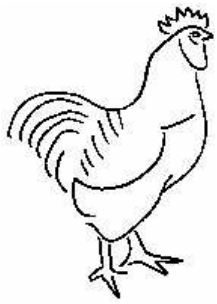
CLTC Documentation Sheet 2:

Setting up DHCP, DNS and NFS on the CLTC Server



Developed by The 'Free Range' Community-Linux Training Centre Project - Version 1.0, January 2003. http://www.fraw.org.uk/cltc/

"*Small islands not capable of protecting themselves are the proper objects for government to take under their care; but there is something absurd, in supposing a Continent to be perpetually governed by an island. In no instance hath nature made the satellite larger than its primary planet*"
Thomas Paine, *Common Sense*

This section deals with setting up some of the basic networking services: Dynamic Host Configuration Protocol (DHCP) to assign IP addresses to client machines; a Domain Name Server (DNS) to run the name-space on the local network; and a Networked File System (NFS) to allow the clients to access a share of the server's disk space in common.

Why use DHCP and DNS?

We don't have to use DNS or DHCP – so why bother? (NFS is needed, so we needn't worry about that).

All computers on a network must have a number. This can be arrived at in two ways: it can be preset by the system designer, or assigned by the server. For any network numbering can be a major problem. Numbering clashes can paralyse a network. Therefore the dynamic allocation of numbers by the server, using DHCP, has become the main method of numbering computers on large networks.

On the CLTC, DHCP has been used for convenience. It makes organising the network clients easier. But it also means that any other computers that are available can be easily connected to the network. We don't have to worry about numbering – DHCP takes care of the configuration.

Just because DHCP is installed doesn't mean that preset numbered machines can't be connected to the system. In fact, switching over the system from DHCP to a preset numbering system, and back again, is a very teaching tool that allows people to understand and configure both methods of numbering.

The main reason for using DNS is that it allows us to simulate the domain name system used on the Internet. Rather than having to remember and use the server's IP address – 192.168.66.1, or whatever it may be for a particular service – those using the network need only remember a simple name – *www.cltc.lan*, *irc.cltc.lan*, *smtp.cltc.lan*, etc.

But DNS itself can also lessen the workload on a network. If we hard-code the IP numbers then every time we have to re-number the network for some reason, we have to re-code the new numbers. By using DNS, all we have to do is change the numbering within the DNS system, and everything else falls into line.

With Linux system, even though we use DNS, it's still possible to hard-code numbers via the *hosts* file. As with DHCP, getting people to change the network naming system from dynamic to fixed and back again can be a useful means of teaching network development and maintenance.

So, we don't have to use DNS or DHCP on such a small network as the CLTC – but we do because of the teaching opportunities it creates.

Configuring DHCP

An example */etc/dhcpd.conf*

```
subnet 192.168.66.0 netmask 255.255.255.0 {
   range 192.168.66.100 192.168.66.199;
   option subnet-mask 255.255.255.0;
   option broadcast-address 192.168.66.255;
   option routers 192.168.66.1;
   option domain-name-servers 192.168.66.1;
   option domain-name "cltc.lan";
   option ip-forwarding on;
   option netbios-node-type 8;
}
```

Most network services are run by programs called 'daemons'. These are memory resident programs that operate automatically from the time your server boots until it is shut down. In this section of the CLTC documentation we deal with the configuration of the dhcpd (DHCP) daemon.

Note that most of this section will involve using text editors to manipulate configuration files, and command line instructions to tell the server to do things. Get used to this. You have to learn these skills to undertake server maintenance, especially if done remotely (e.g., down a telephone line to the cheap server farm on a different continent where you rent web space).

DHCP is very simple to configure. Firstly, you can check the current status of DHCP using the command:

```
/etc/rc.d/init.d/dhcpd status
```

This will tell you whether DHCP is running – and if you've just installed your server, it shouldn't be running.

Firstly, you must create a DHCP configuration file. There is no template for this on the system. Therefore we must create one from scratch and save it as `/etc/dhcpd.conf`. There's an example of the CLTC's *dhcpd.conf* file in the box above.

When you've saved the DHCP configuration file you then test this configuration by issuing the command:

```
dhcpd -f
```

This tries to run DHCP in the foreground (not as a daemon). If your configuration file contains errors allocation, nothing will happen. If you have a good configuration file you should get an error message stating something like '*Can't open lease database /var/lib/dhcp/dhcpd.leases*' (note, some later versions of the program, after *Red Hat 7.1*, don't give you this message as they seem to allocate the leases file themselves).

