send, then a field to send in response. Unless a *send* string ends with '`\c`', `pppd` will follow it by sending a carriage return character (ASCII 0x0d).

Chat scripts are of the form *expect send expect send …* or *expect-send-expect send …*, where the *send* following the hyphen is executed if the preceding *expect* fails to match received text.

Certain special words may be used in the chat script to control the behavior of `pppd` as it attempts to dial. Both `ABORT` and `TIMEOUT` must be in the *expect* phase of the chat script.

`ABORT` *abort-string*
: If `pppd` sees *abort-string* while executing the remainder of the chat script, abort the dialing attempt and note the failure in a log message.

`TIMEOUT` *timeout-time*
: While executing the current chat script, wait *timeout-time* seconds for a response before considering the dialing attempt to have timed out. Writes have a fixed 60-second timeout.

The *expect-send* couplet of `""` `P_`*WORD* sets the line parity accordingly:

`P_ZERO` Transmit characters with the parity bit set to zero (no parity).

`P_ONE` Transmit characters with the parity bit set to one.

`P_EVEN` Transmit characters with even parity. This is the default.

`P_ODD` Transmit characters with odd parity.

In the midst of either an *expect* string or a *send* string, ^*x* gets translated into the appropriate control character, and *x* gets translated into *x*. Other special sequences are:

`\s` Send or receive a space character (ASCII 0x20)

`\t` Send or receive a horizontal tab character (ASCII 0x09)

`\n` Send or receive a line feed character (ASCII 0x0a)

`\r` Send or receive a carriage return character (ASCII 0x0d)

`\\` Send or receive a backslash character (ASCII 0x5c)

`\^` Send or receive a caret character (ASCII 0x5e)

`\`*character*
: Send or receive the single character Ctrl-*character* (ASCII 0x00 through 0x1f)

`\`*ddd* Send or receive a character, specified in octal *d*igits

`\p`	Pause for .25 second before proceeding (send only)
`\d`	Delay for two seconds before proceeding (send only)
`\K`	Send a break (.25 second of zero bits)
`\M`	Disable hangups (sets the POSIX CLOCAL tty mode bit)
`\m`	enable hangups (unsets the CLOCAL bit) (the default)
`\c`	Don't append a carriage return character after sending the preceding string (send only)
`\q`	Don't print following send strings (e.g. a password) in any debugging or logging output. Subsequent `\q` sequences toggle 'quiet' mode.
`\A`	Parse the incoming string as an IP address, written as four decimal numbers separated by periods, and use it for the local end of the point-to-point connection (receive only)

Example

```
#
#       Dialers - PPP dialers file
#
#Dialer Chat script
T1600   ABORT NO\sCARRIER ABORT NO\sDIALTONE ABORT BUSY \
        ABORT RRING\r\n\r\nRRING\r\n\r\nRRING \
        ABORT ERROR TIMEOUT 5 "" AT OK-AT-OK \
        ATS111=0DT\T TIMEOUT 30 CONNECT
```

A.3.5 Filter

The `Filter` file describes how on-demand PPP and SLIP links are to be managed, and how packet filtering (selective packet forwarding) should be done on PPP, SLIP, and frame relay connections. By default, any type of packet causes a dynamic link (if down) to be brought up (connected to its remote end); any packet is allowed to traverse any link; and any packet is sufficient to reset the idle timer on dynamic links, expiration of which would cause it to be shut down. This combination is not always appropriate behavior, so the `Filter` file allows individual control based

on the packet type, source, destination, and other features. These selection criteria may be specified for any of the three phases of operation: bringing up a dynamic link, passing packets on a link, and shutting down a dynamic link due to inactivity. Packet logging detail may also be selected using the same criteria.

Format

Comments begin with a '#' and extend to the end of the line; blank lines, or lines beginning with a '#', are ignored. Upper/lower case distinctions are ignored in hostname specifications, but are significant elsewhere. Fields are separated by horizontal or vertical white space (blanks or tabs or newlines).

A line beginning with a hostname or IP address or the special word `default` marks the beginning of a new set of filtering specifications.

- If the packet was received over, or is about to be transmitted over, a PPP or SLIP link, then the hostname or IP address in the first column of the `Filter` file refers to the peer (system or router or terminal server) at the remote end of the point-to-point (PPP or SLIP) link.

  The filtering specifications will be applied to any packet crossing the point-to-point link connecting this host to the peer named by that initial hostname or IP address.

- If the packet was received over, or is about to be transmitted over, a Frame Relay or Ethernet network, then the hostname or IP address in the first column of the `Filter` file refers to the *Express* router's own IP address on its Frame Relay or Ethernet network interface.

  The filtering specifications will be applied to any packet received over or about to be transmitted over the Frame Relay or Ethernet interface named by that initial hostname or IP address.

The hostname or IP address in the first column of the `Filter` file, and associated with either the link peer or the local Frame Relay interface, is unrelated to the source or destination IP address of any packet crossing the link. If either the link peer's address, or the *Express* router's Frame Relay interface's address, doesn't match any name or address specified in the first column of `Filter` file, the filter specification following the special word `default` will be used.

If a newline is followed by white space, that line is a continuation of the filtering specification already in progress.

There are four keywords to describe the actions taken in response to a particular packet:

`bringup`
: Describes those packets that will initiate a dynamic connection. Packets of this sort also must qualify to *pass* across the link, either by being explicitly mentioned or by inclusion in a larger class in the `pass` section.

`pass`
: Describes those packets that will be allowed to traverse the link on an already-established connection. Only packets which would be passed can cause a dynamic link to be brought up. Any packet that is not passed is discarded, although it may be logged first, if desired.

`keepup`
: Describes packets that will reset the idle timer on a dynamic connection, thereby keeping the line connected.

`log`
: Describes packets whose headers or contents are to be noted in the log file.

After each action keyword comes stanzas, separated by white space, describing packets that fit the criteria for that action. Each stanza is processed in the order shown in the file, and contain restrictions or permissions on the packets encountered. As soon as a pattern or a condition is found that matches the packet in question, the indicated action is taken and the rest of the listed stanzas are ignored (i.e. inclusive or with shortcut evaluation).

Stanzas may contain IP protocol numbers, optionally hyphen-separated ranges of TCP or UDP port numbers along with the `/tcp` or `/udp` qualifier, numbers representing ICMP message types or codes along with the `/icmp` qualifier, service names corresponding to entries in the `services` file, or names or IP addresses of hosts or networks, or the special keyword `all`, which is the default for all actions except the `log` filter, where the default is `!all`. (Usually, it is unnecessary to use `all`; as a convenience, a `!all` is automatically added at the end of a stanza list if the last stanza is not negated, and an `all` is added at the end of a stanza list if the last stanza is negated. For example, in the typical case of `log` this sensibly results in only those packets matching the stanzas shown being sent to the system log, and no others. In the typical case of `pass`, this results in certain listed packets being restricted, but allowing the passage of all others.) If a network is specified, either by name or by address, then the corresponding network mask must also be specified if it is of a different size than the default for that class of network. The network mask and additional 'and' conditions within a stanza are separated by slashes ('/'), and may be specified either as a series of decimal numbers separated by periods, or as a single 32-bit hexadecimal number. The sense of a stanza may be negated by prefixing it with an exclamation mark ('!').

In the `log` filter specification, the special keyword `trace` causes the contents (as well as headers) of the indicated type of packet to be logged. Also in the `log` filter specification, the special flag `rejected` signifies that the packet is to be logged only if it was rejected by the `pass` filter.

Since TCP data streams are opened when the initiator sends a SYN packet to the intended recipient, it is possible to distinguish between outbound (about to be transmitted by this router) and inbound (coming from the other end of the link) uses of TCP applications such as telnet or FTP. The special keyword `syn` allows filtering or logging these connection starters. Qualifying it with `recv` or `send` allows sessions to be started or logged only if they are initiated in the indicated direction. The special keyword `fin` allows filtering or logging the packets that close TCP connections.

The `src` and `dst` keywords serve to distinguish ports, addresses or hostnames, as applying to the source or destination, respectively, of the packet. If both are applied to the same stanza (e.g. .../`src`/`dst`), then both the source and destination address and/or port must match.

The `unreach=` keyword causes an ICMP Destination Unreachable message (RFC 792 and RFC 1122 section 3.2.2.1) to be sent to the packet's source address, bearing the indicated code field, which may be chosen from

`net` The destination network is unreachable.

`host` The destination host is unreachable.

`prot` The designated transport protocol is not supported

`protocol`
The designated transport protocol is not supported

`port` The designated transport protocol (e.g., UDP) is unable to demultiplex the datagram but has no protocol mechanism to inform the sender.

`needfrag`
Fragmentation is needed and the Don't Fragment flag is set.

`srcfail`
Source route failed.

`net-unknown`
The destination network is unknown.

`host-unknown`
The destination host is unknown.

`host-isolated`
: The source host is isolated.

`net-prohibited`
: Communication with the destination network is administratively prohibited.

`host-prohibited`
: Communication with the destination host is administratively prohibited.

`net-tos`
: The destination network is unreachable for the designated type of service.

`host-tos`
: The destination host is unreachable for the designated type of service.

The `ip-opt=` keyword can be used to select packets based on whether they bear various IP options (RFC 1122 section 3.2.1.8 and RFC 791 section 3.1 (pps 16ff)), selected from

`rr` Record Route is used to trace the route an internet datagram takes

`ts` Time Stamp

`security`
: Security is used to carry Security, Compartmentation, User Group (TCC), and Handling Restriction Codes compatible with DOD requirements.

`lsrr` Loose Source Routing is used to route the internet datagram based on information supplied by the source.

`satid` SATNET Stream Identifier (obsolete)

`ssrr` Strict Source Routing is used to route the internet datagram based on information supplied by the source.

`srcrt` Either Loose Source Routing or Strict Source Routing

`any` Any IP option—could even match the No Operation option.

Default Behavior

The following `Filter` file describes the default behavior of `pppd` and `frd`, either in the absence of a filter specification file or in the case of an empty file:

```
#       Filter - Configuration file binding packet
#       types to actions.  Describes the default
#       behavior of the pppd and frd daemons:
default bringup all pass all keepup all log !all
```

The default behavior is no restriction of packets, and no logging. The `frd` daemon does not use the `bringup` or `keepup` filters.

