Network Working Group K. Kinnear
INTERNET DRAFT American Internet Corporation
R. Cole
AT&T MNS
R. Droms
Bucknell University
July 1997
Expires January 1998

An Inter-server Protocol for DHCP
<draft-ietf-dhc-interserver-02.txt>

Status of this Memo

Abstract

The DHCP protocol is designed to allow for multiple DHCP servers, so that reliability of DHCP service can be improved through the use of redundant servers. To provide redundant service, all of the DHCP servers must be configured with the same information about assigned IP addresses and parameters; i.e., all of the servers must be configured with the same bindings. Because DHCP servers may dynamically assign new addresses or configuration parameters, or extend the lease on an existing address assignment, the bindings on some servers may become out of date. The DHCP inter-server protocol provides an automatic mechanism for synchronization of the bindings stored on a set of cooperating DHCP servers.

This draft is a direct extension of draft-ietf-dhc-interserver-00.txt, and represents the merging of ideas from both

draft-ietf-dhc-interserver-alt-00.txt and draft-ietf-dhc-interserver-01.txt. The basic protocol semantics from draft-ietf-dhc-interserver-alt-00.txt were used with the underlying message mapping to SCSP from draft-ietf-dhc-interserver-01.txt. Considerable additional work has been included in this current draft in the area of protocol correctness, detailed work on mapping the protocol to SCSP, and organization of the draft itself.

1. Introduction

DHCP servers manage the assignment of IP address and configuration parameters to IP hosts. The DHCP protocol specification [1] refers to the collection of configuration information assigned to a client as a "binding". The DHCP protocol is designed to allow for multiple DHCP servers, so that reliability of DHCP service can be improved through the use of redundant servers. To provide redundant service, all of the DHCP servers must be configured with the same information about assigned IP addresses and parameters; i.e., all of the servers must be configured with the same bindings. Because DHCP servers may dynamically assign new addresses or configuration parameters, or extend the lease on an existing address assignment, the bindings on some servers may become out of date.

The DHCP inter-server protocol provides an automatic mechanism for synchronization of the bindings stored on a set of cooperating DHCP servers.

The remainder of this document is organized in the following sections:

2. Goals and Requirements

   Defines the requirements and goals for the protocol. Discusses limitations of the protocol. Also contains a definition of several classes of failures as well as a list of specific failures (which provide a useful common ground for discussion).

3. Overview

   Discusses in a general way the content of the information communicated between servers implementing this protocol as well as the way that information is communicated.

   Introduces the three aspects of the protocol: client binding management, address management, and group management.

Defines some key concepts surrounding the allowable "states" of an IP address, including extensions critical to the operation of this protocol.

Gives a brief sketch of the actions required by this protocol for each DHCP client request received by the server.

4. Client Binding Management

Discusses the fundamental messages used by this portion of the protocol, and the ways in which these messages are combined to form higher level operations. Required responses to incoming client binding management requests are explained in this section. The required responses to incoming DHCP client requests are explained in Section 6 below.

5. Address Management

The fundamental messages used by the address management portion of the protocol are explained, as well as how they are combined into higher level operations. The required responses to incoming address management requests are explained in this section, while the required responses to incoming DHCP client requests are explained in Section 6 below.

6. Actions in Response to DHCP Client Messages and Events

The required responses to incoming DHCP client messages and events are discussed in this section.

7. Group Management

The fundamental messages and their combination into higher level operations for the group management portion of the protocol are explained. The actions to take when receiving any of these messages as well as how to utilize them to join or leave a server group are explained.

8. SCSP Message Mapping

The messages described in sections 4, 5, and 7 are mapped into underlying SCSP messages in this section. This includes detailed information on the format of each SCSP message.

9. IP Address State Transition

This protocol expands the possible states for an IP address. The new states are described in Section 3.3. This section

describes all of the transitions between states in detail.

10. Security

The security implications of this draft are discussed in this section.

11. Open Questions

Poses open questions about the protocol. Some questions from draft-ietf-dhc-interserver-00.txt are included verbatim with answers and questions (and some answers) new to this draft are included as well.

12. Acknowledgments

13. References

14. Author's Information

A. Appendix A: An Overview of SCSP

1.1. The Language of Requirements

Throughout this document, the words that are used to define the significance of particular requirements are capitalized. These words are:

o "MUST"

This word or the adjective "REQUIRED" means that the item is an absolute requirement of this specification.

o "MUST NOT"

This phrase means that the item is an absolute prohibition of this specification.

o "SHOULD"

This word or the adjective "RECOMMENDED" means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications should be understood and the case carefully weighed before choosing a different course.

o "SHOULD NOT"

This phrase means that there may exist valid reasons in particular circumstances when the listed behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

- "MAY"

  This word or the adjective "OPTIONAL" means that this item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because it enhances the product, for example; another vendor may omit the same item.

1.2. Terminology

This document uses the following terms:

- "DHCP client"

  A DHCP client is an Internet host using DHCP to obtain configuration parameters such as a network address.

- "client"

  Whenever the term client is used in this draft, it refers to a DHCP client (and not a server communicating with another server using this protocol).

- "DHCP server"

  A DHCP server is an Internet host that returns configuration parameters to DHCP clients.

- "binding"

  A binding is a collection of configuration parameters, including at least an IP address, associated with or "bound to" a DHCP client. Bindings are managed by DHCP servers.

- "active server"

  An active server is one which is capable of offering IP addresses to clients.

- "stable storage"

Every DHCP server is assumed to have some form of what is called "stable storage". Stable storage is used to hold information concerning IP address bindings (among other things) so that this information is not lost in the event of a server failure which requires restart of the server.

2. Goals and Requirements

There are several levels of goals for this protocol. There are a set of requirements with which it must comply, and then there are a set of goals for the protocol and the way that it is used that are listed in priority order.

2.1. Requirements on this Protocol

The following list of requirements must be (and are) achieved by this protocol.

1. Implementations of this protocol work with existing DHCP client implementations based on the DHCP protocol [1]. It must work with today's clients!

2. Implementation works with existing BOOTP relay implementations.

3. Can be specified with sufficient clarity that unique implementa-tions will work well together the first time (e.g. DHCP today largely meets this requirement).

4. Work well with minimum of two and a maximum of 16 servers.

2.2. Goals of this Protocol

The following are the goals of this protocol. These goals are listed in priority order. The protocol meets all of these goals.

1. Avoid binding an IP address to a client while that binding is currently valid for another client. In other words, don't allo-cate the same IP address to two clients.

2. Ensure that an existing client can keep its existing IP address binding if it can communicate with any DHCP server using this protocol -- not just the server that originally offered it the binding.

   DISCUSSION:

There is a subtle but very important point here. For example, assume that there are five servers using this protocol. Everything is running fine, and then the network becomes partitioned, and three servers can communicate among themselves, and the other two can communicate among themselves -- but the set of three cannot communicate with the set of two. Each set, however, can communicate with some clients.

In this situation, every client that can communicate with a DHCP server in either set should be able to continue to use its existing binding, even if the server that originally created the binding is not included in the set of servers with which it can communicate.

3. Do not add any requirement for communication with another server to the processing between a DHCPDISCOVER and a DHCPOFFER or between a DHCPREQUEST and a DHCPACK.

   DISCUSSION:

   This is another subtle point. The implications of this goal are that "lazy" update of IP address binding information is required. In other words, because of this goal, the protocol cannot require one server to update another server with information concerning a new IP address binding prior to sending the DHCPACK to the DHCP client.

   As a result of this goal, a server may fail immediately after sending the DHCPACK to the client but prior to successfully sending a record of that information to any other server. Should this happen, the DHCP client is the only operational machine with a record of this binding -- and the protocol must be (and has been) designed to properly deal with this situation.

3. Ensure that a new client can get an IP address from some server.

4. If a server goes down, and an external agent determines that it is actually down as opposed to running but simply unable to communicate with other servers, then the addresses that it currently owns but are not yet bound may be recovered for use by other servers.

5. Ensure that in the face of partition, where servers continue to run but cannot communicate with each other, the above goals and requirements are met. In addition, when the partition condition is removed, allow graceful automatic re-integration without requiring human intervention.

2.3. Limitations of this Protocol

The following are explicit limitations of this protocol. This is not to say that they are not useful capabilities to have (that's why they are explicitly listed, so that it will be clear that this protocol does not supply them).

1. Determination of permanent server failure.

   The protocol provides a way to propagate information about the permanent failure of a server, but no way to detect a permanent failure. Transient failures are detected, but there is no mechanism in this protocol to determine when a transient failure is really a permanent failure. Some external agent must make this determination -- and must ensure that the server declared permanently failed is not simply partitioned from the other servers and unable to communicate with them. The server which has been declared permanently failed by the external agent MUST be informed of that declaration prior to restart.

   DISCUSSION:

   The existing configuration messages allow one server to declare another server as permanently failed and remove it from the group. That is not the issue. What makes fully automatic determination of permanent server failure impractical is distinguishing between permanent server failure (which is easily defined as transient server failure that has gone on too long) and partition of the group of servers.

   Once communication fails with a server, the other servers cannot know if it is still operating or not, and removing an operating server from the group is an activity fraught with peril.

   This protocol is designed so that a server which is partitioned from the group will re-integrate cleanly when it can communicate again with the rest of the group.

   Group membership protocols typically handle a partition situation (when they bother to handle it at all) by having the partitioned server determine that it has been partitioned and shut itself down. It detects a partition condition in one of two ways: either it can't communicate with the "master", or it can't communicate with the "majority" of the group. In either case, it shuts down.

   We believe that this is not an appropriate response for a

DHCP server. If my DHCP client can talk to a DHCP server, I want my client to continue to operate -- I'm not interested in having the only DHCP server to which I can talk shut itself down!

2. Some addresses are temporarily unavailable during transient server failure.

   The full range of existing IP addresses that are potentially available for allocation is reduced during the period of a transient server failure. The size of the pool of addresses that are available for allocation but not yet allocated SHOULD be configurable for each server. If the server is subsequently declared to have undergone a permanent failure, these addresses will be made available again.

   Note that it is only the addresses not yet allocated but available for allocation that are unusable during the period of a transient server failure. IP addresses that have been allocated to clients may continue to be used by those clients even during server failure. Indeed -- to allow existing clients to be able to renew their existing IP addresses even if the server who granted them the lease has failed is a primary reason why this protocol exists.

2.4. Failures

This section makes explicit both classes of failures as well as a list of specific failure scenarios in order to facilitate discussion of the capabilities of this protocol.

- "transient server failure"

  A transient server failure is one where a server is unable to respond to requests, but later becomes operational and able to respond to requests. Its local stable storage (i.e., whatever mechanism it uses to preserve its binding information) is accurate as of the time that transient server failure began.

- "permanent server failure"

  A permanent server failure is one where a server is unable to respond to requests -- probably for an extended period. While the protocol defined in this document supports declaration of a permanent server failure, the decision that a transient server failure is in reality a permanent server failure is beyond the scope of this protocol.

This determination will be likely be performed by some administrative entity, although in the future a group membership protocol could be integrated with the protocol defined in this document to make such determinations automatically.

o "partition"

A network partition is caused by a failure of the underlying communications substrate, such that two systems that could previously communicate cannot now do so. This may mimic transient server failure, but is not the same because in this case the server that appears to have failed may still be operational and interacting with clients.

There is a form of partition known as "partial partition", where the transitivity of communication usually expected is not achieved. Imagine a set of servers organized (for the purposes of exposition only) as a ring where each server can communicate with its neighbors, but nobody else -- and when the number of servers is greater than three, a partial partition situation exists.

This term may also be used as a noun, as in "each partition may communicate with ...", and in this case it refers to the group of servers which can communicate normally (as distinguished from those with which that group cannot communicate).

o "communication failure"

Communications failure describes the condition where the communication channel between two servers becomes impossible. "Partial communication failure" describes the case where the normally bidirectional communications channel becomes unidirectional, where one server can send to but not receive from another server.

Some examples of the above failures are given below:

1. A single server crashes and reboots. [transient failure]

2. A single server crashes and stays down for a period of hours and then reboots (either automatically or through some external agent). [transient failure]

3. A single server fails and never returns. No permanent failure is declared for this server. [transient failure]

4. A single server fails. A permanent failure is declared for this server. [permanent failure]

5. A group of two servers are partitioned so that they cannot communicate, but each can communicate to some clients. [partition]

6. A group of five servers are partitioned so that three can communicate together and the remaining two can also communicate, but the two partitions cannot communicate. Each partition can communicate with a subset of the clients, and these subsets are disjoint. [partition]

7. A group of five servers are partitioned so that three can communicate together and the remaining two can also communicate, but the two partitions cannot communicate. Each server continues to be able to communicate with all of the clients. [partition]

   DISCUSSION:

      This situation is unlikely to occur, but the protocol should be able to handle it.

8. Server A can send packets to server B, but cannot receive packets from server B. [partial communications failure]

9. There are four servers, A, B, C, and D. A cannot communicate with C, B cannot communicate with D. [partial partition]

DISCUSSION:

This section on failures may well not belong in the final document. For the purposes of review of the rest of the protocol, however, defining a common language to describe failures and giving specific examples of failures as an aid to discussion seemed useful.

3. Overview

At the most basic level, the DHCP protocol specifies the behavior of DHCP servers which communicate with DHCP clients in order to allocate IP address to the clients as well as provide a variety of configuration parameters information to them. It is the allocation of IP addresses to clients by the server that creates a requirement to update what is known as "stable storage" -- typically held on disk. This information is used to "remember" the IP address bindings that have been made by the DHCP server in order to avoid allocating the same IP address to two clients.

The key motivation for an inter-server protocol is the desire to allow a client to continue to use its IP address (i.e., be able to

renew its lease on an IP address) even if the server who initially offered it the lease on its IP address is unavailable for some reason. In addition, no IP address should ever be bound to two clients simultaneously.

Providing multiple DHCP servers to which each client can communicate is the first step in creating this reliable DHCP capability.

In addition, these DHCP servers must communicate among themselves in order to provide this reliable DHCP capability.

3.1. Information Communicated by the Protocol

There are three types of information which must be communicated between servers implementing the server server protocol.

o Client Binding Information

This entire interserver protocol exists in order to allow servers to share information about client bindings of IP addresses. Servers must be able to update other servers about client bindings that they have created, and must be able to receive similar updates from other servers about client bindings that the other servers have made or changed.

o Address Management Information

In order to implement an effective strategy for client binding information updates, this protocol defines some additional states for an IP address beyond those defined or implied by RFC 2131 [1] that are not directly connected with client binding information. The servers need to communicate among themselves concerning these states, and this communication is enabled by the address management information portion of the protocol.

o Group Management Information

While it is possible to conceive of a group of servers statically configured to be part of a server group, the operational characteristics of such an approach are far from pleasant. The group management portion of this protocol allows a server to determine the groups to which another server belongs; determine for each group the current membership in the group; determine for each group the subnets and IP addresses managed by that group; and join or leave a server group.

3.2. Server Groups

Fundamental to this protocol is the "group" of servers which are communicating and with which the clients can communicate in order to provide a reliable DHCP service.

Each server group (SG) to which a server belongs is associated with a particular set of address pools. These address pools are those which exist on a single network segment (sometimes called a single "wire").

An active server can be (and typically would be) a member of several groups simultaneously. This protocol allows a server to join an existing SG. Which SGs a server would join is a configuration issue for a particular server, and outside of the realm of this protocol -- although considerable support is provided in order to make this a solvable problem.

The membership of a particular SG will change over time, and in order to ensure that each server is made aware of any changes in group membership in a timely way, every protocol message which is sent in the inter-server protocol includes a group generation number (with a few exceptions).

Whenever a message is received, the group management layer of the software MUST verify that the group generation number matches the current group generation number for that SG stored in the server. If there is a mismatch, the group management layer will discard the message. It will then attempt to update its knowledge of the current group (and incidentally bring its generation number up to date in the process).

