Connecting a Private Network to an ISP with Solaris

Doug Palmer*

February 15, 2003

Contents

*doug@charvolant.org, http://www.charvolant.org/ doug

List of Figures

1 Introduction

My home network consists of a Sparc running Solaris, A PC with Windows 95 and another PC running Linux. What I wanted to do was allow any machine on this private network to connects to my ISP, ORAC,[17] via a single modem, with an IP address being dynamically assigned when I dial in.

My Linux system is actually my old 486/33 set up to run Linux so that I can have a play with Linux. Unfortunately, this system chokes on any high data transfer rate, so I need to use the Sparc to transfer data. Solaris's PPP support is pretty primitive and, in particular, doesn't seem to support the hooks needed to support NAT and packet filtering over a dynamic link neatly.

Below is a guide to connecting to ORAC (or another ISP) using network address translation on Solaris. To accomplish this, is am using two packages: One is IP-Filter,[11] a neat package that allows me to filter packets and do NAT (network address translation). The other is DP,[5] a PPP package that does support dynamic address assignment.

Note that my way is not the only way. Rachel Polanskis has set up a similar arrangement using Free PPP, rather than my DP.[18, 8]

Conventions

The following typeface conventions are used:

`/path/name`	A path name (directory).
`filename.txt`	A file name (with or without a path).
program	A program, sometimes with a path name.
host.domain.com	A host or machine name, sometimes with a domain attached.
variable	A variable or varying parameter.
PATH	An environment variable.
`DPDIR=/opt/dp`	A fragment of script or configuration file.
`ls -l`	A shell command.
POP3	A protocol.

In many places a sentence naturally ends with a command or file name. A full-stop after a file name is likely to confuse things, so I have adopted the habit of leaving off the final full-stop. The start of a new sentence can be deduced by the initial capital letter.

Changes

This is version 1.1.1 of this document. Changes since the previous version are:

- **Finit** Well, that's all folks. My Solaris system no longer acts as my router/firewall/what-have-you. Instead, I have a Linux system and a permanent connection. Unfortunately, it would appear, IP-Filter doesn't run on most Linux systems. Instead, I'm using the — inferior, in my opinion – IP-Chains.

  If IP-Filter is ever successfully ported (not a trivial task, by all accounts), I'll be back.

- I've added the local DNS reverse lookup file. This has been missing for some time.

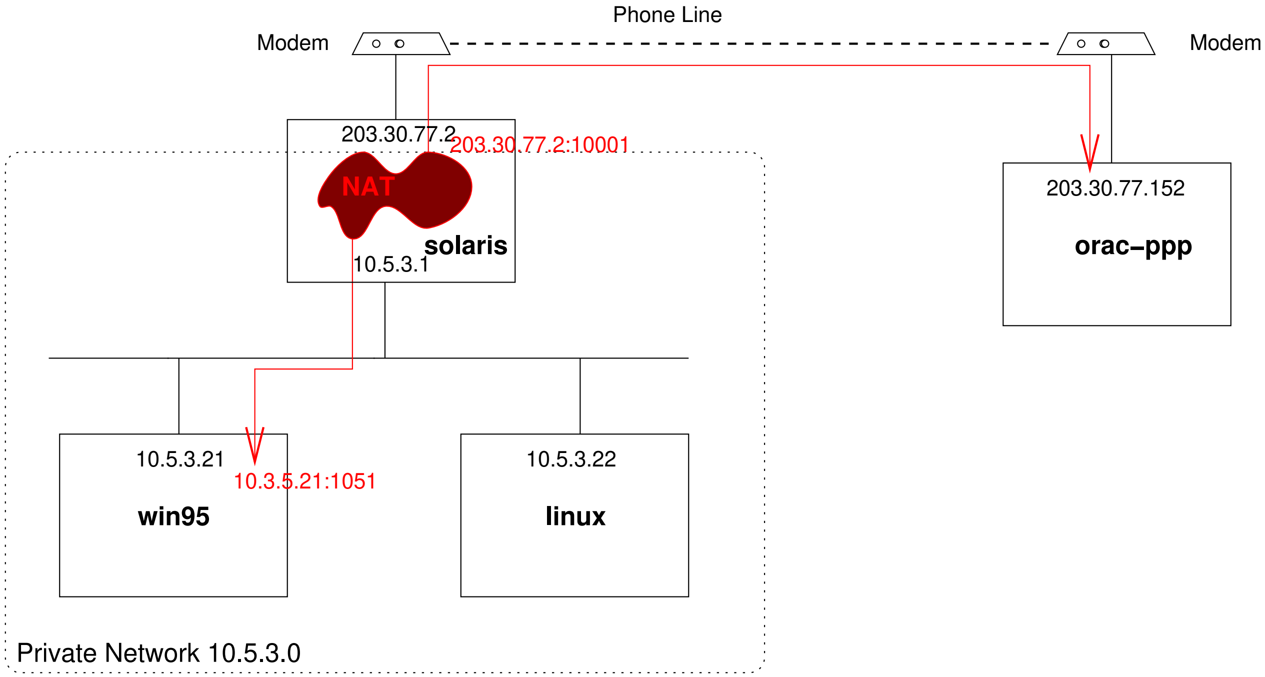

Figure 1: Example Private Network Configuration

2 Theory

Figure 1 shows what is intended. What we have is a private network, 10.5.3.0, which has three machines on it called solaris, win95 and linux. (Not their real names. All names and dates have been changed to protect the innocent.) These machines have IP addresses of 10.5.3.1, 10.5.3.21 and 10.5.3.22 respectively.

solaris is also a router. It has another IP address, 203.30.77.2 which is connected to a modem. The modem, in turn, speaks to a system at ORAC. ORAC is in turn connected to the wider Internet.

If, say, win95 wants to connect to some other host on the Internet, it needs to route all of its packets to solaris. solaris can then send these packets on to ORAC, which can then send them on to whatever host that it needs to talk to.

A problem arises whenever the host that win95 is connected to wants to send something back to win95. The IP address 10.5.3.21 is not recognised on the Internet; any address on the network 10.0.0.0 is assumed to be a private network which shouldn't be out in the wider world. As such, packets intended to return to win95 simply get lost.

Network address translation (NAT) solves this problem. Any TCP or UDP connection goes through a port, with a particular port number. As the packets pass through solaris, the IP address of the sending machine is translated to be 203.30.77.2, the Internet address for solaris, and the port number of 1051 is changed to an unused port number on solaris, in this case 10001. The remote host can now send packets to 203.30.77.2, a valid Internet address at port number 10001. solaris translates the addresses in these packets and sends them on to win95, which isn't even aware of anything odd happening.

The standard protocols for the Internet are specified by the misleadingly-named RFCs, or requests for comment.[23] RFC 1918 contains the (latest) specifications for private networks.[4] RFC 1631 contains the specifications for NAT.[6]

3 Preparation

This section covers the various tools and packages that you need and pre-installation configuration that you will need to do. As well as the packages themselves, you will need a suitable compiler for Solaris, which does not come with a bundled compiler.

3.1 Compilers, etc.

Both the IP-Filter and DP packages come as source code, so you will need a C compiler. Unfortunately, Solaris is missing a C compiler, so:

- If you have a C compiler, then no problem.
- You can buy the Sun C compiler.
- You can get a free C compiler.

I have a free C compiler, the GNU C compiler (gcc), supplied by the magnificent Free Software Foundation.[9] You can get the latest version of the compiler from a number of FTP sites.[10] The version that I have used is 2.8.1.

Since the source code for gcc is compressed and tarred, you will also need to get a copy of gzip, the GNU compression utility. Gzip is also available in source form, with the latest version I have used being 1.2.4. It's also worth noting that WinZip can handle tarred, gzipped files.

Sun have still taken the compilers out of Solaris 2.x, so you're still stuck. However, there are pre-compiled packages for Solaris 2.x gcc available in some of the FTP archives. These binaries are usually behind the source versions, so it makes sense to use these binaries to compile the latest source code so that you have the latest version of the compiler. The Solaris Freeware pages contain source code and packages for a lot of free software for Solaris 2.5, 2.6 and 7.[20]

You will also need a copy of make. The GNU version of make experiences some difficulty with the Makefiles supplied by both IP-Filter and DP. However, the Sun-supplied **/usr/ccs/bin/make** has no problem with either package. Make sure that you have `/usr/ccs/bin` in your `PATH` environment variable before wherever you have GNU make installed.

3.2 Packages

The following packages need to be downloaded for installation.[1]

Package	URL	Version
IP-Filter	`ftp://coombs.anu.edu.au/pub/net/firewall/ip-filter`	3.2.10
DP	`ftp://ftp.acn.purdue.edu/dp`	4.0
BIND	`ftp://ftp.isc.org/isc/bind/src/8.2/`	8.2

It is a good idea to download the documentation package for BIND, as well as the source code. When extracting the files from the BIND archive, be aware that they do not place themselves into a convenient, named directory, but just spew out in place.

3.3 Configuring Your Private Network

Before starting on this adventure, you need to configure the private network correctly so that hosts and network numbers are recognised correctly and so that external host names can be found.

3.3.1 Host and Network Configuration

To configure the private network, you will need to allocate IP numbers to the machines on your network and set up hosts files on the Unix systems that contain the correct host names. If you are setting your systems up from scratch, the configuration scripts will probably set up much of what is in here automatically.

I'm assuming that you have an Ethernet network here. If you have some exotica, such as token-ring, then you will need to find the appropriate names for things.

Choosing a network number The private network must have a network number which comes from a particular set of numbers set aside by the NIC for use as private networks.[16, 4] This way, any packets which stray out of your network will be rejected by Internet routers and not run amok on the information superhighway.

Your choices are: 10, 172.16–172.31 and 192.168.0–192.168.255. It's logically sensible to choose a class C network, such as 192.168.23, but you can also pick a class B or class A network and subnet, if you wish. For reasons of pure spite, I'm going to use an example network which is the class A network, 10, subnetted to be 10.5.3.

From this network number, you need to assign IP numbers to each machine. In this example, the IP numbers are 10.5.3.1 for solaris 10.5.3.21 for win95 and 10.5.3.22 for linux.

Unix-specific Configuration The Unix systems (solaris and linux) can use files in the `/etc` directory to identify hosts and network information. As we configure the other parts of the system, we will add to these files, but for now, we can set up the various files as needs be.

The `/etc/hosts` file The `/etc/hosts` file contains symbolic names for (usually) local hosts. It contains tab separated entries for each host and its IP address. A sample hosts file is:

```
127.0.0.1    localhost loghost
#
#    Private network
#
10.5.3.1     solaris
10.5.3.21    win95
10.5.3.22    linux
```

[1] Versions are the version that I have. There may be more recent versions than this.