To solve the error message we to create the `dhcpd.leases` file. You do this by entering the command:

```
touch /var/lib/dhcp/dhcpd.leases
```

Now you must try the `dhcpd -f` command again. This time DHCP should successfully run. You get a message including the statements '*listening on*' and '*sending on*'. You now exit the foreground execution of `dhcpd` by pressing `Ctrl-C`.

Now issue the command:

```
/etc/rc.d/init.d/dhcpd restart
```

You should see something like '*shutting down dhcpd: [failed]*' followed by '*starting dhcpd: [OK]*'. The reason that you got a 'failed' was that it wasn't running in the first place. But the 'OK' means that the *dhcpd* daemon is now running successfully in the background.

To test DHCP you have to plug one of the client systems, or any computer that is configured with DHCP, into the network. With the server up and running, boot the client. If DHCP works the client will be allocated an IP address. You can check this by logging in as 'root' on the client, and then entering the command:

```
ifconfig eth0
```

If all's well, you'll see an IP address (within the range you set for DHCP), and the netmask, network and broadcast addresses set in *dhcpd.conf*, displayed as part of the client's network configuration. You can then test that the network connection is good by pinging the server from the client with the command:

```
ping 192.168.66.1
```

If the network is functioning properly you'll see

a list of times displayed for the time taken for each ping packet to travel across the network.

Note, as demonstrated here, that it's always a good idea to have an installed Linux client machine prior to setting up the server so that you can test that the server is functioning properly.

How DNS works

Now we move on to the domain name system, DNS. This is more complex, and requires a lot more typing of text files – which must be done precisely.

DNS is run by a daemon called named that is part of a package called BIND. Configuring DNS involves reviewing a number of files. The aim of the following information is not to tell you how to set up your own DNS system. There are many ways of implementing a DNS system, and there are plenty of books on how to do it. Instead, the text below outlines how DNS works within the CLTC network, and how the files are interpreted for that network.

First of all, using a text editor, open up the file */etc/host.conf*. This file specifies the order in which domain names will be 'resolved' by the system (this goes for the clients as well as the server. If the *hosts.conf* file states:

```
order hosts,bind
```

...that means that the static references, in the `/etc/hosts` file, will be used first when the computer tries to resolve a domain name. If this does not resolve the name/number, the the DNS system enabled by BIND is consulted second. If it were:

```
order bind,hosts
```

...the reverse would be the case.

If we were using static host names and IP addresses, we could configure everything using the `/etc/hosts` file. But as outlined earlier, changes to the numbering system would require that every computer on the network has its `/etc/hosts` file edited. And what about the Windows machines – they don't have this system at all.

Therefore we are using BIND to resolve names sent as requests to named across the network. The statement '*order hosts,bind*' is acceptable providing that the `/etc/hosts` file doesn't contain any entries that short circuit the name resolution provided by BIND.

BIND, and the named daemon, rely on the configuration file `/etc/named.conf`. *named.conf* looks like a complex file, but it has a very simple format. Let's look at the simplified extract contained in the box above.

Name resolution works in two directions. Forwards resolution turns domain names into IP numbers. Reverse resolution turns IP numbers into domain names.

An example *named.conf* file

```
# Standard named Options
options {
      directory "/var/named/";
};

# Default 'Hint' Zone
zone  "." {
      type hint;
      file  "named.ca";
};

# Localhost Zones
zone  "0.0.127.in-addr.arpa" {
      type master;
      file  "loopback-reverse.zone";
};
zone  "localhost" {
      type master;
      file  "loopback-forward.zone";
};

# CLTC Master Reverse Zone
zone  "66.168.192.in-addr.arpa" {
      type master;
      file  "cltc-reverse.zone";
};

# CLTC Master Forward Zone
zone  "cltc.lan" {
      type master;
      file  "cltc-forward.zone";
};

# Alphaweb Virtual Server Zone
zone  "alphaweb.lan" {
      type master;
      file  "alphaweb-forward.zone";
};
```

(note: the entries for the other virtual servers are omitted as they are nearly identical to 'alphaweb')

What */etc/named.conf* contains is just pointers to 'zone files' – small configuration files that contain the actual detailed instructions for resolving names to numbers, and vice versa. For a particular range of IP numbers (a single subnet) *named.conf* points to a particular file that provides the domain names for each address. Likewise, for a particular domain name, or a permutation of it, *named.conf* points to file that associates that name with a number.

For each entry the '*type master*' line indicates that this is the master zone file for network. Other types can be treated differently, for example providing a backup should there be problems with the master DNS file.