In this way, any changes in group membership become spread throughout the group as fast as possible -- and no messages that are out of synchronization with the latest concept of group membership can be received.

A server attempts to become a member of a particular group by using the configuration messages described in Section 7 below. In addition, a server can remove another server from the group using these messages -- but in this case an external agent must ensure that the server being removed is truly inactive and not just partitioned.

3.3. Messages and Operations Defined by the Protocol

The protocol requires that servers who implement it can communicate, each with the other, in a point-to-point manner (when all are operating correctly). It allows for the possibility that they can fail

entirely (i.e., crash) or be unable to communicate with each other for a variety of reasons.

Each server will periodically need to communicate with other servers in the group. There are several recurring styles of communication that, if defined, will assist in explaining the major concepts of this protocol. These major styles of group communication are as follows:

There are "messages", which for the purpose of this specification consist of a communication between two servers. Messages are gathered into higher level generic "operations", which describe the form of the operation, and are made up of messages communicated between more than one server. These generic operations are then instantiated into specific operations as part of the various portions of the protocol.

3.3.1. Generic Protocol Messages

Messages are used to communicate between a pair of servers.

o QUERY

A QUERY operation is performed when one server wishes to obtain knowledge about the server cache of another server.

o UPDATE

An UPDATE operation is performed when one server wishes to update the information in the cache of another server.

3.3.2. Generic Protocol Operations

These generic protocol operations are used when a server must communicate with more than one other server.

o POLL

A POLL operation is used when one server must contact every other server in the group using a QUERY message in order to request that they respond with some information (typically concerning an IP address). Usually, if the server executing the POLL cannot contact all of the other servers using the QUERY message, it will use whatever information it could glean from those it could contact.

o COMPLETE POLL

A COMPLETE POLL is like a POLL in that one server attempts to contact every other server using a QUERY message -- but in a COMPLETE POLL it must successfully complete a QUERY with each of them or the operation itself fails to complete.

- o PUSH

  A PUSH operation is used when one server wants to update all of the other servers using an UPDATE message. In a way similar to the POLL operation, a PUSH operation will succeed if the server employing it has managed to contact at least one other server in the group with a successful UPDATE.

- o COMPLETE PUSH

  A COMPLETE PUSH is analogous to a COMPLETE POLL -- the COMPLETE PUSH operation requires the server to attempt to UPDATE every other server in the group. If every server responds successfully to the UPDATE, the COMPLETE PUSH succeeds, otherwise the COMPLETE PUSH fails.

Note that both PUSH and POLL involve operations to all of the servers in the group.

3.3.3. **Specific Protocol Operations**

These above generic forms of inter-server communication are utilized in the following ways in the Client Binding and Address Management.

Client Binding Management:

- o CLIENT BINDING POLL (operation)

  This operation involves one server asking every other server using a QUERY for client binding information concerning a particular IP address. If all of the other servers are not operational, the requesting server will use any information it receives.

- o CLIENT BINDING COMPLETE PUSH (operation)

  This operation involves one server informing all of the other servers using an UPDATE about updated client binding information. While there is utility in reaching even one other server (in some cases) the operation is not deemed to have succeeded unless all of the other servers were successfully updated with the new information.

Address Management:

o UNBINDABLE COMPLETE POLL (operation)

In this operation, all of the other servers are contacted using a QUERY concerning one (or more) IP addresses, and they all report on whether that IP address(es) is UNBINDABLE or not. This operation fails if any server fails to respond to the QUERY or if any server responds to the QUERY with a negative answer (i.e., the IP address is not currently UNBINDABLE). It succeeds only when all of the servers in the server group answer that the address is UNBINDABLE.

o TRANSFER (message)

This message is used to transfer BINDABLE IP addresses from one server to another (used when the SG is partitioned and the normal UNBINDABLE COMPLETE POLL cannot be used to make an IP address BINDABLE, but also when all of the UNBINDABLE IP addresses have already been made BINDABLE by some server).

The information is sent from the initiating to the responding server as a QUERY and includes the subnet specification and the number of BINDABLE IP addresses the initiating server has available for that address pool, and the number of BINDABLE IP addresses it is requesting.

The responding server is free to give the initiating server all, some, or none of the number of IP addresses the initiating server has requested.

3.4. IP Address State

The concept of the state of an IP address is largely implicit in the DHCP RFC [1]. However, in order to manage pools of IP addresses with multiple servers, the states and transitions between them must be made quite explicit.

3.4.1. IP Address State: Basic DHCP Protocol

When an IP address is always controlled by a single DHCP server (implicit in the definition of DHCP in the current DHCP draft [1]) the IP address is either in the BINDABLE state or the BOUND state. The following state diagram represents the states that an IP address may occupy based on the current DHCP draft. (Note that these terms do not appear in [1], but are terms that describe concepts that are

implicit in the RFC.)

```
                      +------------------+
                      |                  |
                      |     BINDABLE     |<--+
                      |                  |   |
                      +------------------+   |
                               |             |
                               V             |
                      +------------------+   |
                      |                  |   |
                      |      BOUND       |---+
                      |                  |
                      +------------------+
```

Figure 3.4.1-1: Basic DHCP IP address state transition diagram

When an IP address transitions from BINDABLE to BOUND, that transition must be recorded in the server's stable storage prior to the transition being "published" to any observer outside of the server.

3.4.2. IP Address State: Extensions to Support the Interserver Protocol

The situation is more complex when multiple servers are managing the same set of IP addresses as required by this protocol. Three new states are defined for an IP address: UNBINDABLE, POLLING, PUSHED and EXPIRED.

This is the state diagram for IP address state required by this protocol:

```
                        +-----------------+
                        |                 |
                        |    UNBINDABLE   |<--------+
                        |                 |         |
                        +-----------------+         |
                                 |                  |
                                 V                  |
                        +-----------------+         |
                        |                 |         |
                        |     POLLING     |-------->|
                        |                 |         |
                        +-----------------+         |
                                 |                  |
                                 V                  |
                        +-----------------+         |
                        |                 |         |
                        |    BINDABLE     |-------->|
                        |                 |         |
                        +-----------------+         |
                                 |                  |
                   -----------------------------    |
                                 V                  |
                        +-----------------+         |
                        |                 |         |
                    +-->|     BOUND       |-------->|
                    |   |                 |         |
                    |   +-----------------+         |
                    |      |     |                  |
                    |      |     V                  |
                    |      |  +-----------------+   |
                    |      |  |                 |   |
                    |      |  |     PUSHED      |-->|
                    |      |  |                 |   |
                    |      |  +-----------------+   |
                    |      |     |                  |
                    |      V     V                  |
                    |   +-----------------+         |
                    |   |                 |         |
                    +<--|    EXPIRED      |-------->+
                        |                 |
                        +-----------------+
```

Figure 3.4.2-1: Extended DHCP IP address state transition diagram
                 required for the Inter-server protocol.

For every server which cooperates using this protocol, an IP address is in one of the following six states:

- o UNBINDABLE

  This state represents the default state for every IP address. Explicit action must be taken to move an IP address from this state into the BINDABLE state. An UNBINDABLE COMPLETE POLL must be performed and must complete successfully.

  Any IP address that has previously been BOUND must retain information concerning the server that PUSHED the binding information, the client to which it was bound, and the lease time for the binding. This information is used when a server is removed from the server group.

- o POLLING

  While an UNBINDABLE COMPLETE POLL operation is being performed, an IP address is in the POLLING state. This ensures that if two servers are simultaneously performing an UNBINDABLE COMPLETE POLL operation that involves the same address that neither of them will succeed in making that address BINDABLE.

- o BINDABLE

  In this state, the IP address is available to be offered to a DHCP client, and if the client accepts the offer, it may be bound to that client.

  An IP address is only BINDABLE by a single server at a time. A server must know for precisely which IP addresses it has on its list of BINDABLE addresses. A server does not know about any other server's list of BINDABLE addresses. (Although performance optimizations are possible where a server may develop hints about this information, they are not required).

  An IP address can move from the BINDABLE state into the BOUND state through the normal activity of the DHCP protocol where a server interacts with a client. When this happens, the Client Binding Management portion of the protocol is used to inform other servers of the change.

  A server can also transfer ownership of a BINDABLE IP address to another server upon request from that other server (and without any interaction beyond that with the other server).

o BOUND

An address that is BOUND is associated with a particular DHCP client, and usually is in use by that client (although it may have abandoned the lease on that IP address). It may be termed BOUND to that client. In the BOUND state the information about the client binding has not been propagated to all of the other servers in the server group.

o PUSHED

An address that is PUSHED is associated with a client in the same was as a BOUND address. However, an address in the PUSHED state indicates that all of the other servers in the server group have been informed of the existence of the binding to this client.

When a DHCP client releases a lease on an IP address it moves from either the BOUND or PUSHED state into the UNBINDABLE state, but no explicit PUSH operation is required.

When the lease time and any grace period implemented by a server both expire, then an IP address moves into the EXPIRED state.

Note that only a server that actually completes a CLIENT BINDING COMPLETE PUSH will place its IP address into the PUSHED state. The servers who receive the CLIENT BINDING COMPLETE PUSH will place their IP addresses into the BOUND state.

DISCUSSION:

> Many DHCP servers implement something called a "grace period", which is a period after the the lease on a binding expires that an IP address will not be offered to another DHCP client. A lease which is in this "grace period" is still BOUND or PUSHED as far as the inter-server protocol is concerned.

o EXPIRED

An IP address is EXPIRED when it was BOUND and the term of the lease (and any implemented grace period) has run out. It may be termed EXPIRED to that client.

An EXPIRED IP address will transition to the UNBINDABLE state when the server who shows it as EXPIRED receives an UNBINDABLE COMPLETE POLL. It will respond to the UNBINDABLE COMPLETE POLL after making the IP address UNBINDABLE.

It may be moved back into the BOUND state by an REQUEST/INIT-REBOOT request from the previously bound client.

Note that an IP address can never go from BOUND to one client to BOUND to another client without first passing through the UNBINDABLE state. The line across the middle of the state transition diagram helps to illustrate this.

Further, note that the transition from POLLING to BINDABLE requires the successful completion of an UNBINDABLE COMPLETE POLL.

3.5. Overview of Server Operation

This section will give a brief sketch of the of the core elements of the Client Binding Management and Address Management parts of the protocol (from the perspective of an already configured group of servers). Many of the possible cases are not described here, and this section is not to be considered definitive. The definitive description of this information is contained in Section 6 and in the case of conflicts with information found there, the information in Section 6 will govern.

3.5.1. DISCOVER

Prior to the receipt of a DISCOVER message, each server should have built up a list of BINDABLE IP addresses -- for two reasons. First, because an UNBINDABLE COMPLETE POLL is required to move an IP address into the BINDABLE state, and an UNBINDABLE COMPLETE POLL may not be possible due to server failure at any given instant. Second, because even if an UNBINDABLE COMPLETE POLL was possible it would generally take too long to do between a DISCOVER and an OFFER message.

A server should offer a BINDABLE address to a client upon receipt of a DISCOVER message.

There are no inter-server protocol activities required when a DIS-COVER is processed and an OFFER is returned to the client (assuming of course that a BINDABLE address was available to be offered).

3.5.2. REQUEST/SELECTING

When a client accepts an offer by sending a SELECTING message, then the server updates its stable storage with the binding information and ACKs the client. It must then perform a CLIENT BINDING COMPLETE PUSH operation to push the binding information to all of the other

servers (to which it can communicate at that time). There are some limitations on the lease time that can be offered to the client until at least one successful CLIENT BINDING COMPLETE PUSH has succeeded for the offering server. See Section 4.4.1 for additional details.

3.5.3. REQUEST/INIT-REBOOT

In the usual case where the server who created the binding for the requesting client managed to PUSH that information to the other servers using a CLIENT BINDING COMPLETE PUSH, the receiving server will have the binding information for this client. If this information can be verified, then ACK the client -- else NAK it.

If the IP address was in the EXPIRED state, then move the IP address to the PUSHED state.

3.5.4. REQUEST/RENEWING

Upon receipt of a RENEWAL message (which is unicast from the client to the server), it is expected that the server will have accurate information concerning the binding of the client. If it does not, process the message like a REBINDING, below. Given that the server has information sufficient to extend the lease, it should update its stable storage with the lease extension, and then ACK the client with the extended time. Then it must perform a CLIENT BINDING COMPLETE PUSH operation to the other servers with the updated binding information.

3.5.5. REQUEST/REBINDING

Upon receipt of a REBINDING message (which is broadcast from the client), the server will check to see if it has any information about the binding for this client. There are several possible cases:

1. Current information shows that this client owns the IP address.

   Extend the lease, update stable storage, ACK the client, and perform a CLIENT BINDING COMPLETE PUSH with the information to the other servers.

2. Current information shows that some other client is BOUND to this IP address.

   This is a problem. Make the IP address UNAVAILABLE (see Section 12 for details).

3. Current information says this IP address is UNBINDABLE.

   In this case, a server has probably created a binding and then failed to propagate the information to this server. Perform a POLL operation to see if any communicating server has any better information.

   If information is returned, then move to the appropriate case in this list.

   If no information is returned, then extend the lease on the IP address, update stable storage, ACK the client, and PUSH the information to the other servers.

3.5.6. RELEASE

When a release is received, if the client matches the binding information in the server, then update stable storage with the release, set the IP address UNBINDABLE, and perform a CLIENT BINDING COMPLETE PUSH to inform other servers.

If the CLIENT BINDING COMPLETE PUSH operation fails due to inability of an UPDATE message to succeed to another server, do nothing.

3.5.7. Expiration

When a lease on an IP address expires, move the lease to the EXPIRED state and update stable storage with this information. From now on, if some server performs an UNBINDABLE COMPLETE POLL operation to gather information about this IP address, make the IP address UNBINDABLE, update stable storage, and respond with the state of the IP address as UNBINDABLE.

3.6. When a server is down or partitioned and can't be contacted

When a server is down or partitioned (i.e., can't be reached), then some aspects of the normal DHCP client processing are different. This section summarizes those differences:

- o Client lease times for new clients will never be greater than MAXIMUM_UNPUSHED_LEASE_TIME, since a CLIENT BINDING COMPLETE PUSH cannot succeed.

- o No UNBINDABLE COMPLETE PUSH will succeed, and thus no server will be able to transition an address from the UNBINDABLE state into

the BINDABLE state. If a server runs low on addresses, it will have to use TRANSFER messages to acquire new addresses from other servers.

4. Client Binding Management

Client binding management is the aspect of the protocol which is concerned with communicating information about client bindings from one server to another. It is the core of the inter-server protocol.

The following messages and operations are used explicitly by a server participating in the interserver protocol when DHCP client requests and events require it, and are used implicitly by the SCSP cache alignment procedure whenever a server (re)establishes communication with another server.

4.1. Client Binding Messages

o CLIENT BINDING UPDATE

Update a single server with client binding information. This operation will not complete successfully unless and until that server is updated with the information being sent.

o CLIENT BINDING QUERY

Query a single server for its client binding information.

4.2. Client Binding Operations

The operations defined in for client binding management are:

o CLIENT BINDING COMPLETE PUSH

This operation involves one server using the UPDATE message to inform all of the other servers about updated client binding information. While there is utility in reaching even one other server (in some cases) the operation is not deemed to have succeeded unless all of the other servers were successfully updated with the new information.

o CLIENT BINDING POLL

This operation involves one server using the QUERY message to inquire of every other server about client binding information concerning a particular IP address. If all of the other servers

are not operational, the requesting server will use any information it receives.

4.3. Client Binding Information

When binding data is sent as part of message concerned with client binding management it contains the following information:

- o IP Address
- o Expiration [expressed as a delta seconds from the current time]
- o Client ID
- o MAC Address [including the hardware type]
- o Last Transaction [selected from the list below]
- o Last Transaction Time [expressed as a delta seconds from the current time]
- o Last Transaction Server [an IP address]

Each server must maintain as part of the binding information the "last transaction time", the "last transaction", and the "last transaction server" associated with that binding.

