

Dynamic Host Configuration (DHC)
Internet-Draft
Intended status: Standards Track
Expires: August 31, 2017

T. Mrugalski
ISC
K. Kinnear
Cisco
February 27, 2017

DHCPv6 Failover Protocol
draft-ietf-dhc-dhcpv6-failover-protocol-06

Abstract

DHCPv6 as defined in "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)" (RFC3315) does not offer server redundancy. This document defines a protocol implementation to provide DHCPv6 failover, a mechanism for running two servers with the capability for either server to take over clients' leases in case of server failure or network partition. It meets the requirements for DHCPv6 failover detailed in "DHCPv6 Failover Requirements" (RFC7031).

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 August 31, 2017.

Copyright Notice

Copyright (c) 2017 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. Introduction	4
2. Requirements Language	5
3. Glossary	5
4. Failover Concepts and Mechanisms	9
4.1. Required Server Configuration	9
4.2. IPv6 Address and Delegable Prefix Allocation	9
4.2.1. Independent Allocation	9
4.2.2. Proportional Allocation	10
4.3. Lazy Updates	13
4.4. Maximum Client Lead Time (MCLT)	13
4.4.1. MCLT example	15
5. Message and Option Definitions	18
5.1. Message Framing for TCP	18
5.2. Failover Message Format	18
5.3. Messages	19
5.3.1. BNDUPD	19
5.3.2. BNDREPLY	20
5.3.3. POOLREQ	20
5.3.4. POOLRESP	20
5.3.5. UPDREQ	20
5.3.6. UPDREQALL	20
5.3.7. UPDDONE	21
5.3.8. CONNECT	21
5.3.9. CONNECTREPLY	21
5.3.10. DISCONNECT	21
5.3.11. STATE	21
5.3.12. CONTACT	21
5.4. Transaction Id	22
5.5. Options	22
5.5.1. OPTION_F_BINDING_STATUS	22
5.5.2. OPTION_F_CONNECT_FLAGS	23
5.5.3. OPTION_F_DNS_REMOVAL_INFO	24
5.5.4. OPTION_F_DNS_HOST_NAME	25
5.5.5. OPTION_F_DNS_ZONE_NAME	25
5.5.6. OPTION_F_DNS_FLAGS	26
5.5.7. OPTION_F_EXPIRATION_TIME	27
5.5.8. OPTION_F_MAX_UNACKED_BNDUPD	28
5.5.9. OPTION_F_MCLT	28
5.5.10. OPTION_F_PARTNER_LIFETIME	29
5.5.11. OPTION_F_PARTNER_LIFETIME_SENT	29
5.5.12. OPTION_F_PARTNER_DOWN_TIME	30
5.5.13. OPTION_F_PARTNER_RAW_CLT_TIME	31

5.5.14.	OPTION_F_PROTOCOL_VERSION	31
5.5.15.	OPTION_F_KEEPLIVE_TIME	32
5.5.16.	OPTION_F_RECONFIGURE_DATA	32
5.5.17.	OPTION_F_RELATIONSHIP_NAME	33
5.5.18.	OPTION_F_SERVER_FLAGS	34
5.5.19.	OPTION_F_SERVER_STATE	35
5.5.20.	OPTION_F_START_TIME_OF_STATE	36
5.5.21.	OPTION_F_STATE_EXPIRATION_TIME	37
5.6.	Status Codes	38
6.	Connection Management	39
6.1.	Creating Connections	39
6.1.1.	Sending a CONNECT message	40
6.1.2.	Receiving a CONNECT message	41
6.1.3.	Receiving a CONNECTREPLY message	42
6.2.	Endpoint Identification	43
6.3.	Sending a STATE message	44
6.4.	Receiving a STATE message	44
6.5.	Connection Maintenance Parameters	45
6.6.	Unreachability detection	45
7.	Binding Updates and Acks	46
7.1.	Time Skew	46
7.2.	Information model	46
7.3.	Times Required for Exchanging Binding Updates	50
7.4.	Sending Binding Updates	51
7.5.	Receiving Binding Updates	54
7.5.1.	Monitoring Time Skew	54
7.5.2.	Acknowledging Reception	55
7.5.3.	Processing Binding Updates	55
7.5.4.	Accept or Reject?	55
7.5.5.	Accepting Updates	58
7.6.	Sending Binding Replies	59
7.7.	Receiving Binding Acks	61
7.8.	BNDUPD/BNDREPLY Data Flow	62
8.	Endpoint States	63
8.1.	State Machine Operation	63
8.2.	State Machine Initialization	67
8.3.	STARTUP State	67
8.3.1.	Operation in STARTUP State	68
8.3.2.	Transition Out of STARTUP State	68
8.4.	PARTNER-DOWN State	70
8.4.1.	Operation in PARTNER-DOWN State	70
8.4.2.	Transition Out of PARTNER-DOWN State	71
8.5.	RECOVER State	71
8.5.1.	Operation in RECOVER State	72
8.5.2.	Transition Out of RECOVER State	72
8.6.	RECOVER-WAIT State	73
8.6.1.	Operation in RECOVER-WAIT State	74
8.6.2.	Transition Out of RECOVER-WAIT State	74

8.7.	RECOVER-DONE State	74
8.7.1.	Operation in RECOVER-DONE State	74
8.7.2.	Transition Out of RECOVER-DONE State	75
8.8.	NORMAL State	75
8.8.1.	Operation in NORMAL State	75
8.8.2.	Transition Out of NORMAL State	76
8.9.	COMMUNICATIONS-INTERRUPTED State	77
8.9.1.	Operation in COMMUNICATIONS-INTERRUPTED State	77
8.9.2.	Transition Out of COMMUNICATIONS-INTERRUPTED State	78
8.10.	POTENTIAL-CONFLICT State	79
8.10.1.	Operation in POTENTIAL-CONFLICT State	80
8.10.2.	Transition Out of POTENTIAL-CONFLICT State	80
8.11.	RESOLUTION-INTERRUPTED State	81
8.11.1.	Operation in RESOLUTION-INTERRUPTED State	82
8.11.2.	Transition Out of RESOLUTION-INTERRUPTED State	82
8.12.	CONFLICT-DONE State	82
8.12.1.	Operation in CONFLICT-DONE State	83
8.12.2.	Transition Out of CONFLICT-DONE State	83
9.	DNS Update Considerations	83
9.1.	Relationship between failover and DNS update	84
9.2.	Exchanging DNS Update Information	85
9.3.	Adding RRs to the DNS	87
9.4.	Deleting RRs from the DNS	87
9.5.	Name Assignment with No Update of DNS	88
10.	Security Considerations	88
11.	IANA Considerations	89
12.	Acknowledgements	91
13.	References	92
13.1.	Normative References	92
13.2.	Informative References	93
	Authors' Addresses	93

1. Introduction

This document defines a DHCPv6 failover protocol, which is a mechanism for running two DHCPv6 servers [RFC3315] with the capability for either server to take over clients' leases in case of server failover or network partition. For a general overview of DHCPv6 failover problems, use cases, benefits, and shortcomings, see [RFC7031].

The failover protocol provides a means for cooperating DHCP servers to work together to provide a service to DHCP clients with availability that is increased beyond that which could be provided by a single DHCP server operating alone. It is designed to protect DHCP 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 for an IPv6 address [RFC3315]

or on an IPv6 prefix [RFC3633] without the DHCP client experiencing lease expiration or a reassignment of a lease to a different IPv6 address or prefix in the event of failure by one or the other of the two servers.

The failover protocol defines an active-passive mode, sometimes also called a 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 receives clients' requests, but responds only to those specifically directed to it). The secondary maintains a copy of the binding database and is ready to take over all incoming queries in case of primary server failure.

The failover protocol is designed to provide lease stability for leases with valid lifetimes beyond a short period. The DHCPv6 Failover protocol MUST NOT be used for new leases shorter than 30 seconds. Leases reaching the end of their lifetime are not affected by this restriction.

The failover protocol fulfills all DHCPv6 failover requirements defined in [RFC7031].

2. 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].

3. Glossary

This is a supplemental glossary that should be combined with definitions in Section 3 of RFC 7031 [RFC7031].

o Absolute Time

The time in seconds since midnight January 1, 2000 UTC, modulo 2^{32}).

o Address Lease

A lease involving an IPv6 address. Typically used when it is necessary to distinguish the lease for an IPv6 address from a lease for a DHCP prefix. See "delegated prefix" and "prefix lease".

o auto-partner-down

A capability where a failover server will move from COMMUNICATIONS-INTERRUPTED state to PARTNER-DOWN state automatically, without operator intervention.

- o Available (Lease or Prefix)

A lease or delegable prefix is available if it could be allocated for use by a DHCP client. It is available on the main server when it is in FREE state and available on the secondary server when it is in the FREE-BACKUP state. Sometimes the term "available" is used when it would be awkward to say "FREE on the primary server and FREE-BACKUP on the secondary server".

- o Binding-Status

A lease can hold a variety of states (see Section 5.5.1 for a list) and these are also referred to as the binding-status of the lease.

- o Delegable Prefix

A prefix from which other prefixes may be delegated, using the mechanisms described in [RFC3633]. A prefix which has been delegated is known as a "delegated prefix" or a "prefix lease".

- o Delegated Prefix

A prefix which has been delegated to a DHCP client as described in [RFC3633]. Depending on the context, a delegated prefix may also be described as a "prefix lease", when it is necessary to distinguish it from an "address lease".

- o Failover endpoint

The failover protocol allows for there to be a unique failover 'endpoint' for each failover relationship in which a failover server participates. The failover relationship is defined by a relationship name, and includes the failover partner IP address, the role this server takes with respect to that partner (primary or secondary), and the prefixes from which addresses can be leased as well as prefixes from which other prefixes can be delegated (delegable prefixes), associated with that relationship. The failover endpoint can take actions and hold unique states. Typically, there is one failover endpoint per partner (server), although there may be more.

- o Failover communication

All messages exchanged between partners.

- o Independent Allocation

An allocation algorithm that splits the available pool of address leases between the primary and secondary servers. It is used for IPv6 address allocations. See Section 4.2.1.

- o Lease

An association of a DHCP client with an IPv6 address or delegated prefix. This might refer to either an existing association or a potential association.

- o MCLT

Maximum Client Lead Time. The fundamental relationship on which much of the correctness of the failover protocol depends is that the lease expiration time known to a DHCP client MUST NOT be greater by more than the MCLT beyond the later of the partner lifetime time acknowledged by that server's failover partner or the current time (i.e., now). See Section 4.4.

- o Partner

The other DHCP server that participates in a failover relationship. When the role (primary or secondary) is not important, the other server is referred to as a "failover partner" or sometimes simply "partner".

- o Prefix Lease

A lease involving a prefix that is or could be delegated, as opposed to a lease for a single IPv6 address. A prefix lease can also be described as a "delegated prefix".

- o Primary Server

First out of two DHCP servers that participate in a failover relationship. When both servers are operating this server handles most of the client traffic. Its failover partner is referred to as secondary server.

- o Proportional Allocation

An allocation algorithm that splits the delegable prefixes between the primary and secondary servers and maintains a more or less

fixed proportion of the delegable prefixes between both servers.
Section 4.2.2.

- o Renew Responsive

A server that is renew responsive will respond to valid DHCP client requests that are directed to it by having an `OPTION_SERVERID` option in the message which contains the DUID of the renew responsive server. See [RFC3315].

- o Responsive

A server that is responsive will respond to all valid DHCP client requests.

- o Secondary Server

Second of two DHCP servers that participate in a failover relationship. Its failover partner is referred to as the primary server. When both servers are operating this server (the secondary) typically does not handle client traffic and acts as a backup to the primary server. It will respond, however, to RENEW requests directed specifically to it.

- o Server

A DHCP server that implements DHCPv6 failover. 'Server' and 'failover endpoint' are synonymous only if the server participates in only one failover relationship.

- o State

The term state is used in two ways in this document. A failover endpoint is always in some state, and there are a series of states that a failover endpoint can move through. See Section 8 for details of the failover endpoint states. A lease also has a state, and this is sometimes referred to as a binding-status. See Section 5.5.1 for a list of the states a lease can hold.

- o Time Context

Each failover server has a clock and a definite idea of the current universal time. Each server's idea of the current time is considered its time context.

- o Unresponsive

A server that is unresponsive will not respond to DHCP client requests.

4. Failover Concepts and Mechanisms

The following concepts and mechanisms are necessary to the operation of the failover protocol, and they are not currently employed by the DHCPv6 protocol [RFC3315]. The failover protocol provides support for all of these concepts and mechanisms.

4.1. Required Server Configuration

Servers frequently have several kinds of leases available on a particular network segment. The failover protocol assumes that both primary and secondary servers are configured identically with regard to the prefixes and links involved in DHCP. For delegated prefixes (involved in proportional allocation) the primary server is responsible for allocating to the secondary server the correct proportion of the available delegated prefixes. IPv6 addresses (involved in independent allocation) are allocated to the primary and secondary servers algorithmically, and do not require an explicit message transfer to be distributed.

4.2. IPv6 Address and Delegable Prefix Allocation

Currently there are two allocation algorithms defined, one for address leases and one for prefix leases.

4.2.1. Independent Allocation

In this allocation scheme, used for allocating individual IPv6 addresses, available IPv6 addresses are permanently (until server configuration changes) split between servers. Available IPv6 addresses are split between the primary and secondary servers as part of initial connection establishment. Once IPv6 addresses are allocated to each server, there is no need to reassign them. The IPv6 address allocation is algorithmic in nature, and does not require a message exchange for each server to know which IPv6 addresses it has been allocated. This algorithm is simpler than proportional allocation since it does not require a rebalancing mechanism. It also assumes that the pool assigned to each server will never deplete.

Once each server is assigned a pool of IPv6 addresses during initial connection establishment, it may allocate its assigned IPv6 addresses to clients. Once a client releases a lease or its lease on an IPv6 address expires, the returned IPv6 address returns to the pool for the server that leased it. A lease on an IPv6 address can be renewed

by a responsive server or by a renew responsive server. When an IPv6 address goes PENDING-FREE (see Section 7.2) it is owned by whichever server it is allocated to by the independent allocation algorithm.

IPv6 addresses (which use the independent allocation approach) are ignored when a server processes a POOLREQ message.

During COMMUNICATION-INTERRUPTED events, a partner MAY continue extending existing address leases as requested by clients. An operational partner MUST NOT lease IPv6 addresses that were assigned to its downed partner and later expired or were released or declined by a client. When it is in PARTNER-DOWN state, a server MUST allocate new leases from its own pool. It MUST NOT use its partner's pool to allocate new leases.

4.2.1.1. Independent Allocation Algorithm

For each address that can be allocated, the primary server MUST allocate only IPv6 addresses when the low-order bit (i.e., bit 127) is equal to 1, and the secondary server MUST allocate only the IPv6 addresses when the low-order bit (i.e., bit 127) is equal to 0.

4.2.2. Proportional Allocation

In this allocation scheme, each server has its own pool of prefixes available for delegation, known as "delegable prefixes". These delegable prefixes may be prefixes from which other prefixes can be delegated or they may be prefixes which are the correct size for delegation but are not, at present, delegated to a particular client. Remaining delegable prefixes are split between the primary and secondary servers in a configured proportion. Note that a delegated prefix (also known as a prefix lease) is not "owned" by a particular server. Only a delegable prefix which is available is "owned" by a particular server -- once it has been delegated (leased) to a client it becomes a prefix lease and 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.

The flow of a delegable prefix is as follows: initially the delegable prefix is part of a larger delegable prefix, all of which are initially owned by the primary server. A delegable prefix may be allocated to the secondary server and then it is owned by the secondary server. Either server can allocate and delegate prefixes out of the delegable prefixes which they own. Once these prefixes are delegated (leased) to clients, the servers cease to own them and they are owned by the clients to which they have been delegated (leased). When the client releases the delegated prefix or the lease

on it expires, it will again become available and will then be a delegable prefix and be owned by the primary.

A server delegates prefixes only from its own pool of delegable prefixes in all states except for PARTNER-DOWN. In PARTNER-DOWN state the operational partner can delegate prefixes from either pool (both its own, and its partner's after some time constraints have elapsed). The operational partner SHOULD allocate from its own pool before using its partner's pool. The allocation and maintenance of these pools of delegable prefixes is important, since the goal is to maintain a more or less constant ratio of delegable prefixes between the two servers.

Each server knows which delegable prefixes are in its own pool as well as which are in its partner's pool, so that it can allocate delegable prefixes from its partner's pool without communication with its partner if that becomes necessary.

The initial allocation of delegable prefixes from the primary to the secondary when the servers first integrate is triggered by the POOLREQ message from the secondary to the primary. This is followed (at some point) by the POOLRESP message where the primary tells the secondary that it received and processed the POOLREQ message. The primary sends the allocated delegable prefixes to the secondary as prefix leases via BNDUPD messages. The POOLRESP message may be sent before, during, or at the completion of the BNDUPD message exchanges that were triggered by the POOLREQ message. The POOLREQ/POOLRESP message exchange is a trigger to the primary to perform a scan of its database and to ensure that the secondary has enough delegable prefixes (based on some configured ratio).

The delegable prefixes are sent to the secondary as prefix leases using the BNDUPD message containing an OPTION_IAPREFIX with a state of FREE-BACKUP, which indicates the prefix lease is now available for allocation by the secondary. Once the message is sent, the primary MUST NOT use these prefixes for allocation to DHCP clients (except when the server is in PARTNER-DOWN state).

The POOLREQ/POOLRESP message exchange initiated by the secondary is valid at any time both partners remain in contact, and the primary server SHOULD, whenever it receives the POOLREQ message, scan its database of delegable prefixes and determine if the secondary needs more delegable prefixes from any of the delegable prefixes which it currently owns.

In order to support a reasonably dynamic balance of the leases between the failover partners, the primary server needs to do additional work to ensure that the secondary server has as many

delegable prefixes as it needs (but that it doesn't have more than it needs).

The primary server SHOULD examine the balance of delegable prefixes between the primary and secondary for a particular prefix whenever the number of possibly delegable prefixes for either the primary or secondary changes by more than a predetermined amount. Typically this comparison would not involve actually comparing the count of existing instances of delegable prefixes, but would instead involve determining the number prefixes that could be delegated given the address ranges of the delegable prefixes allocated to each server. The primary server SHOULD adjust the delegable prefix balance as required to ensure the configured delegable prefix balance, excepting that the primary server SHOULD employ some threshold mechanism to such a balance adjustment in order to minimize the overhead of maintaining this balance.