Internet Firewall Example

A `pass` line like this might be appropriate as a security firewall between an organizational network and the larger Internet:

```
internet-gateway
       bringup !ntp !3/icmp !5/icmp !11/icmp !who !route
               !nntp !89
       pass    nntp/137.39.1.2 !nntp
               telnet/syn/recv/137.175.0.0
               !telnet/syn/recv !ftp/syn/recv
               !login/syn/recv !shell/syn/recv !who
               !sunrpc !chargen !tftp !supdup/syn/recv
               !exec !syslog !route !6000/tcp/syn/send
       keepup  !send !ntp !3/icmp !5/icmp !11/icmp
               !who !route !89
       log     rejected
```

This `pass` specification allows NNTP (Usenet news) transactions with one peer and no others. It allows incoming Telnet sessions from hosts on only one network, disallows all other incoming Telnet, SUPDUP, and FTP sessions, and allows all outgoing Telnet SUPDUP, and FTP sessions. It allows X Window System clients running elsewhere to display on local window servers, but it allows no local X clients to use displays located elsewhere. It disallows all SUN RPC traffic, thereby guarding the local YP/NIS and NFS servers from outside probes and filesystem mounts. Alas, it also disallows local machines from mounting filesystems resident on NFS servers elsewhere, but this can't be helped because NFS uses RPC which is a UDP service, and therefore without the SYN and FIN packets that can be used to

characterize the direction in which a TCP stream is being initiated. It blocks several other sorts of traffic that could be used for nefarious purposes, and the absence of a trailing `!all` means that any traffic not explicitly blocked is permitted to pass.

The `bringup` and `keepup` lines are appropriate for an intermittent dial-up connection, so that various error conditions won't cause the link to be established, nor to keep the call open beyond its usefulness. OSPF (Open Shortest Path First) routing packets (IP protocol number 89, from RFC-1340) will cross the link, but won't cause it to be brought up, nor keep it up if it's otherwise idle. Usenet news traffic won't bring up the link, but once started, the link won't be shut off in the middle of a news batch. The `log rejected` line keeps a record of every packet that is blocked by the `pass` line, so that unsuccessful penetration attempts will be noted.

An Extremely Complex Example

The following `Filter` file instructs `pppd` that a connection to any neighbor except the host *backbone* be brought up in response to any packet except for those generated by NTP, ICMP Destination Unreachable, and rwhod. If those are the only types of packets flowing across the link, it will not be kept up, but all packets are allowed to cross the link while it is up. Packets sent out will not reset the idle timer, but packets received from the peer will. If the peer goes down and modem problems cause the phone not to be hung up, (and the idle command-line argument has been specified) `pppd` will hang up the connection and retry.

In the special case of the host *backbone* (perhaps a server belonging to a network connectivity vendor), only telnet and FTP sessions, SMTP electronic mail, NNTP network news, and Domain Name System queries are considered sufficient cause to bring the link up or to keep it up if otherwise idle.

Once the link is up, all the above plus NTP clock chimes and ICMP messages may flow across the link. No packets to or from a particular host, nor any packets except Domain Name System queries and responses for any host on subnet 42 of the class B network 137.175 are ever allowed to cross the link, nor would they cause the link to be initiated. We allow telnet and FTP sessions only if they are initiated in the outbound direction.

We log one-line descriptions of various ICMP problem messages (Unreachable, Time Exceeded), and the complete contents of ICMP messages reporting IP header problems. We log all telnet and FTP sessions, including inbound attempts (though they will fail because they are excluded in the `pass` specification above). We also log the header of the first packet of any electronic mail message flowing over this link on its way to or from a specific host.

```
#
#       Filter - Express configuration file binding packet
#                types to actions.
#
#       For packets that would pass, these services
#       will bring up the link:
#
backbone  bringup  smtp nntp domain telnet ftp
#
#       Once brought up, these will pass (or not):
#
        pass    !131.119.250.104
                domain/137.175.42.0/255.255.255.0
                !137.175.42.0/0xffffff00
#               (alternative ways of
#                expressing subnet mask)
                !telnet/syn/recv !ftp/syn/recv
                domain smtp nntp ntp icmp telnet ftp
#
#       Packets received for the services shown will
#       reset the idle timer.
#
        keepup  !send smtp nntp domain telnet ftp
#
#       Only these messages will have headers or contents
#       logged, unless higher-level debugging is set:
#
        log     3/icmp 11/icmp 12/icmp/trace
                telnet/syn ftp/syn
                smtp/syn/terminus.netsys.com
#
default bringup !ntp !3/icmp !who
         keepup  !send !ntp !3/icmp !who
```

Recommendations

Simpler filter specifications allow `pppd` and `frd` to start up quicker and run faster, with less processing overhead for each packet, but that overhead is likely to present a problem only at very high line speeds (like T1). The *backbone* example shown

above is severe overkill for the sake of illustration, evolved over a period of several weeks, and took the authors several tries to get right. Start with a simple filter specification and add each special case only as the need arises, usually as the result of watching packet logs. Then test carefully to ensure that your change had only the desired effect.

Be very careful with header logging and even more careful with packet content tracing. Make the selection criteria very narrow, or the syslog file will grow extremely large in a short period of time.

If you specify host names, be sure that their addresses are available locally, even with the connection down. If you find that you must bring up an on-demand connection to resolve a domain name, consider using that host's IP address (decimal numbers separated by periods) in both `Filter` and `Systems` instead.

If you want to specify all Domain Name System traffic, use `domain` which will be expanded to entries for both `53/tcp` and `53/udp` (some DNS traffic uses each transport). To allow queries but disable domain transfers, use `!domain/tcp`.

If your `services` file is missing some application-level protocols that you consider useful, you can populate it with entries from the Assigned Numbers RFC, number 1340. For example, you may find it useful to add lines like

```
gopher      70/tcp
www         80/tcp
kerberos    88/tcp
kerberos    88/udp
snmp        161/udp
nextstep    178/tcp
prospero    191/tcp
prospero    191/udp
x11         6000/tcp
```

if you're using those applications, and if they're not already in your `services` file. If you augment `services` this way, then instead of using entries like

```
default  pass   !6000/tcp/syn/send
```

`Filter` could use entries like

```
default  pass   !x11/syn/send
```

which is much more readable.

See also `pppd`, `frd`, `services`, RFC 791, RFC 792, RFC 1055, RFC 1548, RFC 1332, RFC 1122, RFC 1144, and RFC 1340.

A.3.6 Login

This is a shell script designated in `passwd` for use by incoming dial-in PPP and SLIP users. The line in `passwd` might look like

```
Pthem::Their PPP:Login
```

The `Login` shell script might look something like

`pppd` *hostname*`:`

where *hostname* should be replaced by the hostname of the *Express* router.

A.3.7 Keys

This section describes the format of the configuration files used with the `gw-crypt` option of the `pppd` and `frd` daemons, frequently referred to as the `Keys` file, although the `gw-crypt` option does not have a default filename.

Encryption is not available in products exported from the U.S.A.

The file named in the `gw-crypt` option of `pppd` or `frd` contains key values used for gateway-to-gateway packet encryption. Before transmission, packets with source and destination addresses matching the endpoints on a keys file line are encrypted using DES with the key specified on that keys file line. Upon reception, packets with source and destination addresses matching those on a keys file line are decrypted using DES with the key specified on that keys file line.

Format

Each key specification is on its own line of up to 1023 characters. Longer lines are truncated. Comments in the keys file begin with a '`#`' and extend to the end of the line; blank lines, or lines beginning with a '`#`', are ignored. Fields are separated by horizontal white space (blanks or tabs).

The first two words on a key line are compared with the source and destination addresses of each packet to be transmitted and each received packet. The endpoint address specifications may contain either host or network names, or host or network addresses. If a network is specified, either by name or by address, then the corresponding network mask must also be specified if it is of a different size than the default for that class of network. The mask is separated from the network name or address by a slash ('`/`'), and may be specified either as a series of decimal numbers separated by periods, or as a single 32-bit hexadecimal number, optionally with a C-style '0x' prefix.

The remainder of the key line is a 56 bit (14 digit) hexadecimal number (without the C-style '0x' prefix), used as the DES key between the specified pair of hosts or networks. The digits may be separated by horizontal white space for readability. If the key contains fewer or more than 14 hexadecimal digits, the line is ignored. If the key is weak or semi-weak, a warning message will be printed and the specified key will be used for encryption anyway.

Example

The following keys file provides pppd with keys for use when encrypting or decrypting traffic between the indicated pairs of hosts or networks:

```
#
#       Keys - PPP/Frame Relay encryption keys file
#
#       Format:
#endpoint              endpoint                key
frobozz.foo.com        glitznorf.baz.edu       feed face f00d aa
147.225.0.0            38.145.211.0/0xffffffc0 b1ff a c001 d00d 1
128.49.16.0/0xffffff00 198.137.240.100         0123456789abcd
193.124.250.136        143.231.1.0/0xffffff00  e1c3870e1c3870
```

Recommendations

Avoid using weak or semi-weak keys. These are weak DES keys:

```
00000000000000
FFFFFFFFFFFFFF
1E3C78F1E3C78F
E1C3870E1C3870
```

These are semi-weak DES keys:

```
01FC07F01FC07F
FE03F80FE03F80
1FC07F00FE03F8
E03F80FF01FC07
01C007001E0078
E003800F003C00
1FFC7FF0FFC3FF
FE3FF8FFE1FF87
```

```
003C00F001C007
1E007800E00380
E1FF87FF1FFC7F
FFC3FF0FFE3FF8
```

Security Concerns

Packets' IP headers are not encrypted, though their TCP, UDP, or ICMP headers are encrypted along with the user data portion. This allows encrypted packets to traverse normal internetworks, but permits snoopers to analyze traffic by its endpoints.

Since the TCP, UDP, or ICMP header is encrypted, protocol-based filters along a packet's path will be unable to discern whether it is SMTP, Telnet, or any other network service. This means that encrypted traffic will only permeate packet-filtering firewalls if the firewall allows all traffic between the endpoints, regardless of traffic type. Morning Star Technologies PPP/SLIP software for UNIX systems, and the Morning Star *Express* router, when deployed as the endpoint gateways of the encrypted traffic, decrypt incoming encrypted traffic before applying their configured packet filtering rules.

