DHCP

Jason Healy, Director of Networks and Systems

Last Updated Mar 18, 2008

Contents

Chapter 1

DHCP

Last updated 2008/03/18

1.1 Introduction

The **D**ynamic **H**ost **C**onfiguration **P**rotocol (**DHCP**) is a mechanism for automatically configuring networked computers from a centralized server. Usually, DHCP is used to provide IP addresses to clients without their needing to know anything about the network they are connected to. With more configuration, DHCP can also be set up to provide a rich set of information to clients, from routers to DNS servers all the way up to LDAP providers and diskless bootstrap information.

At Suffield, we use DHCP to automatically assign IP addresses to computers on our network. Because most of our users own laptops, we have the additional challenge of users changing locations (and thus, subnets) frequently. DHCP is used to not only hand out the proper IP address for each client, but also provide subnet- and LAN-specific information to each client so that they can connect to the network without requiring intervention by the user.

1.2 Mac OS X Server Setup

We run DHCP service on machines running Mac OS X Server 10.4. Mac OS X Server comes with its own built-in DHCP server. This server can be configured via the **Server Admin** GUI, and is also integrated into Apple's NetBoot services.

Unfortunately, Apple's DHCP server does not yet implement all of the features we need in a DHCP server, including dynamic DNS updates, failover, and known/unknown client grouping.

For this reason, we choose to install the ISC DHCP server on our Mac OS X Server machines, and configure it as we would on any other standard UNIX machine. (Note that this config would work equally well on a UNIX box, as the version of the ISC server is the same for all platforms.)

1.2.1 Getting dhcpd from DarwinPorts

DarwinPorts is a collection of ready-to-compile software for Mac OS X. If you have not yet installed DarwinPorts, please see our section describing the installation of DarwinPorts. (The section discusses the installation of Subversion, but the DarwinPorts information is not specific to that package.)

Once you have DarwinPorts installed, you simply need to install the `dhcp` port:

```
sudo port install dhcp
```

DarwinPorts will download, unpack, configure, compile, and clean the distribution for you automatically. When done, you should be able to run the following:

```
/opt/local/sbin/dhcpd --help
```

You should get the standard help screen for DHCP. If you do, then the binaries are installed, and you're ready to move on.

Note: as part of the installation process, DarwinPorts creates a new StartupItem called **DarwinPortsStartup**. This item then bootstraps any installed daemons under the DarwinPorts system. In its default state, no services are launched, so the item won't do any harm. We use Mac OS X 10.4 (Tiger), so we will configure **launchd** to start the DHCP server instead.

1.3 Suffield DNS Config Template

To make setting up DHCP servers easier, we have broken our standard DHCP config into several include files, along with a "primary" and "secondary" configuration template file. Taken together, these files can be quickly pieced together to form a functioning primary or secondary DHCP server.

The files are stored in revision control, and can also be viewed directly from the web:

Suffield DHCP Config Template

For servers, we recommend checking the files out of version control. This way, changes to the files can be easily synched up using our version control software.

1.3.1 Differences

Our configuration is broken up into several different files, which makes it easy to plug in different functionality for different servers while maintaining a consistent set of global options. The primary and secondary configs simply include these core options, and only add the small changes required for the symmetry in a failover pair.

In the file segments themselves, these are the major pieces of functionality we implement:

- A *global options* file, which contains all of the server-wide option settings (nameservers, domain names, default lease times, *etc.*).
- A *ddns options* file, which covers all settings related to Dynamic DNS updates (DDNS).
- A *zones* file, which contains all of the forward and reverse DNS zone declarations used by DDNS.
- A *keys* file, to hold DNSSEC keys to digitally sign DNS updates.
- A series of *clients* files, which contain the static host declarations for known clients.
- A *common* file, which ties together all of these options into a single master file.
- *Primary* and *secondary* files, which add the necessary statements for failover operation. These are the files you should use as your main `dhcpd.conf` file on a server.

