A Caching DHCP Relay Agent

Patrick H. Fry[*] Joshua W. Knight[†]
fryp@cs.rpi.edu *joshk@us.ibm.com*

Abstract

Dynamic Host Configuration Protocol[2] (DHCP) is an increasingly popular method for providing configuration information to network hosts. In large-scale enterprise and/or geographically dispersed networks it is useful to provide continuous, reliable DHCP service while minimizing the number of DHCP servers.

This paper describes the Caching DHCP Relay Agent (CDRA) which can reduce the workload of a DHCP server and provide DHCP services during server failures. The CDRA requires minimal configuration and management. Three different types of CDRAs are introduced; one of which, the proxying CDRA, was implemented by modifying the existing DHCP relay agent code for the IBM AIX operating system.

1 Introduction

Dynamic Host Configuration Protocol[2] (DHCP) is an increasingly popular method for providing configuration information to network hosts. In large-scale enterprise and/or geographically dispersed networks it is useful to provide continuous, reliable DHCP service while minimizing the number of DHCP servers.

The role of the DHCP relay agent is to forward messages between DHCP servers and clients. DHCP clients typically broadcast their DHCP requests (UDP broadcast). This is necessary because, on bootup, the client usually knows nothing about its IP network. Because IP broadcasts are not forwarded by routers, it is the responsibility of the relay agent to receive client broadcasts from its local subnets and unicast them to the DHCP servers.

A relay agent eliminates the need for a separate DHCP server for each network segment. In theory, one DHCP server could serve a network of any size. In practice, however, this is not feasible for large networks for a variety of reasons including potentially high network traffic, workload on the DHCP server, and reliability/availability of service to the clients.

Section 2 describes the Caching DHCP Relay Agent (CDRA) which can reduce the workload of a DHCP server and provide DHCP services during server failures. The CDRA is stateless and requires minimal configuration and management. Section 3 details the guidelines which must be followed by a CDRA; including procedures needed to keep the CDRA from giving out invalid information to the clients. Invalid information is client configuration data stored on the CDRA which is no longer valid. Section 4 describes the three

[*]Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY 12180-3590, USA
[†]IBM T.J. Watson Research Center, 30 Saw Mill River Road, Hawthorne, NY 10532, USA

different types of CDRAs. The types are distinguished by how they interact with the DHCP clients and servers. Section 5 covers our implementation of a proxying CDRA using DHCP relay agent code for the IBM AIX operating system. Appendix A contains scenarios which result in invalid IP addresses being given out by a CDRA if the guidelines in Section 3 are not followed. Appendix B details some interesting issues which arose during implementation of the proxying CDRA.

2 Caching DHCP Relay Agent

A Caching DHCP Relay Agent (CDRA) can be thought of as a "smart" relay agent. By saving information from prior conversations between clients and servers, the CDRA can answer some client requests directly. When the CDRA receives a client request, it checks its local database. If the database contains sufficient information to answer the request, the CDRA responds to the client. If not, the CDRA forwards the request to its servers as a normal relay agent would. When a server answers the client request, via the relay agent, the CDRA stores the response for future client queries.

2.1 Network Traffic and Server Workload Reduction

Instead of relaying DHCP messages from a client to the central server, the CDRA can answer many messages directly, without having to forward the request to the server; thus keeping network traffic local. Hand-in-hand with the reduction in network traffic is a reduction in server workload. Because the CDRA is handling client queries normally handled by the server, the server has to respond to fewer messages per client. CDRAs help make the concept of an enterprise DHCP server feasible even for very large networks.

2.2 Reliability and Availability

There are three possible ways to address the issues of reliability and availability. These methods are not mutually exclusive in their benefits and can be used in any combination. One method is to run the DHCP services on a high availability system (e.g. IBM S/390). Another option is to use a pool of redundant DHCP servers. If a server fails, other servers are available to take over. Currently, two competing Internet Drafts, [5] and [3], define a communication protocol between a pool of peer DHCP servers.

