Diald Howto

Table of Contents

Diald Howto....................1
Andrés Seco AndresSH@ctv.es....................1
1.Introduction....................1
2.Copyright and discharge of responsibility....................1
3.Quick Diald operation description....................1
4.Note about authentication....................1
5.Notes about DNS name resolution....................1
6.Connecting a standalone computer to an ISP using a modem and PPP....................2
7.Conecting a computer to a group of different ISPs with a modem and PPP....................2
8.Connecting a proxy/firewall to an ISP using a modem and PPP....................2
9.Programs and versions used....................2
10.More information....................2
1.Introduction....................2
1.1 Objectives....................2
1.2 New versions....................3
1.3 Thanks....................3
2.Copyright and discharge of responsibility....................4
3.Quick Diald operation description....................4
4.Note about authentication....................5
4.1 Username and password – Login and password prompts....................5
4.2 PAP – Password Authentication Protocol....................6
4.3 CHAP – Challenge Authentication Protocol....................6
5.Notes about DNS name resolution....................7
6.Connecting a standalone computer to an ISP using a modem and PPP....................7
6.1 /etc/diald/diald.options or diald.conf file....................8
6.2 Personal filters file....................10
6.3 Making the call....................13
6.4 Connection start script....................14
7.Conecting a computer to a group of different ISPs with a modem and PPP....................14
7.1 Note about sending mail using a relay host....................15
7.2 Scripts to automate multiple connections and changing from one to another....................15
Starting up....................15
New provider....................16
Changing from one to another....................16
8.Connecting a proxy/firewall to an ISP using a modem and PPP....................17
8.1 Example for Debian 2.1....................17
8.2 Example for Suse 6.1....................17
8.3 Example for Slackware 3.6....................19
9.Programs and versions used....................19
10.More information....................20

Diald Howto

Andrés Seco AndresSH@ctv.es

v1.03, April 17, 2000

This document shows some typical scenarios for easy start using Diald. These scenarios include a connection from a standalone computer to an ISP using PPP over a modem without using `pon/poff` *or* `ppp-on/ppp-off` *to a proxy/firewall server with different Internet connections through various ISPs.*

1.Introduction

- 1.1 Objectives
- 1.2 New versions
- 1.3 Thanks

2.Copyright and discharge of responsibility

3.Quick Diald operation description

4.Note about authentication

- 4.1 Username and password – Login and password prompts.
- 4.2 PAP – Password Authentication Protocol
- 4.3 CHAP – Challenge Authentication Protocol

5.Notes about DNS name resolution

6.Connecting a standalone computer to an ISP using a modem and PPP

- 6.1 /etc/diald/diald.options or diald.conf file
- 6.2 Personal filters file
- 6.3 Making the call
- 6.4 Connection start script

7.Conecting a computer to a group of different ISPs with a modem and PPP

- 7.1 Note about sending mail using a relay host
- 7.2 Scripts to automate multiple connections and changing from one to another

8.Connecting a proxy/firewall to an ISP using a modem and PPP

- 8.1 Example for Debian 2.1
- 8.2 Example for Suse 6.1
- 8.3 Example for Slackware 3.6

9.Programs and versions used

10.More information

1.Introduction

1.1 Objectives

This document shows some typical scenarios for easy start using *Diald*. These scenarios include a connection from a standalone computer to an ISP using PPP over a modem without using `pon/poff` or `ppp-on/ppp-off` to a proxy/firewall server with different Internet connections through various ISPs.

In the present document, the following scenarios will be treated:

- Connecting a standalone computer to an ISP using a modem and PPP
- Conecting a computer to a group of different ISPs with a modem and PPP
- Connecting a proxy/firewall to an ISP using a modem and PPP

In following versions of this document, other scenarios will be added, as multiple instances of *Diald*, ISDN lines and lines used to call and receive calls.

Before this document, a Diald−mini−Howto exist, wrote by Harish Pillay `h.pillay@ieee.org`, that presented and example of connection to an ISP using a chat based authentication scheme (login and password previous to the pppd start, with no use of PAP or CHAP).

Example configuration files will be included in this document to serve as starting point to get *Diald* up. To obtain maximum performance and all programs attributes, it is necesary you read all documentation from the programs and reconfigure the example configuration files included here.

Finally, configuration files can be in different directories depending on what GNU/Linux distribution you are using. If you find a file commented here in other directory, please, write me.

1.2 New versions

Latest version of this document can be found in my web page `http://www.ctv.es/USERS/andressh/linux`, in SMGL and HTML formats. Other versions and formats can be found in Spanish in the Insflug web site, `http://www.insflug.org/documentos/Diald−Como/`, and in other languages in the LDP − Linux Documentation Project, `http://www.linuxdoc.org`.