The last transaction time is the time at which the binding changed in response to a request (the last transaction) from the client. The last transaction time is returned in an address information message as a number of seconds from "now".

The possible last transactions are listed below. This list is ordered by the precedence of the transactions and is used to help determine if a response to an address information message contains more recent information than that currently held by a server.

The last transaction is one of the following:

- o DHCPREQUEST/SELECTING
- o DHCPREQUEST/REBINDING
- o DHCPREQUEST/INIT-REBOOT
- o DHCPREQUEST/RENEWING

- o DHCPRELEASE
- o EXPIRATION

The IP address state information is transmitted as well, and it consists of one of the following states:

- o UNBINDABLE
- o POLLING
- o BINDABLE
- o BOUND
- o PUSHED
- o EXPIRED

4.4. Initiating Client Binding Operations and Messages

4.4.1. CLIENT BINDING COMPLETE PUSH

The CLIENT BINDING COMPLETE PUSH operation is initiated whenever the state of a server's client binding cache is changed, typically by the receipt of a DHCP client request or expiration of a lease.

The lease time that is offered to a DHCP client must not be greater than the MAXIMUM-UNPUSHED-LEASE-TIME for that SG until at least one CLIENT BINDING COMPLETE PUSH has succeeded for that client binding. Thus, as long as the state of the IP address is BOUND, then the client should be offered the MAXIMUM-UNPUSHED-LEASE-TIME.

The lease time that is sent to the other servers in the CLIENT BINDING COMPLETE PUSH is the lease time that the server would like to give to the DHCP client, and once a CLIENT BINDING COMPLETE PUSH has succeeded with that lease time in it (and the IP address state is set to PUSHED), then the server is free to actually extend the client's lease on the IP address with that lease time.

The servers which receive the CLIENT BINDING COMPLETE PUSH will place their IP addresses into the BOUND state, not the PUSHED state.

4.4.2. CLIENT BINDING POLL

The CLIENT BINDING POLL is used when the server has received a DHCP client request but believes that it has insufficient or out-of-date information concerning this client's binding. Thus, the CLIENT BINDING POLL is an attempt to gather more recent and up-to-date information from the other servers in the SG.

DISCUSSION:

Is this really necessary? Given that SCSP will "align" the caches of the servers at every reconnect, then what is the value of asking "again"?

4.4.3. CLIENT BINDING UPDATE

The CLIENT BINDING UPDATE is initiated in three ways.

It is initiated at the client binding management level as the underlying operation in a CLIENT BINDING COMPLETE PUSH. It is initiated at the client binding management level when a server realizes that the server who returned information as a result of a CLIENT BINDING QUERY returned information which was less up-to-date than that available to the current server. It is initiated at the SCSP level as part of the cache state alignment process.

4.5. Responding to Client Binding Messages

When a server receives the following client binding messages, it should respond as detailed below. Note that operations consist of multiple messages at the initiator, but that when processing incoming requests, only individual messages are evident.

4.5.1. CLIENT BINDING QUERY

The proper response to a CLIENT BINDING QUERY is to respond with the current information in the client binding cache.

4.5.2. CLIENT BINDING UPDATE

The proper response to a CLIENT BINDING UPDATE is to determine if the information received is more current than that available in the server's cache. If it is not, then respond negatively to this request. If it is, then update the client binding cache, ensure that

the changes have been written to stable storage, and respond successfully. Note that no CLIENT BINDING UPDATE should generate additional client binding message activity (i.e., the CLIENT BINDING UPDATE should not generate a CLIENT BINDING COMPLETE PUSH).

When a CLIENT BINDING UPDATE is received, the IP address should be placed into the BOUND state, not the PUSHED state. Only the actual server performing the CLIENT BINDING COMPLETE PUSH will place its IP address into the PUSHED state.

5. Address Managment

Address management is the aspect of the protocol concerned with managing the state of IP addresses that are not currently bound to any client. It is a necessary part of the protocol in order to support certain goals in the client binding management part of the protocol, principally that of allowing a server to continue to operate even though it was partitioned from other servers in the server group.

5.1. Address Management Operations

o UNBINDABLE COMPLETE POLL

In this operation, all of the other servers are contacted using a QUERY operation concerning one (or more) IP addresses, and they all report on whether that IP address(es) is UNBINDABLE or not. If they are UNBINDABLE, then the current information on that IP address is also reported (as in a CLIENT BINDING POLL). In contrast to a CLIENT BINDING POLL, this operation fails if any server cannot be contacted or if any server answers the QUERY with a negative answer (i.e., the IP address is not currently UNBINDABLE). It succeeds when all of the servers answer that the address is UNBINDABLE.

There is a subtle interaction required with the group management layer of the protocol. A successful UNBINDABLE COMPLETE POLL must be inhibited in certain cases where a server has been removed from a server group.

The case is question is that where a server is removed from a server group by a different server. Immediately after this happens, all UNBINDABLE COMPLETE POLLS must fail for a period equal to the MAXIMUM-UNPUSHED-LEASE-TIME. After that time passes, then UNBINDABLE COMPLETE POLLS may operate as they normally do.

DISCUSSION:

This covers the situation where a server gives a lease to a while both the client and server are partitioned. Then, the server goes away completely. The client stays up, but remains partitioned. Then, the dead server is removed by another server from the server group. At this point, UNBINDABLE COMPLETE POLL operations could (except for the above restriction) begin to complete successfully. However, the client that was given a lease while partitioned along with the server that died certainly has an address, and when the partition is removed (just after the UNBINDABLE COMPLETE POLL operation which declared its IP address now BINDABLE for some server), there would be a very dangerous situation developing.

The solution is to only offer leases to clients of the MAXIMUM-UNPUSHED-LEASE-TIME until the information concerning their client binding reaches all of the other servers in the group. Once that happens, then they can be offered the normal lease time.

Thus, whenever any server is removed from the group (where it doesn't remove itself), then there is a possibility that it may have offered leases to clients about which no other server would have any record. In this case, the remaining servers must wait the MAXIMUM-UNPUSHED-LEASE-TIME before being able to complete an UNBINDABLE COMPLETE POLL and reuse the BINDABLE addresses that the removed server was using.

5.2. Address Management Messages

The following messages are part of the address management portion of the protocol.

- TRANSFER

This message is used to transfer BINDABLE IP addresses from one server to another (especially when the SG is partitioned and the normal UNBINDABLE COMPLETE POLL cannot be used to make an IP address BINDABLE, but also when all of the UNBINDABLE IP addresses have already been made BINDABLE by some server).

The information sent from the initiating to the responding server includes the subnet specification and the number of BINDABLE IP addresses the initiating server has available for that address pool, and the number of BINDABLE IP addresses it is requesting.

The responding server is free to give the initiating server all, some, or none of the number of IP addresses the initiating server has requested.

o UNBINDABLE QUERY

The UNBINDABLE QUERY operation is the primitive query from which the UNBINDABLE COMPLETE POLL is constructed. It is identical to the CLIENT BINDING QUERY defined above in terms of the data returned, although the actions taken when it is received are slightly different.

5.3. **Initiating Address Management Operations and Messages**

o UNBINDABLE COMPLETE POLL (operation)

This operation is initiated when the server detects that it needs to generate more BINDABLE IP addresses. It will initiate this operation whenever the number of BINDABLE IP addresses drops below a configurable threshold.

Prior to initiating this operation, the server must change the state for each IP address that will be part of the UNBINDABLE COMPLETE POLL from UNBINDABLE to POLLING, and commit this state change to stable storage.

DISCUSSION:

Is the commit to stable storage really necessary? Given that we will abandon the POLL if we reboot (presumably), what is the value of remembering that we were doing it?

For every IP address for which the UNBINDABLE COMPLETE POLL operation fails (i.e., some server responds in such a way that indicates that the IP address is not UNBINDABLE, or some server fails to respond at all), the IP address' state should be reset to UNBINDABLE.

o TRANSFER (message)

The TRANSFER message, which attempts to transfer some IP addresses from some other server to the initiating server, is initiated whenever the number of BINDABLE IP addresses in an address pool falls below a configurable threshold.

5.4. Responding to Address Management Messages

- o TRANSFER

  When receiving a TRANSFER message, the responding server inspects its list of BINDABLE addresses for the address pool to which the TRANSFER operation refers. It will attempt to offer the initiating server as many addresses as it requested, with the limitation that it will never give away more than half of its pool of BINDABLE addresses in any one request.

- o UNBINDABLE QUERY

  The responding server will respond to this query just like it responds to a CLIENT BINDING QUERY as far as the information communicated to the initiating server is concerned.

  In addition, if the IP address mentioned in this query was in the EXPIRED state, prior to responding to this message, the responding server will move that IP address to the UNBINDABLE state, commit this change to stable storage, and then respond with information that indicates the IP address in question was UNBINDABLE.

  Note that an UNBINDABLE QUERY will not be generated to any server if at least one server in the SG is currently not able to be contacted, as known by the SCSP "Hello" subprotocol. This will prevent unnecessary transitions from the EXPIRED to the UNBINDABLE state when an UNBINDABLE COMPLETE POLL would not be able to complete in any case.

6. Actions in Response to DHCP Client Messages and Events

This section defines the actions that should be taken in the client binding and address management portions of the protocol when incoming DHCP requests (messages) are received.

DISCUSSION:

There is considerable commonality in the sections that describe the various DHCP client messages below. Once the details have stabilized, it should be possible to compress the explanations.

6.1. DISCOVER

Prior to the receipt of a DISCOVER message, each server should have built of a list of BINDABLE IP addresses -- for two reasons. First, because a CLIENT BINDING COMPLETE POLL is required to get a BINDABLE IP address, and a CLIENT BINDING COMPLETE POLL may not be possible due to server failure at any given instant. Second, because even if a CLIENT BINDING COMPLETE POLL were possible, it would be unwise to require such an operation between a receipt of a DISCOVER message and the response of an OFFER to a client.

There are several cases involved in processing a DISCOVER request, depending on the state of the requested IP address in the DISCOVER request:

o No specific IP address requested.

  Offer a BINDABLE address to the client. Record that this address was offered in the cache memory of the server, but there is no need to update the stable storage of the server with any information. The IP address continues to be BINDABLE as far as the inter-server protocol is concerned.

o Requested IP address is UNBINDABLE.

  If the IP address is UNBINDABLE, then perform a UNBINDABLE COMPLETE POLL operation in an attempt to make the IP address BINDABLE. If the operation is successful, then respond as though the IP address were BINDABLE, below. If the results of the attempt to make the IP address BINDABLE resulted in a discovery that the IP address is now BOUND or PUSHED, then respond as for BOUND our PUSHED, below. Otherwise (i.e., the IP address is BINDABLE for some other server, or no an UNBINDABLE COMPLETE POLL was not possible) then respond as above for "No specific IP address requested".

o Requested IP address is BINDABLE.

  Offer the IP address to the client. IP address remains BINDABLE.

o Requested IP address is BOUND or EXPIRED.

  If the IP address is BOUND or EXPIRED to the requesting client, then set it to BOUND and offer it to the client -- with a lease time of MAXIMUM-UNPUSHED-LEASE-TIME. Otherwise (i.e., the IP address is BOUND or EXPIRED to some other client), respond as in "No specific IP address requested", above.

o Requested IP address is PUSHED.

If the IP address is PUSHED to the requesting client, then offer it to the client -- with a normal lease time. Otherwise (i.e., the IP address is PUSHED to some other client), respond as in "No specific IP address requested", above.

6.2. REQUEST/SELECTING

The client uses a REQUEST/SELECTING to accept the offer of a lease made by a server. When a server receives such a message, and where the server-id option reflects the IP address of that server, then if the IP address is in the following states the server should respond in the following way:

o UNBINDABLE

If the IP address is UNBINDABLE, then perform a UNBINDABLE COMPLETE POLL operation in an attempt to make the IP address BINDABLE. If that operation is successful, then respond as though the IP address were BINDABLE, below. If the results of the attempt to make the IP address BINDABLE resulted in a discovery that the IP address is now BOUND, then respond as for BOUND, below. Otherwise (i.e., the IP address is BINDABLE for some other server, or no a complete POLL was not possible) NAK the REQUEST.

o BINDABLE

If the IP address is BINDABLE and has been offered to the requester, then bind the IP address to the client, set the IP address BOUND, and update stable storage. Then, ACK the client, and finally perform a PUSH operation of the binding information to the other servers.

o BOUND or EXPIRED

If the IP address is BOUND or EXPIRED to the requesting client, then set the state to BOUND, update the expiration time using the normal lease time, update stable storage, ACK the client with the MAXIMUM-UNPUSHED-LEASE-TIME, and perform a CLIENT BINDING COMPLETE PUSH with the normal lease time.

If the IP address is BOUND or EXPIRED to a different client, then NAK this REQUEST.

o PUSHED

If the IP address is PUSHED to the requesting client, set the IP address to be PUSHED, update the expiration time, update stable storage, and ACK the client. Finally, perform a CLIENT BINDING COMPLETE PUSH operation of the updated binding information to the other servers.

Use the normal lease time in all of the above operations.

If the IP address is PUSHED to some other client, then NAK the request.

6.3. REQUEST/INIT-REBOOT

The client uses a REQUEST/INIT-REBOOT to query the server (as part of the client boot process) to determine if a "remembered" binding is still valid. If the requested IP address will be in one of the following states:

o UNBINDABLE

If the IP address is UNBINDABLE, then perform a UNBINDABLE COMPLETE POLL operation in an attempt to make the IP address BINDABLE. If the operation is successful, then respond as though the IP address were BINDABLE, below. If the results of the attempt to make the IP address BINDABLE resulted in a discovery that the IP address is now BOUND, then respond as for BOUND, below. Otherwise (i.e., the IP address is BINDABLE for some other server, or a complete POLL was not possible) NAK the REQUEST.

DISCUSSION:

This means that if a server creates a binding for a client and fails to PUSH the information to any other server prior to undergoing a server failure, and if the client is powered off prior to the time when it will issue a REBINDING message, it will not get back the same lease when it is powered back on. The reasoning for this (and the difference from the REBINDING case below) is that in this case the server has no way to determine if the requested address in the INIT-REBOOT request is current or perhaps very old indeed. In the REBINDING case the client is currently using the address, so the client at least believes that it is current and not in use by some other client. In this case, however, no such assumption is possible.

In the case where a server which creates a binding fails prior to PUSHing the information about a lease to some other server, and the client which receives that binding makes a REBINDING request prior to either failing or being shutdown, it will get back the existing binding upon restart and INIT-REBOOT -- since the REBINDING will have caused a recovery of the binding information and that will have been distributed through a CLIENT BINDING COMPLETE PUSH.

o BINDABLE

If the IP address is BINDABLE, then bind the IP address to the client, set the IP address BOUND, and update stable storage. Then, ACK the client, and finally perform a PUSH operation of the binding information to the other servers.

o BOUND or EXPIRED

If the IP address is BOUND or EXPIRED to the requesting client, then set the state to BOUND, update the expiration time using the normal lease time, update stable storage, ACK the client with the MAXIMUM-UNPUSHED-LEASE-TIME, and perform a CLIENT BINDING COMPLETE PUSH with the normal lease time.

If the IP address is BOUND or EXPIRED to a different client, then NAK this REQUEST.

o PUSHED

If the IP address is PUSHED to the requesting client then set the IP address PUSHED, update the expiration time, update stable storage, and ACK the client. Finally, perform a CLIENT BINDING COMPLETE PUSH operation of the updated binding information to the other servers. Use the normal lease time for all of the above operations.

If the IP address is PUSHED to some other client, then NAK the request.

6.4. REQUEST/RENEWING

Upon receipt of a RENEWAL message (which is unicast from the client to the server), it is expected that the server will have accurate information concerning the binding of the client since this is the server that the client believes most recently sent an ACK to the client concerning this IP address binding.

