Dynamic Host Configuration (DHC)
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2013
T. Mrugalski
ISC
K. Kinnear
Cisco
July 4, 2012

DHCPv6 Failover Design
draft-ietf-dhc-dhcpv6-failover-design-00

Abstract

DHCPv6 defined in [RFC3315] does not offer server redundancy. This document defines a design for DHCPv6 failover, a mechanism for running two servers on the same network with capability for either server to take over clients' leases in case of server failure or network partition. This is a DHCPv6 Failover design document, it is not protocol specification document. It is a second document in a planned series of three documents. DHCPv6 failover requirements are specified in [I-D.ietf-dhc-dhcpv6-failover-requirements]. A protocol specification document is planned to follow this document.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 5, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Glossary

This is a supplemental glossary that should be combined with definitions in Section 3 of [I-D.ietf-dhc-dhcpv6-failover-requirements].

- Failover endpoint - The failover protocol allows for there to be a unique failover 'endpoint' per partner per role per relationship (where role is primary or secondary and the relationship is defined by the relationship-name). This failover endpoint can take actions and hold unique states. Typically, there is a one failover endpoint per partner (server), although there may be more. 'Server' and 'failover endpoint' are synonymous only if the server participates in only one failover relationship. However, for the sake of simplicity 'Server' is used throughout the document to refer to a failover endpoint unless to do so would be confusing.

- Failover transmission - all messages exchanged between partners.

- Independent Allocation - a prefix allocation algorithm to split the available pool of resources between the primary and secondary servers that is particularly well suited for vast pools (i.e. when available resources are not expected to deplete). See Section 6.2 for details.

- Primary Server

- Proportional Allocation - a prefix allocation algorithm to split the available free leases between the primary and secondary servers that is particularly well suited for more limited resources. See Section 6.1 for details.

- Resource - an IPv6 address or a IPv6 prefix.

- Responsive - A server that is responsive, will respond to DHCPv6 client requests.

- Secondary Server

o Server - A DHCPv6 server that implements DHCPv6 failover. 'Server' and 'failover endpoint' as synonymous only if server participates in only one failover relationship.

o Unresponsive - A server that is unresponsive will not respond to DHCPv6 client requests.

3. Introduction

The failover protocol design provides a means for cooperating DHCPv6 servers to work together to provide a DHCPv6 service with availability that is increased beyond that which could be provided by a single DHCPv6 server operating alone. It is designed to protect DHCPv6 clients against server unreachability, including server failure and network partition. It is possible to deploy exactly two servers that are able to continue providing a lease on an IPv6 address [RFC3315] or on an IPv6 prefix [RFC3633] without the DHCPv6 client experiencing lease expiration or a reassignment of a lease to a different IPv6 address in the event of failure by one or the other of the two servers.

This protocol defines active-passive mode, sometimes also called hot standby model. This means that during normal operation one server is active (i.e. actively responds to clients' requests) while the second is passive (i.e. it does receive clients' requests, but does not respond to them and only maintains a copy of lease database and is ready to take over incoming queries in case of primary server failure). Active-active mode (i.e. both servers actively handling clients' requests) is currently not supported for the sake of simplicity. Such mode may be defined as an exension at a later time.

The failover protocol is designed to provide lease stability for leases with lease times beyond a short period. Due to the additional overhead required, failover is not suitable for leases shorter than 30 seconds. The DHCPv6 Failover protocol MUST NOT be used for leases shorter than 30 seconds.

This design attempts to fulfill all DHCPv6 failover requirements defined in [I-D.ietf-dhc-dhcpv6-failover-requirements].

3.1. Additional Requirements

The following requirements are not related to failover mechanism in general, but rather to this particular design.

1. Minimize Asymmetry - while there are two distinct roles in failover (primary and secondary server), the differences between

those two roles should be as small as possible. This will yield a simpler design as well as a simpler implementation of that design.

3.2. Features out of Scope: Load Balancing

It may be tempting to extend DHCPv6 failover mechanism to also offer load balancing, as DHCPv4 failover did. Here is the reasoning for this decision. In general case (not related to failover) load balancing solutions are used when each server is not able to handle total incoming traffic. However, by the very definition, DHCPv6 failover is supposed to assume service availability despite failure of one server. That leads to conclusion that each server must be able to handle whole traffic. Therefore in properly provisioned setup, load balancing is not needed.

4. Protocol Overview

The DHCPv6 Failover Protocol is defined as a communication between failover partners with all associated algorithms and mechanisms. Failover communication is conducted over a TCP connection established between the partners. The protocol reuses the framing format specified in Section 5.1 of DHCPv6 Bulk Leasequery [RFC5460], but uses different message types. Additional failover-specific message types will be defined. All information is sent over the connection as typical DHCPv6 Options, following format defined in Section 22.1 of [RFC3315].

After initialization, the primary server establishes a TCP connection with its partner. The primary server sends a CONNECT message with initial parameters. Secondary server responds with CONNECTACK.

Depending on the failover state of each partner, they MUST initiate one of the binding update procedures. Each server MAY send an UPDREQ message to request its partner to send all updates that have not been sent yet (this case applies when partner has an existing database and wants to update it). Alternatively, a server MAY choose to send an UPDREQALL message to request a full lease database transmission including all leases (this case applies in case of booting up new server after installation, corruption or complete loss of database, or other catastrophic failure).

Servers exchange lease information by using BNDUPD messages. Depending on local and remote state of a lease, a server may either accept or reject the update. Reception of lease update information is confirmed by responding with BNDACK message with appropriate status. The majority of the messages sent over a failover TCP

connection consists of BNDUPD and BNDACK messages.

A subset of available resources (addresses or prefixes) is reserved for secondary server use. This is required for handling a case where both servers are able to communicate with clients, but unable to communicate with each other. After initial connection is established, the secondary server requests a pool of available addresses by sending a POOLREQ message. The primary server assigns a pool to the secondary by transmitting a POOLRESP message and then sending a series of BNDUPD messages. The secondary server may initiate such pool request at any time when maintaining communication with primary server.

Failover servers use a lazy update mechanism to update their failover partner about changes to their lease state database. After a server performs any modifications to its lease state database (assign a new lease, extend an existing one, release or expire a lease), it sends its response to the client's request first (performing the "regular" DHCPv6 operation) and then informs its failover partner using a BNDUPD message. This BNDUPD message SHOULD be sent soon after the response is sent to the DHCPv6 client, but there is no specific requirement of a minimum time in which to do so.

The major problem with lazy update mechanism is the case when the server crashes after sending response to client, but before sending the lazy update to its partner (or when communication between partners is interrupted). To solve this problem, concept known as the Maximum Client Lead Time (MCLT) (initially designed for DHCPv4 failover) is used. The MCLT is the maximum amount of time that one server can extend a lease for a client's binding beyond the time known by its failover partner. See Section 8.4 for detailed desciption how MCLT affects assigned lease times.

Servers verify each others availability by periodically exchanging CONTACT messages. See Section 8.5 for discussion about detecting partner's unreachability.

A server that is being shut down transmits a DISCONNECT message, closes the connection with its failover partner and stops operation. A Server SHOULD transmit any pending lease updates before transmitting DISCONNECT message.

4.1. Failover Machine Sate Overview

The following section provides simplified description of all states. For the sake of clarity and simplicity, it omits important details. For complete description, see Section 9. In case of a disagreement between simplified and complete description, please follow Section 9.

Each server may be in one of the well defines states. In each state a server may be either responsive (responds to clients' queries) or unresponsive (clients' queries are ignored).

A server starts its operation in short-lived STARTUP state. A server determines its partner reachibility and state and usually returns back to the state it was in before shutdown.

During typical operation when servers maintain communication, both are in NORMAL state. In that state only primary responds to clients' requests. A secondary server in unresponsive.

