Dynamic Host Configuration (DHC)
Internet-Draft
Intended status: Informational
Expires: January 5, 2013

T. Mrugalski
ISC
K. Kinnear
Cisco

July 4, 2012

DHCPv6 Failover Requirements
draft-ietf-dhc-dhcpv6-failover-requirements-01

Abstract

The DHCPv6 protocol, defined in [RFC3315] allows for multiple servers to operate on a single network, however it does not define any way to decide which server responds to which client queries. Some sites are interested in running multiple servers in such a way as to provide increased availability in case of server failure. In order for this to work reliably, the cooperating primary and secondary servers must maintain a consistent database of the lease information. [RFC3315] allows for but does not define any redundancy or failover mechanisms. This document outlines requirements for DHCPv6 failover, enumerates related problems, and discusses the proposed scope of work to be conducted. This document does not define a DHCPv6 failover protocol.

Status of this Memo

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

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

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

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

Copyright Notice

Table of Contents

1. Requirements Language

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

2. Introduction

The DHCPv6 protocol, defined in [RFC3315] allows for multiple servers to be operating on a single network, however it does not define any way to decide which server responds to which client queries. Some sites are interested in running multiple servers in such a way as to provide redundancy in case of server failure. In order for this to work reliably, the cooperating primary and secondary servers must maintain a consistent database of the lease information.

This document discusses failover implementations scenarios, failure modes, and synchronization approaches to provide background to the list of requirements for a DHCPv6 failover protocol. It then defines a minimum set of requirements that failover must provide to be useful, while acknowledging that additional features may be specified as extensions. This document does not define a DHCPv6 failover protocol.

3. Definitions

This section defines terms that are relevant to DHCPv6 failover.

Definitions from [RFC3315] are included by reference. In particular, client means any device (e.g., end user host, CPE or other router) that implements client functionality of the DHCPv6 protocol. A server means a DHCPv6 server, unless explicitly noted otherwise. A relay is a DHCPv6 relay.

A binding (or, client binding) is a group of server data records containing the information the server has about the addresses in an IA or configuration information explicitly assigned to the client. Configuration information that has been returned to a client through a policy - for example, the information returned to all clients on the same link - does not require a binding.

DDNS - an abbreviation for "Dynamic DNS", which refers to the capability to update a DNS server's name database using the on-the-wire protocol defined in [RFC2136]. Clients and servers can negotiate the scope of such updates as defined in [RFC4704].

Failover - an ability of one partner to continue offering services provided by another partner, with minimal or no impact on clients.

FQDN - a fully qualified domain name. A fully qualified domain name generally is a host name with at least one zone name. For example "dhcp.example.org" is a fully qualified domain name.

High Availability - a desired property of DHCPv6 servers to continue providing services despite experiencing unwanted events such as server crashes, link failures, or network partitions.

Load Balancing - the ability for two or more servers to each process some portion of the client request traffic in a conflict-free fashion.

Lease - an IPv6 address, an IPv6 prefix or other resource that was assigned ('leased') by a server to a specific client. A lease may include additional information, like associated fully qualified domain name (FQDN) and/or information about associated DNS updates.

Partner - A "partner", for the purpose of this document, refers to

a failover server, typically the other failover server in a failover relationship.

Stable Storage - each DHCP server is required to keep its lease database in some form of storage (known as "stable storage") that will be consistent throughout reboots, crashes and power failures.

Partner Failure - A power outage, unexpected shutdown, crash or other type of failure that renders a partner unable to continue its operation.

4. Scope of work

In order to fit within the IETF process effectively and efficiently, the standardization effort for DHCPv6 failover is expected to proceed with the creation of documents of increasing specificity. It begins with this document specifying the requirements for DHCPv6 failover ("requirements document"). Later documents are expected to address the design of the DHCPv6 failover protocol ("design document"), and if sufficient interest exists, the protocol details required to implement the DHCPv6 failover protocol itself ("protocol document"). The goal of this partitioning is, in part, to ease the validation, review, and approval of the DHCPv6 failover protocol by presenting it in comprehensible parts to the larger community.

Additional documents describing extensions may also be defined.