A CDRA offers a third approach. If a DHCP server fails, the CDRA is capable of providing some services to its clients until the server returns. The CDRA will most often reside on the same network segment as its clients (although it is possible for a CDRA to receive client messages from other relays). If any other segment goes down, communication between CDRA and client will remain possible.

2.3 Automated Lease Renewals

It is difficult for a DHCP server to determine when an IP address is no longer in use by a host. When a DHCP server issues an IP address to a client, the server tells the client the information is only valid for some specified duration (DHCP lease). The server cannot know for sure if the client is no longer using that address until the lease expires. So, if close monitoring of IP address usage is required, the DHCP server must assign short leases. However, these shorter leases require more frequent lease renewals which increase network traffic and server workload.

The CDRA can help allay this problem through the use of *automated lease renewals*. A server grants a long term lease to a client. The CDRA gives this client a short lease and allows the client to renew the shorter lease directly with the CDRA. When a short term lease expires, the CDRA sends a RELEASE message to the server indicating the address is no longer in use.

3 CDRA Design Specifications

As described in Section 2, the CDRA enhances the DHCP relay agent to include additional DHCP server functionality. This section discusses general requirements for all types of CDRAs including some precautionary measures which must be taken to keep the information given to the clients consistent with the information stored on the server.

A CDRA must appear to its clients as either a relay agent or server. In other words, no special interaction should be required of its clients. The CDRA may or may not require changes to the server. For example, the proxying CDRA (Section 4.1) requires no changes to the server, but the address pooling CDRA (Section 4.3) does. The CDRA should also be stateless. No persistent storage should be required, although it may take advantage of this if available. If a CDRA loses its state information (e.g. through a reboot), it must continue to provide reliable service to its clients and servers, albeit a little slower because of loss of cached information.

First, the concept of a “safe network” is described. In a “safe network,” precautionary measures are not required. Second, the necessary precautions for each type of DHCPREQUEST message are given. Third, safeguards for automated lease renewals are detailed. Finally, the benefits and pitfalls of using persistent storage for CDRA recovery are discussed. Appendix A describes scenarios where invalid addresses could be given out by the CDRA if the precautions outlined in this section are not followed.

3.1 Safe Network

From the CDRA's perspective, a “safe network” is a network where the IP addresses provided to clients through the CDRA are provided exclusively by that CDRA. In other words, if an IP address is obtainable through the CDRA, it must the only avenue through which that IP address can be given. If the CDRA fails, no client can obtain an IP address normally given out by that CDRA. In most cases this implies the CDRA is the only relay agent on the network segment which it is managing. For example, consider a geographically distributed environment. If the CDRA is running on a gateway which provides a branch office sole access to the main office then the network is inherently safe.

It is common that a CDRA would be running in a safe network. It would be beneficial to have the ability to disable the precautions listed below in such a network, since they will result in unnecessary delays. For example, a “safe network mode” option could be used in the CDRA's configuration file. Our CDRA implementation has this capability.

3.2 Precautions Before Responding to Client Requests

Of the five types of client DHCP messages[2], only one, the DHCPREQUEST, requires special precautions when being handled by a CDRA. The DHCPREQUEST message is used to request a lease on an IP address or renew and existing lease. This section describes the precautions necessary for each type of DHCPREQUEST.

Appendix A.1 gives examples of situations where multiple clients can be assigned the same IP address if these safeguards are not followed.

- DHCPREQUEST used in response to an OFFER or used in INIT-REBOOT

  In this situation, the client is requesting a lease on an IP address. If the CDRA's response would be an ACK, the CDRA *should* first issue an *ICMP echo request*, a standard method for a network host to determine of an IP address is being used on its network, on the requested IP address. If a host replies, the CDRA *should* respond with a NAK and remove the address from its database.

- DHCPREQUEST used to RENEW a lease (unicast request)

  This message is used by a client to renew an existing lease. This type of message is unicast. No precautions are required.