Comments have a hash symbol (#) in front of them. 127.0.0.1 is the *loopback address*, essentially meaning "this machine". Hosts can have multiple aliases (eg. loghost for localhost in the above example).

The `/etc/networks` file The `/etc/networks` file contains symbolic names for networks, in a similar format to the the `/etc/hosts` file. A sample networks file is:

```
loopback    127      # Intra-machine communication network
private     10.5.3   # Local private network
```

It is not strictly necessary to set up the networks file, but it does help.

The `/etc/netmasks` file If you have been sensible, and chosen a private network from the 192.-168.0–192.168.255 range, then you do not have to worry about setting up the `/etc/netmasks` file. Strictly speaking, you do not have to do this with the other networks as well. However, you can set things up so that you are running a small subnetwork.

The `/etc/netmasks` file says how much of the IP number you have is network and how much is host number. In the example that I am using, the first three IP numbers represent the network and the last number is the host number. A sample `/etc/netmasks` file is:

```
# Network   Mask
#
10.0.0.0    255.255.255.0
```

Solaris-specific Configuration This section contains configuration specific to Solaris systems.

The `/etc/hostname.le0` file consulted at startup to see what host this machine is. The file contains a single line giving the host name (solaris in this case).

The `/etc/nsswitch.conf` file tells the Solaris systems where to go to get names. Solaris can, as well as using the files described above, use a variety of network based name servers.

A pre-built `nsswitch.conf` called `/etc/nsswitch.files` already exists. You can use this file by copying it onto the `/etc/nsswitch.conf` file. When (if?) you set up a DNS server, you will need to modify `/etc/nsswitch.conf` to allow DNS lookup; this is covered in section 3.4 on DNS.

The `/etc/services` file needs to be extended. Solaris does not come with POP3, the mail retrieval protocol listed in its services file. If you plan to use programs that use POP3 on the Solaris system, you will probably have to add the following line to `/etc/services`:

```
pop-3    110/tcp    pop3    # Post Office
```

Routing configuration for the Solaris system, which is itself a gateway is somewhat tricky and is covered in the section on installing DP (section 4.2.3).

Linux-specific Configuration This section contains configuration details specific to Linux. The Linux distribution that was used here was Slackware 3.2 — distributions have come a long way since then.

The `/etc/HOSTNAME` file performs the same function as the Solaris `/etc/hostname.le0` file and contains the same sort of information.

The `/etc/rc.d/rc.inet1` file contains some configuration commands for the Ethernet interface on Linux systems. You will need to edit this file to contain appropriate IP address and gateway information. In the sample network, we have the following lines:

```
# Edit for your setup.
IPADDR="10.5.3.22"    # REPLACE with YOUR IP address!
NETMASK="255.255.255.0"    # REPLACE with YOUR netmask!
NETWORK="10.5.3.0"    # REPLACE with YOUR network address!
BROADCAST="10.5.3.255"    # REPLACE with YOUR broadcast address, if you
            # have one. If not, leave blank and edit below.
GATEWAY="10.5.3.1"    # REPLACE with YOUR gateway address!
```

Windows 95-specific Configuration To start using Windows 95, you will need to install the network driver and TCP/IP network protocol, assuming that you have not done so already. Once this is done, open the Network section of the Control Panel and choose the TCP/IP properties. For the example network, we have:

Tab	Field	Value
IP Address	Specify an IP Address	
	IP Address	10.5.3.21
	Subnet Mask	255.255.255.0
WINS Configuration	Disable WINS Resolution	
Gateway	Add Gateway	10.5.3.1
DNS Configuration	See section 3.4.3 on DNS	

3.4 DNS Configuration

DNS allows systems to look up host names, both within the private network and across the Internet. There are other ways of doing this, for example you could just run identical hosts files on all your machines. But somehow, this is more fun. In addition, Windows 95 machines examine their hosts databases in an odd order which makes using host tables difficult; the win95netbug FAQ rovides information on this, as well as help on how to reconfigure win95 to reorder the lookup order.[24]

You can set up the DNS server on any Unix system. The standard Solaris 2.6 install comes with BIND 4.9.x, suitably altered to deal with the additional Solaris naming systems. However, 4.9.x seems to do a great deal of maintenance work at fairly random times. This is fine if you have a permanent connection, but not so fine with a dialup. As a result, I have got BIND 8.2 from the Internet Software Consortium and installed that instead.[12]

As a result, I'm going to talk about BIND 8.2 here. The version of BIND which comes with Solaris 2.5.1 seems a lot more dialup friendly. So, if you are using an older version of Solaris, you can probably stay with the supplied version; the boot files are considerably different, however, so you will need to examine the documentation.

3.4.1 Compiling BIND 8.2

Compiling BIND 8.2 is fairly straightforward. However, before compiling, you will need to apply any patches to the supplied source code. The command **`patch < patch1`** will apply the patch.

To configure the Makefiles, use **`make depend`** from the `src` directory. To make the programs, use **`make`** To install, first remove the `.settings` file and then use **`make install`** Installation goes into directories under `/usr/local`

The source code for BIND does not come with any manual pages. A separate documentation download contains man pages. The install Makefile for the man pages, however, expects a BSD-style install and, therefore, you will need to put `/usr/ucb` at the start of your `PATH` environment variable if you want a painless install. You may also need to modify the Makefile to remove pre-formatted man pages.

3.4.2 DNS Server Configuration

To configure the DNS server, you need to set up a number of (text) database files. The DNS server daemon (called **named**) first consults a boot file. This boot file tells the daemon to consult a series of further database files which gives it enough information to start serving names.

Choosing a Domain Before setting up your DNS server, you will need to choose a domain name. Since nobody, except you, will accept your domain as a genuine domain (we hope) you have a reasonable amount of latitude over what domain names you can pick.

Since I work for AFS, I have decided to call my domain canberra.afs.net.au I am pretty certain that nobody else is going to use that name.

```
#
# Boot file for server solaris, primary server
# for flibble.orac.net.au.
#

options {
    directory "/var/named";
    forwarders {
        299.18.99.151;
        299.8.183.1;
    };
    forward first;
    dialup yes;
    heartbeat-interval 1440;
};

zone "." {
    type hint;
    file "named.ca";
};

zone "flibble.orac.net.au." {
    type master;
    file "private.hosts";
};

zone "3.5.10.in-addr.arpa." {
    type master;
    file "private.rev";
};

zone "0.0.127.in-addr.arpa." {
    type master;
    file "private.local";
};
```

Figure 2: Sample `named.conf` File for BIND

You'll have to decide what domain name to pick. As a suggestion, if you are connecting to ORAC, then a domain of *whimsical-name*.orac.net.au might be a good idea. On the other hand, I haven't consulted ORAC about this, so it might not.

For the purposes of example, I an going to use the domain name flibble.orac.net.au

The Boot File The boot file is consulted by named to on startup, so that initial options and data can be loaded. A sample `/usr/local/etc/named.conf` file for a server is shown in figure 2.

The meaning of the lines in this file are:

directory The directory in which the database files will be found. The `/var/named` directory is traditional, although inexplicable, since configuration files usually go under pathname/etc

forwarders A list of IP addresses to forward queries to. This saves us from doing all the work of working down the domain tree when finding a domain. You can set your forwarders to the name servers supplied by your ISP.

forward The forwarding behaviour. Queries are first forwarded to the addresses listed in forwarders. Only if no forwarding server responds will this server do its own work. Setting this option to *only* means that this server never does its own search.

dialup Indicate that this is a dialup machine and that, therefore, maintenance work should be grouped together rather than done at any old time. This option is the main reason for my installing BIND 8.2.

```
;
;     Initial cache data for named servers
;     Servers
;
.            518400    IN    NS    D.ROOT-SERVERS.NET.
.            518400    IN    NS    E.ROOT-SERVERS.NET.
.            518400    IN    NS    I.ROOT-SERVERS.NET.
.            518400    IN    NS    F.ROOT-SERVERS.NET.
.            518400    IN    NS    G.ROOT-SERVERS.NET.
.            518400    IN    NS    A.ROOT-SERVERS.NET.
.            518400    IN    NS    H.ROOT-SERVERS.NET.
.            518400    IN    NS    B.ROOT-SERVERS.NET.
.            518400    IN    NS    C.ROOT-SERVERS.NET.
;
;     Addresses
;
D.ROOT-SERVERS.NET.    3600000    IN    A    128.8.10.90
E.ROOT-SERVERS.NET.    3600000    IN    A    192.203.230.10
I.ROOT-SERVERS.NET.    3600000    IN    A    192.36.148.17
F.ROOT-SERVERS.NET.    3600000    IN    A    192.5.5.241
G.ROOT-SERVERS.NET.    3600000    IN    A    192.112.36.4
A.ROOT-SERVERS.NET.    3600000    IN    A    198.41.0.4
H.ROOT-SERVERS.NET.    3600000    IN    A    128.63.2.53
B.ROOT-SERVERS.NET.    3600000    IN    A    128.9.0.107
C.ROOT-SERVERS.NET.    3600000    IN    A    102.33.4.12
```

Figure 3: Initial `named.ca` Cache File for BIND

heartbeat-interval The interval (in minutes) between performing maintenance activities. This option is set, by default, to 60 minutes. Setting it to 1440 minutes means than maintenance is done once a day.

zone Data about a particular domain. The string in double quotes give the domain name, with . being the top-level, root domain. The name server is to be the master server for the domain flibble.orac.net.au (name to IP-address) and the networks 10.5.3 and 127.0.0 (IP-Address to name).

type The type of domain data. *hint* means that the data is there so that the top-level domain servers can be found. *master* means that this is the master server for this domain.

file The file to read the name server data from. These files are discussed below.

The `/var/named/named.ca` File The server, when it is doing its own work, rather than forwarding, needs a start point for searching for domains. This file contains the IP addresses of the servers for the root domain name servers. The base version of this file is shown in figure 3. The first section of this file states that the name servers (NS) for the root domain (.) are the ones listed. The second section says, for each name server, what its IP-address (A) is. The numbers (518400 and 3600000) give the time-outs in seconds for these entries; these figures should be large enough for the time out not to be a problem.

Once you have this file, it's a good idea to pick up the most current official version at `ftp://rs.internic.net/domain/named.root`.

The `/var/named/private.hosts` file This file contains the IP addresses of the private network. A sample `private.hosts` file is shown in figure 4.

Some explanation of the various codes is probably in order:

@ *Domain.* This is a short-hand for the domain given by the named.conf file (flibble.orac.net.au in this case).