1.3 Thanks

I want to be grateful to the people that help me to get my first *Diald* up and running with their example files (somebody who's name i forgot, Mr Cornish Rex, Hoo Kok Mun and John Dalbec), to the people that have wrote me to send corrections and suggestions for this document (Tim Coleman, Jacob Joseph, Paul Schmidt and Jordi Mallach), to the future translators of this document to other languages, and, of course, to all the people that have developed and develops *Diald* for us.

This document was originally wrote in Spanish. The own author translated it, and some people made corrections.

2.Copyright and discharge of responsibility

Information and other contents in this document are the best of our knowledge. However, we may have made errors. So you should determine if you want to follow the instructions given in this document.

Nobody is responsible for any damage to your computer and any other loss derived from the use of the information contained herein.

THE AUTHOR AND MAINTAINERS ARE NOT RESPONSIBLE FOR ANY DAMAGE INCURRED DUE TO ACTIONS TAKEN BASED ON INFORMATION CONTAINED IN THIS DOCUMENT.

Of course, i am open to all type of suggestions and corrections on the content of this document.

3.Quick Diald operation description

In a few words, *Diald* creates a new network interface and sets it as the default gateway. This interface is not real (in the original documentation it is called `proxy interface`). *Diald* monitors this interface, and, when packets arrive, makes a `ppp` connection, waits for it to be stablished and changes the default gateway to this new `ppp` interface (usually `ppp0`).

Diald monitors the interface to determine which packets have been received the interface and their types to decide if they are going to be considered to set the `ppp` connection up, maintain the link, drop it or do nothing, and how long the link should be help up after the packet is transmitted.

Finally, if there is no more traffic and the last packet up time is over, *Diald* will close the link.

You can control days and hours when the link can go up and when it cannot, so you can use the low cost hours/days or low trafic times.

This previous description is valid for *Diald* versions from 0.16.5 to latest (0.99.3 when this document was finished), but latest versions also include aditional attributes such as user enabled list, advanced accounting, better support for ISDN lines, better performance using an `ethertap` device as proxy (this is like a network interface that read/writes over a socket instead of a real network adapter) in place of `slip`, backup connections and other functions.

4.Note about authentication

When you connect to an Internet Services Provider, it is usually necesary that you send an username and a password. This can be accomplished using several methods; the exact method that you use is determined by your provider.

Added to the three shown options, you can use a link without authentication, (generally when the remote end is also yours).

4.1 Username and password − Login and password prompts.

Actually, this is not an usual authentication method to access the Internet through an ISP.

Identification is made before `pppd` is started, and it is the dialer, usually `chat`, who sends the login name and the password. This data is sent in plaintext, so this method should not be considered secure.

An example script for `chat` where you can see how to specify username and password to be sent before running `pppd` would look something like this:

```
ABORT BUSY
ABORT "NO CARRIER"
ABORT VOICE
ABORT "NO DIALTONE"
ABORT "NO ANSWER"
"" ATZ
OK ATDT_TelephoneNumber_
CONNECT \d\c
ogin _Username_
assword _Password_
```

The last 2 lines define username and password, and when to send it (after receiving «ogin» and «assword» respectively. The chat script only needs to see parts of the words «login» and «password» and so we don't check the first letter of each. This is so that we don't need to worry about uppercase/lowercase characters.

Suppose that this script is called `provider`, and it is saved into the `/etc/chatscripts` directory. Then, you can run it with:

```
/usr/sbin/chat −v −f /etc/chatscripts/provider
```

4.2 PAP − Password Authentication Protocol

If the provider you are using requires PAP as the authentication protocol, during the LCP negotiation in PPP this protocol will be asked to use this protocol. When the phone call is connected after using `chat`, `pppd` is started. In this scenario, pppd will send the username and the password, which it will look for in the `/etc/ppp/pap-secrets` file. This file must have read and write permissions only for `root` only, so that nobody else can read the passwords inside it.

PAP is not very secure, as the password is sent in plaintext, so can be read by somebody that monitors your transmission line.

Simple example of `/etc/ppp/pap-secrets`:

```
_Username_ * _Password_
```

4.3 CHAP − Challenge Authentication Protocol

If the provider you are using requires CHAP as the authentication protocol, during the LCP negotiation in PPP this protocol will be asked to use this protocol. When the phone call is connected after using `chat`, `pppd` is started. In this scenario, pppd will send the username and the password, which it will look for in the `/etc/ppp/chap-secrets` file. This file must have read and write permissions only for `root` only, so that nobody else can read the passwords inside it.

CHAP is more secure than PAP, as the password is never sent through the transmission line in plaintext. The authentication server sends a random identifier (the challenge), that the client must encrypt with its password, and then send back to the server.

Simple example of `/etc/ppp/chap-secrets`:

```
_Username_ * _Password_
```

Sometimes an ISP uses PAP and other times CHAP, so it is common to define your username and password in both files.

5.Notes about DNS name resolution

Everytime you connect to an ISP, it is necesary to have configured DNS name resolution, so your computer can find IP addresses associated to a computer name.

IP addresses of your DNS servers are placed into the `/etc/resolv.conf` file.

