Linux Networking Guide: Part 4

DHCP

A simple guide to configuring Linux networks from the command line.

This final article in the series shows how to use DHCP to configure network hosts dynamically. BY BRUCE RICHARDSON.

The first three articles in this series covered all the basics of configuring Linux boxes as network hosts. The emphasis throughout has been on using command line tools and editing the text configuration files. If you have read all three articles you will know that network configuration on Linux boxes is actually very simple.

However, as your network grows it becomes an increasing chore just to keep all those configuration files up to date, particularly if you make significant changes to the structure of your network. Add a nameserver to your network, or change the IP address of a gateway router, and you will have the task to edit the configuration files on each and every network host.

Life would be much easier if it were only possible for your computers to fetch their configurations from a central source. Not only would it make the job of setting up new machines easier but you could make network design changes centrally and have them propagate to all your machines. Laptops, PDAs and other transient devices could be connected to your network and configure themselves automatically.

The good news is that it is possible, using the Dynamic Host Configuration Protocol. This article will show you how to set up DHCP both on the server and client sides and how you can update your DNS information dynamically when IP addresses are assigned through DHCP.

Overview

When a dynamically-configured host is first connected to a network it has no IP address nor any notion of the local subnet address, netmask etc. So it sends a broadcast request for configuration details. If there is a DHCP server on the local subnet it sends a reply, allocating the host an IP address and passing on any other network configuration parameters that it has been supplied with. The dynamically-configured host is now ready to participate normally on the network.

The server can pass on more than just the parameters of the host's network interface. It can store a wide range of network related information, including the local domain name, addresses of DNS servers, routers, WINS servers and much more. It is entirely up to the DHCP client software how much of this is then used to configure a host.

Leases

When a DHCP server assigns an IP address to a host it is not a permanent allocation. The server has available a pool of addresses to which it grants leases. A lease has a set lifespan and must be renewed before it expires if the host is then to retain the same IP address. Typically, DHCP client software will attempt to renew a lease once it is halfway through its lifespan and will repeat the attempt at regular intervals until it is either successful or the lease expires, after which time a new lease must be requested.

This leasing model allows you to have a pool of addresses smaller than the total number of hosts to be connected, if you know that only a certain fraction of those hosts are likely to be connected at any one time.

Finding a Network Card's MAC Address

Many NICs come with their MAC address printed on a label on the card. Another way to find the MAC address is to configure a network interface for it (by, say, getting it to take a dynamic address from DHCP) and then running ifconfig. In the details returned by ifconfig, the MAC address for each configured card is given in the HWaddr field.

It is also possible to associate, using DHCP, a fixed IP address with a particular hostname and/or network card (which latter option, since network cards tend not to flit from device to device, has the effect of associating the IP address with a specific piece of hardware). So if the DHCP client specifies a hostname in its request or if the request originates from a network card with a specific MAC address, then the associated fixed address may be returned.

Choice of Lease Lifespan

DHCP client software may specify a lease lifespan when requesting a lease but the server can have both a default lifespan setting for requests that don't specify and a maximum setting that overrides any request for a greater span. The value you assign to these settings will be significant in the effect on your network.

Firstly, setting a shorter lifespan will mean more frequent renewals and so more DHCP-related noise on the network (as well as making your network more vulnerable to a failure in your DHCP server).

Secondly, it is only when you renewing a lease that the client software checks for other network configuration details. So if you set a seven day lease and then give a new set of name server addresses to the DHCP server, it will then typically take at least three and a half days for that information to propagate throughout the network.

So sysadmins are typically faced with a tension between the optimum network performance and the propagation speed, which only time, experiment and experience can resolve.

The Science Bit

DHCP requests and replies are sent using UDP. The server listens on port 67, the client on port 68.

Some Drawbacks to DHCP

When using DHCP on your network, there are certain questions introduced about the reliability and security. On the reliability side, if your DHCP server fails then your entire network may just grind to a halt. MS Windows workstations are particularly bothersome in this situation and will assign themselves a new address on a reserved subnet, a feature called Automatic Private IP Addressing.

More seriously, the DHCP protocol makes no provisions for security. When a DHCP client sends a broadcast request it accepts the first reply it gets. A malicious person could then subvert hosts on your network by connecting a laptop running its own DHCP server. This isn't quite as calamitous as it sounds, since queries are only sent out by newly connected hosts or those which haven't been able to renew an existing lease. Your only protection is to run some kind of Intrusion Detection Software such as Snort.

One particular possible security hole arises with fixed IP address assignments (as described in the Section called Leases). If no MAC address has been associated with the assignment then the DHCP server has no way of verifying that the requesting host has any right to the hostname it specifies. So it is wise always to specify a MAC address where practical.

Because of these issues, it isn't wise to have your servers configure their network interfaces through DHCP. Leave that for your workstations and configure your servers manually. Otherwise, your entire network will be vulnerable to attack.

The DHCP Server

In this article I will be using the server software that is available from the Internet Software Consortium, which is overwhelmingly the most commonly used on Linux. It should be available as one of your distribution's core packages or you can download the source from the ISC website [1]. On the website you will be able to find source for versions 2.x and 3.x. Examples given here will work with either.