- DHCPREQUEST used to REBIND a lease (broadcast request)

  This message is used by a client to renew an existing lease. It differs from the RENEW in that it is broadcast. If a client is trying to REBIND an address, it was unsuccessful in reaching its server (or CDRA) during the RENEW stage. If, according to the CDRA's client database, the CDRA was supposed to be managing this address, the client probably lost contact with the CDRA. So, before the CDRA ACKs the request, it *must* first renew the lease with the server.

3.3 Automated Lease Renewals

Before sending a DHCPRELEASE message to the central server, the CDRA should make sure the IP address is not being used by some client which may have obtained the address through other means than the CDRA. In most cases, an *ICMP echo request*, a standard method for a network host to determine if an IP address is being used on its network, on the address in question will suffice.

3.4 Using Persistent Storage on a CDRA

Persistent storage on a CDRA helps to maintain some of the efficiencies of the CDRA in the event the CDRA is forced to reboot. On startup, the CDRA can load the database containing DHCP client information and continue to provide service to those clients without having to interact with the DHCP server. For example, consider a power failure affecting DHCP clients, the server and the CDRA. Once the CDRA is back up and has reloaded its previous configuration from persistent store, the CDRA can provide addresses to clients even if DHCP server is unavailable.

When a CDRA starts up and reinitializes by reading the database in its persistent store, it is possible that the managed clients could have interacted with the DHCP server through some other means while the CDRA was unavailable. Therefore information in the client database is suspect. Fortunately, as long as precautions listed in Sections 3.2 and 3.3 are taken, no additional measures are required.

3.5 Intermediate Relay Agent

It is possible for a client request to be forwarded by multiple relay agents before reaching a server. The value in `giaddr` contains the IP address of the interface of the *first* relay which directly received the message from the client. The server uses this address to determine the subnet on which the client resides. The server unicasts a response directly to the IP address listed in `giaddr`. So, none of the intermediate relays are involved in the server response. This makes it impossible for the CDRA to perform any caching for this type of client.

3.6 Security

Currently, there is no IETF (Internet Engineering Task Force) standard for encryption or authentication in DHCP. There are two Internet drafts, [4] and [6], addressing this issue. Since CDRA must read and modify DHCP packets, the CDRA must take an active role in any security implementation. No consideration has yet been given to integrating the CDRA in a secure DHCP environment.

4 Types of CDRAs

This section describes three different types of CDRAs: the proxying CDRA, the spoofing CDRA, and the address pooling CDRA.

4.1 Proxying CDRA

The proxying CDRA identifies itself to the client as a DHCP server. So, even though it's acting as a relay agent in that it forwards client requests to the server, the proxying CDRA modifies the server's response so it looks to the client like the proxying CDRA is the server. The proxying CDRA has the advantage that it works with standard DHCP servers. The server sees the proxying CDRA as a relay agent. However, for the proxying CDRA to provide automated lease renewals (Section 2.3) effectively, it may be useful to identify itself to the server. An implementation of a proxying CDRA is described in Section 5.

4.2 Spoofing CDRA

Unlike the proxying CDRA, the spoofing CDRA does not identify itself as the server to clients. Instead, it *spoofs* server messages. Also, unlike the proxying CDRA, there are special network requirements required for the spoofing CDRA. It must intercept packets which are addressed to another host. For example, lease renewal requests are commonly unicast to the DHCP server. The spoofing CDRA must read these packets, otherwise the spoofing CDRA will not know the current state of the client's lease. It would also be beneficial for the spoofing CDRA to have the ability to prevent the unicast packet from reaching its destination (a necessary function for automated lease renewals). Based on these requirements, a spoofing CDRA would work most effectively running on a network router.

There are some benefits to using a spoofing CDRA over a proxying CDRA. With a proxying CDRA, only one OFFER can be forwarded from the relay to the client. The relay is responsible for determining what the best OFFER is. A spoofing CDRA can forward all received OFFERs to the client and let the client decide which to accept. Also, if the spoofing CDRA fails and there's an alternative path from a client to the server, the client can still renew its lease. If a proxying CDRA fails, none of its clients will renew their lease until entering the REBIND [2] stage because they think their server is the proxying CDRA.