IN *Internet.* Indicates that we are talking about the Internet class of records. Supposedly, there are other possible classes here.

```
;
;    Hosts file for domain flibble.orac.net.au.
;
;name    ttl    class    type    data
;
;    Source of authority
@        IN    SOA    solaris.flibble.orac.net.au. root.solaris.flibble.orac.net.au. (
                2000050201    ; Serial
                10800    ; Refresh - 3 hours
                3600    ; Retry - 1 hour
                432000    ; Expire - 1 week
                86400)    ; Minimum - 1 day
         IN    NS    solaris.flibble.orac.net.au.
;
;    Machines for the flibble.orac.net.au domain
;
;name    ttl    class    type    data
localhost       IN       A       127.0.0.1
solaris         IN       A       10.5.3.1
win95           IN       A       10.5.3.21
linux           IN       A       10.5.3.22

;
;    Aliases
;
mail            IN       CNAME   solaris
www             IN       CNAME   solaris
;
;    Domain mailing addresses
;
flibble.orac.net.au.    IN    MX    10    solaris.flibble.orac.net.au.
flibble.orac.net.au.    IN    A     10.5.3.1
```

Figure 4: Sample `private.hosts` File for BIND

```
;
;	Reverse address file for domain flibble.orac.net.au
;
;name	ttl	class	type	data
;
;	Source of authority
@	IN	SOA	solaris.flibble.orac.net.au. root.solaris.flibble.orac.net.au. (
			2000050201	; Serial
			10800	; Refresh - 3 hours
			3600	; Retry - 1 hour
			432000	; Expire - 1 week
			86400)	; Minimum - 1 day
	IN	NS	solaris.flibble.orac.net.au.
;
;	Machines names
;
;name	ttl	class	type	data
1	IN	PTR	solaris.flibble.orac.net.au.
21	IN	PTR	win95.flibble.orac.net.au.
22	IN	PTR	linux.flibble.orac.net.au.
```

Figure 5: Sample `private.rev` File for DNS

SOA *Source of Authority.* This entry contains information on which machine is the primary name server for information about this domain (solaris.flibble.orac.net.au in this case) and who to contact in the case of trouble (root.solaris.flibble.orac.net.au). The serial number is used to indicate where changes have occurred. The other numbers give the time to expiry of the information broadcast by this name server.

The serial numbers need to increase with each version of the file. A fairly common practice is to use *YYYYMMDDVV* with the year, month and day being the date of update and *VV* being the version number within the day. In the past, serial numbers of the form *1.2* were common, but this is now deprecated.

NS *Name Server.* This line indicates that solaris.flibble.orac.net.au is the name server for this domain.

A *Address.* These lines give the IP addresses of the various hosts.

CNAME *Canonical Name.* These lines give canonical names (aliases) for various common names. These names are not strictly needed, but redirect requests to www.flibble.orac.net.au etc. to solaris.flibble.orac.net.au

MX *Mail Exchange* This line gives the system to which mail addressed to *user*@flibble.orac.net.au should be sent to (solaris in this case).

The `/var/named/private.rev` file This file allows "reverse lookup." With this file, a system can get the name of a host from its IP address. A sample `private.rev` file is shown in figure 5. The PTR code allows an IP address. The 10.5.3. part of the address is derived from the entry in the named.conf file (see section 3.4.2).

The `/var/named/private.local` file This file allows reverse lookup for the localhost address. This file is not strictly necessary. A sample `private.local` file is shown in figure 6.

Starting the named Daemon Once you have the files for named set up, you can start the **/usr/local/sbin/-named** daemon. This daemon will read the `/usr/local/etc/named.conf` file for its configuration.

You will probably want to make the named daemon start up during startup. To do this, modify the `/etc/init.d/inetsvc` file so that the lines which read

```
;
;    Reverse address file for localhost
;
;name    ttl    class    type    data
;
;    Source of authority
@        IN    SOA    solaris.flibble.orac.net.au. root.solaris.flibble.orac.net.au. (
                2000050201    ; Serial
                10800    ; Refresh - 3 hours
                3600    ; Retry - 1 hour
                432000    ; Expire - 1 week
                86400)    ; Minimum - 1 day
        IN    NS    solaris.flibble.orac.net.au.
;
;    Machines names
;
;name    ttl    class    type    data
1        IN    PTR    localhost.
```

Figure 6: Sample `private.local` File for DNS

```
if [ -f /usr/sbin/in.named -a -f /etc/named.boot ]; then
    /usr/sbin/in.named;     echo "starting internet domain name server."
fi
```

now read

```
if [ -f /usr/local/sbin/named -a -f /usr/local/etc/named.conf ]; then
    /usr/local/sbin/named; echo "starting ISC internet domain name server."

elif [ -f /usr/sbin/in.named -a -f /etc/named.boot ]; then
    /usr/sbin/in.named;     echo "starting Solaris internet domain name server."
fi
```

If you have installed named in an automounted local directory, you may need to delay the starting of the daemon somewhat, until the automounter is running.

3.4.3 DNS Client Configuration

Once you have set up the DNS server, you will need to configure each system to use the name server.

Configuring Solaris Clients To configure a Solaris client properly you will need to edit the `/etc/nsswitch.conf` so that the DNS server is consulted. Modify the hosts line in so that it reads:

```
hosts:      files dns
```

This line means that the Solaris system will first look up a name in the `/etc/hosts` file (see section 3.3.1). If the name isn't there, then DNS will be used.

You will also have to set up the `/etc/resolv.conf` file so that the correct name servers are consulted. A sample resolver file is:

```
;
;    Resolver for domain flibble.orac.net.au.
;
domain        flibble.orac.net.au
nameserver    10.5.3.1
nameserver    203.30.77.33
```

The entries in this file are pretty self-evident. The name servers are tried in order, so if the local name server is down, the ORAC name server will be tried instead.

The last thing you need to do is set the local domain name. You probably do not have to do this, but neatness demands it. The two commands you need are **`domainname flibble.orac.net.au`** and **`domainname > /etc/defaultdomain`** where flibble.orac.net.au is replaced by your chosen domain name.

Disabling nscd Cache Refreshes Solaris 2.6 comes with a name caching daemon, **nscd** This daemon keeps a cache of recent name queries to allow a more speedy response to common queries.

nscd is all very nice. But, in it's default configuration, it refreshes the queries in it's cache every hour or so. If you have a dial-out connection, this means that every hour, your line will come up to re-query any names that are in the cache; not a good thing.

To disable the refresh, alter `/etc/nscd.conf` so that the line reading `keep-hot-count hosts 20` now reads `keep-hot-count hosts 0` Then make the **nscd** daemon re-read it's configuration file by sending it a HUP signal.

Configuring Linux Clients Configuring Linux DNS clients is similar to configuring Solaris clients as described above, in section 3.4.3, except that you do not have to configure the `/etc/nsswitch.conf` file and you only need to configure the domain name if you have NIS installed.

Configuring Windows 95 Clients To configure DNS for Windows 95, you need to open the Network section of the Control Panel and choose the TCP/IP properties. For the example network, we have:

Tab	Field	Value
DNS Configuration	Enable DNS	
	Host	win95
	Domain	flibble.orac.net.au
	Add	10.5.3.1
	Add	203.30.77.33

3.4.4 Testing the DNS Configuration

Once you have the DNS configuration working, you will probably want to test it. The basic tool for testing the server configuration is **/usr/local/sbin/nslookup** This tool allows you to interrogate the name server and see how it responds to various questions.

Testing the Server The DNS daemon can be made to provide debugging information by sending it a USR1 signal, via kill. The daemon will start tracing its behaviour by writing records into the file `/var/named/named.run` Sending the daemon a USR2 signal turns off debugging.

The daemon can be made to dump it's current state into the file `/var/named/named_dump.db` by sending it a INT signal.

Testing the clients For Unix systems, using **nslookup** to test client behaviour is obvious. Windows 95 does not have a suitable testing package. I used **WS_Ping** to test my setup.[25] **WS_Ping** is not freeware.

4 DP

By now, you should have already downloaded the DP package, as described in section 3.2. You can unzip and untar this package by using the command **`zcat dp-`*`version`*`.tar.gz | tar xvf -`** (Make sure that you are using the GNU version of programnamezcat here.)

4.1 Compiling DP

Before doing anything, I advise you to read the README file that comes with the DP distribution. You can also browse the HTML documentation in the doc subdirectory.

You may also want to change the of the DPCONF_DIR and DP_DIR variables in the dp.conf file. For example, My system has them set to DPCONF_DIR=/usr/local/dp/conf and DP_DIR=/usr/-local/dp

4.1.1 Making and Installing DP

Make sure that you have /usr/ccs/bin before whereever you have GNU-make in your PATH variable.

The DP installation process modifies the root crontab entries. You may wish to save your original crontab file by using the command **crontab -l > /tmp/root.crontab.orig** so that you can recover from any accidents.

If you are using the GNU C compiler, alter Makefile.conf to use gcc instead of cc

Once you have fixed up the config and make files, you should be able to make the DP file as per the instructions in the README file. You will need to be root to perform the install actions. Once you have finished, you should be ready to start configuration.

4.2 Configuring DP

DP configuration is a fairly messy business. As root, you will need to do a great deal of mucking about with various parts of the system. Once you have finished configuring DP, the Solaris machine will start acting as a router, so there is a fair bit OS configuration as well. Have fun.

4.2.1 Configuring the Port

Using **admintool** use the Serial Port Manager to configure the port that you wish to use as a *Modem - Dial-Out Only* template. Set the baud rate to 38400, or higher if you can manage it. Setting up the modem as dial-out only stops the modem from answering if someone tries to hack in.

4.2.2 Configuring the /etc/hosts and /etc/networks Files

Since the Solaris machine is going to be a router, you will need to add entries to the /etc/hosts and /etc/networks files. You should add the following lines (modified suitably) to the /etc/hosts file:

```
#
#    ORAC Hosts
#
203.30.77.2    solaris-ppp # Gateway to ORAC (fake)
203.30.77.152  orac-ppp    # ORAC address
```

The first line gives the IP-address of the Solaris machine, as it appears to ORAC (and the rest of the world). This address is a fake address, which will be overwritten by the dynamic address assignment when you dial up. The second line gives the address of the ORAC terminal annex; what you will be dialling into.