The primary server can, at any time, send an available delegable prefix to the secondary using a BNDUPD with the state FREE-BACKUP. The primary server can attempt to take an available delegable prefix away from the secondary by sending a BNDUPD with the state FREE. If the secondary accepts the BNDUPD, then the lease is now available to the primary and not available to the secondary. Of course, the secondary MUST reject that BNDUPD if it has already allocated that lease to a DHCP client.

4.2.2.1. Re-allocating Leases

When in PARTNER-DOWN state there is a waiting period after which a delegated prefix can be re-allocated to another client. For delegable prefixes which are "available" when the server enters PARTNER-DOWN state, the period is the MCLT from the entry into PARTNER-DOWN state. For delegated prefixes which are not available when the server enters PARTNER-DOWN state, the period is the MCLT after the later of the following times: the acked-partner-lifetime, the partner-lifetime (if any), the expiration-time, and the entry to PARTNER-DOWN time plus the MCLT.

In any other state, a server cannot reallocate a delegated prefix from one client to another without first notifying its partner (through a BNDUPD message) and receiving acknowledgement (through a BNDREPLY message) that its partner is aware that the first client is not using the lease.

Specifically, an "available" delegable prefix on a server may be allocated to any client. A prefix which was delegated (leased) to a client and which expired or was released by that client would take on a new state, EXPIRED or RELEASED respectively. The partner server

would then be notified that this delegated prefix was EXPIRED or RELEASED through a BNDUPD. When the sending server received the BNDREPLY for that delegated prefix showing it was FREE, it would move the lease from EXPIRED or RELEASED to FREE, and it would be available for allocation by the primary server to any clients.

A server MAY reallocate a delegated prefix in the EXPIRED or RELEASED state to the same client with no restrictions provided it has not sent a BNDUPD message regarding the delegated prefix to its partner. This situation would exist if the prefix lease expired or was released after the transition into PARTNER-DOWN state, for instance.

4.3. Lazy Updates

The DHCPv6 Failover Requirements document includes the requirement that failover must not introduce significant performance impact on server response times (see Sections 7 and 5.2.2 of [RFC7031]). In order to realize this requirement a server implementing the failover protocol must be able respond to a DHCP client without waiting to update its failover partner whenever the binding database changes. The lazy update mechanism allows a server to allocate a new lease or extend an existing lease, respond to the DHCP client, 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 DHCP client with a particular valid lifetime and before updating its failover partner, the failover partner will eventually believe that the client's lease has expired -- even though the DHCP client still retains a valid lease on that address or prefix. It is also possible that the failover partner will have no record at all of the lease being assigned to the DHCP client. Both of these issues are dealt with by use of the MCLT when allocating or extending leases (see Section 4.4).

4.4. Maximum Client Lead Time (MCLT)

In order to handle problems introduced by lazy updates (see Section 4.3), 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 valid lifetime provided to a DHCP client by a server and the valid lifetime known by that server's failover partner.

The MCLT is typically much less than the valid lifetime that a server has been configured to offer a client, and so some strategy must

exist to allow a server to offer the configured valid lifetime to a client. During a lazy update the updating server updates its failover partner with a partner lifetime which is longer than the valid lifetime previously given to the DHCP client and which is longer than the valid lifetime that the server has been configured to give a client. This allows the server to give the configured valid lifetime to the client the next time the client renews its lease, since the time that it will give to the client will not be longer than the MCLT beyond the partner lifetime acknowledged by its partner or the current time.

The fundamental relationship on which the failover protocol depends is: the lease expiration time known to a DHCP client MUST NOT be greater by more than the MCLT beyond the later of the partner lifetime acknowledged by that server's failover partner and the current time.

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

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

In the following explanation, all of the lifetimes are "valid" lifetimes, in the context of [RFC3315].

The different times are:

desired lifetime:

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

actual lifetime:

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

partner lifetime:

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

acknowledged partner lifetime:

The acknowledged partner lifetime is the partner lifetime the partner server has most recently acknowledged in a BNDREPLY message.

4.4.1. MCLT example

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

When a server makes an offer for a new lease on an IPv6 address to a DHCP client, it determines the desired lifetime (in this case, 3 days). It then examines the acknowledged partner lifetime (which in this case is zero) and determines the remainder of the time left to run, which is also zero. It adds the MCLT to this value. Since the actual lifetime cannot be allowed to exceed the remainder of the current acknowledged partner lifetime plus the MCLT, the offer made to the client is for the remainder of the current acknowledged partner lifetime (i.e. zero) plus the MCLT. Thus, the actual lifetime is 1 hour (the MCLT).

Once the server has sent the REPLY to the DHCP client, it will update its failover partner with the lease information using a BNDUPD message. The partner lifetime will be composed of the T1 fraction (1/2) of the actual lifetime added to the desired lifetime. Thus, the failover partner is updated using a BNDUPD with a partner lifetime of 1/2 hour + 3 days.

When the primary server receives a BNDREPLY to its update of the secondary server's (partner's) partner lifetime, it records that as the acknowledged partner lifetime. A server MUST NOT send a BNDREPLY in response to a BNDUPD message until it is sure that the information in the BNDUPD message has been updated in its lease database. See Section 7.5.2. Thus, the primary server in this case can be sure that the secondary server has recorded the partner lifetime in its stable storage when the primary server receives a BNDREPLY message from the secondary server.

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

3 days. So the client may have its lease renewed for the desired lifetime -- 3 days.

When the primary DHCP server updates the secondary DHCP server after the DHCP client's renewal REPLY is complete, it will calculate the partner lifetime as the T1 fraction of the actual client lifetime (1/2 of 3 days this time = 1.5 days). To this it will add the desired lifetime of 3 days, yielding a total partner 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 lifetime so as to be able to always offer the client the desired lifetime.

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

Fundamental relationship:

lease time = floor(desired lifetime, ack-partner-lifetime + MCLT)

Initial conditions: MCLT = 1h, desired lifetime = 3d

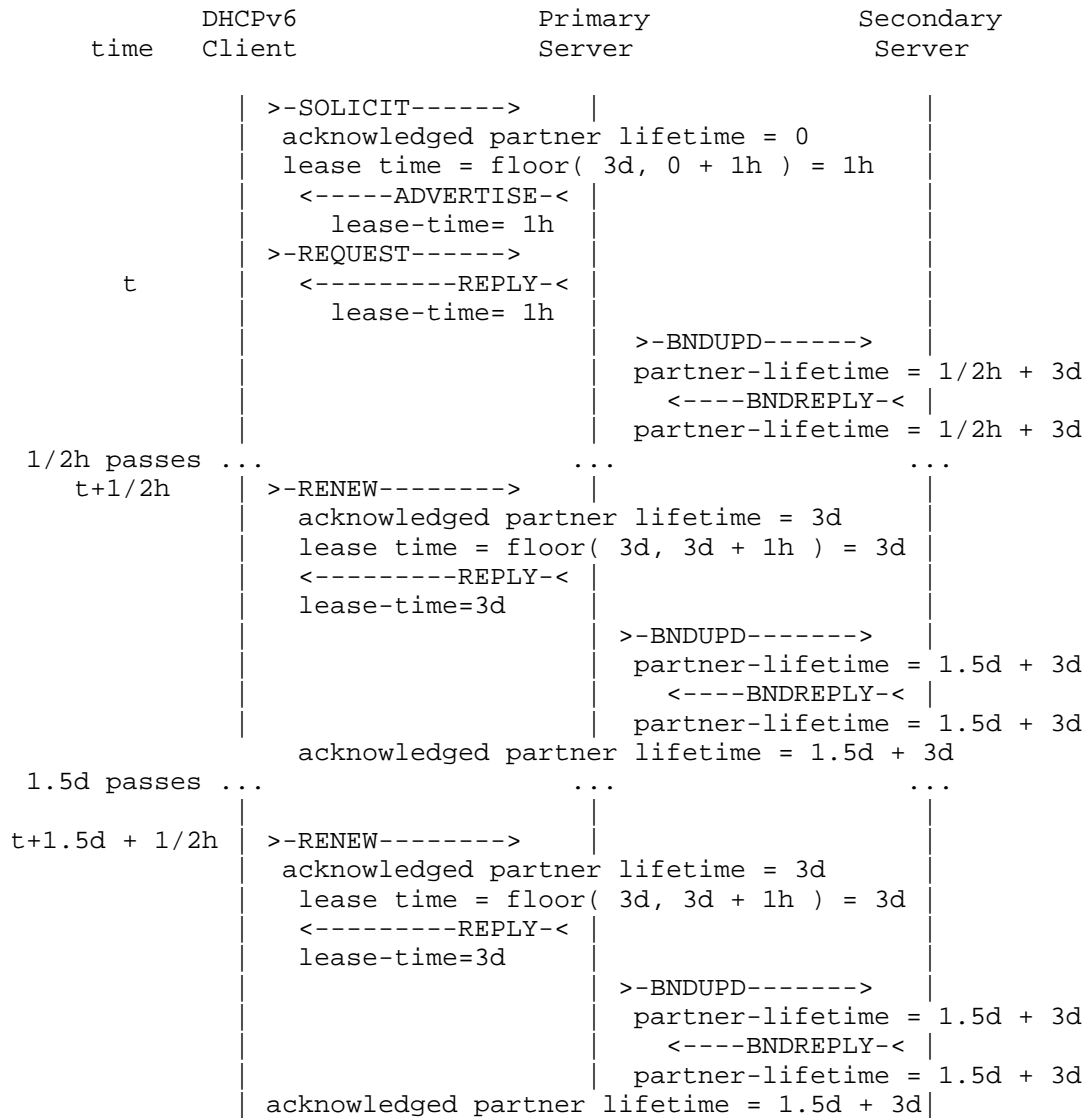


Figure 1: MCLT Example

5. Message and Option Definitions

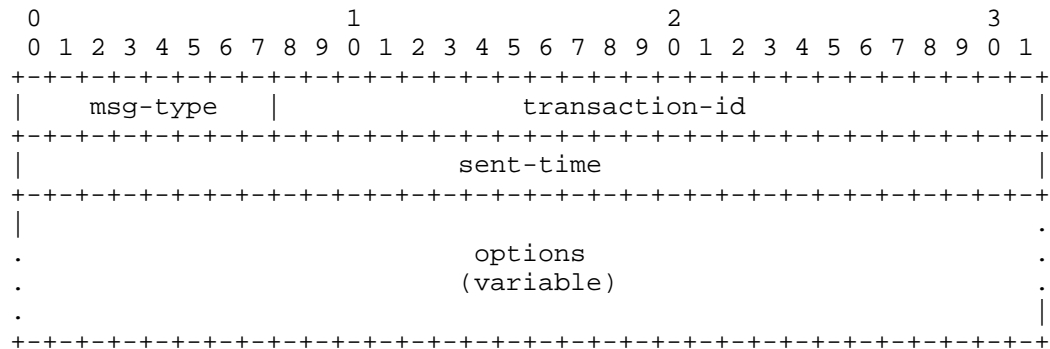
5.1. Message Framing for TCP

Failover communication is conducted over a TCP connection established between the partners. The protocol uses the framing format specified in Section 5.1 of DHCPv6 Bulk Leasequery [RFC5460], but uses different message types with a different message format, described in Section 5.2. The TCP connection between failover servers is made to a specific port, the dhcp-failover port, port 647. All information is sent over the connection as typical DHCP messages that convey DHCP options, following the format defined in Section 22.1 of [RFC3315].

5.2. Failover Message Format

All Failover messages defined below share a common format with a fixed size header and a variable format area for options. All values in the message header and in any included options are in network byte order.

The following diagram illustrates the format of DHCP messages exchanged between failover partners (which is compatible with the format described in Section 6 of [RFC3315]):



msg-type	Identifies the DHCP message type; the available message types are listed below. Note that since the TCP connection for failover is made to a unique port, the msg-type codes are allocated from a registry distinct from the Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Message Types registry.
transaction-id	The transaction ID for this message exchange.
sent-time	The time the message was transmitted (set as close to transmission as practical), in seconds since midnight (UTC), January 1, 2000, modulo 2 ³² . Used to determine the time skew of the failover partners.
options	Options carried in this message. These options are all defined in the Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Option Codes registry. A number of existing DHCPv6 options are used and several more are defined in this document.

5.3. Messages

The following list contains the new message types created for failover communication.

5.3.1. BNDUPD

The binding update message BNDUPD (TBD1) is used to send the binding lease changes to the partner. At most one OPTION_CLIENT_DATA option may appear in a BNDUPD message. Note that not all data in a BNDUPD

is sent in an `OPTION_CLIENT_DATA` option. Information about delegable prefixes not currently allocated to a particular client is sent in `BNDUPD` messages but not within `OPTION_CLIENT_DATA` options. The partner is expected to respond with a `BNDREPLY` message.

5.3.2. `BNDREPLY`

The binding acknowledgement message `BNDREPLY` (TBD2) is used for confirmation of the received `BNDUPD` message. It may contain a positive or negative response (e.g. due to detected lease conflict).

5.3.3. `POOLREQ`

The Pool Request message `POOLREQ` (TBD3) is used by the secondary server to request allocation of delegable prefixes from the primary server. The primary responds with `POOLRESP`.

5.3.4. `POOLRESP`

The Pool Response `POOLRESP` (TBD4) message is used by the primary server to indicate that it has received the secondary's servers request to ensure that delegable prefixes are balanced between the primary and secondary servers. It does not indicate that all of the `BNDUPDs` that might be created from any rebalancing have been sent or responded to, it only indicates reception and acceptance of the task of ensuring the balance is examined and corrected as necessary.

5.3.5. `UPDREQ`

The update request message `UPDREQ` (TBD5) is used by one server to request that its partner sends all binding database changes that have not yet been confirmed. The partner is expected to respond with zero or more `BNDUPD` messages, followed by an `UPDDONE` message that signals that all of the `BNDUPD` messages have been sent and a corresponding `BNDREPLY` message has been received for each of them.

5.3.6. `UPDREQALL`

The update request all `UPDREQALL` (TBD6) is used by one server to request that all binding database information present in the other server be sent to the requesting server, in order to recover from a total loss of its binding database by the requesting server. A server receiving this request responds with zero or more `BNDUPD` messages, followed by an `UPDDONE` that signals that all of the `BNDUPD` messages have been sent and a corresponding `BNDREPLY` message has been received for each of them.

5.3.7. UPDDONE

The update done message UPDDONE (TBD7) is used by the server responding to an UPDREQ or UPDREQALL to indicate that all requested updates have been sent by the responding server and acked by the requesting server.

5.3.8. CONNECT

The connect message CONNECT (TBD8) is used by the primary server to establish a failover connection with the secondary server, and to transmit several important configuration attributes items between the servers. The partner is expected to confirm by responding with CONNECTREPLY message.

5.3.9. CONNECTREPLY

The connect acknowledgement message CONNECTREPLY (TBD9) is used by the secondary server to respond to a CONNECT message from the primary server.

5.3.10. DISCONNECT

The disconnect message DISCONNECT (TBD10) is used by either server when closing a connection and shutting down. No response is required for this message. The DISCONNECT message SHOULD contain an OPTION_STATUS_CODE option with an appropriate status. Often this will be ServerShuttingDown. See Section 5.6. A server SHOULD include a descriptive message as to the reasons causing the disconnect message.

5.3.11. STATE

The state message STATE (TBD11) is used by either server to inform its partner about a change of failover state. In some cases it may be used to also inform the partner about the current state, e.g. after connection is established in COMMUNICATIONS-INTERRUPTED or PARTNER-DOWN states.

5.3.12. CONTACT

The contact message CONTACT (TBD12) is used by either server to ensure that its partner continues to see the connection as operational. It MUST be transmitted periodically over every established connection if other message traffic is not flowing, and it MAY be sent at any time. See Section 6.5.

5.4. Transaction Id

The initiator of a message exchange MUST set the transaction-id. This means that all of the messages above except BNDREPLY, POOLRESP, UPDDONE, and CONNECTREPLY must set the transaction-id. The transaction-id MUST be unique among all currently outstanding messages sent to the failover partner. A straightforward way to ensure this is to simply use an incrementing value, with one caveat. The UPDREQ and UPDREQALL messages may be separated by a considerable time prior to the receipt of an UPDDONE message. While the usual pattern of message exchange would have the partner doing the vast majority of message initiation, it is remotely possible that the partner which initiated the UPDREQ or UPDREQALL messages might also send enough messages to wrap the 24-bit transaction-id and duplicate the transaction-id of the outstanding UPDREQ or UPDREQALL. Thus, it is important to ensure that the transaction-id of a currently outstanding UPDREQ or UPDREQALL is not replicated in any message initiated prior to the receipt of the corresponding UPDDONE.

5.5. Options

The following new options are defined.

5.5.1. OPTION_F_BINDING_STATUS

The binding-status represents an implementation independent representation of the status (or the state) of a lease on an IPv6 address or prefix.

This is an unsigned byte.

The code for this option is TBD13.

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   OPTION_F_BINDING_STATUS   |         option-len         |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| binding-status |
+---+---+---+---+---+

```

```

option-code      OPTION_F_BINDING_STATUS (TBD13).
option-len       1.
binding-status    The binding-status. See below.

```

Value	binding-status
0	reserved
1	ACTIVE
2	EXPIRED
3	RELEASED
4	PENDING-FREE
5	FREE
6	FREE-BACKUP
7	ABANDONED
8	RESET

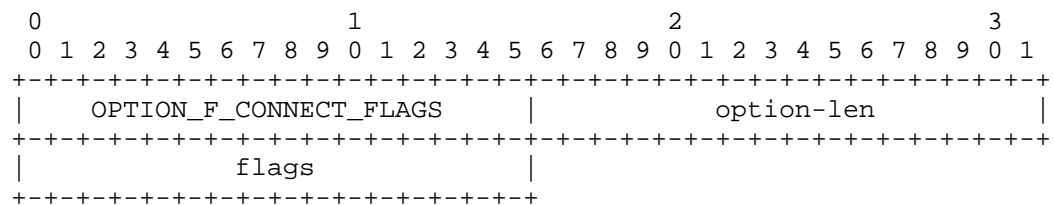
The binding-status values are discussed in Section 7.2

5.5.2. OPTION_F_CONNECT_FLAGS

Flags which indicate attributes of the connecting server.

This consists of an unsigned 16 bit value in network byte order.

The code for this option is TBD14.



