Dynamic Host Configuration (DHC)
Internet-Draft
Intended status: Standards Track
Expires: March 17, 2014

T. Mrugalski
ISC
K. Kinnear
Cisco

September 13, 2013

DHCPv6 Failover Design
draft-ietf-dhc-dhcpv6-failover-design-04

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on March 17, 2014.

Table of Contents

1. Requirements Language 3
2. Glossary 4
3. Introduction 5
3.1. Design Requirements 6
3.2. Features out of Scope: Load Balancing 6
4. Protocol Overview 7
4.1. Failover State Machine Overview 8
4.2. Messages 10
5. Connection Management 12
5.1. Creating Connections 12
5.2. Endpoint Identification 13
6. Resource Allocation 14
6.1. Proportional Allocation 14
6.2. Independent Allocation 17
6.3. Choosing Allocation Algorithm 17
7. Information model 18
8. Failover Mechanisms 23
8.1. Time Skew 23
8.2. Lazy updates 23
8.3. MCLT concept 24
8.3.1. MCLT example 25
8.4. Unreachability detection 26
8.5. Re-allocating Leases 27
8.6. Sending Binding Update 28
8.7. Receiving Binding Update 29
8.8. Conflict Resolution 30
8.9. Acknowledging Reception 32
9. Endpoint States 32
9.1. State Machine Operation 32
9.2. State Machine Initialization 35
9.3. STARTUP State 35
9.3.1. Operation in STARTUP State 36
9.3.2. Transition Out of STARTUP State 36
9.4. PARTNER-DOWN State 38
9.4.1. Operation in PARTNER-DOWN State 38
9.4.2. Transition Out of PARTNER-DOWN State 39
9.5. RECOVER State 39
9.5.1. Operation in RECOVER State 39
9.5.2. Transition Out of RECOVER State 40
9.6. RECOVER-WAIT State 41

9.6.1. Operation in RECOVER-WAIT State 41
9.6.2. Transition Out of RECOVER-WAIT State 41
9.7. RECOVER-DONE State 42
9.7.1. Operation in RECOVER-DONE State 42
9.7.2. Transition Out of RECOVER-DONE State 42
9.8. NORMAL State 43
9.8.1. Operation in NORMAL State 43
9.8.2. Transition Out of NORMAL State 44
9.9. COMMUNICATIONS-INTERRUPTED State 44
9.9.1. Operation in COMMUNICATIONS-INTERRUPTED State 45
9.9.2. Transition Out of COMMUNICATIONS-INTERRUPTED State . 45
9.10. POTENTIAL-CONFLICT State 47
9.10.1. Operation in POTENTIAL-CONFLICT State 47
9.10.2. Transition Out of POTENTIAL-CONFLICT State 47
9.11. RESOLUTION-INTERRUPTED State 48
9.11.1. Operation in RESOLUTION-INTERRUPTED State 49
9.11.2. Transition Out of RESOLUTION-INTERRUPTED State . . . 49
9.12. CONFLICT-DONE State 49
9.12.1. Operation in CONFLICT-DONE State 49
9.12.2. Transition Out of CONFLICT-DONE State 50
10. Proposed extensions 50
10.1. Active-active mode 50
11. Dynamic DNS Considerations 50
11.1. Relationship between failover and dynamic DNS update . . 51
11.2. Exchanging DDNS Information 52
11.3. Adding RRs to the DNS 54
11.4. Deleting RRs from the DNS 54
11.5. Name Assignment with No Update of DNS 55
12. Reservations and failover 55
13. Security Considerations 57
14. IANA Considerations 57
15. Acknowledgements 57
16. References 58
16.1. Normative References 58
16.2. Informative References 58
Authors' Addresses 59

1. Requirements Language

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

2. Glossary

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

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

- DDNS - Dynamic DNS. Typically used as an acronym referring to dynamic update of the DNS.

- 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 associated with that relationship. Note that a single prefix can only be associated with a single failover relationship. This failover endpoint can take actions and hold unique states. Typically, there is one failover endpoint per partner (server), although there may be more. 'Server' and 'failover endpoint' are synonymous only if the server participates in only one failover relationship. However, for the sake of simplicity 'Server' is used throughout the document to refer to a failover endpoint unless to do so would be confusing.

- Failover communication - all messages exchanged between partners.

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

- Lease - an association of a DHCPv6 client with an IPv6 address or delegated prefix.

- Partner - name of the other DHCPv6 server that participates in failover relationship. When the role (primary or secondary) is not important, the other server is referred to as a "failover partner" or simply partner.

- Primary Server - First out of two DHCPv6 servers that participate in a failover relationship. In active-passive mode this is the server that handles most of the client traffic. Its failover partner is referred to as secondary server.

- Proportional Allocation - an allocation algorithm that splits the available resources between the primary and secondary servers and maintains proportions between available resources on both. It is particularly well suited for more limited resources. See Section 6.1 for details.

- Resource - Any type of resource that is managed by DHCPv6. Currently there are three types of such resources defined: a non-temporary IPv6 address, a temporary IPv6 address, and an IPv6 prefix. Other resource types may be defined in the future.

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

- Secondary Server - Second of two DHCPv6 servers that participate in a failover relationship. Its failover partner is referred to as the primary server. In active-passive mode this server (the secondary) typically does not handle client traffic and acts as a backup.

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

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

3. Introduction

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

This protocol defines 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 does receive clients' requests, but does not respond to them and only maintains a copy of lease database and is ready to take over incoming queries in case of primary server failure). Active-active mode (i.e. both servers actively handling clients' requests) is currently not supported for the sake of simplicity. Such a mode is likely to be defined as an extension at a later time and will probably be based on [I-D.ietf-dhc-dhcpv6-load-balancing].

The failover protocol is designed to provide lease stability for leases with lease times beyond a short period. Due in part to the additional overhead required as well as requirements to handle time skew between failover partners (See Section 8.1), failover is not suitable for leases shorter than 30 seconds. The DHCPv6 Failover protocol MUST NOT be used for leases shorter than 30 seconds.

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

3.1. Design Requirements

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

1. Minimize Asymmetry - while there are two distinct roles in failover (primary and secondary server), the differences between those two roles should be as small as possible. This will yield a simpler design as well as a simpler implementation of that design.

3.2. Features out of Scope: Load Balancing

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

It is likely that active-active mode that is essentially a load balancing will be defined as an extension in the near future.

4. Protocol Overview

The DHCPv6 Failover Protocol is defined as a communication between failover partners with all associated algorithms and mechanisms. Failover communication is conducted over a TCP connection established between the partners. The protocol reuses the framing format specified in Section 5.1 of DHCPv6 Bulk Leasequery [RFC5460], but uses different message types. New failover-specific message types are listed in Section 4.2. All information is sent over the connection as typical DHCPv6 messages that convey DHCPv6 options, following the format defined in Section 22.1 of [RFC3315].

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

If the primary server cannot immediately establish a connection with its partner, it will continue to attempt to establish a connection. See Section 5.1 for details.

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

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

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

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

The major problem with a lazy update mechanism is when the server crashes after sending a response to client, but before sending the lazy update to its partner (or when communication between partners is interrupted). To solve this problem, the concept known as the Maximum Client Lead Time (initially designed for DHCPv4 failover) is used. The MCLT is the maximum amount of time that one server can extend a lease for a client's binding beyond the time known by its failover partner. See Section 8.3 for a detailed description how the MCLT affects assigned lifetimes.

Servers verify each others availability by periodically exchanging CONTACT messages. See Section 8.4 for discussion about detecting a partner's unreachability.

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

4.1. Failover State Machine Overview

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

Each server MUST be in one of the well defines states. Depending on its current state a server may be either responsive (responds to clients' queries) or unresponsive (clients' queries are ignored).