DHCPv6 Failover requirements are presented in Section 8.

4.1. Alternatives to Failover

There are many scenarios when it seems that a failover capability would be useful. However, there are often much simpler approaches that will meet the required goals. This section documents examples where failover is not really needed.

4.1.1. Short-lived addresses

There are cases when IPv6 addresses are used only for a short time, but there is a need to have high degree of confidence that those

addresses will be served. A notable example is a PXE scenario where hosts require an address during boot. Address and possibly other configuration parameters are used during boot process and are discarded afterwards. Any lack of available DHCPv6 service at this time may render the devices unbootable.

Instead of deploying failover, it is better to use the much simpler preference mechanism, defined in [RFC3315]. For example, consider two or more servers with each having distinct preference set (e.g. 10 and 20). Both will answer to a client's request. The client should choose the one with larger preference value. In case of failure of the most preferred server, the next server will keep responding to clients' queries. This approach is simple to deploy, but does not offer lease stability, i.e. in case of server failure, clients' addresses and prefixes will change.

4.1.2. Redundant servers

4.1.3. Distributted databases

5. Failover Scenarios

The following section provides several examples of deployment scenarios and use cases that may be associated with capabilities commonly referred to as failover. These scenarios may be inside or outside of scope for DHCPv6 failover protocol as defined by this document. They are enumerated here to provide a common basis for discussion.

5.1. Hot Standby Model

In the simplest case, there are two partners that are connected to the same network. Only one of partners ('primary') provides services to clients. In case of its failure, the second partner ('secondary') continues handling services previously handled by first partner. As both servers are connected to the same network, a partner that fails to communicate with its partner while also receiving requests from

clients may assume with high probability that its partner is down and the network is functional. This assumption may affect its operation.

5.2. Geographically Distributed Failover

Servers may be physically located in separate locations. A common example of such a topology is where a service provider has at least a regional high performance network between geographically distributed datacenters. In such a scenario, one server is located in one datacenter and its failover partner is located in another remote datacenter. In this scenario, when one partner finds that it cannot communicate with the other partner, it does not necessarily mean that the other partner is down.

5.3. Load balancing

A desire to have more than one server in a network may also be created by the desire to have incoming traffic be handled by several servers. This decreases the load each server must endure when all servers are operational. Although such a capability does not, strictly, require failover - it is clear that failover makes such an architecture more straightforward.

Note that in a load balancing situation which includes failover, each individual server MUST be able to handle the full load normally handled by both servers working together, or there is not a true increase in availability.

5.4. 1-to-1, m-to-1 and m-to-m models

A failover relationship for a specific network is provided by two failover partners. Those partners communicate with each other. This scenario is sometimes referred to as the 1-to-1 model and is considered relatively simple. In larger networks one server may be participating in several failover relationships, i.e. it provides failover for several address or prefix pools, each served by separate partners. Such a scenario can be referred to as m-to-1. The most complex scenario - m-to-m - assumes that each partner participates in multiple failover relationships.

5.5. Split prefixes

Due to the extensive IPv6 address space, it is possible to provide semi-redundant service by splitting the available pool of addressees into two or more non-overlapping pools, with each server handling its own smaller pool. Several versions of such a scenario are discussed in [I-D.ietf-dhc-dhcpv6-redundancy-consider].

5.6. Long lived connections

Certain nodes may maintain long lived connections. Since the IPv6 address space is large, techniques exist (e.g. [I-D.ietf-dhc-dhcpv6-redundancy-consider]) that use the easy availability of IPv6 addresses in order to provide increased DHCPv6 availability. However, these approaches do not generally provide for stable IPv6 addresses for DHCPv6 clients should the server with which the client is interacting become unavailable.

5.7. Partial server communication loss

There is a scenario where the DHCPv6 server may be configured to serve clients on one network adapter and communicate with partner server (server-2-server traffic) on a different network adapter. In this scenario, if the server loses connectivity on the network adapter used to communicate with the clients because of network adapter (hardware) failure, there is no intimation of the loss of service to the partner in the DHCPv6 failover protocol. Since the servers are able to communicate with each other, the partner remains ignorant of the loss of service to clients.

