Morning Star PPP

Version 1.3

User Guide

Revision 1.106
5/29/92

Published by
Morning Star Technologies, Inc.
1760 Zollinger Road
Columbus, Ohio 43221
USA

+1 614 451 1883 (voice)
+1 800 558 7827 (voice)
+1 614 459 5054 (FAX)
support@morningstar.com (e-mail)

Contents

Morning Star PPP User Guide

1. Introduction to PPP

Morning Star PPP provides an easy and flexible means of creating wide-area TCP/IP-based networks for transferring data among UNIX systems, and between UNIX systems and others that also implement the Internet standard Point-to-Point Protocol (PPP), or the non-standard but popular Serial Line Internet Protocol (SLIP). Users of Morning Star PPP can connect their isolated UNIX systems to established TCP/IP networks, or can connect geographically separated networks. Morning Star PPP can also run on a hub machine to provide multiple remote systems with transparent dial-in access to the hub and other LAN-connected hosts.

Since users continue to use the UNIX host's native TCP/IP implementation and related utilities, no additional training is needed to make full use of the new link. Because of the transparency of TCP/IP internetwork routing, many may even be unaware that remote facilities are not directly connected to the local network.

By contrast with network routers that require an expensive leased line to provide a constant connection, Morning Star PPP can establish an asynchronous link over standard telephone lines when traffic warrants a call. When the traffic has finished, Morning Star PPP can hang up the connection to save on telephone costs. User applications cannot distinguish between an on-demand dialup link and one that is constantly maintained, except by looking at the clock. Alternatively, where the traffic load requires the increased bandwidth, Morning Star PPP can run over the SnapLink SCSI-attached high speed serial interface to make efficient use of a dedicated line. If the dedicated line is unavailable or fails during use, Morning Star PPP can automatically switch over to a dialup modem to provide backup service.

The UNIX system administrator has complete control over the traffic allowed to traverse the link. Morning Star PPP can turn a UNIX system into a selective-isolation 'firewall' gateway, enhancing the security provided to other systems on the local network.

Morning Star PPP implements HDLC address/control field and protocol field compression, as well as TCP header compression, to reduce transmission overhead to a minimum. This improves both throughput and interactive response on typical modem connections. Its interface to the UNIX host's TCP/IP implements Van Jacobson's TCP 'fast queue' scheme, whereby interactive packets have priority for transmission on a congested link.

2. Using this Manual

This User Guide contains sections on

- Unpacking and installing the software
- Configuring your system to both initiate and answer PPP calls
- Link-level authentication
- Using SLIP framing rather than PPP
- Using synchronous connections and dedicated lines
- Using constantly-open telephone calls
- IP routing tips for various common network topologies
- Taking advantage of the daemon's management and monitoring features
- Tuning your applications to take best advantage of the dialup IP environment
- Asynchronous modem cabling and configuration
- Flow control
- Using Morning Star PPP with other PPP implementations
- Getting help with Morning Star PPP

The portions of this manual following this introductory User Guide have been organized in the style of the standard Berkeley UNIX manual sections. The interface to the host UNIX system's IP implementation is described in section 4. Section 5 describes the format of the configuration files, and the PPP daemon itself is described in section 8.

After the reference manual pages, you'll find

- A troubleshooting guide to help solve common problems
- A brief glossary of networking and modem terminology
- A description of the changes that have occurred with each release of Morning Star PPP
- An index of important points in the rest of the document

3. Related Manuals

The Morning Star PPP daemon is described in detail in the UNIX-style manual page **pppd**(8). Its configuration files are described in **ppp.Auth**(5), **ppp.Devices**(5), **ppp.Dialers**(5), **ppp.Filter**(5), and **ppp.Systems**(5). The IP tunnel driver is described in **tun**(4).

Hardware installation, when necessary, is described in the *Hardware Installation Guide* for the specific communications interface hardware. (Not all Morning Star

PPP installations require additional hardware, so if you are using a serial port-only product, you won't have received a copy of any *Hardware Installation Guide*.)

Last minute hints and installation instructions pertinent to the architecture of your machine can be found in the file **ppp/README**.

The authoritative references on TCP/IP and the Point-to-Point Protocol are the Internet Requests for Comments series, particularly RFCs 1331[1] and 1332[2]. These are occasionally updated by the Point-to-Point Protocol Working Group of the Internet Engineering Task Force (IETF), and are available from the Network Information Center. See the file **rfc-index** in the Morning Star PPP distribution directory for instructions on acquiring an up-to-date copy. Along with RFCs 1331 and 1332, Morning Star PPP version 1.3 conforms with the Link Quality Monitoring protocol defined in RFC 1333[3], and with the May 1992 draft of the Challenge Handshake Authentication Protocol (CHAP) and the Password Authentication Protocol (PAP). Morning Star PPP does not perform 32-bit FCS negotiation.

TCP header compression is described in RFC 1144[4], and reduces the number of protocol overhead bytes required for typical packets in a TCP data connection, to improve interactive responsiveness.

Fast queuing for interactive TCP packets is (so far) best described by its implementation in Van Jacobson's header-compressed SLIP[5].

The Serial Line Internet Protocol is described in RFC 1055[6].

If you do not have access to these documents, please contact Morning Star Technologies as described below to request a copy.

4. Sample Installation

We'll go through the steps necessary to configure a simple but illustrative installation. Our system, *robin*, has a Telebit T1600 asynchronous dialup modem on its **/dev/ttya**, which it uses for its on-demand connection to *lark*. It is also connected

[1] Simpson, W., 'The Point-to-Point Protocol (PPP) for the Transmission of Multi-protocol Datagrams over Point-to-Point Links', RFC 1331, Computer Systems Consulting Services, May 1992.

[2] McGregor, G., 'The PPP Internet Protocol Control Protocol (IPCP)', RFC 1332, Merit, May 1992.

[3] Simpson, W., 'PPP Link Quality Monitoring', RFC 1333, Computer Systems Consulting Services, May 1992.

[4] Jacobson, V., 'Compressing TCP/IP Headers for Low-Speed Serial Links', RFC 1144, Lawrence Berkeley Laboratory, February 1990.

[5] Jacobson, V., ftp.ee.lbl.gov:cslipbeta.tar.Z, if_sl.c, UTSL.

[6] Romkey, J., 'A Nonstandard for Transmission of IP Datagrams Over Serial Lines: SLIP', RFC 1055, The Asylum, June 1988.

to an adjacent machine, *sparrow*, via a short null-modem cable on its **/dev/ttyb**, which it uses for a connection that is constantly active.

4.1. Unpacking the distribution

4.1.1. Electronic distribution

If you received the software distribution via FTP or UUCP, it is a *compress*(1)ed *tar*(1) archive. Create a working directory (it doesn't matter where) and unpack it:

```
% ls mst*
mst-machinetype-ppp.tar.Z
% mkdir /usr/local/src/ppp
% cp mst* /usr/local/src/ppp
% cd /usr/local/src
% zcat ppp/mst-machinetype-ppp.tar.Z | tar xvf -
...lots of output...
% lpr ppp/LICENSE-AGREEMENT ppp/README
%
```

4.1.2. Magnetic media distribution

If you received the software on magnetic media, it is a *tar*(1) archive. Follow the instructions on the label to unpack it. For example, if you are installing SPARC software from a floppy drive:

```
% cd /usr/local/src
% tar xvf /dev/rfd0
...lots of output...
% lpr ppp/LICENSE-AGREEMENT ppp/README
%
```

4.2. Installing the software and documentation

Become the superuser:

```
% su
Password: (enter your root password here)
#
```

Next, create a new group in **/etc/group** or in the *group.byname* map if you use Sun's NIS. Choose a unique ID number for the group, call it *ppp*, and use it later for each dial-in PPP user login added to **/etc/passwd**. The group entry for our example looks like this:

```
ppp:*:42:
```

Run the installation shell script:

```
# cd /usr/local/src/ppp
# ./Install
...lots of output...
#
```

The *Install* script will ask you to confirm that you have read and understood the terms of the License Agreement before it proceeds. If necessary, the *Install* script will ask you questions about your system. It will install the daemon as **/usr/etc/pppd** (suid root, only executable by group *ppp*) and put the manual pages into the appropriate subdirectories of /usr/man. It will create a shell script named **/etc/ppp.uninstall**, to be used in case you need to remove the software from your system. Upon successful conclusion of the *Install* script you will be able to continue with building and installing a custom kernel (if necessary) to support PPP.

4.3. Building and installing a kernel

The contents of the file **/usr/local/src/ppp/README** will describe how to configure and build a new kernel, move it into place, and reboot your system to ensure that it runs correctly. If your system supports dynamically loadable device drivers, you may need only to reboot for those drivers to become available.

The **README** file also describes how to configure the asynchronous terminal interfaces on your machine, if any special arrangements are necessary, and how to configure PPP to use the *SnapLink* interface.

4.4. Installing an updated version

New versions of Morning Star PPP will usually work with the same configuration files as were used by earlier versions, though perhaps their names and locations have been changed. Prior to version 1.2, the configuration files had names like /etc/ppp.systems; since version 1.2 they have had names like /etc/ppp/Systems. **/etc/ppp.uninstall** will not remove any old configuration files, so any passwords or other information they contained will not be lost. You should examine the new example configuration files, because they may contain useful information. In particular, many new modems have been added to Dialers, and the previously-supported modems' descriptions have been improved.

A new version of *pppd* may require a new version of the tunnel driver, which is placed into the correct location by the normal installation process. In particular, the tunnel driver distributed with version 1.1 will not work with the PPP daemon from version 1.2 or later. On SPARC systems, you'll probably want to begin using the dynamically loadable version of the tunnel driver. NeXT systems only support loadable device drivers, so the kernel never needs to be rebuilt. If you were

running version 1.1, you must either (a) remove the 'tun' lines from /sys/sun/conf.c and /sys/conf.common/files.cmn, build a new kernel without the tunnel driver installed, and use the loadable driver instead (only available on SPARC), or (b) build a new kernel containing the new version of the tunnel driver.

4.5. Arranging for PPP connections

The tunnel driver must be installed in the kernel of every host that will use PPP, and minor devices (subdevices) configured for each simultaneous link that will be in operation. To make configuration easier, 'pseudo-device tun' in Berkeley-style kernel configuration files is the same as saying 'pseudo-device tun16'. This will allow your configuration to expand to as many as sixteen neighbors before having to rebuild the kernel with a larger specification like 'pseudo-device tun17'. (The static kernel memory allocated per minor device is under 200 bytes.)

The number of PPP neighbors a system can have is independent of the number of serial ports from which it's accessible, though it is constrained to that many *simultaneous* connections. Outbound PPP links (initiated by this system) are configured independently of inbound PPP links (initiated by a neighbor system). If a link might be initiated upon demand in either direction, then each system must arrange for both an outbound and an inbound connection, as described in the following sections. Inbound connections may come via the system's native serial ports, bus-based expansion ports, SnapLink ports, or a terminal server.

4.5.1. Arranging for outbound PPP connections

Whenever *robin* is booted, one PPP daemon must be started for each outbound link it will establish. In this case, since *robin* will call *lark* on demand, we add the following lines to the appropriate spot in *robin*'s boot-time shell script (**/etc/rc.local** on the example machine):[7]

```
if [ -f /etc/ppp/Startup ]; then
     /etc/ppp/Startup
fi
```

We then make the **/etc/ppp/Startup** script look like this:[8]

```
#!/bin/sh

PATH=/usr/etc:/etc:/usr/bin:/bin

if [ -f /usr/adm/pppd.log ]; then
        mv /usr/adm/pppd.log /usr/adm/OLDpppd.log
fi
```

[7] These lines can be copied from /etc/ppp/rc.local.ex.

[8] An example can be found in /etc/ppp/Startup.ex.

```
echo -n "Starting PPP daemons:"        >/dev/console

pppd robin:sparrow auto dedicated
        (echo -n ' sparrow')          >/dev/console
pppd robin:lark auto idle 150 passive
        (echo -n ' lark')             >/dev/console

echo '.'                               >/dev/console
```

The daemons started here will look through the **Systems** file to find entries for their destinations. The **device** field in **Systems** provides an index to an entry in **Devices**. The *dialer* field on that line in **Devices** provides an index into **Dialers**. (The names and functions were chosen to closely mimic their counterparts used for establishing UUCP connections.)

Notice the use of the *passive* option on the above *pppd* command lines. This tells *pppd* to not start speaking PPP until the other side speaks first. This allows for shorter PPP negotiation times, and is sometimes used on outbound daemons (when *pppd* runs in *auto* mode). At least one end has to speak first, so don't run both ends of the circuit in *passive* mode. Everything will work correctly if you run both ends of the circuit in *active* mode, which is the default behavior if you don't specify *passive*.