A server starts its operation in the short-lived STARTUP state. A server determines its partner reachability and state and sets its own state based on that determination. It typically returns back to the state it was in before shutdown, though the details can be complicated. See Section 9.3.2.

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

If a server discovers that its partner is no longer reachable, it goes to COMMUNICATIONS-INTERRUPTED state. A server must be extra cautious as it can't distinguish if its partner is down or just communication between servers is interrupted. Since communication between partners is not possible, a server must act on the assumption that its partner is up. A failover server must follow a defined procedure, in particular, it MUST NOT extend any lease more than the MCLT beyond its partner's knowledge of the lease expiration time. This imposes an additional burden on the server, in that clients will return to the server for lease renewals more frequently than they would otherwise. Therefore it is not recommended to operate for prolonged periods in this state. Once communication is reestablished, a server may go into NORMAL, POTENTIAL-CONFLICT or PARTNER-DOWN state. It may also stay in COMMUNICATIONS-INTERRUPTED state if certain conditions are met.

Once a server is switched into PARTNER-DOWN (when auto-partner-down is used or as a result of administrative action), it can extend leases, regardless of the original server that initially granted the lease. In that state server handles leases from its own pool, but once its own pool is depleted is also able to serve pool from its downed partner. Some MCLT restrictions no longer apply, but the MCLT still affects whether or not a particular lease can be given to a different client. See Section 9.4.1 for details. Operation in this mode is less demanding for the server that remains operational, than in COMMUNICATIONS-INTERRUPTED state, but PARTNER-DOWN does not offer any kind of redundancy. Even when in PARTNER-DOWN state, a failover server continues to attempt to connect with its failover partner.

A server switches into RECOVER state when any of a variety of conditions are encountered:

- When a backup server contacts its failover partner for the first time.
- When either server discovers that its failover partner has contacted it before but it has no local record of this contact. If the record of previous contact is held in the lease-state database, then this situation implies that the server has lost its lease state database.
- When its failover partner is in PARTNER-DOWN state.

Any of these conditions signal that the server needs to refresh its lease-state database from its partner. Once this operation is complete, it switches to RECOVER-WAIT and later to RECOVER-DONE. See Section 9.6.2.

Once servers reestablish connection, they discover each others' state. Depending on the conditions, they may return to NORMAL or move to POTENTINAL-CONFLICT if the partner is in a state that doesn't allow a simple re-integration of the server's lease state databases. It is a goal of this protocol to minimize the possibility that POTENTIAL-CONFLICT state is ever entered. Servers running in POTENTIAL-CONFLICT do not respond to clients' requests and work only on resolving potential conflicts. Once outstanding lease updates are exchanged, servers move to CONFLICT-DONE or NORMAL states.

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

A server that is being shut down sends a DISCONNECT message. See Section 4.2. A server that receives a DISCONNECT message moves into COMMUNICATIONS-INTERRUPTED state.

4.2. Messages

The failover protocol is centered around the message exchanges used by one server to update its partner and respond to received updates. It should be noted that no specific formats or message type values are assigned in this document. Appropriate implementation details will be specified in a separate protocol specification document. The following list enumerates these messages:

- BNDUPD - The binding update message is used to send the binding lease changes to the partner. One message may contain one or more lease updates. The partner is expected to respond with a BNDACK message.

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

- POOLREQ - The Pool Request message is used by one server (typically secondary) to request allocation of resources (addresses or prefixes) from its partner. The partner responds with POOLRESP.

- POOLRESP - The Pool Response message is used by one server (typically primary) to indicate that it has responded to its partner's request for resources allocation.

- UPDREQ - The update request message is used by one server to request that its partner send all binding database changes that have not been sent and confirmed already. Requested partner is expected to respond with zero or more BNDUPD messages, followed by UPDDONE that signals end of updates.

- UPDREQALL - The update request all is used by one server to request that all binding database information be sent in order to recover from a total loss of its binding database by the requesting server. Requested server responds with zero or more BNDUPD messages, followed by UPDDONE that signal end of updates.

- UPDDONE - The update done message 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.

- CONNECT - The connect message is used by the primary server to establish a high level connection with the other server, and to transmit several important configuration data items between the servers. The partner is expected to confirm by responding with CONNECTACK message.

- CONNECTACK - The connect acknowledgement message is used by the secondary server to respond to a CONNECT message from the primary server.

- DISCONNECT - The disconnect message is used by either server when closing a connection and shutting down. No response is required for this message.

- STATE - The state message 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 current state, e.g. after connection is established in COMMUNICATIONS-INTERRUPTED or PARTNER-DOWN states.

- CONTACT - The contact message is used by either server to ensure that the other server 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.

5. Connection Management

5.1. Creating Connections

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

Every secondary server implementing the failover protocol MUST listen for connection attempts from the primary server.

When a connection attempt succeeds, the primary server which has initiated the connection attempt MUST send a CONNECT message down the connection.

When a connection attempt is received, the only information that the receiving 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 just await the CONNECT message to determine which relationship this connection is to serve.

If it has no secondary relationships with the connecting server, it MUST drop the connection. The goal is to limit the resources expended dealing with attempts to create a spurious failover 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 simply listens for connection attempts from the primary server.

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

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

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

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

5.2. Endpoint Identification

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

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

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

Any failover endpoint can take actions and hold unique states.

This document frequently describes the behavior of the protocol in terms of primary and secondary servers, not primary and secondary failover endpoints. However, it is important to remember that every 'server' described in this document is in reality a failover endpoint that resides in a particular process, and that several failover 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. Conversely, on a particular server, any given prefix will be associated with exactly one failover endpoint.

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

6. Resource Allocation

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

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

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

6.1. Proportional Allocation

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

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

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

Pools governed by proportional allocation are used for allocation when the server is in all states, except PARTNER-DOWN. In PARTNER-DOWN state the healthy partner can allocate from either pool (both its own, and its partner's after some time constraints have elapsed). This allocation and maintenance of these address pools is an area of some sensitivity, since the goal is to maintain a more or less constant ratio of available addresses between the two servers.

The initial allocation 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 resources to the secondary 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 resources (based on some configured ratio).

The primary server SHOULD examine some or all of its database from time to time to determine if resources should be shifted between the primary and secondary (in either direction). The POOLREQ/POOLRESP message exchange allows the secondary server to explicitly request that the primary server examine the entirety of its database to ensure that the secondary has the appropriate resources available.

Servers frequently have several kinds of resources available on a particular network segment. The failover protocol assumes that both primary and secondary servers are configured in such a way that each knows the type and number of resources on every network segment

participating in the failover protocol. The primary server is responsible for allocating the secondary server the correct proportion of available resources of each kind.

The resources are delegated to the secondary using the BNDUPD message with a state of FREE_BACKUP, which indicates the resource is now available for allocation by the secondary. Once the message is sent, the primary MUST NOT use these resources for allocation to DHCPv6 clients.

Available resources can be delegated back to the primary server in certain cases. BNDUPD will contain state FREE for leases that were previously in FREE_BACKUP 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 prefixes and determine if the secondary needs more resources from any of the prefixes.

In order to support a reasonably dynamic balance of the resources between the failover partners, the primary server needs to do additional work to ensure that the secondary server has as many resources as it needs (but that it doesn’t have more than it needs).

The primary server SHOULD examine the balance of available resources between the primary and secondary for a particular prefix whenever the number of available resources for either the primary or secondary changes by more than a configured limit. The primary server SHOULD adjust the available resource balance as required to ensure the configured resource 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.

An example of a threshold approach is: do not attempt to re-balance the prefixes on the primary and secondary until the out of balance value exceeds a configured value.

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

6.2. Independent Allocation

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

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

Resources using the independent allocation approach are ignored when a server processes a POOLREQ message.