If a server discovers that its partner is no longer reachable, it goes to COMMUNICATIONS-INTERRUPTED state. Server must be extra cautious as it can't distingush if its partner is down or just communication between servers is interrupted. Since communication between partners is not possible, a server must act on the assumtion that if its partner is up, it follows defined procedure. In particular, not extend any lease beyond its partner knowledge by at most MCLT. That imposes additional burden on the server. Therefore it is not recommended to operate for prolonged periods in this state. Once communication is reestablished, server may go into NORMAL, POTENTIAL-CONFLICT or PARTNER-DOWN state. It may also stay in COMMUNICATIONS-INTERRUPTED if certain conditions are met.

Once a server is switched into PARTNER-DOWN (when auto-partner-down is used or as a result of administrative action), it can extend leases, regardless of the original server that initially granted the lease. In that state server handles leases from its own pool, but is albo able to serve pool from its downed partner. MCLT restrictions no longer apply. Operation in this mode is less demanding for the server that remains operational, than in COMMUNICATIONS-INTERRUPTED state, but PARTNER-DOWN does not offer any kind of redundancy.

When server loses its database (e.g. due to first time run or catastrophic failure) or detects that is partner is in PARTNER-DOWN state and additional conditions are met, it switches to RECOVER state. In that state server acknowledges that content of its database is doubtful and needs to refresh its database from its partner. Once this operation is done, it switches to RECOVER-WAIT and later to RECOVER-DONE.

Once servers reestablish connection, they discover each others' state. Depending on the conditions, they may return to NORMAL or move to POTENTINAL-CONFLICT in case of unexpected partner's state. It is a goal of this protocol to minimize the possibility that POTENTIAL-CONFLICT state is ever entered. Servers running in POTENTIAL-CONFLICT do not respond to clients' requests and work on

resolving potential conflicts. Once outstanding lease updates are exchanged, servers move to CONFLICT-DONE or NORMAL states.

Servers that are recovering from potential conflict and loose communication, switch to RESOLUTION-INTERRUPTED.

Server that is being shut down, switches briefly to SHUTDOWN state and communicates its state to its partner before actual termination.

5. Connection Management

5.1. Creating Connections

Every server implementing the failover protocol SHOULD attempt to connect to all of its partners periodically, where the period is implementation dependent and SHOULD be configurable. In the event that a connection has been rejected by a CONNECTACK message with a reject-reason option contained in it or a DISCONNECT message, a server SHOULD reduce the frequency with which it attempts to connect to that server but it SHOULD continue to attempt to connect periodically.

When a connection attempt succeeds, if the server generating the connection attempt is a primary server for that relationship, then it MUST send a CONNECT message down the connection. If it is not a primary server for the relationship, then it MUST just drop the connection and wait for the primary server to connect to it.

When a connection attempt is received, the only information that the receiving server has is the IP address of the partner initiating a connection. It also knows whether it has the primary role for any failover relationships with the connecting server. If it has any relationships for which it is a primary server, it should initiate a connection of its own to the partner server, one for each primary relationship it has with that server.

If it has any relationships with the connecting server for which it is a seconary server, it should just await the CONNECT message to determine which relationship this connection is to serve.

If it has no secondary relationships with the connecting server, it SHOULD drop the connection.

To summarize -- a primary server MUST use a connection that it has initiated in order to send a CONNECT message. Every server that is a secondary server in a relationship attempts to create a connection to the server which is primary in the relationship, but that connection

is only used to stimulate the primary server into recognizing that the secondary server is ready for operation. The reason behind this is that the secondary server has no way to communicate to the primary server which relationship a connection is designed to serve.

A server which has multiple secondary relationships with a primary server SHOULD only send one stimulus connection attempt to the primary server.

Once a connection is established, the primary server MUST send a CONNECT message across the connection. A secondary server MUST wait for the CONNECT message from a primary server. If the secondary server doesn't receive a CONNECT message from the primary server in an installation dependent amount of time, it MAY drop the connection and send another stimulus connection attempt to the primary server.

Every CONNECT message includes a TLS-request option, and if the CONNECTACK message does not reject the CONNECT message and the TLS-reply option says TLS MUST be used, then the servers will immediately enter into TLS negotiation.

Once TLS negotiation is complete, the primary server MUST resend the CONNECT message on the newly secured TLS connection and then wait for the CONNECTACK message in response. The TLS-request and TLS-reply options MUST NOT appear in either this second CONNECT or its associated CONNECTACK message as they had in the first messages.

The second message sent over a new connection (either a bare TCP connection or a connection utilizing TLS) is a STATE message. Upon the receipt of this message, the receiver can consider communications up.

A secondary server MUST NOT respond to the closing of a TCP connection with a blind attempt to reconnect -- there may be another TCP connection to the same failover partner already in use.

5.2. Endpoint Identification

The proper operation of the failover protocol requires more than the transmission of messages between one server and the other. Each endpoint might seem to be a single DHCPv6 server, but in fact there are situations where additional flexibility in configuration is useful. A failover endpoint is always associated with a set of DHCPv6 prefixes that are configured on the DHCPv6 server where the endpoint appears. A DHCPv6 prefix MUST NOT be associated with more than one failover endpoint.

The failover protocol SHOULD be configured with one failover

relationship between each pair of failover servers. In this case there is one failover endpoint for that relationship on each failover partner. This failover relationship MUST have a unique name.

There is typically little need for addtional relationships between any two servers but there MAY be more than one failover relationship between two servers -- however each MUST have a unique relationship name.

Any failover endpoint can take actions and hold unique states.

This document frequently describes the behavior of the protocol in terms of primary and secondary servers, not primary and secondary failover endpoints. However, it is important to remember that every 'server' described in this document is in reality a failover endpoint that resides in a particular process, and that several failover end-points may reside in the same server process.

It is not the case that there is a unique failover endpoint for each prefix that participates in a failover relationship. On one server, there is (typically) one failover endpoint per partner, regardless of how many prefixes are managed by that combination of partner and role. Conversely, on a particular server, any given prefix will be associated with exactly one failover endpoint.

When a connection is received from the partner, the unique failover endpoint to which the message is directed is determined solely by the IP address of the partner, the relationship-name, and the role of the receiving server.

6. Resource Allocation

Currently there are two allocation algorithms defined for resources (addresses or prefixes). Additional allocation schemes may be defined as future extensions.

1. Proportional Allocation - This allocation algorithm is a direct application of algorithm defined in [dhcpv4-failover] to DHCPv6. Available resources are split between primary and secondary server. Released resources are always returned to primary server. Primary and secondary servers may initiate a rebalancing procedure, when disparity between resources available to each server reaches a preconfigured threshold. Only resources that are not leased to any clients are "owned" by one of the servers. This algorithm is particularly well suited for scenarios where amount of available resources is limited, as may be the case for prefix delegation. See Section 6.1 for details.

2. Independent Allocation - This allocation algorithm assumes that available resources are split between primary and secondary servers as well. In this case, however, resources are assigned to a specific server for all time, regardless if they are available or currently used. This algorithm is much simpler than proportional allocation, because resource imbalance doesn't have to be checked and there is no rebalancing for independent allocation. This algorithm is particularly well suited for scenarios where the there is an abundance of available resources which is typically the case for DHCPv6 address allocation. See Section 6.2 for details.

6.1. Proportional Allocation

In this allocation scheme, each server has its own pool of available resources. Note that a resource is not "owned" by a particular server throughout its entire lifetime. Only a resource which is available is "owned" by a particular server -- once it has been leased to a client, it is not owned by either failover partner. When it finally becomes available again, it will be owned initially by the primary server, and it may or may not be allocated to the secondary server by the primary server.

So, the flow of a resource is as follows: initially a resource is owned by the primary server. It may be allocated to the secondary server if it is available, and then it is owned by the secondary server. Either server can allocate available resources which they own to clients, in which case they cease to own them. When the client releases the resource or the lease on it expires, it will again become available and will be owned by the primary.

A resource will not become owned by the server which allocated it initially when it is released or the lease expires because, in general, that server will have had to replenish its pool of available resources well in advance of any likely lease expirations. Thus, having a particular resource cycle back to the secondary might well put the secondary more out of balance with respect to the primary instead of enhancing the balance of available addresses or prefixes between them.

