

NAME

dhcpmemu – Emulate multiple DHCP clients

SYNOPSIS

```
dhcpmemu [--listen] [--oneshot] [--promiscuous] [--update] [--withclid] [--withopt82] [--with-
tftp] [--withtod] [--relay-unicast]
        [--bufsize=size-in-K]
        [--giaddr=ip-address]
[--istate=state]
        [--mode=batch|uniform]
        [--number=number]
        [--outfile=filename]
        [--rate=number-per-second]
        [--srcport=portnum]
        [--server=ip-address]
        [--timeout=t1,t2...]
```

DESCRIPTION

dhcpmemu is a multithreaded DHCP multi-client emulator, whose primary purpose is to test the response of DHCP servers under conditions of high load. A pre-compiled input file (default: `./dhcpmemu.ld`) specifies the particulars of the all the clients to be emulated. A thread is created for each client, and these per-client threads conduct the send-receive packet exchanges, waiting for responses, timing out and resending when none is forthcoming, and changing to the client state implied by the DHCP protocol. The most common conversation with a server is DISCOVER - OFFER - REQUEST - ACK but several command line options modify these state-transitions. In particular the conversation may be cut short, or the pseudo-client may begin in some different state.

Each send-reply exchange is timed, and the elapsed time recorded (to `dhcpmemu.tdat` by default). Depending on the parameters specified, and/or the DHCP server configuration, no reply may in fact be forthcoming. That fact too is recorded.

OPTIONS

- `--bufsize=size` The size (kbytes) of the send/receive buffers used by **dhcpmemu**'s socket. The default chosen by the system is often so small that packets are lost due to buffer overflow.
- `-g giaddr --giaddr=giaddr` Set the value of *giaddr* in the outgoing packet. Since each of the pseudo-clients has a phony MAC address, the server response to that address will not be heard by **dhcpmemu**. Setting *giaddr* tricks the server into unicasting to what it believes is a DHCP relay agent. The address chosen should be an IP unicast address of one of the local interfaces, reachable from the server.
- `--istate=state` One of **dhcp_init**, **dhcp_init_reboot**, **dhcp_renewing**, **dhcp_rebinding**, **dhcp_releasing**, **dhcp_informing**. This value overrides the value in the input file. If the IP address in the input file is not correct, the server's response will be a DHCPNAK.
- `-l --listen` Tells **dhcpmemu** to listen for responses. When this is off there can be no meaningful timing measurements. It implies that the emulator is being used to generate noise traffic.
- `-m mode --mode=mode` Either **batch** or **uniform**. In batch mode **dhcpmemu** generates bursts of packets, each containing *rate* packets, with a one second delay between them. In uniform

mode packets are generated uniformly in time, with a $1/rate$ second interval between each. The default is **batch**

- n number** **--number=number**
Emulate *number* different clients. This value is limited by the population contained in the input file.
- oneshot**
Don't go through the complete DHCP packet exchange cycle. Merely wait for the first reply and then quit the thread. Backoff and retransmission are still performed. This option isolates specific transactions of the DHCP state diagram.
- promiscuous**
Listen for all BOOTREPLY port67 traffic. This is useful when running the emulator with real IP assignments. The routing must be such that all such traffic is directed at the box hosting emulator, and that host should **not** be forwarding traffic.
- o filename** **--outfile=filename**
Write time measurements to *filename* (default `./dhcmmemu.tdat`).
- r persec** **--rate=persec**
Rate of client packets per second (default=10). See NOTES regarding high data rates.
- srcport=portnum**
Set the UDP source port number to be *portnum* (default =67 if *giaddr* non zero, =68 otherwise). Normally **dhcmmemu** sends and receives on the same socket. Setting this value has the effect of creating two different sockets, one for sending, the other for receiving. This may permit greater throughput. Be aware that certain server implementations, as a security measure, will discard packets that do not have the proper source port number.
- s server-address** **--server=server-address**
Send packets to the IP address specified. You must set this value unless the server is on the same network as the emulator, or a DHCP relay agent exists on that network. If unset **dhcmmemu** will broadcast packets to 255.255.255.255 but which interface(s) transmit those packets is undetermined.
- t t1,t2...** **--timeouts=t1,t2...**
Specifies the list of timeout values (in seconds) for retransmission of packets when no response is heard. Each per-client thread sends a packet and waits an interval given by the first value in the list. If no response has been received the thread retransmits the packet (but with a different *xid*) and waits for the next interval...and so on. If no response has been received after the last interval failure is recorded. (Default: 2,4,8,16).
- u** **--update**
Update the input file with the IP address obtained from the DHCP server
- withclid**
Send the DHCP *client-identifier* (option 61) as well as the MAC address. The input file contains the DHCP *client-identifier* as well as the MAC address.
- withtftp**
If the DHCP server returns the name of a bootfile, and an IP address of a TFTP server whence that file may be downloaded, proceed, after the final DHCPACK, to use SM TFTP to get that file. The content retrieved isn't saved, but the entire file is downloaded (within the same calling thread) and the time for the complete download is recorded just as for DHCP traffic
- withtod**
If the device is a CDM, send a TOD request to the IP address of the server specified as the TFTP server. (This is the normal CDM behavior: DHCP option 4 isn't used by CDMs). The TOD retrieval happens after TFTP.
- withopt82**
Send option 82, the so-called *relay-agent-option*. In the broadband cable modem world customer premis devices (CPD's) sit behind cable modems (CM's) and option 82 typically contains the MAC address of the CM. Each pseudo client in the input file specifies the MAC address of its associated CM and option 82 is

constructed using that value.

--relay-unicast In most networks, DHCP relay agents only intercept DHCP traffic that was broadcast by the client, and only those packets will arrive at the server with the *giaddr* and option 82 values present. But some networks are configured to intercept even unicast DHCP traffic. This options exists to allow emulation of those networks. Note that the presence of option 82 is still governed by the the values of the *giaddr* of the client being emulated, and the **--withopt82** command line option.

NOTES

No auxiliary utilities are included to display the results of the emulation, which are captured in file **dhcp-memu.tdat**. It is expected that this file be imported into a spreadsheet or in otherwise massaged to summarise the results of the emulation.

The file has three fields per packet sent:

| | |
|---------------------|--|
| <i>XID</i> | An identifier of the packet sent, encoded as N *4096+ S *256+ T where N is the ordinal number identifying the client, S is an integer identifying the client <i>state</i> (from the DHCP state diagram 1=INIT 2=INITREBOOT 3=SELECTING 4=REQUESTING 5=BOUND 6=RENEWING 7=REBINDING -- see RFC2031 for details), and T is the retry number for the packet transmission. |
| <i>usecs</i> | The number of microsecs which elapsed before a reply was received, or minus one if none. Note: clock times as recorded by user level processes need not (and usually are not) accurate to the one microsecond Numbers in the many tens to hundreds of microseconds probably are believable. |
| <i>message-type</i> | The message type of the reply from the DHCP server (2=DHCPOFFER, 5=DHCPACK, 6=DHCPNAK) |

BUGS

Vendor options are not supported so a command line option like *vendor-option-symbol=value* will be ignored.

The **cmu** magic cookie implies a different encoding of the non-fixed-fields, but only **rfc1048** encoding is supported.

SEE ALSO

RFC2031, RFC2032.