During COMMUNICATION-INTERRUPTED events, a partner MAY continue extending existing leases when requested by clients. A healthy partner MUST NOT lease resources that were assigned to its downed partner and later released by a client unless it is in PARTNER-DOWN state. When it is in PARTNER-DOWN state, a server SHOULD use its own pool first and then it MAY start making new assignments from its downed partner’s pool. As the assumption is that independent allocation should be used only when available resources are vast and not expected to be fully used at any given time, it is very unlikely that the server will ever need to use its downed partner pools. This makes a recovery even after prolonged down-time much easier.

6.3. Choosing Allocation Algorithm

All implementations SHOULD support both the proportional allocation algorithm and the independent allocation algorithm. The specific requirements for support (i.e., which algorithm(s) MUST be supported), and the assignment of a specific algorithm to a specific allocation domain, would be documented in any protocol specifications that follow from this document.

The proportional allocation mechanism is more flexible as it can dynamically rebalance available resources between servers. That balance creates an additional burden for the servers and generates more traffic between servers. The proportional algorithm can be considered more efficient at managing available resources, compared to the independent algorithm. That is an important aspect when working in a network that is nearing address and/or prefix depletion.

Independent allocation can be used when the number of available resources are large and there is no realistic danger of running out of resources. Use of the independent allocation makes communication between partners simpler. It also makes recovery easier and potential conflict less likely to appear.

Typically independent allocation is used for IPv6 addresses, because even for /64 pools a server will never run out of addresses to assign, so there is no need to rebalance. For the prefix delegation mechanism, available resources are typically much smaller, so there is a danger of running out of prefixes. Therefore typically proportional allocation will be used for prefix delegations. Independent allocation still may be used, but the implication must be well understood. For example in a network that delegates /64 prefixes out of a /48 prefix (so there can be up to 65536 prefixes delegated) and a 1000 requesting routers, it is safe to use independent allocation.

It should be stressed that the independent allocation algorithm SHOULD NOT be used when the number of resources is limited and there is a realistic danger of depleting resources. If this recommendation is violated, it may lead to a case when one server denies clients due to pool depletion despite the fact that the other partner still has many resources available.

With independent allocation it is very unlikely for a remaining healthy server to allocate resources from its unavailable partner’s pool. That makes recovery easier and any potential conflicts are less likely to appear.

7. Information model

In most DHCP servers a resource (an IP address or a prefix) can take on several different binding-status values, sometimes also called lease states. While no two DHCP server implementations probably 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 denominator binding-status values must be defined. It is not expected that these values correspond with any actual implementation of the DHCP protocol in a DHCP server, but rather that the binding-status values defined in this document should be a common denominator of 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 FREE*. Client identification SHOULD appear.

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

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

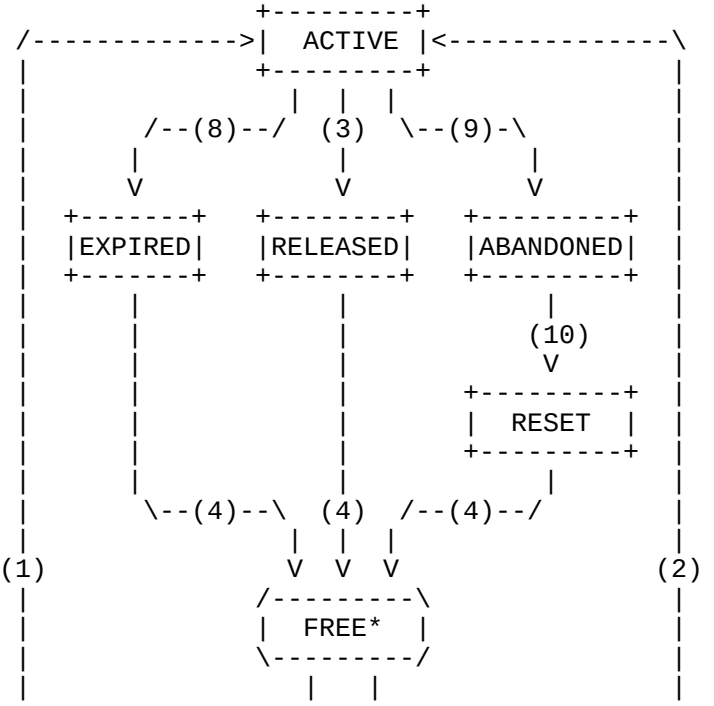
FREE -- Is used when a DHCP server needs to communicate that a resource is unused by any client, but it was not just released, expired or reset by a network administrator. When the partner acks the BNDUPD of a FREE lease, the server marks the lease as available for assignment by the primary server. Note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

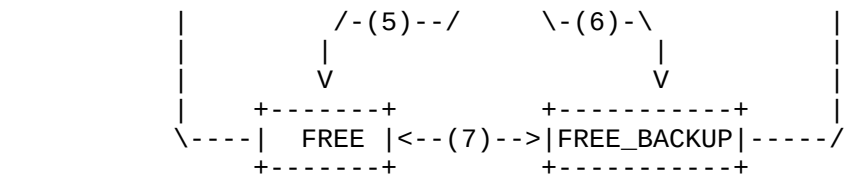
FREE_BACKUP -- indicates that this resource can be allocated by the secondary server to a client at any time. Note that the primary server running in PARTNER-DOWN state, after waiting the MCLT, the resource MAY be allocated to a client by the primary server if proportional algorithm was used. Client identification MAY appear and indicates the last client to have used this resource 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 said lease. Client identification MAY appear.

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

The lease state machine has been presented in Figure 1. 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 FREE*. Once it is reached, the state machine immediately transitions to either FREE or FREE_BACKUP state.





FREE* transition

Figure 1: 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 in 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). Addresses (or 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 resource. One possible example of such use is a Neighbor Discovery or ICMPv6 Echo check if the address is still in use.

The resource that is no longer in use (due to expiration or release), becomes FREE*. Depending of what allocation algorithm is used, the resource that is no longer is use, returns to primary (FREE) or secondary pool (FREE_BACKUP). The conditions for specific transitions are depicted in Figure 2.

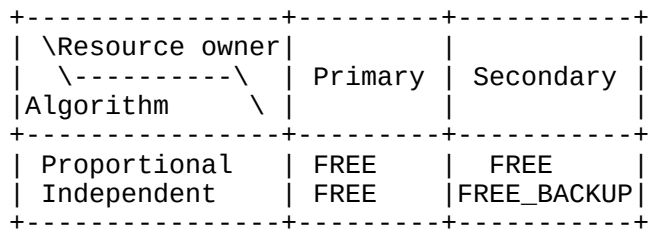


Figure 2: FREE* State Transitions

In case of servers operating in active-passive mode, while a majority of the resources are owned by the primary server, the secondary server will need a portion of the resources to serve new clients while operating in COMMUNICATION-INTERRUPTED state and also in PARTNER-DOWN state before it can take over the entire address pool (after the expiry of MCLT).

The secondary server cannot simply take over the entire resource pool immediately, since it could also be that both servers are able to communicate with DHCP clients, but unable to communicate with each other.

The size of the resource pool allocated to the secondary is specified as a percentage of the currently available resources. Thus, as the number of available resources changes on the primary server, the number of resources available to the secondary server MUST also change, although the frequency of the changes made to the secondary server's pool of address resources SHOULD be low enough to not use significant processing power or network bandwidth.

The required size of this private pool allocated to the secondary server is based only on the arrival rate of new DHCP clients and the length of expected downtime of the primary server, and is not directly influenced by the total number of DHCP clients supported by the server pair.