TODO: Need to rework this v4-specific vocabulary to v6, once we decide how things will look like in v6.

When they are used, these proportional pools are used for allocation when in every state but PARTNER-DOWN state. In PARTNER-DOWN state a failover server can allocate from either pool. This allocation and maintenance of these address pools is an area of some sensitivity, since the goal is to maintain a more or less constant ratio of

available addresses between the two servers.

TODO: Reuse rest of the description from section 5.4 from [dhcpv4-failover] here.

6.2. Independent Allocation

In this allocation scheme, available resources are split between servers. Available resources are split between the primary and secondary servers as part of initial connection establishment. Once resources are allocated to each server, there is no need to reassign them. This algorithm is simpler than proportional allocation since it requires no less initial communicagtion and does not require a rebalancing mechanism, but it assumes that the pool assigned to each server will never deplete. That is often a reasonable assumption for IPv6 addresses (e.g. servers are often assigned a /64 pool that contains many more addresses than existing electronic devices on Earth). This allocation mechanism SHOULD be used for IPv6 addresses, unless configured address pool is small or is otherwise administratively limited.

Once each server is assigned a resource pool during initial connection establishment, it may allocate assigned resources to clients. Once a client release a resource or its lease is expired, the returned resource returns to pool for the same server. Resources never changes servers.

During COMMUNICATION-INTERRUPTED events, a partner MAY continue extending existing leases when requested by clients. A healthy partner MUST NOT lease resources that were assigned to its downed partner and later released by a client unless it is in PARTNER-DOWN state.

6.3. Determining Allocation Approach

6.3.1. IPv6 Addresses

6.3.2. IPv6 Prefixes

7. Information model

TODO: Describe information model here. In particular, we need to describe lease lifecycle here.

TODO: In case of Active-Passive model, while majority of addresses are owned by the primary server, secondary server will need a portion of addresses to serve new clients while operating in communication-

interrupted state as also in partner down state before it can take over the entire address pool (expiry of MCLT). The concept of a percentage of pool reserved for secondary should be described here.

8. Failover Mechanisms

This section lays out an overview of the communication between partners and other mechanisms required for failover operation. As this is a design document, not a protocol specification, high level ideas are presented without implementation specific details (e.g. lack of on-wire formats). Implementation details will be specified in a separate draft.

8.1. Time Skew

Partners exchange information about known lease states. To reliably compare a known lease state with an update received from a partner, servers must be able to reliably compare the times stored in the known lease state with the times received in the update. Although a simple approach would be to require both partners to use synchronized time, e.g. by using NTP, such a service may become unavailable in some scenarios that failover expects to cover, e.g. network partition. Therefore a mechanism to measure and track relative time differences between servers is necessary. To do so, each message MUST contain FO_TIMESTAMP option that contains the timestamp of the transmission in the time context of the transmitter. The transmitting server MUST set this as close to the actual transmission as possible. The receiving partner MUST store its own timestamp of reception event as close to the actual reception as possible. The received timestamp information is then compared with local timestamp.

To account for packet delay variation (jitter), the measured difference is not used directly, but rather the moving average of last TIME_SKEW_PKTS_AVG packets time difference is calculated. This averaged value is referred to as the time skew. Note that the time skew algorithm allows cooperation between clients with completely desynchronized clocks as well as those whose desynchronization itself is not constant.

8.2. Time expression

Timestamps are expressed as number of seconds since midnight (UTC), January 1, 2000, modulo 2^32. Note: that is the same approach as used in creation of DUID-LLT (see Section 9.2 of [RFC3315]).

Time differences are expressed in seconds and are signed.

8.3. Lazy updates

Lazy update refers to the requirement placed on a server implementing a failover protocol to update its failover partner whenever the binding database changes. A failover protocol which didn't support lazy update would require the failover partner update to complete before a DHCPv6 server could respond to a DHCPv6 client request. The lazy update mechanism allows a server to allocate a new or extend an existing lease and then update its failover partner as time permits.

Although the lazy update mechanism does not introduce additional delays in server response times, it introduces other difficulties. The key problem with lazy update is that when a server fails after updating a client with a particular lease time and before updating its partner, the partner will believe that a lease has expired even though the client still retains a valid lease on that address or prefix.

8.4. MCLT concept

In order to handle problem introduced by lazy updates (see Section 8.3), a period of time known as the "Maximum Client Lead Time" (MCLT) is defined and must be known to both the primary and secondary servers. Proper use of this time interval places an upper bound on the difference allowed between the lease time provided to a DHCPv6 client by a server and the lease time known by that server's failover partner.

The MCLT is typically much less than the lease time that a server has been configured to offer a client, and so some strategy must exist to allow a server to offer the configured lease time to a client. During a lazy update the updating server typically updates its partner with a potential expiration time which is longer than the lease time previously given to the client and which is longer than the lease time that the server has been configured to give a client. This allows that server to give a longer lease time to the client the next time the client renews its lease, since the time that it will give to the client will not exceed the MCLT beyond the potential expiration time acknowledged by its partner.

The fundamental relationship on which much of The correctness of this protocol depends is that the lease expiration time known to a DHCPv6 client MUST NOT under any circumstances be more than the maximum client lead time (MCLT) greater than the potential expiration time known to a server's partner.

The remainder of this section makes the above fundamental relationship more explicit.

This protocol requires a DHCPv6 server to deal with several different lease intervals and places specific restrictions on their relationships. The purpose of these restrictions is to allow the other server in the pair to be able to make certain assumptions in the absence of an ability to communicate between servers.

The different times are:

desired valid lifetime:
The desired valid lifetime is the lease interval that a DHCPv6 server would like to give to a DHCPv6 client in the absence of any restrictions imposed by the failover protocol. Its determination is outside of the scope of this protocol. Typically this is the result of external configuration of a DHCPv6 server.

actual valid lifetime:
The actual valid lifetime is the lease interval that a DHCPv6 server gives out to a DHCPv6 client. It may be shorter than the desired valid lifetime (as explained below).

potential valid lifetime:
The potential valid lifetime is the potential lease expiration interval the local server tells to its partner in a BNDUPD message.

acknowledged potential valid lifetime:
The acknowledged potential valid lifetime is the potential lease interval the partner server has most recently acknowledged in a BNDACK message.

8.4.1. MCLT example

The following example demonstrates the MCLT concept in practice. The values used are arbitrarily chosen are and not a recommendation for actual values. The MCLT in this case is 1 hour. The desired valid lifetime is 3 days, and its renewal time is half the valid lifetime.

When a server makes an offer for a new lease on an IP address to a DHCPv6 client, it determines the desired valid lifetime (in this case, 3 days). It then examines the acknowledged potential valid lifetime (which in this case is zero) and determines the remainder of the time left to run, which is also zero. To this it adds the MCLT. Since the actual valid lifetime cannot be allowed to exceed the remainder of the current acknowledged potential valid lifetime plus the MCLT, the offer made to the client is for the remainder of the current acknowledged potential valid lifetime (i.e., zero) plus the MCLT. Thus, the actual valid lifetime is 1 hour.

Once the server has sent the REPLY to the DHCPv6 client, it will update its failover partner with the lease information. However, the desired potential valid lifetime will be composed of one half of the current actual valid lifetime added to the desired valid lifetime. Thus, the failover partner is updated with a BNDUPD with a potential valid lifetime of 3 days + 1/2 hour.

When the primary server receives a BNDACK to its update of the secondary server's (partner's) potential valid lifetime, it records that as the acknowledged potential valid lifetime. A server MUST NOT send a BNDACK in response to a BNDUPD message until it is sure that the information in the BNDUPD message has been updated in its lease database. Thus, the primary server in this case can be sure that the secondary server has recorded the potential lease interval in its stable storage when the primary server receives a BNDACK message from the secondary server.

