DHC WG
Internet-Draft
Updates: 2131 (if approved)
Intended status: Standards Track
Expires: August 17, 2014

Y. Cui
Q. Sun
Tsinghua University
I. Farrer
Deutsche Telekom AG
Y. Lee
Comcast
Q. Sun
China Telecom
M. Boucadair
France Telecom

February 13, 2014

Dynamic Allocation of Shared IPv4 Addresses
draft-csf-dhc-dynamic-shared-v4allocation-00

Abstract

This memo describes the dynamic allocation of shared IPv4 addresses to clients using the DHCPv4 protocol. Address sharing allows a single IPv4 address to be allocated to multiple, active clients simultaneously, each client being differentiated by a unique set of L4 source ports. The changes necessary to existing DHCPv4 client and server behaviour are described and a new DHCPv4 option for provisioning clients with shared IPv4 addresses is included.

Due to the nature of sharing IP addresses, there are necessarily some limitations to the applicability. This memo describes those limitations and recommends suitable architectures and technologies where address sharing may be utilized.

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].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Shortages of available public IPv4 addresses mean that it is not always possible for operators to allocate a full IPv4 address to every customer. This problem may be particularly acute whilst the operator is in the migration phase from a native IPv4 network to a native IPv6 network with IPv4 provided as an overlay service. This is likely to increase the requirement on public IPv4 addresses to provide for both existing and transition networks.

Two main types of solution have emerged to ease the problem:

1. Centralised Network Address Translation (NAT44) in the core network
2. Distributing the same public IPv4 address to multiple clients using non-overlapping layer 4 port sets.

The solution described in this memo is only suitable on the second solution.

[I-D.ietf-dhc-dhcpv4-over-dhcpv6] introduces a "DHCP 4o6 Server", which is capable of servicing both DHCPv6 [RFC3315] and DHCPv4-over-DHCPv6 requests. This enables the provisioning of DHCPv4 based configuration to IPv6 connected clients over IPv6 only transport networks.

One of the benefits of the DHCPv4-over-DHCPv6 based approach is that it allows the dynamic leasing of IPv4 addresses to clients, based on existing mechanisms for address lease management available in DHCPv4 servers. This can make much more efficient use of remaining public IPv4 addresses than static pre-allocation based approaches as only IPv4 clients that are currently active need to be allocated addresses. This memo uses the defined OPTION_PORTPARAMSV4 with DHCPv4 over DHCPv6, achieving the dynamic leasing of the shared IPv4 addresses.

Due to the nature of address sharing in this manner, it is only suitable for specific architectures based on the Address plus Port Model (A+P) [RFC6346]. This model extends the unique identifier for a client from the 32-bit IPv4 address to 48-bits by including the 16-bits of the layer 4 header. Each client is allocated a unique block of layer 4 ports, and the client will generally utilize these restricted source ports by implementing a NAPT44 funtion, translating traffic from the original private IPv4 source address and unrestricted port to the allocated shared IPv4 address and unique restricted port range. [I-D.ietf-softwire-map] and [I-D.ietf-softwire-lw4over6] describe two implemented examples of the

A+P approach which may be suitable for shared, dynamic IPv4 addressing.

The use of shared addressing in other, more traditional deployment architectures must be avoided due to the fundamental incompatibilities of assigning a the same /32 IPv4 address to multiple clients attached to the same layer 2 segment.

This memo also defines OPTION_PORTPARAMSV4, a DHCPv4 option for assigning non-overlapping layer 4 port sets during the IPv4 address allocation process.

Although DHCPv4 over DHCPv6 is used as the underlying DHCPv4 transport mechanism throughout this document, OPTION_PORTPARAMSV4 may also be used in DHCPv4 over IPv6 [I-D.ietf-dhc-dhcpv4-over-ipv6] and other DHCPv4 IPv4 address allocation mechanisms. The usage of OPTION_PORTPARAMSV4 in those cases is out of scope of this document.

2. Functional Overview

Functionally, the dynamic allocation of shared IPv4 addresses by the DHCP 4o6 Server is quite similar to the normal DHCPv4 server dynamic allocation process described in [RFC2131]. The essential difference is that the DHCP 4o6 Server MAY allocate the same IPv4 address to more than one DHCP 4o6 client simultaneously, providing that each address allocation also includes a range of layer 4 source ports unique to that address (i.e. each PSID may only be allocated once per /32 address).