1.3.2 Using the Configuration

To use our base configuration, follow these steps:

1. Check out the stock config from our Subversion repository:

   ```
   cd /etc/

   sudo svn checkout \
   svn://svn.suffieldacademy.org/netadmin/trunk/software/dhcp/dhcp.d \
   dhcp.d
   ```

The command above checks out the stock directory to the name `dhcp.d`. You may choose a different name for the directory, but you will need to change all of the `include` statements in the config file to reflect the new path.

2. If you're using DNSSEC (which we do), you'll need to create an include file with all the DNSSEC keys in it. First, copy our template config:

   ```
   cp /etc/dhcp.d/key.inc.template /etc/dhcp.d/key.inc
   ```

   Then, edit the file and add any keys that are needed for inter-server communication.

3. Our configuration is designed for failover operation, so there are two files to choose from: `primary.conf` and `secondary.conf`. Link the appropriate file to `/etc/dhcpd.conf`:

   ```
   sudo ln -s /etc/dhcp.d/primary.conf /etc/dhcpd.conf
   ```

   or

   ```
   sudo ln -s /etc/dhcp.d/secondary.conf /etc/dhcpd.conf
   ```

4. DHCP will not start until you've created an empty lease file for it. This is to prevent errors if you accidentally move the lease file out of the way for manual recovery:

   ```
   sudo touch /var/db/dhcpd.leases
   ```

5. Our DHCP config is set to log via the `syslog` logging facility under UNIX. However, many systems are not configured to ”listen” for these log announcements. To fix this, edit `/etc/syslog.conf` and add the following line:

   ```
   local2.info /var/log/dhcpd.log
   ```

   (You may use a different log level facility if you wish; `info` prints all lease transactions and may be too verbose for some users.)

   You will need to kill and restart the `syslogd` daemon for these changes to take effect.

1. Finally, copy the LaunchDaemon plist file for DHCP into the server's `/Library/LaunchDaemons/` directory. You can download the file from our DHCP LaunchDaemon repository.

At this point, you are ready to start the DHCP server:

```
sudo launchctl load -w /Library/LaunchDaemons/org.isc.dhcpd.plist
```

Check the system logs to see if the server reports any errors. You may confirm that the service is running by typing:

```
sudo launchctl list
```

The `org.isc.dhcpd` identifier should appear if everything is running properly. Check the `/var/log/dhcpd.log` file to confirm that the server has started (it should print informational messages on startup).

1.4 Using DHCP

Once DHCP is installed and configured, you are ready to run the daemon in a production environment. Normally, you won't need to interfere with the daemon's operation; it should serve requests normally without need for restart or reconfiguration.

From time to time, you will need to reconfigure the server. Normally, this is done to add hosts to the database that have specific names or IP addresses.

1.4.1 Normal Operation Indicators

When the server is working properly, you will see entries in the log like this:

```
Apr 19 08:17:01 dhcpd: DHCPDISCOVER from de:ad:be:ef:00:0d via eth0
Apr 19 08:17:02 dhcpd: DHCPOFFER on 192.168.0.1 to de:ad:be:ef:00:0d via eth0
Apr 19 08:17:03 dhcpd: DHCPREQUEST for 192.168.0.1 from de:ad:be:ef:00:0d (jersey) via eth0
Apr 19 08:17:04 dhcpd: DHCPACK on 192.168.0.1 to de:ad:be:ef:00:0d (jersey) via eth0
```

In this case, a client has broadcast that it wants an address (DISCOVER), the server has made an offer (OFFER), the client confirms the offer by requesting the specific address (REQUEST), and the server confirms that the address is available for use (ACK).

Later, when clients already have an address, they will renew the address by performing the final two steps again (REQUEST/ACK).

You may occasionally see deny messages (DHCPNACK). These are not cause for concern; normally they are sent to clients that have moved between subnets and no longer have a valid address for the subnet they are on. The server sends a negative ACK, and the client begins the negotiation process anew to get a proper address.