When the DHCPv6 client attempts to renew at T1 (approximately one half an hour from the start of the lease), the primary server again determines the desired valid lifetime, which is still 3 days. It then compares this with the remaining acknowledged potential valid lifetime (3 days + 1/2 hour) and adjusts for the time passed since the secondary was last updated (1/2 hour). Thus the time remaining of the acknowledged potential valid interval is 3 days. Adding the MCLT to this yields 3 days plus 1 hour, which is more than the desired valid lifetime of 3 days. So the client is renewed for the desired valid lifetime -- 3 days.

When the primary DHCPv6 server updates the secondary DHCPv6 server after the DHCPv6 client's renewal REPLY is complete, it will calculate the desired potential valid lifetime as the T1 fraction of the actual client valid lifetime (1/2 of 3 days this time = 1.5 days). To this it will add the desired client valid lifetime of 3 days, yielding a total desired potential valid lifetime of 4.5 days. In this way, the primary attempts to have the secondary always "lead" the client in its understanding of the client's valid lifetime so as to be able to always offer the client the desired client valid lifetime.

Once the initial actual client valid lifetime of the MCLT is past, the protocol operates effectively like the DHCPv6 protocol does today in its behavior concerning valid lifetimes. However, the guarantee that the actual client valid lifetime will never exceed the remaining acknowledged partner server potential valid lifetime by more than the MCLT allows full recovery from a variety of failures.

8.5. Unreachability detection

Each partner maintains an FO_SEND timer for each partner connection. The FO_SEND timer is reset every time any message is transmitted. If the timer reaches the FO_SEND_MAX value, a CONTACT message is transmitted and timer is reset. The CONTACT message may be transmitted at any time.

Discussion: Perhaps it would be more reasonable to use echo-reply approach, rather than periodic transmissions?

8.6. Re-allocating Leases

TODO: Describe controlled re-allocation of released/expired leases to different clients.

8.7. Sending Data

Each server updates its failover partner about recent changes in lease states. Each update must include following information:

1. resource type - non-temporary address or a prefix

2. resource information - actual address or prefix

3. valid life time requested by client

4. IAID - Identity Association used by client, while obtaining this lease. (Note1: one client may use many IAID simulatenously. Note2: IAID for IA, TA and PD are orthogonal number spaces.)

5. valid life time sent to client

6. potential valid life time

7. preferred life time sent to client

8. CLTT - Client Last Transaction Time, a timestamp of the last received transmission from a client

9. assigned FQDN names, if any (optional)

Discussion: Do we need T1 as well? Something like next expected client transmission?

Q: Maybe we could reuse IA_NA and IA_PD options here? Yes.

Q: Do we care about preferred lifetime? (presumably no). Certainly

not what was requested by the client.

Q: Do we care about IAID? (presumably yes) Yes.

8.7.1. Required Data

8.7.2. Optional Data

8.8. Receiving Data

8.8.1. Conflict Resolution

TODO: This is just a loose collection of notes. This section will probably need to be rewritten as a a flowchart of some kind.

The server receiving a lease update from its partner must evaluate the received lease information to see if it is consistent with already known state and decide which information - previously known or just received - is "better". The server should take into consideration the following aspects: if the lease is already assigned to specific client, who had contact with client recently, start time of the lease, etc.

The lease update may be accepted or rejected. Rejection SHOULD NOT change the flag in a lease that says that it should be transmitted to the failover partner. If this flag is set, then it should be transmitted, but if it is not already set, the rejection of a lease state update SHOULD NOT trigger an automatic update of the failover partner sending the rejected update. The potential for update storms is too great, and in the unusual case where the servers simply can't agree, that disagreement is better than an update storm.

Discussion: There will definitely be different types of update rejections. For example, this will allow a server to treat differently a case when receiving a new lease that it previously haven't seen than a case when partner sents old version of a lease for which a newer state is known.

8.8.2. Acknowledging Reception

9. Endpoint States

9.1. State Machine Operation

Each server (or, more accurately, failover endpoint) can take on a variety of failover states. These states play a crucial role in determining the actions that a server will perform when processing a

request from a DHCPv6 client as well as dealing with changing external conditions (e.g., loss of connection to a failover partner).

The failover state in which a server is running controls the following behaviors:

- o Responsiveness -- the server is either responsive to DHCPv6 client requests or it is not.
- o Allocation Pool -- which pool of addresses (or prefixes) can be used for allocation on receipt of a SOLICIT message.
- o MCLT -- ensure that valid lifetimes are not beyond what the partner has acked plus the MCLT (or not).

A server will transition from one failover state to another based on the specific values held by the following state variables:

- o Current failover state.
- o Communications status (OK or not OK).
- o Partner's failover state (if known).

Whenever the either of the last two of the above state variables changes state, the state machine is invoked, which may then trigger a change in the current failove state. Thus, whenever the communications status changes, the state machine is processing is invoked. This may or may not result in a change in the current failover state.

Whenever a server transitions to a new failover state, the new state MUST be communicated to its failover partner in a STATE message if the communications status is OK. In addition, whenever a server makes a transition into a new state, it MUST record the new state, its current understanding of its partner's state, and the time at which it entered the new state in stable storage.

The following state transition diagram gives a condensed view of the state machine. If there is a difference between the words describing a particular state and the diagram below, the words should be considered authoritative.

A transition into SHUTDOWN or PAUSED state is not represented in the following figure, since other than sending that state to its partner, the remaining actions involved look just like the server halting in its otherwise current state, which then becomes the previous state upon server restart.

```
      +---------------+  V  +--------------+
      |    RECOVER -|+|  |  |   STARTUP  - |
      |(unresponsive) |  +->+(unresponsive)|
      +------+--------+     +--------------+
      +-Comm. OK             +-----------------+
      |     Other State:     |  PARTNER DOWN - +<----------------------+
      |    RESOLUTION-INTER. | (responsive)    |                       ^
     All     POTENTIAL-      +----+------------+                       |
    Others   CONFLICT------------ | --------+                          |
      |      CONFLICT-DONE     Comm. OK     |     +--------------+     |
   UPDREQ or                 Other State:   |  +--+ RESOLUTION - |     |
   UPDREQALL                  |       |     |  |  | INTERRUPTED  |     |
   Rcv UPDDONE             RECOVER    All   |  |  | (responsive) |     |
      |  +---------------+    |      Others |  |  +------------+-+     |
      +->+RECOVER-WAIT +-| RECOVER    |     |  |         ^     |       |
         |(unresponsive) |  WAIT or   |     |  Comm.     |    Ext.     |
         +-----------+---+  DONE      |     |  OK     Comm.   Cmd----->+
  Comm.---+     Wait MCLT     |       V     V  V     Failed            |
  Changed |          V    +---+   +---+-----+--+-+       |             |
   |  +---+----------++   |       |  POTENTIAL + +-------+             |
   |  |RECOVER-DONE +-|  Wait     |  CONFLICT    +------+              |
   +->+(unresponsive) |  for      |(unresponsive)|   Primary           |
      +------+--------+  Other  +>+----+--------++   resolve     Comm. |
       Comm. OK          State: |      |        ^    conflict  Changed |
  +---Other State:-+   RECOVER  |   Secondary   |       V       V   |  |
  |    |           |     DONE   |    resolve    |   ++----------+---++ |
  | All Others:  POTENT.  |     |   conflict    |   |CONFLICT-DONE-|+| |
  | Wait for    CONFLICT- | ----+    see (9.10) |   | (responsive)   | |
  | Other State:          V            V        |   +------+---------+ |
  | NORMAL or RECOVER    ++------------+---+      Other State: NORMAL  |
  |    |       DONE      |     NORMAL    + +<--------------+           |
  |    +--+----------+-->+   (balanced)    +-------External Command--->+
  |       ^          ^   +--------+--------+       or Other State:     |
  |       |          |            |             |  SHUTDOWN            |
  |   Wait for   Comm. OK  Comm. Failed or      |                      |
  |    Other      Other    Other State: PAUSED  |               External
  |    State:     State:          |             |                Command
  | RECOVER-DONE  NORMAL     Start Safe      Comm. OK                or
  |       |     COMM. INT.  Period Timer    Other State:            Safe
  |    Comm. OK.     |            V          All Others           Period
  |   Other State:   |  +---------+--------+    |             expiration
  |     RECOVER      +--+ COMMUNICATIONS - +----+                      |
  |       +-------------+   INTERRUPTED    |                           |
  RECOVER               |  (responsive)    +-------------------------->+
  RECOVER-WAIT--------->+------------------+
```

Figure 1: Failover Endpoint State Machine

9.2. State Machine Initialization

TODO

9.3. STARTUP State

The STARTUP state affords an opportunity for a server to probe its partner server, before starting to service DHCP clients. When in the STARTUP state, a server attempts to learn its partner's state and determine (using that information if it is available) what state it should enter.