You also need to modify the /etc/networks file, so that it knows about the ORAC network. The following line will do:

```
#
# ORAC network
#
orac        203.30.77
```

```
IF=dp0 SYS=orac
    MODEMS=ktx-a
    PHONE=XXXXXXXX
    LOGIN_SCRIPT=orac-login
    LOGIN_ARGS=LLLLLLL,PPPPPPPP,ppp
    LOGIN=LLLLLLL
    TRACE=orac
    ACCESS=orac
    PPP_ARGS=dynamic,:203.30.77.152
    PRE_AUX=aux_script

MODEM=ktx-a DEV=cua/a BAUD=38400 DIAL_SCRIPT=ktx-dial DIAL_CHARMAP==W-,
```

Figure 7: Sample DP Condiguration File for an Annex

```
IF=dp0 SYS=orac
    MODEMS=maestro-a
    PHONE=XXXXXXXX
    LOGIN_SCRIPT=direct
    TRACE=orac
    ACCESS=orac
    PPP_ARGS=user,LLLLLLL,dynamic,:203.30.77.152
    PRE_AUX=aux_script

MODEM=maestro-a DEV=cua/a BAUD=38400 DIAL_SCRIPT=maestro-dial DIAL_CHARMAP==W-,
```

Figure 8: Sample DP Condiguration File for a Direct (PAP) Connection

4.2.3 Adding a Default Router

On start-up, your system should be set up so that ORAC is the default route for non-local traffic. The way to do this is to create a file called `/etc/defaultrouter` which contains the following line:

```
orac-ppp
```

After rebooting, network traffic will be routed to whatever interface provides a path to orac-ppp

4.2.4 Setting Up the DP Interface File

The file `/etc/dp/conf` (or wherever you have configured the DP files to be) contains the interface definitions for your PPP connections. ORAC now has two lines. The old line allows connections on a 28.8k modem which uses an Annex and a login script. The new line allows connections up to 56k (or will do) and uses the PPP password authentication protocol (PAP).[13]

A sample `/etc/dp/conf` file for the old line is shown in figure 7. The first entry is the interface definition, giving the modem to use, the phone number to call, etc. You will need to replace *XXXXXXXX* with the phone number you intend to call, *LLLLLLL* with your login id and *PPPPPPPP* with your password. It is a good idea to ensure that the configuration file is only readable by root, since your password is in plain text. The second entry gives the modem description with the port to use and the dial script to use. More on these below.

A sample `/etc/dp/conf` file for the new line is shown in figure 8. You will need to replace *XXXXXXXX* with your phone number and *LLLLLLL* with your login id, as above. The modem line gives the definition for a new, faster modem (a Maestro Executive). This new modem can go to speeds of up to 56k. However, Solaris and DP do not seem to support speeds over 38k, so until I figure out how to configure things properly, I'm limited to 33.6k.

Your login id may, in fact, be your full email address as opposed to a straight user name. (eg. *LLLL-LLL*@orac.net.au instead of *LLLLLLL*). If this is so, use the full email address in the `PPP_ARGS` section and in the `upap` file (see section 4.2.8).

Don't try and set up a faster configuration than that supported by the OS. DP has a nasty habit of translating high speeds into really low speeds (56k maps on to 50 baud) if the OS does not support higher speeds. Adding extra lines to the `/etc/ttydefs` file will not help, as DP does the port programming itself.

4.2.5 Setting Up the DP `hostnames` File

The `/etc/dp/hostnames` file contains the addresses between which DP will communicate. A sample `/etc/dp/hostnames` file is:

```
dp0    solaris-ppp    orac-ppp
```

This file means that the interface dp0 is used to communicate between solaris-ppp (ourselves) and orac-ppp (themselves).

4.2.6 Setting Up the Modem Script

The `ktx-dial` entry in the `MODEM` part of the configuration file is a script telling DP how to dial out for a connection. The script I use, for my KTX FAX/modem, is `/opt/dp/script/ktx-dial` and contains the script in figure 9. Essentially, this script configures the modem correctly and then dials the modem. Eventually, it sees whether it gets a suitable response and then continues. Since I live in Australia, the section containing `\xAT E1 X3 B0 &G0 &P1 S0=0 S6=2 S11=95\r` configures the modem according to Austel requirements. If you live somewhere else, you will probably have to configure the modem differently.

The new modem I have is a Maestro Executive. This modem is capable of 56k speed, but is designed for use in country areas and sometimes defaults to an odd line speed to cope with the connection quality. The `/opt/dp/script/maestro-dial` script that I use for the maestro is shown in figure 10. There is a slight change to the initialisation string. The part containing `+MS=11,0,,33600,1` restricts this modem to 33.6k operation.

4.2.7 Setting Up the ORAC Login Script

You can ignore this section if you are using the new connections and PAP; use section 4.2.8 instead. Once you have successfully dialled out, another script is run which allows the ORAC login process to be negotiated. This script is named in the `LOGIN_SCRIPT` argument and, in this case is called `orac-login` which is kept in the file `/opt/dp/script/orac-login` A sample file is shown in figure 11. This file follows the usual Username:/Password:/Annex: prompt sequence, supplying the arguments given by the `LOGIN_ARGS` parameter. On success, you will be logged in to your ISP and ready to go.

4.2.8 Setting Up the PAP Password File

You can ignore this section if you are using a login script (section 4.2.7). To allow PAP to be used, you need to set up a file, `/etc/dp/config/upap` which contains the PAP username and password. *NB. This file* must *be in the `/etc/dp/config` directory. The normal `/etc/dp.conf` settings do not work here.*[2] This file contains passwords in plain text, and should only be readable by root (mode 600).

A sample `/etc/dp/config/upap` file is:

```
# client server secret [IP address ...]
LLLLLLL * PPPPPPP
```

Where *LLLLLLL* is the login id for your ISP and *PPPPPPP* is the password.

[2] Slack, or what?

```
##
##  Dial up a V.32bis or V.32 modem
##
xmit "\xAT Z\r"
{
    alternate
    recv "OK" 5
    alternate
    xmit "\xAT Z\r"
    recv "OK" 5
    alternate
    log "Failed to get attention of modem"
    abort
}
#
#    Configured to Austel requirements
#
xmit "\xAT E1 X3 B0 &G0 &P1 S0=0 S6=2 S11=95\r"
{
    alternate
    recv "OK" 5
    alternate
    log "Failed to configure modem"
    abort
}
xmit "\rATDT$1\r"
mark
{
    alternate
    recv "CONNECT" "BUSY" "NO CARRIER" "NO DIALTONE" "ERROR" 55
    alternate
    log "Modem Timeout"
    abort
}
{
    alternate
    replay
    recv "CONNECT" 0
    go
    alternate
    replay
    recv "BUSY" 0
    log "Busy"
    busy
    alternate
    replay
    recv "NO CARRIER" 0
    log "No carrier"
    alternate
    replay
    alternate
    replay
    recv "NO DIALTONE" 0
    log "No dialtone"
    alternate
    replay
    recv "ERROR" 0
    log "Error received from modem"
}
abort
```

Figure 9: Sample Modem Script for a KTX 28.8k Modem

```
##
##    Dialup script for Maestro E56
##
xmit "\xAT Z\r"
{
    alternate
    recv "OK" 5
    alternate
    xmit "\xAT Z\r"
    recv "OK" 5
    alternate
    log "Failed to get attention of modem"
    abort
}
#
#    Configured to Austel requirements
#    Force a connection at 33.6k until I can get 56k working
#
xmit "\xAT &F B0 &G0 &P1 S0=0 S6=2 S11=95 S29=70 &D2 S95=45 T +MS=11,0,,33600,1\r"
{
    alternate
    recv "OK" 5
    alternate
    log "Failed to configure modem"
    abort
}
xmit "ATDT$1\r"
mark
{
    alternate
    recv "CONNECT" "BUSY" "NO CARRIER" "NO DIALTONE" "ERROR" 55
    alternate
    log "Modem Timeout"
    abort
}
{
    alternate
    replay
    recv "CONNECT" 0
    go
    alternate
    replay
    recv "BUSY" 0
    log "Busy"
    busy
    alternate
    replay
    recv "NO CARRIER" 0
    log "No carrier"
    alternate
    replay
    recv "NO DIALTONE" 0
    log "No dialtone"
    alternate
    replay
    recv "ERROR" 0
    log "Error received from modem"
}
abort
```

Figure 10: Sample Modem Script for a Maestro 56k Modem

```
##
##  Login to a Unix machine as username $1 with password $2 and service $3.
xmit "\x\r\r"
{
  alternate
    recv "name:" 10
  alternate
    log "Sending return to get first login prompt"
    xmit "\r"
    recv "name:" 5
  alternate
    log "Sending return to get first login prompt"
    xmit "\r"
    recv "name:" 5
}
xmit "$1\r"
{
  alternate
    recv "assword:" 35
  alternate
        xmit "\r"
    recv "name:" 10
    xmit "$1\r"
    recv "assword:" 10
}
xmit "$2\r"
{
    alternate
    recv "nnex:" 5
    alternate
    log "Timeout waiting for PPP to start on remote system"
    abort
}
xmit "$3\r"
go
```

Figure 11: Sample Login Script

```
DPaccess 2.0
#
# Options to allow only certain protocols, times of use, or specific machines.
# These limit the conditions in order to initiate a call.
#
timeouts {
    failedcall    3:10    # time to mark network as down after failed call
    callwait    0:50    # time to allow for call to go through
}

timeouts:dial {
    non_stream    10:00
    last_close    10:00
    inactivity  10:00
}

timeouts:answer {
    inactivity    10:00
}
```

Figure 12: Sample DP Access File

4.2.9 Setting Up the ORAC Access File

An access file, specified by the `ACCESS` parameter in the configuration file essentially gives details on various timeouts: when the line should be brought down, when you should retry after an engaged signal, etc. Usually, you want to bring the line down after some period of inactivity.

Since ORAC charges by data-transfer rate, rather than time and is only a local call away, I have configured my interface to wait 10 minutes before bringing the line down. The result is the file `/opt/dp/access/orac`, which is shown in figure 12.

4.2.10 Adjusting the Order of the Start-up File

The DP start script is initially placed in the `/etc/rc2.d` directory as `S99dpconfig` This position is somewhat inconvenient, as IP-Filter should be started after the PPP interface has been started. Renaming `S99dpconfig` to `S93dpconfig` is all that is needed here.

4.2.11 Adjusting the `/opt/dp/aux/aux_script` Script

The `/opt/dp/aux/aux_script` script is a shell script run by DP every time a connection is made. You can use this file to run any commands that may prove useful. Since you will probably be using SMTP and POP from some client, you can probably comment out the `${DPAUX_DIR}/ckmailq` line.

When you configure IP-Filter, you will need to add a line to this file. See section 5.2.2 for information on this line.

4.2.12 Restarting the system

Once you have set up DP to your satisfaction, you can reboot the Solaris system, so that DP starts automatically.

4.3 Testing the DP System

Once you have DP up and running, you should be able to connect up to your ISP via the Solaris system (only). To test the system, you can use the **telnet** and **ftp** programs to see if you can make connections to the outside world.

4.3.1 Routing Problems

If nothing seems to happen, you may have a routing problem. You can get the routing table by using the **netstat -rn** command, which should give you an output like:

```
Routing Table:
  Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
127.0.0.1            127.0.0.1             UH       01214730  lo0
203.30.77.152        203.30.77.2           UH       3      7  dp0
10.5.3.0             10.5.3.1              U        3    666  le0
224.0.0.0            10.5.3.1              U        3      0  le0
default              203.30.77.152         UG       0   1737
```

If you are missing an entry, then your routing table has probably not initialised correctly. Using **netstat -r** will cause the DNS name server to try to look up the names of the various IP addresses. Not very useful, since it also needs a route to the outside.