Configuration

The DHCP server has one configuration file, whose default location is */etc/dhcpd.conf* (you can specify a different file at runtime by passing a parameter on the command line). An example is shown in the DHCP Server Config boxout. As you can see, each line is terminated by a semi-colon, sub-options are contained within braces.

First come the global options. The lease lifespan (measured in seconds) has here been set to one day. There follow settings for the local domain name and and DNS servers. The subnet-mask global option provides a default for any subnet which does not have a netmask specified in its own declaration.

DHCP Server Config

```
# /etc/dhcpd.conf
#

# Option definitions common to ➥
  all supported networks...
default-lease-time 86400;
max-lease-time 86400;
option domain-name ➥
"example.org";
option domain-name-servers ➥
192.168.10.1, 192,168.10.5;
option subnet-mask 255.255.255.0;

# Options for each subnet
subnet 192.168.10.0 netmask ➥
255.255.255.0 {
  range 192.168.10.101 ➥
  192.168.10.200;
  option routers 192.168.10.1;
}

subnet 192.168.11.0 netmask ➥
255.255.255.0 {
  range 192.168.11.51 ➥
  192.168.11.90;
  range 192.168.11.200 ➥
  192.168.11.254;
  option routers 192.168.11.1;
}

subnet 192.168.12.0 netmask ➥
255.255.255.0 {
}

# Options for specific hosts ➥
host marx {
  hardware ethernet ➥
  00:08:20:81:77:82;
  fixed-address 192.168.10.51;
}

host engels {
  fixed-address 192.168.10.52; ➥
  34 }
```

Next we have some subnet declarations, each giving specific options for a subnet to which this machine is connected. The first two declarations each allocate a range of IP addresses to the pool for that subnet and also give the router IP address. The third subnet declaration is empty, indicating that the server will not respond to requests from that subnet.

Important: There must be a subnet declaration for each subnet for which the

host has a configured network interface, unless the server was set at runtime to only listen on specific interfaces (see the Section called Running the Server). In the latter case there must be a declaration for each specified subnet.

Finally, some host declarations, which specify fixed IP addresses for particular hosts. The first declaration specifies a MAC address and so will allocate the IP address to any request coming from that network card, whether or not the request includes the "marx" hostname. Also in contrast, the second declaration means that 192.168.10.52 will be assigned to any request including the "engels" hostname, even if an existing lease has already been granted to another machine using the same name.

Caution – Any fixed IP addresses assigned in host declarations must not be from within ranges that have been assigned to subnet pools.

Running the Server

You can launch the server directly from the command line, as in this example:

```
/usr/sbin/dhcpd -cf /etc/dhcp/dhcpd.conf eth0 eth1
```

In this case the daemon has been told to use an alternate configuration file and to listen only on interfaces eth0 and eth1.

In practice, however, you are best to stop and start the daemon by using the init scripts provided with the package. On Debian, for instance, you would restart the daemon thus:

```
/etc/init.d/dhcp restart
```

If you wanted to pass extra parameters to the daemon you would have to edit */etc/default/dhcp*. If you are using another distribution, please consult your distribution's documentation for details.

The daemon must be restarted for any changes to the configuration file to take effect.

The Lease File

The DHCP server keeps a record of the current leases in a text file (on Debian this is */var/lib/dhcp/dhcp.leases*, with a backup called *dhcp.leases~*. The daemon reloads it on start-up and will fail if it can't find it (which can happen if the daemon fails at a crucial point). If this happens, copy the backup file back to *dhcp.leases* and restart the daemon.

Each record in the leases file records the start and end date/time, MAC address, hostname (if given) and IP address. This can be of use either to other applications or to your own scripts. One example is given in the Section called Dynamic DNS Updates.

Pump Config File

```
# /etc/pump.conf

device eth0 {
        nodns
}

script /usr/local/sbin/dhcp
```

Configuring the Client

For a Linux box to configure its network interfaces using DHCP, it requires a DHCP client. The two most commonly used are pump, a simple client developed by Red Hat, and dhclient, a fully featured client from the Internet Software Consortium.

Both work in the same simple way: when the client is run it sends out a series of broadcast requests until a valid reply is received. The client then configures the network interface and other parameters specified by the DHCP server, after which it runs as a daemon in the background, sending renewal requests as necessary.

Both of the clients can be configured further to specify how they use the data returned to them by the DHCP server and to run a script on the granting or renewal of a lease.

pump

pump doesn't support the full range of configuration options that can be passed through DHCP and isn't as flexible as dhclient but is adequate for most set-ups. It is the default DHCP client for many of the distributions and there should be a package available for you.

Once installed, configuring an interface using pump can be as simple as this:

```
/sbin/pump -i eth0
```

Which will set pump to managing eth0. As soon as it successfully obtains a lease it will configure the interface.