4.3 Address Pooling CDRA

The address pooling CDRA acts more like a server than a relay agent. On startup, a pool of IP addresses are requested from the *real* DHCP server. Once the addresses have been acquired, the Address Pooling CDRA *is* a DHCP server. It gives leased IP addresses to clients without consulting the Central DHCP server. The main difference between this CDRA and a DHCP server is statelessness. No recovery file containing the previous state is required on a restart. Of course, a recovery file would be useful if available. Obtaining a pool of IP addresses from a DHCP server is not part of the standard. Therefore, a communication protocol for requesting and maintaining a pool of IP addresses must be developed.

5 An Implementation of the Proxying CDRA

For this section, the word “relay” is used to indicate the implemented proxying CDRA unless otherwise noted.

To test feasibility, we implemented a proxying CDRA. The existing source code for a normal relay agent was modified to create the proxying CDRA. When the original relay agent received a DHCP message, it determined whether it was a BOOTREQUEST (from a client) or a BOOTREPLY (from a server) and forwarded it accordingly. Table 1 shows what the relay does when a message is received. The relay keeps a database containing information about clients from which it has received messages. The database contains the following information :

- `ClientID`: Unique identifier based on NIC information (hardware address and type of interface)
- `ClientType`: DHCP or BOOTP
- `State`: State of client. Based on the possible states a client can be in as described in [2]
- `ServAddr`: IP address of the client’s server
- `CliAddr`: IP address of the client
- `LeaseTime`: Length of lease
- `LeaseStartTime`: Time (local to relay) when the lease started
- `LastContact`: Last time the relay received a message from the client
- `LastUpdate`: Last time configuration information was refreshed by the server
- `Msg`: Last DHCPACK (DHCP client) or BOOTREPLY (BOOTP client) message received from server destined for this client. This message contains network configuration about the client.

Currently, network configuration information is stored on a client by client basis. This may seem a bit excessive for storing data such as subnet masks, DNS server address, or IP gateway address which could be stored on a subnet by subnet basis, but DHCP servers are allowed to give out different information to each client. So, to keep the proxy relay as flexible as possible, a configuration is stored for each client. A copy of the last DHCPACK or BOOTREPLY is kept. This message contains all the necessary data for responding

DHCP Packet	*Action*
DHCPDISCOVER	Proxy if able, otherwise forward to server
DHCPREQUEST	Proxy if able, otherwise forward to server
DHCPINFORM	Forward to Server
DHCPDECLINE	Forward to Server Remove client from DB
DHCPRELEASE	Forward to server Remove client from DB
DHCPOFFER	Forward to client if client is in INIT state, otherwise silent discard
DHCPACK	Forward to client if client is in REQUESTING, REBOOTING, RENEWING, or REBINDING state , otherwise, silent discard
DHCPNAK	Forward to client if client is in REQUESTING, REBOOTING, RENEWING, or REBINDING state, otherwise, silent discard
BOOTP Packet	
BOOTREQUEST	Spoof if able, otherwise forward to server
BOOTREPLY	Forward to client

Table 1: Proxy CDRA actions categorized by message type

and makes creating the response easy. When the relay is directly responding to a request, it makes a copy of the stored message, modifies a few of the fields, and forwards it to the client. Which fields need to be updated are covered later in this section.

Table 1 indicates actions taken by the relay based on type of message received. The relay, like the server sits and waits to receive a message. The only exception is automated DHCPRELEASE messages created when a client's short term lease expires when using automated lease renewals.

The relay spoofs BOOTP clients. Proxying a BOOTP server for BOOTP clients is unnecessary. Once a BOOTP client has received its configuration, it no longer communicates with the BOOTP server. Recall that the primary reason for proxying rather than spoofing DHCP messages was to have the DHCP client unicast messages (e.g. lease renewals and RELEASE messages) sent to the relay instead of the server.

Appendix B lists some issues which arose during the implementation.