1.4.2 Finding a Client's Address

If you ever need to find the MAC address that goes with an IP address (or vice-versa), just grep the log file for the IP or MAC address. The last matching entry will be the assignment made by the server.

1.4.3 Starting the Daemon

If you're using Mac OS X Server 10.4, the DHCP server should start itself via `launchd` if you've installed our LaunchDaemon item. If you've manually stopped the service, you can start it again using:

```
sudo launchctl load -w /Library/LaunchDaemons/org.isc.dhcpd.plist
```

The `-w` flag makes the load status permanent; that is, the server will continue to start the process from this point forward, even after a reboot of the machine.

If you use a regular UNIX system, launching `dhcpd` is usually accomplished using an `/etc/init.d/` or `/etc/rc.d/` script. Use the method described by your UNIX distribution.

1.4.4 Stoping the Daemon

If you're using Mac OS X Server 10.4, you must use `launchd` to stop the daemon process (if you don't, `launchd` will just start it back up for you). To do this, type the following:

```
sudo launchctl unload -w /Library/LaunchDaemons/org.isc.dhcpd.plist
```

The `-w` flag makes the unload status permanent; that is, the server will never start up again until you re-enable it with `launchctl` (even if you reboot).

If you use a regular UNIX system, stopping `dhcpd` is usually accomplished using an `/etc/init.d/` or `/etc/rc.d/` script. Use the method described by your UNIX distribution. If you're in a pinch, you can also try killing the process directly. Assuming it isn't being respawned by `init`, the DHCP server should exit and stop running.

1.4.5 Loading Configuration Changes

The ISC DHCP server does not have a ”reload” option. Therefore, the only way to re-read the configuration files is to stop the server and start it back up again.

When you've made changes to the configuration and are ready for them to take effect, you should always test the configuration first:

```
sudo /opt/local/sbin/dhcpd -t
```

Confirm that there are no errors reported before you restart the server.

If you're using `launchd` to run DHCP (as we describe above), reloading the configuration is as simple as manually killing the process:

```
sudo killall dhcpd
```

`launchd` will notice that the process has died, and will automatically restart it. Upon restart, the new configuration will be read, and your changes should take effect.

On systems that use a traditional start/stop mechanism, you can simply stop the server and start it immediately after, using the start and stop procedures described above.

1.4.6 Adding Fixed Hosts

While the DHCP server usually tries to give the same address to clients every time, there is no guarantee. Some hosts on the network should be given ”fixed” addresses that never change (such as network equipment, appliances, and other vital equipment).

Note: ”regular” clients can be given a name and parameters that don't change, without needing to assign a specific IP address. If you're adding a record for a user's computer, please see the next section for information on how to add a host record with a dynamic address.

To ensure that a client gets a fixed address every time, follow these steps:

1. Create forward and reverse DNS entries for the client on your DNS server.

2. Edit the section of your configuration files where host declarations are kept. In our config, the host info is kept in the `client_*.inc` files. You should add a stanza that looks like this:

   ```
   host my-new-hostname {
       hardware ethernet fa:ce:ca:fe:ba:be;
       fixed-address new-hostname.example.org;
   }
   ```

This tells the DHCP server to always associate the IP address that `new-hostname.example.org` resolves to with the given Ethernet MAC address.

3. Restart the server to reload the configuration.

1.4.7 Adding Registered Hosts

Under our configuration, the DHCP server knows about three types of clients: ”fixed” clients that are given static IP addresses, ”registered” clients that have some identifying information known about them, and ”rogue” clients that are unknown to the server.

By default, all clients get addresses from the server, so it is not necessary to register a client before connecting it to the network. However, ”registered” clients receive better treatment, in the form of (possibly) longer leases, dynamic DNS updates, and a larger pool of addresses.

