NAME

dhcpd − dynamic host configuration protocol daemon

SYNOPSIS

dhdpd [**−qar**] [**−t**[*level*]] [**−d**[*level*]] [**−f** *configfile*] [**−c** *cachefile*] [**−p** *poolfile*] [*host* ...]

DESCRIPTION

Dhcpd is a client and a server for the Dynamic Host Configuration Protocol. As a client it collects DHCP data to configure the Ethernet networks with, and as a server it answers DHCP queries from other machines.

This manual page describes the operation of **dhcpd**, the associated configuration file is described in **dhcp.conf**(5). (The latter, together with **boot**(8), is of more practical value when it comes to getting a machine's networks interfaces up and running. See the options section below for debugging DCHP problems.)

Initialization

On a normal startup, i.e. none of the **−q**, **−a** or **−r** options are given, **dhcpd** determines what IP devices are present, and which of those are Ethernets. For each network it looks for information in the configuration file as if it were a server answering a query for that network. If any information is found then the IP address is configured and the information stored in the cache file.

Client Operation

For each still unconfigured network a DHCP DISCOVER request is broadcast on that network. If a DHCP OFFER reply is received then a DHCP REQUEST is broadcast for the IP address offered, and if a DHCP ACK is received then the network is configured and the information stored in the cache file.

If no reply is received then another query is sent after 4 seconds, and then again after 8 seconds, doubling each time until 64 seconds. Every 64 seconds thereafter a request is broadcast until a reply is received.

Once configured the DHCP lease, rebind and renew times are computed. At the renew time a DHCP REQUEST is sent to the DHCP server to extend the lease. Normally we get an answer and refresh our information, but if no reply is received we wait for half the remaining time until the rebind time and keep retrying and halving the remaining time. When the rebind time is reached the DHCP REQUEST is broadcast to try and reach some other DHCP server. Halving the remaining time again and again until the lease expires. At that point we go back to square one and broadcast a DHCP DISCOVER.

If at any point a DHCP NAK is received we start over completely. After a DHCP OFFER an ARP request is transmitted just before the DHCP REQUEST to check if the address offered is already in use. If an ARP reply is received before the DHCP ACK then after the ACK we send a DHCP DECLINE to the server to tell that the address isn't what we want and again we start over.

Router Discovery

The gateway offered by the DHCP server is made known to the TCP/IP server by sending an ICMP router advertisement to the local interface with a short lifetime and a low priority. Then up to three router solicitations are broadcast three seconds apart to look for a router. If a router answers with a router advertisement then we no longer worry about routing for that interface. Otherwise the router information is refreshed before it expires and another solicitation is sent out. This happens about twice an hour.

Server Operation

Once all networks so marked are configured the daemon starts answering requests by other machines or relaying requests to other DHCP servers. DHCP requests are answered if information for a client can be found in the configuration file, or if a free address can be found in the pool file, or if a client rerequests an address it already owns.

If the daemon is both a server and a relay for a network then it will try to answer a request and only relay if it has no answer.

Nothing more to do?

If the daemon finds out that all networks have an infinite lease (configured with a fixed address), there is no router information to keep warm, and it isn't a server then it simply exits.

Asynchronous I/O?

Minix doesn't have the asynchronous I/O that Minix-vmd has, so under Minix the daemon only works with one network at a time. If it's stuck on the same network for 32 seconds then that network is closed and another network is tried for 32 seconds. This usually works ok as a client, but as a server it can only handle one network.

OPTIONS

−q Read and print the cache and pool file contents, showing DHCP information for each network, and the IP addresses in the pool with lease times and current/last owners of those addresses.

−a Add the named hosts (or IP addresses) to the pool file.

−r Remove hosts from the pool file.

[**−t**[*level*]]
Set the test level (by default 1). At test level 1 all networks are seen as unconfigured, will not be configured and no data will be put in the cache. The program will just act as-if. At test level 2 the interfaces will not be configured from the configuration file, the data must come from a remote server. At level 3 the renewal, rebind and lease time will be 60, 120 and 180 seconds. At level 4 these times will be 60, 60, and 120. At level 5 these times will be 60, 60, and 60. These test levels are meant to debug the DHCP client code, and are best used with a high debug level.

[**−d**[*level*]]
Set the debug level (by default 1). At debug level 1 the program shows Ethernet and IP addresses as they are determined or configured, DHCP messages sent and received with little detail (one line per message), and memory use. At debug level 2 each DHCP packet is decoded and shown in detail. At debug level 3 device opens and closes are shown. The debugging level may also be increased by 1 at runtime by sending signal **SIGUSR1** or turned off (set to 0) with **SIGUSR2**.

−f *configfile*
Names the configuration file, by default **/etc/dhcp.conf**.

−c *cachefile*
Names the cache file, by default **/usr/adm/dhcp.cache**.

−p *poolfile*
Names the IP address pool, by default **/usr/adm/dhcp.pool**.

SEE ALSO

RFC-2131, **RFC-1533**, **dhcp.conf**(5), **hosts**(5), **ifconfig**(8), **inet**(8), **boot**(8), **inetd**(8), **nonamed**(8).

DIAGNOSTICS

"'/etc/dhcp.conf', line ..."
The program exits on any configuration file error. You have to correct the error and restart the program.

"No lease set for address ..."
There must be a lease time defined for addresses in the pool. Correct and restart the program.

"###### declines #.#.#.# saying '...'"
A client with the given client identifier (usually 01 followed by the client's Ethernet address) declines an IP address, hopefully with a message telling why. This usually means that the IP address is already in use by another host. This program, acting as a client, will tell what other host in its message, but Windows has no additional info alas.

"Got a NAK from #.#.#.# [through #.#.#.#] saying '...'"
The server with the given IP address doesn't want us to have or keep the IP address we were offered or are rerequesting. This could mean that the server has forgotten about us and has given our address to another machine. This is bad if our lease hasn't yet expired. There may be a relay involved, and there may even be a text message with precise information.

"#.#.#.# offered by #.#.#.# is already in use by #:#:#:#:#:#"

We got an ARP reply for an offered address. We won't accept it, and send out a DECLINE when we get an ACK.

"DHCP packet too big, ..."

You've got way to much information in the configuration file, more than fits in a minimum size DHCP packet. (Notify the author if you really need to send more information. He doesn't think anyone needs to.)

"Pool table is corrupt"

You will have to remove and refill the pool file. Chaos may ensue if there are active clients and they don't use ARP to detect each other. (Most do.)

BUGS

There is no randomization of timers. Modern systems don't blink under the load of several clients broadcasting a few packets in sync.

There is no extra time spent waiting for an ARP reply. It is assumed that any IP stack will immediately respond, so that the DHCP server can't possibly beat it at sending out an ACK. (The DHCP server has to commit the lease to stable storage first anyway.)

Way more nonsense can be sent in a DHCP packet that Minix could do something with, but nobody does so we don't bother.

DHCP was invented by a rabid gerbil on speed.

AUTHOR

Kees J. Bot <kjb@cs.vu.nl>