6. Principles of DHCPv6 Failover

This section describes important issues that will affect any DHCPv6 failover protocol. This section is not intended to define implementation details, but rather high level concepts and issues that are important to DHCPv6 failover. These issues form a basis for later documents which deal with the solutions to these issues.

6.1. Failure modes

This section documents failure modes. Each failure mode is listed as either an in-scope or out-of-scope requirement for the failover protocol.

6.1.1. Server Failure

Servers may become unresponsive due to a software crash, hardware failure, power outage or any number of other reasons. The failover partner will detect such event due to lack of responses from the down partner. In this failure mode, the assumption is that the server is the only equipment that is off-line and all other network equipment is operating normally. In particular, communication between other nodes is not interrupted.

When working under the assumption that this is the only type of failure that can happen, the server may safely assume that its partner unreachability means that it is down, so other nodes (clients in particular) are not able to reach it either and no services are provided.

It should be noted that recovery after the failed server is brought back on-line is straightforward, due to the fact that it just needs to download current information from the lease database of the healthy partner and there is no conflict resolution required.

This is by far the most common failure mode between two failover partners.

When the two servers are located physically close to each other, possibly in the same room, the probability that a failure to communicate between failover partners is due to server failure is increased.

6.1.2. Network partition

Another possible cause of partner unreachability is a failure in the network that connects the two servers. This may be caused by failure of any kind of network equipment: router, switch, physical cables, or optic fibers. As a result of such a failure the network is split into two or more disjoint sections (partitions) that are not able to communicate with each other. Such an event is called network partition. If failover partners are located in different partitions, they won't be able to communicate with each other. Nevertheless, each partner may still be able to serve clients that happen to be part of the same partition.

If this failure mode is taken into consideration, a server can't assume that partner unreachability automatically means that its partner is down. They must consider the fact that the partner may continue operating and interacting with a subset of the clients. The

only valid assumption is that partner also detected the network partition event and follows procedures specified for such a

situation.

It should be noted that recovery after partitioned network is rejoined is significantly more complicated than recovery from a server failure event. As both servers may have kept serving clients, they have two separate lease databases, and they need to agree on the state of each lease (or follow any other algorithm to bring their lease databases into agreement).

This failure mode is more likely (though still rare) in the situation where two servers are in physically distant locations with multiple network elements between them. This is the case in geographically distributed failover (see Section 5.2).

6.2. Synchronization mechanisms

Partners must exchange information about changes made to the lease database. There are two types of sychronization methods that may be used.

6.2.1. Lockstep

When a server receives a REQUEST message from a client it consults its lease database and assigns requested addresses and/or prefixes. To make sure that its partner maintains a consistent database, it then sends information about new or just updated lease to the partner and waits for the partner's response. After the response from partner is received the REPLY message is transmitted to the client.

This approach has the benefit of having a completely consistent lease database between partners at all times. Unfortunately, there is a large performance penalty for this approach as each response sent to a client is delayed by the total sum of the delays caused by two transmissions between partners and the processing by the second partner. The second partner is expected to update its own copy of the lease database in permanent storage, so this delay is not negligible, even in fast networks.

6.2.2. Lazy updates

Another approach to synchronizing the lease databases is to transmit the REPLY message to the client before completing the update to the partner. The server sends the REPLY to the client immediately after assigning appropriate addresses and/or prefixes and initiates the partner update at a later time, depending on the algorithm chosen. Another varation of this approach is to initiate transmission to the partner, but not wait for its response before sending the REPLY to the client.

This approach has benefit of a minimal impact on server response times, thus it is much better from a performance perspective. However, it makes the lease databases loosely synchronized between partners. This makes the synchronization more complex (and particularly the re-integration after a network partition event), as there may be cases where one client has been given a lease on an address or prefix of which the partner is not aware (e.g. if server crashes after sending REPLY to the client, but before sending update information to its partner).

7. DHCPv4 and DHCPv6 Failover Comparison

There are significant similarities between existing DHCPv4 and envisaged DHCPv6 failovers. In particular both serve IP addresses to clients, require maintaining consistent databases among partners, need to perform consistent DNS Updates, must be able take over bindings offered by failed partner, must be able to resume operation after partner is recovered. DNS conflict resolution works on the same principles in both DHCPv4 and DHCPv6.