8. Failover Mechanisms

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

8.1. Time Skew

Partners exchange information about known lease states. To reliably compare a known lease state with an update received from a partner, servers must be able to reliably compare the times stored in the known lease state with the times received in the update. Although a simple approach would be to require both partners to use synchronized time, e.g. by using NTP, such a service may not always be available in some scenarios that failover expects to cover. Therefore a mechanism to measure and track relative time differences between servers is necessary. To do so, each message MUST contain information about the time of the transmission in the time context of the transmitter. The transmitting server MUST set this as close to the actual transmission as possible. Transmission here is when data is added to the send queue of the socket (or the equivalent), as the application may not know about the time of the actual transmission of the "wire". 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.

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

8.2. Lazy updates

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

Although the lazy update mechanism does not introduce additional delays in server response times, it introduces other difficulties. The key problem with lazy update is that when a server fails after updating a client with a particular lease time and before updating its partner, the partner will believe that a lease has expired even though the client still retains a valid lease on that address or prefix. It is also possible that the partner will have no record at all of the lease of the resource to the client.

8.3. MCLT concept

In order to handle problem introduced by lazy updates (see Section 8.2), a period of time known as the "Maximum Client Lead Time" (MCLT) is defined and must be known to both the primary and secondary servers. Proper use of this time interval places an upper bound on the difference allowed between the lease time provided to a DHCPv6 client by a server and the lease time known by that server’s failover partner.

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

The fundamental relationship on which much of the correctness of this protocol depends is that the lease expiration time known to a DHCPv6 client MUST NOT be greater by more than the MCLT beyond the potential expiration time known to that server’s failover partner.

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

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

The different times are:

desired valid lifetime:

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

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

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

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

8.3.1. MCLT example

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

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

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

When the primary server receives a BNDACK to its update of the secondary server’s (partner’s) potential valid lifetime, it records

that as the acknowledged potential valid lifetime. A server MUST NOT send a BNDACK in response to a BNDUPD message until it is sure that the information in the BNDUPD message has been updated in its lease database. See Section 8.9. Thus, the primary server in this case can be sure that the secondary server has recorded the potential lease interval in its stable storage when the primary server receives a BNDACK message from the secondary server.

When the DHCPv6 client attempts to renew at T1 (approximately one half an hour from the start of the lease), the primary server again determines the desired valid lifetime, which is still 3 days. It then compares this with the original acknowledged potential valid 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 potential valid interval is 3 days. Adding the MCLT to this yields 3 days plus 1 hour, which is more than the desired valid lifetime of 3 days. So the client is renewed for the desired valid lifetime -- 3 days.

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

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

8.4. Unreachability detection

Each partner MUST maintain a FO_SEND timer for each failover connection. The FO_SEND timer is reset every time any message is transmitted. If the timer reaches the FO_SEND_MAX value, a CONTACT message is transmitted and timer is reset. The CONTACT message may be transmitted at any time. An implementation MAY use additional mechanisms to detect partner unreachability.

Implementers are advised to keep in mind that the timer based CONTACT message mechanism is not perfect and may not detect some failures.

In particular, if the partner is using one interface to reach clients ("downlink") and another to reach its partner ("uplink"), it is possible that communication with the clients will break, yet the mechanism will still claim full reachability. For that reason it is beneficial to share the same interface for client traffic and communication with the failover partner. That approach may have drawbacks in some network topologies.

8.5. Re-allocating Leases

When in PARTNER-DOWN state there is a waiting period after which a resource can be re-allocated to another client. For resources which are available when the server enters PARTNER-DOWN state, the period is the MCLT from the entry into PARTNER-DOWN state. For resources which are not available when the server enters PARTNER-DOWN state, the period is the MCLT after the later of the following times: the potential valid lifetime, the most recently transmitted potential valid lifetime, the most recently received acknowledged potential valid lifetime, and the most recently transmitted acknowledged potential valid lifetime. If this time would be earlier than the current time plus the MCLT, then the time the server entered PARTNER-DOWN state plus the maximum-client-lead-time is used.

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

This could be modeled in the following way. Though this specific implementation is in no way required, it may serve to better illustrate the concept.

An "available" resource on a server may be allocated to any client. A resource which was 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 resource was EXPIRED or RELEASED through a BNDUPD. When the sending server received the BNDACK for that resource showing it was FREE, it would move the resource 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 resource in the EXPIRED or RELEASED state to the same client with no restrictions provided it has not sent a BNDUPD message to its partner. This situation would exist if the lease expired or was released after the transition into PARTNER-DOWN state, for instance.

8.6. Sending Binding Update

This and the following section is written as though every BNDUPD message contains only a single binding update transaction in order to reduce the complexity of the discussion. Servers MAY generate messages with multiple binding update transactions in them, and their partner servers MAY process these messages. Before multiple binding update transactions are to be sent and processed over a failover connection, their use MUST be negotiated during the CONNECT and CONNECTACK connection establishment processing.

Each server updates its failover partner about recent changes in lease states. Each update MUST include at least the following information:

1. resource type - non-temporary address or a prefix. Resource type can be indicated by the container that conveys the actual resource (e.g. an IA_NA option indicates non-temporary IPv6 address);

2. resource information - the actual address or prefix. That is conveyed using the appropriate option, e.g. an IAADDR for an address or an IAPREFIX for a prefix;

3. valid life time sent to client*;

4. 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.)*;

5. Next Expected Client Transmission (renewal time) - time interval since Client Last Transmission Time, when a response from a client is expected*;

6. potential valid life time - a lifetime that the server is willing to set if there were no MCLT/failover restrictions imposed*;

7. preferred life time sent to client - the actual value sent back to the client*;

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

9. Client DUID*.

10. Resource state.

11. start time of state (especially for non-client updates).

Items marked with asterisk MUST appear only if the lease is/was associated with a client. Otherwise it MUST NOT appear.

The BNDUPD message MAY contain additional information related to the updated lease. The additional information MAY include, but is not limited to:

1. assigned FQDN name, defined in [RFC4704];

2. Options Requested by the client, i.e. content of the ORO;

3. Relay Data option from DHCPv6 Leasequery, see [RFC5007] Section 4.1.2.4

4. Any other options the updating partner deems useful.

The receiving partner MAY store any additional information received, but it MAY choose to ignore it as well. Some information may be useful, so it is a good idea to keep or update it. One reason is FQDN information. A server SHOULD be prepared to clean up DNS information once the lease expires or is released. See Section 11 for a detailed discussion about Dynamic DNS. Another reason the partner may be interested in keeping additional data is a better support for leasequery [RFC5007] or bulk leasequery [RFC5460], which features queries based on Relay-ID, by link address and by Remote-ID.

8.7. Receiving Binding Update

When a server receives a BNDUPD message, it needs to decide how to process the binding update transaction it contains and whether that transaction represents a conflict of any sort. The conflict resolution process MUST be used on the receipt of every BNDUPD message, not just those that are received while in POTENTIAL-CONFLICT state, in order to increase the robustness of the protocol.

There are three sorts of conflicts:

1. Two clients, one resource - This is the duplicate resource allocation conflict. There two different clients each allocated the same resource. See Section 8.8.

2. Two resources, one client conflict - This conflict exists when a client on one server is associated with a one resource, and on the other server with a different resource in the same or related prefix. This does not refer to the case where a single client has resources in multiple different prefixes or administrative

domains (i.e. a mobile client that changed its location), but rather the case where on the same prefix the client has a lease on one IP address in one server and on a different IP address on the other server.

This conflict may or may not be a problem for a given DHCP server implementation and policy. If implementations and policies allow, both resources can be assigned to a given client. In the event that a DHCP server requires that a DHCP client have only one outstanding lease of a given type, the conflict MUST be resolved by accepting the lease which has the latest CLTT.

It should be further clarified that DHCPv6 protocol makes assignments based on a (client DUID, resource type, IAID) triplet. The possibility of using different IAIDs was omitted in this paragraph for clarity. If one client is assigned multiple resources of the same type, but with different IAIDs, there is no conflict. Also, IAID values for different resource types are orthogonal, i.e. an IA_NA with IAID=1 is different than an IA_PD with IAID=1 and there is no conflict.