To enable this, the DHCP 4o6 client needs to be extended to implement OPTION_PORTPARAMSV4 (described below). This option is used to indicate to the DHCP 4o6 server the client's support the dynamic allocation of a shared IPv4 address and also for conveying the allocated PSID back to the client.

The server must be extended to implement OPTION_PORTPARAMSV4 so that it can identify clients supporting shared, dynamic address leasing. With this option, the server can dynamically maintain shared IPv4 address leases. The server must also manage unique client leases based on the IPv4 address and PSID tuple, instead of just IPv4 address.

3. Client-Server Interaction

Section 3 of [RFC2131] describes client-server interactions necessary for leasing addresses. The following sections describe the changes necessary for the client and server to implement the dynamic allocation of a shared IPv4 address.

3.1. Allocating a Shared, Dynamic IPv4 Address

Section 3.1 of [RFC2131] describes the client-server interaction for allocating an IPv4 address. The process described below detail the changes necessary for the allocation of a shared IPv4 address.

Using DHCP 4o6, the following DHCPv4 message flow is transported within the DHCPV4-QUERY and DHCPV4-RESPONSE options, which are DHCPv6 options used for carrying DHCPv4 messages.

1. When the client constructs its DHCPv4 DHCPDISCOVER message to be transported within the DHCPv4-query message, the DHCPDISCOVER message MUST include the following options: A client Identifier (constructed as per [RFC4361] and OPTION_PORTPARAMSV4 (described below). The client MAY insert a non-zero value in the PSID-Len field within OPTION_PORTPARAMSV4 to indicate the preferred size of the restricted port range allocation to the DHCP 4o6 Server.
2. Each DHCP 4o6 Server that receives the DHCPDISCOVER message within the DHCPv4-query message responds with a DHCPOFFER message that contains an available IPv4 address in the ’yiaddr’ field. The response MUST also include OPTION_PORTPARAMSV4 containing a restricted port-range. If the received OPTION_PORTPARAMSV4 field contains a non-zero PSID-Len field, the DHCP 4o6 Server MAY allocate a port set of the requested size to the client (depending on policy). The DHCPOFFER message is included into the DHCPv4-response message and sent to the client.
3. The client evaluates all received DHCPOFFER messages and selects one based on the configuration parameters received, such as the size of the offered port set. The client then sends a DHCPREQUEST containing a server identifier and the corresponding OPTION_PORTPARAMSV4 received in the DHCPOFFER message.
4. The server identified in the DHCPREQUEST message (via the siaddr field) creates a binding for the client. The binding includes the client identifier, the IPv4 address and the PSID. These parameters are used by both the server and the client to identify a lease referred to in any DHCP messages. The server responds with a DHCPACK message containing the configuration parameters for the requesting client. Optionally, the the server may also store the IPv6 address that the client has bound the received IPv4 paramters to.
5. The client receives the DHCPACK message with the configuration parameters. The client MUST NOT perform a final check on the address, such as ARPing for a duplicate allocated address.
6. If the client chooses to relinquish its lease by sending a DHCPRELEASE message, the client MUST include the original client identifier, the leased network address and the allocated restricted source ports inlcuded in OPTION_PORTPARAMSV4.

3.2. Reusing a Previously Allocated Shared, Dynamic IPv4 address

If the client remembers the previously allocated address and restricted port range, then the process described in section 3.2 of [RFC2131] must be followed. OPTION_PORTPARAMSV4 MUST be included in the message flow, with the client’s requested port set being included in the DHCPDISCOVER message.

4. Server Behavior

The DHCP 4o6 Server MUST NOT reply with the OPTION_PORTPARAMSV4 until the client has explicitly listed the option code in the Parameter Request List (Option 55) [RFC2132].

The DHCP 4o6 Server SHOULD reply with OPTION_PORTPARAMSV4 if the client includes the option in its Parameter Request List. In order to achieve the dynamic management of IPv4 address and port set in the address sharing environment, the server MUST run an address and port-set pool that plays the same role as address pool in a regular DHCP server. The server MUST use the combination of address and PSID as the key to maintain the state of a lease, and look for an available lease for assignment. The leasing database MUST include the information of the address and PSID.

When a server receives a DHCPDISCOVER message with OPTION_PORTPARAMSV4 in the Parameter Request List from a client, the server chooses an IPv4 address and a port-set for the requesting client. The logic of choosing is similar to that in Section 4.3.1 of [RFC2131]. The difference is the server looks for the client’s binding or an available lease in the server’s pool of addresses and PSIDs. After selecting an available IPv4 address with a PSID, the server sends a DHCPOFFER message to the requesting client.