5.1 Message Data Modification Before Forwarding

Because the relay is acting as a proxy server, it must sometimes change the data in DHCP messages before forwarding to the client or server. For example, when a server sends a packet to a client, the server includes its IP address in the `siaddr` field. The relay must change this entry to its own IP address before forwarding to the client. If the relay is performing automated lease renewals, then the configuration information relative to the leases (ie. IP Address Lease Time, Renewal (T1) Time Value, and Rebinding (T2) Time Value [1]) must also be modified before forwarding to the client. Finally, when a client sends a DHCPREQUEST in response to a DHCPOFFER, it must include the Server Identifier option. This option will contain the address of the relay. If the relay needs to forward this request to the server, it must first change the contents of that option to hold the address of the server which made the offer.

5.2 State of a Client

The relay stores the current state for each client in the variable `State`. The different possible states are roughly synonymous to those shown in Figure 5 of [2] with some shoehorning for BOOTP clients. The relay determines the client's state based on the last message to/from the client. For example, suppose the client sent a lease renewal REQUEST to the relay. If the relay is able to reply to the client, it will place the client in the BOUND state (the relay assumes the client receives the reply). If the relay must forward the request to the client's server, it places the client in the RENEWING state.

Storing the state is necessary for a variety of reasons. The most important reason is for OFFERs from multiple servers. Once the relay has sent an OFFER to a client, the relay places the client in the REQUESTING state. Any more OFFERs received will be silently discarded. This prevents the client from receiving multiple OFFERS from the relay.

5.3 Automated Lease Renewals

The current implementation can perform automated lease renewals as described in Section 2.3. In the relay's configuration file, an administrator can specify the duration of leases which the relay should issue to the clients. A relay will renew these short term leases on the client as long as there is still enough time left on the lease obtained from the server. When the relay receives a lease renewal request from a client, it checks the time remaining on the lease from the server. If the lease has not yet passed the T1 stage, the relay issues a new short term lease to the client. If it's past T1, the relay will still issue a new short term lease to the client (assuming there is enough time on the long term lease to cover it) and it will also forward the lease request to the client's server or all known servers depending on whether the lease has passed T2. When the server sends the ACK back to the relay, the relay updates the lease information in the client DB and silently discards the packet. If a lease renewal request is received which the relay is not able to proxy, the request is forwarded to the client's server or all servers (again, depending on whether the long term lease has passed T2). The relay does not respond to the client until a response is received from the server.

5.4 Persistent Storage

Persistent storage can be used to provide better service after a transient failure of the CDRA. At frequent intervals (by default every 70 seconds), the entire client DB is dumped to disk. Another option would be to write the DB to disk after every packet forwarded to client and/or server. When a CDRA is started up, it will try to reload its previous client DB from disk. This feature is most useful for clients in a remote location where it may be difficult to restore connectivity back to the main campus or an environment where if the server fails, it can take a long time to come back up. With persistent storage on the CDRA, as soon as the CDRA is back up, the clients can retrieve their network configurations.

5.5 Unsafe vs. Safe Networks

The relay is capable of running in safe or unsafe network mode. When running in unsafe network mode, all precautions listed in Section 3 are followed.

5.6 Keeping the Client Data Current

To keep the client data reasonably "fresh", `LastUpdated` is stored for each client. This time value indicates how old a client's network configuration information is. The data is only considered valid for a certain term (default is 9 days). After expiration of the data, the client record is deleted, thus forcing the relay to forward the next client query to the server.

5.7 Possible Extensions, Improvements

In the current implementation, when a client's lease expires, either via a client or relay initiated RELEASE message or through the normal lease timeout, the client entry is removed from the DB. There are two reasons why even this expired data might be useful. Both rely on the principle that the client will most likely be given the same configuration the next time it's requested. Most clients start their initialization sequence by issuing a DHCPDISCOVER. The relay could respond to this message with a DHCPOFFER even though the lease has expired. The client would respond with a DHCPREQUEST of the REQUESTING type. The relay could convert this REQUESTING DHCPREQUEST to a REBOOTING DHCPREQUEST and forward it to the server. This saves the server from having to process the DHCPDISCOVER. The worst case scenario is that the server respond with a NAK and the client has to reissue a DISCOVER. See [2] for more details about different methods of client initialization.