```
option-code    OPTION_F_CONNECT_FLAGS (TBD14).
option-len     2.
flags          flag bits, see below:
```

```

      0                               1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+
|           MBZ                     |F|
+---+---+---+---+---+---+---+---+

```

The bits (numbered from the most-significant bit in network byte-order) are used as follows:

- 0-14: MBZ
Must be zero
- 15 (F): FIXED_PD_LENGTH
Set to 1 to indicate that all prefixes delegated from a given delegable prefix have the same prefix length (size). If this is not set, the prefixes delegated from one delegable prefix may have different sizes.

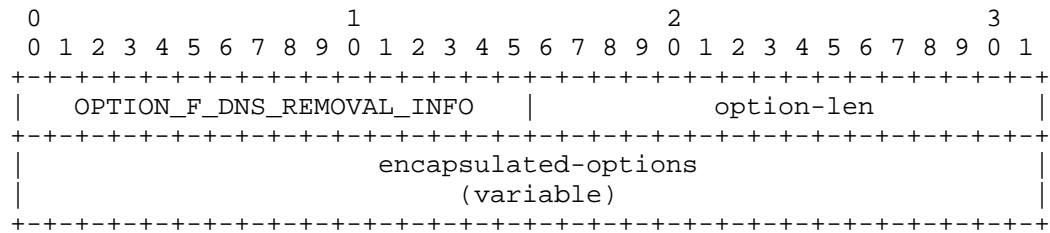
If the FIXED_PD_LENGTH bit is not set, it indicates that prefixes of a range of sizes can be delegated from a given delegable prefix. Note that if the FIXED_PD_LENGTH bit is set, each delegable prefix may have its own fixed size -- this flag does not imply that all prefixes delegated will be the same size, rather that all prefixes delegated from the same delegable prefix will be the same size.

If the FIXED_PD_LENGTH bit is set, the length used for each prefix is specified independent of the failover protocol, but must be known to both failover partners. It might be specified in the configuration for each delegable prefix or it might be fixed for the entire server.

5.5.3. OPTION_F_DNS_REMOVAL_INFO

This option contains the information necessary to remove a DNS name that was entered by the failover partner.

The code for this option is TBD15.



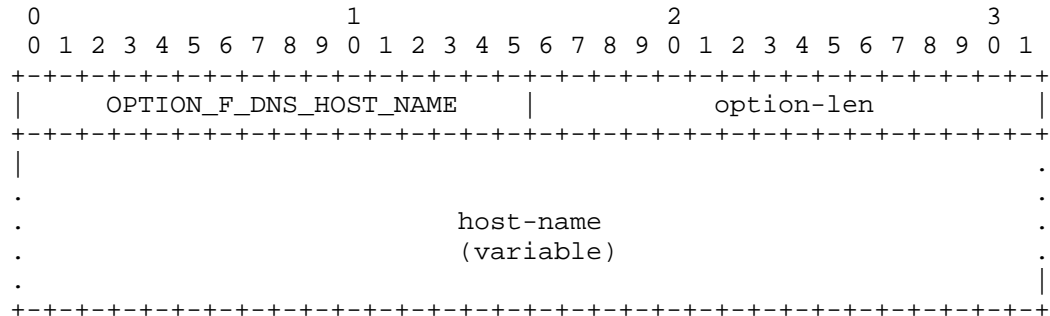
option-code OPTION_F_DNS_REMOVAL_INFO (TBD15).
option-len variable
sub-options Three possible encapsulated options:
 OPTION_F_DNS_HOST_NAME
 OPTION_F_DNS_ZONE_NAME
 OPTION_F_DNS_FLAGS

5.5.4. OPTION_F_DNS_HOST_NAME

Contains the host name that was entered into DNS by the failover partner.

This is a DNS name encoded in [RFC1035] format as specified in Section 8 of [RFC3315].

The code for this option is TBD16.



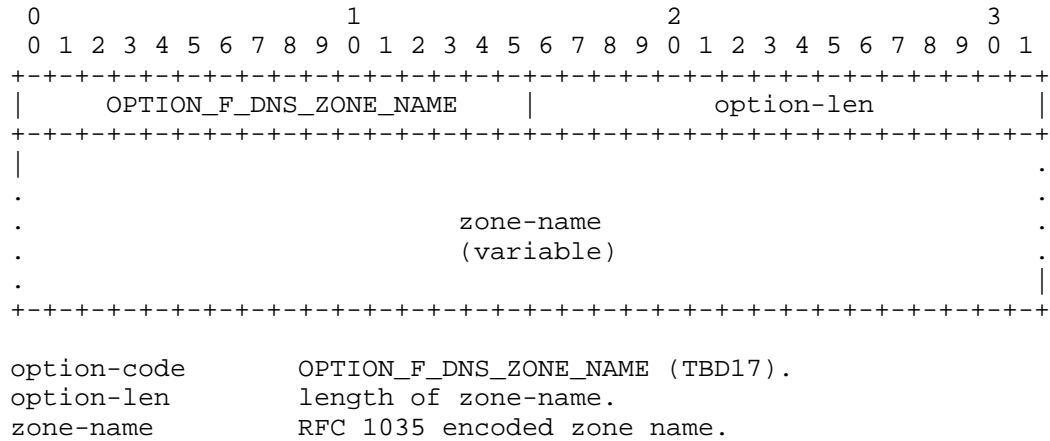
option-code OPTION_F_DNS_HOST_NAME (TBD16).
option-len length of host-name.
host-name RFC 1035 encoded host-name.

5.5.5. OPTION_F_DNS_ZONE_NAME

Contains the zone name that was entered into DNS by the failover partner.

This is a DNS name encoded in [RFC1035] format as specified in Section 8 of [RFC3315].

The code for this option is TBD17.

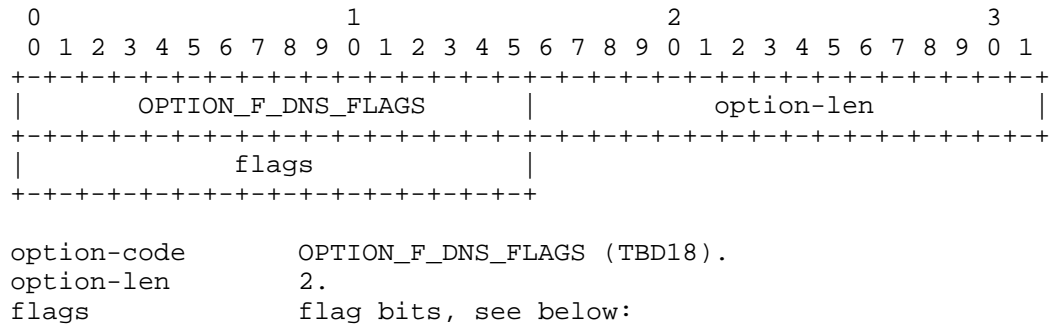


5.5.6. OPTION_F_DNS_FLAGS

Flags which indicate what needs to be done to remove this DNS name.

This consists of an unsigned 16 bit value in network byte order.

The code for this option is TBD18.



```

      0                               1
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+---+---+---+---+---+---+---+---+
|           MBZ           |U|S|R|F|
+---+---+---+---+---+---+---+---+

```

The bits (numbered from the most-significant bit in network byte-order) are used as follows:

- 0-11: MBZ
Must be zero
- 12 (U): USING_REQUESTED_FQDN
Set to 1 to indicate that name used came from the FQDN that was received from the client.
- 13 (S): SYNTHESIZED_NAME
Set to 1 to indicate that the name was synthesized based on some algorithm.
- 14 (R): REV_UPTODATE
Set to 1 to indicate that the reverse zone is up to date.
- 15 (F): FWD_UPTODATE
Set to 1 to indicate that the forward zone is up to date.

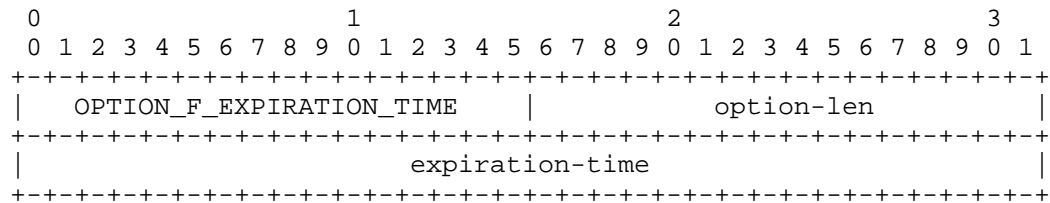
If both the U and S bits are unset, then the name must have been provided from some alternative configuration, such as client registration in some database accessible to the DHCP server.

5.5.7. OPTION_F_EXPIRATION_TIME

The greatest lifetime that this server has ever acked to its partner in a BNDREPLY for a particular lease or prefix. This MUST be an absolute time (i.e. seconds since midnight January 1, 2000 UTC, modulo 2^{32}).

This is an unsigned 32-bit integer in network byte order.

The code for this option is TBD19.



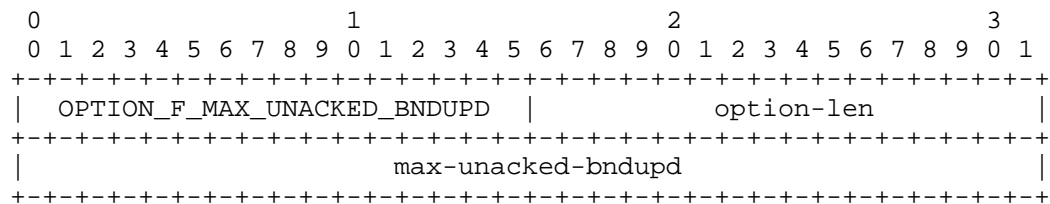
option-code OPTION_F_EXPIRATION_TIME (TBD19).
option-len 4.
expiration-time The expiration time. This MUST be an
 absolute time (i.e. seconds since midnight
 January 1, 2000 UTC, modulo 2^{32}).

5.5.8. OPTION_F_MAX_UNACKED_BNDUPD

The maximum number of BNDUPD messages that this server is prepared to accept over the TCP connection without causing the TCP connection to block.

This is an unsigned 32-bit integer in network byte order.

The code for this option is TBD20.



option-code OPTION_F_MAX_UNACKED_BNDUPD (TBD20).
option-len 4.
max-unacked-bndupd Maximum number of unacked BNDUPD message
 allowed.

5.5.9. OPTION_F_MCLT

The maximum-client-lead-time (MCLT) is the upper bound on the difference allowed between the valid lifetime provided to a DHCP client by a server and the valid lifetime known by that server's failover partner. It is an interval, measured in seconds. See Section 4.4.

This is an unsigned 32-bit integer in network byte order.

The code for this option is TBD21.

```

      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
+-----+-----+-----+-----+-----+-----+-----+-----+
|          OPTION_F_MCLT          |          option-len          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     mclt                         |
+-----+-----+-----+-----+-----+-----+-----+-----+

option-code      OPTION_F_MCLT (TBD21).
option-len       4.
mclt             The maximum-client-lead-time, in seconds.

```

5.5.10. OPTION_F_PARTNER_LIFETIME

The time after which the partner can consider an IPv6 address expired and is able to re-use the IPv6 address. This MUST be an absolute time (i.e. seconds since midnight January 1, 2000 UTC, modulo 2^{32}).

This is an unsigned 32-bit integer in network byte order.

The code for this option is TBD22.

```

      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
+-----+-----+-----+-----+-----+-----+-----+-----+
|  OPTION_F_PARTNER_LIFETIME  |          option-len          |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     partner-lifetime          |
+-----+-----+-----+-----+-----+-----+-----+-----+

option-code      OPTION_F_PARTNER_LIFETIME (TBD22).
option-len       4.
partner-lifetime The partner-lifetime. This MUST be an
                  absolute time (i.e. seconds since midnight
                  January 1, 2000 UTC, modulo  $2^{32}$ ).

```

5.5.11. OPTION_F_PARTNER_LIFETIME_SENT

The time that was received in an OPTION_F_PARTNER_LIFETIME Section 5.5.10 option. This is an exact duplicate (echo) of the time received in the OPTION_F_PARTNER_LIFETIME option, uncorrected and unadjusted in any way. This MUST be an absolute time (i.e. seconds since midnight January 1, 2000 UTC, modulo 2^{32}).

This is an unsigned 32-bit integer in network byte order.

The code for this option is TBD23.

```

      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
+-----+-----+-----+-----+-----+-----+-----+-----+
|OPTION_F_PARTNER_LIFETIME_SENT |               option-len         |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               partner-lifetime-sent                |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

option-code	OPTION_F_PARTNER_LIFETIME_SENT (TBD23).
option-len	4.
partner-lifetime-sent	The partner-lifetime received in an OPTION_F_PARTNER_LIFETIME option. This MUST be an absolute time (i.e. seconds since midnight January 1, 2000 UTC, modulo 2 ³²).

5.5.12. OPTION_F_PARTNER_DOWN_TIME

The time that the partner most recently lost communications with its failover partner. This MUST be an absolute time (i.e. seconds since midnight January 1, 2000 UTC, modulo 2³²).

This is an unsigned 32-bit integer in network byte order.

The code for this option is TBD24.

```

      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
+-----+-----+-----+-----+-----+-----+-----+-----+
|  OPTION_F_PARTNER_DOWN_TIME  |               option-len         |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                               partner-down-time                    |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

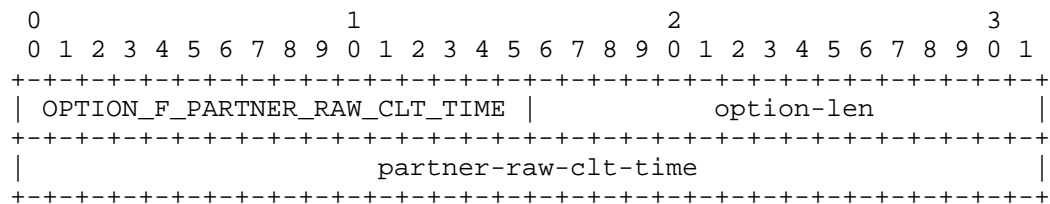
option-code	OPTION_F_PARTNER_DOWN_TIME (TBD24).
option-len	4.
partner-down-time	Contains the partner-down-time. This MUST be an absolute time (i.e. seconds since midnight January 1, 2000 UTC, modulo 2 ³²).

5.5.13. OPTION_F_PARTNER_RAW_CLT_TIME

The time when the partner most recently interacted with the DHCP client associated with this IPv6 address or prefix. This MUST be an absolute time (i.e. seconds since midnight January 1, 2000 UTC, modulo 2^{32}).

This is an unsigned 32-bit integer in network byte order.

The code for this option is TBD25.



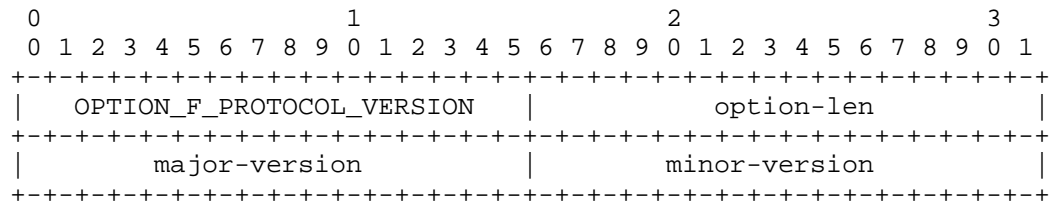
option-code	OPTION_F_PARTNER_RAW_CLT_TIME (TBD25).
option-len	4.
partner-raw-clt-time	Contains the partner-raw-clt-time. This MUST be an absolute time (i.e. seconds since midnight January 1, 2000 UTC, modulo 2^{32}).

5.5.14. OPTION_F_PROTOCOL_VERSION

The protocol version allows one failover partner to determine the version of the protocol being used by the other partner, to allow for changes and upgrades in the future. Two components are provided, to allow for large and small changes to be represented in one 32-bit number. The intent is that large changes would result in an increment of the major-version, while small changes would result in an increment of the minor-version. As subsequent updates and extensions of this document can define changes to these values in any way deemed appropriate no attempt is made to further define large and small in this document.

This consists of two unsigned 16-bit integers, in network byte order.

The code for this option is TBD26.



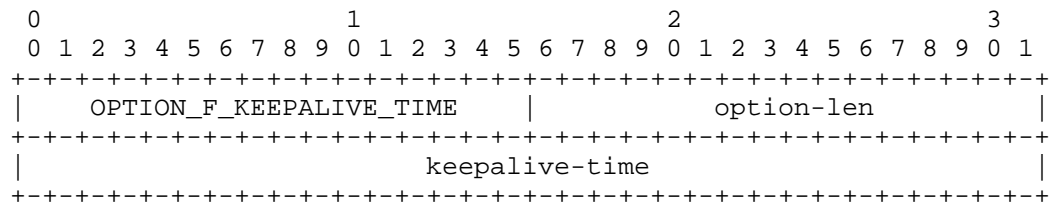
option-code OPTION_F_PROTOCOL_VERSION (TBD26).
option-len 4.
major-version The major version of the protocol. Initially 1.
minor-version The minor version of the protocol. Initially 0.

5.5.15. OPTION_F_KEEPA_LIVE_TIME

The number of seconds (an interval) within which the server must receive a message from its partner, or it will assume that communications from the partner is not ok.

This is an unsigned 32-bit integer in network byte order.

The code for this option is TBD27.

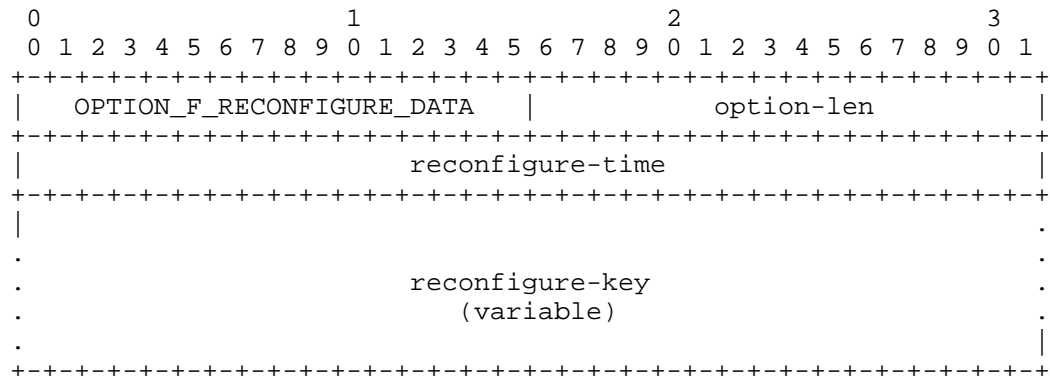


option-code OPTION_F_KEEPA_LIVE_TIME (TBD27).
option-len 4.
receive-time The keepalive-time. An interval of seconds.

5.5.16. OPTION_F_RECONFIGURE_DATA

Contains the information necessary for one failover partner to use the reconfigure-key created on the other failover partner.

The code for this option is TBD28.



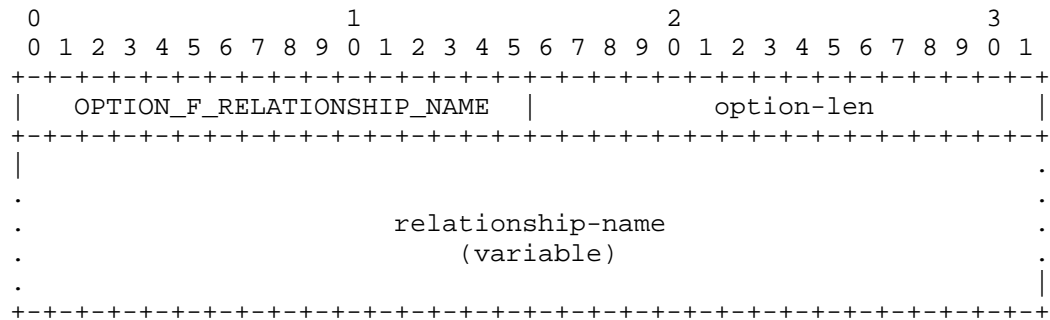
option-code OPTION_F_RECONFIGURE_DATA (TBD28).
option-len 4 + length of reconfigure-key.
reconfigure-time Time at which reconfigure-key was created.
 This MUST be an absolute time (i.e. seconds
 since midnight
 January 1, 2000 UTC, modulo 2³²).
reconfigure-key The reconfigure-key.

5.5.17. OPTION_F_RELATIONSHIP_NAME

A name for this failover relationship. Used to distinguish between relationships when there are multiple failover relationships between two failover servers.

A UTF-8 encoded text string suitable for display to an end user, which MUST NOT be null-terminated.

The code for this option is TBD29.



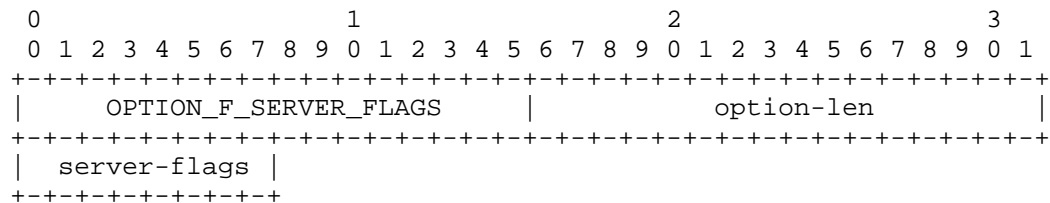
option-code OPTION_F_RELATIONSHIP_NAME (TBD29).
option-len length of relationship-name.
relationship-name A UTF-8 encoded text string suitable for
display to an end user, which MUST NOT be
null-terminated.

5.5.18. OPTION_F_SERVER_FLAGS

The OPTION_F_SERVER_FLAGS option specifies information associated with the failover endpoint sending the option.

This is an unsigned byte.

The code for this option is TBD30.



option-code OPTION_F_SERVER_FLAGS (TBD30).
option-len 1.
server-flags The server flags, see below:

```

  0 1 2 3 4 5 6 7
+-----+
|   MBZ   |A|S|C|
+-----+

```

The bits (numbered from the most-significant bit in network byte-order) are used as follows:

- 0-4 : MBZ
Must be zero
- 5 (A): ACK_STARTUP
Set to 1 to indicate that the OPTION_F_SERVER_FLAGS most recently received contained the STARTUP bit set.
- 6 (S): STARTUP,
MUST be set to 1 whenever the server is in STARTUP state.
- 7 (C): COMMUNICATED
Set to 1 to indicate that the sending server has communicated with its partner.

5.5.19. OPTION_F_SERVER_STATE

The OPTION_F_SERVER_STATE option specifies the endpoint state of the server sending the option.

This is an unsigned byte.

The code for this option is TBD31.

```

      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
+-----+-----+-----+-----+
|   OPTION_F_SERVER_STATE   |   option-len   |
+-----+-----+-----+-----+
| server-state |
+-----+

```

option-code	OPTION_F_SERVER_STATE (TBD31).
option-len	1.
server-state	Failover endpoint state.

Value	Server State	
-----	-----	-----
0	reserved	
1	STARTUP	Startup state (1)
2	NORMAL	Normal state
3	COMMUNICATIONS-INTERRUPTED	Communication interrupted
4	PARTNER-DOWN	Partner down
5	POTENTIAL-CONFLICT	Synchronizing
6	RECOVER	Recovering bindings from partner
7	SHUTDOWN	Shutting down for a long period.
8	RECOVER-DONE	Interlock state prior to NORMAL
9	RESOLUTION-INTERRUPTED	Comm. failed during resolution
10	CONFLICT-DONE	Primary resolved its conflicts

These states are discussed in detail in Section 8.

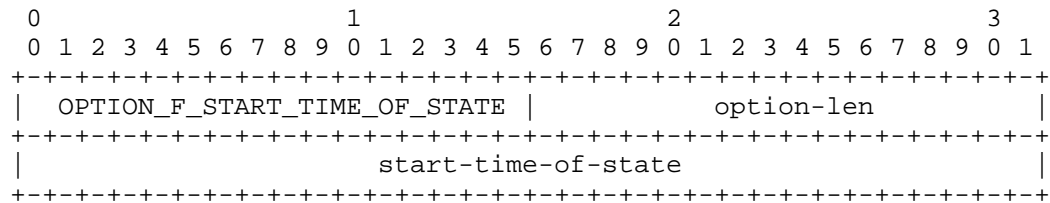
(1) The STARTUP state is never sent to the partner server, it is indicated by the STARTUP bit in the server-flags options (see Section 8.3).

5.5.20. OPTION_F_START_TIME_OF_STATE

The time at which the associated state began to hold its current value. When this option appears in a STATE message, the state to which it refers is the server endpoint state. When it appears in an IA_NA-options, IA_TA-options, or IA_PD-options field, the state to which it refers is the binding-status value in the OPTION_IA_NA, OPTION_IA_TA, or OPTION_IA_PD option, respectively. This MUST be an absolute time (i.e. seconds since midnight January 1, 2000 UTC, modulo 2^{32}).

This is an unsigned 32-bit integer in network byte order.

The code for this option is TBD32.



option-code OPTION_F_START_TIME_OF_STATE (TBD32).
option-len 4.
start-time-of-state The start-time-of-state. This MUST be an
absolute time (i.e. seconds since midnight
January 1, 2000 UTC, modulo 2^{32}).

5.5.21. OPTION_F_STATE_EXPIRATION_TIME

The state-expiration-time is the time at which the current state of this lease will expire. This MUST be an absolute time (i.e. seconds since midnight January 1, 2000 UTC, modulo 2^{32}).

Note that states other than ACTIVE may have a time associated with them. In particular, EXPIRED might have a time associated with it, in the event that some sort of "grace period" existed where the lease would not be reused for a period after the lease expired. The ABANDONED state might have a time associated with it, in the event that the servers participating in failover had a time after which an ABANDONED lease might be placed back into a pool for allocation to a client. In general, if there is an OPTION_STATE_EXPIRATION_TIME associated with a particular state, that indicates the associated state will expire and move to a different state at that time.

This is an unsigned 32-bit integer in network byte order.

The code for this option is TBD33.

```

      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
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| OPTION_F_STATE_EXPIRATION_TIME |                   option-len |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|                               state-expiration-time             |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

```

option-code          OPTION_F_STATE_EXPIRATION_TIME (TBD33).
option-len           4.
state-expiration-time The state-expiration-time. This MUST be an
                      absolute time (i.e. seconds since midnight
                      January 1, 2000 UTC, modulo 2^32).

```

5.6. Status Codes

The following new status codes are defined, to be used in the OPTION_STATUS_CODE option.

AddressInUse (TBD34)

One client on one server has leases that are in conflict with the leases that the client has on another server. Alternatively, the address could be associated with a different IAID on each server.

ConfigurationConflict (TBD35)

The configuration implied by the information in a BNDUPD (e.g. the IPV6 address or prefix address) is in direct conflict with the information known to the receiving server.

MissingBindingInformation (TBD36)

There is insufficient information in a BNDUPD to effectively process it.

OutdatedBindingInformation (TBD37)

Returned when the information in a server's binding database conflicts with the information found in an incoming BNDUPD, and the server believes that the information in its binding database more accurately reflects reality.

ServerShuttingDown (TBD38)

Returned when the server is undergoing an operator directed or otherwise planned shutdown.

DNSUpdateNotSupported (TBD39)

Returned when a server receives a BNDUPD with DNS update information included, and the server doesn't support DNS update.

ExcessiveTimeSkew (TBD40)

Returned when a server detects that the time skew between its current time and its partner's current time is greater than 5 seconds.

6. Connection Management

Communication between failover partners takes place over a long-lived TCP connection. This connection is always initiated by the primary server, and if the long-lived connection is lost it is the responsibility of the primary server to attempt to reconnect to the secondary server. The detailed process used by the primary server when initiating a connection and by the secondary server when responding to a connection attempt documented in Section 6.1 is followed each time a connection is established, regardless of any previous connection between the failover partners.

6.1. Creating Connections

Every primary server implementing the failover protocol **MUST** periodically attempt to create a TCP connection to the dhcp-failover port (647) of all of its configured partners, where the period is implementation dependent and **SHOULD** be configurable. In the event that a connection has been rejected by a **CONNECTREPLY** 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 **MUST** continue to attempt to connect periodically.

Every secondary server implementing the failover protocol **MUST** listen for TCP connection attempts on the dhcp-failover port (647) from a primary server.

After a primary server successfully establishes a TCP connection to a secondary server, it **MUST** continue the connection process as described in Section 8.2 of [RFC7653]. In the language of that section, the primary failover server operates as the "requestor" and the secondary failover server operates as the "DHCP server". The message that is sent over the newly established connection is a **CONNECT** message, instead of an **ACTIVELEASEQUERY** message.

When a connection attempt is received by a secondary server, the only information that the secondary server has is the IP address of the partner initiating a connection. If it has any relationships with the connecting server for which it is a secondary server, it should operate as described in Section 9.1 of [RFC7653], with the exception that instead of waiting for an Active Leasequery message it will wait for a **CONNECT** message. Once it has received the **CONNECT** message, it

will use the information in that message to determine which relationship this connection is to service.

If it has no secondary relationships with the connecting server, it MUST 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 MUST listen for CONNECT messages from the primary server.

When the CONNECT and CONNECTREPLY exchange successfully produces a working failover connection, the next message sent over a new connection is a STATE message. See Section 6.3. Upon the receipt of the STATE message, the receiver can consider communications ok.

6.1.1. Sending a CONNECT message

The CONNECT message is sent with information about the failover configuration on the primary server. The message MUST contain at least the following information in the options area:

- o OPTION_F_PROTOCOL_VERSION containing the protocol version that the primary server will use when sending failover messages.
- o OPTION_F_MCLT containing the configured MCLT.
- o OPTION_F_KEEPALIVE_TIME containing the number of seconds (an interval) within which the server must receive a message from its partner, or it will assume that communications from the partner is not ok.
- o OPTION_F_MAX_UNACKED_BNDUPD containing the maximum number of BNDUPD messages that this server is prepared to accept over the failover connection without causing the connection to block. This is to implement application level flow control over the connection, so that a flood of BNDUPD messages does not cause the connection to block and thereby prevent other messages from being transmitted over the connection and received by the failover partner.
- o OPTION_F_RELATIONSHIP_NAME containing name of the failover relationship to which this connection applies. If there is no OPTION_F_RELATIONSHIP_NAME in the CONNECT message, it indicates that there is only a single relationship between this pair of primary and secondary servers.

- o `OPTION_F_CONNECT_FLAGS` containing information about certain attributes of the connecting servers.

6.1.2. Receiving a `CONNECT` message

A server receiving a `CONNECT` message must process the information in the message and decide whether or not to accept the connection. The processing is performed as follows:

- o `sent-time` - The secondary server checks the `sent-time` to see if it is within 5 seconds of its current time. See Section 7.1. If it is not, return `ExcessiveTimeSkew` in the `OPTION_STATUS_CODE` to reject the `CONNECT` message.
- o `OPTION_F_PROTOCOL_VERSION` - The secondary server decides if the protocol version of the primary server is supported by the secondary server. If it is not, return `NotSupported` in the `OPTION_STATUS_CODE` to reject the `CONNECT` message.
- o `OPTION_F_MCLT` - Use this MCLT supplied by the primary server. Remember this MCLT and use it until a different MCLT is supplied by some subsequent `CONNECT` message.
- o `OPTION_F_KEEPA_LIVE_TIME` - Remember the keepalive-time as the `FO_KEEPA_LIVE_TIME` when implementing the Unreachability Detection algorithm described in Section 6.6.
- o `OPTION_F_MAX_UNACKED_BNDUPD` - Ensure that the maximum amount of unacked `BNDUPD` messages queued to the primary server never exceeds the value in the `OPTION_F_MAX_UNACKED_BNDUPD` option.
- o `OPTION_F_CONNECT_FLAGS` - Ensure that the secondary can process information from the primary as specified in the flags. For example, if the secondary server cannot process prefix delegation with variable sized prefixes delegated from the same delegable prefix, and the primary server says that it can, the secondary should reject the connection.

A `CONNECT` message SHOULD always be followed by a `CONNECTREPLY` message, either to accept the connection or to reject the connection by including an `OPTION_STATUS_CODE` option with an error reject. In order to reject the connection attempt, simply send a `CONNECTREPLY` message with the `OPTION_STATUS_CODE` with the correct status. If accepting the connection attempt, then send a `CONNECTREPLY` message with the following information:

- o `OPTION_F_PROTOCOL_VERSION` containing the protocol version being used by the secondary server when sending failover messages.

- o OPTION_F_MCLT containing the MCLT currently in use on the secondary server. This MUST equal the MCLT that was in the OPTION_F_MCLT option in the CONNECT.
- o OPTION_F_KEEPALIVE_TIME containing the number of seconds (an interval) within which the server must receive a message from its partner, or it will assume that communications from the partner is not ok.
- o OPTION_F_MAX_UNACKED_BNDUPD containing the maximum number of BNDUPD messages that this server is prepared to accept over the failover connection without causing the connection to block. This is to implement application level flow control over the connection, so that a flood of BNDUPD messages does not cause the connection to block and thereby prevent other messages from being transmitted over the connection and received by the failover partner.
- o OPTION_F_CONNECT_FLAGS - Place information into this option to describe the attributes of the secondary server that the primary needs to know about.

After sending a CONNECTREPLY message to accept the primary server's CONNECT message, the secondary server MUST send a STATE message (see Section 6.3).

6.1.3. Receiving a CONNECTREPLY message

A server receiving a CONNECTREPLY message must process the information in the message and decide whether or not to continue to employ the connection. The processing is performed as follows:

- o OPTION_F_PROTOCOL_VERSION - The primary server decides if the protocol version in use by the secondary server is supported by the primary server. If it is not, send a DISCONNECT message and drop the connection. If it is supported, continue processing. It is possible that the primary and secondary server will each be sending different versions of the protocol to the other server. The extent to which this is supported will be in part defined by as yet unknown differences in the protocols that the versions represent, and in part by the capabilities of the two implementations involved in the failover relationship.
- o OPTION_F_MCLT - Compare the MCLT received with the configured MCLT, and if they are different send a DISCONNECT message and drop the connection.

- o `OPTION_F_KEEPA_LIVE_TIME` - Remember the keepalive-time as the `FO_KEEPA_LIVE_TIME` when implementing the Unreachability Detection algorithm described in Section 6.6.
- o `OPTION_F_MAX_UNACKED_BNDUPD` - Ensure that the maximum amount of unacked BNDUPD messages queued to the secondary server never exceeds the value in the `OPTION_F_MAX_UNACKED_BNDUPD` option.
- o `OPTION_F_CONNECT_FLAGS` - Ensure that the primary can process information from the secondary as specified in the flags. For example, if the primary server cannot process prefix delegation with variable sized prefixes delegated from the same delegable prefix, and the secondary server says that it can, the primary should drop the connection.

After receiving a `CONNECTREPLY` message that accepted the primary server's `CONNECT` message, the primary server **MUST** send a `STATE` message (see Section 6.3).

6.2. Endpoint Identification

A failover endpoint is always associated with a set of DHCP prefixes that are configured on the DHCP server where the endpoint appears. A DHCP 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.

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 endpoints 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. On a particular server, any given prefix that participates in failover 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 IPv6 address of the partner, the relationship-name, and the role of the receiving server.

6.3. Sending a STATE message

A server **MUST** send a STATE message to its failover partner whenever the state of the failover endpoint changes. Sending the occasional duplicate STATE message will cause no problems, and not updating the failover partner with information about a failover endpoint state change can, in many cases, cause the entire failover protocol to be inoperative.

The STATE message is sent with information about the endpoint state of the failover relationship. The STATE message **MUST** contain at least the following information in the options area:

- o OPTION_F_SERVER_STATE containing the state of this failover endpoint.
- o OPTION_F_SERVER_FLAGS containing the flag values associated with this failover endpoint.
- o OPTION_F_START_TIME_OF_STATE containing the time when this became the state of this failover endpoint.
- o OPTION_F_PARTNER_DOWN_TIME containing time that this failover endpoint went into PARTNER-DOWN state if this server is in PARTNER-DOWN state. If this server isn't in PARTNER-DOWN state, do not include this option.

The server sending a STATE message **SHOULD** ensure that this information is written to stable storage prior to enqueueing it to its failover partner.