3. binding-status conflict - This is normal conflict, where one server is updating the other with newer information. See Section 8.8 for details of how to resolve these conflicts.

4. configuration conflict -- This kind of conflict stems from a differing configuration on one server than on the other server. It may be transient (last until both servers can process a new configuration) or it may be chronic. It cannot be resolved by communications over the failover connection, but must be resolved (if it is not transient) by administrator action to resolve the conflicts.

8.8. Conflict Resolution

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

When analyzing a BNDUPD message from a partner server, if there is insufficient information in the BNDUPD to process it, then reject the BNDUPD with reject-reason "Missing binding information".

If the resource in the BNDUPD is not a resource associated with the failover endpoint which received the BNDUPD message, then reject it with reject-reason "Illegal IP address or prefix (not part of any address or prefix pool)".

Every BNDUPD message SHOULD contain a client-last-transaction-time option, which 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 on an IP address expired. If there has been no interaction with the DHCP client in question (or there is no DHCP client presently associated with this resource), then there will be no client-last-transaction-time option in the BNDUPD message.

The list in Figure 3 presents the conflict resolution outcome. To "accept" a BNDUPD means to update the server's bindings database with the information contained in the BNDUDP and once the update is complete, send a BNDACK message corresponding to the BNDUPD message. To "reject" a BNDUPD means to leave the server's binding database unchanged and to respond to the BNDUPD with BNDACK with a reject-reason option included.

When interpreting the information in the following table (Figure 3), for those rules that are listed with "time" -- if a BNDUPD doesn't have a client-last-transaction-time value, then it MUST NOT be considered later than the client-last-transaction-time in the receiving server's binding. If the BNDUPD contains a client-last-transaction-time value and the receiving server's binding does not, then the client-last-transaction-time value in the BNDUPD MUST be considered later than the server's.

	binding-status in received BNDUPD.				
binding-status in receiving server	ACTIVE	EXPIRED	RELEASED	FREE FREE_BACKUP	RESET ABANDONED
ACTIVE	accept(5)	time(2)	time(1)	time(2)	accept
EXPIRED	time(1)	accept	accept	accept	accept
RELEASED	time(1)	time(1)	accept	accept	accept
FREE/FREE_BACKUP	accept	accept	accept	accept	accept
RESET	time(3)	accept	accept	accept	accept
ABANDONED	reject(4)	reject(4)	reject(4)	reject(4)	accept

Figure 3: Conflict Resolution

time(1): If the client-last-transaction-time in the BNDUPD is later than the client-last-transaction-time in the receiving server's binding, accept it, else reject it.

time(2): If the current time is later than the receiving server's lease-expiration-time, accept it, else reject it.

time(3): If the client-last-transaction-time in the BNDUPD is later than the start-time-of-state in the receiving server's binding, accept it, else reject it.

(1,2,3): If rejecting, use reject reason "Outdated binding information".

(4): Use reject reason "Less critical binding information".

(5): If the clients in a BNDUPD message and in a receiving server's binding differ, then if the receiving server is a secondary accept it, else reject it with a reject reason of "Fatal conflict exists: address in use by other client".

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

8.9. Acknowledging Reception

Upon acceptance of a binding lease, the server MUST notify its partner that it updated its database. A server MUST NOT send the BNDACK before its database is updated. A BNDACK MUST contain at lease the minimum set of information required to unambiguously identify the BNDUPD that triggered the BNDACK.

9. Endpoint States

9.1. State Machine Operation

Each server (or, more accurately, failover endpoint) can take on a variety of failover states. These states play a crucial role in determining the actions that a server will perform when processing a request from a DHCPv6 client as well as dealing with changing external conditions (e.g., loss of connection to a failover partner).

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

- Responsiveness -- the server is either responsive to DHCPv6 client requests or it is not.

- 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 message.

- 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:

- Current failover state.

- Communications status (OK or not OK).

- Partner’s failover state (if known).

Whenever any of the above state variables changes 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 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.

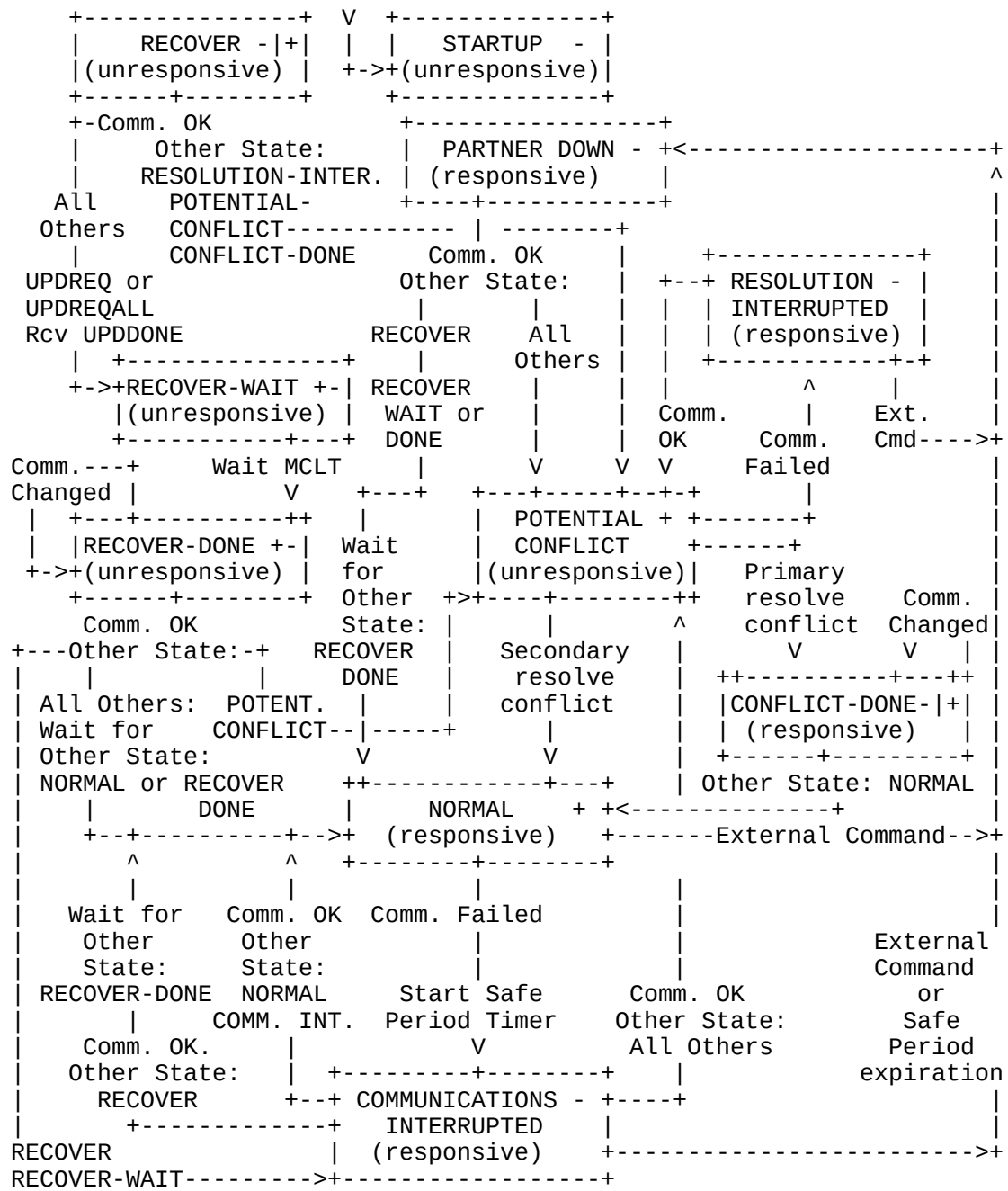


Figure 4: Failover Endpoint State Machine

9.2. State Machine Initialization

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

- Current failover state.
- Previous failover state.
- Start time of current failover state.
- Partner’s failover state.
- Start time of partner’s failover state.
- Time most recent packet 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:

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

9.3. STARTUP State

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

The STARTUP state is not shown with any specific state transitions in the state machine diagram (Figure 4) because the processing during the STARTUP state can cause the server to transition to any of the other states, so that specific state transition arcs would only obscure other information.