Another reason for keeping the expired data has to do with temporary loss of connectivity with the server. If the server is unreachable, the relay could give the client its old expired IP address if requested. This is dangerous in that it is possible the server will give the same address to some other client. DHCP servers often perform DNS updates. If the relay is giving out addresses which it is temporarily unable to confirm with the server, then it should also update the DNS. As mentioned previously, the client DB contains an ACK for each client. Most of this data would be redundant for its clients. A hierarchical database of configuration, organized by server, then by subnet, then by client, could be used for more efficient data storage.

6 Summary and Conclusions

The Caching DHCP Relay Agent (CDRA) is a type of "smart" relay agent. As it relays DHCP messages between clients and servers, it stores information about a client's configuration. In subsequent client requests, the CDRA will attempt to reply to the client without having to forward the message to the server. A CDRA helps to reduce network traffic by keeping DHCP transactions on the client's local network segment. Consequently, the amount of work a DHCP server must do per client is reduced. This helps enable the possibility of an enterprise wide DHCP server even for large corporate networks. Another nice feature of the CDRA is its "plug and play" capability. There is no extra configuration required over that of a normal relay agent. A DHCP server, in comparison, is not stateless.

In a network centric computing environment there are many services which follow the same model as the network configuration data obtained using DHCP. Their data is "read-mostly" with very few changes over time. Some examples include bootup files (OS kernel, runtime libraries, etc.), executable files, and web documents. Perhaps the principles applied here for DHCP could be applied to these other services.

References

[1] Alexander, S., and R. Droms, "DHCP Options and BOOTP Vendor Extensions", RFC 2132, Silicon Graphics, Inc. and Bucknell University, March 1997.

[2] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, Bucknell University, March 1997.

[3] Droms, R., and K. Kinnear, "An Inter-server Protocol for DHCP", Internet Draft draft-ietf-dhc-interserver-alt-00, Bucknell University and American Internet Corporation, April 1997.

[4] Gudmundsson, O., "Security Architecture for DHCP", Internet Draft draft-ietf-dhc-security-arch-01, Trusted Information Systems, July 1997.

[5] Kinnear, K, R. Cole, and R. Droms, "An Inter-server Protocol for DHCP", Internet Draft draft-ietf-dhc-interserver-02, American Internet Corporation, AT&T MNS, and Bucknell University, July 1997.

[6] Patel, B., "Securing DHCP", Internet Draft draft-ietf-dhc-securing-dhc-00, Intel Corporation, July 1997.

A IP Address Assignment Errors

This appendix outlines situations where two clients could end up being assigned the same IP address. These situations can be avoided by implementing the precautions described in Section 3.

A.1 Responding to DHCPREQUEST

- DHCPREQUEST used in response to an OFFER or used in INIT-REBOOT

    1. Client A obtains a leased IP address from CDRA
    2. Client A shuts down without issuing release
    3. Client A starts up and obtains a lease on the same IP address through some other source.
    4. Client A gracefully shuts down with a DHCPRELEASE message
    5. Client B starts up and obtains a lease on A's old IP address unbeknownst to the CDRA.
    6. Client A starts up an obtains a lease from the CDRA on its old address. (CDRA should have issued ICMP echo request first)
    7. Client A and B are using the same IP address

- DHCPREQUEST used to REBIND a lease (broadcast request)

    1. Client A obtains a leased IP address from the CDRA. The CDRA gets a 1 week lease from the server on the address, but only gives the client a 1 hour lease.
    2. Client A shuts down.
    3. Client A starts up and obtains a 1 hour lease directly from the DHCP server. This replaces the 1 week lease the server previously gave through the CDRA.

4. Client A attempts to renew the lease, but the server does not respond.
5. Client A broadcasts the REBIND request. The CDRA responds with another 1 hour lease. (CDRA should have renewed lease with server first)
6. Client A's lease on the server expires.
7. Client B starts up and obtains a lease on client A's address from the server.
8. Client A and B are now using the same IP address.

