Leased line Mini HOWTO

Rob van der Putten <rob%40sput%2Enl>

2005-09-05

Revision History

Revision 2.3b7	2005-09-05	RvdP
	Additional PPPD options and routing	
Revision 2.3b6	2005-01-19	RvdP
	New net howto link	
Revision 2.3b5	2004-12-31	RvdP
	1st XML version	
Revision 2.3b4	2003-10-01	RvdP
	Escaped email address	
Revision 2.3b3	2002-09-19	RvdP
	1st experimental DocBook version	
Revision 2.2	2001-12-05	RvdP
	FDL copyright	
Revision 2.1	2000-08-03	RvdP
	New author email address	
Revision 2.0	2000-04-20	RvdP
	1st LinuxDoc SGML version	

Abstract

Configuring your modem and pppd to use a 2 wire twisted pair leased line.

Table of Contents

The most recent (beta) version of this HOWTO can be found at: http://www.sput.nl/software/leased-line/

Introduction

Copyright and License

What is a leased line

Any fixed, that is permanent, point to point data communications link, which is leased from a telco or similar organisation. The leased line involves cables, such as twisted pair, coax or fiber optic, and may involve all sorts of other hardware such as (pupin) coils, transformers, amplifiers and regenerators.

This document deals with:	Configuring your modem and pppd to use a 2 wire twisted pair leased line.
This document does *NOT* deal with:	SLIP, getting or installing pppd, synchronous data communication, baseband modems, xDSL.

Assumptions

You should already have a working pppd on your system. You also need Minicom or a similar program to configure your modems.

Modem

A leased line is not connected to a telephone exchange and does not provide DC power, dial tone, busy tone or ring signal. This means that your modems are on their own and have to be able to deal with this situation.

You should have 2 identical (including firmware version) *external* modems supporting both leased line and dumb mode. Make sure your modems can actually do this! Also make sure your modem is properly documented. You also need:

- 2 fully wired shielded RS232 cables. The shield should be connected to the connector shell (not pin 1) at both ends (not at one end).
- A RS232 test plug may be handy for test purposes.
- 2 RJ11 cords, one for each end of the leased line.
- A basic understanding of `AT' commands.

Modem Configuration

A note on modem configuration and init strings in general: Configure your modem software such as minicom or (m)getty to use the highest possible speed; 57600 bps for 14k4 and 115200 bps for 28k8 or faster modems. Lots of people use very long and complicated init strings, often starting with AT&F and containing lots of modem brand and -type specific commands. This however is needlessly complicated. Most programs feel happy with the same modem settings, so why not write these settings in the non volatile memory of all your modems, and only use `ATZ' as an init string in all your programs. This way you can swap or upgrade your modems without ever having to reconfigure any of your software.

Most programs require you to use the following settings;

- Fixed baud rate (no auto baud)
- Hardware bidirectional RTS-CTS flow control (no x-on/x-off)
- 8 Bits, no parity, 1 stopbit
- The modem should produce the *TRUE* DCD status (&C1)

- The modem should *NOT* ignore the DTR status (&D2 or &D3)

Check this with AT&V or AT&Ix (consult your modem documentation)

These settings are not necessarily the same as the default factory profile (&F), so starting an init string with AT&F is probably not a good idea in the first place. The smart thing to do is probably to use AT&F only when you have reason to believe that the modem setup stored in the non volatile memory is really screwed up. If you think you have found the right setup for your modems, write it to non volatile memory with AT&W and test it thoroughly with Z-modem file transfers of both ASCII text and binary files. Only if all of this works perfectly should you configure your modems for leased line.

Find out how to put your modem into dumb mode and, more importantly, how to get it out of dumb mode; The modem can only be reconfigured when it is not in dumb mode. Make sure you actually configure your modems at the highest possible speed. Once in dumb mode it will ignore all `AT' commands and consequently will not adjust its speed to that of the COM port, but will use the speed at which it was configured instead (this speed is stored in a S-register by the AT&W command).