Nevertheless, there are significant differences. IPv6 introduced prefix delegation [RFC3633] that is a crucial element of the DHCPv6 protocol. IPv6 also introduced the concept of deprecated addresses with separate prefered and valid lifetimes, both being configured via DHCPv6. Negative response (NACK) in DHCPv4 has been replaced with the ability in DHCPv6 to provide corrected response in a REPLY message that differs from a REQUEST.

Also, the typical large address space (close to 2^64 addresses on /64 prefixes expected to be available on most networks) may make managing address assignment significantly different from DHCPv4 failover. In

DHCPv4 it was not possible to use a hash or calculated technique to divide the significantly more limited address space and therefore much of the protocol that deals with pool balancing and rebalancing might not be necessary and can be done mathematically. And, it may also be possible and reasonable to use a much longer MCLT value -- as reusing an address for a different client is generally not a requirement (at least over the near term) as close to 2^64 addresses may be available.

However, DHCPv6 Prefix Delegation is similar to IPv4 addressing and therefore techniques for pool balancing and rebalancing and shorter MCLT times will be needed.

8. DHCPv6 Failover Requirements

This section summarizes the requirements for DHCPv6 failover.

Certain capabilities may be useful in some, but not all scenarios. Such additional features will be considered optional parts of failover, and will be split and defined in separate documents. As such, this document can be considered an attempt to define requirements for the DHCPv6 failover 'core' protocol.

The core of the DHCPv6 failover protocol is expected to provide the following properties:

1. The number of supported partners MUST be exactly two, i.e. there are at most two servers that are aware of specific lease; this approach is often called 1-to-1 model.

2. For each prefix or address pool, server MUST NOT participate in more than one failover relationship.

3. Server MAY participate in multiple relationships only if those relationships cover different prefix or address pools.

4. One partner MUST be able to continue serving leases offered by the other partner. This property is also sometimes called

'lease stability'. The failure of either failover partner SHOULD pose minimal or no impact on client connectivity. In particular, it MUST NOT force the client to change addresses and/or change prefixes delegated to it. Long-lived connections MUST NOT be disturbed.

5. Prefix delegation MUST be supported.

6. Use of the failover protocol MUST NOT introduce significant performance impact on server response times. Therefore synchronization between partner MUST be done using some form of lazy updates (see Section 6.2.2).

7. The pair of failover servers MUST be able to recover from server down failure (see Section 6.1.1).

8. The pair of failover servers MUST be able to recover from a network partition event (see Section 6.1.2).

9. The design MUST allow secure communication.

10. The definition of extensions to this core protocol SHOULD be allowed, when possible.

High Availability is a property of the protocol that allows clients to receive DHCPv6 services despite the failure of individual DHCPv6 servers. In particular, it means the server that takes over providing service to clients from its failed partner, will continue serving the same addresses and/or prefixes. This property is also sometime called lease stability.

The core protocol MUST be limited to the 1-to-1 model (see Section 5.4). In addition, the core protocol MUST restrict each address or prefix pool to participate in at most one failover relationship. (Note: these are different statements!) If there is sufficient community support for failover servers to participate in more than one failover relationship (thus providing support for a form of m-to-1 deployment), this capability SHALL be defined as an extension to the core failover protocol.

Despite the lack of standardization of DHCPv4 failover, the coexistence of DHCPv4 and DHCPv6 failover MAY be taken into

consideration. In particular, certain features that are common for both IPv4 and IPv6, like DNS Update mechanism SHOULD be taken into consideration.

8.1. Features out of scope

The following features are explictly out of scope.

1. Load Balancing - a capability is considered an extension and MAY be defined in a separate document. It MUST NOT be part of the core protocol, but rather defined as an extension. The primary reason for this the desire to limit core protocol complexity.

2. Configuration synchronization - two failover partners are expected to maintain the same configuration. Mismatched configuration between partners is a frequent problem in failover solutions. Unfortunately, that is an open-ended problem, since different servers have very different config data models.