Another possible source of routing problems is *IP forwarding* being turned off for some reason. You can check this by running the command **ndd -get /dev/tcp ip_forwarding** If this command prints a 1, then the forwarding is turned on and everything is OK. If this command prints a 0, then the forwarding is turned off for some reason. You can turn forwarding on by using the command **ndd -set /dev/tcp ip_forwarding 1** You may wish to add this command to the `/etc/init.d/dpconfig` file to ensure that forwarding is always on at startup.

4.3.2 Tracing Calls

If DP appears to be dialling out, but then noting happens, DP provides a wealth of tracing information.

The file `/var/adm/dp/dp.log` is the log file for DP, you can examine it by using the command **tail -f /var/adm/dp/dp.log** This command will print out new log lines as they come in.

The file `/var/adm/dp/Trace/orac` (or whatever system name you have other than orac) provides a trace of the dial-up process, both modem initialisation and login to the ISP.

5 IP-Filter

By now, you should have already downloaded the IP-Filter package, as described in section 3. You can unzip and untar this package by using the command **zcat ip_fil*version*.tar.gz | tar xvf -** (Make sure that you are using the GNU version of **zcat** here.)

5.1 Compiling IP-Filter

Before doing anything, I advise you to read the `README` and `INSTALL-Sol2` files that come with the IP-Filter distribution.

Once you have fixed up the make file, you should be able to make the package as per the instructions in the `INSTALL.Sol2` file. Make sure that you have `/usr/ccs/bin` in your `PATH` variable before wherever GNU-make is located. To make and install the package, you will need to su to root. When you su to root, you will probably need to add `/usr/ccs/bin` and your compiler and make directory to the root `PATH` variable, so that the **strip** command is available.

5.1.1 A Note for Solaris 7 Users

If you are a user of Solaris 7 and are using **gcc 2.8.1** then you will encounter difficulties with the `stdarg.h` and `vararg.h` files. From the vantage point of Solaris 2.6, I can't provide a sensible solution. However, discussion on the IP-Filter mailing list mentions some work arounds: 1, 2, 3, 4, 5 and 6.

```
#
# Use the internal FTP proxy for outgoing FTP
#
map dp0 10.5.3.0/24 -> 0.0.0.0/32 proxy port ftp ftp/tcp
#
# Map anything going though dpn onto
# the dpn address
#
map dp0 10.5.3.0/24 -> 0.0.0.0/32 portmap tcp/udp 40000:60000
map dp0 10.5.3.0/24 -> 0.0.0.0/32
```

Figure 13: Sample IP-Filter NAT Configuration

5.2 Configuring IP-Filter

IP-Filter, if installed as a package, puts its binaries and man pages under `/opt/ipf` and the configuration files under `/etc/opt/ipf` Once you have installed IP-Filter, there is very little you need to do to set up NAT.

5.2.1 Configuring the `/etc/opt/ipf/ipnat.conf` File

To start using NAT, you will need to create a NAT configuration file, called `/etc/opt/ipf/ipnat.conf` A sample file is shown in figure 13. This file maps TCP and UDP messages coming from the 10.5.3. network (win95 and linux in the case of the example) onto a new port with the IP address of the dp0 interface. The IP address of the dp0 interface is assigned dynamically when the connection is made.

Note that connections from the solaris machine *do not* go through NAT. These connections are routed directly through the dp0 interface. You can make everything go through NAT by changing `10.5.3.0/24` to `0.0.0.0/0`

5.2.2 Configuring the `/etc/opt/ipf/ipf.conf` File

Unix systems tend to be very helpful when supplying services to the outside world. Naturally, these services are magnets to hackers and, now that you have a Unix system connected to the Internet, your private network is vulnerable.

Many ISPs (apparently including ORAC, bless them) already provide packet filtering that prevents some incoming connections to their dial-up systems. This may be all that you need. If this is the case with your ISP, then you can simply leave `/etc/opt/ipf/ipf.conf` empty.

If you have to do your own packet filtering and assuming that you simply want all your outgoing connections to work and any attempts at incoming connections to be blocked, you can set up a simple set of IP-Filter rules for just that purpose. These rules are shown in figure 14.

If you wish to set up a more sophisticated set of packet filters than this, I suggest that you examine the example rules found in `/opt/ipf/examples` You should also consult *Building Internet Firewalls*, which gives tables of packet filtering rules.[1]

Informing IP-Filter That the Dynamic Address has Changed If you are using filtering rules, then you will need to inform IP-Filter of the fact that DP (section 4) has changed your IP address every time it dials up. To do this, you will need to add the line `/sbin/ipf -y` to `/opt/dp/aux/aux_script` This script is run every time DP makes a connection and the command tells IP-Filter to rebuild its filter tables.

5.3 Restarting the System

Once you have set up IP-Filter to your satisfaction, you can reboot the Solaris system, so that IP-Filter module loads and NAT starts automatically. You will see IP-Filter attach itself to le0 and dp0 towards the end of the startup script.

```
#
#
# -------------------------------------------------------------
# Nasty Packets:
#  Block any packets which are too short to be real.
block in log quick all with short
# Block any packets with source routing set
block in log quick all with opt lsrr
block in log quick all with opt ssrr
#
# -------------------------------------------------------------
# Private Network:
# Allow traffic on le0 and lo0 to pass unimpeded
pass in on le0 all
pass out on le0 all
pass in on lo0 all
pass out on lo0 all
#
# -------------------------------------------------------------
# Public Network (dp0):
# That which is not explicitly allowed is forbidden
block in log on dp0 all
block out log on dp0 all
#
# Invalid Internet packets
# Deny reserved addresses
block in log quick on dp0 from 10.0.0.0/8 to any
block in log quick on dp0 from 192.168.0.0/16 to any
block in log quick on dp0 from 172.16.0.0/12 to any
#
# ICMP Protocols
# Allow pings out
pass out log on dp0 proto icmp all keep state
#
# TCP/UDP Protocols
# Allow TCP/UDP requests to go out and keep the results
# flowing back in.
pass out log on dp0 proto tcp/udp from any to any keep state
# Allow FTP data channel back in
pass in quick on dp0 proto tcp from any to any port = ftp-data keep state
pass in quick on dp0 proto tcp from any port = ftp-data to any port > 1023 keep state
#
# Reset/Error for TCP/UDP services, send back TCP-Reset or
# Network unreachable to attempts to initiate connections.
block return-rst in log on dp0 proto tcp from any to any flags S/SA
block return-icmp(net-unr) in log on dp0 proto udp from any to any
```

Figure 14: Sample IP-Filter Packer Filtering Rules

5.4 Testing IP-Filter

Once you have both DP and IP-Filter up and running, you should be able to connect up to your ISP via the any system. You can test the system by trying to make connections to the outside world via your private network systems — win95 and linux for the example.

Manual pages for the IP-Filter programs can be found in `/opt/ipf/man` If you are doing a lot of testing, then you will probably want to add this to your `MANPATH` environment variable.

You can examine the state of NAT by using **`/sbin/ipnat -l`** This command will print out the current set of NAT rules and the current set of redirections.

You can examine the filtering process by using **`/opt/ipf/bin/ipmon -t`** This command prints out a continuous trace of the logged packets passing through the filter. You can examine the filtering statistics by using **`/sbin/ipfstat`**

References

[1] D. Brent Chapman and Elizabeth D. Zwicky. *Building Internet Firewalls*. O'Reilly, 1995.

[2] Bryan Costales and Eric Allman. *Sendmail*. O'Reilly, second edition, 1997.

[3] D. Croker, N. Freed, J. Klensin, M. Rose, and E. Stefferud. *SMTP Service Extensions*, number 1869 in RFC.
ftp://ftp.isi.edu/in-notes/rfc1869.txt.

[4] G. DeGroot, D. Karrenberg, E. Lear, R. Moskowitz, and Y. Rekhter. *Address Allocation for Private Internets*, number 1918 in RFC.
ftp://ftp.isi.edu/in-notes/rfc1918.txt.

[5] *DP — Dialup PPP*.
http://www.acn.purdue.edu/dp/dp.htm.

[6] K. Egevang and P. Francis. *The IP Network Address Translator (Nat)*, number 1631 in RFC.
ftp://ftp.isi.edu/in-notes/rfc1631.txt.

[7] N. Freed, J. Klensin, and K. Moore. *SMTP Service Extension for Message Size Declaration*, number 1870 in RFC.
ftp://ftp.isi.edu/in-notes/rfc1870.txt.

[8] *Free PPP*.
ftp://cs.anu.edu.au/pub/software/ppp.

[9] *The Free Software Foundation*.
http://www.fsf.org.

[10] *Free Software Foundation — FTP Sites*.
http://www.fsf.org/order/ftp.html.

[11] *IP-Filter — A TCP/IP Packet Filtering Package*.
http://cheops.anu.edu.au/~avalon.

[12] *Internet Software Consortium*.
http://www.isc.org.

[13] B. Lloyd and W. Simpson. *PPP Authentication Protocols*, number 1334 in RFC.
ftp://ftp.isi.edu/in-notes/rfc1334.txt.

[14] J. Myers and M. Rose. *Post Office Protocol — Version 3*, number 1939 in RFC.
ftp://ftp.isi.edu/in-notes/rfc1939.txt.

[15] R. Nelson. *Some Observations on Implementations of the Post Office Protocol (POP3)*, number 1957 in RFC.
ftp://ftp.isi.edu/in-notes/rfc1957.txt.

[16] *The NIC*.
http://www.internic.net.

[17] *ORAC Internet Service Provider*.
http://www.orac.net.au.

[18] Rachel Polanskis. *Solaris PPP/NAT Howto*.
http://photon.nepean.uws.edu.au/ppp/index.html.

[19] J. Postel. *Simple Mail Transfer Protocol*, number 821 in RFC.
ftp://ftp.isi.edu/in-notes/rfc821.txt.

[20] *Solaris Freeware*.
http://smc.vnet.net/solaris_2.6.html.

[21] *Simple Mail Transfer Protocol (SMTP)*, number 10 in STD.
http://www.garlic.com/~lynn/rfcdoc.htm#std53.

[22] *Post Office Protocol — Version 3 (POP3)*, number 53 in STD.
http://www.garlic.com/~lynn/rfcdoc.htm#std53.

[23] Lynn Wheeler. *IETF RFC Indexes*.
http://www.garlic.com/~lynn/rfcietff.htm.

[24] *Win95NetBug FAQ*. (apparently defunct).
http://www-leland.stanford.edu/~llurch/win95netbugs/faq.html.

[25] *WSPing*.
http://www.ipswitch.com/Products/WS_Ping/index.html.