Now configure your modem as follows;

- Reset on DTR toggle (&D3, this is sometimes a S register). This setting is required by some ISP's!
- Leased line mode (&L1 or &L2, consult your modem documentation)
- The remote modem auto answer (S0=1), the local originate (S0=0)
- Disable result codes (Q1, sometimes the dumb mode does this for you)
- Dumb mode (\D1 or %D1, this is sometimes a jumper) In dumb mode the modem will ignore all AT commands (sometimes you need to disable the ESC char as well).

Write the configuration to non-volatile memory (&W).

Test

Now connect the modems to 2 computers using the RS232 cables and connect the modems to each other using a RJ11 lead. Use a modem program such as Minicom (Linux), procom or telix (DOS) on both computers to test the modems. You should be able to type text from one computer to the other and vice versa. If the screen produces garbage check your COM port speed and other settings. Now disconnect and reconnect the RJ11 cord. Wait for the connection to reestablish itself. Disconnect and reconnect the RS232 cables, switch the modems on and off, stop and restart Minicom. The modems should always reconnect at the highest possible speed (some modems have speed indicator leds). Check whether the modems actually ignores the ESC (+++) character. If necessary disable the ESC character.

If all of this works you may want to reconfigure your modems; Switch off the sound at the remote modem (M0) and put the local modem at low volume (L1).

Examples

Hi-Tech

This is a rather vague `no name clone modem'. Its config string is however typical and should work on most modems.

Originate (local):	ATL1&C1&D3&L2%D1&W&W1

Answer (remote):	ATM0L1&C1&D3&L2%D1S0=1&W&W1

Tornado FM 228 E

This is what should work;

Originate (local):	ATB15L1Q1&C1&D3&L2&W&W1
Answer (remote):	ATM0B15M0Q1&C1&D3&L2S0=1&W&W1

Move the dumb jumper from position 2-3 to 1-2.

Due to a firmware bug, the modems will only connect after being hard reset (power off and on) while DTR is high. I designed a circuit [http://www.sput.nl/hardware/modem-reset.html#l2h] which hard resets the modem on the low to high transition of DTR. The FreeBSD pppd however, isn't very happy about this. By combining the setting &D0 with a circuit [http://www.sput.nl/hardware/modem-reset.html#h2l] which resets on the high to low transition instead, this problem can be avoided.

Tron DF

The ESC char should be disabled by setting S2 > 127;

Originate:	ATL1&L1Q1&C1&D3S2=171\D1&W
Answer:	ATM0&L2Q1&C1&D3S0=1S2=171\D1&W

US Robotics Courier V-Everything

The USR Sportster and USR Courier-I do not support leased line. You need the Courier V-everything version for this job. There is a webpage on the USR site `explaining' how to set-up your Courier for leased line. However, if you follow these instructions you will end up with a completely brain dead modem, which can not be controlled or monitored by your pppd.

The USR Courier can be configured with dip switches, however you need to feed it the config string first. First make sure it uses the right factory profile. Unlike most other modems it has three; &F0, &F1 and &F2. The default, which is also the one you should use, is &F1. If you send it an AT&F, however it will load the factory profile &F0! For the reset on DTR toggle you set bit 0 of S register 13. This means you have to set S13 to 1. Furthermore you need set it to leased line mode with &L1; ATS13=1&L1&W The dip switches are all default except for the following:

3	OFF Disable result codes
4	ON Disable offline commands
5	ON For originate, OFF For answer
8	OFF Dumb mode

PPPD

You need a pppd (Point to Point Protocol Daemon) and a reasonable knowledge of how it works. Consult the relevant RFC's or the Linux PPP HOWTO [http://www.tldp.org/HOWTO/PPP-HOWTO/index.html] if necessary. Since you are not going to use a login procedure, you don't use (m)getty and you do not need a (fake) user associated with the pppd controlling your link. You are not going to dial so you don't need any chat scripts either. In fact, the modem circuit and configuration you have just build, are rather like a fully wired null modem cable. This means you have to configure your pppd the same way as you would with a null modem cable.