Perform the following actions if the IP address being renewed (i.e., the IP address in ciaddr) is in one of these states:

o UNBINDABLE

If the IP address is UNBINDABLE, then perform an UNBINDABLE COM-PLETE POLL operation in an attempt to make the IP address BIND-ABLE. If the operation is successful, then respond as though the IP address were BINDABLE, below. If the results of the attempt to make the IP address BINDABLE resulted in a discovery that the IP address is now BOUND, then respond as for BOUND, below.

If the IP address is determined to be BINDABLE for some other server, then NAK the request, and set the IP address to be UNAVAILABLE since this likely represents a duplicate allocation of an IP address (see Section 11, Open Questions, for details).

Otherwise NAK the request.

o BINDABLE

If the IP address is BINDABLE, then bind the IP address to the client, set the IP address BOUND, and update stable storage. Then, ACK the client, and finally perform a PUSH operation of the binding information to the other servers.

o BOUND or EXPIRED

If the IP address is BOUND or EXPIRED to the requesting client, then set the state to BOUND, update the expiration time using the normal lease time, update stable storage, ACK the client with the MAXIMUM-UNPUSHED-LEASE-TIME, and perform a CLIENT BINDING COM-PLETE PUSH with the normal lease time.

If the IP address is BOUND or EXPIRED to a different client, then NAK this REQUEST.

o PUSHED

If the IP address is PUSHED to the requesting client then set the IP address PUSHED, update the expiration time, update stable storage, and ACK the client. Finally, perform a CLIENT BINDING COMPLETE PUSH operation of the updated binding information to the other servers. Use the normal lease time for all of the above operations.

If the IP address is PUSHED to some other client, then NAK the request and set the IP address to UNAVAILABLE. (see Section 11,

Open Questions, for details).

6.5. REQUEST/REBINDING

Upon receipt of a REBINDING message (which is broadcast from the client), the server will check to the state of the address requested for rebinding (i.e., the ciaddr). There are several cases possible:

- UNBINDABLE

  If the IP address is UNBINDABLE, then perform an UNBINDABLE COMPLETE POLL operation in an attempt to make the IP address BINDABLE. If the operation is successful, then respond as though the IP address were BINDABLE, below. If the results of the attempt to make the IP address BINDABLE resulted in a discovery that the IP address is now BOUND, then respond as for BOUND, below.

  If the IP address is determined to be BINDABLE for some other server, then NAK the request. Set the IP address to be UNAVAILABLE since this likely represents a duplicate allocation of an IP address (see Section 11, Open Questions, for details).

  If no information is returned from any server that this IP address is anything but UNBINDABLE, then consider the address BOUND to this client, and proceed as in BOUND below.

  DISCUSSION:

  This is one of the key points of the inter-server protocol. In this case, a server has created a binding and then failed prior to telling any other server about that binding. Eventually, the client to whom that binding was made will attempt a REQUEST/REBINDING and contact a different server. That different server will be able to determine nothing about that IP address. As far as can be determined, it is not BOUND to any client, and it is not BINDABLE for any other server. In this restricted case, the server will renew the lease for the client and move the IP address into the BOUND state -- and PUSH this information to the rest of the servers.

  How can this be safe? Well, remember that the client is presently using the IP address to make this request. In this limited case where a server crashes before PUSHing information about a BOUND IP address to any other server, the client to whom the IP address is BOUND is the only running machine with any record of that binding. In this case, the DHCP servers will accept that client's information about the binding as

correct.

o BINDABLE

If the IP address is BINDABLE, then bind the IP address to the client, set the IP address BOUND, and update stable storage. Then, ACK the client, and finally perform a PUSH operation of the binding information to the other servers.

o BOUND or EXPIRED

If the IP address is BOUND or EXPIRED to the requesting client, then set the state to BOUND, update the expiration time using the normal lease time, update stable storage, ACK the client with the MAXIMUM-UNPUSHED-LEASE-TIME, and perform a CLIENT BINDING COMPLETE PUSH with the normal lease time.

If the IP address is BOUND or EXPIRED to a different client, then NAK this REQUEST.

o PUSHED

If the IP address is PUSHED to the requesting client then set the IP address PUSHED, update the expiration time, update stable storage, and ACK the client. Finally, perform a CLIENT BINDING COMPLETE PUSH operation of the updated binding information to the other servers. Use the normal lease time for all of the above operations.

If the IP address is PUSHED to some other client, then NAK the request and set the IP address to UNAVAILABLE. (see Section 11, Open Questions, for details).

6.6. RELEASE

When a RELEASE is received, an IP address will be in one of the following states:

o UNBINDABLE

If the IP address is UNBINDABLE, then perform a CLIENT BINDING POLL operation in an attempt to determine if this IP address is BOUND to any client.

If the results of the POLL operation indicate that the IP address is now BOUND, then respond as for BOUND, below.

If the IP address is determined to be BINDABLE for some other server, then NAK the request. Set the IP address to be UNAVAILABLE since this likely represents a duplicate allocation of an IP address (see Section 11, Open Questions, for details).

Otherwise, ignore the RELEASE.

o BINDABLE

If the IP address is BINDABLE, ignore the RELEASE.

o BOUND, PUSHED, or EXPIRED

If the IP address is BOUND, PUSHED, or EXPIRED to the requesting client set the IP address to be UNBINDABLE, update stable storage, and perform a CLIENT BINDING COMPLETE PUSH to update the other servers with this information.

6.7. Lease Period Expiration

When the lease period on a BOUND or PUSHED IP address expires, set the IP address to be EXPIRED and update stable storage.

7. Group Management

The group management part of the protocol is concerned with configuring a server into or out of a server group (SG). It allows discovery of information concerning the configuration of an existing server group as well as the address pools that are managed by a server group. While it is possible to conceive of a statically defined server group, the operational characteristics (both for group startup as well as removal of a server from a group) are quite painful.

Group management messages are used add a server to a group as well as to remove a server from a group. A server must add itself to a group -- it cannot be added by another server. A server may be removed by any server in the group, including itself.

In addition to changing the group membership, group management messages are used to keep the various servers up to date with respect to the current membership of the group.

Once a server successfully become part of a group using the group management messages, it the goes into the SCSP protocol. This protocol determines which servers in the SG are currently in communication with this server, and starts an automatch "cache alignment" process

with each connected server.

7.1. Group Management Operations

o SG CHANGE

The SG CHANGE operation is a two-stage operation made up of a propose and then a commit phase. It uses the SG PROPOSE CHANGE and SG COMMIT CHANGE messages as part of this operation. It is used to change the membership of the group, either to add a server or to remove a server.

7.2. Group Management Messages

o SG DISCOVERY QUERY

The first stage of becoming a server participating in the inter-server protocol is to determine the existing SG ID for each SG for which participation in the inter-server protocol is desired.

Assuming that a server has been provided or can discover the IP address of a server maybe in a group to which it wants to join, a server who wants to become a member of a group will send a SG DISCOVERY QUERY message to that server.

The reply to the SG DISCOVERY QUERY message is a message which contains the list of SG identifiers for all of the groups to which the replying server belongs. These SG ids can then be used in SG CONFIGURATION messages to determine more information about each SG.

This operation is performed only upon one server at a time, since at this point there is no notion of a "current" server group.

o SG CONFIGURATION QUERY

The SG CONFIGURATION QUERY operation has several suboperations, corresponding to the following types of configuration information: subnets, IP addresses, client configuration information, and vendor specific information.

Each SG CONFIGURATION QUERY operation is read-only to the receiving server. The particular SG CONFIGURATION QUERY suboperations are:

- o Subnets

  The specific subnets managed by this SG are returned in this as part of this operation.

- o IP Addresses

  The IP addresses which are managed by this SG within this sub-net are return as the result of this operation.

- o Client Configuration Information

  The client configuration information associated with this sub-net is returned as the result of this operation.

- o Vendor Specific Information

  Provision is made for vendor specific configuration information to be returned in the SG CONFIGURATION message. Its format is TBD, but should be regular even though vendor specific.

o SG PROPOSE CHANGE UPDATE

The SG PROPOSE CHANGE UPDATE message is sent to all of the servers in a SG to propose a new membership in the server group. The information sent with this message is an updated list of the servers in the group. The servers to add to the group and servers to remove from the group are both listed in the same message.

o SG COMMIT CHANGE UPDATE

The SG COMMIT CHANGE UPDATE message is sent to all of the servers in the SG to commit a change the was proposed in a SG PROPOSE CHANGE operation.

7.3. Initiating Group Management Operations and Messages

7.3.1. SG CHANGE (operation)

The SG CHANGE operation consists of the the following steps:

o Determine the group membership using an SG CONFIGURATION message.

Find out to whom to send all of the SG CHANGE messages.

- o Send a SG PROPOSE CHANGE message to every member of the SG.

  This message has the current group specifier in the message, along with the new group membership. As the joining server cycles through the existing members of the group, it will be rationalizing the group specifiers among the group and the entire group's picture of the membership of the group. If it encounters a server whose view of the group membership lags behind that of the server from which the joining server received its idea of group membership, then it will bring that server up to date.

  If, on the other hand, it encounters a server that has a more up to date version of the group membership than the one from which it is operating, it will have to update its idea of the group membership and then start the proposal sequence over. All of the servers with which it has created proposals will be forced to update their view of group membership as part of this process.

  At the end of this process of proposal generation, all of the servers in the group share a common picture of both the group membership as well as the current proposal.

- o Reverify the group membership from at lease one server using an SG CONFIGURATION message.

  This is to ensure that all of the members of the group have actually been sent a SG PROPOSE CHANGE message.

- o Check the proposal timer.

  The initiating server must have started a timer when it sent out the first SG PROPOSE CHANGE message, and if that timer has less than time/2 time left on it, the joining server SHOULD start the process over.

- o Send a SG COMMIT CHANGE message to every member of the SG.

  As soon as this completes successfully with one server, the server has changed the membership of the group, but the initiating server MUST continue to try to update the other servers as long as they remain in the server group.

7.3.2. SG DISCOVERY QUERY (message)

This is sent when a server wishes to know the groups to which another server is a member. It is used primarily when starting up a server in the initial discovery of the server group configuration.

7.3.3. SG CONFIGURATION QUERY (message)

This message is sent to determine the details of the configuration of the server group. A server would typically initiate these messages as part of the process of confirming that it wished to be part of a particular server group.

The SG CONFIGURATION QUERY operation has several suboperations, corresponding to the following types of configuration information:

o Subnets

The specific subnets managed by this SG are returned in this as part of this operation.

o IP Addresses

The IP addresses which are managed by this SG within this subnet are return as the result of this operation.

o Client Configuration Information

The client configuration information associated with this subnet is returned as the result of this operation.

o Vendor Specific Information

Provision is made for vendor specific configuration information to be returned in the SG CONFIGURATION QUERY message. Its format is TBD, but should be regular even though vendor specific.

7.4. Responding to Group Management Messages

7.4.1. SG PROPOSE CHANGE UPDATE

Upon receipt of a SG PROPOSE CHANGE UPDATE message, if no existing proposal exists that has not timed out, a server will create a single "proposed" group specifier from the current group specifier by incrementing the group sequence number by 1. The creation of this proposed group specifier will inhibit the creation of another proposed group specifier for a 30 seconds.

If an existing proposal exists that has not timed out, the responding will respond negatively to the SG PROPOSE CHANGE UPDATE message.

DISCUSSION:

Clearly a deadlock situation can occur where two servers are trying to join a group at the same time, and each is working from "opposite ends" of the group. In this case, where the joining server gets a failure from a SG PROPOSE CHANGE UPDATE message due to the existence of a valid proposal that has not timed out, then the joining server should backoff an amount of time that is based in part on its IP address before trying again. The exact algorithm is TBD.

This proposed group specifier will not be used in any messages until it moves to the accepted stage and become the current group specifier (see below for how it does that).

If a second SG PROPOSE CHANGE UPDATE request is received from a server, that message will supersede the existing proposal and the timer will be reset.

DISCUSSION

Is there some possible attack here? Should we limit one servers proposals from tying up the "proposal" for more than 3 minutes at a time, for instance?

7.4.2. SG COMMIT CHANGE UPDATE

Upon receipt of a SG COMMIT CHANGE UPDATE message, the current proposal is compared with the data in the SG COMMIT CHANGE UPDATE message, and if it compares successfully, the proposed new group becomes the current group and the group specifier is changed.

Once a SG COMMIT CHANGE UPDATE message is received, the receiving server MUST examine all of its IP addresses. For every IP address for which the "last transaction server" is a server which was previously in the group and is now not in the group, the following action should be taken:

If the IP address is shown as ever having been BOUND to a client, and if that client does not now have a different IP address, then the IP address should be set to BOUND to that client, the lease time should be restarted for the previously recorded lease time.

DISCUSSION:

This is a key aspect of the protocol in terms of safely removing possibly partitioned servers from the group. The specific case

that this protects against is as follows.

If a connected server creates a client binding, and successfully performs a CLIENT BINDING COMPLETE PUSH operation, and then renews its client's lease for the full lease time -- and then becomes partitioned, there can be problems if that server is ultimately removed from the group much later. If the server is partitioned for longer than the client's lease time, and if all of the other servers move this IP address to EXPIRED, and if then some server tries (unsuccessfully) to perform an UNBINDABLE COMPLETE POLL -- which will move the EXPIRED addresses to UNBINDABLE. Now, the partitioned server has updated the client several times, and the other servers by this time all believe that the IP address is UNBINDABLE. If the partitioned server then fails and is removed from the SG -- the other servers could (in the absence of the above algorithm) believe that they only need wait the MAXIMUM-UNPUSHED-LEASE-TIME before then can make those UNBINDABLE addresses BINDABLE. But in this case that would cause a failure. Thus, when a server is removed from a SG, each remaining server must look around for any IP addresses that it previously PUSHED, and set them up with their previous maximum lease time in order to catch this case.

7.4.3. SG DISCOVERY QUERY

The server groups to which the current server belongs are returned as the response to an SG DISCOVERY QUERY message.

7.4.4. SG CONFIGURATION QUERY

The SG CONFIGURATION QUERY operation has several suboperations, corresponding to the following types of configuration information:

- Subnets

  The specific subnets managed by this SG are returned in this as part of this operation.

- IP Addresses

  The IP addresses which are managed by this SG within this subnet are return as the result of this operation.

- Client Configuration Information

  The client configuration information associated with this subnet is returned as the result of this operation.

- o Vendor Specific Information

  Provision is made for vendor specific configuration information to be returned in the SG CONFIGURATION QUERY message. Its format is TBD, but should be regular even though vendor specific.

8. SCSP Message Mapping

This section develops the SCSP capabilities supporting the DHCP interserver protocol. The Server Cache Synchronization Protocol (SCSP) is found in [1]. The organization of this section is 1) we present a brief overview of SCSP (and refer to appendices for a more detailed discussion), 2) we discuss the mapping of the DHCP inter-server protocol onto SCSP and how the various DCHP interserver messages are mapped into SCSP messages, 3) we identify the modifications to the SCSP protocol as identified in [1] necessary for the mapping of the DHCP interserver protocol onto SCSP, 4) we present the specific formats of the DHCP protocol specific SCSP records and 5) we present a list of the open issues with respect to the mapping onto SCSP.

8.1. SCSP Overview

The Server Cache Synchronization Protocol (SCSP) is a protocol which provides the generic functions necessary to provide loose synchronization between a set of distributed databases. The protocol, which is presented in [2], was developed to specifically address to issues associated with synchronizing the caches of redundant servers which provide the server functionality of a specific client-server protocol. SCSP was built based upon the extensive experience in developing and running link state routing protocols such as OSPF [3]. Client server protocols for which a redundant server capability is being developed using SCSP are NHRP [4] and ATM ARP [5]. Here we present the use of SCSP to synchronize servers supporting the DHCPv4 client-server protocol.

The SCSP protocol consist of three separate sub-protocols, i.e.,

- o The "Hello" protocol: this protocol defines and maintains the status of the inter-server connection,

- o The "Cache Alignment" protocol: this protocol defines the cache synchronization capability for new servers and servers that, for whatever reason, have lost synchronization, and

- o The "Client State Update" protocol: this protocol provides the ongoing server cache synchronization through asynchronous client

state updates.