A Mail Handling

For the most part, you should be able to handle mail using your ISP's POP servers and SMTP server. However, for a variety of reasons, I am now running several mailboxes on my system and wished to set up a hidden mail system. Below lies madness.

What I have done is configured my Solaris system so that it acts as the mail server for my internal network. At intervals, my local mail server checks the mail boxes at my ISP's mail server, downloads any new mail and then places it in appropriate mail boxes locally. From there, any local mail clients can pick mail up from my POP3 server. Similarly, any mail being sent is passed through my **sendmail** program, which handles any aliases, redirects any local mail and passes the rest on to my ISP's **sendmail**.

Go Away!

This section is for seriously unhinged people, or people with some need that can't be neatly handled. If you can, simply set your systems up so that they communicate with your ISP's mail server for POP and SMTP. You'll thank me for this advice. You really will.

You're still reading? OK, the reason for setting this up is that it gives you great control over how mail flows in and out of your system. In particular, it allows you to run alias lists without too much pain and catch mail going to yourself before it goes out into the big wide world. It also allows you to download mail from a single mailbox on your ISP and distribute it to several internal mailboxes or aliases. The reasons, and there are many, for not doing this are:

- You will have to download and set up several new packages.

- You will need to develop a good knowledge of **sendmail** configuration files.
- You will be tied into receiving mail a fixed times, rather than checking whenever you feel like it; this is not strictly true, but it is for all intents and purposes.
- You will have several administrative headaches, particularly whenever mail is rejected and gets stuck in your mail queue.
- If you are planning to set all this up to use majordomo, you are going to have a really rotten time; I think it's possible, but I retired from the fray in confusion.

A.1 Theory

A.1.1 Outgoing Mail

The basic arrangement for outgoing mail is quite simple and shown in figure 15. **Sendmail** is designed to decide how mail should be handled, based on the mail address and forward it to the appropriate destination. This is exactly what we wish to do here. Mail destined for the outside world should simply be forwarded to **sendmail** on your ISP's mail server. Mail destined for a local mailbox should be diverted into that mailbox.

If that were all there is to it, it should be possible to use the **sendmail** configuration that comes with Solaris out of the box. However, there are two, additional complications:

- Mail sent from this domain will have the local domain (flibble.orac.net.au or whatever) attached to the senders name. Since this is not a recognised mail address, any return mail will be sent to an undeliverable address. What you want to do instead is replace your domain name with the domain name of your ISP as it leaves your network.
- Any mail sent to you, but having your ISP's domain name should be diverted to the local mailbox; it is pretty pointless to send such mail into the outside world, only to have it come back again. However, any mail sent to another address at your ISP should still be sent to the ISP.

Sendmail is flexible enough to allow this sort of thing to be done. You just have to learn to navigate its weird configuration syntax.

A.1.2 Incoming Mail

Incoming mail appears in two stages, shown in figure 16. The first stage gets mail from your ISP's POP3 server and delivers it into your local mailboxes. This stage is shown by the blue arrows in figure 16. The second stage is when one of your users wants to get their mail. This mail is sent via your local POP3 server. The red arrows show this path.

A program called **fetchmail** gets mail for your system. **Fetchmail** logs into the POP3 server on your ISP and checks to see whether there is any mail available. If there is mail available, then **fetchmail** delivers the mail locally. **Fetchmail** uses **sendmail** to deliver the received messages, this allows you to handle any aliases or other forwarding neatly. **Fetchmail** can be configured to check multiple mailboxes on the ISP side. **Fetchmail** can also be configured to divide the mail delivered to a single ISP mailbox into multiple local mailboxes, based on the To: line.

Once the mail has been delivered into your local mailboxes, any POP3 clients you have locally need to connect to a local POP3 server. This server then gets the mail from the local mailbox and delivers it to the client.

A.1.3 More Information

As always, O'Reilly have a guide to sendmail.[2] This is a very thick book, especially when you consider the terseness of most O'Reilly books.

The RFCs which make up SMTP are RFC 821, RFC 1869 and RFC 1870.[19, 3, 7]. Together, they make up STD-10.[21]

The RFCs which make up POP3 are RFC 1939, with some additional information in RFC 1957.[14, 15] RFC 1939 forms STD-53.[22]

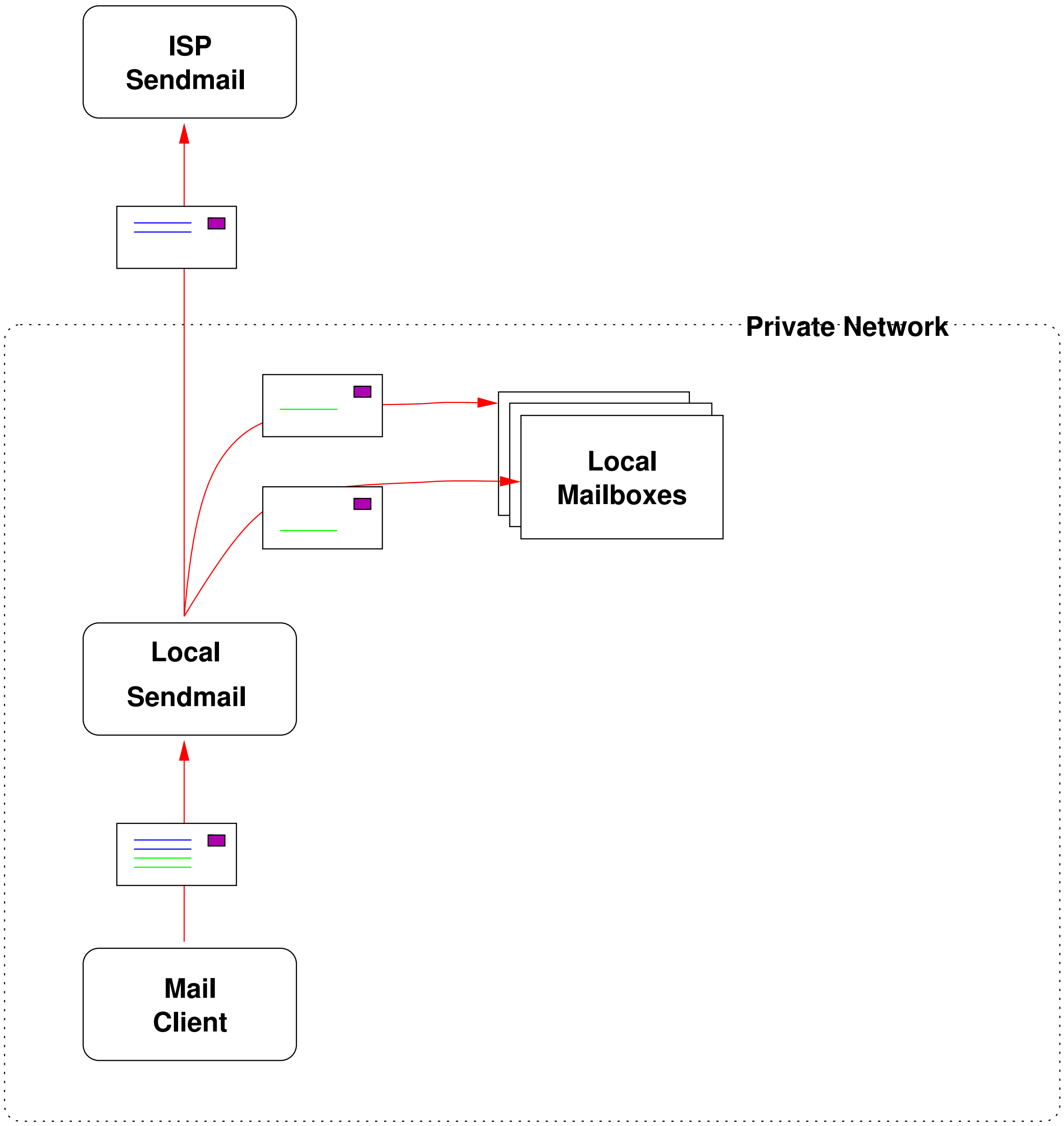

Figure 15: Mail Handling — Outgoing Mail

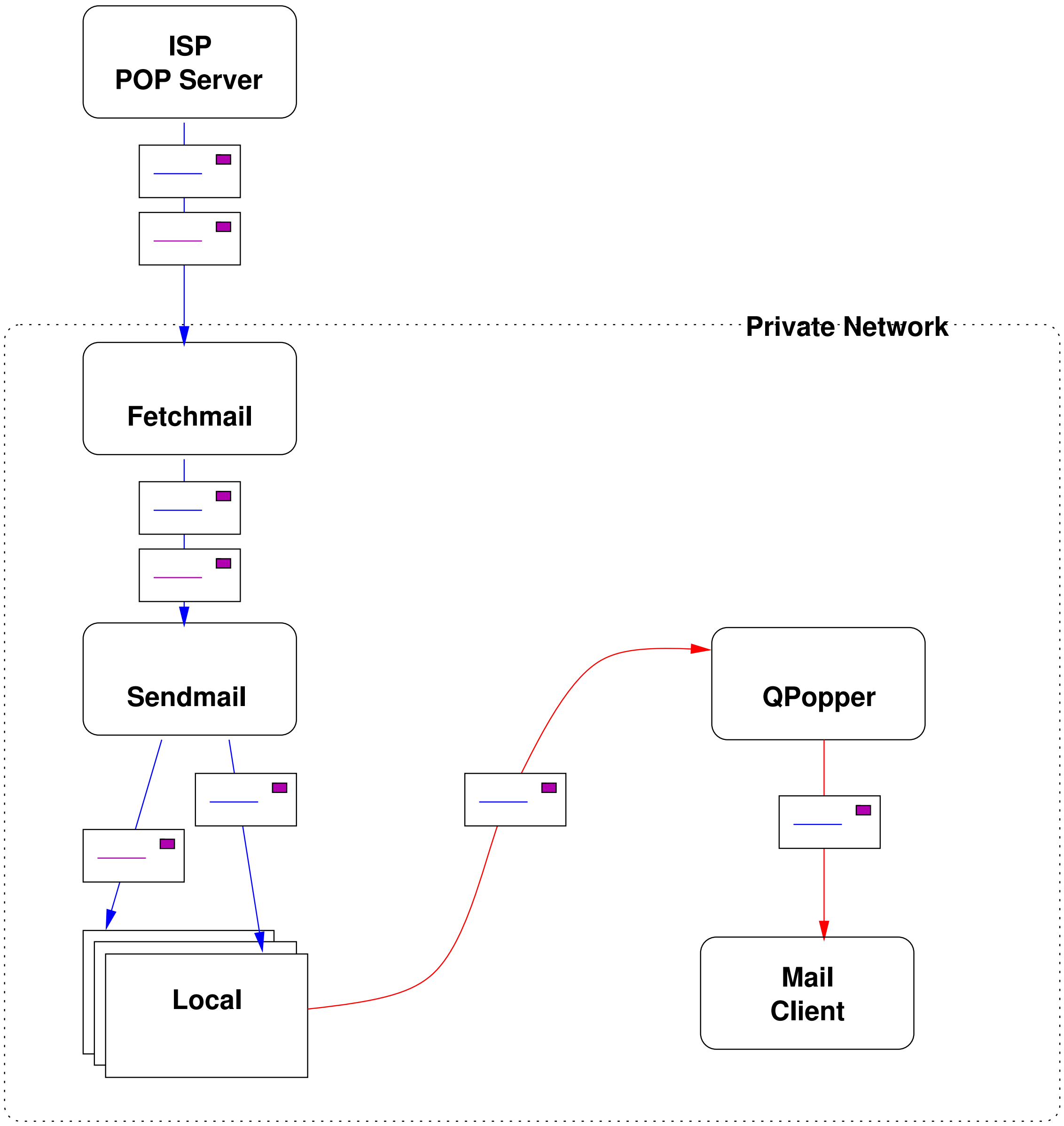

Figure 16: Mail Handling — Incoming Mail

A.2 Preparation

If you've got this far, I think I can assume that you have the various compilers, etc. mentioned in section 3. You do not need any additional tools.