See also `pppd` and `frd`.

A.4 Routing File Formats

A.4.1 gated.conf

The gated config file `gated.conf` consists of a sequence of statements terminated by semi-colons (‘`;`’). Statements are composed of tokens separated by white space, which can be any combination of blanks, tabs and newlines.

Comments may be specified in either of two forms. One form starts with a pound-sign (‘`#`’) and runs to the end of the line. The other form is “C” style, which starts with a “`/*`” and continues until it reaches “`*/`”.

There are eight classes of statements. The first two classes may be specified in the configuration file in any order:

directives These statements are immediately acted upon by the parser. They are used to specify included files and the directory in which they reside. Unlike other statements which terminate with a semicolon, directive statements terminate with a newline.

trace These statements control tracing options.

The six remaining classes must be specified in order.

options These statements allow specification of some global options.

interface These statements specify interface options.

definition These statements specify options, the autonomous system and martian networks.

protocol These statements enable or disable protocols and set protocol options.

route Static routes are defined by route statements.

control Control statements define routes that are imported from routing peers and routes that are exported to these peers.

Detailed definitions of these classes of statements follow. Primitives that are used in the following definitions are:

host Any host. A *host* may be specified by its IP address or by a domain name. If a domain name is specified that has multiple IP addresses it is considered an error. The host bits in the IP address must be non-zero.

network Any network. A *network* may be specified by its IP address or a network name. The host bits in a network specification must be zero. The keyword `default` may also be used to specify the default network (0.0.0.0).

destination
Any *host* or *network*.

dest-mask Any *host* or *network* with an optional mask:

`all`
network
network `mask` *mask*
network `mask-length` *bits*
`host` *host*

autonomous system
A number between 1 and 65534 assigned by the Internet Assigned Numbers Authority to represent an autonomous system. A *mask* is a dotted quad specifying which bits of the destination are significant. The token `all` may be used to specify that any IP address may be matched. The number of contiguous bits may be used instead of an explicit mask.

gateway A *gateway* must be a *host* on an attached network.

interface An *interface* may be specified by IP address, domain name, or interface name. Be careful with the use of interface names as some systems allow more than one address per interface.

gateway-list
A *gateway-list* is a list of one or more gateways.

interface-list
An *interface-list* is a list of one or more interface names, wildcard names (names without a number) or addresses, or the token `all`, which refers to all interfaces.

preference
A *preference* is used to determine the order of routes to the same destination in a routing table. `Gated` allows one route to a destination per protocol/per autonomous system. In the case of multiple routes the route to use is chosen by *preference*, which is a number between

0 and 255, with 0 being the most preferred and 255 being the least preferred.

In case of a preference tie, if the two routes are from the same protocol and from the same autonomous system, `gated` will chose the route with the lowest *metric*. Otherwise `gated` will choose the route with the lowest numeric next-hop gateway address.

metric Is a valid metric for the specified protocol.

Directive Statements

`%include "`*filename*`"`
Causes the specified file to be parsed completely before resuming with this file. Nesting up to 10 levels is supported.

Trace Statements

`tracefile ["`*filename*`" [replace]][size` *size*`[k|m]files` *files*`];`
Specifies the file to contain tracing output. If a *filename* is specified, trace information is appended to this file unless `replace` is specified.

If specified, *size* and *files* cause the trace file to be limited to *size*, with *files* files kept (including the active file). The backup file names are created by appending a period and a number to the trace file name, starting with ".0". The minimum size that can be specified is 10k, the minimum number of files that can be specified is 2. The default is not to rotate log files.

`traceoptions` *option* [*option* [...]][`except` *option* [*option* [...]]]`;`
Changes the tracing options to those specified. If *none* is the only option specified, tracing is turned off. If the *except* keyword is specified, flags listed before it are turned on and flags listed after it are turned off. This is a simple method to turn on all but a few flags. Trace options are:

`all` Turn on all of the tracing options below except `nostamp`.

`general`
Turn on `internal`, `external` and `route`.

`internal`
Internal errors and informational messages.

`external`
: External errors.

`nostamp`
: Do not timestamp all messages in the trace file.

`mark` Output a message to the trace log every 10 minutes to allow verification that `gated` is still running.

`task` Task scheduling, signal handling and packet reception.

`timer` Timer scheduling.

`lex` Objects the lexical analyzer locates in the config file.

`parse` Tokens the parser recognizes in the config file.

`route` Changes to the `gated` routing table.

`kernel` Changes to the kernel's routing table.

`bgp` BGP packets sent and received. May be modified by `update` and `protocol`.

`egp` EGP packets sent and received. May be modified by `update` and `protocol`.

`rip` RIP packets sent and received. May be modified by `update`.

`hello` HELLO packets sent and received. May be modified by `update`.

`ospf` OSPF packets sent and received. May be modified by `update`.

`icmp` ICMP redirect packets sent and received. May be modified by `update`.

Note that redirects processed are traced under the `route` option.

`snmp` SNMP packets sent and received. May be modified by `update`.

`protocol`
: Provide messages about protocol state machine transitions when used with EGP, BGP, or OSPF.

`update` Trace the contents of protocol packets.

`adv` Policy list allocation.

Options Statements

`options` *option-list* `;`

Sets `gated` options:

`noinstall`
: Do not change kernel's routing table. Useful for verifying configuration files.

`gendefault`
: BGP and EGP neighbors should cause the internal generation of a default route when up. This route will not be installed in the kernel's routing table, but may be announced by other protocols. Announcement is controlled by referencing the special protocol `default`.

`nosend`
: Do not send any packets. This allows running `gated` on a live network to test protocol interactions without actually participating in the routing protocols. The packet traces in the `gated` log can be examined to verify that `gated` is functioning properly. This is most useful for RIP and HELLO. This does not apply to BGP packets.

`noresolv`
: Do not try to resolv symbolic names into IP addresses by using the host/network tables or Domain Name System. This is intended for systems where a lack of routing information could cause a DNS lookup to hang.

`syslog`
: Controls the amount of data `gated` logs to the system log. The default is equivalent to `syslog upto info`.

Interface Statements

`interfaces {`

`options` [`strictifs`] [`scaninterval` *time*] `;`

`interface` *interface-list interface-options* `;`

`define` *address* [`broadcast` *broadcast-addr*|`pointopoint` *local-addr*] [`netmask` *netmask*] [`multicast`] `;`

`};`

`options`

Sets some global options related to interfaces.

Options are:

`strictifs`
: Indicates that it is a fatal error to reference an interface in the configuration file that is not listed in a define statement or not present when `gated` is started. Without this option a warning message will be issued and `gated` will continue.

`scaninterval` *time*
: Specifies how often `gated` scans the kernel interface list for changes. The default is every 15 seconds on most systems, 60 seconds on systems that pass interface status changes through the routing socket (i.e. BSD 4.4). Note that `gated` will also scan the interface list on receipt of a SIGUSR2.

`define` Defines interfaces that may not be present when `gated` is started. `Gated` considers it an error to reference a nonexistent interface in the config file. This clause allows specification of that interface so it can be referenced in the config file.

Definition keywords are:

`broadcast` *broadcast-address*
: Defines the interface as broadcast capable (i.e. Ethernet and Token Ring) and specifies the broadcast address.

`pointopoint` *local-address*
: Defines the interface as a point to point interface (i.e. SLIP or PPP) and specifies the address on the local side. For this type of interface the interface *address* specifies the address of the remote host.

  An interface not defined as broadcast or pointopoint is assumed to be non-broadcast multiaccess (NBMA), such as an X.25 network.

`netmask` *subnetmask*
: Specifies the non-standard subnet mask to be used on this interface. Note that this currently ignored on pointopoint interfaces.

`multicast`
: Specifies the interface is multicast capable.

`interface`

Sets interface options on the specified interfaces. An interface list is `all` or a list of interface names (see warning about interface names), domain names, or numeric addresses.

Options are:

`preference` *preference*
: Sets the *preference* for routes to this interface when it is up, defaults to 0.

`down` preference *preference*
: Sets the *preference* for routes to this interface when `gated` believes it to be down due to a lack of received routing information, defaults to 120.

`passive`
: Prevents `gated` from changing the preference of the route to this interface if it is believed to be down due to lack of received routing information.

`simplex`
: Defines an interface as unable to hear its own broadcast packets. Currently defining an interface as `simplex` is functionally equivalent to defining it as `passive`.

`reject` Specifies that the address loopback interfaces which match these criteria will be used as the local address when installing reject routes in the kernel. Should only be used with systems based on BSD 4.3 Tahoe or earlier which have installed a reject/blackhole pseudo interface.

`blackhole`
: Specifies that the address loopback interfaces which match these criteria will be used as the local address when installing blackhole routes in the kernel. Should only be used with systems based on BSD 4.3 Tahoe or earlier which have installed a reject/blackhole pseudo interface.

Definition Statements

`autonomoussystem` *autonomous system* **;**

Sets the autonomous system of this router to be *autonomous system*. This option is required if BGP or EGP are in use.

`routerid` *interface* `;`
: Sets the router identifier for use by the BGP and OSPF protocols. The default is the address of the first interface encountered by `gated`. The address of a non-POINTOPOINT interface is preferred over the local address of a POINTOPOINT interface and an address on a loopback interface that is not the loopback address (127.0.0.1) is most preferred.

`martians {`
: *martian-list*
`};`
Defines a list of martian addresses about which all routing information is ignored. The *martian-list* is a semicolon-separated list of symbolic or numeric hosts with optional masks. See *dest-mask*.

Protocol Statements

Enables or disables use of a protocol and controls protocol options. These may be specified in any order.

For all protocols, `preference` controls the choice of routes learned via this protocol or from this autonomous system in relation to routes learned from other protocols/autonomous systems. The default metric used when propagating routes learned from other protocols is specified with `defaultmetric` which itself defaults to the highest valid metric for this protocol, for many protocols this signifies a lack of reachability.

For distance vector IGPs with no explicit connections or authentication (RIP and HELLO) and redirects (ICMP), the `trustedgateways` clause supplies a list of gateways providing valid routing information; routing packets from other gateways are ignored. This defaults to all gateways on the attached networks.