These sub-protocols define the semantics and high-level syntax of generic message sets and their exchanges in support of the capabilities provided. The SCSP associates replica databases into Server Groups (SG). The SCSP supports both point-to-point and point-to-multipoint connections between the local servers (LS) and the directly connected servers DCS(es). We discuss each of these sub-protocols in more detail in the appendices below.

SCSP defines five message types in the operation of the above subprotocols:

- o Hello
- o Cache Alignment (CA)
- o Cache State Update (CSU) Solicit (CSU_Sol)
- o CSU Request (CSU_Req)
- o CSU Reply (CSU_Rep).

The Hello and the CA messages are used within the Hello and the Cache Alignment subprotocol respectively. The CSU_Sol, CSU_Req and CSU_Rep messages are used to distribute cache records between the distributed servers of a server group. Full records are called Client State Advertisement (CSA) records. Summary records, which are essentially pointers to the full records, are called Client State Advertisement Summary (CSAS) records.

For a server to request a particular record, it can send a CSU_Sol message containing the CSAS to indicate the full record of interest. A server which receives a CSU_Sol is required to respond with a CSU_Req message containing the full CSA record associated with the CSAS of the CSU_Sol. The soliciting server follows the receipt of the CSU_Req with a CSU_Rep to acknowledge receipt. A server which wishes to communicate a full record to the rest of the SG would transmit a CSU_Req message containing the full CSA record. This is acknowledged with a CSU_Rep message.

DISCUSSION

In some cases the CSU_Sol, CSU_Req, CSU_Rep sequence is overkill when one wants to perform a simple query operation. See the discussion at the end of Section 8.3 for more details.

For now we accept that these capabilities are generically provided

discuss the DHCPv4 interserver protocol specific overlay on SCSP.

8.2. Mapping DHCP interserver onto SCSP

This section presents the relationship of SCSP to the DHCP interserver protocol, the assumptions made in developing this relationship and the specific mappings of DHCP interserver messages into SCSP.

The assumptions made in defining the DHCP client/server protocol mapping onto SCSP are the following:

o On the Issue of Protocol Encapsulation:

The assumption is that the SCSP messages, and in fact all interserver messages, are to be defined over UDP. Currently the SCSP messages within [2] are LLC/SNAP encapsulated.

o On the Interserver over SCSP Layering Model:

The interserver group management protocol will initialize a server into the group upon initial join, re-booting or reconnecting. Once this is complete the interserver group management protocol will initialize the SCSP protocol to handle the ongoing operation of the interserver cache alignment and address management functions.

o On the DHCP Interserver Sub-Protocols:

The current thinking goes as follows. The draft specification defines three DHCP interserver sub-protocols, i.e., the 'Client Binding Management' protocol (see Section 4), the 'Address Management' protocol (see Section 5), and the 'Group Management' protocol (see Section 7). The 'Client Binding Management' sub-protocol addresses the core of the interserver protocol in that it distributes and maintains the client binding records over the distributed SG. This sub-protocol is to be mapped onto SCSP and is assigned a unique SCSP 'Protocol ID' value, e.g., the SCSP ProtID = 4 assigned to DCHP. For this draft we assume that the Group Management sub-protocol is run on a separate UDP port from the SCSP UDP port. The Group Mgmt sub-protocols will be assigned a unique UDP port number = tbd. We had no compelling reason to carry the Address Management subprotocol on SCSP as for the Client Binding protocol, however for this draft we mantain both these sub-protocols within SCSP. If at a later date it is deemed useful to separate these two protocol 1) we can define separate SCSP protocol types for the Cache Management and the Address Management protocols, yet support them with a common Hello protocol link via the Hello protocol Family type field or 2)we can move

the address management sub-protocol out from SCSP as in the case of the Group management sub-protocol.

The mappings between the interserver messages and the SCSP messages will cover the interserver messages handling client binding and address management, but not the group management protocol functions of the interserver protocol. The group management messages are to be defined outside of SCSP, however these messages will follow the syntax of the SCSP message sets to simplify the parsing of the total message sets required within the DHCP interserver protocol.

The client binding management operations are CLIENT BINDING COMPLETE PUSH and CLIENT BINDING POLL. CLIENT BINDING COMPLETE PUSH is required to distribute binding information and to increase the initial lease period to the desirable lease period. The CLIENT BINDING POLL is required to solicit information on client bindings in the event that the specific server has no record of the client requested binding. The Interserver messages supporting these operations are the CLIENT BINDING UPDATE and the CLIENT BINDING QUERY messages, respectively. The SCSP records for these operations are 'Binding' records for the update and query messages.

The Address Management operations are UNBINDABLE COMPLETE POLL and TRANSFER. The UNBINDABLE COMPLETE POLL initializes an address as bindable by the LS. The TRANSFER allows for the transfer of a block of bindable addresses between servers. The Interserver messages supporting these operations are the UNBINDABLE QUERY and the TRANSFER messages. The SCSP records for these operations are 'Address' records for the UNBINDABLE QUERY and 'Bindable Block Address' records for the TRANSFER messages.

The Group Management messages are SG DISCOVERY Query, SG CONFIGURATION QUERY, SG PROPOSE CHANGE UPDATE and SG COMMIT CHANGE UPDATE. The SCSP records associated with these operations are 'SG Specifier' records for the SG DISCOVERY QUERY, 'SG Subnets' records for the SG CONFIGURATION QUERY, 'SG Members' records for the SG DISCOVERY Query, and 'SG Proposed Members' records for the SG PROPOSE CHANGE UPDATE and SG COMMIT CHANGE UPDATE messages.

- On DHCP Interserver Authentication:

  The interserver protocol will rely on the authentication extensions within SCSP for the SCSP message authentication between servers within a server group. The authentication of the interserver group management protocol messages are tbd.

o On the Notion of Server Ownership of Binding Records:

It will be assumed that once the initial client binding record is generated by a particular server, that record will indicate that server as the originating server in the SCSP 'Originating Server ID' field. Any further changes to that binding, whether by the originating server or by another server, e.g., the originating server is down and the client is Rebinding and getting a lease extension from another server, that server does change the Originating Server ID in the SCSP record field to indicate itself as the last transaction server.

o On a More Efficient Cache Alignment Process:

The cache alignment process can be made more efficient if the servers time stamp their cache records. In the event that the connections between servers fails, the servers determine and record the failure time. Upon reconnecting and cache alignment, the SCSP CRL list can be limited to those records that are more 'recent' than the failure and therefore greatly reduce the time and the bandwidth required. The details are presented below.

Also, it is not necessary to perform a cache alignment of the address records for the proper operation of the Interserver protocol. Therefore, we assume that the SCSP cache alignment process will not include these address records when building the SCSP CRL.

o On the More Recent Record Determination:

SCSP relies on the ability of identifying the more recent-ness of records when aligning and updating the cache based upon the CSA Sequence Number. For binding records this implies that in situations where it is clear that a single server is updating the binding, e.g., extending the lease, then it should increment the CSA Sequence number by one. However there are situations in DHCP where multiple servers can simultaneously update the client binding and it is not clear which of these updated bindings is accepted by the client, e.g., the client is in the rebinding state and the originating server is down and the other servers received the client broadcast request and the client gets multiple DHCPACKs extending the lease. In these situations the servers are required to increment the CSA sequence numbers by one and indicate that they are the last transaction server. Then, when a server caches the record, if it already has a cache record for that binding (as indicated by the Cache Key) it should replace the existing record only if the new record indicates a lease period which is greater than the existing record.

o On Maximally Defined Binding Records (or the B.Hibbs' Question):

B.Hibbs' posed the question regarding the nature of the configuration synchronization of the servers within the same SG; Does the DHCP Interserver protocol require synchronization of all configuration parameters or a subset? We are assuming that there is a minimal set of configuration and client binding information to be synchronized across the members of the SG to ensure the correct operation of the DHCP Client/Server protocol. This information must be carried in the interserver messages to synchronize the members in the SG with respect to this information. Further, there may be other client binding information that the members want to communicate; we currently have this information encoded as optional in this draft.

The parameters encoded into the 'Client Binding' records are those which are minimally required for the correct operation of the DHCP Client/Server protocol. The interserver protocol should allow for situations where the configuration of the servers of the same server group are not strictly aligned; their configurations are only required to be aligned in the specification of the subnets and masks that are covered with a SG and the list of assignable addresses within each of the subnets. However, because clients DHCPDISCOVER messages can contain client specific requests for parameters, it may be desirable to embed a fuller set of parameters (committed to the client in the DHCPOFFER message) within the CSA record. This fuller set of parameters may be included in the initial CLIENT BINDING COMPLETE PUSH (encoded in the optional fields location in the record). The server in receipt of a CLIENT BINDING COMPLETE PUSH may chose not to cache or forward these optional parameters.

o On Knowledge Obtained Through the SCSP Hello protocol:

The SCSP Hello protocol maintains current status of the interserver connectivity through a polling mechanism. This status information can be used to influence the actions of the LS, e.g., in the event that the LS has lost connectivity from a DCS, then it should not perform a COMPLETE POLL operation.

o On the SG Connectivity:

It is likely that the servers of the SG are required to be fully interconnected, i.e., a LS is a DCS to all other servers of the SG. It was first thought that this would aid in determining the status of the SG, i.e., whether the SG was 'up' (fully functioning) or 'down' (not fully functioning). However on further inspection this is not true, i.e., the loss of connectivity

between a pair of servers in a fully connected SG does not imply that the other servers are not still connected to the other servers. Full mesh connectivity may still be required for the correct operation of the Address Management protocol. This is currently under study.

When a new server wishes to join a server group, it must initialize itself to the other members of the server group through the above defined interserver Group Management Protocol. Once this has occurred, the local server must initiate SCSP which then will align its client binding cache to that of the server group. It should then acquire Bindable addresses and fully participate in the on-going client binding update functions of the server group.

This process is outlined in the below state diagram for the DHCP interserver protocol. The Group Management protocol handles the new server joining the group. Once this has occurred, the new server and all the other servers of the server group initiate the SCSP Hello Protocol on a pairwise basis. Per the discussion in the SCSP specification, once bi-directional connectivity is re-verified and now monitored within the SCSP Hello protocol, the servers enter into the cache alignment and then the ongoing cache and address management functions. In the event that the servers transition to the 'DOWN' state, polling will continue until connectivity is re-established.

The Group Management Protocol does not allow additions to the membership in the event that the SG is down. However it does allow for the removal of a server from the SG while another server is re-booting or disconnected. Therefore a re-booting or re-connecting server cannot be assured that the SG generation has remained constant during the 'DOWN' period. Therefore, in the event that the generation number of the SG has changed as indicated through the generation number contained within the interserver messages, the server needs to update its notion of the server group through the procedures identified in the group management protocol prior to aligning its cache.

```
                           +------------+
                           |   Group    |
                           | Management |
                           |  Protocol  |
                           +------------+
                                 |
                                 |
                                 V
                          +------------+
                          |    SCSP    |
                          |   Hello    |
                          +------------+
                           /     ^   \
                          /      |    \
                         V       |     V
              +--------------+   |    +---------------+
              |'Binding Mgmt'|   |    |Null'Addr Mgmt'|
              |    Cache     |---+----|     Cache     |
              |  Alignment   |   |    |   Alignment   |
              +--------------+   |    +---------------+
                     |           |           |
                     |           |           |
                     V           |           V
              +--------------+   |    +------------+
              |'Binding Mgmt'|   |    | 'Addr Mgmt'|
              | Cache Update |---+----|Cache Update|
              +--------------+        +------------+
```

Figure 8.2-1 Interserver State Flow Diagram

For operational efficiency, the servers should implement a scheme to limit the number of cache records to exchange during the cache align- ment process. For example, a SG could easily be managing 10,000 client records and the bandwidth requirements to pass even the sum- mary records required to build the CRL table can be quite large. Therefore, for the 'Cache Management' sub-protocol, the servers should record the times at which the cache entries were received or created or modified. When the CAFSM transitions for a particular DCS to the down state, t(down) should be recorded. Then when the CAFSM enters the cache alignment state, the CRL list is to be built up based upon only those records with time stamps more recent then t(down) - F, where F is a factor to be set to a multiple of the Hel- loInterval x DeadFactor. We recommend that the multiple be 10. In the event that the LS crashed (causing the transition to the down state), then t(down) should be set to the last record time stamp when the LS reboots. In the event that the server has just joined the SG, the CRL should be built up from all of the current cache records.

The interserver messages associated with the Client Binding Management are: CLIENT BINDING QUERY for the CLIENT BINDING POLL operation, and CLIENT BINDING UPDATE for the CLIENT BINDING COMPLETE PUSH operation. These are discussed in detail in the following list items:

o The CLIENT BINDING QUERY message queries another server regarding the status of a particular binding. Within the SCSP protocol, this exchange is accomplished by the LS sending a Client State Update_Solicit (CSUS) message with the Client State Advertisement Summary (CSAS) 'Address record' of the IP address in question. The DCS responds with the CSU_Request message with the Client State Update (CSU) record associated with the CSAS. The LS then replies with a CSU_Reply with the 'A-bit' set.

o The CLIENT BINDING UPDATE message updates another server with a new, or changed, client binding. Within the SCSP protocol, this exchange is accomplished with the CSU_Request message carrying the specific CSA 'Binding record' of the client binding in question. The DCS responds with the CSA-Reply with the 'A-bit' set.

The interserver messages associated with the Address Management are: UNBINDABLE QUERY for the UNBINDABLE COMPLETE POLL operation, and TRANSFER messages for the TRANSFER operation. These are discussed in detail in the following list items:

o The UNBINDABLE QUERY message queries another server of the SG regarding the status of a particular address with the intent of making that address bindable to the LS. Within the SCSP protocol, this exchange is accomplished by the LS sending a CSU_Solicit with the CSAS 'Address' record of the IP address in question to all other servers of the SG. The DCSes respond with the CSU_Request message with the CSA 'Address' record indicating the status of the address within the DCS. The LS then replies with the CSU_Reply message to the DCS with the 'A-bit' set.

o The 'TRANSFER' operation is initiated by the LS to request a transfer of bindable addresses from the DCS to the LS. Within the SCSP protocol, this exchange is accomplished by a two step process. First, the LS sends a CSU_Request message with the CSA 'Subnet Bindable Addresses' record to the DCS, which then responds with a CSU_Reply. The CSA 'Subnet Bindable Addresses' record indicates the subnet in question, the number of BINDABLE addresses owned by the LS and the number of additional BINDABLE addresses the LS is requesting. Second, this is immediately followed by the DCS sending a CSU_Request message with a CSA 'Subnet Bindable Address' record for the given subnet in question. The DCS' CSA 'Subnet Bindable Addresses' record indicates the subnet

in question and the number and address of the IP addresses that the DCS is transferring to the LS based upon it's previous request. This is based upon the DCS' current understanding of the supply of bindable addresses within the LS and its local knowledge of its own set of bindable addresses for this subnet. This CSU_Request will generate a CSU_Reply from the originating LS. When sending the CSU_Request message, the DCS sets the addresses it is transferring to the LS as UNBINDABLE. The LS then moves these addresses to its list of BINDABLE addresses and sends a CSU_Reply to the DCS with the 'A-bit' set.