The STARTUP state is not shown with any specific state transitions in the state machine diagram (Figure 1) because the processing during the STARTUP state can cause the server to transition to any of the other states, so that specific state transition arcs would only obscure other information.

9.3.1. Operation in STARTUP State

The server MUST NOT be responsive in STARTUP state.

Whenever a STATE message is sent to the partner while in STARTUP state the STARTUP flag MUST be set the message and the previously recorded failover state MUST be placed in the server-state option.

9.3.2. Transition Out of STARTUP State

The following algorithm is followed every time the server initializes itself, and enters STARTUP state.

Step 1:

If there is any record in stable storage of a previous failover state for this server, set PREVIOUS-STATE to the last recorded value in stable storage, and go to Step 2.

If there is no record of any previous failover state in stable storage for this server, then set the PREVIOUS-STATE to RECOVER and set the TIME-OF-FAILURE to 0. This will allow two servers which already have lease information to synchronize themselves prior to operating.

In some cases, an existing server will be commissioned as a failover server and brought back into operation where its partner is not yet available. In this case, the newly commissioned failover server will not operate until its partner comes online -- but it has operational responsibilities as a DHCP server nonetheless. To properly handle

this situation, a server SHOULD be configurable in such a way as to move directly into PARTNER-DOWN state after the startup period expires if it has been unable to contact its partner during the startup period.

Step 2:

If the previous state is one where communications was "OK", then set the previous state to the state that is the result of the communications failed state transition (if such transition exists -- some states don't have a communications failed state transition, since they allow both commun- ications OK and failed).

Step 3:

Start the STARTUP state timer. The time that a server remains in the STARTUP state (absent any communications with its partner) is implementation dependent but SHOULD be short. It SHOULD be long enough for a TCP connection to be created to a heavily loaded partner across a slow network.

Step 4:

Attempt to create a TCP connection to the failover partner.

Step 5:

Wait for "communications OK".

When and if communications become "okay", clear the STARTUP flag, and set the current state to the PREVIOUS-STATE.

If the partner is in PARTNER-DOWN state, and if the time at which it entered PARTNER-DOWN state (as received in the start-time-of-state option in the STATE message) is later than the last recorded time of operation of this server, then set CURRENT-STATE to RECOVER. If the time at which it entered PARTNER-DOWN state is earlier than the last recorded time of operation of this server, then set CURRENT-STATE to POTENTIAL-CONFLICT.

Then, transition to the current state and take the "communications OK" state transition based on the current state of this server and the partner.

Step 6:

If the startup time expires the server SHOULD go transition to the PREVIOUS-STATE.

9.4. PARTNER-DOWN State

PARTNER-DOWN state is a state either server can enter. When in this state, the server assumes that it is the only server operating and serving the client base. If one server is in PARTNER-DOWN state, the other server MUST NOT be operating.

9.4.1. Operation in PARTNER-DOWN State

The server MUST be responsive in PARTNER-DOWN state.

It will allow renewal of all outstanding leases on IP addresses. For those IP addresses for which the server is using proportional allocation, it will allocate IP addresses from its own pool, and after a fixed period of time (the MCLT interval) has elapsed from entry into PARTNER-DOWN state, it will allocate IP addresses from the set of all available IP addresses.

Any IP address tagged as available for allocation by the other server (at entry to PARTNER-DOWN state) MUST NOT be allocated to a new client until the maximum-client-lead-time beyond the entry into PARTNER-DOWN state has elapsed.

A server in PARTNER-DOWN state MUST NOT allocate an IP address to a DHCP client different from that to which it was allocated at the entrance to PARTNER-DOWN state until the maximum-client-lead-time beyond the maximum of the following times: client expiration time, most recently transmitted potential-expiration-time, most recently received ack of potential-expiration-time from the partner, and most recently acked potential-expiration-time to the partner. If this time would be earlier than the current time plus the maximum-client-lead-time, then the time the server entered PARTNER-DOWN state plus the maximum-client-lead-time is used.

The server is not restricted by the MCLT when offering lease tmes while in PARTNER-DOWN state.

In the unlikely case, when there are two servers operating in a PARTNER-DOWN state, there is a change od duplicate leases assigned. This leads to a POTENTIAL-CONFLICT (unresponsive) state when they re-establish contact. The duplicate lease issue can be postponed to a large extent by the server giving new leases from its own pool. Therefore the server operating in PARTNER-DOWN state MUST use its own pool first for new leases before assigning any leases from its downed partner pool.

9.4.2. Transition Out of PARTNER-DOWN State

When a server in PARTNER-DOWN state succeeds in establishing a connection to its partner, its actions are conditional on the state and flags received in the STATE message from the other server as part of the process of establishing the connection.

If the STARTUP bit is set in the server-flags option of a received STATE message, a server in PARTNER-DOWN state MUST NOT take any state transitions based on reestablishing communications. Essentially, if a server is in PARTNER-DOWN state, it ignores all STATE messages from its partner that have the STARTUP bit set in the server-flags option of the STATE message. THIS NEEDS TO BE MOVED

If the STARTUP bit is not set in the server-flags option of a STATE message received from its partner, then a server in PARTNER-DOWN state takes the following actions based on the state of the partner as received in a STATE message (either immediately after establishing communications or at any time later when a new state is received)

If the partner is in:

NORMAL, COMMUNICATIONS-INTERRUPTED, PARTNER-DOWN, POTENTIAL-CONFLICT, RESOLUTION-INTERRUPTED, or CONFLICT-DONE state

transition to POTENTIAL-CONFLICT state

If the partner is in:

RECOVER, RECOVER-WAIT, SHUTDOWN, PAUSED state

stay in PARTNER-DOWN state

If the partner is in:

RECOVER-DONE state

transition into NORMAL state

9.5. RECOVER State

This state indicates that the server has no information in its stable storage or that it is re-integrating with a server in PARTNER-DOWN state after it has been down. A server in this state MUST attempt to refresh its stable storage from the other server.

9.5.1. Operation in RECOVER State

The server MUST NOT be responsive in RECOVER state.

A server in RECOVER state will attempt to reestablish communications with the other server.

9.5.2. Transition Out of RECOVER State

If the other server is in POTENTIAL-CONFLICT, RESOLUTION-INTERRUPTED, or CONFLICT-DONE state when communications are reestablished, then the server in RECOVER state will move to POTENTIAL-CONFLICT state itself.

If the other server is in any other state, then the server in RECOVER state will request an update of missing binding information by sending an UPDREQ message. If the server has determined that it has lost its stable storage because it has no record of ever having talked to its partner, while its partner does have a record of communicating with it, it MUST send an UPDREQALL message, otherwise it MUST send an UPDREQ message.

It will wait for an UPDDONE message, and upon receipt of that message it will transition to RECOVER-WAIT state.

If communications fails during the reception of the results of the UPDREQ or UPDREQALL message, the server will remain in RECOVER state, and will re-issue the UPDREQ or UPDREQALL when communications are re-established.