In a standalone computer connecting to Internet, this file usually contains the IP addresses of your ISP's DNS servers:

```
#/etc/resolv.conf file for ISPname
nameserver 111.222.333.444
nameserver 222.333.444.555
```

In a proxy/firewall computer, this file usually contains its own IP address (or the loopback address, 127.0.0.1), and this computer includes a DNS server that translates DNS names to IP addresses by querying external DNS servers.

```
#/etc/resolv.conf file for local DNS resolution
nameserver 127.0.0.1
```

Installation of a local DNS server is out of the scope of this document. There is a lot of documentation about this, but a good and quick approach can be found in the DNS−Howto (`http://www.linuxdoc.org/HOWTO/DNS-HOWTO.html`).

6.Connecting a standalone computer to an ISP using a modem and PPP

When configuring *Diald* to connect your computer to an ISP, the next steps will be necesary:

- Getting the *Diald* package installed. The quickest way is to install the package that comes with your GNU/Linux distribution.
- Configure DNS resolver (`/etc/resolv.conf` file).
- Check that you can call an ISP. If your GNU/Linux distribution includes an utility to configure a connection, the quickest way will be to use it (pppconfig in Debian, kppp if you use KDE, etc). If you are having problems connecting to an ISP, the PPP−Howto (`http://www.linuxdoc.org/HOWTO/PPP-HOWTO.html`), Modem−Howto (`http://www.linuxdoc.org/HOWTO/Modem-HOWTO.html`) and Serial−Howto (`http://www.linuxdoc.org/HOWTO/Serial-HOWTO.html`) documents can help you.
- Configure username and password in the `/etc/ppp/pap-secrets` and

`/etc/ppp/chap-secrets` files, as mentioned in previous sections.

And finally, going into *Diald*:

- Prepare the *Diald* configuration file (`/etc/diald/diald.options` for version 0.16.5 and `/etc/diald/diald.conf` for later versions).
- Prepare filters file `/etc/diald/standard.filter`, or better, leave that file as is, and modify a copy of it that you can call `/etc/diald/personal.filter`.
- Prepare the script to make the call (`/etc/diald/diald.connect` with execute permissions for root) and instruction file for `chat` (`/etc/chatscripts/provider`), that will be used by the previous script.
- Prepare scripts to be run when the link goes up and down (`/etc/diald/ip-up` and `/etc/diald/ip-down`) if you want to use it (both must have execute permissions for root).
- Prepare script to set and delete routes (`/etc/diald/addroute` and `/etc/diald/delroute`) if you want (both must have execute permissions for root). This step is not necesary if you only use a single *Diald* instance.
- Finally, start the `diald` daemon («`/etc/init.d/diald start`» in Debian, «`/etc/rc.d/init.d/diald start`» in RedHat). Normally, *Diald* package installation process prepares the scripts to run *Diald* when the computer boot up in the /etc/rcX.d directories.

If you make any change in the *Diald* config file when it is running, it is necesary to restart it («`/etc/init.d/diald restart`» in Debian, «`/etc/rc.d/init.d/diald restart`» in RedHat).

6.1 /etc/diald/diald.options or diald.conf file

In this example file you must check for:

- Comm port where your modem is connected. Option `device`.
- Comm port speed to talk with modem. Option `speed`.
- User name to be used in ppp. Option `pppd-options`.
- Retry counters and timers.
- Enabled connection hours. Options `restrict`.
- Decide if you want to use the `ip-up` and `ip-down` scripts. Options `ip-up` and `ip-down`.
- Decide if you want to use the `addroute` and `delroute` scripts. Options `addroute` and `delroute`. Generally it is not needed to modify this scripts, but if you use more than one instance of *Diald* or have a complex configuration, you need it.
- Decide if you use the standar or personal filter file. Options `include`.

```
##########################
# /etc/diald/diald.options

# Device where your modem is connected
device /dev/ttyS0

# Log file
```

```
accounting-log /var/log/diald.log

# Monitoring queue
#fifo /var/run/diald/diald.fifo

# Debug activation
# Activating debug reduces performance
#debug 31

# We use PPP as encapsulator
mode ppp

# Local IP (when you connect this address is automatically modified
# with the ip assigned by your ISP if you use the dinamic option).
local 127.0.0.5

# Remote IP (when you connect this address is automatically modified
# with the ip of the remote server that receives our call).
remote 127.0.0.4

# Subnet mask for the wan link
netmask 255.255.255.0

# The IP addresses will be asigned when connection starts.
dynamic

# If link goes down by remote end, start it again only if there is
# outgoing packets.
two-way

# When link is up, route directly to the real ppp interface, not the proxy
# interface. Not to do this is a performance lost of about 20 per cent.
# There are old kernels that do not support reroute. See diald manual for
# more information
reroute

# Diald will set up the default route the the SLIP interface used as proxy
defaultroute

# Script to set up personalized routes
#addroute "/etc/diald/addroute"
#delroute "/etc/diald/delroute"

# Scripts to execute when the link is up and ready or down and closed.
# In Diald versions 0.9x there is another option called ip-goingdown that
# can be used to run commands when the link is going to be down but is
# still up.
ip-up /etc/diald/ip-up
#ip-down /etc/diald/ip-down

# Scripts used to connect or disconnect the interface
connect "/etc/diald/diald.connect"
#disconnect "/etc/diald/diald.disconnect"

# Use UUCP lock to signal the device is being used
#lock

# We connect over a modem. WARNING: Do not especify this options in the
# ppp options file, because they will conflict with the diald options. To
# see what ppp options that you can not use in the pppd-options option,
# see the diald man page and search for pppd-options
modem
```

```
crtscts
speed 115200

# Some timers and retry options
# See Diald man page for more information
connect-timeout 120
redial-timeout 60
start-pppd-timeout 120
died-retry-count 0
redial-backoff-start 4
redial-backoff-limit 300
dial-fail-limit 10

# Options to be passed to pppd
# This options can be included in the /etc/ppp/options file, that are the
# default options for pppd, but if you need to use different
# configurations of diald for more than one instance, you must put it here
# noauth - do not ask remote for authenticaion.
#          "Infovía Plus" (Spain) do not identify to our machine
# user - our username and isp. Ask your isp for the sintaxis. Some isps,
#        do not need the @isp
pppd-options noauth user usuario@isp

# Hour restriccions.
# This section must be before filters.
# The restrict command is experimental, and can change in other versions
# of diald. Check the man page. (this example has been checked for 0.16,
# but i think it runs in later versions).
# Example: only use in the night from monday to friday, and all day in
# saturday and sunday.
restrict 8:00:00 18:00:00 1-5 * *
down
restrict * * * * *

# No special tarificaion considerations
# (first seconds included in the setup cost, tarify unit in seconds,
# time in seconds to check if it is good to go down)
#impulse 0,0,0
# Bononet Noche (Spain-Telefónica) is billed in seconds after the 160
# first seconds
impulse 160,0,0
# if it would be billed in minuttes and the first 10 will be billed
# always:
#impulse 600,60,10

# Standar filters
#include /etc/diald/standard.filter
# or personal filters
include /etc/diald/personal.filter
```

6.2 Personal filters file

Manipulation of this file must be done very carefully. This file is used to decide when and why to start up the line, maintain it, bring down the line or ignore a packet, depending on the traffic type.

Generally, the *Diald* standar filter file is sufficient for most cases, but perhaps, it may be too restrictive or not restrictive enough in some situations. The `personal.filter` file that is shown has some corrections over the original from the 0.16 version.