The interserver messages associated with the Group Management operations are: SG DISCOVERY QUERY, SG CONFIGURATION QUERY, SG PROPOSE CHANGE UPDATE, and SG COMMIT CHANGE UPDATE messages. These are discussed in detail in the following list items:

o The SG DISCOVERY QUERY message queries the DCS for its list of current SG in which it is participating. Within the SCSP protocol, this exchange is accomplished by the LS sending a CSU_Solicit with the CSAS 'Server Groups' record and the DCS replys with the CSU_Request message containing the CSA 'Server Groups' record. This record contains the list SG specifiers, i.e., SG ID and SG Generation Number (GN) pairs. The LS replies with a CSU_Reply.

o The SG CONFIGURATION QUERY message queries the DCS for its configuration information. This information is passed within the 'SG Subnets Configuration' record. The LS initiates this query by sending a CSU_Solicit containing the CSAS 'SG Subnets Configuration' summary record. The responds with a CSU_Request containing the CSA 'SG Subnets Configuration' record. The LS replies with the CSU_Reply message.

o The SG PROPOSE CHANGE UPDATE message proposes the new member to the rest of the SG. This is accomplished with a SCSP CSU_Req message carrying the 'SG Proposed Members' record. The SG COMMIT CHANGE UPDATE message consummates the new server joining the SG. Once the joining member has received positive CSU_Reply from all of the current members of the SG as part of the proposal phase, it then moves to the join commit phase. The new server now issues an SCSP CSU_Req message with the 'SG Members' record carrying the newly joined member to the list of servers of the SG.

o The SG PROPOSE CHANGE UPDATE message may also be used to propose the removal of an existing server from the membership of the SG. This is accomplished with a SCSP CSU_Req message carrying the 'SG Proposed Members' record containing all of the existing members of the SG minus the server ID to be removed. The SG COMMIT

CHANGE UPDATE message consummates the existing server leaving the SG. Once the removing member, i.e., the member who is actively removing the existing member from the group, has received positive CSU_Reply from all of the current members of the SG (except for the member being removed) as part of the proposal phase, it then moves to the remove commit phase. The removing server now issues an SCSP CSU_Req message with the 'SG Members' record carrying the new membership minus the removed server.

8.3. Necessary Modifications to SCSP

The SCSP modifications required to support the DHCP interserver protocol are as follows:

- The operation of the SCSP protocol in this application is initiated upon the successful completion of the interserver 'Group Management Protocol'.

- The SCSP messages, and in fact all of the DHCP interserver messages are carried in UDP packets. Therefore a UDP port number needs to be defined for SCSP.

  DISCUSSION:

  Currently SCSP is defined only for NMBA networks. This manifests itself in two ways; a) the operation of the SCSP protocol is initiated upon the establishment of NBMA connectivity, i.e., a virtual circuit being established, and b) the SCSP messages are encapsulated into link level frames using the LLC/SNAP encapsulation method.

  Instead of relying upon the establishment of a virtual circuit connection, the interserver protocol will initiate the SCSP protocol based upon the results of the 'Group Management Protocol'. This divorces the operation of the interserver protocol from the specifics of the link layer. Also, by carrying the messages within UDP, the protocol achieves independence in the deployment and proximity of the servers which are members of the same server group, i.e., servers are not required to have an interface on a common subnet.

  Because SCSP provides a generic capability to synchronize caches in distributed servers, it is best to define a separate UDP port number for the 'generic' SCSP protocol and a separate UDP port for the DHCP interserver Group Management protocol. These UPD port numbers are tbd.

- A SG Generation Number SCSP extension field needs to be defined.

  DISCUSSION:

  We have defined the notion of a Server Group Generation Number to distinguish between the various instantiations of a particular SG. The membership of a particular SG will change over time. Because it is necessary for the correct operation of the DHCP interserver protocol for each server to know the current membership, it was deemed necessary to define a Generation Number which is incremented each time a new server joins the SG or an existing server is removed from the SG. This GN is to be carried in every interserver message. No obvious place existed with the SCSP message formats to carry such information. Therefore, we have chosen to define a new SCSP extension type and will carry the GN in this method.

- Some modification to the Authentication extension in the SCSP protocol may be required.

  DISCUSSION:

  Currently SCSP states that the authentication extension covers the SCSP message other than the extensions. However we have chosen to carry a new extension within the SCSP messages; the Generation Number. Ideally we would prefer that this extension be protected by the authentication extension. Because it is not, we will also include the Generation Number in the SG Specifier record. Through this record a server may reverify the current Generation Number through a protected channel.

- The three step Solicit_Request_Reply seems excessive when one server wishes to simply query another server. Perhaps this could be simplified (when desirable) by adding a bit to the CSU_Solicit message indicating whether the soliciting server wishes the DCS to expect or not to expect a CSU-Rep from the soliciting server.

  DISCUSSION:

  Currently SCSP states that the three step process of CSU_Sol followed by a CSU_Req which is then followed by a CSU_Rep. In certain situations this may be a desirable sequence. However, in other situations it may not be necessary. When the CSU_Sol is sent a CSUSReXmtInterval timer is set which tracks the status of the receipt of the requested CSU_Req records. For simply queries, this re-transmit timer may be sufficient. Therefore, it seems reasonable that DCS should expect a CSU_Rep from the LS which sent the CSU_Sol message.

8.4. DHCP Specific CSA and CSAS Records

This section presents the CSA and the CSAS records specific to the DHCP inter-server protocol. The mappings of the interserver protocol onto SCSP messages discussed in the previous section relys upon the definition of a number of record types. These record types will be distinguished within the CSAS defined 'Cache Key', which for the purpose of running the DHCP interserver protocol will consist of a TYPE/Key pair. The following CSAS and CSA record types are required to run the interserver protocol:

For Client Binding Management:

- o Binding Record - contains the complete client binding information.

For Address Management:

- o Address Record - contains the status of a specific IP address, e.g., unbindable, bindable, bound, expired, etc.

- o Subnet Bindable Record - contains information regarding the subnet addresses, e.g., number of bindable addresses.

For Group Management:

- o SG Specifier Record - contains the current Server Group specifiers, i.e., the SG ID (which is fixed for the duration of the life of the SG) and the SG Generation Number which is incremented for each new server add or old server delete.

- o SG Members Record - contains the current list of member servers of the SG.

- o SG Subnets Configuration Record - contains a list of all subnets, i.e., subnet address and mask, for all of the subnets served by the SG as well as the assignable addresses per subnet, and potentially other configuration parameters necessary for the proper operation of the DHCP interserver protocol.

- o SG Proposed Members Record - contains a list of the proposed member servers of the SG used in the group join proposal process. This record has a finite duration associated with it and times out if the proposed join fails.

8.4.1. The SCSP CSAS Records for the Interserver Protocol

The CSAS record is completely specified in [2]. The format of the CSAS record is:

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |         Hop Count             |       Record Length           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | Cache Key Len | Orig ID Len   |N|      unused                 |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                      CSA Sequence Number                      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                    Cache Key    (variable)                    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                      Originator ID  (variable)                |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8.4.1-1 SCSP CSAS Record Format

where:

- o Hop Count - this represents the number of hops that the record may take before being dropped.

- o Record Length - this is the length in bytes of the CSAS record if stand-alone, otherwise it is the length in bytes of the CSAS record and the protocol specific part of the cache entry com-bined, i.e., the length of the CSA record.

- o Cache Key Length - this is the length of the Cache Key field in bytes.

- o Originator ID Length - this is the length of the Originator ID field in bytes.

- o N bit - this bit, when set, signifies a Null record. This may be the case when the LS receives a solicitation for a record that has been released by the DHCP client.

- o CSA Sequence Number - this field contains the sequence number that identifies the 'newness' of a CSA record instance being sum-marized. This number is assigned by the originator of the CSA record, i.e., the last transaction server.

- o Cache Key - is an opaque string used by the receiving server to identify the cache entry referred to by the record. For the purposes of running the DHCP interserver protocol, the Cache Key will be encoded as a Type/Key pair, where the type is an 8 bit field and the length of the Key is derived from the Cache Key Length field in the header. The Type indicates the type of record and equivalently the Interserver message type, e.g., Unbindable Address Query, SG Configuration Query, etc. The 8 bit type encodings are defined in the table below.

- o Originator ID - this field contains an ID which is administratively assigned to the server which is the originator of the CSA record. For the DHCP interserver mapping, the the Originating Server ID is chosen to be the IP address of the server. In the event that the server has multiple IP addresses assigned to it, then the Originating Server ID is set to the IP address with the highest value.

The CSAS record is specified by SCSP except for the specifics of the Cache Key and the Originator ID.

For the purpose of the DHCP interserver specification, the Originating Server ID is chosen to be the IP address of the server. In the event that the server has multiple IP addresses assigned to it, then the Originating Server ID is set to the IP address with the highest value.

The Cache Key used is dependent upon the specific CSAS record in question. The table below identifies the specific Cache Keys for the various CSAS records within the DHCP interserver protocol. These are composed of a type and key field, both of which are identified in the table.

Table 8.4.1-1 Cache Keys for the various CSAS and CSA records

Record Type	Encoding	Key
Client Binding	0x00	Client ID or hwaddr
Address	0x10	IP addr
Subnet Bindable Addrs	0x11	Subnet/Mask *
SG Specifiers	0x20	IP addr
SG Subnet Configs	0x21	SG ID
SG Members	0x22	SG ID/SG GN **
SG Proposed Members	0x23	SG ID/SG GN **

* The subnet address and the subnet mask will be encoded as 32 bit strings with the subnet address followed by the subnet mask.

** The SG ID and SG GN are encoded as 16 bit strings with the SG ID first, immediately followed by the SG GN.

8.4.2. The SCSP CSA Records for the Interserver Protocol

There are several types of DHCP specific CSA records defined corresponding to each of the CSAS record types discussed above and found in Table 8.4.1-1.

For many of these records, DHCP options appear in the records in the same format as specified in [7].

The records are:

- The Client Binding record carries the complete client binding information. The Key for this record is the chaddr or the 'client ID' from the optional DHCP extension. This is utilized in the Cache Mgmt sub-protocol in handling the COMPLETE PUSH, POLL and SCSP cache alignment operations.

- The Address record carries the information required to achieve the desired response from the CSU_Solicit message. The Key is the IP address. This is utilized in the Address Mgmt sub-protocol in handling the UNBINDABLE COMPLETE POLL operation.

- o The Subnet Bindable Address record carries the information required to determine the status of the available IP addresses which are bindable to the DCS and which it is will to transfer to the LS. The Key for this record is the subnet address and mask of the subnet in question. This is utilized in the Address Mgmt sub-protocol by the TRANSFER operation.

- o The SG Specifier record contains the total list of SG specifiers, i.e., SG ID and SG GN pairs, of which the server in question is currently a member. This is utilized in the Group Mgmt sub-protocol by the DISCOVERY operation. The Key for this record is the Server ID, i.e., the IP address of the server.

- o The SG Members record contains a list of the Server IDs which comprise the SG in question. This is utilized in the Group Mgmt sub-protocol by the DISCOVER MEMBERS operation. The Key for this record is the SG Specifier, i.e., the SGID and SG GN pair.

- o The SG Proposed Members record contains a list of the SG members, including the newly proposed member, of the server group. This is utilized in the Group Mgmt sub-protocol by the PROPOSE JOIN operation. The Key for this record is the SG Specifier, i.e., the SGID and SG GN pair where the SG GN is one greater than the current GN of the SG.

8.4.2.1. **Binding Records**

The approach taken in defining the Client Binding record is as follows. It is possible, while still maintaining the correct operation of the DHCP client/server protocol, to have the different server configurations within the same server group with respect to certain parameters. For these parameters we do not require synchronization of the server configurations and we make the passing of these parameters as optional. However there are some configuration parameters and binding information which is critical to the correct operation of the protocol. For these client parameters we require that they be included in the Client Binding records. The minimal, required set of parameters to be sent in the Client Binding are the IP address (ciaddr), the lease period, the last transaction type, the client hardware address, the Client-Identifier and the Renewel (T1) and Rebinding (T2) Time values (if present in the DHCP options extensions of the DHCPACK).

The format of the CSA Binding record for the DCHP inter-server protocol is:

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                      CSAS Record  (variable)                  |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |  LTT  |resrv'd|   HTYPE       |    HLEN       |    resrv'd    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                       CHADDR  (HLEN in octets)                |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                       CIADDR  (4 octet)                       |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                Last Transaction Time (4 octet)                |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |     IP Address Lease Time (encoded as tag=51) (6 octet)       |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |     Optional ClientID (encoded as tag=61) (variable)          |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |     Optional Renewal Time (encoded as tag=58) (6 octet)       |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |     Optional Rebinding Time (encoded as tag=59) (6 octet)     |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |     Other desirable DCHP extensions   (variable)              |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |   End Option (encoded as in BOOTP options, tag=255) (1 octet) |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8.4.2.1-1 DHCP inter-server CSA Binding record format

where:

- o CSAS Record - represents the full CSAS record as identified in Section 8.4.1.

- o LLT - indicates the Last Transaction Type. The allowed LTTs are: DHCPREQUEST/SELECTING (0x0), DHCPREQUEST/REBINDING (0x3), DHCPRE-QUEST/RENEWING(0x2), DHCPREQUEST/INIT-REBOOT (0x1), DHCPRELEASE (0x4), and EXPIRATION (0x5).

- o HTYPE - hardware address type (defined in [1])

- o HLEN - hardware address length

- o CHADDR - client hardware address

- o CIADDR - client IP address (if assigned). If not assigned, this field is all 0s.

- o Last Transaction Time - the time from now in seconds of the last transaction time associated with the LTT as indicated in the message.

- o IP Address Lease Time - the IP Address Lease Time encoded as in the DHCP options and BOOTP vendor extensions defined in [7]. This represents the time from now that the client lease is to expire.

- o (Optional) Client ID - this field is the optional Client ID encoded as in the DHCP options and BOOTP vendor extensions defined in RFC 2132 [7]. If present, the Client ID is the 'search string'.

- o (Optional) Renewal Time - this field is the optional Client Renewal Time (T1) as encoded in the DHCP options and BOOTP vendor extensions defined in RFC 2132 [7].

- o (Optional) Rebinding Time - this field is the optional Client Rebinding Time (T2) as encoded in the DHCP options and BOOTP vendor extensions defined in RFC 2132 [7].

- o Remaining Options - any remaining options carried in the original DHCPOFFER message to the client encoded as in the DHCP options and BOOTP vendor extensions defined in [7]

- o End option - determines the end of the CSAS record

DISCUSSION:

As discussed in the previous section on the CSAS record format, the format shown above is intended to be the Binding type CSA record. The binding record is used in the PUSH and COMPLETE PUSH operations to transfer to the DCSes the newly created or changed binding and in the cache alignment procedures. The structure of the Client Binding is defined, for the propose of the DHCP interserver protocol into a mandatory part and an optional part. The mandatory part is everything upto and including the (Optional) Rebinding Time. The optional part is everything following the (Optional) Rebinding Time. The PUSHing server may include any additional parameters which were part of the DHCPACK message to the client within the Client Binding Record and encode this as defined in the the DHCP options and BOOTP vendor extensions defined in RFC 2132 [7]. The server which is the recipient of the PUSH may chose to save and forward these optional parameters in the record or may chose not to save and forward these optional parameters.

8.4.2.2. Address Records

The format of the CSA Address record for the DCHP inter-server protocol is:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      CSAS Record  (variable)                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  ST   |                       reserved                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8.4.2.2-1 DHCP inter-server CSA Address record format

where:

- o CSAS Record - represents the full CSAS record as identified in Section 8.4.1.

- o ST - represents the state of the (client) record, e.g., unbindable, bindable, bound, expired, polling, static

  DISCUSSION:

  The Address record is used within the UNBINDABLE COMPLETE POLL operation to move an unbindable address to a bindable address. The POLLed server returns the Address record indicating the current status of the address within the server. If all of the servers indicate that the address is unbindable, then and only then will the LS move the address to its Bindable pool.

  The ST field indicates the servers view of the state of the address. The states (defined in Section 3.4.2) are: UNBINDABLE, POLLING, BINDABLE, BOUND, PUSHED, and EXPIRED.

The IP address states are encoded in the following manner:

Table 8.4.2.2-1 IP Address State Encodings

IP Address State	Encoding
UNBINDABLE	0x01
POLLING	0x02
BINDABLE	0x03
BOUND	0x04
PUSHED	0x05
EXPIRED	0x06

8.4.2.3. **Subnet Bindable Addresses Record**

The CSA Subnet Bindable Addresses record indicates the set of addresses that a server is willing to TRANSFER to a requesting server. This record is used in the TRANSFER operation.