Registered hosts should be used for any hosts that should receive a consistent name in DNS, but that might travel between subnets (and thus need a different IP at different times). It’s somewhat of a comprimise between ”rogue” hosts and ”fixed” hosts.

To add a registered host, follow these steps:

1. Edit the section of your configuration files where host declarations are kept. In our config, the host info is kept in the `client_*.inc` files. You should add a stanza that looks like this:

   ```
   host my-new-hostname {
       hardware ethernet de:ad:be:ef:00:00;
       ddns-hostname "my-new-hostname";
   }
   ```

   This tells the DHCP server to always associate the given DNS hostname with the given Ethernet MAC address. The Dynamic DNS (DDNS) features of the DHCP server will ensure that the client always gets the same DNS name, even if the IP address it resolves to changes.

2. Restart the server to reload the configuration.

1.4.8 Failover

Our default configuration employs a peered failover system to ensure reliable service. The ISC DHCP server supports the DHCP failover protocol with only minimal extra configuration.

This section gives a brief guide for using failover and recovering from disasters. For more detailed information, please see the manual page for `dhcpd.conf` and `dhcpd`.

For the most part, failover requires no intervention on the part of the administrator. When one server fails, the other notices (you'll see a log message), and will attempt to service clients that were handled by the peer server.

For short outages (server reboots, configuration changes, etc), you don't need to worry about failover. When the other machine comes back, it will automatically syncronize with the running server.

For long outages (days, weeks, months), you may wish to manually place the running server into **partner-down** mode. This tells the running server that its peer is known to be down, so it can take over full service for the network.

Getting In to the Partner-Down State

Note: if you place a server into **partner-down** mode, you must be **absolutely sure** that the peer is legitimately down, and not just out of contact with the server. If the failed server is still running, and the running server begins to take over its addresses, then both servers might hand out the same addresses to different clients.

If a peer in a failover setup fails, and you wish to have the surviving server take over all service for the network, follow these steps:

1. Stop the functioning server, so that it no longer is servicing clients (make sure the server does not automatically restart if you are using `launchd`).

2. Edit the `/var/db/dhcpd.leases` file, and search for the **last** peer declaration stanza. Remove the date from the "`my`" portion of the state, so it is blank. Change the "`my`" portion of the state to have a status of `partner-down`. Leave the "`peer`" portion of the state alone. For example:

```
failover peer "suffield" state {
    my state partner-down;
    peer state communications-interrupted at 2 2005/08/23 13:14:15;
}
```

3. Start the server back up again, and verify that the state is now **partner-down** in the log files.

Over time, the running server will reclaim leases that used to be handled by the failed server.

Getting Out of the Partner-Down State

Note: you **must not** bring a failed peer back up unless the running peer is still functioning and ready to communicate with the failed peer. If you do otherwise, the restored peer will assume that nothing was wrong, and begin serving addresses normally. The restored peer must be able to communicate with the running peer right away so that it can discover that the other peer is operating in the **partner-down** state, and synchronize its database correctly.

After a prolonged outage where the running server has been placed in the **partner-down** state, when you are ready to bring a failed peer back online you simply need to confirm that the two servers can communicate with each other.

Start the failed peer up normally. It should restore communication with the running peer, discover that the running peer took over service, and synchronize its database with the running peer. Once the two machines have synchronized, the restored machine will wait for **MCLT** before serving clients. This gives any clients that received a lease from the failed peer time to expire, preventing conflicts.

At this point, both servers should return to their normal state, and begin serving clients in failover mode again.

Recovering from a Lost Lease Database

If a peer fails catastrophically and loses its lease database, you do not need to take any special action to recover. When the failed peer is repaired and able to communicate with the running peer, you may start it normally.

The failed peer will note that it has no state information, and assume that it needs to synchronize its state. It will download a copy of the lease database from the running peer, wait **MCLT** to prevent address conflicts, and then begin serving requests again normally.