3. m-to-m model (see section Section 5.4)

4. servers participating in multiple failover relationships for any given pool.

9. Related work

This section describes related work. Readers may benefit from familiarizing themselves with these approaches, their advantages and limitations.

9.1. DHCPv4 failover concepts

9.1.1. Goals of DHCPv4 Failover

1. Provide a high availability DHCP service by leveraging the hooks built into DHCPv4 [RFC2131] and its usual implementation to support multiple servers able to respond to client requests. These hooks are:

   (a) The broadcast of DHCPDISCOVER requests.

(b) The transition from unicast for DHCPREQUEST/RENEW to broadcast for DHCPREQUEST/REBIND.

(c) The usual implementation of DHCPv4 relay agents to allow forwarding of DHCPv4 requests to multiple different DHCPv4 servers.

2. Produce a minimal impact on performance of the DHCPv4 server.

3. Prevent duplicate IP address allocation even in the event of a network partition.

4. Allow multiple failover relationships per server.

5. Standardize only the minimum necessary to provide a high availability DHCP service. In particular, avoid standardizing the interchange of configuration information.

9.1.2. Goals lead to Concepts

The goal to have a minimal performance impact on the operation of the DHCPv4 servers participating in failover is the driving force behind the design of the DHCPv4 failover protocol.

The steps in this chain of reasoning are as follows:

1. To avoid the major performance impact that a lockstep update of a failover partner would inflict, use a lazy update approach (see Section 6.2.2).

2. When using lazy update, there is always the problem that the failover server could crash after it has responded to some number of DHCPv4 clients and before it has updated its partner with the lease information it provided to those clients.

3. Thus, when one failover server cannot communicate with another failover server, it cannot know what that other failover server

has told any of its DHCPv4 clients.

This creates an obvious problem.

The central concept in the DHCPv4 failover design is to place a limit on the uncertainty described in point #3, above. The DHCPv4 failover protocol is designed to ensure that every failover server knows at all times, not exactly what its failover partner has told to the DHCPv4 clients with which it is communicating, but rather the limits of what its failover partner could have told any DHCPv4 clients with which it was communicating.

This is done by ensuring that no DHCPv4 server participating in a failover relationship will ever offer a lease time to any DHCPv4 client that is more than an agreed-upon value beyond that known by its failover partner.

This agreed-upon value is called the "Maximum Client Lead Time", and abbreviated MCLT.

Thus, every DHCPv4 failover partner needs to know what its partner knows about every lease in the server, and it needs to ensure that it will never provide a lease time to any DHCPv4 client that is beyond what its partner believes the current lease time to be plus the MCLT.

Given this fundamental guarantee, if one failover server cannot communicate with its failover partner, then it knows the limits of what any DHCPv4 client of that missing partner might have for a lease time. If this failover server waits until it believes a lease has expired and then also waits until the MCLT has passed, it knows that the lease is sure to have expired (or the DHCPv4 client will have tried to renew the lease and communicated with the remaining DHCPv4 server). (We will deal with network partition events below.)

In order to allow a remaining failover server to provide service to newly arrived DHCPv4 clients, while waiting out the MCLT beyond the lease expiration (if any), the protocol provides for allocation of some percentage of the available leases to each failover partner.

A failover server which cannot commuicate with its partner must therefore wait out the MCLT beyond the lease expiration (if any) of IP addresses before it can allocate them to DHCPv4 clients. This could impact the server's ability to provide available IP addresses to newly arrived DHCPv4 clients. To prevent this impact the DHCPv4 failover protocol divides the allocation of the available leases between each failover partner. The protocol supports periodic rebalancing of the allocation of these available leases.

9.1.3. Use of the MCLT in practice

From the above discussion, it should be clear how to avoid re-using an IP address before it has expired. The MCLT is central to the operation of the protocol. One server cannot offer a lease to a DHCPv4 client that is more than the MCLT beyond the current lease time for this client that is known by the failover partner. From this standpoint, it would be good for the MCLT to be as long as possible. However, in a failure situation, waiting the MCLT beyond the current lease time in order to reuse a leased lease would suggest that the MCLT should be as short as possible.