9.3.1. Operation in STARTUP State

The server MUST NOT be responsive to DHCPv6 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.

9.3.2. Transition Out of STARTUP State

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

Step 1:

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

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

In some cases, an existing server will be commissioned as a failover server and brought back into operation where its partner is not yet available. In this case, the newly commissioned failover server will not operate until its partner comes online -- but it has operational responsibilities as a DHCP server nonetheless. To properly handle this situation, a server SHOULD be configurable in such a way as to move directly into PARTNER-DOWN state after the startup period expires if it has been unable to contact its partner during the startup period.

Step 2:

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 communications 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:

Attempt to create a TCP connection to the failover partner.

Step 5:

Wait for "communications OK".

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

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

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

Step 6:

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

9.4. PARTNER-DOWN State

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

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.

9.4.1. 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 on resources. For those resources for which the server is using proportional allocation, it will allocate resources 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 IP addresses from the set of all available pools. Server SHOULD fully deplete its own pool, before starting allocations from its downed partner's pool.

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

A server in PARTNER-DOWN state MUST NOT allocate a resource 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 potential-expiration-time, most recently received ack of potential-expiration-time from the partner, and most recently acked potential-expiration-time to the partner. If this time would be earlier than the current time plus the maximum-client-lead-time, then the time the server entered PARTNER-DOWN state plus the maximum-client-lead-time is used.

The server is not restricted by the MCLT when offering lease times 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 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.

9.4.2. Transition Out of PARTNER-DOWN State

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

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

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

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

- If the partner is in: [RECOVER, RECOVER-WAIT] state stay in PARTNER-DOWN state

- If the partner is in: [RECOVER-DONE] state transition into NORMAL state

9.5. RECOVER State

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

9.5.1. Operation in RECOVER State

The server MUST NOT be responsive in RECOVER state.

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

9.5.2. Transition Out of RECOVER State

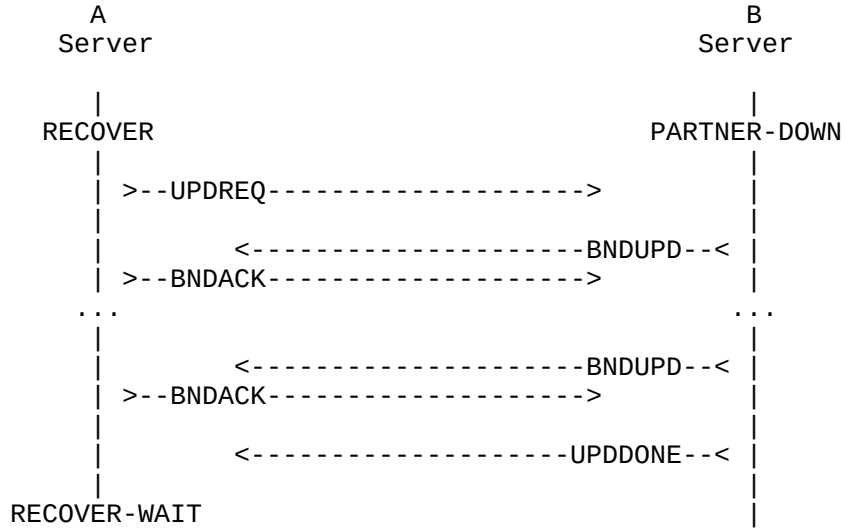
If the other server is in POTENTIAL-CONFLICT, RESOLUTION-INTERRUPTED, or CONFLICT-DONE state when communications are reestablished, then the server in RECOVER state will move to POTENTIAL-CONFLICT state itself.

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

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

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

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



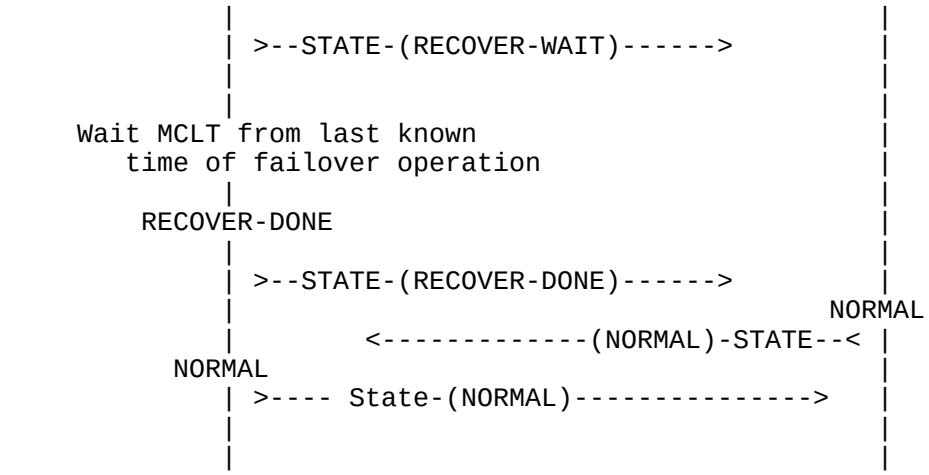


Figure 5: Transition out of RECOVER state

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

9.6. RECOVER-WAIT State

This state indicates that the server has 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.

9.6.1. Operation in RECOVER-WAIT State

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

9.6.2. Transition Out of RECOVER-WAIT State

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

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

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

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

If communications fails while a server is in RECOVER-WAIT state, it has no effect on the operation of this state. The server SHOULD continue to operate its timer, and 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 the timer.

9.7. RECOVER-DONE State

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

9.7.1. Operation in RECOVER-DONE State

A server in RECOVER-DONE state SHOULD be unresponsive, but MAY respond to RENEW requests but MUST only change the state of resources that appear in the RENEW request. It MUST NOT allocate any additional resources when in RECOVER-DONE state.

9.7.2. Transition Out of RECOVER-DONE State

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

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

9.8. NORMAL State

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

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

When the above process is complete, if the server entering NORMAL state is a secondary server, then it will request resources (addresses and/or prefixes) for allocation using the POOLREQ message.

9.8.1. Operation in NORMAL State

Primary server is responsive in NORMAL state. Secondary is unresponsive in NORMAL state.

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

Lease time calculations
: As discussed in Section 8.3, the lease interval given to a DHCP client can never be more than the MCLT greater than the most recently received potential-expiration-time from the failover partner or the current time, whichever is later.

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

Lazy update of partner server
: After sending 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.

Reallocation of resources between clients
: Whenever a client binding is released or expires, a BNDUPD message must be sent to the partner, setting the binding state to RELEASED or EXPIRED. However, until a BNDACK is received for this message, the resource cannot be allocated to another client. It cannot be allocated to the same client again if a BNDUPD was sent, otherwise it can. See Section 8.5 for details.

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

9.8.2. Transition Out of NORMAL State

If an external command is received by a server in NORMAL state informing it that its partner is down, then transition into PARTNER-DOWN state. Generally, this would be an unusual situation, where some external agency knew the partner server was down 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 8.4), 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.