6.4. Receiving a STATE message

A server receiving a STATE message must process the information in the message and decide how to react to the information. The processing is performed as follows:

- o OPTION_F_SERVER_STATE - If this represents a change in state for the failover partner, react according to the direction in Section 8.1. If the state is not PARTNER-DOWN, clear any memory of the partner-down-time.

- o `OPTION_F_SERVER_FLAGS` - Remember these flags in an appropriate data area so they can be referenced by code implementing other parts of this document.
- o `OPTION_F_START_TIME_OF_STATE` - Remember this information in an appropriate data area.
- o `OPTION_F_PARTNER_DOWN_TIME` - Remember this information in an appropriate data area if the value of the `OPTION_F_SERVER_STATE` is `PARTNER-DOWN`.

A server receiving a `STATE` message SHOULD ensure that this information is written to stable storage.

6.5. Connection Maintenance Parameters

The following parameters and timers are used to ensure the integrity of the connections between two failover servers.

Parameter	Default	Description

<code>FO_KEEPALIVE_TIMER</code>	timer	counts down to time connection assumed dead due to lack of messages
<code>FO_KEEPALIVE_TIME</code>	60	maximum time server will consider connection still up with no messages
<code>FO_CONTACT_PER_KEEPALIVE_TIME</code>	4	number of <code>CONTACT</code> messages to send during partner's <code>FO_KEEPALIVE_TIME</code> period
<code>FO_SEND_TIMER</code>	timer	counts down to time to send next <code>CONTACT</code> message
<code>FO_SEND_TIME</code>	15	maximum time to wait between sending <code>CONTACT</code> messages if no other traffic Created from partner's <code>FO_KEEPALIVE_TIME</code> divided by <code>FO_CONTACT_PER_KEEPALIVE_TIME</code>

6.6. Unreachability detection

Each partner MUST maintain an `FO_SEND_TIMER` for each failover connection. The `FO_SEND_TIMER` for a particular connection is reset to `FO_SEND_TIME` every time any message is transmitted on that connection, and counts down once per second. If the timer reaches zero, a `CONTACT` message is transmitted on that connection and the timer for that connection is reset to `FO_SEND_TIME`. The `CONTACT`

message may be transmitted at any time. An implementation MAY use additional mechanisms to detect partner unreachability.

The `FO_SEND_TIME` is initialized from the configured `FO_KEEPALIVE_TIME` divided by `FO_CONTACT_PER_KEEPALIVE_TIME`. When a `CONNECT` or `CONNECTREPLY` message is received on a connection, the received `OPTION_F_KEEPALIVE_TIME` option is checked, and the value in that option is used to calculate the `FO_SEND_TIME` for that connection by dividing the value received by the configured `FO_CONTACT_PER_KEEPALIVE_TIME`.

Each partner MUST maintain an `FO_KEEPALIVE_TIMER` for each failover connection. This timer is initialized to `FO_KEEPALIVE_TIME` and counts down once per second. It is reset to `FO_KEEPALIVE_TIME` whenever a message is received on that connection. If it ever reaches zero, that connection is considered dead. In addition, the `FO_KEEPALIVE_TIME` for that connection MUST be sent to the failover partner on every `CONNECT` or `CONNECTREPLY` messages, in the `OPTION_F_KEEPALIVE_TIME` option.

7. Binding Updates and Acks

7.1. 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. The failover protocol adopts the simple approach of requiring that the failover partners use some mechanism to synchronize the clocks on the two servers to within an accuracy of roughly 5 seconds.

A mechanism to measure and track relative time differences between servers is necessary to ensure this synchronization. To do so, each message contains the time of the transmission in the time context of the transmitter in the sent-time field of the message (see Section 5.2). The transmitting server MUST set this as close to the actual transmission as possible. The receiving partner MUST store its own timestamp of reception as close to the actual reception as possible. The received timestamp information is then compared with local timestamp.

7.2. Information model

In most DHCP servers a lease on an IPv6 address or a prefix can take on several different binding-status values, sometimes also called lease states. While no two DHCP server implementations will have exactly the same possible binding-status values, [RFC3315] enforces

some commonality among the general semantics of the binding-status values used by various DHCP server implementations.

In order to transmit binding database updates between one server and another using the failover protocol, some common binding-status values must be defined. It is not expected that these values correspond with any actual implementation of the DHCPv6 protocol in a DHCP server, but rather that the binding-status values defined in this document should be convertible back and forth between those defined below and those in use by many DHCP server implementations.

The lease binding-status values defined for the failover protocol are listed below. Unless otherwise noted below, there MAY be client information associated with each of these binding-status value.

ACTIVE -- The lease is assigned to a client. Client identification data MUST appear.

EXPIRED -- indicates that a client's binding on a given lease has expired. When the partner acks the BNDUPD of an expired lease, the server sets its internal state to PENDING-FREE. Client identification SHOULD appear.

RELEASED -- indicates that a client sent a RELEASE message. When the partner acks the BNDUPD of a released lease, the server sets its internal state to PENDING-FREE. Client identification SHOULD appear.

PENDING-FREE -- Once a lease is expired or released, its state becomes PENDING-FREE. Depending on which algorithm and which pool was used to allocate a given lease, PENDING-FREE may either mean FREE or FREE-BACKUP. Implementations do not have to implement this PENDING-FREE state, but may choose to switch to the destination state directly. For clarity of representation, this transitional PENDING-FREE state is treated as a separate state.

FREE -- Is used when a DHCP server needs to communicate that a lease is unused by any client, but it was not just released, expired or reset by a network administrator. When the partner acks the BNDUPD of a FREE lease, the server marks the lease as available for assignment by the primary server. Note that on a secondary server running in PARTNER-DOWN state, after waiting the MCLT, the lease MAY be allocated to a client by the secondary server. Client identification MAY appear and indicates the last client to have used this lease as a hint.

FREE-BACKUP -- indicates that this lease can be allocated by the secondary server to a client at any time. Note that on the

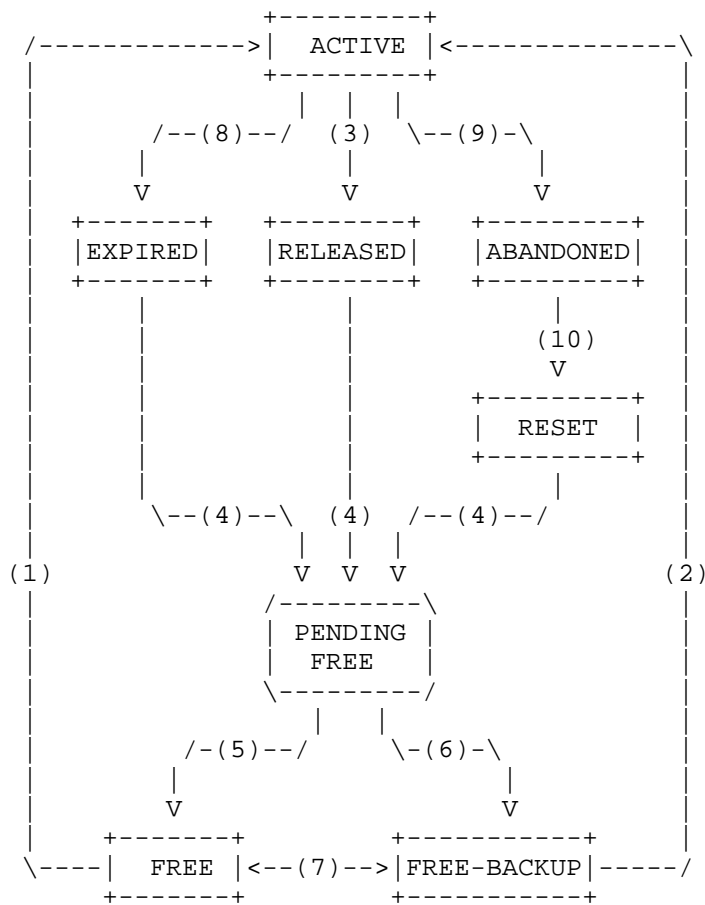
primary server running in PARTNER-DOWN state, after waiting the MCLT, the lease MAY be allocated to a client by the primary server if proportional algorithm was used. Client identification MAY appear and indicates the last client to have used this lease as a hint.

ABANDONED -- indicates that a lease is considered unusable by the DHCP system. The primary reason for entering such state is reception of DECLINE message for the lease. Client identification MAY appear.

RESET -- indicates that this lease was made available by operator command. This is a distinct state so that the reason that the lease became FREE can be determined. Client identification MAY appear.

Which binding-status values are associated with a timeout is implementation dependent. Some binding-status values such as ACTIVE will have a timeout value in all implementations, while others such as ABANDONED will have a timeout value in some implementations and not in others. In some implementations a binding-status value may be associated with a timeout in some circumstances and not in other circumstances. The receipt of a BNDUPD with a particular binding-status value and an OPTION_F_STATE_EXPIRATION_TIME indicates that this particular binding-status value is associated with a timeout.

The lease state machine is presented in Figure 2. Most states are stationary, i.e. the lease stays in a given state until external event triggers transition to another state. The only transitive state is PENDING-FREE. Once it is reached, the state machine immediately transitions to either FREE or FREE-BACKUP state.



PENDING-FREE transition

Figure 2: Lease State Machine

Transitions between states are results of the following events:

1. Primary server allocates a lease.
2. Secondary server allocates a lease.
3. Client sends RELEASE and the lease is released.
4. Partner acknowledges state change. This transition MAY also occur if the server is in PARTNER-DOWN state and the MCLT has passed since the entry into RELEASED, EXPIRED, or RESET states.

5. The lease belongs to a pool that is governed by the proportional allocation, or independent allocation is used and this lease belongs to primary server pool.
6. The lease belongs to a pool that is governed by the independent allocation and the lease belongs to the secondary server.
7. Pool rebalance event occurs (POOLREQ/POOLRESP messages are exchanged). Delegable prefixes belonging to the primary server can be assigned to the secondary server pool (transition from FREE to FREE-BACKUP) or vice versa.
8. The lease has expired.
9. DECLINE message is received or a lease is deemed unusable for other reasons.
10. An administrative action is taken to recover an abandoned lease back to usable state. This transition MAY occur due to an implementation specific handling on ABANDONED lease. One possible example of such use is a Neighbor Discovery or ICMPv6 Echo check if the address is still in use.

The lease that is no longer in use (due to expiration or release), becomes PENDING-FREE. Depending on what allocation algorithm is used, the lease that is no longer in use, returns to the primary (FREE) or secondary pool (FREE-BACKUP). The conditions for specific transitions are depicted in Figure 3.

Lease owner		
Algorithm	Primary	Secondary
Proportional	FREE	FREE-BACKUP
Independent	FREE	FREE

Figure 3: PENDING-FREE State Transitions

7.3. Times Required for Exchanging Binding Updates

Each server must keep track of the following specific times beyond those required by the base DHCP protocol [RFC3315].

expiration-time

The greatest lifetime that this server has ever acked to its failover partner in a BNDREPLY.

acked-partner-lifetime

The greatest lifetime that the failover partner has ever acked to this server in a BNDREPLY.

partner-lifetime

The time that we will send (or have sent) the partner, which will be the time after which the partner can consider the lease expired. When we receive a BNDUPD this value can be updated from the received OPTION_F_EXPIRATION_TIME.

client-last-transaction-time

The time when this server most recently interacted with the client associated with this lease.

partner-raw-clt-time

The time when the partner most recently interacted with the client associated with this lease. This time remains exactly as it was received by this server, and MUST NOT be adjusted to be in the time context of this server.

start-time-of-state

The time when the binding-status of this lease was changed to its current value.

state-expiration-time

The time when the current state of this lease will expire.

7.4. Sending Binding Updates

Every BNDUPD message contains information about either a single client binding in an OPTION_CLIENT_DATA option that include IAADDR or IAPREFIX options associated with that client, or a single prefix lease in an OPTION_IAPREFIX option for prefixes that are currently not associated with any clients.

All information about a particular client binding MUST be contained in a single OPTION_CLIENT_DATA option (see Section 4.1.2.2 of [RFC5007]). The OPTION_CLIENT_DATA option contains at least the data shown below in its client-options section:

- o OPTION_CLIENTID containing the DUID of the client most recently associated with this lease MUST appear;

- o OPTION_LQ_BASE_TIME containing the absolute time that the information was placed into this OPTION_CLIENT_DATA option (see Section 6.3.1 of [RFC7653]) MUST appear;
- o OPTION_VSS (see Section 3.4 of [RFC6607]) This option MUST NOT appear if the information in this OPTION_CLIENT_DATA option is associated with the global, default VPN. This option MUST appear if the information in this OPTION_CLIENT_DATA option is associated with a VPN other than the global, default VPN. Support of [RFC6607] is not required, and OPTION_VSS is only used if a VPN other than the global, default VPN is used, which requires support of [RFC6607];
- o OPTION_F_RECONFIGURE_DATA containing the time and reconfigure key, if any;
- o OPTION_LQ_RELAY_DATA containing information described in Section 4.1.2.4 of [RFC5007], if any exists;
- o OPTION_IA_NA or OPTION_IA_TA for an IPv6 Address or OPTION_IA_PD for an IPv6 Prefix. More than one of either of these options MAY appear if there are more than one associated with this client. At least one MUST appear;
 - * IAID - Identity Association used by the client, while obtaining a given lease. (Note1: one client may use many IAIDs simultaneously. Note2: IAID for IA, TA and PD are orthogonal number spaces.);
 - * T1 time sent to client;
 - * T2 time sent to client;
 - * Inside of the IA_NA-options, IA_TA-options, or IA_PD-option sections:
 - + OPTION_IAADDR for an IPv6 address or an OPTION_IAPREFIX for a IPv6 prefix MUST appear;
 - IPv6 Address or IPv6 Prefix (with length);
 - preferred lifetime sent to client;
 - valid lifetime sent to client;
 - Inside of the IAaddr-options or IAprefix-options:

- o OPTION_F_BINDING_STATUS containing the binding-status MUST appear;
- o OPTION_F_START_TIME_OF_STATE containing the start-time-of-state MUST appear;
- o OPTION_F_STATE_EXPIRATION_TIME (absolute) containing the state-expiration-time*;
- o OPTION_CLT_TIME (relative) containing the client-last-transaction-time. See [RFC5007] for this option;
- o OPTION_F_PARTNER_LIFETIME (absolute) containing partner-lifetime*;
- o OPTION_F_PARTNER_RAW_CLT_TIME (absolute) containing the partner-raw-clt-time;
- o OPTION_F_EXPIRATION_TIME (absolute) containing the expiration-time*;
- o DHCP_O_CLIENT_FQDN containing the FQDN information associated with this lease and client, if any;

Information about a prefix lease is contained in a single OPTION_IAPREFIX option. Only a single OPTION_IAPREFIX option may appear in a BNDUPD message outside of an OPTION_CLIENT_DATA option. In detail:

- o OPTION_IAPREFIX for a prefix lease;
 - * IPv6 Prefix (with length);
 - * Inside of the IAprefix-options section:
 - + OPTION_VSS (see Section 3.4 of [RFC6607]) This option MUST NOT appear if the information in this OPTION_IAPREFIX option is associated with the global, default VPN. This option MUST appear if the information in this OPTION_IAPREFIX option is associated with a VPN other than the global, default VPN. Support of [RFC6607] is not required, and OPTION_VSS is only used if a VPN other than the global, default VPN is used, which requires support of [RFC6607];
 - + OPTION_LQ_BASE_TIME containing the absolute time that this information was placed into this OPTIONS_IAPREFIX option (see Section 6.3.1 of [RFC7653]) MUST appear;

- + OPTION_F_BINDING_STATUS containing the binding-status MUST appear;
- + OPTION_F_START_TIME_OF_STATE containing the start-time-of-state MUST appear;
- + OPTION_F_STATE_EXPIRATION_TIME (absolute) containing the state-expiration-time*;
- + OPTION_F_PARTNER_LIFETIME (absolute) containing partner-lifetime*;
- + OPTION_F_EXPIRATION_TIME (absolute) containing the expiration-time*;

Items marked with a single asterisk (*) MUST appear only if the value in the OPTION_F_BINDING_STATUS is associated with a timeout, otherwise it MUST NOT appear. See Section 7.2 for details.

The OPTION_CLT_TIME MUST, if it appears, be the time that the server last interacted with the DHCP client. It MUST NOT be, for instance, the time that the lease expired if there has been no interaction with the DHCP client in question.

A server SHOULD be prepared to clean up DNS information once the lease expires or is released. See Section 9 for a detailed discussion about DNS update. Another reason the partner may be interested in keeping additional data is to enable better support for Leasequery [RFC5007], Bulk Leasequery [RFC5460] or Active Leasequery [RFC7653], some of which feature queries based on Relay-ID, by link address and by Remote-ID.

7.5. Receiving Binding Updates

7.5.1. Monitoring Time Skew

The sent-time from the Failover message is compared with the current time of the receiving server as recorded when it received the message. The difference is noted, and if it is greater than 5 seconds the receiving server SHOULD drop the connection. A message SHOULD be logged to signal the reason for the connection being dropped.

Any time can be before, after, or essentially the same as another time. Any time which ends up being +/- 5 seconds of another time SHOULD be considered to be representing the same time when performing a comparison between two times.

7.5.2. Acknowledging Reception

Upon acceptance of a binding update, the server MUST notify its partner that it has processed the binding update (and updated its lease state database if necessary) by sending a BNDREPLY. A server MUST NOT send the BNDREPLY before its binding database is updated.

7.5.3. Processing Binding Updates

When a BNDUPD is received it MUST contain either a single OPTION_CLIENT_DATA option or a single OPTION_IAPREFIX option.

When analyzing an BNDUPD message option from a partner server, if there is insufficient information in the BNDUPD message to process it, then it is rejected with an OPTION_STATUS_CODE of "MissingBindingInformation".

The server receiving a BNDUPD update from its partner must evaluate the received information in each OPTION_CLIENT_DATA or IAPREFIX option to see if it is consistent with the server's already known state, and if it is not, decide to accept or reject the information. Section 7.5.4 provides the details how the server makes this determination.

A server receiving a BNDUPD message MUST respond to the sender of that message with a BNDREPLY message which contains the same transaction-id as the BNDUPD message. This BNDREPLY message MUST contain either a single OPTION_CLIENT_DATA option or a single OPTION_IAPREFIX option, corresponding to whatever was received in the BNDUPD message.