If an UPDDONE message isn't received within an implementation dependent amount of time, and no BNDUPD messages are being received, the connection SHOULD be dropped.

```
                   A                                        B
                 Server                                  Server

                   |                                        |
                RECOVER                               PARTNER-DOWN
                   |                                        |
                   | >--UPDREQ-------------------->         |
                   |                                        |
                   |        <----------------------BNDUPD--< |
                   | >--BNDACK-------------------->         |
                  ...                                      ...
                   |                                        |
                   |        <----------------------BNDUPD--< |
                   | >--BNDACK-------------------->         |
                   |                                        |
                   |        <---------------------UPDDONE--< |
                   |                                        |
              RECOVER-WAIT                                  |
                   |                                        |
                   | >--STATE-(RECOVER-WAIT)------>         |
                   |                                        |
                   |                                        |
          Wait MCLT from last known                         |
             time of failover operation                     |
                   |                                        |
              RECOVER-DONE                                  |
                   |                                        |
                   | >--STATE-(RECOVER-DONE)------>         |
                   |                                     NORMAL
                   |        <--------------(NORMAL)-STATE--< |
                NORMAL                                      |
                   | >---- State-(NORMAL)---------------->  |
                   |                                        |
                   |                                        |
```

Figure 2: Transition out of RECOVER state

If, at any time while a server is in RECOVER state communications fails, the server will stay in RECOVER state. When communications are restored, it will restart the process of transitioning out of RECOVER state.

9.6. RECOVER-WAIT State

This state indicates that the server has done an UPDREQ or UPDREQALL and has received the UPDDONE message indicating that it has received all outstanding binding update information. In the RECOVER-WAIT state the server will wait for the MCLT in order to ensure that any

processing that this server might have done prior to losing its stable storage will not cause future difficulties.

9.6.1. Operation in RECOVER-WAIT State

The server MUST NOT be responsive in RECOVER-WAIT state.

9.6.2. Transition Out of RECOVER-WAIT State

Upon entry to RECOVER-WAIT state the server MUST start a timer whose expiration is set to a time equal to the time the server went down (if known) or the time the server started (if the down-time is unknown) plus the maximum-client-lead-time. When this timer expires, the server will transition into RECOVER-DONE state.

This is to allow any IP addresses that were allocated by this server prior to loss of its client binding information in stable storage to contact the other server or to time out.

If this is the first time this server has run failover -- as determined by the information received from the partner, not necessarily only as determined by this server's stable storage (as that may have been lost), then the waiting time discussed above may be skipped, and the server may transition immediately to RECOVER-DONE state.

If the server has never before run failover, then there is no need to wait in this state -- but, again, to determine if this server has run failover it is vital that the information provided by the partner be utilized, since the stable storage of this server may have been lost.

If communications fails while a server is in RECOVER-WAIT state, it has no effect on the operation of this state. The server SHOULD continue to operate its timer, and the timer expires during the period where communications with the other server have failed, then the server SHOULD transition to RECOVER-DONE state. This is rare -- failover state transitions are not usually made while communications are interrupted, but in this case there is no reason to inhibit the timer.

9.7. RECOVER-DONE State

This state exists to allow an interlocked transition for one server from RECOVER state and another server from PARTNER-DOWN or COMMUNICATIONS-INTERRUPTED state into NORMAL state.

9.7.1. Operation in RECOVER-DONE State

A server in RECOVER-DONE state MUST respond only to DHCPREQUEST/RENEWAL and DHCPREQUEST/REBINDING DHCP messages.

9.7.2. Transition Out of RECOVER-DONE State

When a server in RECOVER-DONE state determines that its partner server has entered NORMAL or RECOVER-DONE state, then it will transition into NORMAL state.

If communications fails while in RECOVER-DONE state, a server will stay in RECOVER-DONE state.

9.8. NORMAL State

NORMAL state is the state used by a server when it is communicating with the other server, and any required resynchronization has been performed. While some bindings database synchronization is performed in NORMAL state, potential conflicts are resolved prior to entry into NORMAL state as is binding database data loss.

When entering NORMAL state, a server will send to the other server all currently unacknowledged binding updates as BNDUPD messages.

When the above process is complete, if the server entering NORMAL state is a secondary server, then it will request IP addresses for allocation using the POOLREQ message.

9.8.1. Operation in NORMAL State

When in NORMAL state a server will operate in the following manner:

Lease time calculations

As discussed in Section 8.4, the lease interval given to a DHCP client can never be more than the MCLT greater than the most recently received potential- expiration-time from the failover partner or the current time, whichever is later.

As long as a server adheres to this constraint, the specifics of the lease interval that it gives to a DHCP client or the value of the potential-expiration-time sent to its failover partner are implementation dependent.

Lazy update of partner server

After sending an REPLY that includes lease update to a client, the server servicing a DHCP client request attempts to update its partner with the new binding information. Server transmits both

desired valid lifetime and actual valid lifetime.

Reallocation of IP addresses between clients
Whenever a client binding is released or expires, a BNDUPD message must be sent to the partner, setting the binding state to RELEASED or EXPIRED. However, until a BNDACK is received for this message, the IP address cannot be allocated to another client. It cannot be allocated to the same client again if a BNDUPD was sent, otherwise it can. See Section 8.6.

In normal state, each server receives binding updates from its partner server in BNDUPD messages. It records these in its client binding database in stable storage and then sends a corresponding BNDACK message to its partner server.

9.8.2. Transition Out of NORMAL State

If an external command is received by a server in NORMAL state informing it that its partner is down, then transition into PARTNER-DOWN state. Generally, this would be an unusual situation, where some external agency knew the partner server was down. Using the command in this case would be appropriate if the polling interval and timeout were long.

If a server in NORMAL state fails to receive acks to messages sent to its partner for an implementation dependent period of time, it MAY move into COMMUNICATIONS-INTERRUPTED state. This situation might occur if the partner server was capable of maintaining the TCP connection between the server and also capable of sending a CONTACT message every tSend seconds, but was (for some reason) incapable of processing BNDUPD messages.

If the communications is determined to not be "ok" (as defined in Section 8.5), then transition into COMMUNICATIONS-INTERRUPTED state.

If a server in NORMAL state receives any messages from its partner where the partner has changed state from that expected by the server in NORMAL state, then the server should transition into COMMUNICATIONS-INTERRUPTED state and take the appropriate state transition from there. For example, it would be expected for the partner to transition from POTENTIAL-CONFLICT into NORMAL state, but not for the partner to transition from NORMAL into POTENTIAL-CONFLICT state.

If a server in NORMAL state receives any messages from its partner where the PARTNER has changed into SHUTDOWN state, the server should transition into PARTNER-DOWN state.

9.9. COMMUNICATIONS-INTERRUPTED State

A server goes into COMMUNICATIONS-INTERRUPTED state whenever it is unable to communicate with its partner. Primary and secondary servers cycle automatically (without administrative intervention) between NORMAL and COMMUNICATIONS-INTERRUPTED state as the network connection between them fails and recovers, or as the partner server cycles between operational and non-operational. No duplicate IP address allocation can occur while the servers cycle between these states.

When a server enters COMMUNICATIONS-INTERRUPTED state, if it has been configured to support an automatic transition out of COMMUNICATIONS-INTERRUPTED state and into PARTNER-DOWN state (i.e., a "safe period" has been configured, see section 10), then a timer MUST be started for the length of the configured safe period.

A server transitioning into the COMMUNICATIONS-INTERRUPTED state from the NORMAL state SHOULD raise some alarm condition to alert administrative staff to a potential problem in the DHCP subsystem.

9.9.1. Operation in COMMUNICATIONS-INTERRUPTED State

In this state a server MUST respond to all DHCP client requests. When allocating new lease, each server allocates from its own pool, where the primary MUST allocate only FREE resources (addresses or prefixes), and the secondary MUST allocate only BACKUP resources (addresses or prefixes). When responding to RENEW messages, each server will allow continued renewal of a DHCP client's current lease on an IP address or prefix irrespective of whether that lease was given out by the receiving server or not, although the renewal period MUST NOT exceed the maximum client lead time (MCLT) beyond the latest of: 1) the potential valid lifetime already acknowledged by the other server, or 2) the lease- expiration-time , or 3) the potential valid lifetime received from the partner server.