You can get both QPopper and Fetchmail from the Solaris freeware page, either as pre-compiled packages or as source code.[20] I am assuming that you will be using the source code and compiling it, in which case you can get the packages at:[3]

Package	URL	Version
QPopper	`ftp://opcom.sun.ca/pub/freeware/SOURCES/qpop2.2.tar.gz`	2.2
Fetchmail	`ftp://opcom.sun.ca/pub/freeware/SOURCES/fetchmail-3.9.5.tar.gz`	3.9.5

A.3 QPopper

By now you have already downloaded the QPopper source code, as described in section A.2. You can unpack this code by using the command **`zcat qpop.`*`version`*`.tar.gz | tar xvf -`**

A.3.1 Making and Installing QPopper

Make sure that you are using the **/usr/ccs/bin/make** version of make. To make QPopper, cd into the qpopper directory and type **`make solaris2`** Once finished, you should have an executable called **popper.solaris2**

To install QPopper, you will need to decide where to put the resulting executable and man pages. For the sake of argument, assume that you are installing into `/usr/local` Copy the popper into the library directory, using **`cp popper.solaris2 /usr/local/lib/popper`** Copy the man page into the man directory, using **`cp popper.8 /usr/local/man/man8/popper.8`**

A.3.2 Configuring QPopper

To configure QPopper, you need to make it available as a service. Since the popper is only used occasionally, it makes sense to have it start up when someone wants to connect to it.

The POP3 Service Solaris does not come with the POP3 service listed in its services file. If you have not already done so, you should add the following line to the `/etc/services` file:

```
pop-3           110/tcp                         # Post Office
```

The Server Since the popper is a service that is started on demand, rather than running in background, the **inetd** daemon needs to be informed of its existence. Add the following line to the `/etc/inetd.conf` file:

```
#
#       QPopper POP3 Server
#
pop-3 stream tcp nowait root /usr/local/lib/popper popper -s
```

Once you have done this, you will need to make the **inetd** daemon re-read its configuration file. You do this, by sending a HUP signal to the **inetd** daemon.

A.3.3 Testing QPopper

The easiest test for QPopper is to connect up to it via a **telnet** session and see if you can get a response. Figure 17 shows a sample session.

[3] Versions are the version that I have. There may be more recent versions than this.

```
doug@solaris 12 % telnet localhost pop-3
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
+OK QPOP (version 2.2) at solaris starting.  <630.897723195@solaris>
USER doug
+OK Password required for doug.
PASS PPPPPPPP
+OK doug has 0 messages (0 octets).
QUIT
+OK Pop server at solaris signing off.
Connection closed by foreign host.
```

Figure 17: Testing QPopper

If you are unable to connect up to the `+OK` response, then the popper is not starting for some reason. Check the `/etc/inetd.conf` file and the `/etc/services` file, then see if killing and restarting the **inetd** daemon will improve anything.

Once you have connected successfully, send yourself some mail using the **/usr/bin/mail** program and see if you can get your mail from the machine, using Netscape or some other POP client.

A.4 Fetchmail

By now you have already downloaded the fetchmail source code, as described in section A.2. You can unpack this code by using the command **`zcat fetchmail.`*`version`*`.tar.gz | tar xvf -`**

A.4.1 Making and Installing Fetchmail

Make sure that you are using the **/usr/ccs/bin/make** version of make. Fetchmail comes with a configuration script. To set up the makefile, use **`./configure`** Once you have configured fetchmail, you will need to modify the Makefile to include the `resolv` library. Modify the `LOADLIBS` line, so that it looks like: `LOADLIBS = -lresolv -lsocket -lnsl -lfl`

If you want to install fetchmail into a directory other than `/usr/local/{bin,lib,man}` then you will need to modify the `prefix` and `bindir` etc. entries in the Makefile.

To make fetchmail, just use **`make`** to make the program and **`make install`** to install it.

A.4.2 Configuring Fetchmail

Fetchmail can be run by anyone. Each user can have a separate `.fetchmailrc` file in their home directory, which they can use to get mail on demand.

For the purposes of this document, I am going to assume that fetchmail will be run by a central user — root — which gets mail from whatever mailboxes are needed and forwards the results to various users. This approach rather assumes that you are running a small network where there is a single user (you) in various guises, or a few users who trust each other.

The `.fetcmailrc` File The root user needs to have a `/.fetchmailrc` file configured so that fetchmail knows where to get mail and what POP3 accounts to get it from. A sample .fetchmailrc file is shown in figure 18. This file tells fetchmail to contact the ISP's mail server and check for mail first for a user called doug and then for a user called alison. Mail sent to the user doug@orac.net.au is delivered to the local user douglas. Mail sent to the user alison@orac.net.au is delivered to the local user called alison.

Note that the `/.fetchmailrc` file has passwords in plain text in it. This file should be set to mode 600 to avoid snooping eyes.

```
#
#       Sample .fetcmailrc file
#
poll pop.nat.orac.net.au
        with protocol pop3
        username doug password PPPPPPPP is douglas
        username alison password XXXXXXXX is alison
;
```

Figure 18: Sample `.fetchmailrc` File

It is possible to configure fetchmail to get messages from a single ISP mailbox and distribute the results to several local mailboxes, a process known as *multidrop*. The fetchmail man pages contain a description of how to do this.

Crontab Entries Fetchmail essentially polls your ISP's POP3 server to see if there is anything worth downloading. In theory, you can run fetchmail as a daemon and have it regularly poll for incoming mail. However, with the sort of dial on demand PPP interface we have here, fetchmail will always be bringing the link up to see if there is anything there. Fetchmail can stop itself from polling on inactive links, but only on Linux systems.

The simplest solution is to simply set up **cron** entries to check for mail at regular points. Sample **crontab** entries are:

```
0 8,10,12,14,18 * * 1-5 /usr/local/bin/fetchmail -s
0 0,5,17 * * * /usr/local/bin/fetchmail -s
```

These entries start fetchmail in silent mode at 12am,5am,8am,10am,12pm,2pm,5pm and 6pm on weekdays and 12am,5am and 5pm on weekends.

A.4.3 Testing Fetchmail

Fetchmail is fairly easy to test. Once you have set up your `.fetchmailrc` file, using **`/usr/local-/bin/fetchmail -v`** will give you a verbose trace of its operations.

A.5 Sendmail

Sendmail comes as part of the Solaris package. However, to cope with our special circumstances, a certain amount of configuration needs to be done.

A.6 Configuring Sendmail

To configure sendmail, you need to edit the `/etc/mail/sendmail.cf` file. Sun has thoughtfully provided two sample configuration files, `/etc/mail/main.cf` and `/etc/mail/subsidiary.cf` Our sendmail system is the main sendmail system for the network; it decides where to send everything. We simply need to make a few modifications to the `/etc/mail/main.cf` file so that it meets our requirements.

A.6.1 Configuring sendmail.cf

To modify `sendmail.cf` you first need to make a copy of `main.cf` and modify it. Change directory to `/etc/mail` and type **`cp main.cf sendmail.cf`** Afterwards, you'll need to make some changes to `sendmail.cf` summarised in figures 19 and 20. Rather than do all this work, you can get a complete copy of the modified file at `http://users.orac.net.au/~doug/network/sendmail.cf` You will need to further modify this file so that it has the correct domain names, etc.

Once you have changed `sendmail.cf` (and if you have sendmail running) you will need to inform sendmail of the fact that a change has occurred. To do this, you will need to send a HUP signal to the **sendmail** daemon.

The /etc/mail/localusers Local Users File The `sendmail.cf` file, described above, uses a file called `/etc/mail/localusers` to see who is a local user. Mail addressed to an address of the form *localuser*@your.isp.domain, eg. doug@orac.net.au, get redirected into local mailboxes, rather than sent out to the ISP. The `localusers` file simply contains a list of local names, one to a line. As a sample, my localusers file contains:

```
doug
alison
```

If you wanted to get more sophisticated, you could pipe the contents of your `/etc/passwd` file through a suitable awk script.

Whenever you change the `localusers` file, you will need to inform the **sendmail** daemon that a configuration change has occurred, using the HUP signal.

A Note on Aliases One of the major motivations for setting up your own mailing systems is the ability to use the aliases file. Note that **sendmail** checks the alias file for local aliases only. An alias with a domain name attached will be sent to the outside world.

A.6.2 Testing Sendmail

Testing **sendmail** is liable to be a long and tedious process and prone to errors. When you start testing, it is a good idea to have your modem turned off, so that your ISP is not bombarded by strange and undeliverable messages. Another good idea, until you are happy, is to rename your `sendmail.cf` file to something like `test.cf` and run **sendmail** using the **`-C test.cf`** option.

Testing Address Rewriting To test address rewriting, you will need to start **sendmail** in address testing mode and see how well it looks after the various addresses you give it. To start **sendmail** in address testing mode, use the command **`/usr/lib/sendmail -C test.cf -bt`** You will be presented by a prompt, where you can try out various addresses and see how they are rewritten. Since you have the modem turned off, you may have to wait until the DNS server times out before a prompt appears. Samples, with comments, are shown in figures 21, 22 and 23.

A.6.3 Testing Mailing

Once you are happy with the address rewriting, you can start sending mail about the place. Have a few friends on tap who will send you replies, or a mail redirection service so that you can send mail to yourself. Any mail that you send yourself should get diverted into the local mailbox. You can examine the *Received by:* trail in the returning mail to see whether your setup is working correctly.

Sendmail traffic is logged, by default, to the `/var/log/syslog` file. Having a window up displaying the output of **`tail -f /var/log/syslog`** is a good idea, while you are testing.

