diald
DIAL 'D' FOR DAEMON

The original aim of diald was to bring up and take down a dialup link to the Internet on demand. As Robert Morris explains this is just the start of what diald has to offer

A dial on demand Internet link works very well over ISDN (although it does work on a traditional modem link too), with the advent of unmetered Internet accounts. It is ideally suited to home network or small office applications and *diald* provides an Open Source alternative to the more expensive dedicated ISDN routers.

Functionality

diald's functionality can be summed up as a daemon that controls and monitors a non-full-time IP connection. This functionality consists of three elements – connecting and disconnecting on demand – allowing manual control (both locally and remotely), and providing monitoring of the status of the connection.

The on demand functionality is implemented with a proxy interface, using the Linux ethertap device. This is a virtual network interface, which gives any packets routed to it to a userland process – *diald* in this case. This device is set up just like any other network interface – and in the normal configuration of using *diald* to manage a link to the Internet, the machine's default route would point to the proxy interface. Thus any traffic destined for the "outside world" is handed over to *diald*. When *diald* receives a packet and "triggers" (i.e. decides to bring the link up), it removes the proxy interface and then runs *pppd* to bring up the real interface. The routing is adjusted automatically, and the trigger packet fed back to the kernel, which then routes it in the normal way. *diald* then monitors the ppp device and, when it sees it is idle, kills *pppd* and reinstates the proxy device ready for the next trigger.

/etc/diald.conf

```
mode ppp
device /dev/ttyS0
speed 57600
modem
lock
crtscts
authsimple /etc/diald.auth
tcpport 1020
connect "chat '' ATZ OK
ATDT08001234567 CONNECT"
defaultroute
dynamic
local 192.168.0.1
remote 192.168.0.2
include
/usr/lib/diald/standard.filter
```

Simple local control of *diald* can be achieved using signals. The two most useful are SIGUSR1, which tells *diald* to immediately bring the link up, and SIGINT, which immediately takes the link down (although leaving *diald* running). These can be useful for implementing regular timed connections (such as for mail polls) from *cron*, for example.

diald also has a more complex command interface, which is available locally through a named pipe, and remotely using a TCP port (or of course locally by connecting to localhost). Authentication must be performed before any control commands are permitted. The named pipe and TCP port interfaces are not enabled unless the applicable commands are specified in your *diald* configuration file. Once authenticated, a number of commands can be issued to control the link, for example: *up* to bring the link up; *down* to take the link down; *force* to bring the link up and force it to remain up until *unforce* is specified; and *block* to bring the link down and block connection attempts until *unblock* is specified. The command set is documented in the *diald-control* manpage.

The TCP port also provides monitoring of the link status, including whether demand mode is enabled or disabled, or the connection is forced or blocked. Full details of this are in the *diald-monitor* manpage.

Configuration

diald is available in both traditional .tar.gz and rpm archives from *http://diald.sourceforge.net*. At the time of going to press, the latest version is 1.0.

In most installations you'll want to ensure that *diald* is started at boot up. If you've installed from source then you may have to manually add a line to a startup file such as */etc/rc.local*

You should recognise some of these commands from the *pppd* options file – in fact the commands relating to the modem (*device*, *speed*, *modem*, *lock*, *crtscts*) and chat script (*connect*) behave in exactly the same way. The reason they're specified here is that *diald* speaks to the modem itself, and runs the chat script (as specified by *connect*) prior to handing control over to *pppd*. Therefore you should not specify the modem and chat script commands in

/etc/ppp/options when *pppd* is being called by *diald*.

The remaining commands configure the TCP port and authentication for controlling *diald*, set the default route to point to *diald*, and use dynamic addressing. *diald* needs the *local* and *remote* commands when dynamic addressing is used – these specify temporary IP addresses for *diald* to use for its proxy interface, since the real addresses are only established after the link has been brought up.