4.5.2. Arranging for inbound PPP connections

On machines which will accept only incoming calls no *pppd*s need to be started at boot time, since those *pppd*s will be started when the PPP logins occur. Machines that both initiate and receive calls must start *pppd* at boot time, and must also prepare accounts for incoming connections. In our example, user accounts must be created on both *lark* and *sparrow* for *robin* to be able to login and initiate PPP connections to them. In this case, since we expect *robin* to dial them on demand, we add an entry (using vipw or our favorite editor) to both *lark*'s and *sparrow*'s **/etc/passwd** files:

```
Probin::105:42:Robin's PPP login:/etc/ppp:/etc/ppp/Login

# passwd Probin
New password: some password
Retype new password: some password
#
```

The '105' in the password entry is a unique uid for this PPP login, and the '42' is the gid corresponding to the 'ppp' group in **/etc/group**. The login shell, **/etc/ppp/Login**, is a shell script that looks like this:[9]

[9] You can copy it from /etc/ppp/Login.ex.


```
#!/bin/sh

PATH=/usr/etc:/etc:/usr/bin:/bin

mesg n
stty -tostop
exec pppd `hostname`: idle 130 crtscts
```

Make sure it is executable:

```
# chmod 755 /etc/ppp/Login
#
```

The PPP user's login shell script can, of course, be located anywhere and can be named almost anything; we just use the above for purposes of illustration. On a NeXT workstation, the name of the user's login shell script may not contain the letter 'r', because of an undocumented feature that would cause **/bin/sh** to consider itself to be a security-restricted shell. Also, the 'crtscts' option is unavailable for *pppd* running on NeXT systems; enable **ttydfa** or **ttydfb** in **/etc/ttys** instead. The 'crtscts' option is unavailable on Silicon Graphics systems for similar reasons; use **/dev/ttyf2** for a modem with hardware flow control instead.

The PPP daemon should be mode 4750 (suid root and executable by members of its group but not by general users), and its group ownership should match that of the inbound PPP logins. In this example,

```
# chgrp ppp /usr/etc/pppd
# ls -lg /usr/etc/pppd
-rwsr-x---  1 root   ppp   338527 Oct 2 07:37 /usr/etc/pppd
#
```

In most cases, all (inbound) PPP logins can use the same generic Login script. If you want some specific host to start *pppd* with a special option (e.g. require authentication) then have that login account use a private login shell, for example, a **/etc/ppp/Login-***host* that looks something like this:

```
#!/bin/sh

PATH=/usr/etc:/etc:/usr/bin:/bin

mesg n
stty -tostop
exec pppd `hostname`:robin idle 130 crtscts requireauth
```

And then:

```
# chmod 755 /etc/ppp/Login-robin
#
```

4.6. Creating the PPP configuration files

Now we can start building our configuration files. See **ppp.Auth**(5), **ppp.Devices**(5), **ppp.Dialers**(5), **ppp.Filter**(5), **ppp.Systems**(5), and **Pppd**(8) for descriptions of all the configuration options, and see the **Examples/** subdirectory in the software distribution for some example configurations. (Each complete entry in **Dialers**, **Devices**, and **Systems** must be typed on a single line, except for the chat script fields, which can be continued on the next physical line by ending lines to be continued with a backslash '\' character.)

The **Systems**, **Devices**, and **Dialers** files are only used for placing on-demand calls (calls placed by a *pppd* started with the 'auto' option in response to a packet sent to its interface). The **Systems** file is scanned for entries matching the destination address supplied on *pppd*'s command line. If the matching entry's device name is 'ACU', the **Dialers** file is searched for an entry whose speed field matches the speed field of the entry in the **Systems** file. The **Dialers** file is then searched for an entry whose dialer name field matches the dialer name field of the entry in the **Devices** file. If the **Systems** file's device field is *not* 'ACU', then the **Devices** file is instead searched for an entry with 'Direct' in the dialer name field, and the **Dialers** file is not consulted.

The **Devices** file (in **/etc/ppp**, or in **/usr/lib/ppp** on SCO systems) looks like this:

```
T1600    cua      38400     crtscts
Direct   cub      38400     crtscts
```

(The 'crtscts' option is unavailable on NeXT systems; use the device name cufa for outbound connections instead.) The **Dialers** file looks like this:[10]

```
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
```

This elaborate dialer string will cause *pppd* to abort the connection attempt if any of several things go wrong with the telephone call, then will disable UUCP protocol spoofing in the modem before dialing the destination telephone number. Here is the **Filter** file:[11]

```
default bringup !ntp !3/icmp !who
        keepup  !send !ntp !3/icmp !who
        pass    !route
        log     !nntp tcp/syn/recv
```

It says to bring the connection up for any traffic except for Network Time Protocol packets, ICMP Network Unreachable messages, and packets from the *rwho*

[10] It is available, along with many others, in /etc/ppp/Dialers.ex.

[11] Another example is available in /etc/ppp/Filter.ex.

daemon. After the link is up, the same packets will keep it alive, except for those we send. All packets will be passed except for RIP routing messages, and a message will be printed to the log file whenever an inbound TCP session is established, except for NNTP connections.

Here is the **Systems** file:

```
lark Any;50 ACU 38400 5551212 ogin: Probin word: \qmypasswd
sparrow Any;50 cub 38400 0 "" \r ogin: Probin word: \qmypasswd
```

Last, set their file protections appropriately:

```
# chmod 644 Devices Dialers Filter
# chmod 600 Systems
# chown root Systems
#
```

4.7. Testing the installation

Our PPP configuration is now complete, so we can go ahead and start it up for testing:

```
# pppd robin:lark auto passive
#
```

then make sure it got started:

```
# tail -f /usr/adm/pppd.log &
2/14-08:17:54-5997 Morning Star Technologies PPP
2/14-08:17:54-5997 Version 1.3 [21-Feb-1992 07:43:43]
2/14-08:17:54-5997 du0: pppd robin:lark auto passive
```

then use *telnet* to bring up the link, but then type ^D to exit the login:

```
# telnet lark
Trying lark ...
Connected to lark.
Escape character is '^]'.

SunOS 4.1 UNIX (lark) (ttyp3)

lark login: ^DConnection closed by foreign host.
#
```

Notice how the log shows the link was initiated:

```
2/14-08:18:45-5997 tcp 137.175.2.11/1204 -> 131.187.1.131/telnet
                                                     40 syn bringup
2/14-08:18:45-5997 Dialing lark (cua/38400/5551212/T1600)
2/14-08:19:11-5997 PPP connected to 131.187.1.131
2/14-08:20:34-5997 tcp 137.175.2.2/smtp <- 129.221.8.2/947 40 syn
```

This log file snippet shows that our *telnet* session to *lark* was the cause for the link being initiated. It also shows that the 'log' line in **Filter** is operating as expected, and that the first incoming TCP connection delivered mail (by SMTP).

Morning Star PPP is installed, configured, running and connected on both ends of one of our links. When the link is up, users should be able to access *robin* from a remote machine via *rlogin*. You should also be able to *telnet*, *ftp*, etc. between the two. If either machine is connected to a local area network, you can set up IP routing on the two networks so that hosts on either can communicate with hosts on the other, using the machines on the ends of the PPP link as routers.

5. Special Link Management Issues

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

5.1. Active vs. Passive

Morning Star PPP 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.

Most implementations (including Morning Star PPP) expect to actively initiate the negotiation process, but the negotiation may progress more slowly if both ends are active. If a system running such a PPP dials into a system running Morning Star PPP, the answering *pppd*'s default active behavior may result in a slight connection delay. In this case, it may be desirable for the Login script to invoke *pppd* with the *passive* option, although it would be preferable to configure the caller to be passive if possible.

5.2. 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 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.

5.3. The *exec* option

The *exec* option provides a way to trigger an action when a link is established, and a different action when the link is taken down. It can be used on either the calling or answering end of the link, and the specified command is executed immediately after the link is either established or hung up. The command or shell script specified as the argument to *exec* is invoked with either 'up' or 'down' as its first argument, depending upon whether the link has just been established or just been hung up, respectively. The second argument is the IP address of the peer, and the arguments with which *pppd* was invoked are provided to the *exec*ed command as its third and subsequent arguments.

Suppose your Login shell script looks like

```
#!/bin/sh

PATH=/usr/etc:/etc:/usr/bin:/bin

mesg n
stty −tostop
exec pppd ‘hostname‘: idle 130 crtscts exec /etc/ppp/Exec
```

/etc/ppp/Exec could be an executable shell script that looks like

```
#!/bin/sh
```

```
case "$1" in
    up) echo link is up at `date` > /dev/console ;
        /usr/lib/sendmail -q < /dev/null ;;
  down) echo link is down at `date` > /dev/console ;;
esac
```

This will cause *sendmail* to run its queue whenever a remote system establishes a connection, possibly triggering a delivery to that system.

Since *pppd* runs **suid root**, anything specified in the Exec script will be run as root, and therefore presents a potential risk if mismanaged. Take appropriate security precautions with the Exec script, to ensure that it cannot be modified by non-**root** users.

5.4. 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, Morning Star PPP 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, the *exec* option, 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 *crtscts* on most systems, or use **cufa** and **ttydfa** on NeXT systems) 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, and cannot be used with the HDLC device driver or the SnapLink synchronous interface.

You can use Morning Star PPP 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 UNIX end of the point-to-point networking interface. See Systems.ex for an example use of this facility.

5.5. Synchronous PPP

Morning Star PPP can run in synchronous mode using the Morning Star SnapLink SCSI-attached high speed serial interface at line speeds up to T1 (1.544Mb/s). On Sun SPARCstations, it can run at up to 64Kb/s on the system's native serial ports, though speeds above 38400 are unpleasant without the SnapLink hardware because of the extreme I/O interrupt load. On 68040 NeXTstations, it can run at up to 19200 bits per second; higher speeds will cause frequent framing errors. TCP user data will arrive undamaged, but line efficiency will decline as the line speed increases.

To prepare your UNIX system to use the SnapLink interface, follow the instructions in the SnapLink Hardware Installation Guide. If it's supported on your system, installation of the HDLC device driver is discussed in the file **README** included with Morning Star PPP.

For Morning Star PPP to connect at T1 speeds to another host or router using port 0 on a SnapLink that emulates SCSI disk device 2, use a **Systems** entry like this:

```
hostname Any;5 rsd2a/0 1536000 0
```

The **Devices** entry looks like this:

```
Direct  rsd2a/0 1536000
```

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 38400'), after the entry for the dedicated link. *Pppd* will by default ask the peer to send LCP Link-Quality-Report messages. If the dedicated line fails, *pppd* will stop receiving the reports, and will terminate the connection when enough reports have been lost to drive the measured link quality below the configured threshold (see the next section 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, using the modem to dial up and reestablish IP traffic. Users will notice that the link is slower, but it 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, the dedicated connection.

5.6. Link Quality Monitoring

Link Quality Monitoring is an optional facility defined by PPP and provided by some PPP implementations which allows PPP to make policy decisions about the quality of a communications line. For example, if an intercontinental telephone call is of such poor quality that significant quantities of packets are being damaged in transit, the PPP implementations on each end can decide to hang up and redial in hopes of getting a better connection. This can happen without user intervention or awareness, because even if data packets are 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. Another case in which LQM is useful is in the quick detection and response to total link failures, providing users with failover protection, as described in the section above.

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 Morning Star PPP to switch to LCP Echo-Requests, MST PPP 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.

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.

5.7. 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 *login* process is run on the device and both ends of the circuit actively try to connect to their peer. Each machine’s **Startup** script should contain a line like

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

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

```
Direct cua 38400 crtscts
```

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.

5.8. 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.)

5.9. Link-Level Authentication

Morning Star PPP implements both the Password Authentication Protocol (PAP) and the Challenge Handshake Authentication Protocol (CHAP). If *pppd* is given either of the *requireauth* or *requirechap* options, 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.

6. Observing *pppd*'s progress

Pppd records its actions in two places: the argument vector and the log file.

6.1. The Argument Vector

As *pppd* runs, it updates its argument vector. You can see its progress by typing **ps axww | grep ppp** (or **ps -ef | grep ppp** on a System V machine). Some UNIX systems don't show the current argument vector in *ps*, but on those that do, 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**.
- The name of the serial interface device in use. If an inbound *pppd* is using **/dev/ttya**, this field would say **ttya**. If an outbound *pppd* is using **/dev/cub**, it would say **cub**.
- The Internet address of the peer.
- 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.

- The idle timer value: the number of seconds that have passed since the link last carried a packet described in the *keepup* line.

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

6.2. The Log File

The log file (default **/usr/adm/pppd.log**) also contains reports about the daemon's progress. Higher debugging levels provide more detailed logging output. At extremely high debugging levels such as 9 or 11, 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 the daemon's progress through the chat scripts in **Systems** and **Dialers**.