Routing packets may be sent not only to the remote end of point-to-point links and the broadcast address of broadcast-capable interfaces, but also to specific gateways if they are listed in a `sourcegateways` clause and `yes` or `on` is specified. If `nobroadcast` is specified, routing updates will be sent only to gateways listed in the `sourcegateways` clause, and not at all to the broadcast address. Disabling the transmission and reception of routing packets for a particular protocol may be specified with the `interface` clause. An `interface` clause which disables sending or receiving protocol packets may be overridden for specific peers using the `trustedgateways` and `sourcegateways` clauses.

For exterior protocols (BGP, EGP), the autonomous system advertised to the peer is specified by the global `autonomoussystem` clause unless overridden by the `asout` parameter. The incoming autonomous system number is not verified

unless `peeras` is specified. Specifying `metricout` fixes the outgoing metric for all routes propagated to this peer. If the peer does not share a network, `interface` can be used to specify which interface address to use when communicating with this peer and `gateway` can be used to specify the next hop to use for all routes learned from this peer. An internal default is generated when routing information is learned from a peer unless the `nogendefault` parameter is specified.

Any protocol can have a `traceoptions` clause, which enables tracing for a particular protocol, group or peer. The allowable protocol-specific options are: `all`,`general`,`internal`,`external`,`route`,`update`,`task`,`timer`,`protocol`, or `kernel`.

```
rip yes|no|on|off [{
        broadcast ;
        nobroadcast ;
        nocheckzero ;
        preference preference ;
        defaultmetric metric ;
        interface interface-list [noripin] [noripout] [metricin metric]
              [metricout metric]
              [version 1]|[version 2 [multicast|broadcast]];
        ...
        trustedgateways gateway-list ;
        sourcegateways gateway-list ;
        traceoptions traceoptions ;
        } ] ;
```

If the `rip` clause is not specified the default is `rip on`. `Nobroadcast` specifies that RIP packets will only be sent to gateways listed in the `sourcegateways` clause, if there are any. If `yes` or `on` is specified, RIP will assume `nobroadcast` if there is only one interface and `broadcast` if there is more than one. `Broadcast` specifies that RIP packets will always be generated. `Nocheckzero` specifies that RIP should not make sure that the reserved fields in RIP packets are zero.

> Note that using `broadcast` with only one interface is useful only when propagating static routes or routes learned from another protocol. This will cause data packets to travel across the same network twice, which may be tolerable in certain configurations.

The default `metricout` is zero; the default `metricin` is the kernel interface metric plus 1 (the default RIP hop count).

If the `version` is specified as or defaults to 1, RIP version 2 packets will never be sent except in response to a v2 POLL packet. If the `version` is specified as 2, RIP version 2 packets will be sent to the RIP multicast address if possible, or to the broadcast address, unless the method is explicitly specified.

The default `metric` is 16; the default `preference` is 100.

`hello yes|no|on|off [{`
`broadcast ;`
`nobroadcast ;`
`preference` *preference* `;`
`defaultmetric` *metric* `;`
`interface` *interface-list* [`nohelloin`] [`nohelloout`] [`metricin` *metric*] [`metricout` *metric*]`;`
…
`trustedgateways` *gateway-list* `;`
`sourcegateways` *gateway-list* `;`
`traceoptions` *traceoptions* `;`
`}] ;`

If `yes` or `on` is specified, HELLO will assume `nobroadcast` if there is only one interface, and `broadcast` if there is more than one. If the HELLO clause is not specified the default is `hello off`. `Broadcast` specifies that HELLO packets will be generated. `Nobroadcast` specifies that HELLO packets will only be sent to gateways listed in the `sourcegateways` clause, if there are any.

Note that using `broadcast` with only one interface is useful only when propagating static routes or routes learned from another protocol. This will cause data packets to travel across the same network twice, which may be tolerable in certain configurations.

The default `metricout` is zero, the default `metricin` is a translation of the kernel interface metric into a hello metric plus 100 (the default HELLO hop count).

The default `metric` is 30000; the default `preference` is 90.

```
ospf yes|no|on|off [{
        [defaults {
              preference preference ;
              cost cost ;
              tag [tag | as [as_tag]] ;
              type 1|2 ;
        };]
        [exportlimit routes ;]
        [exportinterval time ;]
        [traceoptions traceoptions ;]
        [monitorauthkey authkey ;]
        [area area {
              authtype 0|1|none|simple ;
              stub [cost cost ];
              networks {
                    network [mask mask ;]
              };
              stubhosts {
                    host cost cost ;
              };
              interface interface [cost cost] {
                    [enable|disable] ;
                    retransmitinterval time ;
                    transitdelay time ;
                    priority priority ;
                    hellointerval time ;
                    routerdeadinterval time ;
                    authkey auth_key ;
              };
              interface interface nonbroadcast [cost cost] {
                    pollinterval time ;
                    routers {
                    gateway [eligible] ;
                    ...
                    };
                    [enable|disable] ;
```

```
            retransmitinterval time ;
            transitdelay time ;
            priority priority ;
            hellointerval time ;
            routerdeadinterval time ;
            authkey auth_key ;
      };
};]
[backbone {
      authtype 0|1|none|simple ;
      networks {
            network [mask mask] ;
      };
      subhosts {
            host cost cost ;
      };
      interface interface [cost cost] {
            [enable|disable] ;
            retransmitinterval time ;
            transitdelay time ;
            priority priority ;
            hellointerval time ;
            routerdeadinterval time ;
            authkey auth_key ;
      };
      ...
      interface interface nonbroadcast [cost cost] {
            pollinterval time ;
            routers {
                  gateway [eligible] ;
                  ...
            };
            [enable|disable] ;
            retransmitinterval time ;
            transitdelay time ;
            priority priority ;
```

```
            hellointerval time ;
            routerdeadinterval time ;
            authkey auth_key ;
        };
        ...
        virtuallink neighborid address transitarea area {
            [enable|disable] ;
            retransmitinterval time ;
            transitdelay time ;
            priority priority ;
            hellointerval time ;
            routerdeadinterval time ;
            authkey auth_key ;
        };
        ...
};]

}];
```

interface An interface is specified with an address, a name, a wildcard name (name without any number), or `all`. Multiple interface clauses may be specified with different parameters, the parameters used are accumulated from the interface clauses. If a parameter is specified more than once the instance with the most specific interface reference is used. The order of precedence is address, name, wildcard name, `all`.

cost A number between 0 and 65535 specifying an OSPF internal cost.

tag The OSPF tag (an unsigned 31-bit number) to be placed on all routes exported by `gated` into OSPF.

as_tag The OSPF-BGP tag (an unsigned 12-bit number) to be placed on all routes export by `gated` into OSPF. When `tag as [`*`as_tag`*`]` is used, tag fields are automatically generated and the as_tag field is assigned if specified.

metric A number between 0 and 16777215 specifying an OSPF external (ASE) cost.

area A dotted quad or a number between 1 and 4294967295. Area 0 is always referred to as the `backbone`.

auth_key One to eight decimal digits separated by periods, a one to eight byte hexadecimal string preceded by `0x`, or a one to eight character string in double quotes.

priority A number between 0 and 255 specifying the priority of becoming the designated router on this interface.

OSPF inter- and intra-area routes are always imported into the `gated` routing table with a preference of 10. It would be a violation of the protocol to do otherwise so it is not possible to override this. OSPF Autonomous System External (ASE) routes are imported with a preference of 150. This default may be changed with the `preference` keyword in the defaults section. ASE routes are exported at a rate of 100 ASEs every 1 second; these parameters can be tuned with the `exportlimit` and `exportinterval` parameters.

`Gated` routes are exported to OSPF as ASEs with a default cost of 0 and a type of 1. By default, the tag is calculated from the AS path of the route being exported (`tag as`). These may all be changed in the `defaults` section.

OSPF areas may be specified in any order, but the `backbone` area must be specified last.

Reconfiguration (SIGHUP) is currently disabled when OSPF is enabled. This will hopefully be fixed in a future release.

```
egp yes|no|on|off [{
        [preference preference ;]
        [defaultmetric metric ;]
        [packetsize maxpacketsize ;]
        [traceoptions traceoptions ;]
        [group
                [peeras autonomous system]
                [localas autonomous system]
                [maxup number]
                [preference preference]
                {
                        neighbor host
```

```
                    [metricout metric]
                    [nogendefault]
                    [importdefault]
                    [exportdefault]
                    [gateway gateway]
                    [lcladdr local-address]
                    [sourcenet network]
                    [minhello min_hello]
                    [minpoll min_poll]
                    [traceoptions traceoptions]
                    ;
                    ...
        };
        ...]
} ] ;
```

`Packetsize` specifies the size, in bytes, of the largest EGP packet that will be accepted or sent. A `group` lists a group of EGP peers in one autonomous system. `Maxup` specifies the maximum number of peers that will be maintained in the Up state. `Importdefault` and `exportdefault` tell `gated` to import or export the default route (0.0.0.0) in updates exchanged with an EGP neighbor. If not specified, the default network is ignored when exchanging EGP updates. `Sourcenet` specifies the network to query in EGP Poll packets; this is normally the shared network. The minimum EGP hello and poll intervals acceptable may be specified with the `minhello` and `minpoll` arguments, respectively. These are both specified as a time in seconds, minutes:seconds or hours:minutes:seconds. Any number of `group` clauses may be specified containing any number of `neighbor` clauses. Any parameters from the `neighbor` clause may be specified on the `group` clause to provide defaults for the group.

The *local-address* is used to set the local address to be used when there is a choice of interfaces. If not specified it defaults to whichever interface is shared with the neighbor. If a network is not shared with the neighbor, `gateway` may be used to specify the next-hop gateway to use when installing routes learned from this neighbor. In this case the default interface is the one shared with the specified gateway.