What *named.conf* says, taking the CLTC's revese zone as an example, is that any request to resolve a number to a name (reverse resolution) in the 192.168.66.X subnet (numbers 192.168.66.0 to 192.168.66.255) has the details contained in the file *cltc-reverse.zone*. BIND interprets the numbers in reverse – which is why it appears as 66.168.192. The *in-addr-arpa* part is shorthand for the 'subnet of IP numbers 0 to 255'. '*arpa*' refers to the first networking system developed for 'arpanet', the first version of the Internet develop for the military in the USA.

Any request for a name to IP resolution (forward resolution) relies on the zone file associated with that domain name. Taking the CLTC's forward zone as an example, any request that contains the domain *cltc.lan* has the details contained in the file *cltc-forward.zone*.

The zone files are kept in the location defined by the *options* statement contained at the beginning of the *named.conf* file. In this case, the directory `/var/named`.

If you look at the CLTC's *named.conf* file you'll see that there are a number of zone files used within the CLTC's DNS system:

- As standard, you have forward and reverse zones for the local host, and for a default 'hint' zone (called *named.ca*).
- There is a forward and reverse zone for the CLTC domain.
- Finally, each of the virtual web servers has its own forward zone file to resolve the server name to its IP address. This includes the virtual servers that are 'name based' and resolve to the same address.

If you wish to modify the DNS for the whole network, all you have to do is edit the zone files. To plug in a whole new service, with a new name and IP address, you would need to create a forward and reverse zone file, and then create a new entry in *named.conf*. This will then, after telling BIND to read the new files, make the new service instantly accessibly to the whole network.

How zone files work

Zone files are complex, and are very sensitive to simple mistakes made in typing (extra spaces, missed full stops, lack of brackets, etc.). Examples of reverse and forward zone files are shown in the box on the next page.

Both types of zone file begin with information on how they are to be used – this is called the *start of authority* (SOA) record. This indicates the edition (*serial*) of the zone file, the frequency with which the master server should be checked to see if new versions have been created (*refresh*/*retry*), and the length of time the record should used (*expire*/*minimum*).

The reverse zone file maps IP addresses to names within a particular subnet. Remember that addresses are handled in subnets, so each reference to a reverse zone can handle up to 250 or so IP addresses (certain addresses are used for network functions, so you can't use all 256).

The number in the CLTC reverse zone represents all the IP address within the 192.168.66.* subnet. Note that there are different types of record. The '*IN*' in the line refers to an Internet-style reference. The '*PTR*' refers to a domain name pointer. The '*NS*' refers to a name for the name server for the domain.

Within the CLTC reverse zone file, reverse references for the addresses 192.168.66.0 to 192.168.66.255 can be held. For example, '1', meaning 192.168.66.1, will map to *cltc.lan*;

An example reverse zone file

```
@       IN    SOA   @ root.localhost (
                    2 ; serial
                    28800 ; refresh
                    7200 ; retry
                    604800 ; expire
                    86400 ; minimum
                    )
@       IN    NS    dns.cltc.lan.
1       IN    PTR   cltc.lan.
10      IN    PTR   alphaweb.lan.
11      IN    PTR   betaweb.lan.
12      IN    PTR   gammaweb.lan.
13      IN    PTR   deltaweb.lan.
```

An example forward zone file

```
@       IN    SOA   @ root.localhost (
                    20020101 ; serial
                    28800 ; refresh
                    7200 ; retry
                    604800 ; expire
                    86400 ; minimum
                    )
@       IN    NS    dns.cltc.lan
@       IN    MX    10   smtp.cltc.lan
www     IN    A     192.168.66.1
smtp    IN    A     192.168.66.1
ftp     IN    A     192.168.66.1
telnet  IN    A     192.168.66.1
mail    IN    A     192.168.66.1
irc     IN    A     192.168.66.1
news    IN    A     192.168.66.1
dns     IN    A     192.168.66.1
server  IN    A     192.168.66.1
@       IN    A     192.168.66.1
```

but '10', meaning 192.168.66.10, maps to the 'alpha' virtual web server *www.alphaweb.lan*. If a particular number has no name mapped to it then the reverse resolution will return an error.

The forward zone file map a name to an IP address. It has a similar format to the reverse zone file. The '*A*' refers to an *address record*, an IP address to which the hostname relates. 'MX' refers to a 'mail exchange' address which can receive email for that domain. But instead of dealing with IP numbers in a subnet, the forward zone file deals with the different host names that prefix the domain name to which the zone file relates.

Now remember that the entry in the *named.conf* file that points to this file was *cltc.lan*. The *cltc.lan* name is represented in the zone file by the *@* symbol. Therefore each of these lines represent a host name, or 'virtual domain', that prefix *cltc.lan*. With the example given, all the names point to the CLTC server IP – 192.168.66.1. But this need not be the case. If we had more than one server on the network, for example separate mail and web servers, one name could point to one IP address and the other to another. It could even point to machines somewhere else on the Internet. In this way we can link resources together using host/domain names.