The format of the CSA Subnet Bindable Addresses record for the DCHP inter-server protocol is:

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                      CSAS Record  (variable)                  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | No. Addresses |No. Addr.Ranges|R|  reserved   |No.Ownd|No.Reqd|
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |               List of IP Addresses                            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8.4.2.3-1 DHCP inter-server CSA Subnet Bindable Addresses record format

where:

- o CSAS Record - represents the full CSAS record as identified in Section 8.4.1.

- o No. Address - indicates the number of IP addresses contained within the subnet record. These are the addresses that the DCS is transferring to the LS as part of the TRANSFER operation. This is set to 0 when the R-bit is set to 1 (see R-bit below).

- o No. Addr. Ranges - indicates the number of IP address ranges of the form 135.16.114.5 to 135.16.114.235. These will immediately follow the listing of the individual addresses. This is set to 0 when the R-bit is set to 1 (see R-bit below).

- o R - represents the request bit. When this bit is set to 1, it indicates that the LS is requesting BINDABLE addresses from the DCS as part of the TRANSFER operation. When it is set to 0, it indicates that the DCS is transferring these addresses to the LS.

- o No. Ownd - indicates the current number of BINDABLE addresses owned by the LS when the R-bit is set to 1.

- o No.Reqd - indicates the number of additional BINDABLE addresses requested by the LS when the R-bit is set to 1.

- o List of IP Addresses - this is a consecutive list of IP address and address ranges.

DISCUSSION:

The Subnet record is used in the TRANSFER operation to indicate 1) the list of bindable IP addresses that the DCS is willing to transfer to the LS when the R bit is 0, and 2) the IP addresses that the LS is requesting when the R bit is 1.

Further, it may be useful to develop similar records for Subnet UNBINDABLE, BOUND, PUSHED, and EXPIRED address. They can have an identical record format and be distinguished through the 8 bit type field encoded into the SCSP Cache Key. The utility of these record types is TBD.

8.4.2.4. SG Specifier Record

The CSA SG Specifier Record indicates the total list of DHCP Interserver protocol Server Groups that the DCS is currently a member. This is used in the Group Management subprotocol during the initial contact of a prospective new member to the Server Group.

The format of the CSA SG Specifier Record for the DCHP inter-server protocol is:

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                      CSAS Record  (variable)                  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |No. Specifiers |             reserved                          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |               List of Specifier Pairs                         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8.4.2.4-1 DHCP inter-server CSA SG Specifiers record format

where:

- o CSAS Record - represents the full CSAS record as identified in Section 8.4.1.

- o No. Specifiers - is a count of the number of specifier pairs contained within this CSA record.

- o List of Specifier Pairs - represents a consecutive listing of the specifier pairs of which the DCS is current a mamber. The encoding of the specifier pairs is SG ID first, which is a 16 bit string, followed by the SG Generation Number, which is also a 16-bit string.

  DISCUSSION:

  This record is initially requested by a server which is interested in joining a DHCP Interserver Server Group and has been configured with the IP address of a server to first contact. The first contacted server then replies with the SG Specifier record. This record can also be solicited when a server, which an existing member of a group becomes uncertain regarding the current Generation Number of the group.

  The SG Generation Number, obtained from this record, is carried in every DHCP Interserver protocol message, encoded as an extension to the SCSP message extension fields. The extension encoding is TBD.

8.4.2.5. **SG Subnets Configuration Record**

The CSA SG Subnet Configuration Record carries SG configuration information necessary to ensure the correct protocol operation of the group. The encoding of this record is essentially the subnet address and mask followed by the pool of addresses which are dynamically

managed by the Server Group for this subnet. The encoding of the address pool with be consistent with the address pool encoding of the Subnet Bindable Addresses Record discussed in Section 8.4.2.3 above. Other configuration parameters may be including if deemed important to the correct operation of the DHCP interserver protocol.

Section 7.2 specifies that additional information (specifically client configuration information and vendor specific configuration information) will be also be available. The precise details of how this information is encoded is TBD.

The format of the CSA SG Subnets Configuration Record for the DCHP inter-server protocol is:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      CSAS Record  (variable)                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | No. Subnets |               reserved                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Subnet Address                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Subnet Mask                                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Address Pool of first subnet (variable)    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Subnet Address                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                               ...
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Address Pool of last subnet  (variable)    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8.4.2.5-1 DHCP inter-server CSA SG Subnets Configuration record format

where:

- o CSAS Record - represents the full CSAS record as identified in Section 8.4.1.

- o No. Subnets - indicates the number of subnet configurations contained in this record.

- o Subnet Address - this is the subnet address of the subnet for which the following address pool is related.

- o Subnet Mask - this is the mask of the subnet in question.

- o Address pool of subnet - this is a listing of the address pool for which this SG can allocate from for this particular subnet. The encoding will follow the address pool encoding for the Subnet Bindable Addresses record. Therefore, the address pool should contain two count fields, the first indicating the number of individually listed addresses, followed by another field indicating the number of address ranges. These are then followed by the list of individual IP addresses and then the list of address ranges.

  DISCUSSION:

  The total list of configuration items to be incorporated into this record needs to be further fleshed out. Currently this record is planned to contain a list of the subnets and the address pools associated with each from which this SG can allocate. If other configuration parameters are deemed necessary for the proper operation of the DHCP Interserver protocol, then these need to be incorporated into this record.

8.4.2.6. SG Members Record

The CSA SG Members Record indicates the list of the current SG members, in the opinion of the sending server, including itself.

The format of the CSA SG Members Record for the DCHP inter-server protocol is:

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                      CSAS Record  (variable)                  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | No. Server IDs|P|             reserved                        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                      List of Server IDs                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8.4.2.6-1 DHCP inter-server CSA SG Members record format

where:

- o CSAS Record - represents the full CSAS record as identified in Section 8.4.1.

- o No. Server IDs - this is the number of Server IDs contained within this record.

- o P bit - the Proposal bit is used to indicate that this record is a current group members record (here set to 0) or a proposed group members record (discussed in the next section).

- o List of the Server IDs - this is a consecutive list of Server IDs which comprise this server's view of the current SG membership. The Server IDs are IP addresses associated with one of the server's interfaces.

8.4.2.7. SG Proposed Members Record

The CSA SG Proposed Members Record indicates the list of the current SG members, in the opinion of the sending server, and adding itself. This is a temporary record (with a lifetime associated with the period during which a Group Management SG CHANGE operation has to complete). Once the SG COMMIT CHANGE UPDATE is received, this record replaces the old SG Members record as the new member record containing the newly joined server.

The format of the CSA SG Proposed Members Record for the DCHP inter-server protocol is:

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                      CSAS Record  (variable)                  |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | No. Server IDs|P|           reserved                          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |               List of Server IDs                              |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8.4.2.7-1 DHCP inter-server CSA SG Proposed Members format

where:

- o CSAS Record - represents the full CSAS record as identified in Section 8.4.1.

- o No. Server IDs - this is the number of Server IDs contained within this record.

- o P bit - the Proposal bit is used to indicate that this record is a proposed group members record (here set to 1) or a current group members record (discussed in the previous section).

- o List of the Server IDs - this is a consecutive list of Server IDs which comprise the sending server's view of the proposed SG membership. The Server IDs are IP addresses associated with one of the server's interfaces.

  DISCUSSION:

  This record contains the proposed group membership from the view of the proposing server. This record conceptually has a temporary lifetime associated with the period for which a group join proposal can live. If a server receives a SG COMMIT CHANGE UPDATE message, then this record becomes the new SG Members record. If a SG COMMIT CHANGE UPDATE message is not received within the appropriate period, then this record expires. If the server receives a second SG PROPOSE CHANGE UPDATE message while another Proposed Members record is active, it should NAK this second Proposed Members record. Only one group join can be in process at any given time.

8.5. Open Questions with the Mapping onto SCSP

The following questions are identified as outstanding issues to be resolved for the CSAS and CSA record definitions to be considered complete:

- o SCSP is currently LLC/SNAP encapsulated. We are proposing that a UDP port be defined to carry SCSP messages for DHCP. In fact we are proposing that the entire DHCP interserver protocol be run over UDP.

- o SCSP has currently reserved its Protocol ID = 4 for DHCP. This draft discusses DHCPv4 Interserver protocol and therefore the SCSP Protocol ID reservation should reflect that fact. If a DHCPv6 extension to this draft were developed it would require a separate SCSP Protocol ID.

- o SCSP dropped support for message fragmentation. We need to look into the size required for the various records defined in this draft and, if necessary, consider how to handle records larger than can fit into a single UDP packet.

- o Need to give further thought to the partitioning of the DHCP interserver protocol into three separate but related subprotocols; the Group Management, the Binding Management and the Address Management subprotocols. Currently this draft has these as separate subprotocols, with the Group Management subprotocol run separate from the SCSP protocol and in fact on a different UDP port as the SCSP protocol. The Group Management does however share common message semantics and syntax with the SCSP messages in order to simplify parsing the various messages associated with the DHCP interserver protocol. The Binding Management and the Address Management subprotocols are run on top of SCSP with a single Protocol ID.

- o We need to explicitly discuss the method used to authenticate the DHCP Interserver protocol messages. Current thinking is to use the SCSP authentication extensions. This should be investigated and should be consistent with the 'Security Architecture for DHCP' draft [8].

9. IP Address State Transitions

The possible states of an IP address were defined in Section 3.2.2, and the state transition diagram appears there. The state transitions though which an IP address can move were discussed implicitly in Section 6 in the context of the receipt of DHCP messages from DHCP clients. However, an explicit examination of the processing required of a server by this protocol on each of the state transitions will serve to highlight some important aspects of this protocol.

The IP address state transitions are handled in the following way:

- o UNBINDABLE -> POLLING

  When a server attempts to make a particular IP address BINDABLE, it first moves that IP address into the POLLING state. Once in this state, if queried about whether that IP address is UNBINDABLE, the server will reply negatively.

- o UNBINDABLE -> BOUND

  When a server is removed from a server group, all of the IP addresses must be scanned to see if any of them show that server as the server who performed the last transaction (as set by that server successfully completing a CLIENT BINDING COMPLETE PUSH). For all of those IP addresses, if there is a client recorded in the IP address, and if that client does not have a currently different binding, then that IP address must be set to BOUND and the lease time must be reset to the value sent in the latest CLIENT

BINDING COMPLETE PUSH.

The only states from which this transition will be made are UNBINDABLE and EXPIRED.

o POLLING -> BINDABLE

A fundamental point and guarantee of this state transition diagram is that for an IP address to move from the UNBINDABLE state (where it is not owned by any server) through the POLLING state and on to the BINDABLE state (where it is owned by a single server) requires the server seeking to own the IP address to contact all of the other servers in the group. It requires an UNBINDABLE COMPLETE POLL to complete successfully.

The server attempting to move an IP address from the UNBINDABLE through the POLLING and on to the BINDABLE state must ask every other server in the group if it believes that the IP address is currently UNBINDABLE using an UNBINDABLE COMPLETE POLL. If any server says that the IP address is either BINDABLE (i.e., it currently owns the IP address) or BOUND (i.e., a client currently owns the IP address), then the server attempting to move the IP address from the UNBINDABLE to BINDABLE state MUST abandon the attempt. If any server fails to respond at all, the server MUST abandon the attempt as well.

DISCUSSION:

> In addition (and this is important!) if the server attempting to move the IP address from the UNBINDABLE state through the POLLING state and on to the BINDABLE state fails to hear from some other server, then the attempt cannot complete. This means that if a server cannot communicate with every other server (due to communications failure, transient server failure, or network partition) then this state transition cannot be made.

Thus, all addresses in the UNBINDABLE state will stay in that state while any server in the group is out of communication with the group for any reason at all.

Of course, the detailed description of the protocol suggests that a server build up a supply of BINDABLE IP addresses so that in the event of server failure it has BINDABLE addresses that are available to offer to new DHCP clients.

o BINDABLE -> BOUND

Once an IP address is BINDABLE it may be BOUND to a client through the normal actions of the DHCP protocol. Once a server has received a DHCPREQUEST/SELECTING message from a client it can move the IP address into the BOUND state, update its stable storage, and reply with a DHCPACK message to the client.

After the DHCPACK has been sent, the DHCP server MUST also attempt to update all servers in the group with information indicating that the IP address is now BOUND to a particular client. It must perform a CLIENT BINDING COMPLETE PUSH operation with this information.

An IP address that is BOUND will always result in a lease time that is no greater than the MAXIMUM-UNPUSHED-LEASE-TIME when given to a client, although the normal lease time is used in all interactions with other servers.

DISCUSSION:

In an ideal world, the server who created the binding would always succeed in updating all other servers in the group with the binding information. Then, in the event that the binding server failed at some later time, another server to whom the client could broadcast would receive a DHCPREQUEST/REBINDING request and could reply with updated binding information.

However, there is obviously a window where a server can crash after sending a DHCPACK and prior to updating even one additional server. This protocol has been designed so that not only is the process of updating all of the servers in the group with information concerning a new binding "lazy" (i.e., performed after the actual binding is made), but also unnecessary for correct operation. The protocol only requires that a server try to update the other servers -- not that it succeed at updating even one server.

The protocol accomplishes this by allowing a server to respond to a DHCPREQUEST/REBINDING message from a client without any information having been propagated from the server who created the binding. Thus, a server who receives a rebinding request for an IP address about which it has no information must check with all available servers in the group, but in the absence of information to the contrary arriving within a relatively short timeout period, the server should respond to the rebinding request with an extension of the existing lease on the IP address.

- BINDABLE -> UNBINDABLE

  A server can relinquish an IP address in the BINDABLE state that it owns simply by responding to requests for information about the IP address as if it were UNBINDABLE. No explicit action need be taken other than to respond correctly to POLL operations from other servers.

- BOUND -> PUSHED

  Once an IP address that is BOUND to a client has a CLIENT BINDING COMPLETE PUSH succeed (and that means succeed to all of the servers), then it moves from the BOUND to the PUSHED state. At this point, the normal lease time may be returned to the client on the next renewal or discover or rebinding.

  Note that only the server which executes the CLIENT BINDING COMPLETE PUSH will set its IP address into the PUSHED state. The state that it PUSHes to the other servers is BOUND.

- BOUND -> UNBINDABLE

  In order for an IP address to move from the BOUND to the UNBINDABLE state, the client that owns the IP address (i.e., to which it is BOUND) must send a DHCPRELEASE message. In this case, the receiving server (which may or may not be the server who created original binding) will update its stable storage with information that the IP address is not currently BOUND by any client. It should then transmit this information to all other servers to which it can communicate at that time by performing a CLIENT BINDING COMPLETE PUSH operation.

  In the event that the server fails to update any other server with the new information about the IP address prior to undergoing some failure, then the worst that will happen is that the other servers will believe that an IP address is in the BOUND state when it need not be. Ultimately the lease on the IP address will expire.

- BOUND -> EXPIRED

  Any server which has information concerning a BOUND IP address may determine that the lease on the IP address has expired, and after an appropriate grace period has elapsed, that the IP address should be moved to the EXPIRED state. A record of the client to which the IP address was BOUND must be kept.

- o PUSHED -> UNBINDABLE

  In order for an IP address to move from the PUSHED to the UNBIND-ABLE state, the client that owns the IP address (i.e., to which it is BOUND) must send a DHCPRELEASE message. In this case, the receiving server (which may or may not be the server who created original binding) will update its stable storage with information that the IP address is not currently BOUND by any client. It should then transmit this information to all other servers to which it can communicate at that time by performing a CLIENT BINDING COMPLETE PUSH operation.

  In the event that the server fails to update any other server with the new information about the IP address prior to undergoing some failure, then the worst that will happen is that the other servers will believe that an IP address is in the PUSHED state when it need not be. Ultimately the lease on the IP address will expire.