A.2 Automated Lease Renewals

Observe the timeline below. `CDRA Down` indicates a transient failure of the CDRA from the perspective of the client. At `CDRA Up`, connectivity has been restored.

```
Lease  CDRA    Lease          CDRA
Start  Down   Renewal   *      Up    XXX
--+------+-------+------+-------+------+--->
```

At `*`, the client was unable to reach the CDRA to renew its lease, so a DHCPREQUEST broadcast is sent out. Somehow, this renewal request reaches the server. Two relevant cases are:

i. Client is given a new lease with old IP address

ii. Client is given a new IP address

At `XXX`, the lease given to the client by the CDRA expires. The CDRA would normally send a DHCPRELEASE to the server. With case (ii), that's fine, but with case (i), it's not. So, the CDRA *should* first send out an ICMP echo request, before sending the DHCPRELEASE message.

A.3 Persistent Storage

1. Client A requests IP address from Proxy CDRA
2. CDRA grants a short lease to client A
3. ****** CDRA fails and must be restarted ******
4. Client A reboots and issues a DHCPDISCOVER
5. DHCP server receives client A's discover message
6. Server grants a lease to client A on same IP address as CDRA had given.
7. Client A gracefully shuts down and issues a DHCPRELEASE to the server
8. Client B requests an IP address and receives client A's old address from the server
9. ****** CDRA is now available again with configuration from persistent storage ******

10. Client A starts up again and request an IP address
11. CDRA gives client A its old address (CDRA should have issued an ICMP echo request first)
12. Client A and B are now using the same IP address

B Complications in the Implementation of the Proxying CDRA

Some issues arose during the implementation of the proxying CDRA. For some, a fix was put in place (not necessarily the best fix) and for others the problem still exists.

- New address REQUEST vs. lease renewal REQUEST

  The relay saves the latest ACK for each client, whether it's an ACK in response to a new address request or for a lease renewal. The relay uses this ACK to respond to both types of REQUESTs and also to DISCOVERs (by converting it to an OFFER). This will work if the client follows the recommendation stated in the RFC [2] that a client *should* always include the same options in its DISCOVERs and REQUESTs. However, if a client doesn't follow this guideline, the results may be unreliable.

- Client-identifier option

  There exists a DHCP option called client-identifier [1]. A client can use this field to provide a unique identifier as opposed to using its network interface hardware address. Currently, the relay ignores this field. While the server will use this as the identifier for the client if it's present, the server does not include it in any of its messages. So, the relay would need to store both hardware address and client-identifier, to be able identify which client a server message was destined for. The relay should store the client-identifier if it's included in the client's message and use the client-identifier, hardware address pair as the client's key in the DB.

- What happens if the client request changes?

  The relay doesn't examine a client's message other than identifying what type of DHCP message it is. If the client were requesting some new information, the relay wouldn't know it.

- The server-identifier option

  The server-identifier is included in some client requests. The clients think of the relay as their server, so clients will use the relay's address in the server-identifier field. Before forwarding this message to the server, the relay must change this field to the real server's address. For each client, `ServAddr` is stored for this reason. It is possible for the relay to lose this information under certain conditions (eg. reboot). Currently, if the relay does not know the correct address, it sends a NAK to the client, forcing it to restart its initialization and send out a DISCOVER which the relay can forward to all its servers. Another option might be to remove that field from the packet before forwarding to the servers.

- Keeping the information current

  The network configuration information which the relay is storing should not change very often. However, the relay has no way of knowing when it does change. The `LastUpdate` field of the relay's client records puts a date on the configuration information. Section 5.6 describes how the timestamp is used.

- Choosing the optimal OFFER

  It is possible for a client to get more than one OFFER when it issues a DISCOVER. The relay can only forward one OFFER to a client. Otherwise it would appear to the client as if the same server had given multiple OFFERs. So, the relay must choose. Currently, the relay chooses using FCFS. Other options might be longest offered lease, or load balancing clients between servers.

- Validity of client data

  The relay does not check the IP address it's offering the client to see if it's valid on the client's subnet. So, if a client moves to a different subnet being serviced by the same relay, it's possible for the relay to offer an invalid IP address.