In next versions of this document, other commented more restrictive examples will be included.

```
# /etc/diald/personal.filter
# Filter rules shown are the same as in the standard.filter with the
# following changes:
#
# Change 10 to 4 minuttes in "any other tcp conection".
# Added "ignore tcp tcp.fin" to ignore the FIN ACK packets.
# Ignore icmp packets (ping and traceroute don't fire up the interface).
#

# This is a pretty complicated set of filter rules.
# (These are the rules I use myself.)
#
# I've divided the rules up into four sections.
# TCP packets, UDP packets, ICMP packets and a general catch all rule
# at the end.

ignore icmp any

#------------------------------------------------------------------------------
# Rules for TCP packets.
#------------------------------------------------------------------------------
# General comments on the rule set:
#
# In general we would like to treat only data on a TCP link as significant
# for timeouts. Therefore, we try to ignore packets with no data.
# Since the shortest possible set of headers in a TCP/IP packet is 40 bytes,
# any packet with length 40 must have no data riding in it.
# We may miss some empty packets this way (optional routing information
# and other extras may be present in the IP header), but we should get
# most of them. Note that we don't want to filter out packets with
# tcp.live clear, since we use them later to speedup disconnects
# on some TCP links.
#
# We also want to make sure WWW packets live even if the TCP socket
# is shut down. We do this because WWW doesn't keep connections open
# once the data has been transfered, and it would be annoying to have the link
# keep bouncing up and down every time you get a document.
#
# Outside of WWW the most common use of TCP is for long lived connections,
# that once they are gone mean we no longer need the network connection.
# We don't neccessarily want to wait 10 minutes for the connection
# to go down when we don't have any telnet's or rlogin's running,
# so we want to speed up the timeout on TCP connections that have
# shutdown. We do this by catching packets that do not have the live flag set.

# --- start of rule set proper ---

# When initiating a connection we only give the link 15 seconds initially.
# The idea here is to deal with possibility that the network on the opposite
# end of the connection is unreachable. In this case you don't really
# want to give the link 10 minutes up time. With the rule below
# we only give the link 15 seconds initially. If the network is reachable
# then we will normally get a response that actually contains some
# data within 15 seconds. If this causes problems because you have a slow
```

```
# response time at some site you want to regularly access, you can either
# increase the timeout or remove this rule.
accept tcp 15 tcp.syn

# Keep named xfers from holding the link up
ignore tcp tcp.dest=tcp.domain
ignore tcp tcp.source=tcp.domain

# (Ack! SCO telnet starts by sending empty SYNs and only opens the
# connection if it gets a response. Sheesh..)
accept tcp 5 ip.tot_len=40,tcp.syn

# keep empty packets from holding the link up (other than empty SYN packets)
ignore tcp ip.tot_len=40,tcp.live

# Modification by Andres Seco to ignore the FIN ACK packets.
ignore tcp tcp.fin

# make sure http transfers hold the link for 2 minutes, even after they end.
# NOTE: Your /etc/services may not define the tcp service www, in which
# case you should comment out the following two lines or get a more
# up to date /etc/services file. See the FAQ for information on obtaining
# a new /etc/services file.
accept tcp 120 tcp.dest=tcp.www
accept tcp 120 tcp.source=tcp.www
# Same for https
accept tcp 120 tcp.dest=tcp.443
accept tcp 120 tcp.source=tcp.443

# Once the link is no longer live, we try to shut down the connection
# quickly. Note that if the link is already down, a state change
# will not bring it back up.
keepup tcp 5 !tcp.live
ignore tcp !tcp.live

# an ftp-data or ftp connection can be expected to show reasonably frequent
# traffic.
accept tcp 120 tcp.dest=tcp.ftp
accept tcp 120 tcp.source=tcp.ftp

#NOTE: ftp-data is not defined in the /etc/services file provided with
# the latest versions of NETKIT, so I've got this commented out here.
# If you want to define it add the following line to your /etc/services:
# ftp-data        20/tcp
# and uncomment the following two rules.
#accept tcp 120 tcp.dest=tcp.ftp-data
#accept tcp 120 tcp.source=tcp.ftp-data

# If we don't catch it above, give the link 10 minutes up time.
#accept tcp 600 any
# Modificacion de Andres Seco. Solo dejar 4 minutos mas.
accept tcp 240 any

# Rules for UDP packets
#
# We time out domain requests right away, we just want them to bring
# the link up, not keep it around for very long.
# This is because the network will usually come up on a call
# from the resolver library (unless you have all your commonly
# used addresses in /etc/hosts, in which case you will discover
# other problems.)
# Note that you should not make the timeout shorter than the time you
```

```
# might expect your DNS server to take to respond. Otherwise
# when the initial link gets established there might be a delay
# greater than this between the initial series of packets before
# any packets that keep the link up longer pass over the link.

# Don't bring the link up for rwho.
ignore udp udp.dest=udp.who
ignore udp udp.source=udp.who
# Don't bring the link up for RIP.
ignore udp udp.dest=udp.route
ignore udp udp.source=udp.route
# Don't bring the link up for NTP or timed.
ignore udp udp.dest=udp.ntp
ignore udp udp.source=udp.ntp
ignore udp udp.dest=udp.timed
ignore udp udp.source=udp.timed
# Don't bring up on domain name requests between two running nameds.
ignore udp udp.dest=udp.domain,udp.source=udp.domain
# Bring up the network whenever we make a domain request from someplace
# other than named.
accept udp 30 udp.dest=udp.domain
accept udp 30 udp.source=udp.domain
# Do the same for netbios-ns broadcasts
# NOTE: your /etc/services file may not define the netbios-ns service
# in which case you should comment out the next three lines.
ignore udp udp.source=udp.netbios-ns,udp.dest=udp.netbios-ns
accept udp 30 udp.dest=udp.netbios-ns
accept udp 30 udp.source=udp.netbios-ns
# keep routed and gated transfers from holding the link up
ignore udp tcp.dest=udp.route
ignore udp tcp.source=udp.route
# Anything else gest 2 minutes.
accept udp 120 any

# Catch any packets that we didn't catch above and give the connection
# 30 seconds of live time.
accept any 30 any
```

6.3 Making the call

`/etc/diald/diald.connect` file (it must have execute permission):

```
/usr/sbin/chat -f /etc/chatscripts/provider
```

`/etc/chatscripts/provider` file. In this example file you must check the destination phone number:

```
ABORT BUSY
ABORT "NO CARRIER"
ABORT VOICE
ABORT "NO DIALTONE"
```

```
ABORT "NO ANSWER"
"" ATZ
OK ATDT123456789
CONNECT \d\c
```

6.4 Connection start script

It must have execute permission.

This script can be used to many tasks (synchronize time, send the queued mail, get incoming mail, etc.).

In the example, a message is sent to `root` with data passed to the script (interface, subnet mask, local ip address, remote ip address and cost for routing):

```
#!/bin/sh

iface=$1
netmask=$2
localip=$3
remoteip=$4
metric=$5

# Set the time and date
# netdate ntp.server.somecountry

# Run the mail queue
# runq

echo `date` $1 $2 $3 $4 $5 | mail -s "diald - conecting" root@localhost
```

7.Conecting a computer to a group of different ISPs with a modem and PPP

Many times, one standalone computer does not only connect to just one network. It is common to connect to different networks or to the Internet using some different service providers. In this case, changing configuration files each time you want to connect to a different site can be annoying.