When the server receives a DHCPREQUEST message with OPTION_PORTPARAMSV4, the server MUST determine the client’s state according to related parameters (Section 4.3.2 of [RFC2131]) and the value of OPTION_PORTPARAMSV4.

Upon reception of a DHCPRELEASE message with OPTION_PORTPARAMSV4, the server looks for the lease using the address in the message and the PSID value in the OPTION_PORTPARAMSV4, and marks it as unallocated.

The port-set assignment MUST be coupled with the address assignment process. Therefore server MUST assign the address and port set in the same DHCP messages. The lease information for the address is applicable to the port-set as well.

4.1. Leasing Shared and Non-Shared IPv4 Addresses from a Single DHCP 4o6 Server

A single DHCP 4o6 server may have clients that do not support OPTION_PORTPARAMS as well as those that do. As the rules for the allocation of shared addresses differ from the rules for full IPv4 address assigment, the DHCP 4o6 server MUST implement a mechanism to ensure that clients which do not support OPTION_PORTPARAMS do not receive shared addresses. For example two separate IPv4 addressing pools could be used, one of which allocates IPv4 addresses and PSIDs only to clients which have requested them.

5. Client Behavior

The DHCP client applying for a port-set MUST include the OPTION_PORTPARAMSV4 code in the Parameter Request List (Option 55). The client retrieves a port set using the value contained in OPTION_PORTPARAMSV4.

When the client renews or releases the DHCP lease, it MUST put the values of offset, PSID length and PSID into the OPTION_PORTPARAMSV4, and send to the server within corresponding DHCPv4 messages.

In the DHCPDISCOVER message, the client MAY use a non-zero value for the PSID-len field within OPTION_PORTPARMAS. This is used by the client to request a specific size of port-set (i.e. the number of source ports that it will be allocated).

5.1. Client Usage of a Shared Address

As a single IPv4 address is being shared between a number of different clients, the allocated shared address is only suitable for certain uses. The client MUST implement a function to ensure that only the allocated layer 4 ports of the shared IPv4 address are used for sourcing new connections.

The client MUST apply the following rules for any traffic to or from the shared /32 IPv4 address:

- Only port-aware protocols or ICMP implementing [RFC5508] MUST be used
- All connections originating from the shared IPv4 address MUST use a source port taken from the allocated restricted port range.
- The client MUST NOT accept inbound connections on ports outside of the allocated restricted port range.

In order to prevent addressing conflicts which could arise from the allocation of the same IPv4 addreses, the client MUST NOT configure the received restricted IPv4 address on-link.

The mechanism by which a client implements these rules is outside of the scope of this document.

In the event that the DHCPv4 over DHCPv6 configuration mechanism fails for any reason, the client MUST NOT configure an IPv4 link-local address [RFC3927](taken from the 169.254.0.0/16 range).

6. Additional Changes to RFC 2131

In addtion to the changes mentioned elsewhere in this document, the following changes to the behaviour described in [RFC2131] are necessary in order to implement dynamic allocation of a shared IPv4 address.

Section 2.2 The client MUST NOT probe a newly received IPv4 address (e.g. with ARP) to see if it is in use by another host.
Section 3.1 Item 5. The client MUST NOT perform a final check on the assigned IPv4 address.

7. DHCPv4 Port Parameters Option

The Port Paramaters Option for DHCPv4 specifies the restricted set of layer 4 source ports that are necessary to dynamically allocate a shared address. The option uses the same fields as the MAP Port Parameters Option described in Section 4.4 of [I-D.ietf-softwire-map-dhcp], implemented as a DHCPv4 option. This is to maintain compatibility with existing implementations.