9.9. COMMUNICATIONS-INTERRUPTED State

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

When a server enters COMMUNICATIONS-INTERRUPTED state, if it has been configured to support an automatic transition out of COMMUNICATIONS-INTERRUPTED state and into PARTNER-DOWN state (i.e., a auto-partner-down has been configured), then a timer MUST be 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.

9.9.1. 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 resources, and the secondary MUST allocate only FREE_BACKUP resources. When responding to RENEW messages, each server will allow continued renewal of a DHCP client's current lease on a resource irrespective of whether that lease was given out by the receiving server or not, although the renewal period MUST NOT exceed the maximum client lead time (MCLT) beyond the latest of: 1) the potential valid lifetime already acknowledged by the other server, or 2) now, or 3) the potential valid lifetime received from the partner server.

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

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

9.9.2. Transition Out of COMMUNICATIONS-INTERRUPTED State

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

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

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

- 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.

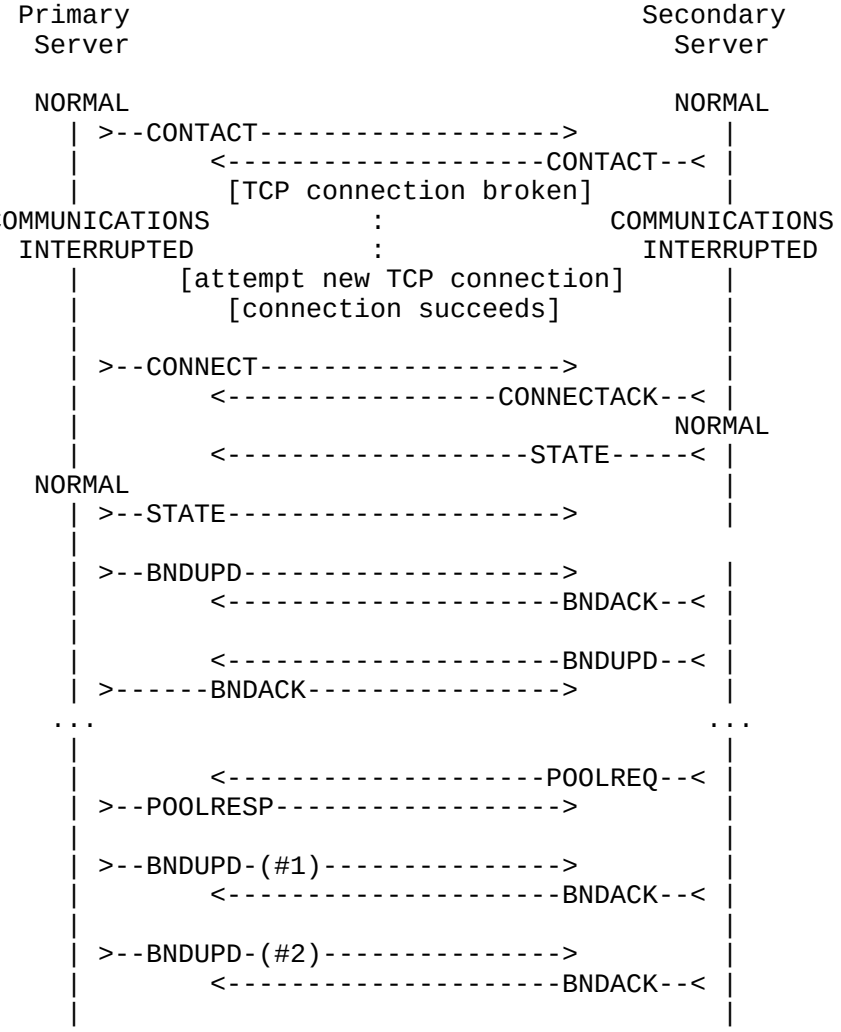


Figure 6: Transition from NORMAL to COMMUNICATIONS-INTERRUPTED and back (example with 2 addresses allocated to secondary)

9.10. POTENTIAL-CONFLICT State

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

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

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

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

9.10.1. Operation in POTENTIAL-CONFLICT State

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

9.10.2. Transition Out of POTENTIAL-CONFLICT State

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

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

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

Primary Server	Secondary Server

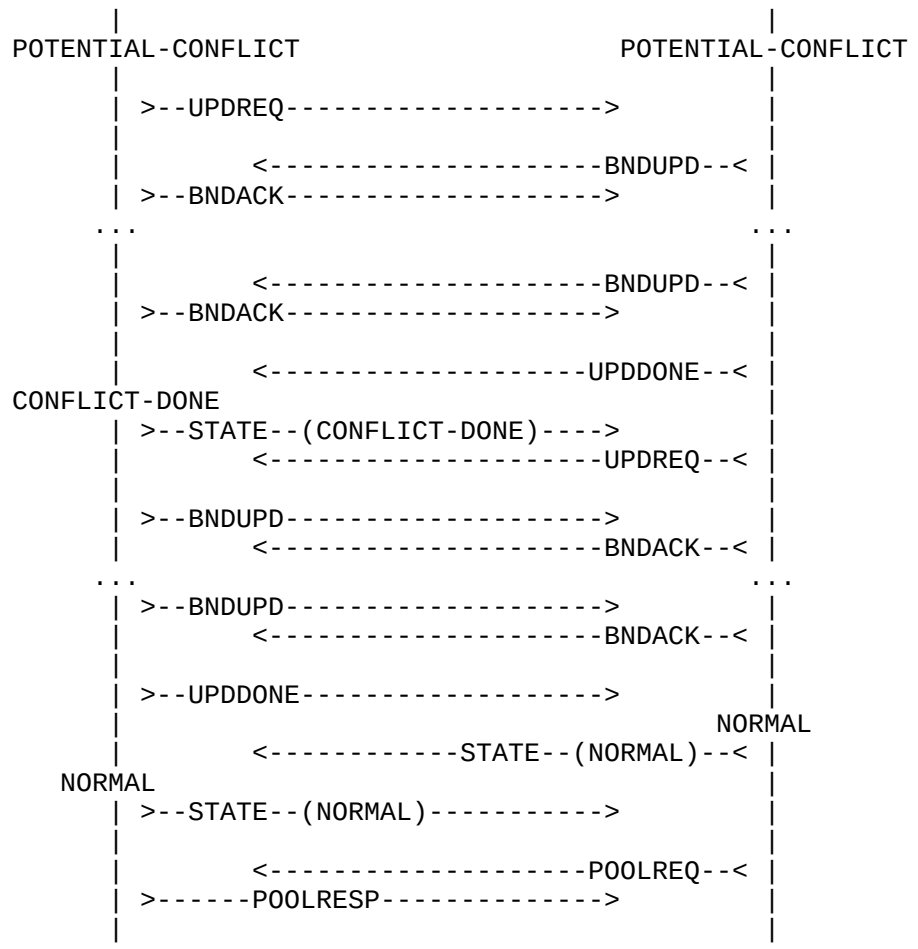


Figure 7: Transition out of POTENTIAL-CONFLICT

9.11. RESOLUTION-INTERRUPTED State

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

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 service available. 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.

9.11.1. Operation in RESOLUTION-INTERRUPTED State

In this state a server MUST respond to all DHCP client requests. When allocating new resources, each server SHOULD allocate from its own pool (if that can be determined), where the primary SHOULD allocate only FREE resources, and the secondary SHOULD allocate only FREE_BACKUP resources. 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 potential valid lifetime already acknowledged by the other server or 2) now or 3) potential valid lifetime received from the partner server.

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

9.11.2. Transition Out of RESOLUTION-INTERRUPTED State

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

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

9.12. CONFLICT-DONE State

This state indicates that during the process where the two servers are attempting to re-integrate with each other, the primary server has received all of the updates from the secondary server. It 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 will be more apparent when load balancing extension will be defined.