The solution i propose here consist in using different sets of configuration files for each different connection. You can find here some scripts to automate changing from one to another.

7.1 Note about sending mail using a relay host

If your email client program uses a local Message Transfer Agent with a `smtp` relay host to send all messages, or if you use a email client program that sends the messages directly to your provider's `smtp` server, changing where you are connecting means you need to reconfigure this option for the `smtp` relay server. This is because the providers usually check if the receipt mailbox is local or to any domain directly maintained by this provider or if the origin ip address is from the range of ip addresses that this provider assigns, to avoid having an open relay server that can be used to send spam, anonymous message and so on.

In the following examples, you will find how to change this parameter in the *Smail* configuration files in a simple configuration where all external messages are sent to a `smtp` relay server. If you use another Message Transfer Agent (MTA) in your system, you can send me what you must change in your MTA to include it here. If you use an email client program that directly sends to the external `smtp` server (Kmail, Netscape, etc.) send me your changes too.

7.2 Scripts to automate multiple connections and changing from one to another

Starting up

First of all, create a subdirectory of `/etc/diald` called `providers` where you store your scripts to automatically change from one to another provider and the subdirectories with the set of files to configure each of the providers connections.

With the next script this directory is created and filled with the current configuration files from *Diald*, *chat*, *pppd* and *Smail*, that will be treated as a template for the next configurations.

```
#!/bin/sh
#File /etc/diald/providers/setupdialdmultiprovider
mkdir /etc/diald/providers
mkdir /etc/diald/providers/setup
cp /etc/ppp/pap-secrets /etc/diald/providers/setup
cp /etc/ppp/chap-secrets /etc/diald/providers/setup
cp /etc/resolv.conf /etc/diald/providers/setup
cp /etc/diald/diald.options /etc/diald/providers/setup
cp /etc/diald/standard.filter /etc/diald/providers/setup
cp /etc/diald/personal.filter /etc/diald/providers/setup
cp /etc/diald/diald.connect /etc/diald/providers/setup
cp /etc/chatscripts/provider /etc/diald/providers/setup
cp /etc/diald/ip-up /etc/diald/providers/setup
cp /etc/diald/ip-down /etc/diald/providers/setup
cp /etc/smail/routers /etc/diald/providers/setup
```

New provider

With the next script the template configuration will be copied to a new directory to prepare it for a new provider connection or a new net connection. This script (`/etc/diald/providers/newdialdprovider`) will need a parameter with the provider or net name.

```
#!/bin/sh
#File /etc/diald/providers/newdialdprovider
mkdir /etc/diald/providers/$1
cp /etc/diald/providers/setup/* /etc/diald/providers/$1
```

Now, you will modify as you need the new files in `/etc/diald/providers/provdidername`, being `providername` the parameter passed to `newdialdprovider`.

Changing from one to another

At the end, with this script you will change all your configuration files related to *Diald* to connect to another provider or net. I use symbolic links to avoid using duplicate files. Using symbolic links, if you change any config file in its original location like `/etc/resolv.conf`, the change is really made in the `/etc/diald/providers/providername/resolv.conf` file.

```
#!/bin/sh
#File /etc/diald/providers/setdialdprovider
/etc/init.d/diald stop
#wait for Diald to stop.
sleep 4
ln -sf /etc/diald/providers/$1/pap-secrets /etc/ppp
ln -sf /etc/diald/providers/$1/chap-secrets /etc/ppp
ln -sf /etc/diald/providers/$1/resolv.conf /etc
ln -sf /etc/diald/providers/$1/diald.options /etc/diald
ln -sf /etc/diald/providers/$1/standard.filter /etc/diald
ln -sf /etc/diald/providers/$1/personal.filter /etc/diald
ln -sf /etc/diald/providers/$1/diald.connect /etc/diald
ln -sf /etc/diald/providers/$1/provider /etc/chatscripts
ln -sf /etc/diald/providers/$1/ip-up /etc/diald
ln -sf /etc/diald/providers/$1/ip-down /etc/diald
ln -sf /etc/diald/providers/$1/routers /etc/smail
/etc/init.d/diald start
```

8.Connecting a proxy/firewall to an ISP using a modem and PPP

Connecting a private net to the Internet with dedicated server which handles packet routing from the local network to the Internet along with proxy/caching services and security firewalling is a complex theme that is beyond the scope of this document. There are other «Howto» documents that handle these topics much more comprehensively. At the end of this document you can find a list of links and references to such documents.

Here, we are only configuring *Diald* supposing that the computer already uses IP–Masquerading, has a web proxy like *Squid* or similar working, an ISP connection correctly configured and that access security to TCP/UDP ports have been revised (`/etc/inetd.conf` file and others like `securetty`, `host.allow`, etc).