The construction and usage of OPTION_PORTPARAMSV4 is

```
    0                   1
                           2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | option-code   |  Length       |    offset     |   PSID-Len    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |              PSID             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: DHCPv4 Port Parameters Option

o option-code: OPTION_PORTPARAMSV4 (TBA)
o option-length: 3

- offset: (PSID offset) 8 bits long field that specifies the numeric value for the MAP algorithm’s excluded port range/offset bits (A-bits), as per section 5.1.1 in [I-D.ietf-softwire-map]. Allowed values are between 0 and 16, with the default value being 4 for a MAP client. This parameter is unused by a Lightweight 4over6 client and should be set to 0.
- PSID-len: Bit length value of the number of significant bits in the PSID field (also known as ’k’). When set to 0, the PSID field is to be ignored. After the first ’a’ bits, there are k bits in the port number representing valid of PSID. Subsequently, the address sharing ratio would be 2^k.
- PSID: Explicit 16-bit (unsigned word) PSID value. The PSID value algorithmically identifies a set of ports assigned to a CE. The first k-bits on the left of this 2-octets field is the PSID value. The remaining (16-k) bits on the right are padding zeros.

[I-D.ietf-softwire-map] (Section 5.1) provides a full description of how the PSID is interpreted by the client.

When receiveing the Port Parameters option with an explicit PSID, the client MUST use this explicit PSID in configuring its DHCPv4 over DHCPv6 interface.

8. Security Consideration

8.1. Denial-of-Service

The solution is generally vulnerable to DoS when used on a shared medium or when access network authentication is not a prerequisite to IP address assignment. The solution SHOULD only be used on point-to-point links, tunnels, and/or in environments where authentication at link layer is performed before IP address assignment, and not shared medium.

8.2. Port Randomization

Preserving port randomization [RFC6056] may be more or less difficult depending on the address sharing ratio (i.e., the size of the port space assigned to a CPE). The host can only randomize the ports inside a fixed port range [RFC6269].

More discussion to improve the robustness of TCP against Blind In-Window Attacks can be found at [RFC5961]. Other means than the (IPv4) source port randomization to provide protection against attacks should be used (e.g., use [I-D.vixie-dnsext-dns0x20] to protect against DNS attacks, [RFC5961] to improve the robustness of TCP against Blind In-Window Attacks, use IPv6).

A proposal to preserve the entropy when selecting port is discussed in [I-D.bajko-pripaddrassign].

9. IANA Considerations

IANA is kindly requested to allocate the following DHCPv4 option code: TBD for OPTION_PORTPARAMSV4.

10. Acknowledgements

This document is merged from [I-D.sun-dhc-port-set-option] and [I-D.farrer-dhc-shared-address-lease].

11. References

11.1. Normative References

[I-D.ietf-dhc-dhcpv4-over-dhcpv6]
Sun, Q., Cui, Y., Siodelski, M., Krishnan, S., and I. Farrer, "DHCPv4 over DHCPv6 Transport", draft-ietf-dhc-dhcpv4-over-dhcpv6-04 (work in progress), January 2014.

[I-D.ietf-softwire-map]
Troan, O., Dec, W., Li, X., Bao, C., Matsushima, S., Murakami, T., and T. Taylor, "Mapping of Address and Port with Encapsulation (MAP)", draft-ietf-softwire-map-10 (work in progress), January 2014.

[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.

[RFC2132] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor Extensions", RFC 2132, March 1997.

[RFC4361] Lemon, T. and B. Sommerfeld, "Node-specific Client Identifiers for Dynamic Host Configuration Protocol Version Four (DHCPv4)", RFC 4361, February 2006.

[RFC5961] Ramaiah, A., Stewart, R., and M. Dalal, "Improving TCP's Robustness to Blind In-Window Attacks", RFC 5961, August 2010.

[RFC6056] Larsen, M. and F. Gont, "Recommendations for Transport-Protocol Port Randomization", BCP 156, RFC 6056, January 2011.

[RFC6269] Ford, M., Boucadair, M., Durand, A., Levis, P., and P. Roberts, "Issues with IP Address Sharing", RFC 6269, June 2011.

11.2. Informative References

[I-D.bajko-pripaddrassign]
Bajko, G., Savolainen, T., Boucadair, M., and P. Levis, "Port Restricted IP Address Assignment", draft-bajko-pripaddrassign-04 (work in progress), April 2012.

[I-D.farrer-dhc-shared-address-lease]
Farrer, I., "Dynamic Allocation of Shared IPv4 Addresses using DHCPv4 over DHCPv6", draft-farrer-dhc-shared-address-lease-00 (work in progress), June 2013.

[I-D.ietf-dhc-dhcpv4-over-ipv6]
Cui, Y., Wu, P., Wu, J., Lemon, T., and Q. Sun, "DHCPv4 over IPv6 Transport", draft-ietf-dhc-dhcpv4-over-ipv6-08 (work in progress), October 2013.

[I-D.ietf-softwire-lw4over6]
Cui, Y., Qiong, Q., Boucadair, M., Tsou, T., Lee, Y., and I. Farrer, "Lightweight 4over6: An Extension to the DS-Lite Architecture", draft-ietf-softwire-lw4over6-06 (work in progress), February 2014.

[I-D.ietf-softwire-map-dhcp]
Mrugalski, T., Troan, O., Dec, W., Bao, C., leaf.yeh.sdo@gmail.com, l., and X. Deng, "DHCPv6 Options for configuration of Softwire Address and Port Mapped Clients", draft-ietf-softwire-map-dhcp-06 (work in progress), November 2013.

[I-D.sun-dhc-port-set-option]