7. IP Routing Tips

The UNIX host's IP implementation sees Morning Star PPP as a point-to-point network connection 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 host or 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 is cumbersome, because any topology change requires that all the machines even remotely involved be reconfigured by editing their boot-time shell 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 all the hosts involved. The RIP protocol[12] used by *routed* is generally considered inadequate for describing transient point-to-point links. You would do better to install the Gate Daemon[13]. If you use *gated*, you should invoke *pppd* with the *netmask* argument set to the same value as used on the LAN, if that subnet mask is different from the default for the class of your network.

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

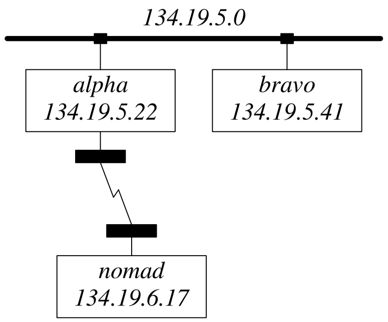
[13] gated.cornell.edu:pub/gated/gated-2.0.1.14.tar.Z or gated-alpha.tar.Z, also available via anonymous PPP/FTP from Morning Star Technologies.

7.1. Connecting a Host to a LAN

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

7.1.1. Separate Network

The method that will cause the least confusion is to assign a network or subnet for use by all the remote machines. For example, 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 hosts *alpha* (134.19.5.22) and *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 above. A laptop SPARCstation clone named *nomad* wishes to connect to the network.



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

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

or, if all the names can be found in the local **/etc/hosts** (or resolved via NIS/YP, NetInfo, the Domain Name Service, or some other mechanism 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 set up 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 as

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

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

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

7.1.1.1. Proxy ARP

The Proxy ARP[14] daemon can be used in a subnetted environment to accomodate hosts with older IP implementations that don't work correctly with subnetting. Suppose, in the 'Separate Network' example above, the host *bravo* were running a version of IP based on BSD 4.1c or 4.2, rather than on the more recent BSD 4.3 networking code. To enable *bravo* to find its way to *nomad*, one could run a **proxyarpd** on *alpha* and provide it the following **proxytab** configuration file:

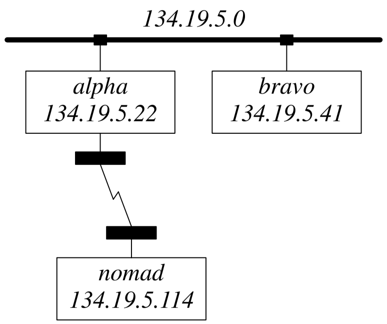
```
# Proxy ARP table for proxyarpd
# Service provided for hosts on 134.19.5.0 that do not understand
# internet subnetting.
#
# Fields are:
# 1) interface to service (same as on command line)
# 2) net to tell attached hosts that we "are"
# 3) host on net 1) to send traffic to (gateway)
#
le0   134.19.6.0   134.19.5.22
```

7.1.2. ARP[15] 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. 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 hosts *alpha* (134.19.5.22) and *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 above. A laptop SPARCstation clone named *nomad* wishes to connect to the network.

[14] Available from ftp.cis.ohio-state.edu:pub/proxyarp/proxyarpd.shar.Z or via anonymous PPP/FTP from Morning Star Technologies.

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



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 set up 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 as

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

Alpha's **/etc/rc.local** would contain a line like

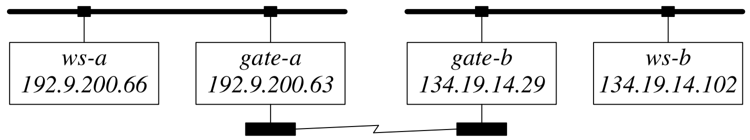
```
arp -s nomad `ifconfig le0 | sed -n "/ether/s/ether//gp"` pub
```

(If your system's Ethernet interface is named something other than 'le0', use that name instead.) This would add a permanent entry to *alpha*'s ARP table, and cause it to be provided to other systems on the local Ethernet. 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.

7.2. Connecting two LANs

Suppose *gate-a* (192.9.200.63) and *ws-a* (192.9.200.66) are on one LAN, with *gate-a* supporting a modem. Also suppose *gate-b* (134.19.14.29) and *ws-b* (134.19.14.102) are on another LAN across town, with *gate-b* supporting a modem.



gate-b's PPP daemon should be started as

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

and *gate-b* should have a route like

```
route add net 192.9.200.0 gate-a 1
```

ws-b should have a route like

```
route add net 192.9.200.0 gate-b 1
```

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

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

and *gate-a* should have a route like

```
route add net 134.19.14.0 gate-b 1
```

or, depending upon the structure of LAN b, maybe

```
route add net 134.19.0.0 gate-b 1
```

ws-a should have a route like

```
route add net 134.19.14.0 gate-a 1
```

or, again depending upon the structure of LAN b, perhaps

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

7.3. Connecting a host or 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 *hotel* supported a modem to call a POP, it would start its PPP daemon like

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

and would arrange a route as

```
route add default pop.foo.net 1
```

Any hosts on the LAN 'behind' *hotel* would set a route like

```
route add default hotel 1
```

A machine's default route should point to the next machine along its path 'outward' to the Internet. If *hotel* 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 *hotel*'s packets to their destination, whether on the LAN or elsewhere on the corporate internet or onto the Internet.

7.4. A Common Test Setup

Upon receiving Morning Star PPP, users often want to test its performance, and to practice configuring the software by running it between two workstations in one location, before moving one end of the PPP link to a remote site. This is a good strategy, providing that proper caution is employed while assigning addresses to the endpoints of the PPP link.

To avoid routing ambiguities, the endpoints of the PPP link should be given addresses on a different IP network (or subnet) number than the network (or subnet) connecting the machines over Ethernet. For example, if the Ethernet interfaces are at 134.19.5.22 and 134.19.5.41, the PPP interfaces should **not** be on the 134.19.5 subnet. Instead, the tester should pick an unused subnet number from the same IP address space, or even use something totally unassigned. A good choice might be for one end of the PPP link to be 134.19.17.53 and the other to be 134.19.17.68. This way, normal services will continue to flow across the Ethernet as usual. To cause traffic to cross the PPP link (whether a null-modem cable or a modem pair), make the connection from one testing machine to the IP address of the other machine's PPP interface (e.g. *telnet 134.19.17.68*).

8. Observations about on-demand dialup IP

8.1. Application behavior

The scheme used to support on-demand dialups in Morning Star PPP depends upon network- and higher-layer protocol timeouts typically being longer than the time needed to allocate a modem, dial it, complete the call, login to the remote system, start up the PPP protocol handler there, and establish LCP and IPCP connectivity. This is a reasonable assumption, since a local call will typically require about 25 seconds between the daemon's receipt of the first 'bringup' packet and the establishment of IP-level connectivity (of that, more than 20 seconds are consumed with dialing the call and training the modems). This scheme works even if the first remote modem is busy and the dialing pppd must try a second telephone number, so long as the dialer script is carefully written to include appropriate ABORT and TIMEOUT strings. The user's application will simply wait for the connection to be established and continue as if the connection had always been in place. This even works well when a link times out with active TCP sessions in place, such as a *telnet* connection to the remote end. When the user types, or when an application writes output, a new call is placed from the end sending the data and the session is reestablished without the application noticing anything was amiss.

Unfortunately, if the connection cannot be reestablished, TCP will time out and the application may fail ungracefully. This most often happens during times of resource contention (no free modems to place or answer the call) or acute connectivity problems (telephone system call-routing breakdown or extreme service degradation). The effect will be the same as if the machine at the 'other' end had crashed ungracefully, since the application will only be able to tell that it is unable to exchange packets with its partner.

8.2. Dealing with bandwidth limitations

The Internet's TCP/IP protocol suite is remarkably flexible and robust, with a wide variety of applications for use over media ranging from multigigabit data superhighways to long-delay satellite channels to local area networks to dialup modems to amateur radio to carrier pigeons[16]. While by definition all IP-based applications may be transparently used over links traversing any combination of media, some applications are better suited to media with the relatively low bandwidth and moderate latency and delay characteristics of asynchronous dialup connections.

[16] Waitzman, D., 'Standard for the transmission of IP datagrams on avian carriers', RFC 1149, BBN, April 1990.

8.2.1. Applications with few problems on thin wires

PPP over current-technology dial modems[17] provides satisfactory performance for most typical TCP/IP and UDP/IP wide-area applications. Users will experience useful file transfer throughput (V.32/V.42bis yields 1.7–2.8 Kbytes/sec for a typical large binary data file) and reasonable interactive response in applications such as *telnet*, *rlogin*, and the like. Plenty of bandwidth is available for network infrastructure applications such as SMTP electronic mail, NNTP network news, NTP clock synchronization, Domain Name Service, etc.

8.2.2. Applications with some problems on thin wires

The speed limitation of the dialup medium becomes more apparent when larger amounts of data are transferred, particularly for interactive applications such as *xterm* or other display-oriented programs. While large bitmap editors and CAD packages will operate correctly, users may find that the modem simply isn't able to keep up with their demands for interactive responsive.

8.2.3. Applications requiring special care on thin wires

Applications that are heavy users of communications bandwidth, such as Sun's Network File System (NFS), should be used with care over dialup links. NFS is able to easily overwhelm a modem link. While performance is acceptable for read-only file access, NFS's synchronous write semantics will make file creation and writing operations noticeably slower than on a local file system. Attempting to execute a program stored on a remotely-mounted filesystem will have this problem too, since paging will occur from the remote executable image.

Fortunately, there are usually other approaches more appropriate for wide-area networking over low-bandwidth links to solving the problems that NFS addresses. Use Berkeley's *rdist* system to keep the important segments of two sites' file structures in reasonable synchrony. Also, use a package like *ange-ftp.el*[18] in your GNU Emacs to automatically manage FTP connections that are transparently associated with your editing buffers. This gives the appearance of the ability to edit files while they reside on a remote machine, without needing to use NFS to mount the remote filesystems across the relatively slow PPP link.

To use NFS over dialup links, increase the *timeo* timeout parameter in **/etc/fstab** to as much as 750 (i.e. 75 seconds) to accommodate network delays such as

[17] CCITT V.32 (9600) or V.32bis (14400) carrier, with V.42 error correction and V.42bis data compression, and the host serial interface latched at 38400

[18] Written by Andy Norman <ange@hplb.hpl.hp.com>, available from ftp.gnu.ai.mit.edu:/ange-ftp/ange-ftp.el.Z, or via anonymous UUCP from osu-cis, or via anonymous PPP/FTP from Morning Star Technologies.

establishing modem connections.

8.2.4. Preparing the user community for thin wire connectivity

It is important to prepare your user community so that they are familiar with and understand the limitations of low-speed IP connectivity. Test critical applications with your modems to verify their suitability in the new environment before widespread deployment. Often, an application can be more cleverly partitioned between 'local' and 'remote' parts to reduce the communication bandwidth needed for satisfactory operation. Current modem technology simply does not provide Ethernet-speed connectivity over dialup lines. If the users (or the writers of applications) appreciate the existence of connectivity itself, their expectations are more likely to be satisfied.

9. Cabling

Most systems and modems use the EIA RS-232[19] signaling conventions for their asynchronous serial ports. In the tables below, there is no difference between the pins referred to as 'DTE pins' or 'Modem pins', nor between 'Local pins' and 'Remote pins', except notational convenience. If any of the following cables have the same physical connector type on each end (both DB-25 or both MiniDIN-8), they are completely symmetrical and may be flipped end-for-end with no effect.

9.1. Asynchronous Modem cables

If your system is to be connected to an asynchronous modem, and if the system uses DB-25 connectors, the pins on the system (DTE) end of the cable should be connected straight through to the same-numbered pins on the DB-25 modem end.

[19] Electronic Industries Association, EIA Standard RS-232-C, 'Interface Between Data Terminal Equipment and Data Communications Equipment Employing Serial Binary Data Interchange', August 1969.

In order of importance for proper operation, the pins you should connect are

DTE DB-25 pin	RS-232 Signal	Modem DB-25 pin	Direction
1	Protective Ground	1	both
7	GND (Signal Ground)	7	both
2	TD (Transmit Data)	2	from DTE
3	RD (Receive Data)	3	from Modem
6	DSR (Data Set Ready)	6	from Modem
20	DTR (Data Terminal Ready)	20	from DTE
4	RTS (Request To Send)	4	from DTE
5	CTS (Clear To Send)	5	from Modem
8	DCD (Data Carrier Detect)	8	from Modem
22	RI (Ring Indicator)	22	from Modem

NeXTstation serial ports have MiniDIN-8 connectors. A cable to connect a NeXTstation to a DB-25 asynchronous modem looks like[20]

DTE MiniDIN-8 pin	RS-232 Signal	Modem DB-25 pin	Direction
1	DTR	20	from DTE
2	DCD	8	from Modem
3	TD	2	from DTE
4	GND	7	both
5	RD	3	from Modem
6	RTS	4	from DTE
7	*N/C*	*N/C*	*N/C*
8	CTS	5	from Modem
N/C	DSR	6	from Modem
N/C	RI	22	from Modem

9.2. Asynchronous ‘Null-Modem’ cables

If your system is to be connected directly to another system, the pins on the local system’s end of the cable should be connected with the symmetric-function pins on the remote system’s end. This is commonly called a null-modem cable.