Basically, the only need is to reconfigure the rules for masquerading/filtering/accessing each time the set of interfaces change, that is, when the interface ppp0 is stablished and when it is deleted. A good location to do that are the ip–up and ip–down scripts from *pppd*.

8.1 Example for Debian 2.1

With Debian, it is sufficient to install the *ipmasq* package answering that you want to change rules sinchronously with *pppd* when seting it up. Two scripts will be created inside `/etc/ppp/ip-up.d` and `/etc/ppp/ip-down.d` directories to call `/sbin/ipmasq`, a script that analizes existing interfaces and makes a simple configuration that is valid in many cases, but you can personalize it using rule files in `/etc/ipmasq/rules`.

The only correction after installing this package is to change when the startup script for *ipmasq* is run, deleting the symbolic link from `/etc/rcS.d` and creating a new one in `/etc/rc2.d` to run it after `S20diald`. Now, when `ipmasq` is executed to analyze interfaces `sl0` already exist. `S90ipmasq` is a good name for this symbolic link to `/etc/init.d/ipmasq`.

Using Debian there is no need to worry about the kernel version, as the `/sbin/ipmasq` script uses `ipfwadm` or `ipchains` as needed.

8.2 Example for Suse 6.1

This example is from Mr Cornish Rex, `troll@tnet.com.au`.

The following ip–masp and routing control commands are for use with version 2.2 kernels, using ipchains, but they are not valid for version 2.0 kernels.

We are going to supose that the ethernet interface has the 192.168.1.1 ip address with 16 bit netmask, that is, 255.255.0.0.

This is the `/etc/ppp/ip-up` file:

```
#!/bin/sh
# $1 = Interface
# $2 = Tty device
# $3 = speed
# $4 = local ip
# $5 = remote ip
# $6 = ipparam
/sbin/ipchains -F input
/sbin/ipchains -P input DENY
/sbin/ipchains -A input -j ACCEPT -i eth0 -s 192.168.0.0/16 -d 0.0.0.0/0
/sbin/ipchains -A input -j DENY -p udp -i $1 -s 0.0.0.0/0 -d $4/32 0:52 -l
/sbin/ipchains -A input -j DENY -p udp -i $1 -s 0.0.0.0/0 -d $4/32 54:1023 -l
/sbin/ipchains -A input -j DENY -p tcp -i $1 -s 0.0.0.0/0 -d $4/32 0:112 -l
/sbin/ipchains -A input -j DENY -p tcp -i $1 -s 0.0.0.0/0 -d $4/32 114:1023 -l
/sbin/ipchains -A input -j DENY -p tcp -i $1 -s 0.0.0.0/0 -d $4/32 6000:6010 -l
/sbin/ipchains -A input -j DENY -p icmp --icmp-type echo-request \
-i $1 -s 0.0.0.0/0 -l
/sbin/ipchains -A input -j DENY -p icmp -f -i $1 -s 0.0.0.0/0 -l
/sbin/ipchains -A input -j DENY -p udp -i $1 -s 0.0.0.0/0 -d $4/32 5555 -l
/sbin/ipchains -A input -j DENY -p udp -i $1 -s 0.0.0.0/0 -d $4/32 8000 -l
/sbin/ipchains -A input -j DENY -p tcp -i $1 -s 0.0.0.0/0 -d $4/32 8000 -l
/sbin/ipchains -A input -j DENY -p udp -i $1 -s 0.0.0.0/0 -d $4/32 6667 -l
/sbin/ipchains -A input -j DENY -p tcp -i $1 -s 0.0.0.0/0 -d $4/32 6667 -l
/sbin/ipchains -A input -j DENY -p tcp -i $1 -s 0.0.0.0/0 -d $4/32 4557 -l
/sbin/ipchains -A input -j DENY -p tcp -i $1 -s 0.0.0.0/0 -d $4/32 4559 -l
/sbin/ipchains -A input -j DENY -p tcp -i $1 -s 0.0.0.0/0 -d $4/32 4001 -l
/sbin/ipchains -A input -j DENY -p tcp -i $1 -s 0.0.0.0/0 -d $4/32 2005 -l
/sbin/ipchains -A input -j DENY -p tcp -i $1 -s 0.0.0.0/0 -d $4/32 6711 -l
/sbin/ipchains -A input -j DENY -i $1 -s 192.168.0.0/16 -d 0.0.0.0/0 -l
/sbin/ipchains -A input -j ACCEPT -i $1 -s 0.0.0.0/0 -d $4/32
/sbin/ipchains -A input -j ACCEPT -i lo -s 0.0.0.0/0 -d 0.0.0.0/0
/sbin/ipchains -A input -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0 -l

/sbin/ipchains -F output
/sbin/ipchains -P output DENY
/sbin/ipchains -A output -j ACCEPT -i eth0 -s 0.0.0.0/0 -d 192.168.0.0/16
/sbin/ipchains -A output -j DENY -i $1 -s 192.168.0.0/16 -d 0.0.0.0/0 -l
/sbin/ipchains -A output -j ACCEPT -i $1 -s $4/32 -d 0.0.0.0/0
/sbin/ipchains -A output -j ACCEPT -i lo -s 0.0.0.0/0 -d 0.0.0.0/0
/sbin/ipchains -A output -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0

/sbin/ipchains -F forward
/sbin/ipchains -P forward DENY
/sbin/ipchains -M -S 120 120 120
/sbin/ipchains -A forward -j MASQ -s 192.168.1.0/24
/sbin/ipchains -A forward -j DENY -s 0.0.0.0/0 -d 0.0.0.0/0

exit 0
```