Qiong, Q., Lee, Y., Sun, Q., Bajko, G., and M. Boucadair, "Dynamic Host Configuration Protocol (DHCP) Option for Port Set Assignment", draft-sun-dhc-port-set-option-02 (work in progress), October 2013.

[I-D.vixie-dnsext-dns0x20]
Vixie, P. and D. Dagon, "Use of Bit 0x20 in DNS Labels to Improve Transaction Identity", draft-vixie-dnsext-dns0x20-00 (work in progress), March 2008.

[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.

[RFC3927] Cheshire, S., Aboba, B., and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses", RFC 3927, May 2005.

[RFC5508] Srisuresh, P., Ford, B., Sivakumar, S., and S. Guha, "NAT Behavioral Requirements for ICMP", BCP 148, RFC 5508, April 2009.

[RFC6346] Bush, R., "The Address plus Port (A+P) Approach to the IPv4 Address Shortage", RFC 6346, August 2011.

Authors' Addresses

Yong Cui
Tsinghua University
Beijing 100084
P.R.China

Phone: +86-10-6260-3059
Email: yong@csnet1.cs.tsinghua.edu.cn

Qi Sun
Tsinghua University
Beijing 100084
P.R.China

Phone: +86-10-6278-5822
Email: sunqi@csnet1.cs.tsinghua.edu.cn

Ian Farrer
Deutsche Telekom AG
CTO-ATI, Landgrabenweg 151
Bonn, NRW 53227
Germany

Email: ian.farrer@telekom.de

Yiu L. Lee
Comcast
One Comcast Center
Philadelphia PA 19103
USA

Email: yiu_lee@cable.comcast.com

Qiong Sun
China Telecom
Room 708, No.118, Xizhimennei Street
Beijing 100035
P.R.China

Phone: +86-10-58552936
Email: sunqiong@ctbri.com.cn

Mohamed Boucadair
France Telecom
2330 Central Expressway
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Dynamic Host Configuration (DHC)
Internet-Draft
Obsoletes: 3315,3633,3736,7083 (if approved)
Intended status: Standards Track
Expires: August 26, 2015

T. Mrugalski, Ed.
M. Siodelski
ISC
B. Volz, Ed.
A. Yourtchenko
Cisco
M. Richardson
SSW
S. Jiang
Huawei
T. Lemon
Nominum

February 22, 2015

Dynamic Host Configuration Protocol for IPv6 (DHCPv6) bis
draft-dhcwg-dhc-rfc3315bis-04

Abstract

The Dynamic Host Configuration Protocol for IPv6 (DHCP) enables DHCP servers to pass configuration parameters such as IPv6 network addresses to IPv6 nodes. It offers the capability of automatic allocation of reusable network addresses and additional configuration flexibility. This protocol is a stateful counterpart to "IPv6 Stateless Address Autoconfiguration" (RFC 4862), and can be used separately or concurrently with the latter to obtain configuration parameters.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction and Overview

This document describes DHCP for IPv6 (DHCP), a client/server protocol that provides managed configuration of devices.

DHCP can provide a device with addresses assigned by a DHCP server and other configuration information, which are carried in options. DHCP can be extended through the definition of new options to carry configuration information not specified in this document.

DHCP is the "stateful address autoconfiguration protocol" and the "stateful autoconfiguration protocol" referred to in "IPv6 Stateless Address Autoconfiguration" [RFC4862].

This document also provides a mechanism for automated delegation of IPv6 prefixes using DHCP. Through this mechanism, a delegating router can delegate prefixes to requesting routers.

The operational models and relevant configuration information for DHCPv4 [RFC2132][RFC2131] and DHCPv6 are sufficiently different that integration between the two services is not included in this document. [RFC3315] suggested that future work might be to extend DHCPv6 to carry IPv4 address and configuration information. However, the current consensus of the IETF is that DHCPv4 should be used rather than DHCPv6 when conveying IPv4 configuration information to nodes. [RFC7341] describes a transport mechanism to carry DHCPv4 messages using the DHCPv6 protocol for the dynamic provisioning of IPv4 address and configuration information across IPv6-only networks.

The remainder of this introduction summarizes DHCP, explaining the message exchange mechanisms and example message flows. The message flows in Section 1.2 and Section 1.3 are intended as illustrations of DHCP operation rather than an exhaustive list of all possible client-server interactions. Section 5 provides an overview of common operational models. Section 18, Section 19, and Section 20 explain client and server operation in detail.

1.1. Protocols and Addressing

Clients and servers exchange DHCP messages using UDP [RFC0768]. The client uses a link-local address or addresses determined through other mechanisms for transmitting and receiving DHCP messages.

A DHCP client sends most messages using a reserved, link-scoped multicast destination address so that the client need not be configured with the address or addresses of DHCP servers.

To allow a DHCP client to send a message to a DHCP server that is not attached to the same link, a DHCP relay agent on the client’s link will relay messages between the client and server. The operation of the relay agent is transparent to the client and the discussion of message exchanges in the remainder of this section will omit the description of message relaying by relay agents.

Once the client has determined the address of a server, it may under some circumstances send messages directly to the server using unicast.

1.2. Client-server Exchanges Involving Two Messages