The default `metric` is 255; the default `preference` is 200.

```
bgp yes|no|on|off [{
        [preference preference ;]
        [defaultmetric metric ;]
        [traceoptions traceoptions ;]
        [group type external|internal|igp|test peeras peeras
              [metricout metric]
              [localas localas]
              [nogendefault]
              [gateway gateway]
              [preference preference]
              [lcladdr local-address]
              [holdtime time]
              [traceoptions traceoptions]
              [version version]
              [passive]
              [importdefault]
              [exportdefault]
              [sendbuffer bufsize]
              [recvbuffer bufsize]
              [spoolbuffer bufsize]
              [keepall]
              {
                    [allow { dest-mask ... } ;]
                    [peer host
                          [metricout metric]
                          [localas localas]
                          [nogendefault]
                          [gateway gateway]
                          [preference preference]
                          [lcladdr local-address]
                          [holdtime time]
                          [traceoptions traceoptions]
                          [version version]
                          [passive]
                          [importdefault]
                          [exportdefault]
                          [sendbuffer bufsize]
                          [recvbuffer bufsize]
```

```
                    [spoolbuffer bufsize]
                    [keepall]
                    ;]
               ...
        };
        ...]
} ] ;
```

BGP peers are assigned to groups based on the `type` and `peeras`; it is not possible to have two groups with the same `type` and `peeras`. `Peer` specifies the address of each BGP peer. `Group` options provide the defaults for all peers within that group.

`Peeras` is the autonomous system expected from a peer. `Metricout` is the default metric to use when sending to this peer. `Localas` specifies the autonomous system advertised to this peer, the default is that which has been set globally. `Nogendefault` specifies that this peer should not cause the automatic default to be generated.

The *local-address* specifies the address to be used on the local end of the TCP connection with the peer. For `external` peers the *local-address* must be on an interface which is shared with the peer (or for a non-local peer's configured next-hop gateway when the `gateway` option is used to specify this) and a session with the peer will be opened only when an interface with the appropriate local address through which the peer (gateway) address is directly reachable is operating. For other types of peers, a peer session will be maintained when any interface with the specified local address is operating. In either case incoming connections will only be recognized as matching a configured peer if they are addressed to the configured local address.

`Holdtime` specifies the BGP holdtime to use with this peer. `Traceoptions` specify tracing options for this peer (and are not yet implemented). `Version` specifies the version of the BGP protocol to use with this peer. If not specified, the highest supported version is used first and then version negotiation is attempted. `Passive` specifies that active opens to this peer should not be attempted. `Importdefault` and `exportdefault` control whether the default network (0.0.0.0) can be exchanged with this peer. `Keepall` is used to retain routes learned from a peer that contain one of our autonomous system numbers in their path.

`Sendbuffer` and `recvbuffer` control the amount of buffering asked of the kernel; the default is to configure the maximum supported, up to 65535 bytes. `Spoolbuffer` is used to indicate that BGP should buffer data for peers when the kernel queues are full; the default is to break the connection. These options are normally not needed on properly functioning systems.

If a `metric` is not specified, the default is not to send a metric. The default `preference` is 170, the default `holdtime` is 180, and the default `version` is 3.

`redirect yes|no|on|off [{`
`preference` *preference* **;**
`interface` *interface-list* [noredirects] **;**
`trustedgateways` *gateway-list* **;**
`traceoptions` *traceoptions* **;**
`}` **] ;**

`Redirect` controls whether `gated` makes routing table changes based on ICMP redirects when not functioning as a router. When functioning as a router (i.e. any interior routing protocols (RIP, HELLO, OSPF) are participating in routing on any interface), ICMP redirects are disabled. When ICMP redirects are disabled, `gated` must actively remove the effects of redirects from the kernel as the kernel always processes ICMP redirects.

The default `preference` is 30.

`snmp yes|no|on|off [{`
`preference` *preference* **;**
`traceoptions` *traceoptions* **;**
`port` *port* **;**
`}` **] ;**

Controls whether `gated` tries to contact the SMUX SNMP daemon to register supported variables. The default is `on`. The default `preference` is 50. The default `port` is 199 (SMUX).

Static Statements

Static routes are specified with `static` clauses.

static {

dest-mask `gateway` *gateway* [*gateway2* [*gateway3* […]]]
[`interface` *interface-list*]
[`preference` *preference*]
[`retain`] [`reject`] [`blackhole`] [`noinstall`] `;`

…

dest-mask `interface` *interface* [`preference`
preference] [`retain`] [`reject`] [`blackhole`] [`noinstall`] `;`
…

`};`

Any number of `static` statements may be specified, each containing any number of static route definitions. The first form defines a static route through one or more gateways. If multiple gateways are specified, they are limited by the number of multipath destinations supported (on Unix this is almost always one). Only gateways on interfaces that are configured and up are used.

The second form defines a static interface route which is used for primitive support of multiple networks on one interface.

The interface list on the first form restricts static routes to a specific set of interfaces.

`Retain` causes the route to be retained in the kernel after `gated` is shut down. `Reject` causes all packets to this route to be rejected. `Blackhole` causes all packets to this route to be silently discarded. `Reject` and `blackhole` are not supported by all systems. `Noinstall` is used to prevent this route from being installed in the kernel

The preference for static routes defaults to 60.

Control Statements

Importation of routes from routing protocol peers and exportation of routes to routing protocol peers are controlled by `import` and `export` clauses.

`import proto bgp|egp as` *autonomous system* `restrict ;`

`import proto bgp|egp as` *autonomous system*

[`preference` *preference*] {
import-list

```
};
import proto bgp aspath aspath_spec restrict ;
import proto bgp aspath aspath_spec

      [preference preference] {
      import-list

};
import proto rip|hello|redirect restrict ;
import proto rip|hello|redirect

      [preference preference] {
      import-list

};
import proto rip|hello|redirect interface interface-list restrict ;
import proto rip|hello|redirect interface interface-list

      [preference preference] {
      import-list

};
import proto rip|hello|redirect gateway gateway-list restrict ;
import proto rip|hello|redirect gateway gateway-list

      [preference preference] {
      import-list

};
import proto ospfase [tag ospf_tag] restrict ;
import proto ospfase [tag ospf_tag]

      [preference preference] {
      import-list
```

`};`

If an OSPF type is specified, only routes of that type will be considered for import, otherwise either type will be considered. If an `ospf_tag` specification is given, only routes matching that tag specification will be considered, otherwise any tag will be considered. An OSPF tag specification may is a decimal, hexadecimal or dotted quad number.

If more than one `import` statement relevant to a protocol is specified, they are processed most specific to least specific (i.e. for RIP and HELLO, `gateway`, `interface`, and `protocol`), then in the order specified in the config file.

import-list

An *import-list* consists of zero or more destinations (with optional mask). One of two parameters may be specified, `restrict` to prevent a set of destinations from being imported, or a specific `preference` for this set of destinations.

dest-mask [[`restrict`] | [`preference` *preference*]] `;`

Note that the contents of an `import-list` are sorted internally so that entries with the most specific masks are examined first. The order in which *dest-mask* entries are specified does not matter.

If no import list is specified, all routes will be accepted. If an import list is specified, the import list is scanned for a match. If no match is found, the route is discarded. Rephrased, a `all restrict` entry is assumed in an import list.

`export proto bgp|egp as` *autonomous system* `restrict ;`

`export proto bgp|egp as` *autonomous system* [`metric` *metric*] `{`

export-list

`};`

`export proto rip|hello restrict ;`

`export proto rip|hello` [`metric` *metric*] `{`

export-list

`};`

`export proto rip|hello interface` *interface-list* `restrict ;`

`export proto rip|hello interface` *interface-list* [`metric` *metric*] `{`

export-list

`};`

`export proto rip|hello gateway` *gateway-list* `restrict ;`

`export proto rip|hello gateway` *gateway-list* [`metric` *metric*] `{`

export-list

`};`

`export proto ospfase` [`type 1|2`] [`tag` *ospf_tag*] `restrict ;`

`export proto ospfase` [`type 1|2`] [`tag` *ospf_tag*] [`cost` *ospf_cost*] `{`

export-list

`};`

export-list

The *export-list* specifies exportation based on the origin of a route to a destination:

`proto bgp|egp as` *autonomous system* [`restrict`] | [`metric` *metric*] [`{`

announce-list

`};`

`proto rip|hello|direct|static|default` [`restrict`] | [`metric` *metric*] [`{`

announce-list

`};`

`proto rip|hello|direct|static|default interface` *interface-list* [`restrict`] | [`metric` *metric*] [`{`

announce-list

`};`

`proto rip|hello gateway` *gateway-list* [`restrict`] | [`metric` *metric*] [`{`

announce-list

```
};
proto ospf [restrict] | [metric metric] [ {
        announce-list
};
proto ospfase [restrict | metric metric]] [ {
        announce-list
};
proto proto aspath aspath_spec [restrict] | [metric metric] [ {
        announce-list
};
proto proto tag tag [restrict] | [metric metric] [ {
        announce-list
};
```

If a `tag` is specified, only routes with that tag will be considered, otherwise any tag will be considered. An OSPF tag on an `export` statement may be a decimal, hexadecimal, or `AS` to generate a tag based on the AS path of the route being announced. An OSPF tag on an *export-list* is just an 31 bit number that is matched against the tag present (if any) on that route.

If more than one export statement relevant to a protocol is specified, they are processed most specific to least specific (i.e. for RIP and HELLO, `gateway`, `interface`, and `protocol`), then in the order specified in the config file.

By default, interface routes are exported to all protocols. RIP and HELLO also export their own routes. An `export` specification with just a `restrict` will prevent these defaults from being exported. Note that it is not possible to change the metric RIP and HELLO use for their own routes; any attempt to override it will be silently ignored.

Any protocol may be specified for *import-lists* referring to `aspaths` and `tags`. AS paths are most meaningful with BGP and OSPF ASE routes, but are generated for all routes. Tags are currently only meaningful for OSPF ASE routes.

announce-list

An *announce-list* consists of zero or more destinations (with optional mask). One of two parameters may be specified, `restrict` to prevent a set of destinations from being exported, or a specific `metric` for this set of destinations.

dest-mask [[`restrict`] | [`metric` *metric*]] **;**

Note that the contents of an *announce-list* are sorted internally so that entries with the most specific masks are examined first. The order in which *dest-mask* entries are specified does not matter.

If no *announce-list* is specified, all destinations are announced. If an *announce-list* is specified, an `all restrict` is assumed. Therefore, an empty *announce-list* is the equivalent of `all restrict`.

Note that to announce routes which specify a next hop of the loopback interface (i.e. static and internally generated default routes) via RIP or HELLO it is necessary to specify the metric at some level in the propagate clause. Just setting a default metric for RIP or HELLO is not sufficient.

aspath_spec

An AS path specification is used to match one or more AS paths.