[20] ‘Network and System Administration’, Chapter 13 ‘Attaching Peripheral Devices’, section on ‘Serial Port Connector and Cable Specifications’, pp. 186–188, NeXT Computer Inc., 1990.

In order of importance for proper operation, the pins you should attach between two systems with DB-25 connectors are

Local DB-25 pin	Local RS-232 Signal	Remote DB-25 pin	Remote RS-232 Signal
1	Protective Ground	1	Protective Ground
7	GND	7	GND
2	TD	3	RD
3	RD	2	TD
8	DCD	20	DTR
20	DTR	8	DCD
4	RTS	5	CTS
5	CTS	4	RTS

NeXTstation serial ports have MiniDIN-8 connectors. A null-modem cable to connect a NeXTstation to another system that has DB-25 connectors looks like

Local MiniDIN-8 pin	Local RS-232 Signal	Remote DB-25 pin	Remote RS-232 Signal
1	DTR	8	DCD
2	DCD	20	DTR
3	TD	3	RD
4	GND	7	GND
5	RD	2	TD
6	RTS	5	CTS
7	*N/C*	*N/C*	*N/C*
8	CTS	4	RTS
N/C	DSR	6	DSR
N/C	RI	22	RI

A null-modem cable to connect a pair of NeXTstations looks like

Local MiniDIN-8 pin	Local RS-232 Signal	Remote MiniDIN-8 pin	Remote RS-232 Signal
1	DTR	2	DCD
2	DCD	1	DTR
3	TD	5	RD
4	GND	4	GND
5	RD	3	TD
6	RTS	8	CTS
7	*N/C*	7	*N/C*
8	CTS	6	RTS

10. Flow control

If your UNIX system supports it, use out-of-band 'hardware' (RTS/CTS) flow control between your system and your modem, or between two systems over a null-modem cable. On most systems, specify 'crtscts' either on the *pppd* command line, or in the Optional Parameters field of the line in **Devices**(5). On a NeXT, specify 'cufa' in the Device field in **Devices**(5) and use the name 'ttydfa' in **/etc/ttys**.

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**(5), and use 'cua' on a NeXT.

The daemon's default behavior is to use no flow control.

SunOS 4.1.1 implements hardware flow control on the native CPU serial ports only for transmission[21]. Workstations running this OS respect the modem's use of CTS (Clear To Send) to indicate its readiness to receive characters from the host, but do not use RTS (Request To Send) to throttle the modem's (incoming) data stream to the host. This means that when presented with a heavy incoming traffic load, the Sun's native CPU serial ports will be overloaded, and the console will report a series of occasional 'zs*x*: silo overflow' errors. This is unfortunate but unavoidable by *pppd*, introduces no errors into the user data being transferred across the network, and does not significantly affect the PPP link's throughput or responsiveness.

11. Modem parameters

[21] See the Output Modes section of **termio**(4) and the IOCTLS section of **zs**(4s).

11.1. General Recommendations

You should connect your modem to your computer at the highest asynchronous serial speed that both can support (typically 38400 baud, as found in **/usr/include/sys/ttydev.h** or **/usr/include/sys/termio.h**), 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 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 PPP installation allows inbound and outbound connections over any available modem. To allow bidirectional use of the modem, plus use by other applications (such as *uucp*, *tip*, *cu*, or remote logins), 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 both your computer and 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 **crtscts** 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 accomodate 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.

11.2. Specific examples

In the file **Examples/Dialers.ex** you'll find dialer descriptions for several types of modems, including register setup and switch setting summaries. Since your modems will likely be used for purposes besides PPP (e.g. UUCP or interactive users), it's best to set their default parameters to accommodate dial-in applications and have outgoing UUCP or PPP dialers change them if necessary. The PPP-specific register settings in **Dialers.ex** ensure that an otherwise general-purpose modem will work as well as possible with PPP, for the duration of the PPP session.

In the examples below, we'll show how to configure a typical modern modem, including a picture of its working register settings and a command string that will set it up correctly. We'll also describe what each of those register settings means, so that if you use a different type of modem, you should be able to set it up similarly.

We find that a Telebit T1600 works well with the following register settings:

```
at&v
T1600 - Version LA1.00  - Active Configuration
 B1  E1  L0  M0  Q2  T   V1  X12  Y0
&C1 &D3 &G0 &J0 &L0 &Q0 &R3 &S1 &T4 &X0
S000:1  S001=0   S002=43  S003=13 S004=10 S005=8 S006=2  S007:120
S008=2  S009=6   S010=14  S011=70 S012=50 S018=0 S025=5  S026=1
S038=0  S041=0   S045=0   S046=0  S047=4  S048:1 S050=0  S051:6
S056=17 S057=19  S058:2   S059:15 S060=0  S061:0 S062=15 S063:2
S064=0  S068=255 S069=0   S090=0  S093=8  S094=1 S100=0  S102=0
S104=0  S105=1   S111=255 S112=1  S180=2  S181=1 S183=25 S190=1
S253=10 S254=255 S255=255
OK
```

You can make a T1600 work like this by giving it the following command:

```
at &f s0=1s7=120s48=1s51=6s58=2s59=15s61=0s63=2tq2x12&c1&d3&s1 &w
```

You can make a Telebit QBlazer work like this by giving it the following command:

```
at &f s0=1s7=120s48=1s51=6s58=2s59=15s61=0s63=2tq0x4&c1&d3&s1 &w
```

You can make a Telebit T3000 work like this by giving it the following command:

```
at &f9 s0=1s7=120s48=1s51=6s59=15s61=0s63=1tq2x12&c1&d3&s1 &w
```

The T1600 string accomplishes the following:

&f	Loads factory default settings.
s0=1	Answer after one ring.
s7=120	Wait 120 seconds for a valid carrier tone to be sent from the remote modem.
s48=1	Compare all eight bits when checking for control characters.
s51=6	Latch the DTE interface at 38400 bps.
s58=2	Use full-duplex RTS/CTS flow control. When RTS is OFF, the modem will not send data to the local DTE. When RTS is ON the modem sends data to the local DTE.
s59=15	Use the following result code suffixes: Possible link protocol suffixes: nothing, REL, or LAPM Possible compression suffixes: nothing or COMP
s61=0	Local reaction to a break signal on the serial interface is the same as specified in S63.

s63=2 When a break signal is transmitted by the local DTE during an error controlled connection, the modem will discard any buffered data and send the break.

q2 Modem reports result codes when originating a call, but does not return result codes when answering a call.

t Modem uses DTMF tone dialing.

x12 Result codes include the following:
BUSY, DIALING, ERROR, NO CARRIER, NO DIALTONE, OK, RING, RRING, CONNECT 300, CONNECT 1200, CONNECT 2400, CONNECT 4800, CONNECT 7200, CONNECT 9600, CONNECT 12000, CONNECT 14400, CONNECT 19200, and CONNECT 38400.

&c1 When a carrier is detected from the remote modem, the DCD signal is turned ON after the CONNECT result code is sent to the DTE. The DCD signal is turned OFF when the carrier is dropped.

&d3 The modem resets and enters command mode when the DTR signal is switched from ON to OFF, recalling the current user configuration parameters from nonvolatile memory.

&s1 DSR is ON after the answer tone is detected and stays ON throughout the connection.

&w Save the current configuration settings to nonvolatile RAM.

If you have some other sort of modem that's not described here, you should configure it with similar functions (e.g. RTS/CTS flow control) using that modem's own command set and register semantics. Several modem types are described in **Dialers.ex**.

12. Interoperability with other PPP implementations

Morning Star PPP has been tested with several other PPP implementations, including the Telebit NetBlazer, FTP Software's PC/TCP, Brixton Systems' BrxPPP, the cisco AGS and CGS, the 3Com NETBuilder, the Network Application Technology LANB-820, Merit's PPP servers, KA9Q, and the Perkins/Clements/Fox/Christy free PPP for SunOS. It has also run successfully (in SLIP mode) with the SLIP implementations in several types of terminal servers, PCs, Macintoshes, and other workstations.

12.1. The Telebit NetBlazer

Morning Star PPP has been successfully tested with version 1.41x11 of Telebit's NetBlazer software. If your NetBlazer is running an earlier release, you may need to put *novjcomp* on the *pppd* command line.

12.1.1. Dialing into a NetBlazer from a UNIX system

This configuration can occur when several remote workstations dial into a NetBlazer serving as a hub for a LAN, or when connecting a remote workstation or LAN to the Internet through an IP connectivity vendor's NetBlazer POP.

12.1.1.1. Configuring the NetBlazer to receive the call from the UNIX system

The NetBlazer may be set up as if the UNIX machine were just another remote 'dynamic-interface' NetBlazer, using the PPP encapsulation protocol in 'packet mode'.

Configure the NetBlazer as described in chapter 3 of the NetBlazer Reference Manual, noting particularly the discussion of the *ipdial* command in the section on Setting Up Remote IP Destinations. When queried for 'Name of dial IP interface to add', respond with the name of the UNIX system. Morning Star PPP's authentication mechanisms have not been tested with the NetBlazer's 'crypto handshake'. Use PPP rather than SLIP, and specify 1500 for the MTU. When asked to enter the password for the dial-in user for use by the UNIX system, note that password for later inclusion in the UNIX system's **Systems** entry that describes how to dial into this NetBlazer.

12.1.1.2. Configuring the UNIX system to call the NetBlazer

In your UNIX system's **Systems** file, the username provided by the login portion of the chat script will be the name of the UNIX system itself. The password will be the one provided when arranging for the dial-in user on the NetBlazer.

In your **/etc/ppp/Startup** shell script (called from **/etc/rc.local**), start the Morning Star PPP daemon as you normally would when calling another MST PPP site.

12.1.2. Dialing into a UNIX system from a NetBlazer

This arrangement would occur if a hub NetBlazer needed to connect on demand to several remote workstations.

12.1.2.1. Configuring the UNIX system to receive the call from the NetBlazer

In order to accommodate a NetBlazer dialing into a UNIX system running Morning Star PPP, the following changes must be made to the UNIX system. First, make an entry in **/etc/gettytab** that's a duplicate of the entry you would normally have used, except with the addition of the **:p8:** (no parity) parameter[22] (see **gettytab**(5)), like this:

```
h|std.38400|38400-baud:sp#38400:
w|P.38400|38400-baud:sp#38400:p8:
```

Then change **/etc/ttytab** to use the new entry, like this:

```
#ttya   "/usr/etc/getty std.38400"      dialup  on
ttya    "/usr/etc/getty P.38400"        dialup  on
```

This change in parity will have an adverse effect on the ability of some other users with other machines and other applications to dial into this port. Users of Morning Star PPP can compensate by including the pair

```
"" P_ZERO
```

in the **Systems** entry for dialing into this system. Future releases of the NetBlazer software may include the ability to talk to a typical 7-bit UNIX *getty*, so the **:p8:** switch would become unnecessary, and all normal UNIX dial-ins would work as expected.

The login shell script for the NetBlazer's incoming connection must tell the NetBlazer what it expects to hear ('Packet mode enabled') before starting the PPP LCP option negotiation phase of the connection. Make the file **/etc/ppp/Login-netblazer** look like this:

```
#!/bin/sh

PATH=/usr/bin:/usr/etc:/usr/ucb:/etc:/bin

mesg n
stty -tostop
echo Packet mode enabled
exec pppd `hostname`: idle 130 crtscts
```

Make sure it is executable:

```
# chmod 755 /etc/ppp/Login-netblazer
```

Don't specify *crtscts* if you're on a NeXT system; use 'cufa' for outbound links and **ttydfa** in **/etc/ttys** instead.

If the NetBlazer hangs up before it sees the 'Packet Mode Enabled' welcome message, it could be seeing too much text from the UNIX system first. You can keep that 'user' from seeing the **/etc/motd** by saying

```
# touch ~netblazer-login/.hushlogin
```

[22] The **:p8** *gettytab* parameter works in SunOS 4.1 and later versions, and on other operating systems supported by Morning Star PPP, but a bug in **login** caused **:p8** not to work in SunOS 4.0.

where *netblazer-login* is the name of the Netblazer's entry in the UNIX system's **passwd** database.

12.1.2.2. Configuring the NetBlazer to call the UNIX system

Add the following entry to the NetBlazer's CHAT.TXT file:

```
# For a UNIX system running Morning Star Technologies PPP
#  no crypto challenge/response, longer timeouts, short
#  expect strings login shell script must `echo Packet mode
#  enabled` before `exec pppd`
:unix
e1,20,30,2,in:
u2,5,100,3
s3,5,100,4,\r
e4,10,30,5,word:
o99,Packet mode enabled
p5,5,100,6
s6,5,100,7,\r
e7,20,30,99,Packet mode enabled
o32,in:

s30,5,100,31,\r\r
e31,20,100,32,in:
u32,5,100,33
s33,5,100,34,\r
e34,10,100,35,word:
o99,Packet mode enabled
p35,5,100,36
s36,5,100,37,\r
e37,20,100,99,Packet mode enabled
o100,in:

r99,0
r100,1
#
```

You'll need to reboot the NetBlazer to cause any CHAT.TXT changes to take effect.