When a DHCP client does not need to have a DHCP server assign it IP addresses, the client can obtain configuration information such as a list of available DNS servers [RFC3646] or NTP servers [RFC4075] through a single message and reply exchanged with a DHCP server. To obtain configuration information the client first sends an Information-request message to the All_DHCP_Relay_Agents_and_Servers multicast address. Servers respond with a Reply message containing the configuration information for the client.

This message exchange assumes that the client requires only configuration information and does not require the assignment of any IPv6 addresses.

When a server has IPv6 addresses and other configuration information committed to a client, the client and server may be able to complete the exchange using only two messages, instead of four messages as described in the next section. In this case, the client sends a Solicit message to the All_DHCP_Relay_Agents_and_Servers requesting the assignment of addresses and other configuration information. This message includes an indication that the client is willing to accept an immediate Reply message from the server. The server that is willing to commit the assignment of addresses to the client immediately responds with a Reply message. The configuration information and the addresses in the Reply message are then immediately available for use by the client.

Each address assigned to the client has associated preferred and valid lifetimes specified by the server. To request an extension of the lifetimes assigned to an address, the client sends a Renew message to the server. The server sends a Reply message to the client with the new lifetimes, allowing the client to continue to use the address without interruption.

1.3. Client-server Exchanges Involving Four Messages

To request the assignment of one or more IPv6 addresses, a client first locates a DHCP server and then requests the assignment of addresses and other configuration information from the server. The client sends a Solicit message to the All_DHCP_Relay_Agents_and_Servers address to find available DHCP servers. Any server that can meet the client’s requirements responds with an Advertise message. The client then chooses one of the servers and sends a Request message to the server asking for confirmed assignment of addresses and other configuration information. The server responds with a Reply message that contains the confirmed addresses and configuration.

As described in the previous section, the client sends a Renew message to the server to extend the lifetimes associated with its addresses, allowing the client to continue to use those addresses without interruption.

2. Requirements

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in [RFC2119].

This document also makes use of internal conceptual variables to describe protocol behavior and external variables that an implementation must allow system administrators to change. The specific variable names, how their values change, and how their settings influence protocol behavior are provided to demonstrate protocol behavior. An implementation is not required to have them in the exact form described here, so long as its external behavior is consistent with that described in this document.

3. Background

The IPv6 Specification provides the base architecture and design of IPv6. Related work in IPv6 that would best serve an implementor to study includes the IPv6 Specification [RFC2460], the IPv6 Addressing Architecture [RFC4291], IPv6 Stateless Address Autoconfiguration [RFC4862], IPv6 Neighbor Discovery Processing [RFC4861], and Dynamic Updates to DNS [RFC2136]. These specifications enable DHCP to build upon the IPv6 work to provide both robust stateful autoconfiguration and autoregistration of DNS Host Names.

The IPv6 Addressing Architecture specification [RFC4291] defines the address scope that can be used in an IPv6 implementation, and the various configuration architecture guidelines for network designers of the IPv6 address space. Two advantages of IPv6 are that support for multicast is required and nodes can create link-local addresses during initialization. The availability of these features means that a client can use its link-local address and a well-known multicast address to discover and communicate with DHCP servers or relay agents on its link.

IPv6 Stateless Address Autoconfiguration [RFC4862] specifies procedures by which a node may autoconfigure addresses based on router advertisements [RFC4861], and the use of a valid lifetime to support renumbering of addresses on the Internet. In addition, the protocol interaction by which a node begins stateless or stateful autoconfiguration is specified. DHCP is one vehicle to perform stateful autoconfiguration. Compatibility with stateless address autoconfiguration is a design requirement of DHCP.

IPv6 Neighbor Discovery [RFC4861] is the node discovery protocol in IPv6 which replaces and enhances functions of ARP [RFC0826]. To understand IPv6 and stateless address autoconfiguration, it is strongly recommended that implementors understand IPv6 Neighbor Discovery.

Dynamic Updates to DNS [RFC2136] is a specification that supports the dynamic update of DNS records for both IPv4 and IPv6. DHCP can use the dynamic updates to DNS to integrate addresses and name space to not only support autoconfiguration, but also autoregistration in IPv6.