`aspath` *regexp* `origin` [`igp`|`egp`|`incomplete`|`any`]

where the *regexp* is a regular expression over the set of AS numbers as defined in RFC-1164 section 4.2.

See also `arp`, `gated`, `ifconfig`, `netstat`, `ripquery`, RFC 891 (DCN Local-Network Protocols (HELLO)), RFC 904 (Exterior Gateway Protocol Formal Specification), RFC 911 (EGP Gateway under Berkeley UNIX 4.2), RFC 1058 (Routing Information Protocol), RFC 1163 (A Border Gateway Protocol (BGP)), RFC 1164 (Application of the Border Gateway Protocol in the Internet), RFC 1247 (OSPF Specification, Version 2), and RFC 1227 (SNMP MUX Protocol and MIB).

A.5 Network Management File Formats

A.5.1 acl.parties

SNMP Access Control List

A.5.2 smp.parties

Access information for SMP entities

A.5.3 snmpd.config

The `snmpd.config` file is read by the `snmpd` daemon for setting initial values of installation-dependent variables, SNMP community names, SNMP trap community names, and whether authentication-failure traps should be generated.

Entries are one per line in one of the following forms:

`sysDescr` *string*
: The SNMP variable *sysDescr.0* is set to *string*, which should reflect the full name and version identification of the system's hardware type, software operating system, and networking software.

`sysLocation` *string*
: The SNMP variable *sysLocation.0* is set to *string*, which should reflect the physical location of the device as installed.

`sysContact` *string*
: The SNMP variable *sysContact.0* is set to *string*, which should identify the contact person for this managed node, together with information on how to contact this person.

`snmpEnableAuthenTraps` *value*
: The SNMP variable *snmpEnableAuthenTraps.0* is set to *value*, which should be either `1` (enabled) or `2` (disabled). This variable indicates whether the SNMP agent process is permitted to generate authentication-failure traps.

`community` *community-name IP-address-in-dot-notation privileges*
: These entries specifies the manager stations authorized to issue queries and commands to the agent. If the *IP-address-in-dot-notation* is `0.0.0.0`, any address may communicate on that community name.

The *privileges* field should be set to `read` for read only, `write` for read/write, or `none` to lock out a community name.

`trap` *community-name IP-address-in-dot-notation*
: These entries specify the manager stations which are to receive trap messages emitted from the agent.

A '`#`' in column 1 indicates the beginning of a comment, which extends through the end of the line.

Appendix B

Troubleshooting Guide

B.1 Getting Help

If you have any problems with the Morning Star *Express* router, first contact the vendor or reseller from which you purchased the product. If you purchased it directly from Morning Star Technologies, send electronic mail to support@MorningStar.Com, or call +1 800 558 7827 or +1 614 451 1883 and ask for the technical support group, or send a FAX to +1 614 459 5054.

B.2 Diagnostic Codes

When the *Express* or *Express 2E* router is in various states, its front-panel 7-segment display shows two-character diagnostic codes. Each code listed below corresponds to the indicated state or problem condition.

bo **Bo**oting (loading software) from flash memory

cl **Cl**earing RAM during the boot process

rc Executing a command in **`rc.boot`**

PA The router software is in a **pa**nic condition, and is in the process of crashing. Some diagnostic information will be scrolled through the two 7-segment displays, then sent to the console device(s), before the reboot is attempted.

E1-E4 The `mse` boot file is corrupted

nb The router can't find the `mse` boot file in the flash memory

This list is growing.

B.3 Sharing Experiences

Users of the Morning Star *Express* exchange hints and experiences on the mailing list Express-Users@MorningStar.Com. If you'd like to participate in the discussion, write to

Express-Users-Request@MorningStar.Com

to indicate your interest in being added to the roster. You can catch up on past discussions in

ftp.MorningStar.Com:pub/Express/express-users-mail-archive

You can browse the discussion archive using WAIS as

wais.MorningStar.Com/express-users

Appendix C

Glossary

ARP Address Resolution Protocol, a family of mechanisms by which a node on a network finds information about other nodes on the same network. Usually involves knowing an IP address and requesting a corresponding network MAC address from a knowledgeable server.

AUI Attachment Unit Interface, the electrical interface from an Ethernet DTE to a transceiver.

bps Bits per second.

CCITT The French acronym of the International Telegraph and Telephone Consultative Committee, an organization that publishes international data communications standards.

CD Carrier Detect, also called DCD, Data Carrier Detect. The RS-232 signal that a modem will assert when it has established communications with a remote modem and is ready to send data.

Challenge Handshake Authentication Protocol A challenge-response LCP authentication protocol resistant to playback attacks. CHAP runs after LCP negotiation is complete but before any NCPs are started.

CHAP See *Challenge Handshake Authentication Protocol*.

DCD See *CD*.

DCE Data Communications Equipment, something that can connect to a DTE. Typically a modem or CSU/DSU.

DLCI Data Link Connection Identifier, used to identify a connection to a Frame Relay switch.

DNS See *Domain Name System*.

Domain Name System The distributed host and network name database system used to translate between Internet addresses and their associated host and network names. RFC 1034, *Domain Names—Concepts and Facilities* is an introduction to the DNS. RFC 1035, *Domain Names—Implementation and Specification* describes the protocols and data types used.

Dotted Quad An IP address expressed as four decimal numbers separated by periods (e.g. `192.0.2.42`).

DTE Data Terminating Equipment, something that can connect to a DCE. Typically a computer, router, or terminal.

DTR Data Terminal Ready, the RS-232 signals that a DTE asserts to tell a DCE that it is ready to communicate. A reasonably configured modem will not answer an incoming call unless DTR is present.

FCS See *Frame Check Sequence*.

Frame Check Sequence The (usually) 16-bit field transmitted after all data in a frame but before the closing flag. Its value is computed using the data portion of the frame and a binary polynomial, and its purpose is to provide the receiver with a fairly reliable check against data corruption during transmission. An "FCS error" indicates that a frame was damaged in transit.

FTP The internet File Transfer Protocol, described in RFC 959, is used to copy data files across TCP/IP networks.

hardware flow control Out of band flow control defined in EIA RS-232-D that provides bidirectional flow control using the RTS (Request To Send) and CTS (Clear To Send) signals. The DTE (typically a computer) asserts RTS when it is able to accept input, and the DCE (typically a modem) asserts CTS when it is able to accept input.

HDLC High-Level Data Link Control, defined in ISO 3309, is a bit-oriented framing and addressing scheme which is used as the basis for most modern synchronous wide-area data communications protocols, including PPP, X.25, SNA, OSI, and Frame Relay.

Internet Protocol The layer of the Internet family of protocols that is responsible for packet routing and datagram fragmentation and reassembly.

Internet Protocol Control Protocol The NCP for IP, the Internet Protocol.

Inverse ARP A technique used on Frame Relay to discover the IP address of the remote system.

IP See *Internet Protocol*.

IPCP See *Internet Protocol Control Protocol*.

ISO The International Standards Organization publishes a broad range of international standards for industry, including a large number that are identical or nearly identical to CCITT standards.

LCP See *Link Control Protocol*.

Link Control Protocol The protocol that establishes, configures, and tests the data link connection.

Link Quality Monitoring A negotiable feature of LCP that allows a PPP implementation to determine when and how often the link is losing data. If both ends of a PPP link agree to perform LQM, they will periodically send Link-Quality-Report messages to their peer containing various sequence numbers, state variables, and byte and packet counts. This information is then used to measure how reliable the connection is, and can be used to decide when to shut the connection down and perhaps try again.

Link-Quality-Report An LCP message sent after successfully negotiating the LQM option. The LQR contains various state variables, packet counts, and byte counts.

LMI See *Local Management Interface*.

Local Management Interface Periodic "heartbeat" frames exchanged between the *Express* and a Frame Relay switch.

LQM See *Link Quality Monitoring*.

LQR See *Link-Quality-Report*.

MAC address The address used by the Media Access Control sublayer of the datalink layer of a network. On Ethernets, this is a 48-bit address, frequently expressed as six hexadecimal numbers separated by colons.

Magic Number A 32-bit random number, unique to each end of each PPP link, used to determine if the link is looped back. If a PPP implementation discovers that its peer has the same Magic Number, it will strongly suspect that it is talking to itself.

Maximum Receive Unit The size of the data field in the largest PPP message a PPP implementation can receive.

Maximum Transmission Unit The size of the largest IP datagram an IP interface can send.

MNP The Microcom Network Protocols include several error correction and data compression schemes for use in modems.

MRU See *Maximum Receive Unit*.

MTU See *Maximum Transmission Unit*.

NCP See *Network Control Protocol*.

Network Control Protocol Any of a group of protocols that run after LCP has successfully connected and whose purpose is to establish and configure a network-layer protocol.

Network-Layer Protocol The member of any protocol family corresponding to Level 3 on the ISO protocol stack. One example is IP.

NFS The Sun Network Filesystem protocol, described in RFC 1094, provides transparent remote access to shared files across networks using the Sun RPC protocol.

NNTP The Network News Transfer Protocol, described in RFC 977, is used for transporting and reading network news articles across internet connections.

NTP The Network Time Protocol, described in RFC 1129, is used to keep the clocks of internet-connected hosts synchronized.

octet A term used by the CCITT and some RFCs to refer to an 8-bit byte.

PAP See *Password Authentication Protocol*.

Password Authentication Protocol A simple LCP authentication protocol that sends an identifying name and an associated password. PAP runs after LCP negotiation is complete but before any NCPs are started.

peer A protocol implementation at the other (far) end of the link.

PEP The Packetized Ensemble Protocol is a proprietary modulation and error correction technique used in some Telebit modems. It can achieve throughput of up to 16000 bps, and is able to establish and maintain connections over noisy lines better than V.32, but it is asymmetric, with slow line turnaround and latency.

POP Point Of Presence, a location at which an Internet connectivity vendor provides services

Point-to-Point Protocol The Internet standards-track data-link protocol for carrying multi-protocol datagrams (including IP) over serial links.

PPP See *Point-to-Point Protocol*.

RARP Reverse Address Resolution Protocol, a family of mechanisms by which a node on a network finds information about other nodes on the same network. Usually involves knowing a network MAC address and requesting a corresponding IP address from a knowledgeable server.

Request For Comments The Internet name for a standard.

RFC See *Request For Comments*.