When configuring the NetBlazer's dynamic interface to call the UNIX system as described above in the section on **Configuring the NetBlazer to receive the call from the UNIX system**, specify **unix** for the chat script, rather than the default **ics**.

12.2. FTP Software Inc's PC/TCP Network Software for DOS

Morning Star PPP interoperates with version 2.05 of FTP Software's PC/TCP PPP (PC-227) implementation. You must specify *rfc1172-addresses* and *nolqm* on the *pppd* command line.

12.2.1. Getting ready for a DOS PC to dial into a UNIX system

The login script used by the PC should specify the IP addresses of both the UNIX system and the PC, and should specify **passive** and **rfc1172-addresses** for option negotiation. It should not specify an idle timer because PC/TCP is unable to re-open a timed-out connection.

12.2.2. Getting a DOS PC ready to dial into a UNIX system

The PC should be configured as described in FTP Software's manual. No special arrangements are required in order to talk to a UNIX system running Morning Star PPP.

12.3. Brixton Systems' BrxPPP

Morning Star PPP interoperates with version 1.2.1 of Brixton Systems' BrxPPP software for SPARC systems. You must specify either *echolqm* or *nolqm* on the MST *pppd* command line, because BrxPPP doesn't yet support the new style of Link Quality Monitoring, yet doesn't respond with a Protocol-Reject to *pppd*'s initial Link Quality Report message.

12.4. Network Application Technology LANB-820

Morning Star PPP interoperates with Network Application Technology's LANB-820 router. You must specify the *nomru*, *nomagic*, *noipaddress*, and *nolqm* options on the *pppd* command line, or the NAT router will send malformed LCP Configure-Ack messages to MST PPP.

13. Getting Help

Call Morning Star Technologies' technical support staff at +1 800 558 7827 or +1 614 451 1883, or address electronic mail to them as

```
Support@MorningStar.Com
```

Join the electronic mailing list of Morning Star PPP users to share hints, tips, critiques, feature suggestions, questions about configuration quirks, and anything else related to the product. Morning Star will announce new releases and newly-supported hardware on the list. Our engineering, support, and marketing departments are well represented on the list, so that we can listen to your concerns and offer help too.

To send a message to the other members of the list, address it to

```
PPP-Users@MorningStar.Com
```

To request addition to the list, write to


```
PPP-Users-request@MorningStar.Com.
```

14. Getting updated software and documentation

Install entries in your **Systems** file, as found in **/etc/ppp/Systems.ex**, so that you can dial into 137.175.42.42 using either V.32 or PEP modems. Then start a *pppd* as described in **/etc/ppp/Startup.ex**. Access to the archive is also available via the Internet by connecting to ftp.morningstar.com.

Use 'ping' or 'ftp' to 137.175.42.42 to cause pppd to dial the Morning Star Technologies support machine. If you use ftp, it will fail to connect because the IP session addresses on your local machine are not assigned until login is complete, so you will need to run ftp again. Open an ftp session to 137.175.42.42 (or to ftp.morningstar.com if you will be coming in over the Internet rather than dialing directly). When you are asked, provide the user name 'anonymous' and provide your e-mail address for the password. Navigate the archive using the standard ftp commands, and retrieve the files you need. When you request a file that is protected, you will be denied access until you provide the correct group password for that file. While your software is under warranty, or while you have a current support contract, you can get a valid password from the Morning Star Technologies support hotline.

In the example below, the requestor wanted to get the current MST PPP distribution for SPARC systems, but received the "550 ... Permission denied" response before identifying herself to the FTP server. Upon calling MST's support hotline, since her support agreement was effective through December of 1992, she found that she could use the group password 'blURfg3l' to unlock the software. That password will be valid through the end of her support agreement, and she will use it at her convenience to get new software from time to time.

Here is a typical session:

```
5:44pm> 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:jru): anonymous
331 Guest login ok, send e-mail address as password.
Password:
230-Welcome to the Morning Star Technologies
    support server at Thu Dec 26 17:44:13 1991
230-
230 Guest login ok, access restrictions apply.
ftp> cd pub/ppp
250 CWD command successful.
ftp> bin
200 Type set to I.
```

```
ftp> hash
Hash mark printing on (8192 bytes/hash mark).
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 2801
-rw-r--r-- 1 59 staff    1365 Jul 15 21:26 Free-SLIP-and-PPP
-rw-rw-r-- 2 59 staff    2884 Dec 17 05:01 MST-PPP-At-A-Glance
-rw-r--r-- 2 56 staff  269671 Aug 16 20:13 PPP-Literature.ps.Z
-rw-r--r-- 1 59 staff  371586 Sep  6 16:46 PPP-Literature.ps.Z.uue
-rw-r--r-- 2 56 staff    4018 Aug 22 13:24 PPP-Literature.txt
-rw-rw-r-- 2 59 staff    3784 Aug 22 12:27 PPP-Sun-comp.newprod
-rw-r--r-- 2 59 staff    3049 Aug 14 13:55 PPP-Sun-press-release
-rw-r----- 1 59 sparc  400445 Dec 26 18:16 mst-sparc-ppp-1.2.tar.Z
-rw-r----- 1 59 ultrix 400589 Dec 26 21:54 mst-ultrix-ppp-1.2.tar.Z
-rw-rw-r-- 1 59 daemon   8397 Oct 30 20:22 ppp-button.ps
-rw-rw-r-- 2 59 daemon 121881 Dec 24 02:33 ppp-doc.shar.Z
-rw-r--r-- 1 51 staff  425209 Jul 16 02:02 ppp-mail-archive.Z
-rw-rw-r-- 1 51 staff   22840 Oct 28 12:45 ppp-refcard-back.ps.Z
-rw-rw-r-- 1 51 staff   15814 Oct 28 12:45 ppp-refcard-front.ps.Z
-rw-rw-r-- 3 59 staff  176186 Dec 20 21:31 ppp-users-mail-archive
drwxrwsr-x 2 59 staff     512 Nov 16 05:59 rfc
-rw-rw-r-- 2 59 daemon 112901 Dec 24 02:33 user-guide-1up.ps.Z
-rw-rw-r-- 2 59 daemon 121001 Dec 24 02:33 user-guide-2up.ps.Z
-rw-rw-r-- 1 59 staff  164580 Dec 19 12:25 user-guide-2up.ps.Z.uue
-rw-rw-r-- 1 59 daemon 121395 Dec 24 02:33 user-guide-book.ps.Z
#
226 Transfer complete.
1510 bytes received in 0.77 seconds (1.9 Kbytes/s)
ftp> get user-guide-2up.ps.Z
200 PORT command successful.
150 Opening BINARY mode data connection
    for user-guide-2up.ps.Z (121001 bytes).
##########################
226 Transfer complete.
local: user-guide-2up.ps.Z remote: user-guide-2up.ps.Z
121001 bytes received in 0.74 seconds (1.6e+02 Kbytes/s)
ftp> get mst-sparc-ppp-1.2.tar.Z
200 PORT command successful.
550 mst-sparc-ppp-1.2.tar.Z: Permission denied.
ftp> quote site group sparc9212
200 Request for access to group sparc9212 accepted.
ftp> quote site gpass blURfg3l
200 Group access enabled.
ftp> get mst-sparc-ppp-1.2.tar.Z
200 PORT command successful.
150 Opening BINARY mode data connection for
    mst-sparc-ppp-1.2.tar.Z (400445 bytes).
###########################################
###########################################
```



```
226 Transfer complete.
local: mst-sparc-ppp-1.2.tar.Z remote: mst-sparc-ppp-1.2.tar.Z
400445 bytes received in 2.7 seconds (1.5e+02 Kbytes/s)
ftp> quit
221 Goodbye.
5:45pm>
```

Because the scheme used to assign IP addresses to incoming anonymous PPP calls is based on their incoming port, a reconnection attempt after an idle timeout or a line failure may be unsuccessful. If the second call, intended to reconnect the TCP stream, comes in on a different TTY port on MST's PPP server, the calling system will be assigned a different IP address, and the two systems will have no way of reconnecting the TCP streams. Just exit the ftp and start over.

Another characteristic of the auto-dial scheme may cause your first FTP client attempt to be unable to connect with the MST FTP server, even though your PPP daemon is successfully connected. Simply start a new FTP session and continue with the rest of your business.

If you want to use Morning Star's PPP dialups to test your own configuration, you might be surprised or disappointed by some activities that are disallowed by the filters and timers that are in place on the link. The server imposes a 120-second idle timer. Only FTP and ICMP packets are allowed to cross the link. If you try to telnet to 137.175.42.42 you won't get a response, but that's not necessarily because your end is misconfigured. Try using ping(8) or ftp(1) to test the link instead. Also, you may be disappointed that we don't allow IP routing to the rest of our network (nor to the general Internet) across these anonymous dialup links, but you shouldn't be surprised.

If you're using MST PPP on SCO Unix V.3.2, your system's IP doesn't work well with the address assignment scheme on MST's anonymous-PPP server, nor on Merit or on PSI's Host-DCS service either. To prepare for a server to assign an address, MST PPP configures the 'local' end of the point-to-point link as 0.0.0.0. After starting the daemon to call such a server, you must manually set a non-zero address, as below:

```
# pppd :137.175.42.42 auto
# ifconfig du0 137.175.42.3 137.175.42.42
# ftp 137.175.42.42
^C
# tail -f /usr/adm/pppd.log (wait until it says 'PPP connected')
^C
# ftp 137.175.42.42
Connected to 137.175.42.42
220 remora FTP server (Version 6.14 Thu Dec 26
    11:24:28 EST 1991) ready.
Name (137.175.42.42:jru):
```

If the call fails or times out, the interface must be ifconfig'd by hand before retrying.

15. Credits

The *pppd* daemon was written by Karl Fox <karl@MorningStar.Com>, with assistance from Bob Sutterfield <bob@MorningStar.Com>, Kim Toms <kim@MorningStar.Com>, and Steve Wilson <steve@MorningStar.Com>. The documentation was written by Bob Sutterfield and revised by Mark Seiden <mis@seiden.com>. Parts of the PPP protocol engine are derived from code written by Drew Perkins <ddp@andrew.cmu.edu>, now <ddp@interstream.com>. The TCP header compression feature is derived from code written by Van Jacobson <van@helios.ee.lbl.gov>. The CHAP feature uses the RSA Data Security, Inc. MD5 Message-Digest Algorithm.

16. Copyright Information

NAME

tun – IP network tunnel driver

CONFIG

pseudo-device tun[*n*]

SYNOPSIS

#include <sys/tun.h>
open("/dev/tun*n*", mode);

DESCRIPTION

When IP packets are written to **/dev/tun***n*, they will be received by the kernel's IP layer on the network interface **tun***n*. When the kernel's IP layer sends packets to the IP interface **tun***n*, they will be available for reading on **/dev/tun***n*.

Instead of having hardware and an associated kernel interface that support network functions, the **tun** driver allows a network interface to be implemented as a user-space process. While talking to the same set of tunnel drivers on the same system, different network interface processes can implement different IP encapsulation methods, such as RFC 877 for use over CCITT X.25-based public data networks, or RFC 1055 SLIP or RFC 1331/1332 PPP for use over dedicated lines and dialup modems.

The **tun** driver provides support for a pair of devices collectively known as an *IP tunnel*. The two devices comprising a tunnel are known as the *inbound* and *outbound sides*, similar to the pairing between **/dev/tty***n* (the inbound terminal) and **/dev/cu***n* (the outbound 'auto-call unit' available on many systems). The outbound side's minor device number is that of the inbound side plus 128, though they together appear to IP as one interface. If both the inbound and outbound sides of a tunnel device are open, packets received from IP are delivered to only the inbound side.

If a TCP packet received from IP is part of a telnet, rlogin, or FTP command stream, it will be put in a *fast queue*. All packets in the fast queue are delivered to the user before any packets in the normal queue.

IOCTLS

A few special **ioctl**s are provided for use on the **/dev/tun***n* devices to supply the functionality needed by applications programs to emulate real hardware interfaces. The complete list of supported **ioctl**s is:

TUIOSPTPT Set or clear the IFF_POINTOPOINT in the associated network interface. **TUIOSADRMD** Set or clear 'address mode', in which packets read are prefaced with four octets containing the destination IP address in network byte order. The third argument is a pointer to an integer

containing either a zero or a one, indicating whether 'address mode' should be cleared or set, respectively. If both 'address mode' and 'packed buffer mode' are set, each packet's length will come first, followed by the packet's destination address, followed by the packet itself.