You can modify pump's behaviour by passing it further command line options or by editing its configuration file, */etc/pump.conf*. The example file shown in the Pump Config File boxout tells pump not to rewrite */etc/resolv.conf* if it receives DNS configuration information with the lease for eth0 and to run the user-written script */usr/local/sbin/dhcp* whenever a lease is granted, renewed or released. The script is passed the action ('up', 'renewal' or 'down'), IP address and interface name as parameters.

dhclient

Using dhclient to configure an interface is just as simple as pump:

```
/sbin/dhclient eth0
```

You can also modify dhclient's behaviour by editing */etc/dhclient.conf*, though in most cases it will function perfectly well without a configuration file. The options for dhclient configuration are much more complex and flexible than pump, as we show in the dhclient Config File boxout.

The global options specify firstly that dhclient should try to obtain a lease for 60 seconds before giving up and secondly that it should wait a further 30 seconds before trying again.

The interface declaration sets options for dhclient to use when obtaining leases for the eth0 interface. In this case, dhclient should identify the hostname as "marx", request an hour-long lease and add 127.0.0.1 to the list of name servers it receives from the server. The request option specifies what information dhclient should ask for and the require option tells dhclient to reject entirely any response which doesn't include a subnet mask and list of name servers.

It is possible to have dhclient run user-defined scripts when either obtaining or renewing leases but as this is a more complex affair than with pump you should read all the man pages that come with the dhclient package before you attempt this.

Your distribution will have a dhclient package and you can also get the source code from the ISC website (see the Info boxoutat the end of this article).

Doing it the Easy Way

Thankfully, you rarely need to bother with any of the above complexity, nor with running the DHCP client yourself. In all the major distributions you simply

dhclient Config File

```
# /etc/dhclient.conf

timeout 60;
retry 30;

interface "eth0" {
  send host-name "marx";
  send dhcp-lease-time 3600;
  prepend domain-name-servers ➥
  127.0.0.1;
  request subnet-mask, ➥
  broadcast-address, routers,➥
  domain-name, ➥
  domain-name-servers, host-name;
  require subnet-mask, ➥
  domain-name-servers;
}
```

have to specify in the network config files that an interface should use DHCP. When the network interface is brought up (for example using the ifup command), the networking scripts will use whichever of the clients is installed.

In the Example Interface Config Files boxout you can see an example of how this is done on Debian and Red Hat.

Dynamic DNS Updates

Historically, one drawback to configuring network hosts dynamically has been that their details are not stored in DNS. The DNS standard now, however, includes a mechanism for sending updates to a DNS server. This makes it possible to update your DNS records to reflect the leases given out by your DHCP server.

In configuring your network for DDNS it is possible for the DHCP server and/or the client to send the updates to the name server and then to use secure keys for protection. For simplicity's sake, this example allows only the server to send updates and does not use security.

Example Interface Config Files

```
Debian config file:
# /etc/network/interfaces

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp

Red Hat config file:

# /etc/sysconfig/➥
network-scripts/ifcfg-eth0

DEVICE=eth0
ONBOOT=yes
BOOTPROTO=dhcp
```

BIND Configured for Dynamic Updates

```
# /etc/named.conf

options {
  directory "/var/cache/bind";
};

zone "." {
  type hint "/etc/bind/db.root";
  file "/etc/bind/db.root";
};

zone "internal" {
  type master;
  file "db.internal";
  allow-update {192.168.10.6;};
};

zone "0.0.127.in-addr.arpa" {
  type master;
  file "db.root";
};

zone "10.168.192.in-addr.arpa" {
  type master;
  file "db.10.168.192";
  allow-update {192.168.10.6;};
  };
```

Configuring Bind

The first step is to configure your name server to allow dynamic updates. For BIND 8.x, this can be done as shown in the BIND Configured for Dynamic Updates boxout. In this case, the BIND config file from the previous article in this series has been modified to allow updates to the main domain. The alteration is simply the two allow-update directives, which tell the respective forward and reverse zones that they should accept updates from 192.168.10.6 (the address of the DHCP server).

Configuring DHCPD 3.x

If you want the DHCP server to do the updates itself, you need version 3.x. You should add the following options to *dhcpd.conf* (altering the values to suit your own network):

```
ddns-domainname➥
"example.org";
ddns-update-style "interim";
deny client-updates;

zone example.org. {
        primary 192.168.10.1;
}

zone 254.10.168.192.in-addr.➥
 arpa. {
        primary 192.168.10.1;
}
```

The primary setting gives the IP address of the name server to send the updates to.

Working with DHCPD 2.x

If you have DHCPD 2.x then the DHCP server itself cannot perform updates. There are alternatives, however. Stephen Carville has written some perl scripts that can be used to monitor the *dhcpd.leases* file and send updates using the nsupdate binary from the BIND package [2].

Endnotes

This article, the last in the series, has shown you how to set up a DHCP server, how to configure workstations using DHCP and how to set up dynamic DNS updates. The four articles should provide you with all the information you need to set up simple Linux networks. Hopefully, they also provide enough tasters to make you ambitious to try greater things with your systems. ■

INFO

[1] ISC DHCP tools: *http://www.isc.org/products/DHCP/*

[2] Stephen Carville's DHCP-DNS: *http://www.heronforge.net/~stephen/DHCP-DNS/dhcp-dns.html*

[3] Secure DDNS HOWTO: *http://ops.ietf.org/dns/dynupd/secure-ddns-howto.html*