RI Ring Indicator, the RS-232 signal that a modem will assert when an incoming call would cause a telephone to ring.

RPC The Sun Remote Procedure Call protocol, described in RFC 1050, is used to distribute applications over multiple networked computers.

Serial Line Internet Protocol A simple protocol for carrying IP datagrams over serial links. No error detection or configuration negotiation is included.

Simple Management Protocol Simple Management Protocol] A more general followon to SNMP.

Simple Network Management Protocol Simple Network Management Protocol] A protocol for managing networks and network-connected devices. See RFC 1157 and RFC 1442.

SLIP See *Serial Line Internet Protocol*.

SMP See *Simple Management Protocol*.

SMTP The Simple Mail Transfer Protocol, described in RFC 821, is used to deliver electronic mail messages across internet networks.

SMUX The SNMP Multiplex protocol.

SNMP See *Simple Network Management Protocol*.

TCP The Transaction Control Protocol, a datagram delivery service that provides a reliable sequenced data stream between endpoints of a connection.

telnet The protocol used to establish terminal sessions across internet networks. See RFC 854 for the protocol specification. There are many other related RFCs that describe various telnet options.

transceiver A device that provides the electrical interface between an Ethernet AUI connector and the actual cable, whether thicknet (10BASE5), thinnet (10BASE2), or twisted pair (10BASET).

UDP The User Datagram Protocol, a datagram delivery service that provides neither sequencing nor reliability.

V.32 A 9600 bps full-duplex modem modulation scheme with fallback to 4800 bps.

V.32bis A 14400 bps full-duplex modem modulation scheme with fallback to 12000, 9600, 7200, and 4800 bps.

V.35 A widely used (especially in the U. S.) but supposedly obsolete high speed synchronous serial interface standard.

V.42 An error correction scheme used by many V.32 and V.32bis modems that carries asynchronous data using the LAPM protocol over a synchronous connection.

V.42bis A data compression method used by many V.32 and V.32bis modems. V.42bis can only be used over V.42.

VJ The initials of Van Jacobson, an engineer at Lawrence Berkeley Laboratories who has done lots of good things for the Internet protocols, including inventing a TCP header compression method described in RFC 1144.

Appendix D

Bibliography

You can get RFCs via anonymous FTP from nic.ddn.mil:/rfc* or from ftp.uu.net:/inet/rfc/*.

D.1 Introductory Readings

This information was taken from RFC 1463 and other sources.

Introductory Papers

- Kehoe, Brendan P. (1992) "Zen and the Art of the Internet: A Beginner's Guide to the Internet," (first edition) 95 p.
- Krol, E. and E. Hoffman. (1993) "What is the Internet?" 11 p. (FYI 20, RFC 1462).
- Malkin, G. and A. Marine. (1992) "FYI on Questions and Answers: Answers to Commonly Asked 'New Internet User' Questions," 32 p. (FYI 4, RFC 1325).
- LaQuey, Tracy with Jeanne C. Ryer. (1992) "The Internet Companion," 30 p. (on-line chapters from book published by Addison-Wesley)

Introductory Books: Basic User Guides

- Kehoe, Brendan P. (1993) Zen and the Art of the Internet: A Beginner's Guide, (second edition) 112 p. Prentice Hall, Englewood Cliffs, NJ.
- Krol, Ed. (1992) The Whole Internet User's Guide and Catalog, 400 p. O'Reilly & Assoc., Inc. Sebastopol, CA.

- LaQuey, Tracy with Jeanne C. Ryer. (1992) The Internet Companion: A Beginner's Guide to Global Networking, 208 p. Addison-Wesley, Reading, MA.

Introductory Books: Connection Starters

- SRI International. (1992) Internet: Getting Started, 318 p. SRI International, 333 Ravenswood Ave., Rm. EJ291, Menlo Park, CA 94025.

Internet services and resources

- Martin, J. (1993) "There's Gold in them thar Networks! or Searching for Treasure in all the Wrong Places," 39 p. (RFC 1402/FYI 10).
- Merit Network, Inc. (1992) "Cruise of the Internet," Merit Network Inc., Ann Arbor, MI. (Disk based tutorial available for Macintosh or Windows).
- Metz, Ray (1992) Directory of Directories on the Internet, 175 p. Meckler, Westport, CT.
- NSF Network Service Center. (nd) "Internet Resource Guide," NSF Network Service Center, Cambridge, MA.

Internet networks

- Frey, Donnalyn and Rick Adams. (1991) !%@:: A Directory of Electronic Mail Addressing and Networks, (second edition) 436 p. O'Reilly & Assoc. Inc. Sebastopol, CA.
- LaQuey, Tracy L. (1990) User's Directory of Computer Networks, 653 p. Digital Press, Bedford, MA.
- Quarterman, John S. (1990) The Matrix: Computer Networks and Conferencing Systems Worldwide, 746 p. Digital Press, Bedford, MA.

Introducing the Internet Protocols

- Comer, Douglas E. (1991) Internetworking With TCP/IP: Volume I, Principles, Protocols, and Architecture, (second edition). 547 p. Prentice Hall, Englewood Cliffs, NJ
- Hedrick, Charles L. (1987) "Introduction to the Internet Protocols," 34 p. Rutgers University Computer Science Facilities Group, Piscataway, NJ.
- Lynch, Daniel C. & Marshall T. Rose (eds). (1993) The Internet System Handbook, 822 p. Addison-Wesley, Reading, MA.

Further Reading

- Bowers, K. L. et al. (1990) "FYI on Where to Start: A Bibliography of Internetworking Information," 42 p. (RFC 1175/FYI 3).

- Malkin, G. & T. LaQuey Parker. (1993) "Internet Users' Glossary," 53 p. (RFC 1392/FYI 18).

D.2 Matrix's book list

This section was taken from RFC 1432 and other sources.

Author	Pp.	Price	Audience	Type	Other Networks
LaQuey & Ryer	208	$10.95	public	user guide	some
Kehoe	112	$22	technical	user guide	minimal
Krol	376	$24.95	researchers	guide, catalog	minimal
Kochmer	450	$39.95	researchers	guide, catalog	some
Marine, et al.	380	$39	administrative contacts	context	some
Dern	?	?	new users	user guide	chapters
Lane & Summerhill	200	$37.50	information professionals	primer	?
Malamud	376	$26.95	varied	travelog	some
Quarterman & Wilhelm	448	$42.50	varied	standards	minimal
Lynch & Rose	822	$40	technical	standards	minimal
Tennant, et al.	142	$45	professionals	textbook	?
Benedikt	444	$15.95	varied	anthology	some
Kahin	446	$34.95	faculty	scholarly	variable
Parkhurst	86	$10.50	librarians	scholarly	some
McClure, et al.	746	$45	varied	scholarly	some
Levy	473	$4.95	public	history	some
Raymond	453	$10.95	varied	dictionary	some
Stoll	332	$19.95	public	spy story	some
Hafner & Markoff	368	$22.95	public	journalism	some
Denning	574	$23.95	public	scholarly	some
Sterling	352	$23	public	documentary	some
IRG	240	$15	technical	catalog	minimal
NorthWestNet	297	$20	technical	catalog	minimal
Frey & Adams	436	$26.95	varied	desk ref.	many
LaQuey (UDCN)	645	$34.95	varied	directory	several
Quarterman	746	$50	varied	context	all

LaQuey & Ryer Tracy LaQuey, and Jeanne C. Ryer, The Internet Companion: A Beginner's Guide to Global Networking, p. 208, Addison-Wesley, Reading, MA, October 1992. $10.95. ISBN 0-201-62224-6.

Kehoe Brendan P. Kehoe, Zen and the Art of the Internet: A Beginner's Guide, p. 112, Prentice-Hall, Englewood Cliffs, NJ, July 1992. $22.00. ISBN 0-13-

010778-6.

Krol Ed Krol, The Whole Internet User's Guide & Catalog, p. 376, O'Reilly & Associates, Inc., Sebastopol, CA, 13 September 1992. $24.95. ISBN 1-56592-025-2.

Kochmer Jonathan Kochmer, and NorthWestNet, The Internet Passport: NorthWestNet's Guide to Our World Online, 4th ed., p. 450, NorthWestNet, Bellevue, WA, 1993. $39.95. ISBN 0-9635281-0-6.

Marine April Marine, ed., Internet: Getting Started, p. 380, SRI International, Menlo Park, CA, September 1992. $39.00. ISBN [none].

Dern Daniel P. Dern, The New User's Guide to the Internet, McGraw-Hill, New York, forthcoming in 1993. ISBN 0-07-016510-6 (hc). ISBN 0-07-16511-4 (pbk).

Lane & Summerhill Elizabeth S. Lane, and Craig A. Summerhill, An Internet Primer for Information Professionals: A Basic Guide to Networking Technology, p. 200, Meckler Corp., Westport, CT, forthcoming in 1992. $37.50. ISBN 0-88736-831-X.

Malamud Carl Malamud, Exploring the Internet: A Technical Travelogue, p. 376, Prentice-Hall, Englewood Cliffs, NJ, August 1992. $26.95. ISBN 0-13-296898-3.

Quarterman & Wilhelm John S. Quarterman, and Susanne Wilhelm, UNIX, POSIX, and Open Systems: The Open Standards Puzzle, p. 446, Addison-Wesley, Reading, MA, 1993. $42.50. ISBN 0-201-52772-3.

Lynch & Rose Daniel C. Lynch and Marshall T. Rose, eds., The Internet System Handbook, p. 822, Addison-Wesley, Reading, MA, 1993. $40, ISBN 0-201-56741-5.

Tennant Roy Tennant, John Ober, Anne G. Lipow, and Foreword by Clifford Lynch, Crossing the Internet Threshold: an Instructional Handbook, p. 142 pages, 1993. $45.00. ISBN 1-882208-01-3.

Benedikt Michael Benedikt, ed., Cyberspace: First Steps, p. 444, MIT Press, Cambridge, MA, 1991. $15.95. ISBN 0-262-02327-X.

Kahin Brian Kahin, ed., Building Information Infrastructure: Issues in the Development of the National Research and Education Network, p. 446, McGraw-Hill Primis, New York, 1992. $34.95, ISBN: 0-390-03083-X.