- o PUSHED -> EXPIRED

  Any server which has information concerning a PUSHED IP address may determine that the lease on the IP address has expired, and after an appropriate grace period has elapsed, that the IP address should be moved to the EXPIRED state. A record of the client to which the IP address was PUSHED must be kept.

- o EXPIRED -> UNBINDABLE

  If any server asks for information concerning this IP address, then the receiving server should set the IP address to be UNBIND-ABLE, update its stable storage, and respond to the requesting server.

- o EXPIRED -> BOUND

  If a server receives a message from a client and the IP address is EXPIRED, but was last BOUND or PUSHED to that client, then the IP address can be moved back into the BOUND state. This is possible because no other server can have attempted to make this IP address BINDABLE. If it had, the IP address would not be in the EXPIRED state anymore, but in the UNBINDABLE state (see the EXPIRED -> UNBINDABLE transition above).

  Another reason this transition can occur is as follows. When a server is removed from a server group, all of the IP addresses must be scanned to see if any of them show that server as the server who performed the last transaction (as set by that server

successfully completing a CLIENT BINDING COMPLETE PUSH). For all of those IP addresses, if there is a client recorded in the IP address, and if that client does not have a currently different binding, then that IP address must be set to BOUND and the lease time must be reset to the value sent in the latest CLIENT BINDING COMPLETE PUSH.

The only states from which this transition will be made are UNBINDABLE and EXPIRED.

10. Security Considerations

Minimal security would be provided by configuring every server in a group with the IP addresses of the allowable servers that could ever join that group.

Some additional security is created by using the SCSP security mechanism, although there are limitations to that for other than the client binding management part of the protocol.

Other, more powerful security approaches are and must be addressed prior to further progress on this protocol.

11. Open Questions

The following open questions set off by the "*" character remain from Ralph Droms' original draft: draft-ietf-dhc-interserver-00.txt. Comments have been added in square brackets []. Additional open questions new to this draft are listed with the "o" character.

* Each server must know all other servers.

  Requiring each server to know about every other server imposes additional administrative overhead in the configuration of DHCP servers. However, this configuration overhead is probably minimal relative to any other configuration required for DHCP servers.

  [The group management messages in Section 7 provide a step towards an answer here. A server needs to know only one other server.]

* Each server must contact all other servers before reassigning an address.

[This is fundamental if we wish to use the "lazy synchronization" mode -- you can't get one without the other.]

There is a potential issue here in which no new DHCP clients can be configured if any of the DHCP servers cannot be contacted. Servers can mitigate this problem by maintaining a list of pre-checked addresses that can be allocated without contacting all other servers at the time of address allocation.

The protocol may need additional definition of specific actions on the part of DHCP servers in response to situations in which a server cannot contact all other servers. [Added a lot of these in this draft.]

* Servers cooperating to achieve "fair" distribution of available addresses.

  The protocol may need additional mechanisms or definition of default behavior through which servers cooperate among themselves to ensure that each has a sufficient pool of prechecked-addresses on each network.

  [Not yet addressed, and needs work. Initial thinking is that all addresses should be allocated to some server, so that if the event of a SG where one member can't be contacted, the maximum addresses are available for TRANSFER operations as necessary.]

* User intervention in case of database incoherency.

  Fixing the collective database on the DHCP servers in case of a problem could be a *real* nightmare.

* Potential deadlock in checking address - suppose two servers check the same address for reassignment simultaneously?

  [Solved with the introduction of the POLLING state.]

* Potential configuration for new server?

  One ancillary use of the inter-server protocol might be in con-figuring new DHCP servers. Suppose the inter-server protocol were extended to allow download of a server's configuration file and to allow addition of a new server to the list of DHCP servers. A new server might be configured by simply giving it the address of an existing server. The new server could then download a list of all other known servers, the pool of candidate addresses, any special configuration information (e.g., vendor class information) and the existing bindings. The new server

could also announce itself to all of the other existing servers.

[Much of this is in the current draft, principally in the group management configuration messages. At this stage, a server can figure out which groups correspond with which subnets, which addresses that group manages on that subnet, and some additional configuration information. This is considerable distance towards both ensuring that all servers in the SG have compatible configurations, as well as towards one server downloading configuration data from another server.

Downloading configuration files would not be a great idea for servers which don't use configuration files.]

* DHCP server maintenance

  There is likely an opportunity for the development of a server management tool that would download the database information from all servers and check for conflicts/inconsistencies such as assignment of an IP address to multiple clients, bindings that are not replicated across all servers, bindings that have inconsistent lease expiration times, etc.

o Group-id selection.

  The group-id's for various groups need to be sufficiently unique that no server will ever be a member of two groups with the same group-id. No mechanism is provided yet in this protocol to generate group-id's which conform to this requirement.

  Possibly a group-id can be synthesized in some manner to ensure that they conform to this requirement.

o The original draft discussed the requirement for each server to have a synchronized clock using available time synchronization protocols. That requirement has been removed in this draft, and in its place all times are sent in "seconds from now" as a signed 32 bit number. There is clearly a bit of additional complexity required to do this, but we have been so impressed at how well DHCP works with "relative" instead of "absolute" time that we felt the complexity of using relative time worth it (since using synchronized time is not without its own complexities).

o UNAVAILABLE IP addresses

  There are several cases where a server can determine that some sort of serious error has occurred, and apparently an IP address is in an inconsistent state. In these cases, the server should

make the IP address UNAVAILABLE -- i.e., no other server should be able to operate on it. Just what is necessary to make this happen? Could it be a passive response to address information messages, or must it involve a complete push to all of the other servers, and a new IP address state?

12. Acknowledgments

Many of the ideas in this proposal are due to Jeff Mogul, Greg Minshall, Rob Stevens, Walt Wimer, Ted Lemon and the DHC working group. Thanks to all who have contributed their ideas and participated in the discussion of the inter-server protocol.

At American Internet, Brad Parker and Mark Stapp have been key contributors to the design discussions that have resulted in our contributions to the this draft. They have each invested many hours of work in this protocol.

13. References

[1] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.

[2] Luciani, J., Armitage, G., Halpern, J., "Server Cache Synchronization Protocol (SCSP)", draft-ietf-ion-scsp-01.txt.

[3] Moy, J. "OSPF Version 2", IETF RFC1247, July 1991.

[4] Luciani, J., "A Distributed NHRP Service Using SCSP", draft-ietf-ion-scsp-nhrp-00.txt.

[5] Luciani, J., Fox, B., "A Distributed ATMARP Service Using SCSP", draft-ietf-ion-scsp-atmarp-00.txt.

[6] Reynolds, J., Postel, J., "Assigned Numbers", Internet STD 2, Internet RFC 1340, USC/Information Sciences Institute, July 1992.

[7] Alexander, S., Droms, R., "DHCP Options and BOOTP Vendor Extensions", Internet RFC 2132, March 1997.

[8] Gudmundsson, Olafur, "Security Architecture for DHCP", draft-ietf-dhc-security-arch-00.txt.

14. Author's information

Kim Kinnear
American Internet Corporation
4 Preston Ct.
Bedford, MA 01730-2334

Phone: (617) 276-4587
EMail: kinnear@american.com

Robert G. Cole
AT&T Laboratories
Managed Network Solutions Division
Rm. 3L-533
101 Crawfords Corner Road
Holmdel, NJ 07733

Phone: (908) 949-1950
EMail: rgc@qsun.att.com

Ralph Droms
Computer Science Department
323 Dana Engineering
Bucknell University
Lewisburg, PA 17837

Phone: (717) 524-1145
EMail: droms@bucknell.edu

Appendix A: An Overview of SCSP

This appendix presents an overview of the SCSP protocol and supplements Section 8.2 in the main text of this specification. For a complete discussion of the SCSP protocol see [2].

This appendix is divided into three following sections on the SCSP Hello, Cache Alignment and Cache Update subprotocols respectively. The last section of this appendix presents a summary of the SCSP message sets.

A.1 The SCSP "Hello" Sub-protocol Overview

The function of the SCSP "Hello" protocol is to monitor the status of the LS to DCS connection. The LS must be configured with the addresses of its DCSs. The protocol contains a 'Family ID' which allows for the multiplexing of multiple protocol specific SCSP implementations to rely on a single Hello mechanism between each server pair. For each DCS (whether the low level connection is point-to-point or point-to-multipoint), the LS maintains an Hello Finite State Machine (HFSM). The HFSM is shown in the figure below.

```
                          +---------------+
                          |               |
                 +------->|     DOWN      |<-------+
                 |        |               |        |
                 |        +---------------+        |
                 |            |       ^            |
                 |            |       |            |
                 |            |       |            |
                 |            |       |            |
                 |            V       |            |
                 |        +---------------+        |
                 |        |               |        |
                 |        |    WAITING    |        |
                 |     +--|               |--+     |
                 |     |  +---------------+  |     |
                 |     |    ^           ^    |     |
                 |     |    |           |    |     |
                 |     V    |           |    V     |
               +---------------+     +---------------+
               |  BIDIRECTION  |---->|  UNIDIRECTION |
               |               |     |               |
               |  CONNECTION   |<----|  CONNECTION   |
               +---------------+     +---------------+
```

Figure A.1-1 The Hello Finite State Machine

Key:

1: Link layer connection is established

2: Transition based upon the receipt of a Hello message (and whether the LS ID is found in the Rec ID portion of the message

3: Hello Interval * Dead Factor exceeded

4: Loss of link layer connectivity

The LS to DCS connections are initialized into the down state. The numbers in the figure refer to the actions discussed in the Key that cause a transition in the HFSM (Note: These numbers didn't appear in the original figure in [2], and are TBD). The Hello protocol employs poll messages to monitor the status of the LS to DCS connections.

The Hello messages contain the ID s of the DCS s that the LS has received a Hello message from. The LS' HFSM uses these ID s to determine the status of the HFSM for each of the DCS s. Multiple DCS ID s are present in order to support point-to-multipoint connections. The messages also contain two fields; the Polling Interval and the Dead Factor. The product of the Polling Interval and the Dead Factor determines the length of time that the HFSM will hold open a connection without receiving a Hello from a peer DCS and transitioning the HFSM for that DCS to the Wait state.

A.2 The SCSP "Cache Alignment" Sub-protocol

The Cache Alignment protocol supports the initial server cache synchronization process of an LS with its DCSs. This process may occur at initial boot time of the server, at reconnect time of the server to the network, or other possible initialization or failure recovery scenarios. Like the Hello protocol, the Cache Alignment (CA) protocol maintains a Cache Alignment Finite State Machine (CAFSM) for each of its DCSs to monitor the status of its cache alignment. The figure below shows the CAFSM and indicates some of the triggers that would cause the state transitions to occur.

```
                      +------------+
                      |            |
                 +--->|    DOWN    |
                 |    |            |
                 |    +------------+
                 |          |
                 |          |
                 |          V
                 |    +------------+
                 |    |Master/Slave|
                 |----|            |<---+
                 |    |Negotiation |    |
                 |    +------------+    |
                 |          |           |
                 |          |           |
                 |          V           |
                 |    +------------+    |
                 |    |   Cache    |    |
                 |----|            |----|
                 |    | Summarize  |    |
                 |    +------------+    |
                 |          |           |
                 |          |           |
                 |          V           |
                 |    +------------+    |
                 |    |   Update   |    |
                 |----|            |----|
                 |    |   Cache    |    |
                 |    +------------+    |
                 |          |           |
                 |          |           |
                 |          V           |
                 |    +------------+    |
                 |    |            |    |
                 +----|  Aligned   |----+
                      |            |
                      +------------+
```

Figure A.2-1 Cache Alignment Finite State Machine

Key:

     1: When HFSM reaches Bi-directional state

2: HFSM transitions out of Bi-directional state

3: Master/Slave relationship is established

4: Once both LS and DCS exchange CA messages, both with O-bit set to 0, then CRL is complete

5: E.g., Errored sequence number

6: Full cache update achieved

(Note: The key numbers don't appear in the figure in [2],a and are TBD.)

Each of the CAFSMs is coupled with the respective HFSMs in the LS. The CAFSM is initialized in the Down state. It transitions to the Master/Slave Negotiation state when the corresponding HFSM transitions to the Bi-Directional state. The CAFSM transitions back to the Down state in the event that the corresponding HFSM transitions out of the Bi-Directional state.

In the Master/Slave state the LS-DCS pair negotiate who is to be the master of the connection during the cache alignment process. In the Cache Summary state the LS/DCS pair exchange Client State Advertisement Summary (CSAS) records within the CA messages. The servers use these message exchanges to build a Client State Advertisement Request List (CRL). The CRL indicates the portions of the respective server caches that are out of alignment. The cache mis-alignment (as indicated in the local CRL) is resolved in the Update Cache state where the servers exchange full client state information in CSA records within the CSU messages, only where mis-alignment occurs. Once the CRL is resolved, the LS/DCS caches are aligned and the CAFSM transitions to the Aligned state.

The protocol further defines the high-level syntax of a generic CA message as discussed in a later section of this appendix.

A.3 The SCSP "Client State Update" Sub-protocol Overview

The purpose of the Client State Update (CSU) protocol is to provide a capability to constantly update the server caches through asynchronous CSU message exchanges. These updates are necessary because the status of the clients are in constant flux. Unlike the other two sub-protocols, the Client State Update protocol does not maintain a separate finite state machine. Instead, the activity of this protocol is tied to the CAFSM.

Each CSU can contain zero or more Client State Advertisement records.

The LS may send and receive CSUs when the corresponding CAFSM is in either the Aligned or the Cache Update states. The CSU protocol defines both CSU requests and reply messages. As consistent throughout the definition of the SCSP, the CSU protocol supports both point-to-point and point-to-multipoint connections.

A.4 The SCSP Message Set Overview

The structure of the SCSP messages is a)a fixed length, generic header, b) a SCSP message specific part header of variable length, c) an fixed length, message field and d) zero, one or more SCSP message specific records. This is shown in the following figure.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | Version       |   type        |       Packet Size             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |     IP Checksum               |    Start of Extensions        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |          SCSP Message Specific part (variable)                |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |       Protocol ID             |           SG  ID              |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |       unused                  |          Flags                |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | Sender ID Len | Recvr  ID Len |       No. of Records          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                Sender ID   (variable)                         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                Receiver ID   (variable)                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |         SCSP Message Specific Records   (variable)            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A.4-1 SCSP Message Format

where

o Version - is the version of the SCSP protocol defined in [2]

o type - represents the SCSP message type, i.e., CA, Hello, CSU_Req, CSU_Reply, and CSU_Solicit

o Packet Size -

The SCSP messages have identical syntax except for the 1) the SCSP message specific part header and 2) the SCSP message specific part record. The following table summarizes the content of these specific parts:

Table A.4-1 SCSP Message Specific Parts

	Hello	CA	CSUS	CSU_Req	CSU_Reply
SCSP mesg spec header	hello int, dead fac., Family ID	CSA Seq.No.	null	null	null
SCSP mesg spec record	Additional Recvr ID records	CSAS Rec.	CSAS Rec.	CSA Rec.	CSAS Rec.

The detailed formats of the various SCSP messages are given in [2]. However, two SCSP message specific records are of particular interest to the development of the DHCP interserver specification. These are: 1) the CSAS record and 2) the CSA record. The CSAS record is defined within the SCSP specification as:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         Hop Count             |       Record Length           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Cache Key Len | Orig ID Len   |N|      unused                 |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      CSA Sequence Number                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Cache Key    (variable)                    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      Originator ID  (variable)                |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A.4-2 SCSP CSAS Record Format

See Section 8.4.1 for details.

The CSA record is defined within the SCSP specification as:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                         CSAS Record                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          Client/Server Protocol Specific Part Cache Entry     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure A.4-3 SCSP CSA Record Format

The CSA records for the DHCP interserver mapping to SCSP are defined in Section 8.4.2.

[end of document <draft-ietf-dhc-interserver-02.txt>]