For a reliable link, your setup should meet the following criteria;

- Shortly after booting your system, pppd should raise the DTR signal in your RS232 port, wait for DCD to go up, and negotiate the link.
- If the remote system is down, pppd should wait until it is up again.
- If the link is up and then goes down, pppd should reset the modem (it does this by dropping and then raising DTR), and then try to reconnect.
- If the quality of the link deteriorates too much, pppd should reset the modem and then reestablish the link.
- If the process controlling the link, that is the pppd, dies, a watchdog should restart the pppd.

Configuration

Suppose the modem is connected to COM2, the local IP address is `Loc_Ip' and the remote IP address is `Rem_Ip'. We want to use 576 as our MTU. The `/etc/ppp/options.ttyS1` would now be:

```
crtscts
mru 576
mtu 576
passive
Loc_Ip:Rem_Ip
-chap
modem
#noauth
-pap
persist
#maxfail 0
#holdoff 10
```

Stuff like `asyncmap 0', `lock', `modem' and `-detach' are probably already in `/etc/ppp/options`. If not, add them to your `/etc/ppp/options.ttyS1`. So, if the local system is 192.168.1.1 and the remote system is 10.1.1.1, then `/etc/ppp/options.ttyS1` on the local system would be:

```
crtscts
mru 576
mtu 576
passive
192.168.1.1:10.1.1.1
-chap
modem
#noauth
-pap
persist
#maxfail 0
#holdoff 10
```

The options.ttyS1 on the remote system would be:

```
crtscts
mru 576
mtu 576
passive
10.1.1.1:192.168.1.1
-chap
modem
#noauth
-pap
persist
#maxfail 0
#holdoff 10
```

The passive option limits the number of (re)connection attempts. The persist option will keep pppd alive in case of a disconnect or when it can't connect in the first place. If you telnet a lot while doing filetransfers (FTP or webbrowsing) at the same time, you might want to use a smaller MTU and MRU such as 296. This will make the remote system more responsive. If you don't care much about telnetting during FTP, you could set the MTU and MRU to 1500. Keep in mind though, that UDP cannot be fragmented. Speakfreely [http://www.fourmilab.ch/netfone/] for instance uses 512 byte UDP packets. So the minimum MTU for speakfreely is 552 bytes. The noauth option may be necessary with some newer distributions. `maxfail 0' may be necessary with newer PPPDs. After the connection is lost, PPPD will wait for a while before reconnecting. This time can be set with the holdoff option. The default holdoff used to be 30 seconds, but is now zero. A holdoff of 10 is often recommended.

Scripts

Starting the pppd and keeping it alive

You could start the pppd form a boot (rc) script. However, if you do this, and the pppd dies, you are without a link. A more stable solution, is to start the pppd from `/etc/inittab`;

```
s1:23:respawn:/usr/sbin/pppd /dev/ttyS1 115200
```

This way, the pppd will be restarted if it dies. Make sure you have a `-detach' option (nodetach on newer systems) though, otherwise inittab will start numerous instances of pppd, while complaining about `respawning too fast'.

Note: Some older systems will not accept the speed `115200'. In this case you will have to set the speed to 38400 and set the `spd_vhi' flag with setserial. Some systems expect you to use a `cua' instead of `ttyS' device.

Setting the routes

The default route can be set with the defaultroute option or with the `/etc/ppp/ip-up` script;

```
#!/bin/bash
case $2 in
     /dev/ttyS1)
          /sbin/route add -net 0.0.0.0 gw Rem_Ip netmask 0.0.0.0
          ;;
esac
```

Ip-up can also be used to sync your clock using netdate.