An OPTION_CLIENT_DATA option or an OPTION_IAPREFIX option in the BNDREPLY which is accepted SHOULD NOT contain an OPTION_STATUS_CODE unless a status message needs to be sent to the failover partner, in which case it SHOULD include an OPTION_STATUS_CODE option with a status code indicating success and whatever message is needed.

To indicate rejection of the information in an OPTION_CLIENT_DATA option, or an OPTION_IAPREFIX option, an OPTION_STATUS_CODE SHOULD be included with a status code indicating an error in the OPTION_CLIENT_DATA option or OPTION_IAPREFIX option in the BNDREPLY message.

7.5.4. Accept or Reject?

The first task in processing the information in an OPTION_CLIENT_DATA option or OPTION_IAPREFIX option is extract the client information (if any) and lease information out of the option, and to access the

address lease or prefix lease information in the server's binding database.

If an `OPTION_VSS` option is specified in the `OPTION_CLIENT_DATA` option or `OPTION_IAPREFIX` option, if the VPN specified in the `OPTION_VSS` option does not appear in the configuration of the receiving server, reject the entire `OPTION_CLIENT_DATA` option or `OPTION_IAPREFIX` option with the reject-reason "ConfigurationConflict".

If the lease specified in the `OPTION_CLIENT_DATA` option or `OPTION_IAPREFIX` option is not a lease associated with the failover endpoint which received the `OPTION_CLIENT_DATA` option, then reject it with reject-reason "ConfigurationConflict".

In general, acceptance or rejection is based around the comparison of two different time values, one from the `OPTION_CLIENT_DATA` option or `OPTION_IAPREFIX` option in the `BNDUPD` message, and one from the receiving server's binding database associated with the address or prefix lease found in the `BNDUPD` message. The time for the `BNDUPD` message where the `OPTION_F_BINDING_STATUS` is `ACTIVE`, `EXPIRED`, or `RELEASED` is the `OPTION_CLT_TIME` if one appears, and the `OPTION_F_START_TIME_OF_STATE` if one does not. For other binding-status values, the time for the `BNDUPD` message is the later of the `OPTION_CLT_TIME` if one appears, and the `OPTION_F_START_TIME_OF_STATE`. The time for the lease in the server's binding database is the `client-last-transaction-time`, if one appears, and the `start-time-of-state` if one does not.

The basic approach is to compare these times, and if the one from the `BNDUPD` message is clearly later, then accept the information in the `OPTION_CLIENT_DATA` option or `OPTION_IAPREFIX` option. If the one from the server's binding database is clearly later, then reject the information in the `BNDUPD` message. The challenge comes when they are essentially the same (i.e., +/- 5 seconds). In this case they are considered identical, despite the minor differences. The table below (Figure 4) contains the rules for dealing with all of these situations.

		binding-status in received OPTION_CLIENT_DATA or OPTION_IAPREFIX			
binding-status in receiving server's lease state DB		ACTIVE	EXPIRED	RELEASED	
					FREE RESET FREE-BACKUP ABANDONED
ACTIVE		accept(3)	time(1)	accept	time(1)
EXPIRED		accept	accept	accept	accept
RELEASED		accept	accept	accept	accept
FREE/FREE-BACKUP		accept	accept	accept	accept
RESET		time(2)	accept	accept	accept
ABANDONED		accept	accept	accept	accept

Figure 4: Conflict Resolution

accept: If the time value in the OPTION_CLIENT_DATA option or OPTION_IAPREFIX option is later than the time value in the server's binding database, accept it, else reject it.

time(1): If the current time is later than the receiving server's state-expiration-time, accept it, else reject it.

time(2): If the OPTION_CLT_TIME value (if it appears) in the OPTION_CLIENT_DATA is later than the start-time-of-state in the receiving server's binding, accept it, else reject it.

accept,time(1),time(2): If rejecting, use reject reason "OutdatedBindingInformation".

accept(3): If the client in an OPTION_CLIENT_DATA option and in a receiving server's binding differ, then if time(2) or the receiving server is a secondary accept it, else reject it with a reject reason of "AddressInUse". If the clients match, accept the update.

The lease update may be accepted or rejected. If a lease is rejected with "OutdatedBindingInformation", then the flag in the lease that indicates the partner should be updated about the information in this lease SHOULD be set, otherwise it SHOULD NOT be changed. If this flag was previously not set, then an update MAY be transmitted immediately to the partner (though the BNDREPLY to this BNDUPD SHOULD be sent first). If this flag was previously set an update SHOULD NOT be transmitted immediately to the partner. In this case, an update will be sent during the next periodic scan, but not immediately, thus preventing a possible update storm should the servers be unable to agree. Ultimately, the server with the most recent binding information should have its update accepted by its partner.

7.5.5. Accepting Updates

When the information in an `OPTION_CLIENT_DATA` option or `OPTION_IAPREFIX` option has been accepted, some of that information is stored in the receiving server's binding database, and corresponding a `OPTION_CLIENT_DATA` option or `OPTION_IAPREFIX` option is entered into a `BNDREPLY`. The information to enter into the `OPTION_CLIENT_DATA` option or `OPTION_IAPREFIX` option in the `BNDREPLY` is described in Section 7.6.

The information contained in an accepted `OPTION_CLIENT_DATA` option is stored in the receiving server's binding database as follows:

1. The `OPTION_CLIENTID` is used to find the client.
2. The other data contained in the top level of the `OPTION_CLIENT_DATA` option is stored with the client as appropriate.
3. For each of the `OPTION_IA_NA`, `OPTION_IA_TA`, or `OPTION_IA_PD` option in the `OPTION_CLIENT_DATA` option and for each of the `OPTION_IAADDR` or `OPTION_IAPREFIX` options in the `IA_*` options:
 1. `OPTION_F_BINDING_STATUS` is stored as the binding-status
 2. `OPTION_F_PARTNER_LIFETIME` is stored in the expiration-time
 3. `OPTION_F_STATE_EXPIRATION_TIME` is stored in the state-expiration-time
 4. `OPTION_F_CLT_TIME` (which MUST NOT be converted with the corrected-base-time, but MUST be converted with the raw value from the `OPTION_LQ_BASE_TIME`) is stored in the partner-raw-clt-time
 5. `OPTION_F_PARTNER_RAW_CLT_TIME` (which MUST NOT be corrected with the time-correction) replaces the client-last-transaction-time if it is later than the current client-last-transaction-time.
 6. `OPTION_F_EXPIRATION_TIME` replaces the partner-lifetime if it is later than the current partner-lifetime.

The information contained in an accepted top level `OPTION_IAPREFIX` option is stored in the receiving server's binding database as follows:

1. The IPv6 Prefix is used to find the prefix.

2. Inside of the IAPrefix-options section:

1. `OPTION_F_BINDING_STATUS` is stored as the binding-status
2. `OPTION_F_PARTNER_LIFETIME` (if any) is stored in the expiration-time
3. `OPTION_F_STATE_EXPIRATION_TIME` (if any) is stored in the state-expiration-time
4. `OPTION_F_EXPIRATION_TIME` (if any) replaces the partner-lifetime if it is later than the current partner-lifetime.

7.6. Sending Binding Replies

A server MUST respond to every BNDUPD message with a BNDREPLY message. The BNDREPLY message MUST contain an `OPTION_CLIENT_DATA` option if the BNDUPD message contained an `OPTION_CLIENT_DATA` option, or it MUST contain an `OPTION_IAPREFIX` option if the BNDUPD message contained an `OPTION_IAPREFIX` option. The BNDREPLY message MUST have the same transaction-id as the BNDUPD message to which it is a response.

Acceptance or rejection of all or a particular part of the BNDUPD message is signaled with a `OPTION_STATUS_CODE` option. An `OPTION_STATUS_CODE` option containing a status-code representing an error is significant, while an `OPTION_STATUS_CODE` option whose status-code contains success is considered informational but does not affect the processing of the BNDREPLY message when it is received by the server that sent the BNDUPD message.

Rejection of all or part of the information in a BNDUPD message is signaled in a BNDREPLY message by use of the `OPTION_STATUS_CODE` message with an error in the status-code field. This rejection can take place at either of two levels -- the top level of the option hierarchy, or the bottom level of the option hierarchy:

Entire BNDUPD: The `OPTION_STATUS_CODE` containing an error is present in the outermost option of the BNDREPLY -- either the single `OPTION_CLIENT_DATA` option or the single `OPTION_IAPREFIX` option. An example of this sort of error might be that a VSS option was present and specified a VPN that might not exist in the receiving server.

Single address or prefix: The `OPTION_STATUS_CODE` containing an error is present in a single `IAADDR` or `IAPREFIX` option which is itself contained in an `OPTION_IA_NA`, `OPTION_IA_TA`, or `OPTION_IA_PD` option. An example of this sort of error might be that a

particular IPv6 address was specified in an IAADDR option that doesn't appear in the receiving server's configuration.

Rejection present at either of these levels indicates rejection of all of the information contained in the option (including any other options contained in that option) where the OPTION_STATUS_CODE option containing an error appears. The converse is not true -- an OPTION_STATUS_CODE option containing success does not signify that all of the contained information has been accepted.

If the BNDREPLY message contains an OPTION_CLIENT_DATA option, then the OPTION_CLIENT_DATA option MUST contain at least the data shown below in its client-options section:

- o OPTION_CLIENTID containing the DUID of the client most recently associated with this IPv6 address*;
 - o OPTION_VSS from the BNDUPD, if any.
 - o OPTION_IA_NA or OPTION_IA_TA for an IPv6 Address or OPTION_IA_PD for an IPv6 Prefix. More than one of either of these options MAY appear if there are more than one associated with this client;
- * Inside of the IA_NA-options, IA_TA-options, or IA_PD-option sections:
- + OPTION_IAADDR for an IPv6 address or an OPTION_IAPREFIX for a IPv6 prefix;
 - IPv6 Address or IPv6 Prefix (with length);
 - Inside of the IAaddr-options or IAprefix-options:
 - o OPTION_STATUS_CODE containing an error code, or containing a success code if a message is required. An OPTION_STATUS_CODE option SHOULD NOT appear with a success code unless a message associated with the success code needs to be included. The lack of an OPTION_STATUS_CODE option is an indication of success.
 - o OPTION_F_BINDING_STATUS containing the binding-status received in the BNDUPD;
 - o OPTION_F_STATE_EXPIRATION_TIME (absolute) containing the state-expiration-time received in the BNDUPD;

- o OPTION_F_PARTNER_LIFETIME_SENT (absolute) containing a duplicate of the OPTION_F_PARTNER_LIFETIME received in the BNDUPD;

If the BNDREPLY message contains a top level OPTION_IAPREFIX option, then the OPTION_IAPREFIX option MUST contain at least the data shown below:

- o IPv6 Prefix (with length);
- o IAprefix-options:
 - * OPTION_VSS from the BNDUPD, if any.
 - * OPTION_STATUS_CODE containing an error code, or containing a success code if a message is required. If the information in the corresponding OPTION_IAPREFIX in the BNDUPD was accepted, and no status message was required (which is the usual case), no OPTION_STATUS_CODE option appears.
 - * OPTION_F_BINDING_STATUS containing the binding-status received in the BNDREPLY;
 - * OPTION_F_STATE_EXPIRATION_TIME (absolute) containing the state-expiration-time received in the BNDREPLY;
 - * OPTION_F_PARTNER_LIFETIME_SENT (absolute) containing a duplicate of the OPTION_F_PARTNER_LIFETIME received in the BNDREPLY;

7.7. Receiving Binding Acks

When a BNDREPLY is received the overall OPTION_CLIENT_DATA option or the overall OPTION_IAPREFIX option may contain an OPTION_STATUS_CODE containing an error, representing a rejection of the entire BNDUPD. An enclosed OPTION_IA_NA, OPTION_IA_TA, or OPTION_IA_PD option may also contain an OPTION_STATUS_CODE containing an error which indicates that everything in containing option has been rejected. Or an individual IAADDR or IAPREFIX option may contain an OPTION_STATUS_CODE option containing an error, indicating that the IAADDR or IAPREFIX option has been rejected. An OPTION_STATUS_CODE containing a success code has no bearing on the acceptance status of the BNDREPLY at any level.

Receipt of a rejection (or a part of a BNDREPLY that has been rejected) requires no processing other than remembering that it has been encountered.

The information contained in the BNDREPLY in an OPTION_CLIENT_DATA that represents an acceptance is stored with the appropriate client and lease, as follows:

1. The OPTION_CLIENTID is used to find the client.
2. For each of the OPTION_IA_NA, OPTION_IA_TA, or OPTION_IA_PD option in the OPTION_CLIENT_DATA option and for each of the OPTION_IAADDR or OPTION_IAPREFIX options they contain:
 1. OPTION_F_PARTNER_LIFETIME_SENT is stored in the acked-partner-lifetime
 2. The time partner-lifetime is set to 0, to indicate that nothing additional needs to be sent to the partner.

Alternatively, the BNDREPLY may contain a top level OPTION_IAPREFIX option, representing information concerning a single prefix lease. If the IAPrefix-options section of the OPTION_IAPREFIX option contains an OPTION_STATUS_CODE representing an error, then it is considered a rejection of the corresponding BNDUPD message. If the OPTION_IAPREFIX option does not contain an OPTION_STATUS_CODE option or if the OPTION_STATUS_CODE option contains a success status, then the three items in the following list are stored in the lease state database, in the section associated with the prefix lease represented by the OPTION_IAPREFIX option.

1. OPTION_F_BINDING_STATUS containing the binding-status received in the BNDREPLY;
2. OPTION_F_STATE_EXPIRATION_TIME (absolute) containing the state-expiration-time received in the BNDREPLY;
3. OPTION_F_PARTNER_LIFETIME_SENT (absolute) containing a duplicate of the OPTION_F_PARTNER_LIFETIME received in the BNDREPLY;

7.8. BNDUPD/BNDREPLY Data Flow

The following diagram shows the relationship of the times described in Section 7.3 with the options used to transmit them. It also relates the times on one failover partner to the other failover partner.

```

----- BNDUPD -----

Source on      OPTION_F in      Storage on
Sending Server -> BNDUPD message -> Receiving Server

                                [ always update ]

partner-lifetime      PARTNER_LIFETIME      expiration-time
client-last-transaction-time  CLT_TIME      (uncorrected)
start-time-of-state   START_TIME_OF_STATE   partner-raw-clt-time
state-expiration-time STATE_EXPIRATION_TIME state-expiration-time

                                [update only if received > current]

expiration-time      EXPIRATION_TIME      partner-lifetime
partner-raw-clt-time PARTNER_RAW_CLT_TIME  client-last-transaction-time

----- BNDREPLY -----

Storage on      OPTION_F in      Storage on
Receiving Server <- BNDUPD message <- Sending Server

                                [ always update ]

acked-partner-lifetime PARTNER_LIFETIME_SENT duplicate of received
                                PARTNER_LIFETIME
(nothing to update)    STATE_EXPIRATION_TIME state-expiration-time
-----

```

Figure 5: BNDUPD and BNDREPLY Time Handling

8. Endpoint States

8.1. 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 DHCP 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 DHCP client requests, it is renew responsive, or it is unresponsive.
- o Allocation Pool -- which pool of addresses (or prefixes) can be used for advertisement on receipt of a SOLICIT or allocation on receipt of a REQUEST, RENEW or REBIND 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 any of the above state variables change state, the state machine is invoked, which may then trigger a change in the current failover state. Thus, whenever the communications status changes, the state machine 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.

In the diagram below, the word (responsive) (r-responsive) or (unresponsive) appears in the states, and refers to whether the server in this state is allowed to responsive, renew responsive, or unresponsive respectively.

In the state transition diagram below, the "+", "-", or "*" in the upper right corner of each state is a notation about whether communication is ongoing with the other server, with "+" meaning that

communications are ok, "-" meaning communications are interrupted,
and "*" meaning that communications may be ok or interrupted.

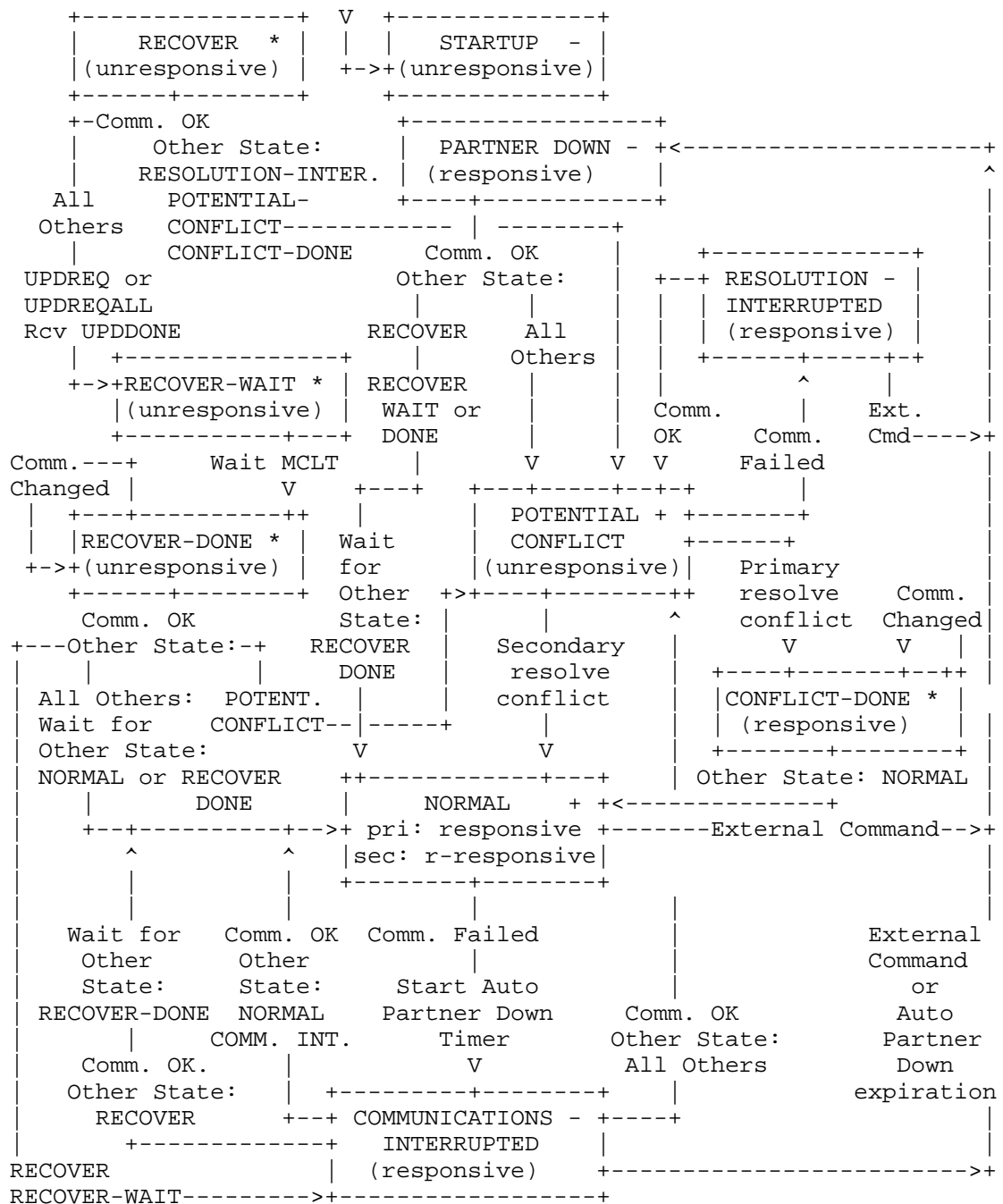


Figure 6: Failover Endpoint State Machine

8.2. State Machine Initialization

The state machine is characterized by storage (in stable storage) of at least the following information:

- o Current failover state.
- o Previous failover state.
- o Start time of current failover state.
- o Partner's failover state.
- o Start time of partner's failover state.
- o Time most recent message received from partner.

