NAME

dhcpd - Dynamic Host Configuration Protocol Server

SYNOPSIS

dhcpd [**-p** *port*] [**-f**] [**-d**] [**-q**] [**-t** | **-T**] [**-cf** *config-file*] [**-lf** *lease-file*] [**-tf** *trace-output-file*] [**-play** *trace-playback-file*] [*if0* [*...ifN*]]

DESCRIPTION

The Internet Software Consortium DHCP Server, dhcpd, implements the Dynamic Host Configuration Protocol (DHCP) and the Internet Bootstrap Protocol (BOOTP). DHCP allows hosts on a TCP/IP network to request and be assigned IP addresses, and also to discover information about the network to which they are attached. BOOTP provides similar functionality, with certain restrictions.

CONTRIBUTIONS

This software is free software. At various times its development has been underwritten by various organizations, including the ISC and Vixie Enterprises. The development of 3.0 has been funded almost entirely by Nominum, Inc.

At this point development is being shepherded by Ted Lemon, and hosted by the ISC, but the future of this project depends on you. If you have features you want, please consider implementing them.

OPERATION

The DHCP protocol allows a host which is unknown to the network administrator to be automatically assigned a new IP address out of a pool of IP addresses for its network. In order for this to work, the network administrator allocates address pools in each subnet and enters them into the dhcpd.conf(5) file.

On startup, dhcpd reads the *dhcpd.conf* file and stores a list of available addresses on each subnet in memory. When a client requests an address using the DHCP protocol, dhcpd allocates an address for it. Each client is assigned a lease, which expires after an amount of time chosen by the administrator (by default, one day). Before leases expire, the clients to which leases are assigned are expected to renew them in order to continue to use the addresses. Once a lease has expired, the client to which that lease was assigned is no longer permitted to use the leased IP address.

In order to keep track of leases across system reboots and server restarts, dhcpd keeps a list of leases it has assigned in the dhcpd.leases(5) file. Before dhcpd grants a lease to a host, it records the lease in this file and makes sure that the contents of the file are flushed to disk. This ensures that even in the event of a system crash, dhcpd will not forget about a lease that it has assigned. On startup, after reading the dhcpd.conf file, dhcpd reads the dhcpd.leases file to refresh its memory about what leases have been assigned.

New leases are appended to the end of the dhcpd.leases file. In order to prevent the file from becoming arbitrarily large, from time to time dhcpd creates a new dhcpd.leases file from its in-core lease database. Once this file has been written to disk, the old file is renamed *dhcpd.leases~*, and the new file is renamed dhcpd.leases. If the system crashes in the middle of this process, whichever dhcpd.leases file remains will contain all the lease information, so there is no need for a special crash recovery process.

BOOTP support is also provided by this server. Unlike DHCP, the BOOTP protocol does not provide a protocol for recovering dynamically-assigned addresses once they are no longer needed. It is still possible to dynamically assign addresses to BOOTP clients, but some administrative process for reclaiming addresses is required. By default, leases are granted to BOOTP clients in perpetuity, although the network administrator may set an earlier cutoff date or a shorter lease length for BOOTP leases if that makes sense.

BOOTP clients may also be served in the old standard way, which is to simply provide a declaration in the dhcpd.conf file for each BOOTP client, permanently assigning an address to each client.

Whenever changes are made to the dhcpd.conf file, dhcpd must be restarted. To restart dhcpd, send a SIGTERM (signal 15) to the process ID contained in */var/run/dhcpd.pid*, and then re-invoke dhcpd. Because the DHCP server database is not as lightweight as a BOOTP database, dhcpd does not automatically restart itself when it sees a change to the dhcpd.conf file.

Note: We get a lot of complaints about this. We realize that it would be nice if one could send a SIGHUP to the server and have it reload the database. This is not technically impossible, but it would require a great deal of work, our resources are extremely limited, and they can be better spent elsewhere. So please

don't complain about this on the mailing list unless you're prepared to fund a project to implement this feature, or prepared to do it yourself.

COMMAND LINE

The names of the network interfaces on which dhcpd should listen for broadcasts may be specified on the command line. This should be done on systems where dhcpd is unable to identify non-broadcast interfaces, but should not be required on other systems. If no interface names are specified on the command line dhcpd will identify all network interfaces which are up, eliminating non-broadcast interfaces if possible, and listen for DHCP broadcasts on each interface.

If dhcpd should listen on a port other than the standard (port 67), the **-p** flag may used. It should be followed by the udp port number on which dhcpd should listen. This is mostly useful for debugging purposes.

To run dhcpd as a foreground process, rather than allowing it to run as a daemon in the background, the **-f** flag should be specified. This is useful when running dhcpd under a debugger, or when running it out of inittab on System V systems.