This is the `/etc/ppp/ip-down` file:

```
#!/bin/sh
# $1 = Interface
```

```
# $2 = Tty device
# $3 = Speed
# $4 = Local ip
# $5 = Remote ip
/sbin/ipchains -F input
/sbin/ipchains -F output
/sbin/ipchains -F forward
/sbin/ipchains-restore < /etc/ppp/orig.chains
```

Last file in last script, orig.chains, is the following file (original status of ipchains):

```
# orig.chains
# created with: ipchains-save > orig.chains
:input ACCEPT
:forward ACCEPT
:output ACCEPT
-A input -s 0.0.0.0/0.0.0.0 -d 192.168.1.1/255.255.255.255
-A output -s 192.168.1.1/255.255.255.255 -d 0.0.0.0/0.0.0.0
```

8.3 Example for Slackware 3.6

This example is from Hoo Kok Mun, `hkmun@pacific.net.sg`.

This is the most simple example i have seen, but fully functional. From the beginning, this example configures masquerading, before the `sl0` interface exists, and it does not change when the `ppp0` interface appears. If you need advanced security considerations, it may be a little limited.

```
#/etc/rc.d/rc.local
/sbin/ipfwadm -F -p deny
/sbin/ipfwadm  -F -a m -S 192.168.0.0/24 -D 0.0.0.0/0
```

As you can see, it is for version 2.0 kernels.

9.Programs and versions used

To write this document i have used the following diald versions:

- Diald 0.16.5 – Last version maintained by the original diald autor.
- Diald 0.99.3 – Last version until the first edition of this document.

And the following pppd versions:

- pppd 2.3.5

Diald 0.16.5 version is perhaps the most extended, and the one that many Linux distributions include. It is suficient for many sites, and it is very reliable, but, of course, later versions have many interesting capabilites.

10.More information

Original information from where this document has been obtained can be found in the man pages about `diald`, `diald−examples`, `diald−control`, `diald−monitor`, `dctrl`, `pppd`, `chat`, as well as from information in the /usr/doc directories and in web pages of this packages:

- New Diald Official Home Page: `http://diald.sourceforge.net/`
- Download of new versions: `ftp://diald.sourceforge.net/pub/diald/`
- Previous Diald home page: `http://diald.unix.ch`
- Old Diald home page until 0.16.5 version: `http://www.loonie.net/~erics/diald.html`
- pppd FTP site: `ftp://cs.anu.edu.au/pub/software/ppp/`
- Other site: `http://www.p2sel.com/diald`
- One more: `http://rufus.w3.org/linux/RPM/`

There is a mailing list for discussion about diald on David S. Miller's mailing list server at vger.rutgers.edu. To subscribe, send a message to `Majordomo@vger.rutgers.edu` with the text «subscribe linux−diald» IN THE MESSAGE BODY.

An archive of the list can be found in `http://www.geocrawler.com`.

There are also multiple RFC documents (Request For Comments) that define how the PPP encapsulated lines and its associated protocols (LCP, IPCP, PAP, CHAP, ...) must work. You can find these documents in the /usr/doc/doc−rfc directory and some World Wide Web sites, like `http://metalab.unc.edu` and `http://nic.mil/RFC`. You can ask for information about RFCs in `RFC−INFO@ISI.EDU`.

The following «Howtos» can help you:

- DNS−HOWTO − `http://www.linuxdoc.org/HOWTO/DNS-HOWTO.html`
- Firewall−HOWTO − `http://www.linuxdoc.org/HOWTO/Firewall-HOWTO.html`
- IP−Masquerade−HOWTO − `http://www.linuxdoc.org/HOWTO/IP-Masquerade-HOWTO.html`
- IPCHAINS−HOWTO − `http://www.linuxdoc.org/HOWTO/IPCHAINS-HOWTO.html`
- Modem−HOWTO − `http://www.linuxdoc.org/HOWTO/Modem-HOWTO.html`
- NET3−4−HOWTO − `http://www.linuxdoc.org/HOWTO/NET3-4-HOWTO.html`
- PPP−HOWTO − `http://www.linuxdoc.org/HOWTO/PPP-HOWTO.html`
- Serial−HOWTO − `http://www.linuxdoc.org/HOWTO/Serial-HOWTO.html`