The state machine is initialized by reading these data items from stable storage and restoring their values from the information saved. If there is no information in stable storage concerning these items, then they should be initialized as follows:

- o Current failover state: Primary: PARTNER-DOWN, Secondary: RECOVER
- o Previous failover state: None.
- o Start time of current failover state: Current time.
- o Partner's failover state: None until reception of STATE message.
- o Start time of partner's failover state: None until reception of STATE message.
- o Time most recent message received from partner: None until message received.

8.3. 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 6) 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.

8.3.1. Operation in STARTUP State

The server **MUST NOT** be responsive to DHCP clients in STARTUP state.

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

8.3.2. Transition Out of STARTUP State

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

The variables PREVIOUS-STATE and CURRENT-STATE are defined for use in the algorithm description below. PREVIOUS-STATE is simply for storage of a state, while CURRENT-STATE not only stores the current state but also changes the current state of the failover endpoint to whatever state is set into the CURRENT-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:

Implementations will differ in the ways that they deal with the state machine for failover endpoint states. In many cases, state

transitions will occur when communications goes from "OK" to failed, or from failed to "OK", and some implementations will implement a portion of their state machine processing based on these changes.

In these cases, during startup, 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 when in that state (if such transition exists -- some states don't have a communication failed state transition, since they allow both communications 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:

If the server is a primary server: attempt to create a TCP connection to the failover partner. If the server is a secondary server, listen on the failover port and wait for the primary server to connect. See Section 6.1.

Step 5:

Wait for "communications OK".

When and if communications become "OK", 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 prior to communications becoming "OK", the server SHOULD transition to the PREVIOUS-STATE.

8.4. 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.

A server can enter PARTNER-DOWN state either as a result of operator intervention (when an operator determines that the server's partner is, indeed, down), or as a result of an optional auto-partner-down capability where PARTNER-DOWN state is entered automatically after a server has been in COMMUNICATIONS-INTERRUPTED state for a pre-determined period of time.

8.4.1. Operation in PARTNER-DOWN State

The server MUST be responsive in PARTNER-DOWN state, regardless if it is primary or secondary.

It will allow renewal of all outstanding leases.

For delegable prefixes it will allocate leases from its own pool, and after a fixed period of time (the MCLT interval) has elapsed from entry into PARTNER-DOWN state, it may allocate delegable prefixes from the set of all available pools. Server MUST fully deplete its own pool, before starting allocations from its downed partner's pool.

IPv6 addresses available for independent allocation by the other server (at entry to PARTNER-DOWN state) SHOULD NOT be allocated to a client. If one elects to do so anyway, they MUST NOT be allocated to a new client until the MCLT beyond the entry into PARTNER-DOWN state has elapsed.

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

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

In the unlikely case when there are two servers operating in a PARTNER-DOWN state, there is a chance of duplicate leases for the same prefix to be 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 granting new leases first 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.

8.4.2. 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. 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.

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)

- o If the partner is in: [NORMAL, COMMUNICATIONS-INTERRUPTED, PARTNER-DOWN, POTENTIAL-CONFLICT, RESOLUTION-INTERRUPTED, or CONFLICT-DONE] state, then transition to POTENTIAL-CONFLICT state
- o If the partner is in: [RECOVER, RECOVER-WAIT] state stay in PARTNER-DOWN state
- o If the partner is in: [RECOVER-DONE] state transition into NORMAL state

8.5. 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.

8.5.1. 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.

8.5.2. 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 communication 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.

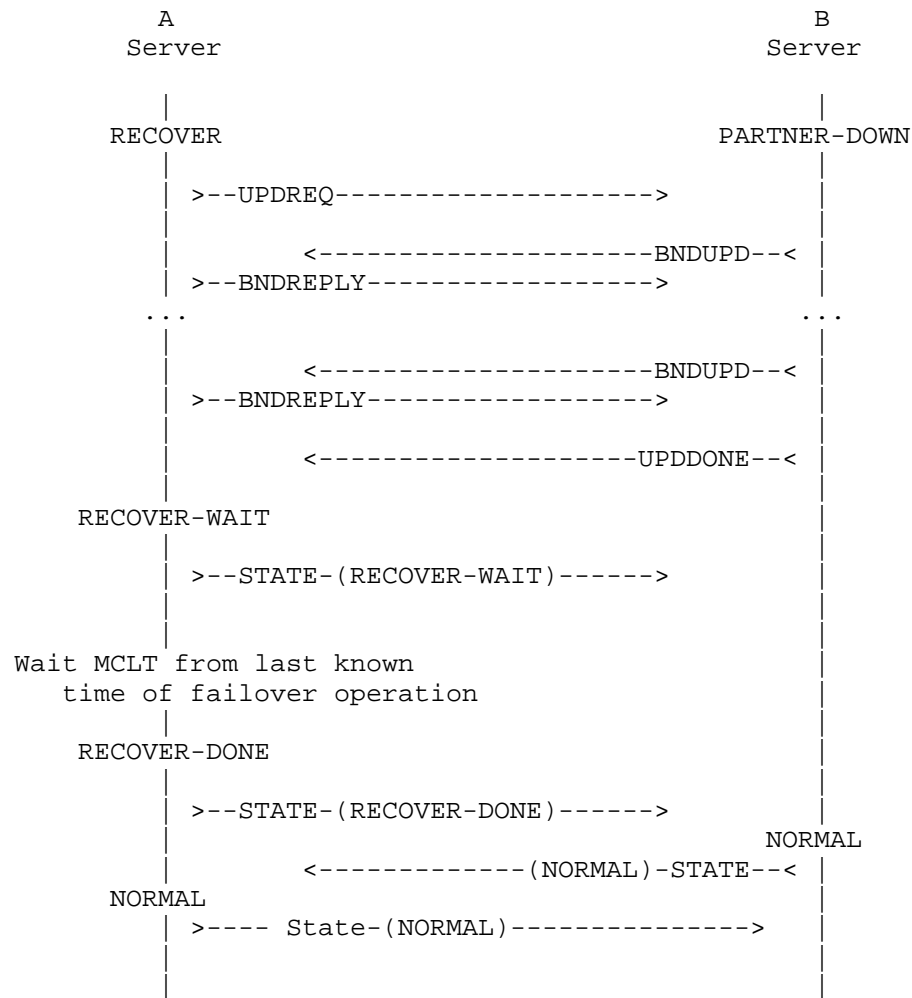


Figure 7: Transition out of RECOVER state

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

8.6. RECOVER-WAIT State

This state indicates that the server has sent 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.

8.6.1. Operation in RECOVER-WAIT State

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

8.6.2. 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 IPv6 addresses or prefixes 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 the server has never before run failover, then there is no need to wait in this state and the server **MAY** transition immediately to RECOVER_DONE state. However, 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 communication 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 if 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 this transition.

8.7. 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.

8.7.1. Operation in RECOVER-DONE State

A server in RECOVER-DONE state **SHOULD** be renew responsive, and **MAY** respond to RENEW requests but **MUST** only change the state of a lease that appears in the RENEW request. It **MUST NOT** allocate any additional leases when in RECOVER-DONE state and should only respond only to RENEW requests where it already has a record of the lease.

8.7.2. 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 the partner server enters RECOVER or RECOVER-WAIT state, this server transitions to COMMUNICATIONS-INTERRUPTED.

If the partner server enters POTENTIAL-CONFLICT state then this server enters POTENTIAL-CONFLICT state as well.

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

8.8. 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 binding 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 delegable prefixes for allocation using the POOLREQ message.

8.8.1. Operation in NORMAL State

The primary server is responsive in NORMAL state. The secondary is renew responsive in NORMAL state.

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

Valid lifetime calculations

As discussed in Section 4.4, the lease interval given to a DHCP client can never be more than the MCLT greater than the most recently acknowledged partner lifetime received 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 partner lifetime sent to its failover partner are implementation dependent.

Lazy update of partner server

After sending a REPLY that includes a lease update to a client, the server servicing a DHCP client request attempts to update its partner with the new binding information. See Section 4.3.

Reallocation of leases 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 BNDREPLY is received for this message, the lease 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 4.2.2.1 for details.

In NORMAL state, each server receives binding updates from its partner server in BNDUPD messages (see Section 7.5.5). It records these in its binding database in stable storage and then sends a corresponding BNDREPLY message to its partner server (see Section 7.6).

8.8.2. 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 prior to the failover server discovering it on its own.

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 periodically, but was (for some reason) incapable of processing BNDUPD messages.

If the communications is determined to not be "ok" (as defined in Section 6.6), 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 a DISCONNECT message from its partner, the server should transition into COMMUNICATIONS-INTERRUPTED state.

8.9. 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 lease 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., auto-partner-down has been configured), then a timer is started for the length of the configured auto-partner-down 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.

8.9.1. Operation in COMMUNICATIONS-INTERRUPTED State

In this state a server MUST respond to all DHCP client requests. When allocating new leases, each server allocates from its own pool, where the primary MUST allocate only FREE delegable prefixes, and the secondary MUST allocate only FREE-BACKUP delegable prefixes, and each server allocates from its own independent IPv6 address ranges. When responding to RENEW messages, each server will allow continued renewal of a DHCP client's current lease regardless of whether that lease was given out by the receiving server or not, although the renewal period MUST NOT exceed the MCLT beyond the latest of: 1) the partner lifetime already acknowledged by the other server, or 2) now, or 3) the partner lifetime received from the partner server.

However, since the server cannot communicate with its partner in this state, the acknowledged partner lifetime will not be updated despite continued RENEW message processing. This is likely to eventually cause the actual lifetimes to converge to the MCLT (unless this is greater than the desired lease time, which would be unusual).

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

8.9.2. Transition Out of COMMUNICATIONS-INTERRUPTED State

If the auto-partner-down 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:

- o NORMAL or COMMUNICATIONS-INTERRUPTED: Transition into the NORMAL state.
- o RECOVER: Stay in COMMUNICATIONS-INTERRUPTED state.
- o RECOVER-DONE: Transition into NORMAL state.
- o PARTNER-DOWN, POTENTIAL-CONFLICT, CONFLICT-DONE, or RESOLUTION-INTERRUPTED: Transition into POTENTIAL-CONFLICT state.

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

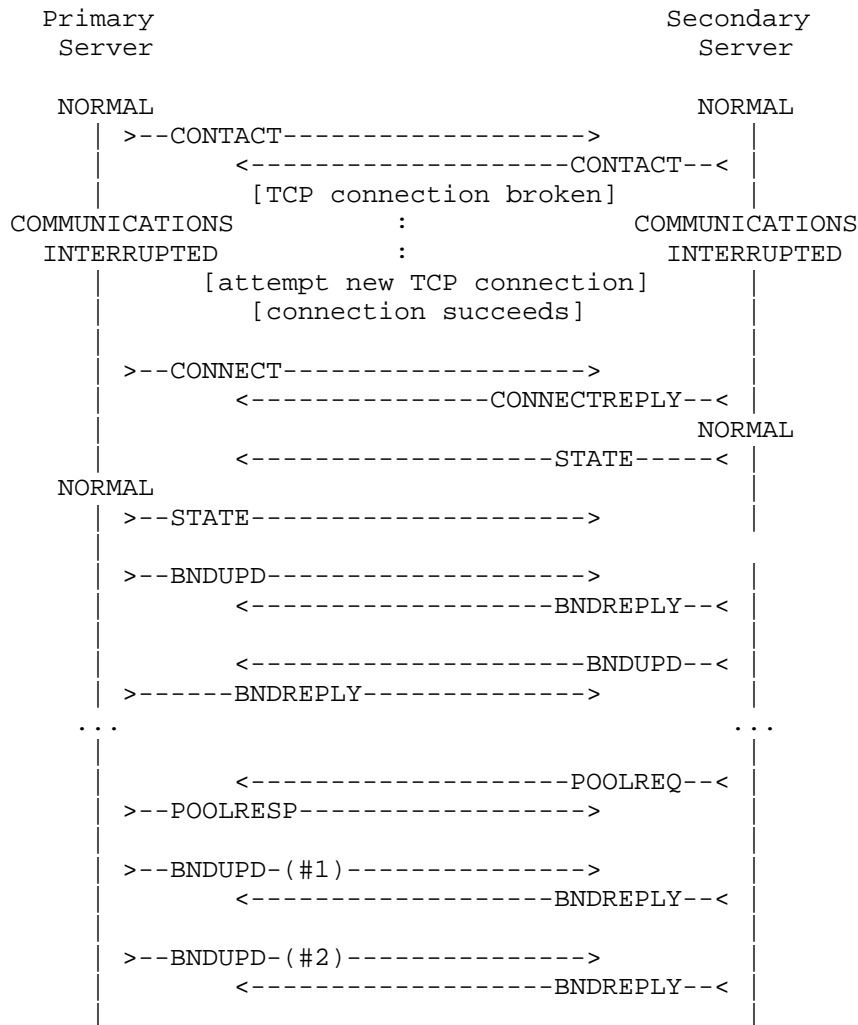


Figure 8: Transition from NORMAL to COMMUNICATIONS-INTERRUPTED and back

8.10. 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 lease has been offered and accepted by two different clients.

It is a goal of the failover 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 which the primary server has not yet acknowledged 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.

8.10.1. Operation in POTENTIAL-CONFLICT State

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

8.10.2. Transition Out of POTENTIAL-CONFLICT State

If communication 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.

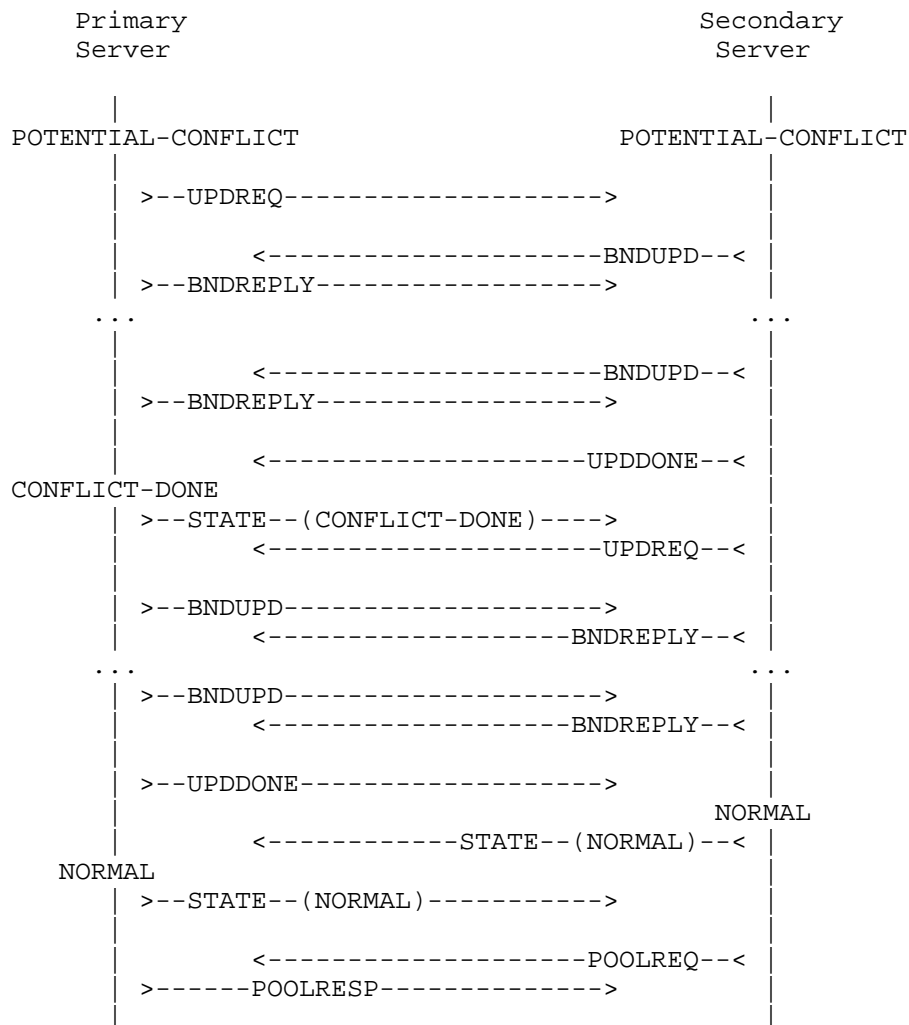


Figure 9: Transition out of POTENTIAL-CONFLICT

8.11. RESOLUTION-INTERRUPTED State

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

The RESOLUTION-INTERRUPTED state exists because servers are not responsive in POTENTIAL-CONFLICT state, and if one server drops out of service while both servers are in POTENTIAL-CONFLICT state, the server that remains in service will not be able to process DHCP

client requests and there will be no DHCP server available to process client requests. The RESOLUTION-INTERRUPTED state is the state that a server moves to if its partner disappears while it is in POTENTIAL-CONFLICT state.