However, since the server cannot communicate with its partner in this state, the acknowledged potential valid lifetime will not be updated in any new bindings. This is likely to eventually cause the actual valid lifetimes to be the current time plus the MCLT (unless this is greater than the desired-client-lease- time).

The server should continue to try to establish a connection with its partner.

9.9.2. Transition Out of COMMUNICATIONS-INTERRUPTED State

If the safe period timer expires while a server is in the COMMUNICATIONS-INTERRUPTED state, it will transition immediately into PARTNER-DOWN state.

If an external command is received by a server in COMMUNICATIONS-INTERRUPTED state informing it that its partner is down, it will transition immediately into PARTNER-DOWN state.

If communications is restored with the other server, then the server in COMMUNICATIONS-INTERRUPTED state will transition into another state based on the state of the partner:

- NORMAL or COMMUNICATIONS-INTERRUPTED: Transition into the NORMAL state.

- RECOVER: Stay in COMMUNICATIONS-INTERRUPTED state.

- RECOVER-DONE: Transition into NORMAL state.

- PARTNER-DOWN, POTENTIAL-CONFLICT, CONFLICT-DONE, or RESOLUTION-INTERRUPTED: Transition into POTENTIAL-CONFLICT state.

- SHUTDOWN: Transition into PARTNER-DOWN state.

The following figure illustrates the transition from NORMAL to COMMUNICATIONS-INTERRUPTED state and then back to NORMAL state again.

```
      Primary                                       Secondary
       Server                                         Server

       NORMAL                                         NORMAL
         | >--CONTACT------------------->              |
         |        <--------------------CONTACT--<      |
         |          [TCP connection broken]            |
    COMMUNICATIONS          :               COMMUNICATIONS
      INTERRUPTED           :                 INTERRUPTED
         |       [attempt new TCP connection]          |
         |          [connection succeeds]              |
         |                                             |
         | >--CONNECT------------------->              |
         |        <-----------------CONNECTACK--<      |
         |                                        NORMAL
         |        <-------------------STATE-----<      |
       NORMAL                                          |
         | >--STATE--------------------->              |
         |
         | >--BNDUPD-------------------->              |
         |        <---------------------BNDACK--<      |
         |                                             |
         |        <---------------------BNDUPD--<      |
         | >------BNDACK---------------->              |
        ...                                           ...
         |                                             |
         |        <--------------------POOLREQ--<      |
         | >--POOLRESP-(2)-------------->              |
   t>      |                                             |
         | >--BNDUPD-(#1)--------------->              |
         |        <---------------------BNDACK--<      |
         |                                             |
         |        <--------------------POOLREQ--<      |
         | >--POOLRESP-(0)-------------->              |
         |                                             |
         | >--BNDUPD-(#2)--------------->              |
         |        <---------------------BNDACK--<      |
         |                                             |
```

Figure 3: Transition from NORMAL to COMMUNICATIONS-INTERRUPTED and back (example with 2 addresses allocated to secondary)

9.10. POTENTIAL-CONFLICT State

This state indicates that the two servers are attempting to reintegrate with each other, but at least one of them was running in a state that did not guarantee automatic reintegration would be possible. In POTENTIAL-CONFLICT state the servers may determine that

the same resource has been offered and accepted by two different clients.

It is a goal of this protocol to minimize the possibility that POTENTIAL-CONFLICT state is ever entered.

When a primary server enters POTENTIAL-CONFLICT state it should request that the secondary send it all updates of which it is currently unaware by sending an UPDREQ message to the secondary server.

A secondary server entering POTENTIAL-CONFLICT state will wait for the primary to send it an UPDREQ message.

9.10.1. Operation in POTENTIAL-CONFLICT State

Any server in POTENTIAL-CONFLICT state MUST NOT process any incoming DHCP requests.

9.10.2. Transition Out of POTENTIAL-CONFLICT State

If communications fails with the partner while in POTENTIAL-CONFLICT state, then the server will transition to RESOLUTION-INTERRUPTED state.

Whenever either server receives an UPDDONE message from its partner while in POTENTIAL-CONFLICT state, it MUST transition to a new state. The primary MUST transition to CONFLICT-DONE state, and the secondary MUST transition to NORMAL state. This will cause the primary server to leave POTENTIAL-CONFLICT state prior to the secondary, since the primary sends an UPDREQ message and receives an UPDDONE before the secondary sends an UPDREQ message and receives its UPDDONE message.

When a secondary server receives an indication that the primary server has made a transition from POTENTIAL-CONFLICT to CONFLICT-DONE state, it SHOULD send an UPDREQ message to the primary server.

```
       Primary                                     Secondary
       Server                                       Server

          |                                           |
   POTENTIAL-CONFLICT                        POTENTIAL-CONFLICT
          |                                           |
          | >--UPDREQ-------------------->            |
          |                                           |
          |        <---------------------BNDUPD--<    |
          | >--BNDACK-------------------->            |
         ...                                         ...
          |                                           |
          |        <---------------------BNDUPD--<    |
          | >--BNDACK-------------------->            |
          |                                           |
          |        <--------------------UPDDONE--<    |
   CONFLICT-DONE                                      |
          | >--STATE--(CONFLICT-DONE)---->            |
          |        <---------------------UPDREQ--<    |
          |                                           |
          | >--BNDUPD-------------------->            |
          |        <---------------------BNDACK--<    |
         ...                                         ...
          | >--BNDUPD-------------------->            |
          |        <---------------------BNDACK--<    |
          |                                           |
          | >--UPDDONE------------------->            |
          |                                         NORMAL
          |        <------------STATE--(NORMAL)--<    |
      NORMAL                                          |
          | >--STATE--(NORMAL)----------->            |
          |                                           |
          |        <--------------------POOLREQ--<    |
          | >------POOLRESP-(n)---------->            |
          |              addresses                    |
```

Figure 4: Transition out of POTENTIAL-CONFLICT

9.11. RESOLUTION-INTERRUPTED State

This state indicates that the two servers were attempting to reintegrate with each other in POTENTIAL-CONFLICT state, but communications failed prior to completion of re-integration.

If the servers remained in POTENTIAL-CONFLICT while communications was interrupted, neither server would be responsive to DHCP client requests, and if one server had crashed, then there might be no server able to process DHCP requests.

When a server enters RESOLUTION-INTERRUPTED state it SHOULD raise an alarm condition to alert administrative staff of a problem in the DHCP subsystem.

9.11.1. Operation in RESOLUTION-INTERRUPTED State

In this state a server MUST respond to all DHCP client requests. When allocating new resources (addresses or prefixes), each server SHOULD allocate from its own pool (if that can be determined), where the primary SHOULD allocate only FREE resources, and the secondary SHOULD allocate only BACKUP resources. When responding to renewal requests, each server will allow continued renewal of a DHCP client's current lease irrespective of whether that lease was given out by the receiving server or not, although the renewal period MUST NOT exceed the maximum client lead time (MCLT) beyond the latest of: 1) the potential valid lifetime already acknowledged by the other server or 2) the lease-expiration-time or 3) potential valid lifetime received from the partner server.

However, since the server cannot communicate with its partner in this state, the acknowledged potential valid lifetime will not be updated in any new bindings.

9.11.2. Transition Out of RESOLUTION-INTERRUPTED State

If an external command is received by a server in RESOLUTION-INTERRUPTED state informing it that its partner is down, it will transition immediately into PARTNER-DOWN state.

If communications is restored with the other server, then the server in RESOLUTION-INTERRUPTED state will transition into POTENTIAL-CONFLICT state.

9.12. CONFLICT-DONE State

This state indicates that during the process where the two servers are attempting to re-integrate with each other, the primary server has received all of the updates from the secondary server. It make a transition into CONFLICT-DONE state in order that it may be totally responsive to the client load, as opposed to NORMAL state where it would be in a "balanced" responsive state, running the load balancing algorithm.