This tension is resolved by anticipating the need to extend lease times when communicating with the failover partner. The first lease offered to a DHCPv4 client can be only as long as the MCLT. However, when the failover server updates its partner, it updates the partner with the desired lease time plus the MCLT. Thus, when the client returns with a renewal request at halfway through the MCLT, the failover server can extend its lease for only the lease time known by the partner plus the MCLT. But the partner now knows the desired lease time, so that the server can extend the lease for as long as it was configured since it had pre-updated the failover partner with this time.

Using this approach, one can keep the MCLT relatively short, say 1 hour, and still offer leases of any desired extent to clients -- once the failover partner has been updated with the desired lease time.

9.1.4. Network Partition Events

It is clear that when one failover server finds that it is unable to communicate with its failover partner, it is impossible for that server to tell if its failover partner is down or if the communication path to that failover partner is broken, a situation known as "network partition" (see Section 6.1.2). The DHCPv4 failover protocol distinguishes between these different situations by having different failover states to represent "communications-interrupted" situations and a "partner-down" situations. The expectation is that (at least in some situations) it requires an operator action to distinguish between a communications-interrupted and partner-down event. In particular, the DHCPv4 failover protocol does not conflate these two situations.

Correct handling of network partition events requires that a failover server unable to communicate with its failover partner (but not yet informed that its failover partner is down), must not re-allocate an IP address from one DHCPv4 client to another. Available addresses

may be allocated to any DHCPv4 client.

After a failover server has been informed that its partner is down, it can reallocate an IP address from one DHCPv4 client to another once it has waited the MCLT beyond the lease expiration of that IP address. This need to be informed by an external entity that the failover partner is down is the only impact of correctly handling network partition events. Of course, specific implementations can assume that an unreachable failover partner is down after a shorter or longer time, thus limiting the support for a network partition event.

9.1.5. Conflict Resolution

Whenever one failover server receives an update from its failover partner, it needs to decide if the update it has received is "better" than the information it has in its own database concerning the DHCPv4 client or the lease on the IPv4 address. The DHCPv4 failover protocol does not mandate the details of this decision, but this activity must be part of any DHCPv4 implementation. In most cases, comparing the times associated with the failover update with the times held in the server's own database will allow this decision to be made.

9.1.6. Load Balancing

The DHCPv4 Load Balancing protocol [RFC3074] integrates with the DHCPv4 failover protocol by defining the way that each server decides which DHCPv4 clients to process. Use of load balancing with the DHCPv4 failover protocol is an optional extension to the failover protocol. Both a simple active -- passive relationship without load balancing is defined as well as a more complex active -- active relationship.

9.2. DHCPv6 Redundancy Considerations

[I-D.ietf-dhc-dhcpv6-redundancy-consider] specifies an interim architecture to provide a semi-redundant DHCPv6 solution before the availability of vendor or standard based solutions. The proposed architecture may be used in wide range of networks. Two notable deployment models are discussed: service provider and enterprise network environments. The described architecture leverages only

existing and implemented DHCPv6 standards.

10. Security Considerations

TBD...

11. IANA Considerations

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

12. Acknowledgements

This document extensively uses concepts, definitions and other parts of [dhcpv4-failover] document. Authors would like to thank Shawn Routhier, Qin Wu, Jean-Francois Tremblay, Frank Sweetser, Jiang Sheng, Yu Fu, Greg Rabil, Bernie Volz, and Vithalprasad Gaitonde for their comments and feedback.

This work has been partially supported by 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).

13. References

13.1. Normative References

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

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.

[RFC3074] Volz, B., Gonczi, S., Lemon, T., and R. Stevens, "DHC Load Balancing Algorithm", RFC 3074, February 2001.

[RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C.,

and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.

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

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

13.2. Informative References

[I-D.ietf-dhc-dhcpv6-redundancy-consider]
Tremblay, J., Brzozowski, J., Chen, J., and T. Mrugalski,

"DHCPv6 Redundancy Deployment Considerations", draft-ietf-dhc-dhcpv6-redundancy-consider-02 (work in progress), October 2011.

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

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

Authors' Addresses

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

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

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

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