TUIOGADRMD Get the current status of 'address mode'.

TUIOSPKBMD Set or clear 'packed buffer mode' where multiple packets are encoded in single read/write buffers. The third argument is a pointer to an integer containing either a zero or a one, indicating whether 'packed buffer mode' should be cleared or set, respectively. If set (1), each packet is preceded by four octets representing the next packet's length in octets. The following packet will then be aligned to the next multiple of four octets. If cleared (0), packets will be delivered one per read(3) from the tunnel device. If both 'address mode' and 'packed buffer mode' are set, each packet's length will come first, followed by the packet's address, followed by the packet itself.

TUIOGPKBMD Get the current status of 'packed buffer mode'.

TUIOSPKMAX Set the max number of IP frames to send back in a packet buffer read.

TUIOGPKMAX Get the PKMAX value.

TUIOSPKPAD Set the number of long word zeroes to put on the front of each packet read in packed buffer mode.

TUIOGPKPAD Get the number of pad words.

TUIOSNAME Set the interface name (may only be invoked by the super-user).

TUIOGNAME Get the interface name.

FIONBIO Set or clear non-blocking mode for I/O operations.

EXAMPLES

```
#include <sys/tun.h>
int tun_fd = -1, len;
char *packet;

tun_fd = open("/dev/tun0", O_RDWR);
ioctl(tun_fd, TUIOSNAME, "du");
len = read(tun_fd, packet, size);
write(tun_fd, packet, len);
```

ERRORS

If a packet is delivered to the interface for an address family other than AF_INET, EAFNOSUPPORT will be returned.

FILES

/dev/tun0 through **/dev/tun127** ‘inbound’ tunnel devices
/dev/tun128 through **/dev/tun255** ‘outbound’ tunnel devices

SEE ALSO

if(4n), ppp.Auth(5), ppp.Devices(5), ppp.Dialers(5), ppp.Filter(5), ppp.Systems(5), pppd(8), RFC 1331, RFC 1332, RFC 1144, RFC 1055, RFC 877.

COPYRIGHT INFORMATION

NAME

Auth – PPP authentication file format

DESCRIPTION

The file **/etc/ppp/Auth** (or **/usr/lib/ppp/Auth** on SCO systems) contains values used by Morning Star PPP’s implementation of the link-level authentication protocols, PAP (Password Authentication Protocol) and CHAP (Challenge Handshake Authentication Protocol). This implementation conforms to the May 1992 draft of *The PPP Authentication Protocols*.

FORMAT

Each authentication specification is on its own single line of up to 1023 characters. 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 PAP authentication, the first word on the line is used for the *peer-id* during negotiation, and the second word is used for the *password*. If *pppd* is using CHAP authentication, the first word on the line is used for the *name* during negotiation, and the second word is used for the *secret*. The default value used for *name* by *pppd* is the *hostname*(1) of the machine it is running on.

In the midst of either the name/secret or peer-id/password string, ^*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 of our peer, as found in the Name field of a received CHAP Challenge or Response message, or as found in the Peer-ID field of a sent or received PAP Authenticate-Request message. This is often, but not necessarily, the peer’s hostname.

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.
```

SECURITY CONCERNS

The file **/etc/ppp/Auth** should be mode 600 or 400, and owned by root.

SEE ALSO

tun(4), ppp.Devices(5), ppp.Dialers(5), ppp.Filter(5), ppp.Systems(5), services(5), pppd(8), RFC 792, RFC 1331, RFC 1332, RFC 1144.

Brian Lloyd and William A. Simpson, ''The PPP Authentication Protocols,'' Internet Draft, May 1992.

COPYRIGHT INFORMATION

NAME

Devices – PPP physical device description file format

DESCRIPTION

The file **/etc/ppp/Devices** (or **/usr/lib/ppp/Devices** on SCO systems) associates dialer types with physical devices and speeds. *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, or the name of an external dialer program.

device
: The name of the device in the **/dev** directory (**ttya**, **cua**, etc.). Device names for SnapLink connections are followed by a slash and the port number in use (**rsd2a/0**, **rrz4a/2**, etc.).

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 (as found in <sys/ttydev.h>) or must begin with them (2400, 38400, 19200-PEP, etc.), or must be speeds of which the SnapLink hardware is capable (9600, 56000, 64000, 1536000, etc.)

optional parameters
: Any special handling for this device. Currently supported values include

crtscts
: Specifies that the 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. On NeXT systems, enable **ttydfa** or **ttydfb** in **/etc/ttys** instead. On Silicon Graphics

systems systems, use **/dev/ttyf2** for a modem with hardware flow control instead.

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.

internal-clocking
: The SnapLink will provide the synchronous clock signal. By default, it expects the modem, CSU/DSU or modem eliminator to provide the clock signal. Internal-clocking cannot be used with RS-232 cables on the SnapLink.

32-bit-fcs
: The SnapLink 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. 32-bit FCS is only available when running synchronous PPP on the SnapLink.

min-flags=*minflags*
: The number of additional HDLC flag characters the SnapLink should insert between data frames. The default and minimum is 2; the maximum is 16.

EXTERNAL DIALER

The external dialer program is run with the following arguments:

device name
: The contents of the Device field from the Devices entry.

speed
: The contents of the Speed field of the Systems and Devices entries.

telephone number
: The contents of the Phone Number field of the Systems entry.

optional parameters
: Copied from the Optional Parameters section of the Devices entry.

If the external dialer program exits with status 0, then the dial attempt is considered to have succeeded. Any other exit status indicates a failure.

EXAMPLE

```
#
#       Devices - PPP devices file
#
#Dialer    device    speed      Optional parameters
T2500-PEP  cua       19200-PEP  crtscts
T1600      cub       38400      crtscts
Direct     rsd0a/0   1536000    internal-clocking
Oddball    rsd0a/1   65535      internal-clocking cua 9600 5551212
```

In the last line of this example, the 64Kb synchronous modem on the SnapLink's port 1 has an asynchronous dialer interface attached to the workstation's port 'a'. The Systems line would look like

```
host Oddball rsd0a/1 65535 0
```

There must be a program (or an executable shell script) called **/etc/ppp/Oddball** that dials the modem when invoked as

```
Oddball rsd0a/1 65535 0 internal-clocking cua 9600 5551212
```

A warning message will be printed for each unrecognized optional parameter if the *debug* level is 2 or more.

CAVEATS

The **crtscts** flag is unsupported on the daemon's command line or in **Filter** on some systems, because hardware flow control is configured instead by choosing a specific tty device. On a NeXTstation, use the devices **/dev/cufa** and **/dev/ttydfa** for hardware-flow-controlled dialout and dialin connections, respectively. On Silicon Graphics systems systems, use **/dev/ttyf2** for a modem with hardware flow control (see *serial*(7)).

The external dialer is invoked as **root**, so you should take appropriate security precautions with its content and file protection.

SEE ALSO

tun(4), ppp.Auth(5), ppp.Dialers(5), ppp.Filter(5), ppp.Systems(5), pppd(8), RFC 1331, RFC 1332, RFC 1144, RFC 1055.

COPYRIGHT INFORMATION

NAME

Dialers – PPP dialer description file format

DESCRIPTION

The file **/etc/ppp/Dialers** (or **/usr/lib/ppp/Dialers** on SCO systems) describes how to dial each type of modem attached to the UNIX system that is to be made available for outbound PPP 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.

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 '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 the log file.

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 carat character (ASCII 0x5e)

^<character> — Send or receive the single character Ctrl-<character> (ASCII 0x00 through 0x1f)

\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)

\c — Don't append a carriage return character after sending the preceding string (send only)

\q — Don't print succeeding send strings (e.g. a password) in any debugging or logging output. Subsequent **\q** sequences toggle ‘quiet’ mode.

\T Insert the telephone number (found in the fifth field of **Systems**) here

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
#
T2500-PEP \
        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\sFAST
#
USRv32bis \
        ABORT ERROR ABORT NO\sANSWER ABORT NO\sCARRIER \
        ABORT BUSY ABORT RRING\r\n\r\nRRING\r\n\r\nRRING \
        ABORT NO\sDIAL\sTONE TIMEOUT 5 "" AT&F OK-ATQ0-OK \
        ATB0E0X7&B1&H1&I0&K3&R2&S1 OK-AT-OK \
        ATS01=1S02=255S19=0 OK-AT-OK ATDT\T TIMEOUT 30 CONNECT
```

SEE ALSO

tun(4), ppp.Auth(5), ppp.Devices(5), ppp.Filter(5), ppp.Systems(5), pppd(8), RFC 1331, RFC 1332, RFC 1144, RFC 1055.

COPYRIGHT INFORMATION

NAME

Filter – PPP packet filter specification file format

DESCRIPTION

The file **/etc/ppp/Filter** (or **/usr/lib/ppp/Filter** on SCO systems) describes how on-demand PPP links are to be managed. By default, any type of packet causes the link (if down) to be brought up (connected to its remote end); any packet is allowed to traverse the link; and any packet is sufficient to reset the idle timer, expiration of which would cause the link to be shut down. This combination is not always appropriate behavior, so the filter file allows individual control based on the packet type and its source or destination. These selection criteria may be specified for any of the three phases of operation: bringing up the link, passing packets on the link, and shutting down the 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).

If a line begins with a hostname or IP address or the special word ‘default’, that line is considered to be the beginning of a new set of filtering specifications. 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 by *pppd* in response to a particular packet:

bringup Describes those packets that will cause a call to be placed and a connection initiated. 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 the link to be brought up. Any packet that is not passed is optionally logged, then discarded.

keepup Describes packets that will reset the idle timer, 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, *pppd* takes the indicated action and ignores the rest of the listed stanzas (i.e. inclusive *or* with shortcut evaluation).

Stanzas may contain hyphen-separated ranges of TCP or UDP port numbers, numbers representing ICMP message types or codes (which can be found in **<netinet/ip_icmp.h>**, service names corresponding to entries in **/etc/services**, or names or IP addresses of hosts or networks, or the special keyword 'all', which is the default for all actions except 'log', where the default is '!all'. (Usually, it is unnecessary to use 'all'; as a convenience, *pppd* automatically adds a '!all' at the end of a stanza list if the last stanza *is not* negated, and add an 'all' 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 logged, 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 subnet mask must also be specified if it is of a different size than the default for that class of network. The netmask and additional 'and' conditions within a stanza are separated by slashes ('/'). 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 written to the log file.

Since TCP data streams are opened when the initiator sends a SYN packet to the intended recipient, *pppd* can distinguish between outbound (sent from this host) 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.

EXAMPLES

The following **Filter** file describes the default behavior of *pppd*, either in the absence of a filter specification file or in the case of an empty file:


```
#       Filter - PPP configuration file,
#       binding packet types to actions.
#       Describes the default behavior of the daemon:
default bringup all pass all keepup all log !all
```

The default is no restriction of packets and no logging.

The following **Filter** file instructs the daemon 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 -       PPP 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
```

The following **Filter** file may be used at a 'firewall' gateway. It allows NTP, DNS, NNTP, mail, finger, and ICMP traffic to pass unhindered. Telnet and FTP are allowed only for outbound sessions, and all other traffic is blocked.

```
default bringup !ntp !3/icmp !who
         keepup  !ntp !3/icmp !who
         pass    ntp domain nntp smtp finger icmp
                 !telnet/syn/recv telnet
                 !ftp/syn/recv ftp ftp-data
```

RECOMMENDATIONS

Simpler filter specifications allow *pppd* 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 log file will grow extremely large in a short period of time. Also, if the daemon is running on a diskless workstation or if the log file is on a NFS-mounted file system, excessive amounts of logging information will drastically impede the daemon's ability to process at high packet rates. Remember, NFS writes are synchronous.

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** 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'. Similarly, some systems' older **/etc/services** files, as distributed by the manufacturer, list NTP as a TCP service. When the current UDP NTP implementation was installed on your system, the administrator may have left the old 123/tcp entry along with the correct 123/udp. The correct solution is to remove the 123/tcp entry from **/etc/services**. A workaround would be to specify 123/udp in **Filter**.

DEC ULTRIX 4.2 and some other systems may have no entry for FTP's data socket in their **/etc/services** file. If you want to log the bulk data connections as well as the control connections, you'll need to either add an entry for 'ftp-data' to **/etc/services**, or use '20/tcp' explicitly in **Filter**. The former is preferable because it will cause the log file entry to contain the symbolic name ('ftp-data') rather than the socket/protocol notation.

SEE ALSO

tun(4), ppp.Auth(5), ppp.Devices(5), ppp.Dialers(5), ppp.Systems(5), services(5), pppd(8), RFC 792, RFC 1331, RFC 1332, RFC 1144, RFC 1055.

COPYRIGHT INFORMATION

NAME

Systems – PPP neighboring systems description file format

DESCRIPTION

The file **/etc/ppp/Systems** (or **/usr/lib/ppp/Systems** on SCO systems) describes how to connect with neighboring systems via PPP.

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.

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 date 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 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.

The retry delay time may be followed by a hyphen and 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.

During the delay following an unsuccessful call, any level 7 debugging messages written to **pppd.log** will have the message 'dial failed' appended.

device
: If set to ACU, any device in **Devices** with a matching speed may be used. The device's dialer chat script will be executed first, followed by the **Systems** chat script.

If set to the name of a device in the **/dev** directory (**ttya**, **cua**, etc.), then there must be a corresponding **Direct** entry in **Devices**, **Dialers** will not be consulted, and only the **Systems** chat script will be executed.

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 (2400, 38400, 19200-PEP, etc.).

phone number
: The value to replace the **\T** escape sequence in the dialer script

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 follow it by sending a 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 the log file.

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 carat character (ASCII 0x5e)

^<character>
Send or receive the single character Ctrl-<character> (ASCII 0x00 through 0x1f)

\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)

\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 Telebit PEP modem with its DTE interface set at 19200 bps. We call host 'nobody' using a V.32/V.42/V.42bis modem that's capable of driving a 38400 DTE, and we are connected to host 'someone' via a direct cable attached to /dev/ttya.

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 'nobody' 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
#
me Any ACU 19200PEP 5551212 login: Pwe word: \qfoObar
nobody Any ACU 38400 5551213 gin: Pthey word: \qbaZz1ng
you Direct cua 38400 0 gin:--gin: Pthem word: \qmeumBle
```

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 desireable 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** 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 the Morning Star PPP User Guide for more details.