Parkhurst Carol A. Parkhurst, ed., Library Perspectives on NREN: The National Research and Education Network, p. 86, LITA, Chicago, 1990. $10.50. ISBN 0-8389-7477-5.

McClure Charles McClure, Ann Bishop, Philip Doty, and Howard Rosenbaum, The National Research And Education Network (NREN): Research and Policy Perspectives, p. 746, Ablex Press, Norwood, NJ, 1991. $45 personal; $95 institutional. ISBN 0-89391-813-X.

Levy Steven Levy, Hackers: Heroes of the Computer Revolution, p. 473, Anchor Press/Doubleday, Garden City, NY, 1984. $17.95. ISBN 0-385-19195-2 (hc). $4.95 ISBN 0-440-13405-6 (pbk).

Raymond & Steele Eric S. Raymond, ed., Guy Steele, The New Hacker's Dictionary, p. 453, MIT Press, Cambridge, MA, 1991. ISBN 0-262-18145-2 (hc). $10.95 ISBN 0-262-68069-6 (pbk).

Stoll Clifford Stoll, The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage, p. 332, Doubleday, New York, 1989. $19.95. ISBN 0-385-24946-2 (alk. paper).

Hafner & Markoff Katie Hafner, and John Markoff, Cyberpunk, p. 368, Simon & Schuster, New York, 1991. $22.95. ISBN 0-671-68322-5.

Denning Peter J. Denning, ed., Computers Under Attack: Intruders, Worms, and Viruses, p. 574, ACM Press/Addison-Wesley, Reading, MA, 1990. $23.95, ISBN 0-201-53067-8.

Sterling Bruce Sterling, The Hacker Crackdown: Law and Disorder on the electronic frontier, p. 352, Bantam, New York, 1992. $23. ISBN 0-553-08058-X.

NorthWestNet NorthWestNet, NorthWestNet User Services Internet Resource Guide, p. 297, NorthWestNet, Bellevue, WA, 1992. $20. ISBN [none].

IRG NNSC, Internet Resource Guide, p. 240, NSF Network Service Center (NNSC), BBN, Cambridge, MA, 1991. $15. ISBN [none].

Frey & Adams Donnalyn Frey, and Rick Adams, !%@:: A Directory of Electronic Mail Addressing and Networks, p. 436, O'Reilly & Associates, Sebastopol, CA, January 1991. $26.95. ISBN 0-937-17515-3 (pbk.).

LaQuey (UDCN) Tracy Lynn LaQuey, Users' Directory of Computer Networks, p. 645, Digital Press, Bedford, MA, 1989. $34.95 Digital Part Number EY-C200E-DP; Digital Press ISBN 1-555-58047-5; Prentice-Hall ISBN 0-13-950262-9.

Quarterman John S. Quarterman, The Matrix: Computer Networks and Conferencing Systems Worldwide, p. 746, Digital Press, Bedford, MA, 1990. $50. Digital order number EY-C176E-DP-SS, Digital Press ISBN 155558-033-5, Prentice-Hall ISBN 0-13-565607-9.

D.3 Other Materials

RFC 1180 *A TCP/IP Tutorial* by T. Socolofsky and C. Kale (1991, 28 pps.)

Intro *Introduction to the Internet Protocols* by Charles L. Hedrick (1987, 27 pps.)

RFC 1206 *Answers to Commonly asked "New Internet User" Questions* by G. Malkin and A. Marine (1991, 32 pps.)

RFC 1207 *Answers to Commonly asked "Experienced Internet User" Questions* by G. Malkin, A. Marine, and J. Reynolds (1991, 15 pps.)

RFC 1208 *A Glossary of Networking Terms* by O. Jacobsen and D. Lynch (1991, 18 pps.)

RFC 1173 *Responsibilities of host and network managers: A summary of the "oral tradition" of the Internet* by J. VanBokkelen (1990, 5 pps.)

Intro-2 *Introduction to Administration of an Internet-based Local Network* by Charles L. Hedrick (1988, 46 pps.)

RFC 1118 *The Hitchhiker's Guide to the Internet* by Ed Krol (1989, 24pps.)

RFC 1175 *A bibliography of internetworking information* by K. L. Bowers (1990, 42 pps.)

Reading List *Network Manager's Reading List: TCP/IP, UNIX, and Ethernet* by Charles Spurgeon (1990, 27 pps.)

Appendix E

Release Notes

E.1 Version 1.1.1, February 19 1994

- First production release. Greater stability than any pre-release (0.9.*), with improved flexibility and features.

E.2 Version 1.1.5, March 15 1994

- Fixed a nagging Sonic driver problem that caused mysterious hangs.

E.3 Version 1.1.6, March 24 1994

- Fixed a problem in `frd` that could cause all transmitted packets to be sent to the `lo0` interface instead of out the mstfilename tty interface.

E.4 Version 1.1.10, April 7 1994

- Include LQM in transmitted LCP Configure-Requests.
- Fix `getty` *cdwait* – it didn't work at all.
- Fix a bug in `cu`. Getting a `SIGHUP` just after typing `~.` could kill our parent shell.

- Fix a problem with the gzip uncompressor that would cause hour-long hangs if the compressed file contained an internal filename (as built by `gzip` *file* as opposed to `gzip` < *file* > *file.gz*)
- No longer will we crash when receiving data on a port which is not open. In the past, we'd consume all memory.
- Fix a buffer leak that would cause `frd` to lose an mbuf if it couldn't be sent because the DLCI was down.
- Insert a space after tab-expanded commands or filenames in the shell.

E.5 Version 1.1.65, May 4 1994

- Don't process received IP packets in PPP if IPCP isn't up.
- Add PPP CCP Predictor-1 data compression.
- Fix a bug that would cause us to get the wrong filter by matching on the wrong address.
- Don't try to reverse map the remote address in ftpd for the status line, just use the IP address.
- Add `sgetty` to handle incoming switched synchronous calls.
- Add `SITE EXEC` to `ftpd` to ease administration.
- Add `rechap` to `pppd` to allow us to periodically re-authenticate the peer.
- Add `max-configure`, `max-terminate`, and `max-failure` for compliance with section 4.8 of RFC 1548 .
- Add the ability to do gateway encryption on packets traversing Ethernet interfaces.
- Fix a seldom seen but long standing bug that could cause incoming TCP calls to fail with a "bad tty name".
- Don't make `pppd` exit on SIGHUP or hangup if "dedicated" is specified.
- Remove `pppd`'s "debug 12" timeout debugging code.
- Add interpretation of "$tty" shell variable.

E.6 Version 1.1.85, May 20 1994

- If `snmpd` isn't up when `pppd` starts, make `pppd` retry periodically.
- Add the 'P_AUTO' transmit parity option, which shifts automatically between the other modes depending on the parity of received characters matching 'expect' strings. Initial default is P_ZERO for all interfaces.
- Flush tty input in `getty` before printing the `login:` prompt, to get rid of any modem noise or typeahead.
- When `pppd` gets a bad FCS, print a special message if the asyncmap caused us to discard characters.
- Fix time zone processing. It was completely broken in 1.1.65.
- Don't process a frame we're flushing in SLIP mode when we finally reach the end.
- Change frame relay T1.617 Annex-D IE (information element) codes from 0x5? to 0x0?.

E.7 Version 1.1.89, May 24 1994

- Correct counting of ifInOctets on frame relay and PPP interfaces.
- Add `lmi` *q.933* LMI type to `frd`. NTT uses Q.933-compatible switches.
- Rename `lmi` *annex-d* to `lmi` *t1.617*, but retain the `lmi` *annex-d* synonym for now.

E.8 Version 1.1.111, June 20 1994

- LLC encapsulated frames were handled incorrectly in ether_input().
- Numerous minor `pppd` fixes, for conformance with dusty corners of the RFCs
  - Ignore malformed Configure-Requests; don't Configure-Reject them.
  - Compare the value size, not just the value, when checking a received CHAP Response.

  - Don't flush packets because of bad protocol bytes (odd or even) – let the upper layers handle that.
  - Discard too-long received packets.
  - Nak LCP MRU, Asyncmap, and Magic-Number options with bogus lengths rather than Rejecting them.
  - Nak received Magic-Number options with zero values.
  - Send our negotiated Magic-Number in transmitted Echo-Reply messages rather than sending zeros when `nolqm` was specified.
  - Truncate transmitted Code-Reject and Echo-Reply messages that exceed the established MRU of the peer.
  - Discard received NCP messages with length field less than 4.
  - Ignore received Terminate-Ack messages with mismatching ID fields.
  - Interpret empty Code-Reject messages as RXJ+ rather than the more severe RXJ- FSM event.
  - Ignore received Protocol-Reject messages with length field less than 6.
  - Shut down LCP immediately after sending a PAP Authenicate-Nak.
  - Ignore received PAP Authenticate-Ack and Authenticate-Nak messages with mismatching ID fields.

- Set the SNMP speed field in `pppd` and `frd` if we are told the speed, even if the circuit is externally clocked.
- Make sure we don't have multiple timeouts for `pppd`'s modem signal polling routine pending at once. This bug could cause softclock stack overflow panics.
- Widen the `wkup` field (now `wkups`) of `ps -l` to accommodate larger values.
- Don't run the timeout queue in `pppd` on the interrupt stack. Some timeouts can sleep!
- Keep the `od` command from causing watchdog timer traps when run on a long file.
- Add the `pppd` ability to define two idle timers, one for when no TCP sessions are active and the other for when there are.

- Fixed a bug that would discard packets destined for us if our local IP address ended in an octet between 224 and 239, inclusive.
- Make sure `pppd` deletes its route when it exits (when appropriate).

E.9 Version ???, Under Construction July 18 1994

- Flush output in `ping` if it gets errors on each message.
- Drop outbound packets on `lo0` if it's `ifconfig`ed down.
- Display the status of the Ethernet 'promiscuous' bit with the `ifconfig` command.
- Don't print a warning message in `frd` when we receive IP packets in SNAP format.
- Print `frd` messages about unrecognized packet types at `debug` *5* instead of `debug` *2*.
- Print the value of `LastOutPackets` in the verbose received LQR message, not the value of `LastOutLQRs` instead.
- Count `ifInOctets`, `ifOutOctets`, and `ifOutUniPackets` on SLIP links in `pppd`.
- Don't display LQM throughput percentages in the command line if LCP isn't up.

Index