4. Terminology

This section defines terminology specific to IPv6 and DHCP used in this document.

4.1. IPv6 Terminology

IPv6 terminology relevant to this specification from the IPv6 Protocol [RFC2460], IPv6 Addressing Architecture [RFC4291], and IPv6 Stateless Address Autoconfiguration [RFC4862] is included below.

address
An IP layer identifier for an interface or a set of interfaces.

host
Any node that is not a router.

IP
Internet Protocol Version 6 (IPv6). The terms IPv4 and IPv6 are used only in contexts where it is necessary to avoid ambiguity.

interface
A node’s attachment to a link.

link
A communication facility or medium over which nodes can communicate at the link layer, i.e., the layer immediately below IP. Examples are Ethernet (simple or bridged); Token Ring; PPP links, X.25, Frame Relay, or ATM networks; and Internet (or higher) layer "tunnels", such as tunnels over IPv4 or IPv6 itself.

link-layer identifier
A link-layer identifier for an interface. Examples include IEEE 802 addresses for Ethernet or Token Ring network interfaces, and E.164 addresses for ISDN links.

link-local address
An IPv6 address having a link-only scope, indicated by having the prefix (FE80::/10), that can be used to reach neighboring nodes attached to the same link. Every interface has a link-local address.

multicast address
An identifier for a set of interfaces (typically belonging to different nodes). A packet sent to a multicast address is delivered to all interfaces identified by that address.

neighbor A node attached to the same link.

node A device that implements IP.

packet An IP header plus payload.

prefix The initial bits of an address, or a set of IP addresses that share the same initial bits.

prefix length The number of bits in a prefix.

router A node that forwards IP packets not explicitly addressed to itself.

unicast address An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address.

4.2. DHCP Terminology

Terminology specific to DHCP can be found below.

allocatable resource (or resource). It is an address, a prefix or any other allocatable resource that may be defined in the future. Currently there are three defined allocatable resources: non-temporary addresses, temporary addresses and delegated prefixes.

appropriate to the link An address is "appropriate to the link" when the address is consistent with the DHCP server’s knowledge of the network topology, prefix assignment and address assignment policies.

binding 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. A binding containing information about an IA is indexed by the tuple <DUID, IA-type, IAID> (where IA-type is the type of address in the IA; for example, temporary). A binding containing configuration information for a client is indexed by <DUID>.

configuration parameter — An element of the configuration information set on the server and delivered to the client using DHCP. Such parameters may be used to carry information to be used by a node to configure its network subsystem and enable communication on a link or internetwork, for example.

delegating router: — The router that acts as a DHCP server, and is responding to the prefix request.

DHCP — Dynamic Host Configuration Protocol for IPv6. The terms DHCPv4 and DHCPv6 are used only in contexts where it is necessary to avoid ambiguity.

DHCP client (or client) — A node that initiates requests on a link to obtain configuration parameters from one or more DHCP servers. Depending on the purpose of the client, it may feature the requesting router functionality, if it supports prefix delegation.

DHCP domain — A set of links managed by DHCP and operated by a single administrative entity.

DHCP realm — A name used to identify the DHCP administrative domain from which a DHCP authentication key was selected.

DHCP relay agent (or relay agent) — A node that acts as an intermediary to deliver DHCP messages between clients and servers. In certain configurations there may be more than one relay agent between clients and servers, so a relay agent may send DHCP messages to another relay agent.

DHCP server (or server) — A node that responds to requests from clients, and may or may not be on the same link as the client(s). Depending on its capabilities, it may also feature the functionality of delegating router, if it supports prefix delegation.

DUID

A DHCP Unique IDentifier for a DHCP participant; each DHCP client and server has exactly one DUID. See Section 10 for details of the ways in which a DUID may be constructed.

IA

Identity Association: A collection of allocatable resources assigned to a client. Each IA has an associated IAID. A client may have more than one IA assigned to it; for example, one for each of its interfaces. Each IA holds one type of address; for example, an identity association for temporary addresses (IA_TA) holds temporary addresses (see "identity association for temporary addresses") and identity association for prefix delegation (IA_PD) holds delegated prefixes. Throughout this document, "IA" is used to refer to an identity association without identifying the type of allocatable resources in the IA. At the time of writing this document, there are 3 IA types defined: IA_NA, IA_TA and IA_PD. New IA types may be defined in the future.

IAID