To have dhcpd log to the standard error descriptor, specify the **-d** flag. This can be useful for debugging, and also at sites where a complete log of all dhcp activity must be kept but syslogd is not reliable or otherwise cannot be used. Normally, dhcpd will log all output using the syslog(3) function with the log facility set to LOG_DAEMON.

Dhcpd can be made to use an alternate configuration file with the **-cf** flag, or an alternate lease file with the **-lf** flag. Because of the importance of using the same lease database at all times when running dhcpd in production, these options should be used **only** for testing lease files or database files in a non-production environment.

When starting dhcpd up from a system startup script (e.g., /etc/rc), it may not be desirable to print out the entire copyright message on startup. To avoid printing this message, the **-q** flag may be specified.

The DHCP server reads two files on startup: a configuration file, and a lease database. If the **-t** flag is specified, the server will simply test the configuration file for correct syntax, but will not attempt to perform any network operations. This can be used to test the a new configuration file automatically before installing it.

The **-T** flag can be used to test the lease database file in a similar way.

The **-tf** and **-play** options allow you to specify a file into which the entire startup state of the server and all the transactions it processes are either logged or played back from. This can be useful in submitting bug reports - if you are getting a core dump every so often, you can start the server with the **-tf** option and then, when the server dumps core, the trace file will contain all the transactions that led up to it dumping core, so that the problem can be easily debugged with **-play**.

The **-play** option must be specified with an alternate lease file, using the **-lf** switch, so that the DHCP server doesn't wipe out your existing lease file with its test data. The DHCP server will refuse to operate in playback mode unless you specify an alternate lease file.

CONFIGURATION

The syntax of the dhcpd.conf(5) file is discussed separately. This section should be used as an overview of the configuration process, and the dhcpd.conf(5) documentation should be consulted for detailed reference information.

Subnets

dhcpd needs to know the subnet numbers and netmasks of all subnets for which it will be providing service. In addition, in order to dynamically allocate addresses, it must be assigned one or more ranges of addresses on each subnet which it can in turn assign to client hosts as they boot. Thus, a very simple configuration providing DHCP support might look like this:

```
subnet 239.252.197.0 netmask 255.255.255.0 {
  range 239.252.197.10 239.252.197.250;
}
```

Multiple address ranges may be specified like this:

```
subnet 239.252.197.0 netmask 255.255.255.0 {
  range 239.252.197.10 239.252.197.107;
  range 239.252.197.113 239.252.197.250;
}
```

If a subnet will only be provided with BOOTP service and no dynamic address assignment, the range clause can be left out entirely, but the subnet statement must appear.

Lease Lengths

DHCP leases can be assigned almost any length from zero seconds to infinity. What lease length makes sense for any given subnet, or for any given installation, will vary depending on the kinds of hosts being served.

For example, in an office environment where systems are added from time to time and removed from time to time, but move relatively infrequently, it might make sense to allow lease times of a month of more. In a final test environment on a manufacturing fbor, it may make more sense to assign a maximum lease length of 30 minutes - enough time to go through a simple test procedure on a network appliance before packaging it up for delivery.

It is possible to specify two lease lengths: the default length that will be assigned if a client doesn't ask for any particular lease length, and a maximum lease length. These are specified as clauses to the subnet command:

```
subnet 239.252.197.0 netmask 255.255.255.0 {
  range 239.252.197.10 239.252.197.107;
  default-lease-time 600;
  max-lease-time 7200;
}
```

This particular subnet declaration specifies a default lease time of 600 seconds (ten minutes), and a maximum lease time of 7200 seconds (two hours). Other common values would be 86400 (one day), 604800 (one week) and 2592000 (30 days).

Each subnet need not have the same lease—in the case of an office environment and a manufacturing environment served by the same DHCP server, it might make sense to have widely disparate values for default and maximum lease times on each subnet.

BOOTP Support

Each BOOTP client must be explicitly declared in the dhcpd.conf file. A very basic client declaration will specify the client network interface's hardware address and the IP address to assign to that client. If the client needs to be able to load a boot file from the server, that file's name must be specified. A simple bootp client declaration might look like this:

```
host haagen {
  hardware ethernet 08:00:2b:4c:59:23;
  fixed-address 239.252.197.9;
  filename "/tftpboot/haagen.boot";
}
```

Options

DHCP (and also BOOTP with Vendor Extensions) provide a mechanism whereby the server can provide the client with information about how to configure its network interface (e.g., subnet mask), and also how the client can access various network services (e.g., DNS, IP routers, and so on).