TODO: We do not support load balancing, so CONFLICT-DONE is actually equal to NORMAL. Need to remove CONFLICT-DONE and replace all its references to NORMAL.

9.12.1. Operation in CONFLICT-DONE State

A primary server in CONFLICT-DONE state is fully responsive to all DHCP clients (similar to the situation in COMMUNICATIONS-INTERRUPTED state).

If communications fails, remain in CONFLICT-DONE state. If communications becomes OK, remain in CONFLICT-DONE state until the conditions for transition out become satisfied.

9.12.2. Transition Out of CONFLICT-DONE State

If communications fails with the partner while in CONFLICT-DONE state, then the server will remain in CONFLICT-DONE state.

When a primary server determines that the secondary server has made a transition into NORMAL state, the primary server will also transition into NORMAL state.

9.13. PAUSED State

TODO: Remove PAUSED state completely

This state exists to allow one server to inform another that it will be out of service for what is predicted to be a relatively short time, and to allow the other server to transition to COMMUNICATIONS-INTERRUPTED state immediately and to begin servicing all DHCP clients with no interruption in service to new DHCP clients.

A server which is aware that it is shutting down temporarily SHOULD send a STATE message with the server-state option containing PAUSED state and close the TCP connection.

While a server may or may not transition internally into PAUSED state, the 'previous' state determined when it is restarted MUST be the state the server was in prior to receiving the command to shut-down and restart and which precedes its entry into the PAUSED state. See Section 9.3.2 concerning the use of the previous state upon server restart.

When entering PAUSED state, the server MUST store the previous state in stable storage, and use that state as the previous state when it is restarted.

9.13.1. Operation in PAUSED State

Server MUST NOT perform any operation while in PAUSED state.

9.13.2. Transition Out of PAUSED State

A server makes a transition out of PAUSED state by being restarted. At that time, the previous state MUST be the state the server was in prior to entering the PAUSED state.

9.14. SHUTDOWN State

This state exists to allow one server to inform another that it will be out of service for what is predicted to be a relatively long time, and to allow the other server to transition immediately to PARTNER-DOWN state, and take over completely for the server going down.

When entering SHUTDOWN state, the server MUST record the previous state in stable storage for use when the server is restarted. It also MUST record the current time as the last time operational.

A server which is aware that it is shutting down SHOULD send a STATE message with the server-state field containing SHUTDOWN.

9.14.1. Operation in SHUTDOWN State

A server in SHUTDOWN state MUST NOT respond to any DHCP client input.

If a server receives any message indicating that the partner has moved to PARTNER-DOWN state while it is in SHUTDOWN state then it MUST record RECOVER state as the previous state to be used when it is restarted.

A server SHOULD wait for a few seconds after informing the partner of entry into SHUTDOWN state (if communications are okay) to determine if the partner entered PARTNER-DOWN state.

9.14.2. Transition Out of SHUTDOWN State

A server makes a transition out of SHUTDOWN state by being restarted.

10. Proposed extensions

The following section discusses possible extensions to the proposed failover mechanism. Listed extensions must be sufficiently simple to not further complicate failover protocol. Any proposals that are considered complex will be defined as stand-alone extensions in separate documents.

10.1. Active-active mode

A very simple way to achieve active-active mode is to remove the restriction that seconary server MUST NOT respond to SOLICIT and REQUEST messages. Instead it could respond, but MUST have lower preference than primary server. Clients discovering available servers will receive ADVERTISE messages from both servers, but are expected to select the primary server as it has higher preference value configured. The following REQUEST message will be directed to primary server.

Discussion: Do DHCPv6 clients actually do this? DHCPv4 clients were rumored to wait for a "while" to accept the best offer, but to a first approximation, they all take the first offer they receive that is even acceptable.

The benefit of this approach, compared to the "basic" active--passive solution is that there is no delay between primary failure and the moment when secondary starts serving requests.

Discussion: The possibility of setting both servers preference to an equal value could theoretically work as a crude attempt to provide load balancing. It wouldn't do much good on its own, as one (faster) server could be chosen more frequently (assuming that with equal preference sets clients will pick first responding server, which is not mandated by DHCPv6). We could design a simple mechanism of dynamically updating preference depending on usage of available resources. This concept hasn't been investigated in detail yet.

11. Dynamic DNS Considerations

TODO: Descibe DNS Updates challenges in failover environment. It is nicely described in Section 5.12 of [dhcpv4-failover].

12. Reservations and failover

TODO: Describe how lease reservation works with failover. See Section 5.13 in [dhcpv4-failover].

13. Protocol entities

Discussion: It is unclear if following sections belong to design or protocol draft. It is currently kept here as a scratchbook with list of things that will have to be defined eventually. Whether or not it will stay in this document or will be moved to the protocol spec

document is TBD.

13.1. Failover Protocol

This section enumerates list of options that will be defined in failover protocol specification. Rough description of purpose and content for each option is specified. Exact on wire format will be defined in protocol specification.

1. OPTION_FO_TIMESTAMP - convey information about timestamp. It is used by time skew measurement algorithm (see Section 8.1).

13.2. Protocol constants

This section enumerates various constants that have to be defined in actual protocol specification.

1. TIME_SKEW_PKTS_AVG - number of packets that are used to calculate average time skew between partners. See (see Section 8.1).

14. Open questions

This is scratchbook. This section will be removed once questions are answered.

Q: Do we want to support temporary addresses? I think not. They are short-lived by definition, so clients should not mind getting new temporary addresses.

Q: Do we want to support CGA-registered addresses? There is currently work in DHC WG about this, but I haven't looked at it yet. If that is complicated, we may not define it here, but rather as an extension. [If it moves forward, we need to support it.]

15. Security Considerations

TODO: Security considerations section will contain loose notes and will be transformed into consistent text once the core design solidifies.

16. IANA Considerations

IANA is not requested to perform any actions at this time.

17. Acknowledgements

This document extensively uses concepts, definitions and other parts of [dhcpv4-failover] document. Authors would like to thank Shawn Routher, Greg Rabil, and Bernie Volz for their significant involvement and contributions. Authors would like to thank VithalPrasad Gaitonde for his insightful comments.

This work has been partially supported by Department of Computer Communications (a division of Gdansk University of Technology) and the Polish Ministry of Science and Higher Education under the European Regional Development Fund, Grant No. POIG.01.01.02-00-045/09-00 (Future Internet Engineering Project).

18. References

18.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.

[RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, December 2003.

[RFC4704] Volz, B., "The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Client Fully Qualified Domain Name (FQDN) Option", RFC 4704, October 2006.

[RFC5460] Stapp, M., "DHCPv6 Bulk Leasequery", RFC 5460, February 2009.

18.2. Informative References

[I-D.ietf-dhc-dhcpv6-failover-requirements]
Mrugalski, T. and K. Kinnear, "DHCPv6 Failover Requirements", draft-ietf-dhc-dhcpv6-failover-requirements-00 (work in progress), October 2011.

[I-D.ietf-dhc-dhcpv6-redundancy-consider]
Tremblay, J., Brzozowski, J., Chen, J., and T. Mrugalski, "DHCPv6 Redundancy Deployment Considerations",

draft-ietf-dhc-dhcpv6-redundancy-consider-02 (work in progress), October 2011.

[RFC2136] Vixie, P., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.

[dhcpv4-failover] Droms, R., Kinnear, K., Stapp, M., Volz, B., Gonczi, S., Rabil, G., Dooley, M., and A. Kapur, "DHCP Failover Protocol", draft-ietf-dhc-failover-12 (work in progress), March 2003.

Authors' Addresses

Tomasz Mrugalski
Internet Systems Consortium, Inc.
950 Charter Street
Redwood City, CA 94063
USA

Phone: +1 650 423 1345
Email: tomasz.mrugalski@gmail.com

Kim Kinnear
Cisco Systems, Inc.
1414 Massachusetts Ave.
Boxborough, Massachusetts 01719
USA

Phone: +1 (978) 936-0000
Email: kkinnear@cisco.com