Of course the route set in ip-up is not necessarily the default route. Your ip-up sets the route to the remote network while the ip-up script on the remote system sets the route to your network. If your network is 192.168.1.0 and your ppp interface 192.168.1.1, the ip-up script on the remote machine looks like this;

```
#!/bin/bash
case $2 in
   /dev/ttyS1)
      /sbin/route add -net 192.168.1.0 gw 192.168.1.1 netmask 255.255.255.0
      ;;
esac
```

The `case $2' and `/dev/ttyS1)' bits are there in case you use more than one ppp link. Ip-up will run each time a link comes up, but only the part between `/dev/ttySx)' and `;;' will be executed, setting the right route for the right ttyS. You can find more about routing in the Linux Networking HOWTOs [http://www.tldp.org/HOWTO/HOWTO-INDEX/networking.html] section on routing.

Some systems use dynamic ttys, in which case you can't route on a tty basis. In this case it might be handy to translate the ip address to a ppp interface and then do the routing (and firewalling) on a ppp interface basis. For this purpose I edited `/etc/ppp/ip-up`;

```
# These variables are for the use of the scripts run by run-parts
PPP_IFACE="$1"
PPP_TTY="$2"
PPP_SPEED="$3"
PPP_LOCAL="$4"
PPP_REMOTE="$5"
PPP_IPPARAM="$6"
export PPP_IFACE PPP_TTY PPP_SPEED PPP_LOCAL PPP_REMOTE PPP_IPPARAM

# translate ip to ppp
echo $PPP_IFACE > "/var/run/ppp/if-$PPP_LOCAL"
sleep 1
# Rerun firewall.
/usr/local/sbin/rc.block

# Take care of the (default) route(s)
case $PPP_LOCAL in
 "My_Ip_Address")
  /sbin/route add -net 0.0.0.0 gw $PPP_REMOTE netmask 0.0.0.0
  ;;

esac

# Fix things missed at boot
if ! ( netstat -an | grep 'My_Ip_Address:53' > /dev/null 2>&1 )
then
 # Just booted
 # Sync clock
 /usr/local/sbin/ntpdate.sh &
 # Set the null routes
 /usr/local/sbin/null-route.sh &
 # Bind 9 needs this;
 sleep 1
 /etc/init.d/bind9 restart
fi

# An audiable notification
/bin/echo -ne "\007" >> /dev/tty1
```

Replace 'My_Ip_Address' with your Ip address. `/usr/local/sbin/ntpdate.sh` synchronises the clock. It stops the NTPD, syncs using ntpdate and then starts the NTPD again. `/usr/local/sbin/null-route.sh` is a script which sets null routes;

```
#!/bin/bash
route add -net 10.0.0.0    netmask 255.0.0.0   reject
route add -net 172.16.0.0  netmask 255.240.0.0 reject
route add -net 192.168.0.0 netmask 255.255.0.0 reject
```

If you have RFC 1918 addresses in use, the above null routes won't interfere provided you use a smaller netmask. A network 192.168.1.0/24 won't be bothered by the null route 192.168.0.0/16;

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
255.255.255.255 0.0.0.0         255.255.255.255 UH    0      0        0 eth1
195.190.249.4   0.0.0.0         255.255.255.255 UH    0      0        0 ppp0
10.0.0.0        0.0.0.0         255.255.255.0   U     0      0        0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     0      0        0 eth1
192.168.0.0     -               255.255.0.0     !     0      -        0 -
172.16.0.0      -               255.240.0.0     !     0      -        0 -
10.0.0.0        -               255.0.0.0       !     0      -        0 -
0.0.0.0         195.190.249.4   0.0.0.0         UG    0      0        0 ppp0
```

Test

Test the whole thing just like the modem test. If it works, get on your bike and bring the remote modem to the remote side of your link. If it doesn't work, one of the things you should check is the COM port speed; Apparently, a common mistake is to configure the modems with Minicom using one speed and then configure the pppd to use an other. This will *NOT* work! You have to use the same speed all of the time!