These options can be specified on a per-subnet basis, and, for BOOTP clients, also on a per-client basis. In the event that a BOOTP client declaration specifies options that are also specified in its subnet declaration, the options specified in the client declaration take precedence. A reasonably complete DHCP configuration might look something like this:

```
subnet 239.252.197.0 netmask 255.255.255.0 {
  range 239.252.197.10 239.252.197.250;
  default-lease-time 600 max-lease-time 7200;
  option subnet-mask 255.255.255.0;
  option broadcast-address 239.252.197.255;
  option routers 239.252.197.1;
  option domain-name-servers 239.252.197.2, 239.252.197.3;
  option domain-name "isc.org";
}
```

A bootp host on that subnet that needs to be in a different domain and use a different name server might be declared as follows:

```
host haagen {
  hardware ethernet 08:00:2b:4c:59:23;
  fixed-address 239.252.197.9;
  filename "/tftpboot/haagen.boot";
  option domain-name-servers 192.5.5.1;
  option domain-name "vix.com";
}
```

A more complete description of the dhcpd.conf file syntax is provided in dhcpd.conf(5).

OMAPI

The DHCP server provides the capability to modify some of its configuration while it is running, without stopping it, modifying its database files, and restarting it. This capability is currently provided using OMAPI - an API for manipulating remote objects. OMAPI clients connect to the server using TCP/IP, authenticate, and can then examine the server's current status and make changes to it.

Rather than implementing the underlying OMAPI protocol directly, user programs should use the dhcpctl API or OMAPI itself. Dhcpctl is a wrapper that handles some of the housekeeping chores that OMAPI does not do automatically. Dhcpctl and OMAPI are documented in **dhcpctl(3)** and **omapi(3)**.

OMAPI exports objects, which can then be examined and modified. The DHCP server exports the following objects: lease, host, failover-state and group. Each object has a number of methods that are provided: lookup, create, and destroy. In addition, it is possible to look at attributes that are stored on objects, and in some cases to modify those attributes.

THE LEASE OBJECT

Leases can't currently be created or destroyed, but they can be looked up to examine and modify their state.

Leases have the following attributes:

state *integer* lookup, examine

1 = free
2 = active
3 = expired
4 = released
5 = abandoned
6 = reset
7 = backup
8 = reserved
9 = bootp

ip-address *data* lookup, examine

The IP address of the lease.

dhcp-client-identifier *data* lookup, examine, update

The client identifier that the client used when it acquired the lease. Not all clients send client identifiers, so this may be empty.

client-hostname *data* examine, update
The value the client sent in the host-name option.

host *handle* examine
the host declaration associated with this lease, if any.

subnet *handle* examine
the subnet object associated with this lease (the subnet object is not currently supported).

pool *handle* examine
the pool object associted with this lease (the pool object is not currently supported).

billing-class *handle* examine
the handle to the class to which this lease is currently billed, if any (the class object is not currently supported).

hardware-address *data* examine, update
the hardware address (chaddr) field sent by the client when it acquired its lease.

hardware-type *integer* examine, update
the type of the network interface that the client reported when it acquired its lease.

ends *time* examine
the time when the lease's current state ends, as understood by the client.

tstp *time* examine
the time when the lease's current state ends, as understood by the server.

tsfp *time* examine
the time when the lease's current state ends, as understood by the failover peer (if there is no failover peer, this value is undefined).

cltt *time* examine
The time of the last transaction with the client on this lease.

THE HOST OBJECT

Hosts can be created, destroyed, looked up, examined and modified. If a host declaration is created or deleted using OMAPI, that information will be recorded in the dhcpd.leases file. It is permissible to delete host declarations that are declared in the dhcpd.conf file.

Hosts have the following attributes:

name *data* lookup, examine, modify
the name of the host declaration. This name must be unique among all host declarations.

group *handle* examine, modify
the named group associated with the host declaration, if there is one.

hardware-address *data* lookup, examine, modify
the link-layer address that will be used to match the client, if any. Only valid if hardware-type is also present.

hardware-type *integer* lookup, examine, modify
the type of the network interface that will be used to match the client, if any. Only valid if hardware-address is also present.

dhcp-client-identifier *data* lookup, examine, modify
the dhcp-client-identifier option that will be used to match the client, if any.

ip-address *data* examine, modify
a fixed IP address which is reserved for a DHCP client that matches this host declaration. The IP address will only be assigned to the client if it is valid for the network segment to which the client is connected.

statements *data* modify
a list of statements in the format of the dhcpd.conf file that will be executed whenever a message

from the client is being processed.

known *integer* examine, modify
if nonzero, indicates that a client matching this host declaration will be treated as *known* in pool permit lists. If zero, the client will not be treated as known.