Finally, *standard.filter* is included – this line is essential, because *standard.filter* contains all the rules specifying what types of packets *diald* will trigger on or ignore, and how long the link will be initially brought up for, etc.

This is a minimal ppp config, since many of the usual *pppd* commands are dealt with by *diald* instead. Obviously you would need an entry in pap-secrets or chap-secrets to specify the secret for the *user* and *remotename* combination you've specified here.

To use the TCP port for anything other than monitoring, you need to authenticate to *diald*. Two authentication schemes are supported – "simple", and PAM. The simple scheme is meant for applications where all the clients are trusted. If you use it, you should make very sure that your TCP port is firewalled from the outside world and only open to hosts on your local network. You specify an auth file with the *authsimple* command, and this file should contain one or more lines in the following format (my */etc/diald.auth* given as an example):

```
rob      up,down,force,block
```

Here, the user "rob" is authorised to issue the commands shown. Once connected to the TCP port, you need only send *auth simple rob* and then you may proceed to send other commands. No password etc, is required (use PAM authentication for this).

If you're using the rpm, create a file in */etc/sysconfig/network-scripts* for each copy of *diald* you want to start, with the prefix *dialdcfg-*. If you only want a single instance of *diald*, you can simply do a *touch /etc/sysconfig/network-scripts/dialdcfg-internet*, and place all your configuration in diald.conf.

You can set up multiple instances of *diald*, for example I have one instance which connects to the Internet and another to connect to the office dial-in service. To run multiple instances, you create one *dialdcfg* file for each, and put a "DIALDOPTIONS=" line in each. I like to put connection-specific configuration in config files under */etc/diald*, and then simply put an "include" command in the DIALDOPTIONS line, to keep the configuration easy to read. Obivously, make sure only one instance has the *defaultroute* command. You can use *addroute* to specify a script to do your own routing.

/etc/ppp/options

```
lock
user rob
remotename internet
noauth
```

Pitfalls

Using *diald* has its downsides. These relate to dial on demand solutions in general. If you're using an ordinary dialup account with dynamic address allocation, it can be annoying if *diald* takes the link down on you, and brings it back up, causing your address to change – this breaks any open connections in *ssh*, FTP and so on. HTTP and POP however (which is what most desktop users will be using) don't keep TCP connections open once the data is transferred, so they work just fine.

Secondly, if you're using a modem with *diald* to provide on demand Internet access to Windows clients (a common arrangement in a small business installation), you may find that, because the clients are not aware that the connection is dialup, they time out whilst waiting for the modem to negotiate. This can be frustrating for users. With ISDN, where the connection time is only a second or so, *diald* works quite nicely.

Finally, be careful if you're using an ordinary 0845 account – *diald* may trigger when you don't want it to. Whether it be a mis-configured daemon that tries to connect to an external IP address in the middle of the night, or an anti-virus utility on a user's desktop machine that tries to download an update from its Web site every time it is started up.

Other applications

diald is useful in other applications too. When *mode dev* is specified in the configuration, *diald* effectively hands control of bringing the connection up and down to scripts that you specify. In this way *diald* can be used to monitor and control any type of link whatsoever. For example, it could be used as a front-end to a VPN tunnel – creating the tunnel only as traffic arrives and destroying it afterwards.

Another alternative application is using *diald* to manage a backup Internet connection, to be activated on failure of the primary link (ADSL in this case, or leased line etc). *diald* was configured to connect to a normal Internet dialup account, but with demand mode disabled and no default route (since the default route is the ADSL line). A small Windows applet was provided that sits in the bottom right-hand corner of the users' desktops, which they could use to activate the dialup connection in the event that their ADSL line stopped working.

If you've got any type of link that you want to either bring up and take down transparently, or let users control from their desktops, then take a look at *diald*.

The author

Robert Morris is a freelance Linux professional, and a contributor to the *diald* project. He can be contacted at: *rob@r-morris.co.uk*