B To Do

This appendix contains comments on any thing that I haven't worked out how to deal with yet. Comments are welcome.

Windows 95 Socket Connections Timing Out This is the major thing that I have yet to figure out. It takes approximately 30 seconds for the dial-out to complete. During this time, it appears that Windows 95 decides that the connection has failed and reports a time-out. Afterwards, of course, everything is fine.

```
21c21
< Lmmaildomain
---
> #Lmmaildomain
```

We have no sendmailvars table.

```
26c26
< Dj$m
---
> #Dj$m
28c28
< #Dj$w.$m
---
> Dj$w.$m
```

We want this machine to appear inside the flibble.orac.net.au domain, as solaris.flibble.orac.net.au or whatever you have named your machine and domain.

```
34c34
< DMsmartuucp
---
> DMether
```

By default, we pass things in a chunk across the network.

```
37,38c37,39
< DR ddn-gateway
< CR ddn-gateway
---
> # Forward any mail I don't understand on to my ISP
> DR smtp.nat.orac.net.au
> CR smtp.nat.orac.net.au
```

Where to send mail that is outside my understanding. Any mail which is not local gets sent on to the ISP's mail server, and then on to the outside world.

```
73,75c74,90
< # Example:
< # DmCS.Podunk.EDU
< # Cm cs cs.Podunk.EDU
---
>
> #
> # Everything looks like it is coming from my ISP, although
> # I know better.
> Dmorac.net.au
> Cm orac.net.au
```

Although my domain name is flibble.orac.net.au I want any mail sent to appear to come from my ISP. That way, any returned mail will end up in the right place.

Figure 19: Changes to the `sendmail.cf` File — Part 1

```
>
> # My local domain is really the DNS domain that I am actually
> # running in.
> # Domains which are really local domains
> DLflibble.orac.net.au
> CL flibble.orac.net.au
```

This is a list of the domains that are truly local. Any mail addressed to this domain is delivered via the local mailer.

```
>
> # I don't want to send mail to myself outwards
> # Users which are really local users
> FU /etc/mail/localusers
>
```

Any user who is listed in the file `/etc/mail/localusers` gets sent locally, even if it is addressed to them at the ISP. This way, local mail is not sent out just so that it can be sent back again.

```
236c251,252
< R$*<@$*$=m>$* $1<@$2LOCAL>$4 convert local domain
---
> R$*$=U<@$=m>$* $1$2<@$L>$4 localise true local users
> R$*<@$*$=L>$* $1<@$2LOCAL>$4 convert true local domain
```

Any users who are listed in ¡STRONG¿/etc/mail/localusers¡/STRONG¿ are redirected locally. Mail sent to the local domain name is also sent locally.

```
271c287
< R$+ $@$1<@$k>tack on my mbox host name
---
> R$+ $@$1<@$m>tack on my mbox host name
275c291
< R$+ $@$1<@$k>tack on my mbox host name
---
> R$+ $@$1<@$m>tack on my mbox host name
```

Mail sent from here appears to originate at my ISP, rather than at this machine.

Figure 20: Changes to the `sendmail.cf` File – Part 2

Names by themselves are delivered locally.

```
> 30 doug
rewrite: ruleset 30   input: doug
rewrite: ruleset  3   input: doug
rewrite: ruleset  3 returns: doug
rewrite: ruleset  0   input: doug
rewrite: ruleset  9   input: doug
rewrite: ruleset  9 returns: doug
rewrite: ruleset  0 returns: $# local $: doug
rewrite: ruleset 30 returns: $# local $: doug
```

Addresses which are part of my ISP's domain, but are listed in the `/etc/mail/localusers` table are rewritten by ruleset 6 to be delivered locally.

```
> 30 doug@orac.net.au
rewrite: ruleset 30   input: doug @ orac . net . au
rewrite: ruleset  3   input: doug @ orac . net . au
rewrite: ruleset  6   input: doug < @ orac . net . au >
rewrite: ruleset  6 returns: doug < @ LOCAL >
rewrite: ruleset  3 returns: doug < @ LOCAL >
rewrite: ruleset  0   input: doug < @ LOCAL >
rewrite: ruleset 30   input: doug
rewrite: ruleset  3   input: doug
rewrite: ruleset  3 returns: doug
rewrite: ruleset  0   input: doug
rewrite: ruleset  9   input: doug
rewrite: ruleset  9 returns: doug
rewrite: ruleset  0 returns: $# local $: doug
rewrite: ruleset 30 returns: $# local $: doug
rewrite: ruleset  0 returns: $# local $: doug
rewrite: ruleset 30 returns: $# local $: doug
```

Anything directed to my local domain is delivered locally.

```
> 30 doug@flibble.orac.net.au
rewrite: ruleset 30   input: doug @ flibble . orac . net . au
rewrite: ruleset  3   input: doug @ flibble . orac . net . au
rewrite: ruleset  6   input: doug < @ flibble . orac . net . au >
rewrite: ruleset  6 returns: doug < @ LOCAL >
rewrite: ruleset  3 returns: doug < @ LOCAL >
rewrite: ruleset  0   input: doug < @ LOCAL >
rewrite: ruleset 30   input: doug
rewrite: ruleset  3   input: doug
rewrite: ruleset  3 returns: doug
rewrite: ruleset  0   input: doug
rewrite: ruleset  9   input: doug
rewrite: ruleset  9 returns: doug
rewrite: ruleset  0 returns: $# local $: doug
rewrite: ruleset 30 returns: $# local $: doug
rewrite: ruleset  0 returns: $# local $: doug
rewrite: ruleset 30 returns: $# local $: doug
```

Anybody in my ISP's domain not listed in the `/etc/mail/localusers` table gets passed on to the ISP's mail system.

```
> 30 someone@orac.net.au
rewrite: ruleset 30   input: someone @ orac . net . au
rewrite: ruleset  3   input: someone @ orac . net . au
rewrite: ruleset  6   input: someone < @ orac . net . au >
rewrite: ruleset  6 returns: someone < @ orac . net . au >
rewrite: ruleset  3 returns: someone < @ orac . net . au >
rewrite: ruleset  0   input: someone < @ orac . net . au >
rewrite: ruleset  9   input: someone < @ orac . net . au >
rewrite: ruleset  9 returns: someone < @ orac . net . au >
rewrite: ruleset  0 returns: $# ether
$@ mail . orac . net . au $: someone < @ orac . net . au >
rewrite: ruleset 30 returns: $# ether
$@ mail . orac . net . au $: someone < @ orac . net . au >
```

But the local domain is still delivered locally.

```
> 30 someone@flibble.orac.net.au
rewrite: ruleset 30   input: someone @ flibble . orac . net . au
rewrite: ruleset  3   input: someone @ flibble . orac . net . au
rewrite: ruleset  6   input: someone < @ flibble . orac . net . au >
rewrite: ruleset  6 returns: someone < @ LOCAL >
rewrite: ruleset  3 returns: someone < @ LOCAL >
rewrite: ruleset  0   input: someone < @ LOCAL >
rewrite: ruleset 30   input: someone
rewrite: ruleset  3   input: someone
rewrite: ruleset  3 returns: someone
rewrite: ruleset  0   input: someone
rewrite: ruleset  9   input: someone
rewrite: ruleset  9 returns: someone
rewrite: ruleset  0 returns: $# local $: someone
rewrite: ruleset 30 returns: $# local $: someone
rewrite: ruleset  0 returns: $# local $: someone
rewrite: ruleset 30 returns: $# local $: someone
```

Figure 21: **sendmail** Address Re-Writing — Part 1

```
> 30 a.n.other@awm.gov.au
rewrite: ruleset 30   input: a . n . other @ awm . gov . au
rewrite: ruleset  3   input: a . n . other @ awm . gov . au
rewrite: ruleset  6   input: a . n . other < @ awm . gov . au >
rewrite: ruleset  6 returns: a . n . other < @ awm . gov . au >
rewrite: ruleset  3 returns: a . n . other < @ awm . gov . au >
rewrite: ruleset  0   input: a . n . other < @ awm . gov . au >
rewrite: ruleset  9   input: a . n . other < @ awm . gov . au >
rewrite: ruleset  9 returns: a . n . other < @ awm . gov . au >
rewrite: ruleset  0 returns: $# ether
$@ smtp . nat . orac . net . au $: a . n . other < @ awm . gov . au >
rewrite: ruleset 30 returns: $# ether
$@ smtp . nat . orac . net . au $: a . n . other < @ awm . gov . au >
```

Anybody else gets passed on to the ISP's mail system, to be forwarded from there. We do not deliver directly, as that means that each individual domain in the address list needs to be delivered separately. This could mean quite a bit of traffic; let the ISP do the work.

Figure 22: **sendmail** Address Re-Writing — Part 2

```
> 11 doug
rewrite: ruleset 11   input: doug
rewrite: ruleset 11 returns: doug < @ orac . net . au >
> 21 doug
rewrite: ruleset 21   input: doug
rewrite: ruleset 21 returns: doug < @ orac . net . au >
```

Ruleset 11 is used to pre-process the sender when mail is being delivered by the *ether* agent. Similarly, ruleset 21 pre-processes the receiver.

If we haven't got a domain name, then add our ISP's name.

```
> 11 doug<@orac.net.au>
rewrite: ruleset 11   input: doug < @ orac . net . au >
rewrite: ruleset 11 returns: doug < @ orac . net . au >
> 11 a.n.other<@awm.gov.au>
rewrite: ruleset 11   input: a . n . other < @ awm . gov . au >
rewrite: ruleset 11 returns: a . n . other < @ awm . gov . au >
> 21 doug<@orac.net.au>
rewrite: ruleset 21   input: doug < @ orac . net . au >
rewrite: ruleset 21 returns: doug < @ orac . net . au >
> 21 a.n.other<@awm.gov.au>
rewrite: ruleset 21   input: a . n . other < @ awm . gov . au >
rewrite: ruleset 21 returns: a . n . other < @ awm . gov . au >
```

Addresses with domain names attached get left alone.

Figure 23: **sendmail** Address Re-Writing — Part 3

The win95netbug FAQ suggests that there are registry parameters which can be used to control the behaviour of Win95's socket parameters.[24] I guess that I just haven't found the right combination yet.

Actually, this has stopped happening. My PC had a disk crash and reinstallation made the problem go away. Other opinion on this matter is that it is still a problem, however.

Working Out What the Numbers Mean The IP-Filter monitoring and testing programs produce all kinds of interesting numbers. I wonder what they mean?

Speeds Greater Than 33.6k I am now using a sparc Ultra-1, which should handle 56k port speeds without difficulty. However, any attempt to talk to the modem at speeds above 38.4k causes the modem to not recognise commands.

Suspicion points towards the Ultra-1 UARTs not really being able to handle high speeds.