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

8.11.1. Operation in RESOLUTION-INTERRUPTED State

In this state a server MUST respond to all DHCP client requests. When allocating new leases, each server SHOULD allocate from its own pool (if that can be determined), where the primary SHOULD allocate only FREE leases, and the secondary SHOULD allocate only FREE-BACKUP leases. When responding to renewal requests, each server will allow continued renewal of a DHCP client's current lease independent 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 partner lifetime already acknowledged by the other server or 2) now or 3) partner lifetime received from the partner server.

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

8.11.2. 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.

8.12. 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 makes a transition into CONFLICT-DONE state in order that it may be totally responsive to the client load. There is no operational difference between CONFLICT-DONE and NORMAL for primary as in both states it responds to all clients' requests. The distinction between CONFLICT-DONE and NORMAL states is necessary in the event that a load-balancing extension is ever defined.

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

8.12.2. Transition Out of CONFLICT-DONE State

If communication 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. DNS Update Considerations

DHCP servers (and clients) can use DNS Updates as described in RFC 2136 [RFC2136] to maintain DNS name-mappings as they maintain DHCP leases. Many different administrative models for DHCP-DNS integration are possible. Descriptions of several of these models, and guidelines that DHCP servers and clients should follow in carrying them out, are laid out in RFC 4704 [RFC4704].

The nature of the failover protocol introduces some issues concerning DNS updates that are not part of non-failover environments. This section describes these issues, and defines the information which failover partners should exchange in order to ensure consistent behavior. The presence of this section should not be interpreted as requiring an implementation of the DHCPv6 failover protocol to also support DNS updates.

The purpose of this discussion is to clarify the areas where the failover and DHCP DNS update protocols intersect for the benefit of implementations which support both protocols, not to introduce a new requirement into the DHCPv6 failover protocol. Thus, a DHCP server which implements the failover protocol MAY also support DNS updates, but if it does support DNS updates it SHOULD utilize the techniques described here in order to correctly distribute them between the failover partners. See RFC 4704 [RFC4704] as well as RFC 4703 [RFC4703] for information on how DHCP servers deal with potential conflicts when updating DNS even without failover.

From the standpoint of the failover protocol, there is no reason why a server which is utilizing the DNS update protocol to update a DNS server should not be a partner with a server which is not utilizing the DNS update protocol to update a DNS server. However, a server which is not able to support DNS update or is not configured to support DNS update SHOULD output a warning message when it receives BNDUPD messages which indicate that its failover partner is configured to support the DNS update protocol to update a DNS server. An implementation MAY consider this an error and refuse to accept the BNDUPD by returning the status `DNSUpdateNotSupported` in an `OPTION_STATUS_CODE` option in the BNDREPLY message, or it MAY choose to operate anyway, having warned the administrator of the problem in some way.

9.1. Relationship between failover and DNS update

The failover protocol describes the conditions under which each failover server may renew a lease to its current DHCP client, and describes the conditions under which it may grant a lease to a new DHCP client. An analogous set of conditions determines when a failover server should initiate a DNS update, and when it should attempt to remove records from the DNS. The failover protocol's conditions are based on the desired external behavior: avoiding duplicate address and prefix assignments; allowing clients to continue using leases which they obtained from one failover partner even if they can only communicate with the other partner; allowing the secondary DHCP server to grant new leases even if it is unable to communicate with the primary server. The desired external DNS update behavior for DHCPv6 failover servers is similar to that described above for the failover protocol itself:

1. Allow timely DNS updates from the server which grants a lease to a client. Recognize that there is often a DNS update lifecycle which parallels the DHCP lease lifecycle. This is likely to include the addition of records when the lease is granted, and the removal of DNS records when the lease is subsequently made available for allocation to a different client.
2. Communicate enough information between the two failover servers to allow one to complete the DNS update 'lifecycle' even if the other server originally granted the lease.
3. Avoid redundant or overlapping DNS updates, where both failover servers are attempting to perform DNS updates for the same lease-client binding.

4. Avoid situations where one partner is attempting to add RRs related to a lease binding while the other partner is attempting to remove RRs related to the same lease binding.

While DHCPv6 servers configured for DNS update typically perform these operations on both the AAAA and the PTR resource records, this is not required. It is entirely possible that a DHCPv6 server could be configured to only update the DNS with PTR records, and the DHCPv6 clients could be responsible for updating the DNS with their own AAAA records. In this case, the discussions here would apply only to the PTR records.

9.2. Exchanging DNS Update Information

In order for either server to be able to complete a DNS update, or to remove DNS records which were added by its partner, both servers need to know the FQDN associated with the lease-client binding. In addition, to properly handle DNS updates, additional information is required. All of the following information needs to be transmitted between the failover partners:

1. The FQDN that the client requested be associated with the lease. If the client doesn't request a particular FQDN and one is synthesized by the failover server or if the failover server is configured to replace a client requested FQDN with a different FQDN, then the server generated value would be used.
2. The FQDN that was actually placed in the DNS for this lease. It may differ from the client requested FQDN due to some form of disambiguation or other DHCP server configuration (as described above).
3. The status of and DNS update operations in progress or completed.
4. Information sufficient to allow the failover partner to remove the FQDN from the DNS should that become necessary.

These data items are the minimum necessary set to reliably allow two failover partners to successfully share the responsibility to keep the DNS up to date with the leases allocated to clients.

This information would typically be included in BNDUPD messages sent from one failover partner to the other. Failover servers MAY choose not to include this information in BNDUPD messages if there has been no change in the status of any DNS update related to the lease.

The partner server receiving BNDUPD messages containing the DNS update information SHOULD compare the status information and the FQDN

with the current DNS update information it has associated with the lease binding, and update its notion of the DNS update status accordingly.

Some implementations will instead choose to send a BNDUPD without waiting for the DNS update to complete, and then will send a second BNDUPD once the DNS update is complete. Other implementations will delay sending the partner a BNDUPD until the DNS update has been acknowledged by the DNS server, or until some time-limit has elapsed, in order to avoid sending a second BNDUPD.

The FQDN option contains the FQDN that will be associated with the AAAA RR (if the server is performing an AAAA RR update for the client). The PTR RR can be generated automatically from the IPv6 address or prefix value. The FQDN may be composed in any of several ways, depending on server configuration and the information provided by the client in its DHCP messages. The client may supply a hostname which it would like the server to use in forming the FQDN, or it may supply the entire FQDN. The server may be configured to attempt to use the information the client supplies, it may be configured with an FQDN to use for the client, or it may be configured to synthesize an FQDN.

Since the server interacting with the client may not have completed the DNS update at the time it sends the first BNDUPD about the lease binding, there may be cases where the FQDN in later BNDUPD messages does not match the FQDN included in earlier messages. For example, the responsive server may be configured to handle situations where two or more DHCP client FQDNs are identical by modifying the most-specific label in the FQDNs of some of the clients in an attempt to generate unique FQDNs for them (a process sometimes called "disambiguation"). Alternatively, at sites which use some or all of the information which clients supply to form the FQDN, it's possible that a client's configuration may be changed so that it begins to supply new data. The server interacting with the client may react by removing the DNS records which it originally added for the client, and replacing them with records that refer to the client's new FQDN. In such cases, the server SHOULD include the actual FQDN that was used in subsequent DNS update options in any BNDUPD messages exchanged between the failover partners. This server SHOULD include relevant information in its BNDUPD messages. This information may be necessary in order to allow the non-responsive partner to detect client configuration changes that change the hostname or FQDN data which the client includes in its DHCPv6 requests.

9.3. Adding RRs to the DNS

A failover server which is going to perform DNS updates SHOULD initiate the DNS update when it grants a new lease to a client. The server which did not grant the lease SHOULD NOT initiate a DNS update when it receives the BNDUPD after the lease has been granted. The failover protocol ensures that only one of the partners will grant a lease to any individual client, so it follows that this requirement will prevent both partners from initiating updates simultaneously. The server initiating the update SHOULD follow the protocol in RFC 4704 [RFC4704]. The server may be configured to perform a AAAA RR update on behalf of its clients, or not. Ordinarily, a failover server will not initiate DNS updates when it renews leases. In two cases, however, a failover server MAY initiate a DNS update when it renews a lease to its existing client:

1. When the lease was granted before the server was configured to perform DNS updates, the server MAY be configured to perform updates when it next renews existing leases.
2. If a server is in PARTNER-DOWN state, it can conclude that its partner is no longer attempting to perform an update for the existing client. If the remaining server has not recorded that an update for the binding has been successfully completed, the server MAY initiate a DNS update. It MAY initiate this update immediately upon entry to PARTNER-DOWN state, it may perform this in the background, or it MAY initiate this update upon next hearing from the DHCP client.

Note that, regardless of the use of failover, there is a use case for updating the DNS on every lease renewal. If there is a concern that the information in the DNS does not match the information in the DHCP server, updating the DNS on lease renewal is one way to gradually ensure that the DNS has information that corresponds correctly the information in the DHCP server.

9.4. Deleting RRs from the DNS

The failover server which makes a lease PENDING-FREE SHOULD initiate any DNS deletes, if it has recorded that DNS records were added on behalf of the client.

A server not in PARTNER-DOWN state "makes a lease PENDING-FREE" when it initiates a BNDUPD with a binding-status of FREE, FREE-BACKUP, EXPIRED, or RELEASED. Its partner confirms this status by acking that BNDUPD, and upon receipt of the BNDREPLY the server has "made the lease PENDING-FREE". Conversely, a server in PARTNER-DOWN state "makes a lease PENDING-FREE" when it sets the binding-status to FREE,

since in PARTNER-DOWN state no communications is required with the partner.

It is at this point that it should initiate the DNS operations to delete RRs from the DNS. Its partner SHOULD NOT initiate DNS deletes for DNS records related to the lease binding as part of sending the BNDREPLY message. The partner MAY have issued BNDUPD messages with a binding-status of FREE, EXPIRED, or RELEASED previously, but the other server will have rejected these BNDUPD messages.

The failover protocol ensures that only one of the two partner servers will be able to make a lease PENDING-FREE. The server making the lease PENDING-FREE may be doing so while it is in NORMAL communication with its partner, or it may be in PARTNER-DOWN state. If a server is in PARTNER-DOWN state, it may be performing DNS deletes for RRs which its partner added originally. This allows a single remaining partner server to assume responsibility for all of the DNS update activity which the two servers were undertaking.

Another implication of this approach is that no DNS RR deletes will be performed while either server is in COMMUNICATIONS-INTERRUPTED state, since no leases are moved into the PENDING-FREE state during that period.

A failover server SHOULD ensure that a server failure while making a lease PENDING-FREE and initiating a DNS delete does not somehow leave the lease with a RR in the DNS with nothing recorded in the lease state database to trigger a DNS delete.

9.5. Name Assignment with No Update of DNS

In some cases, a DHCP server is configured to return a name to the DHCP client but not enter that name into the DNS. This is typically a name that it has discovered or generated from information it has received from the client. In this case this name information SHOULD be communicated to the failover partner, if only to ensure that they will return the same name in the event the partner becomes the server to which the DHCP client begins to interact.

10. Security Considerations

DHCPv6 failover is an extension of a standard DHCPv6 protocol, so all security considerations from [RFC3315], Section 23 and [RFC3633], Section 15 related to the server apply.

The use of TCP introduces some additional concerns. Attacks that attempt to exhaust the DHCP server's available TCP connection resources can compromise the ability of legitimate partners to

receive service. Malicious requestors who succeed in establishing connections but who then send invalid messages, partial messages, or no messages at all can also exhaust a server's pool of available connections.

DHCPv6 failover can operate in secure or insecure mode. Secure mode (using TLS) would be indicated when the TCP connection between failover partners is open to external monitoring or interception. Insecure mode should only be used when the TCP connection between failover partners remains within a set of protected systems. Details of such protections are beyond the scope of this document. Failover servers **MUST** use the approach documented in Section 9.1 of [RFC7653] to decide to use or not to use TLS when connecting with the failover partner.

The threats created by using failover directly mirror those from using DHCPv6 itself: information leakage through monitoring, and disruption of address assignment and configuration. Monitoring the failover TCP connection provides no additional data beyond that available from monitoring the interactions between DHCPv6 clients and the DHCPv6 server. Likewise, manipulating the data flow between failover servers provides no additional opportunities to disrupt address assignment and configuration beyond that provided by acting as a counterfeit DHCP server. Protection from both threats is easier than with basic DHCPv6, as only a single TCP connection needs to be protected. Either use secure mode to protect that TCP connection or ensure that it can only exist with a set of protected systems.

When operating in secure mode, TLS [RFC5246] is used to secure the connection. The recommendations in [RFC7525] **SHOULD** be followed when negotiating a TLS connection.

Servers **SHOULD** offer configuration parameters to limit the sources of incoming connections through validation and use of the digital certificates presented to create a TLS connection. They **SHOULD** also limit the number of accepted connections and limit the period of time during which an idle connection will be left open.

Authentication for DHCPv6 messages [RFC3315] **MUST NOT** be used to attempt to secure transmission of the messages described in this document. If authentication is desired, secure mode using TLS **SHOULD** be employed as described in Sections 8.2 and 9.1 of [RFC7653].

11. IANA Considerations

IANA is requested to assign values for the following new DHCPv6 Message types in the registry maintained in <http://www.iana.org/assignments/dhcpv6-parameters>:

- o BNDUPD (TBD1)
- o BNDREPLY (TBD2)
- o POOLREQ (TBD3)
- o POOLRESP (TBD4)
- o UPDREQ (TBD5)
- o UPDREQALL (TBD6)
- o UPDDONE (TBD7)
- o CONNECT (TBD8)
- o CONNECTREPLY (TBD9)
- o DISCONNECT (TBD10)
- o STATE (TBD11)
- o CONTACT (TBD12)

IANA is requested to assign values for the following new DHCPv6 Option codes in the registry maintained in <http://www.iana.org/assignments/dhcpv6-parameters>:

- OPTION_F_BINDING_STATUS (TBD13)
- OPTION_F_CONNECT_FLAGS (TBD14)
- OPTION_F_DNS_REMOVAL_INFO (TBD15)
- OPTION_F_DNS_HOST_NAME (TBD16)
- OPTION_F_DNS_ZONE_NAME (TBD17)
- OPTION_F_DNS_FLAGS (TBD18)
- OPTION_F_EXPIRATION_TIME (TBD19)
- OPTION_F_MAX_UNACKED_BNDUPD (TBD20)
- OPTION_F_MCLT (TBD21)
- OPTION_F_PARTNER_LIFETIME (TBD22)

OPTION_F_PARTNER_LIFETIME_SENT (TBD23)

OPTION_F_PARTNER_DOWN_TIME (TBD24)

OPTION_F_PARTNER_RAW_CLT_TIME (TBD25)

OPTION_F_PROTOCOL_VERSION (TBD26)

OPTION_F_KEEPALIVE_TIME (TBD27)

OPTION_F_RECONFIGURE_DATA (TBD28)

OPTION_F_RELATIONSHIP_NAME (TBD29)

OPTION_F_SERVER_FLAGS (TBD30)

OPTION_F_SERVER_STATE (TBD31)

OPTION_F_START_TIME_OF_STATE (TBD32)

OPTION_F_STATE_EXPIRATION_TIME (TBD33)

IANA is requested to assign values for the following new DHCPv6 Status codes in the registry maintained in <http://www.iana.org/assignments/dhcpv6-parameters>:

AddressInUse (TBD34)

ConfigurationConflict (TBD35)

MissingBindingInformation (TBD36)

OutdatedBindingInformation (TBD37)

ServerShuttingDown (TBD38)

DNSUpdateNotSupported (TBD39)

ExcessiveTimeSkew (TBD40)

12. Acknowledgements

This document extensively uses concepts, definitions and other parts of an effort to document failover for DHCPv4. Authors would like to thank Shawn Routhier, Greg Rabil, Bernie Volz and Marcin Siodelski for their significant involvement and contributions. In particular, Bernie Volz and Shawn Routhier provided detailed and substantive technical reviews of the draft. Authors would like to thank

VithalPrasad Gaitonde, Krzysztof Gierlowski, Krzysztof Nowicki and Michal Hoefft for their insightful comments.

13. References

13.1. Normative References

- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <<http://www.rfc-editor.org/info/rfc1035>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <<http://www.rfc-editor.org/info/rfc2136>>.
- [RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <<http://www.rfc-editor.org/info/rfc3315>>.
- [RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, DOI 10.17487/RFC3633, December 2003, <<http://www.rfc-editor.org/info/rfc3633>>.
- [RFC4703] Stapp, M. and B. Volz, "Resolution of Fully Qualified Domain Name (FQDN) Conflicts among Dynamic Host Configuration Protocol (DHCP) Clients", RFC 4703, DOI 10.17487/RFC4703, October 2006, <<http://www.rfc-editor.org/info/rfc4703>>.
- [RFC4704] Volz, B., "The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Client Fully Qualified Domain Name (FQDN) Option", RFC 4704, DOI 10.17487/RFC4704, October 2006, <<http://www.rfc-editor.org/info/rfc4704>>.
- [RFC5007] Brzozowski, J., Kinnear, K., Volz, B., and S. Zeng, "DHCPv6 Leasequery", RFC 5007, DOI 10.17487/RFC5007, September 2007, <<http://www.rfc-editor.org/info/rfc5007>>.

- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.
- [RFC5460] Stapp, M., "DHCPv6 Bulk Leasequery", RFC 5460, DOI 10.17487/RFC5460, February 2009, <<http://www.rfc-editor.org/info/rfc5460>>.
- [RFC6607] Kinnear, K., Johnson, R., and M. Stapp, "Virtual Subnet Selection Options for DHCPv4 and DHCPv6", RFC 6607, DOI 10.17487/RFC6607, April 2012, <<http://www.rfc-editor.org/info/rfc6607>>.
- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<http://www.rfc-editor.org/info/rfc7525>>.
- [RFC7653] Raghuvanshi, D., Kinnear, K., and D. Kukrety, "DHCPv6 Active Leasequery", RFC 7653, DOI 10.17487/RFC7653, October 2015, <<http://www.rfc-editor.org/info/rfc7653>>.

13.2. Informative References

- [RFC7031] Mrugalski, T. and K. Kinnear, "DHCPv6 Failover Requirements", RFC 7031, DOI 10.17487/RFC7031, September 2013, <<http://www.rfc-editor.org/info/rfc7031>>.

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 Avenue
Boxborough, Massachusetts 01719
USA

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