There are also two important lines. Firstly, like the reverse zone file, the *dns.cltc.lan* line has a '*NS*' record definition. This tells BIND that this particular host is a name server. Secondly, the *smtp.cltc.lan* line has an '*MX*' record definition This tells BIND that this host is a mail exchange server that can receive email addressed to this domain. You can have more than one mail exchange record, and the priority given to which is interpreted first is given by the number following the 'MX' identifier. The lower the number, the greater priority the mail server has. The purpose of this is to enable a single site to have more than one mail server. Then if one fails, then the next takes over to ensure mail is received. We don't need multiple redundant mail servers on the CLTC network, so this zone file only has one 'MX' record.

Configuring DNS

To implement DNS on a network you have to write a:

- reverse zone files for each IP subnet used;
- forward zone files for each domain name or virtual domain name;
- forward zone files for your local host machine's loopback address; and
- forward zone files for the default 'hint' zone.

This requires careful attention. There is a utility, called *bindconf*, that can generate these files automatically for you. We found that this did not give the flexibility we required. So in the end we just wrote the zone files using a

text editor.

Once you have prepared the *named.conf* and zone files, you can get the named daemon running. To test DNS you need to start named. You do this by issuing the command:

```
/etc/rc.d/init.d/named start
```

If all is well you should get an 'OK' message. Now you can test the DNS configuration by doing name/IP resolution requests with the *host* command. For example:

- `host www.cltc.lan` should return the IP address *192.168.66.1*.
- `host 192.168.66.1` should return the domain *cltc.lan*.

If you need to edit the configuration files for DNS, you must always remember to reload the configuration into the databases used by BIND/named. It is also important that if your zone file contains a serial number on the SOA record, you must increment that number. If you do not then BIND will ignore the new record because it will think it is the same as the old one. For this reason its easier to use a reverse-order date at the serial number – for example `200301291540`, meaning 15:40h on 29th January 2003.

After editing the zone files, you can reload the zone configuration using the command:

```
ndc reload
```

Configuring NFS

The Networked File System (NFS) works by 'exporting' a directory from the server to the client machines. Client machines can mount and use this directory just as they would any other disk media on the system. This allows resources to be shared between users of the network.

NFS is run by different daemons – `nfsd` and a few of the `rpc` services. You can check the status of NFS using the command:

```
/etc/init.d/nfs status
```

(note that NFS doesn't contain the control scripts in `/etc/rc.d/init.d`, but in the directory `/etc/init.d`)

This will tell you the status of the daemons – or if they're not working you get nothing.

The server directories to be exported are specified in the `/etc/exports` file. This contains one line (this is on one single line, but it's been slit here):

```
/home/export
      192.168.66.0/24(rw,insecure)
```

The *exports* file has two components to each line:

The first part is the path to the directory that you wish to export. We've set up a directory within the `/home` directory that will contain all the information exported from the server (although the convention is to put exported directories in an `/export` directory at the root level). This same directory will serve information for the Samba file shares.

The next part is the identity of a machine that is allowed to access the exported directory, and, within brackets and separated by commas (no spaces), some control information to regulate access.

The example above doesn't use a machine name as the identity, it uses an entire subnet. `192.168.66.0/24` is an address and bit mask combination that covers all IP addresses from 192.168.66.0 to 192.168.66.255. This means any machine on the CLTC's subnet can access NFS.

The access controls used are '`rw,insecure`'. '*rw*' allows both read and write access, and '*insecure*' blocks the requirement for authentication. There are many other control options that can be used – for example using '*ro*' makes access read-only.

Once the `/etc/exports` directory is prepared you can start and stop the NFS daemon:

```
/etc/init.d/nfs start
/etc/init.d/nfs stop
```

You can check the status of NFS using the command:

```
/etc/init.d/nfs status
```

If you ever modify the NFS configuration file you need to restart NFS for those changes to take effect. However, if you don't want to stop

NFS you can get NFS to re-read the contents of the *exports* file using the command:

```
exportfs -r
```

Running the services automatically at boot time

Finally, we need to automate the loading of the daemons at boot time. To load `dhcpd`, `named` and `nfsd` at automatically you have to enable them in the list of services to be loaded when Linux starts. You can do using the utility `ntsysv` (outlined at the end of sheet 1B).

After running `ntsysv`, select the options `dhcpd`, `named`, and `nfs`. The services will then start automatically at boot time.

The configuration of NFS, DNS and DHCP is now complete.