SECURITY CONCERNS

The file **/etc/ppp/Systems** should be mode 600.

SEE ALSO

tun(4), ppp.Auth(5), ppp.Sevices(5), ppp.Dialers(5), ppp.Filter(5), pppd(8), RFC 1331, RFC 1332, RFC 1144, RFC 1055.

COPYRIGHT INFORMATION

NAME

pppd – PPP daemon

SYNOPSIS

pppd [options...]

DESCRIPTION

Pppd is a daemon process used in UNIX systems to manage connections to other hosts using PPP, the Point to Point Protocol, or SLIP, the Serial Line Internet Protocol. It uses the UNIX host's native serial ports or the Morning Star SnapLink SCSI-attached high speed serial interface. It communicates with the UNIX kernel's own TCP/IP implementation via the Morning Star IP tunnel driver. (see *tun*(4))

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. still

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 messges. 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.

nodetach — Don't detach from the controlling terminal in 'autocall' mode. When used with 'log -', this can be useful for watching the progress of the PPP session.

log log-file — Append logging messages to log-file (default: **/usr/adm/pppd.log**).

filter filter-file — Look in filter-file for packet filtering and link management information (default: **/etc/ppp/Filter** or **/usr/lib/ppp/Filter** on SCO systems)

debug debug-level — Set the log file 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

5 Other LCP messages, more detailed LQM reports

6 IP interface changes

7 IP message summary

8 More detailed procedure call messages

9 All PPP messages (without framing)

10 Timers, even more detailed procedure call messages

11 Characters read or written

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

exec exec-cmd — Run 'exec-cmd **up** addr args' when the link comes up, and 'exec-cmd **down** addr args' when it goes down. *Addr* is the IP address of the peer, and *args* is the list of arguments given to pppd.

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 log file 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 /dev/tty).

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

poll poll-rate — Set SnapLink polling frequency, in polls per second. Recommend values are 20, 50, or 100 (default 50).

ignore-cd Ignore the state of the CD (Carrier Detect, also called DCD, Data Carrier Detect) signal. This is useful for systems that don't support CD but want to run PPP over a dedicated line.

crtscts Set the 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. This option is not available on all systems. On NeXT systems, enable **ttydfa** or **ttydfb** in **/etc/ttys** instead. On Silicon Graphics systems, use **/dev/ttyf2** for a modem with hardware flow control instead (see *serial*(7)). On SCO systems, **crtscts** cannot be used with either **crtscts-rtsflow** or **crtscts-crtsfl**.

crtscts-rtsflow As above in **crtscts**, but sets both CTSFLOW and RTSFLOW. (SCO UNIX systems only, and cannot be used with either **crtscts** or **crtscts-crtsfl**)

crtscts-crtsfl As above in **crtscts**, but sets CRTSFL and clears both CTSFLOW and RTSFLOW. (SCO UNIX systems only, and cannot be used with either **crtscts** or **crtscts-rtsflow**)

xonxoff Set the 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.

nomagic Disable LCP Magic Number negotiation.

mru mru-size Set LCP Maximum Receive Unit value to mru-size for negotiation. The default is 1500 (256 on Sun-3) for PPP, 1006 for SLIP.

nomru Disable LCP Maximum Receive Unit negotiation, and use 1500 (256 on Sun-3) for our interface.

active Begin both LCP and IPCP negotiations with an Active-Open event (the default).

passive Begin both LCP and IPCP negotiations with a Passive-Open event.

timeout restart-time Set the LCP and IPCP option negotiation restart timers (default 3 seconds).

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 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 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).

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 255, 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.

oldchap
: Interoperate with older CHAP implementations.

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 returned by the *gethostname*(2) system call or the *hostname*(1) command.

SIGNALS

Upon reception of the following signals, *pppd* closes and reopens the log file, then takes the indicated actions:

SIGINT
: Disconnect gracefully from an active session. If in 'autocall' mode, re-read the filter file and 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 in 'autocall' mode, re-read the filter file. If 'up' was specified, attempt to

re-establish the link. Exit if not in ‘autocall’ mode.

SIGTERM Disconnect gracefully from an active session, then exit.

SIGUSR1 Re-read the filter file, and increment the verbosity level for debugging information written to the log file.

SIGUSR2 Re-read the filter file, and reset the debugging verbosity level to the base value (1 unless **debug 0** was supplied on the command line).

SIGALRM Re-read the filter file.

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 **/etc/rc.local**:

```
if [ -f /etc/ppp/Startup ]; then
     /etc/ppp/Startup
fi
```

Then make **/etc/ppp/Startup look like this:**

```
#!/bin/sh

PATH=/usr/etc:/bin:/usr/bin

if [ -f /usr/adm/pppd.log ]; then
        mv /usr/adm/pppd.log /usr/adm/OLDpppd.log
fi

echo -n "Starting PPP daemons:"        >/dev/console

pppd oursystem:backbonesystem auto up
        (echo -n ' backbonesystem')    >/dev/console
pppd oursystem:theirsystem auto idle 120
        (echo -n ' theirsystem')       >/dev/console

echo '.'                               >/dev/console
```

To allow a PPP implementation running on ‘theirsystem’ to dial into ‘oursystem’, insert the following into /etc/passwd on ‘oursystem’:

```
Pthem:?:105:20:Their PPP:/etc/ppp:/etc/ppp/Login
```

where group 20 is the gid of the ppp group which owns /usr/etc/pppd, and **/etc/ppp/Login** is an executable shell script that looks something like


```
#!/bin/sh
PATH=/usr/bin:/usr/etc:/bin
mesg n
stty -tostop
exec pppd `hostname`:
```

RECOMMENDATIONS

Use host names when running **/etc/ppp/Startup** from **/etc/rc.local** 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.

ENVIRONMENT

The environment variable **PPPHOME**, if present, specifies the directory in which *pppd* looks for its configuration files (**Filter** and **Auth** for all connections, along with **Systems**, **Devices**, and **Dialers** if the connection is ‘outbound’). You can specify **PPPHOME** either in the **Startup** script or in an incoming connection’s **Login** script. If **PPPHOME** is not present, *pppd* will expect to find its configuration files in **/etc/ppp/***.

SECURITY CONCERNS

Pppd should be mode 4750, owned by root, and executable only by the members of the group containing all the incoming PPP login ‘users’.

SEE ALSO

tun(4), ppp.Auth(5), ppp.Devices(5), ppp.Dialers(5), ppp.Filter(5), ppp.Systems(5), RFC 1331, RFC 1332, RFC 1172, RFC 1144, RFC 1055.

CREDITS

Parts of the protocol engine are derived from code written by Drew D. Perkins <ddp@andrew.cmu.edu>, now <ddp@interstream.com>. The TCP header compression feature is derived from code written by Van Jacobson <van@helios.ee.lbl.gov>. The CHAP feature uses the RSA Data Security, Inc. MD5 Message-Digest Algorithm.

COPYRIGHT INFORMATION

Troubleshooting Guide

This Troubleshooting Guide suggests solutions to common problems users experience with Morning Star PPP.

Scenario: *Pppd* dials, the modems connect, the modem data lights flash briefly, and then the log file shows ‘Hangup’ or ‘SIGHUP’.

Solution: Increase *pppd*’s debug level to 2 (chat script processing) to see if any error messages are printed by the peer at startup. ‘Login incorrect’ indicates that the login and/or password in the calling machine’s **Systems** file don’t match those in the answering machine’s **passwd** file. ‘/etc/ppp/Login: Permission denied’ indicates a file protection problem with the login shell script. ‘/usr/etc/pppd: Permission denied’ indicates that the PPP user account cannot execute the daemon, probably because the user account is not in the ‘ppp’ group that owns the daemon, while the daemon is mode 4750 for security reasons.

Scenario: *Pppd* dials out, the modems connect, the log file says ‘Call succeeded’, but *ps* shows *pppd*’s state to be ‘connecting’ until the modems disconnect about a minute later.

Solution: Check the log file for messages indicating negotiation problems.

Increase *pppd*’s debugging level to 2 (input framing errors). Look in the pppd.log file for warnings indicating damaged received messages such as ‘Bad FCS received’, ‘Frame too long’, or ‘Missed ALLSTATIONS’.

Increase *pppd*’s debugging level to 11 (show all characters read or written) to see if the peer is sending anything at all. PPP messages should initially look like 7E FF 7D 23 ... 7E. Look for corrupted characters (e.g. 7F instead of FF) or other parity bit problems.

Scenario: PPP connects, and I can ping the other end, but I can’t use telnet, rlogin, or ftp.

Solution: Check the log file for any messages indicating negotiation problems.

Increase *pppd*’s debugging level to 2 (input framing errors). Look in the pppd.log file for warnings from ppp indicating damaged messages such as ‘Bad FCS received’, ‘Frame too long’, ‘Missed ALLSTATIONS’, ‘Bad protocol’, or ‘Short frame received’.

Increase *pppd*’s debugging level to 7 (IP message summary). When telnet is started, make sure a tcp packet is sent. A tcp packet should

be received in response.

Restart *pppd* with the **novjcomp** option. If this fixes the problem, then the peer may be using a buggy IPCP option negotiation algorithm. Try running *pppd* with the **rfc1172-vj** or the **rfc1172-typo-vj** option. If they don’t work, go back to **novjcomp** and complain to the vendor of the peer’s implementation of PPP.

Scenario: PPP connects and everything works, but it is slow and erratic.

Solution: Is your Telebit modem using the PEP protocol? PPP over PEP is normally slow and erratic.

Increase *pppd*’s debugging level to 2 (input framing errors). Make sure that received frames are arriving undamaged and that you aren’t getting an excessive number of FCS errors.

Make sure that the telephone lines are relatively clean. Noisy phone lines can cause FCS or other transmission errors, or can cause V.42 or MNP error correcting protocols to retransmit excessively, or can cause the modems to retrain frequently, slowing the connection.

Make sure the serial port is not running faster than your computer can comfortably handle.

Make sure that flow control is set up properly on the modem and on the serial port it is connected to. If you are using hardware flow control, make sure the cable connects RTS and CTS (RS-232 pins 4 and 5).

Scenario: When I use telnet, ping or some other program to attempt a connection, the modem doesn’t start dialing and nothing is written to the log.

Solution: Increase *pppd*’s debugging level to 7 (IP message summary). If it now shows the packets, but says ‘dial failed’, then PPP is waiting after a failed call attempt until the call retry delay timer expires. Use SIGINT to tell *pppd* to reset the timer, change the Systems file entry if it is set to use too long a delay, or wait until the timer expires.

Scenario: *Pppd* installs a route when it starts (as shown by *netstat -r*), but after a while it disappears.

Solution: Are you running *routed*? *Routed* is pretty stupid about point-to-point connections; its RIP protocol can’t adequately describe them to other *routed*’s, and it really gets confused about intermittent connections, over which it sometimes can’t hear from another *routed* to reassure itself that the link is still up, and therefore deserving of a route.

We suggest running *gated* instead, available via anonymous FTP from gated.cornell.edu or via anonymous PPP/FTP from Morning Star Technologies.

Scenario: PPP won't connect and the log file shows 'IPCP: Received bad Configure-Reject'

Solution: Restart *pppd* with the **novjcomp** option. If this fixes the problem, then the peer may be using a buggy IPCP option negotiation algorithm. Try running *pppd* with the **rfc1172-vj** or the **rfc1172-typo-vj** option. If they don't work, go back to **novjcomp** and complain to the vendor of the peer's implementation of PPP.