9.12.1. Operation in CONFLICT-DONE State

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

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

9.12.2. Transition Out of CONFLICT-DONE State

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

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

10. Proposed extensions

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

10.1. Active-active mode

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

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

11. Dynamic DNS Considerations

DHCP servers (and clients) can use DNS Dynamic 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 dynamic DNS (DDNS) 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 DDNS updates.

The purpose of this discussion is to clarify the areas where the failover and DHCP-DDNS protocols intersect for the benefit of implementations which support both protocols, not to introduce a new requirement into the DHCPv6 failover protocol. Thus, a DHCPv6 server which implements the failover protocol MAY also support dynamic DNS updates, but if it does support dynamic 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 DHCPv6 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 DDNS protocol to update a DNS server should not be a partner with a server which is not utilizing the DDNS protocol to update a DNS server. However, a server which is not able to support DDNS or is not configured to support DDNS SHOULD output a warning message when it receives BNDUPD messages which indicate that its failover partner is configured to support the DDNS protocol to update a DNS server. An implementation MAY consider this an error and refuse to operate, or it MAY choose to operate anyway, having warned the administrator of the problem in some way.

11.1. Relationship between failover and dynamic 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 DDNS 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 DDNS behavior for DHCP failover servers is similar to that described above for the failover protocol itself:

1. Allow timely DDNS updates from the server which grants a lease to a client. Recognize that there is often a DDNS 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 leased resource 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 DDNS update 'lifecycle' even if the other server originally granted the lease.

3. Avoid redundant or overlapping DDNS updates, where both failover servers are attempting to perform DDNS 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 DHCP servers configured for DDNS typically perform these operations on both the AAAA and the PTR resource records, this is not required. It is entirely possible that a DHCP 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.

11.2. Exchanging DDNS Information

In order for either server to be able to complete a DDNS 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 DDNS 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 resource. 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 DDNS 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 resources 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 DDNS update related to the lease.

The partner server receiving BNDUPD messages containing the DDNS information SHOULD compare the status information and the FQDN with the current DDNS information it has associated with the lease binding, and update its notion of the DDNS status accordingly.

Some implementations will instead choose to send a BNDUPD without waiting for the DDNS update to complete, and then will send a second BNDUPD once the DDNS update is complete. Other implementations will delay sending the partner a BNDUPD until the DDNS 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 IP 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 DDNS 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 DDNS 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 DHCP requests.

11.3. Adding RRs to the DNS

A failover server which is going to perform DDNS updates SHOULD initiate the DDNS update when it grants a new lease to a client. The server which did not grant the lease SHOULD NOT initiate a DDNS 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 DDNS updates when it renews leases. In two cases, however, a failover server MAY initiate a DDNS update when it renews a lease to its existing client:

1. When the lease was granted before the server was configured to perform DDNS updates, the server MAY be configured to perform updates when it next renews existing leases. The server which granted the lease is the server which should initiate the DDNS update.

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 DDNS 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.

11.4. Deleting RRs from the DNS

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

A server not in PARTNER-DOWN state "makes a resource 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 BNDACK the server has "made the resource FREE". Conversely, a server in PARTNER-DOWN state "makes a resource 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 DDNS operations to delete RRs from the DDNS. Its partner SHOULD NOT initiate DDNS deletes for DNS records related to the lease binding as part of sending the BNDACK 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 resource FREE*. The server making the resource 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 DDNS deletes for RRs which its partner added originally. This allows a single remaining partner server to assume responsibility for all of the DDNS activity which the two servers were undertaking.

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

11.5. Name Assignment with No Update of DNS

In some cases, a DHCP server is configured to return a name to the DHCPv6 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 DHCPv6 client begins to interact.

12. Reservations and failover

Some DHCP servers support a capability to offer specific preconfigured resources to DHCP clients. These are real DHCP clients, they do the entire DHCP protocol, but these servers always offer the client a specific pre-configured resource, and they offer that resource to no other clients. Such a capability has several names, but it is sometimes called a "reservation", in that the resource is reserved for a particular DHCP client.

In a situation where there are two DHCP servers serving the same prefix without using failover, the two DHCP server's need to have disjoint resource pools, but identical reservations for the DHCP clients.

In a failover context, both servers need to be configured with the proper reservations in an identical manner, but if we stop there problems can occur around the edge conditions where reservations are made for resource that has already been leased to a different client. Different servers handle this conflict in different ways, but the goal of the failover protocol is to allow correct operation with any server's approach to the normal processing of the DHCP protocol.

The general solution with regards to reservations is as follows. Whenever a reserved resource becomes FREE (i.e., when first configured or whenever a client frees it or it expires or is reset), the primary server MUST show that resource as FREE (and thus available for its own allocation) and it MUST send it to the secondary server in a BNDUPD with a flag set showing that it is reserved and with a status of FREE_BACKUP.

Note that this implies that a reserved resource goes through the normal state changes from FREE to ACTIVE (and possibly back to FREE). The failover protocol supports this approach to reservations, i.e., where the resource undergoes the normal state changes of any resource, but it can only be offered to the client for which it is reserved.

From the above, it follows that a reservation solely on the secondary will not necessarily allow the secondary to offer that address to client to whom it is reserved. The reservation must also appear on the primary as well for the secondary to be able to offer the resource to the client to which it is reserved.

When the reservation on a resource is cancelled, if the resource is currently FREE and the server is the primary, or FREE_BACKUP and the server is the secondary, the server MUST send a BNDUPD to the other server with the binding-status FREE and an indication that the resource is no longer reserved.

13. 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.

As traffic exchange between clients and server is not encrypted, an attacker that penetrated the network and is able to intercept traffic, will not gain any additional information by also sniffing communication between partners.

An attacker that is able to impersonate one partner can efficiently perform a denial of service attack on the remaining uncompromised server. Several techniques may be used: pretending that conflict resolution is required, requesting rebalance, claiming that a valid lease was released or declined etc. For that reason the communication between servers SHOULD support failover connections over TLS, as explained in Section 5.1. Such secure connections SHOULD be optional and configurable by the administrator.

A server MUST NOT operate in PARTNER-DOWN if its partner is up. Network administrators are expected to switch the remaining active server to PARTNER-DOWN state only if they is sure that its partner server is indeed down. Failing to obey this requirement will result in both servers likely assigning duplicate leases to different clients. Implementers should take that into consideration if they decide to implement the auto-partner-down timer-based transition to PARTNER-DOWN state.

Running a network protected by DHCPv6 failover requires more resources than running without it. In particular some of the resources are allocated to the secondary server and they are not usable in a normal (i.e. non failures) operation immediately, though over time they will be rebalanced and end up on the server that needs them. While limiting this pool may be preferable from resource utilization perspective, it must be a reasonably large pool, so the secondary may take over once the primary becomes unavailable.

14. IANA Considerations

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

15. Acknowledgements

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

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

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

16. References

16.1. Normative References

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

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

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

[RFC4703] Stapp, M. and B. Volz, "Resolution of Fully Qualified Domain Name (FQDN) Conflicts among Dynamic Host Configuration Protocol (DHCP) Clients", RFC 4703, October 2006.

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

[RFC5007] Brzozowski, J., Kinnear, K., Volz, B., and S. Zeng, "DHCPv6 Leasequery", RFC 5007, September 2007.

16.2. Informative References

[I-D.ietf-dhc-dhcpv6-failover-requirements]
Mrugalski, T. and K. Kinnear, "DHCPv6 Failover Requirements", draft-ietf-dhc-dhcpv6-failover-requirements-07 (work in progress), July 2013.

[I-D.ietf-dhc-dhcpv6-load-balancing]
Kostur, A., "DHC Load Balancing Algorithm for DHCPv6", draft-ietf-dhc-dhcpv6-load-balancing-00 (work in progress), December 2012.

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

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

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

Authors' Addresses

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

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

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

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