THE GROUP OBJECT

Named groups can be created, destroyed, looked up, examined and modified. If a group declaration is created or deleted using OMAPI, that information will be recorded in the dhcpd.leases file. It is permissible to delete group declarations that are declared in the dhcpd.conf file.

Named groups currently can only be associated with hosts - this allows one set of statements to be efficiently attached to more than one host declaration.

Groups have the following attributes:

name *data*
the name of the group. All groups that are created using OMAPI must have names, and the names must be unique among all groups.

statements *data*
a list of statements in the format of the dhcpd.conf file that will be executed whenever a message from a client whose host declaration references this group is processed.

THE CONTROL OBJECT

The control object allows you to shut the server down. If the server is doing failover with another peer, it will make a clean transition into the shutdown state and notify its peer, so that the peer can go into partner down, and then record the "recover" state in the lease file so that when the server is restarted, it will automatically resynchronize with its peer.

On shutdown the server will also attempt to cleanly shut down all OMAPI connections. If these connections do not go down cleanly after five seconds, they are shut down pre-emptively. It can take as much as 25 seconds from the beginning of the shutdown process to the time that the server actually exits.

To shut the server down, open its control object and set the state attribute to 2.

THE FAILOVER-STATE OBJECT

The failover-state object is the object that tracks the state of the failover protocol as it is being managed for a given failover peer. The failover object has the following attributes (please see **dhcpd.conf (5)** for explanations about what these attributes mean):

name *data* examine
Indicates the name of the failover peer relationship, as described in the server's **dhcpd.conf** file.

partner-address *data* examine
Indicates the failover partner's IP address.

local-address *data* examine
Indicates the IP address that is being used by the DHCP server for this failover pair.

partner-port *data* examine
Indicates the TCP port on which the failover partner is listening for failover protocol connections.

local-port *data* examine
Indicates the TCP port on which the DHCP server is listening for failover protocol connections for this failover pair.

max-outstanding-updates *integer* examine
Indicates the number of updates that can be outstanding and unacknowledged at any given time, in this failover relationship.

mclt *integer* examine
Indicates the maximum client lead time in this failover relationship.

load-balance-max-secs *integer* examine

Indicates the maximum value for the secs field in a client request before load balancing is bypassed.

load-balance-hba *data* examine
Indicates the load balancing hash bucket array for this failover relationship.

local-state *integer* examine, modify
Indicates the present state of the DHCP server in this failover relationship. Possible values for state are:

1 - partner down
2 - normal
3 - communications interrupted
4 - resolution interrupted
5 - potential conflict
6 - recover
7 - recover done
8 - shutdown
9 - paused
10 - startup
11 - recover wait

In general it is not a good idea to make changes to this state. However, in the case that the failover partner is known to be down, it can be useful to set the DHCP server's failover state to partner down. At this point the DHCP server will take over service of the failover partner's leases as soon as possible, and will give out normal leases, not leases that are restricted by MCLT. If you do put the DHCP server into the partner-down when the other DHCP server is not in the partner-down state, but is not reachable, IP address assignment conflicts are possible, even likely. Once a server has been put into partner-down mode, its failover partner must not be brought back online until communication is possible between the two servers.

partner-state *integer* examine
Indicates the present state of the failover partner.

local-stos *integer* examine
Indicates the time at which the DHCP server entered its present state in this failover relationship.

partner-stos *integer* examine
Indicates the time at which the failover partner entered its present state.

hierarchy *integer* examine
Indicates whether the DHCP server is primary (0) or secondary (1) in this failover relationship.

last-packet-sent *integer* examine
Indicates the time at which the most recent failover packet was sent by this DHCP server to its failover partner.

last-timestamp-received *integer* examine
Indicates the timestamp that was on the failover message most recently received from the failover partner.

skew *integer* examine
Indicates the skew between the failover partner's clock and this DHCP server's clock

max-response-delay *integer* examine
Indicates the time in seconds after which, if no message is received from the failover partner, the partner is assumed to be out of communication.

cur-unacked-updates *integer* examine
Indicates the number of update messages that have been received from the failover partner but not yet processed.

FILES

/etc/dhcpd.conf, /var/lib/dhcp/dhcpd.leases, /var/run/dhcpd.pid, /var/lib/dhcp/dhcpd.leases~.

SEE ALSO

dhclient(8), dhcrelay(8), dhcpd.conf(5), dhcpd.leases(5)

AUTHOR

dhcpd(8) was originally written by Ted Lemon under a contract with Vixie Labs. Funding for this project was provided by the Internet Software Consortium. Version 3 of the DHCP server was funded by Nominum, Inc. Information about the Internet Software Consortium is available at **http://www.isc.org/isc**. Information about Nominum and support contracts for DHCP and BIND can be found at **http://www.nominum.com**.