Scenario: PPP won't come up, and the peer complains that it is has received a request for IPCP configuration option 3.

Solution: Restart *pppd* with the **rfc1172-addresses** option, and report a bug to the vendor of the peer's implementation of PPP. If the peer doesn't recognize IPCP configuration option 3, it should respond with an IPCP Configure-Reject rather than exiting altogether.

Scenario: The PPP link comes up, but a minute later the log file shows **LQM: LQRs: 0/5**, then **LQM: Too many Link-Quality-Reports lost**, then *pppd* closes the connection.

Solution: Restart *pppd* with either the **echolqm** or the **nolqm** option, and report a bug to the vendor of the peer's implementation of PPP. If the peer doesn't recognize the new-style Link Quality Report, it should respond with a Protocol-Reject rather than not responding at all. When *pppd* sees the Protocol-Reject, it will fall back to using LCP Echo-Requests for its link status monitoring functions.

Scenario: When the modem abruptly gets hung up, *pppd* doesn't notice.

Solution: Make sure that the modem drops the Carrier Detect (CD, also called DCD, Data Carrier Detect) signal when carrier is lost. On a Telebit T1600, set '&C1'.

Make sure that the cable properly connects CD (RS-232 pin 8) from the modem through to the serial port, and that the *pppd* command line doesn't include the **ignore-cd** option.

Scenario: Shortly after the Systems chat script completes, the log file records a SIGHUP and the connection drops. But the remote modem didn't actually hang up the connection, and *tip* works correctly.

Solution: Make sure that the modem asserts Carrier Detect (CD, also called DCD, Data Carrier Detect) signal when it is actually talking to a

remote modem. On a Telebit T1600, set '&C1'.

Make sure that the cable properly connects CD (RS-232 pin 8) from the modem through to the serial port. *pppd* doesn't check the state of DCD until after the Systems chat script completes, about the time that PPP negotiations begin. If the modem isn't asserting DCD, or if the cable isn't delivering the signal to the serial port, *pppd* won't be able to tell that the modem is connected. To test this, or to work around a defective cable, start *pppd* with the **ignore-cd** option.

Scenario: *Pppd* won't hang up the phone, even though it was started with **idle 300**.

Solution: Increase *pppd*'s debugging level to 7 (IP frame summary) and watch for logged frames that *don't* say '!keepup'. If you don't want those messages to keep your line active, put them in the 'keepup' filter in the **Filter** file.

Scenario: *Pppd* dies with a 'Fatal error: ...' or 'Fatal system error: ...' message.

Solution: That's probably a bug. Send e-mail to support@MorningStar.Com containing relevant portions of pppd.log and your configuration files (edit out the passwords and phone number first) and we'll fix it.

Scenario: When you telnet to test if the answering PPP account is set up correctly, you will be greeted with what looks like line noise. This is the PPP Link Control Protocol Configure-Request packet, but it takes a long time for *pppd* to give up and go away.

Solution: The easy way to instruct the answering *pppd* to exit and close the connection is to type the four-character sequence of a tilde ('~', ASCII 0x7E) followed by three control-Cs (^C, ASCII 0x03).

Scenario: *Pppd* won't dial out; the log file says 'Device is locked'.

Solution: *Pppd* uses lock files compatible with other programs to avoid using a device already in use by another program, and to keep other programs from interfering with devices in use by *pppd*. If the pppd.log file says 'Device is locked', look for a lock file such as LCK..cua in /usr/spool/locks, /usr/spool/uucp, /usr/spool/uucp/LCK, /var/spool/locks, or wherever your system keeps tty lock files. Each lock file should contain the process id of the owner process. If the lock file is 4 bytes long, the process id is a 32-bit binary number. If the lock file is 11 bytes long, the process id is printed in ASCII. Look in the lock file for the owner pid (use *od -d* for binary files and *cat* for ASCII lock files), and then use *ps* to look for the owner.

The University of Toronto SLIP implementation for SunOS 4.1 uses lock files that are name-compatible with those used by HDB UUCP and Morning Star PPP. That is, they're in /usr/spool/locks named LCK..cua or LCK..ttyb or whatever is appropriate. Unfortunately, UTSLIP's lock files are of zero length, which makes it impossible to determine the pid of the owner process. Later instances of *uucico* or *pppd* will examine the contents of any lock files they find, to see whether the creating process is still running, or whether it perhaps died ungracefully and didn't clean up its lock files. If the latter, *uucico* and *pppd* know that they can usurp the claimed but expired lock on that interface.

When *pppd* finds a lock file of zero length, it has no way of knowing the pid of the process that created it, so to be safe, it gives up trying to open the associated tty.

Scenario: With debug set to 2 or more, I get messages in the log file like 'Bad FCS received', 'Bad protocol (even), flushing frame', 'Short frame received (3 bytes)', 'Frame too long, flushing frame', 'Missed ALL-STATIONS, flushing frame', or 'Missed UI, flushing frame'.

Solution: Some of the incoming messages are getting damaged in transit. If your throughput is erratic or lower than you expect, then refer to the section above starting with 'PPP connects and everything works, but it is slow and erratic'.

If the messages only happen rarely, then don't worry about them. The errors may be due to intermittent line noise in the phone lines, or your computer may occasionally lose incoming characters. Suspect this last possibility if the errors tend to occur when receiving large amounts of data.

Scenario: My NeXT system can't run Morning Star PPP, but it has always run Marble Teleconnect just fine. Whenever the MST PPP daemon starts, the it complains in the log file about problems using **du0**.

Solution: Remove Marble Teleconnect and reboot your NeXT system. Its use of the device name **du0** collides with MST PPP's use of the name, and there's nothing Marble can do that MST PPP can't do.

Scenario: You need to use XON/XOFF flow control (perhaps you're running on a DECstation under Ultrix 4.2) but the peer (perhaps a Telebit Net-Blazer running Fred 1.5) asks for an Asynchronous-Control-Character-Map of 0x0, rather than 0xA0000. Performance is poor.

SOlution: Use the arguments 'escape 0x11 escape 0x13' to cause *pppd* to ignore the peer's requested asyncmap and escape those would-be flow control characters anyway, rather than sending them (when they appear in user data) in a way that would cause the peer to think they're intended as actual in-band flow control.

Glossary

Active-Open — An event internal to the PPP configuration finite state machine that causes PPP to start sending Configure-Request messages to the peer.

bps — Bits per second.

CCITT — The French acronym of the International Telegraph and Telephone Consultative Committee, an organization that publishes international data communications standards.

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*.

DNS — See *Domain Name System*.

Domain — 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.

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 — 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 — The NCP for IP, the Internet Protocol.

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.

LQM — See *Link Quality Monitoring*.

LQR — See *Link-Quality-Report*.

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.

MNP — The Microcom Network Protocols include several error correction and data compression schemes for use in modems.

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.

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.

PAP — See *Password Authentication Protocol*.

Passive-Open — An event internal to the PPP configuration finite state machine that causes PPP to move from the Closed state to the Listen state, where it awaits Configure-Request messages from the peer.

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 — The PPP 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.

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*.

Request For Comments — The Internet name for a standard.

RFC — See *Request For Comments*.

RPC — The Sun Remote Procedure Call protocol, described in RFC 1050, is used to distribute applications over multiple networked computers.

SCSI — The Small Computer System Interface is a standard for connecting high-speed intelligent peripherals, such as disk or tape drives, to computers.

Serial Line Internet Protocol — A simple protocol for carrying IP datagrams over serial links. No error detection or configuration negotiation is included.

SLIP — See *Serial Line Internet Protocol*.

SMTP — The Simple Mail Transfer Protocol, described in RFC 821, is used to deliver electronic mail messages across internet networks.

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.

UUCP — The Unix-to-Unix Copy Program is used to transfer data files and electronic mail, usually across dial-up modem connections, between Unix systems.

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.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.

Release Notes

This section describes the changes that have occurred in each successive release of Morning Star PPP.

Version 1.3 - June 5, 1992

Features Added

- Support for IBM RS6000 under AIX, Silicon Graphics Indigo and Personal Iris, and BSDI
- Tunnel driver support for multiprocessor Suns under SunOS 4.1.2
- Speedier daemon startup
- More useful logging messages
- Improved performance and responsiveness on Suns
- Allow interoperation with old CHAP implementations
- Allow better interoperation with old (but buggy) IPCP implementations
- LQM updated to February 1992 draft
- CHAP updated to April MD5 draft
- Finite state machine updated to December 1991 draft
- Address discovery in SLIP dialups to cisco terminal servers (e.g. PSI Host-DCS)
- Use **PPPHOME** environment variable to specify config file directory
- Synchronous lines pay attention to Carrier Detect signal
- Support for external dialer programs
- *Exec* script receives peer's IP address as second argument
- Increase max MTU size from 1500 to 2048
- Better interoperation with buggy peers

Bugs Fixed

- Fixed netmask handling at shutdown time
- Fixed IPCP address negotiation bug
- Fixed SIGSEGV when daemon is started with no environment
- Handle Protocol-Rejects properly
- Fix SLIP 'dial failed' problem

- Fix SLIP VJ handling when we receive a compressed frame
- Improved SnapLink buffer flow control
- Fix handling of Configure-NAKs with large MRUs
- Fix SunOS panic problem when we receive 'ping -lR'
- Fix transmitted frame when A/C compression is off but protocol compression is on
- Fix handling of IPCP Configure-Rejects that don't reject the address
- CHAP left Auth file open
- Fix SCO dialer exit status problem added with 'exec' option
- Fix some panic-inducing race conditions
- Fix PAP negotiation bug when requireauth and nochap are both specified
- Document some previously-undocumented SnapLink-specific Devices options
- Fix frame fragmenting problem when pppd generates its own frame (e.g. LQR) in the midst of sending user data frame.
- Fix nooptions overzealousness
- Fix NeXT SnapLink crashing problem
- Fix EINTR handling in chat script processing
- Fix IPCP Compression-Protocol negotiation bug; it was parsing the vjslots and vjcid fields incorrectly in received frames
- More SCO modem and flow control handling improvements
- Fix SCSI read and write block sizes on SCO and NeXT
- Make SunOS 4.0.3 a separate distribution (for both Sun-3 and SPARC) because it lacks tcdrain().

Version 1.2.1 - February 5, 1992

Features Added

- Dynamically loadable tunnel driver on Sun-3
- Hardware flow control support on SCO UNIX
- Ability to run an arbitrary program or shell script when the connection comes up or goes down
- CHAP upgraded to December 1991 draft
- IPCP address negotiation upgraded to December 1991 draft

- Support for yet more dialer types
- More useful logging messages

Bugs Fixed

- Avoid dividing by zero in LQM throughput calculations
- Handle CRC errors correctly when using VJ TCP header compression
- Slight performance improvements on Suns
- Don't continuously send LCP Echo-Requests during or after link shutdown
- More reliable chat script processing

Version 1.2 - December 27, 1991

Features Added

- Support for SCO 386 UNIX (V.3.2), Sun-3/SunOS4.1, and DECstation/Ultrix 4.2
- Dynamically loadable tunnel driver on SPARC
- Failover between multiple connection methods
- Added ability to 'escape' arbitrary character values
- Improve ability to work over dedicated lines
- Authentication now supports address restrictions
- Detect line failures with Link Quality Monitoring, with LCP Echo-Requests
- CHAP code conforms to October 1991 draft
- Configuration files contained in their own directory
- Documentation improved with table of contents, index
- Chat scripts can be continued over multiple lines
- Can work on serial interfaces that don't provide Carrier Detect signal
- Better information in log file during connection establishment

Bugs Fixed

- Fixed SLIP SIGHUP "Bad file descriptor" bug
- Fixed filter file "log tcp/syn" bug
- Fixed NeXT hangup bug
- Same daemon supports SPARC from SunOS 4.0.3 forward

- Finite state machine begins with Active-Open in all cases
- Fix SLIP VJ-uncompression bug that excites the 'panic: mclput' bug in SunOS 4.1 and 4.1.1
- Fix route deletion when using dialins but not dialouts
- Negotiated behavior is now asymmetrical
- Tunnel driver now supports more than ten peers
- Better Ultrix serial port support
- Close stdin, stdout, stderr after forking for autodial

Version 1.1 - October 8, 1991

Features Added

- Add pause and break to chat scripts
- Work with more UUCP port-locking mechanisms
- Allow peer to assign an IP address
- Synchronous support on SnapLink up to T1
- Synchronous support on Sun-4c and NeXT serial ports
- Allow ranges of ports in filter specs
- Support for draft (6-byte) VJ negotiation
- Add SLIP support
- Upgrade finite state machine to new draft specs
- Authentication with CHAP

Bugs Fixed

- Lock file is removed if error occurs partway through opening a device
- Packets transmitted with correct PPP Address/Control and Protocol field compression
- More useful messages about various failures

Version 1.0 - August 22, 1991

Original release on Sun-4 and Solbourne

Index