Identity Association IDentifier: An identifier for an IA, chosen by the client. Each IA has an IAID, which is chosen to be unique among IAIDs for IAs of a specific type, belonging to that client.

IA_NA

Identity association for Non-temporary Addresses: An IA that carries assigned addresses that are not temporary addresses (see "identity association for temporary addresses")

IA_TA

Identity Association for Temporary Addresses: An IA that carries temporary addresses (see [RFC4941]).

IA_PD

Identity Association for Prefix Delegation: A collection of prefixes assigned to the requesting router. Each IA_PD has an associated IAID. A requesting router may have more than one IA_PD assigned to it; for example, one for each of its interfaces.

message — A unit of data carried as the payload of a UDP datagram, exchanged among DHCP servers, relay agents and clients.

Reconfigure key — A key supplied to a client by a server used to provide security for Reconfigure messages.

requesting router: — The router that acts as a DHCP client and is requesting prefix(es) to be assigned.

singleton option: — An option that is allowed to appear only once. Most options are singletons.

relaying — A DHCP relay agent relays DHCP messages between DHCP participants.

transaction ID — An opaque value used to match responses with replies initiated either by a client or server.

5. Operational Models

This section describes some of the current most common DHCP operational models. The described models are not mutually exclusive and are sometimes used together. For example, a device may start in stateful mode to obtain an address, and at a later time when an application is started, request additional parameters using stateless mode.

5.1. Stateless DHCP

Stateless DHCP [RFC3736] is used when DHCP is not used for obtaining an allocatable resource, but a node (DHCP client) desires one or more DHCP "other configuration" parameters, such as a list of DNS recursive name servers or DNS domain search lists [RFC3646]. Stateless may be used when a node initially boots or at any time the software on the node requires some missing or expired configuration information that is available via DHCP.

This is the simplest and most basic operation for DHCP and requires a client (and a server) to support only two messages - Information-request and Reply. Note that DHCP servers and relay agents typically also need to support the Relay-Forw and Relay-Reply messages to accommodate operation when clients and servers are not on the same link.

5.2. DHCP for Non-Temporary Address Assignment

This model of operation was the original motivation for DHCP and is the "stateful address autoconfiguration protocol" for IPv6 [RFC2462]. It is appropriate for situations where stateless address autoconfiguration is not desired, because of network policy, additional requirements (such as updating the DNS with forward or reverse resource records), or client specific requirements (i.e., some prefixes are only available to some clients) which are not possible using stateless address autoconfiguration.

The model of operation for non-temporary address assignment is as follows. The server is provided with IPv6 prefixes from which it may allocate addresses to clients, as well as any related network topology information as to which prefixes are present on which links. A client requests a non-temporary address to be assigned by the server. The server allocates an address or addresses appropriate for the link on which the client is connected. The server returns the allocated address or addresses to the client.

Each address has an associated preferred and valid lifetime, which constitutes an agreement about the length of time over which the client is allowed to use the address. A client can request an extension of the lifetimes on an address and is required to terminate the use of an address if the valid lifetime of the address expires.

Typically clients request other configuration parameters, such as the domain server addresses and search lists, when requesting addresses.

5.3. DHCP for Prefix Delegation

The prefix delegation mechanism, originally described in [RFC3633], is another stateful mode of operation and intended for simple delegation of prefixes from a delegating router (DHCP server) to requesting routers (DHCP clients). It is appropriate for situations in which the delegating router does not have knowledge about the topology of the networks to which the requesting router is attached, and the delegating router does not require other information aside from the identity of the requesting router to choose a prefix for delegation. For example, these options would be used by a service provider to assign a prefix to a Customer Premise Equipment (CPE) device acting as a router between the subscriber’s internal network and the service provider’s core network.

The design of this prefix delegation mechanism meets the requirements for prefix delegation in [RFC3769].

The model of operation for prefix delegation is as follows. A delegating router is provided IPv6 prefixes to be delegated to requesting routers. Examples of ways in which the delegating router may be provided these prefixes is given in Section 19.4. A requesting router requests prefix(es) from the delegating router, as described in Section 19.3. The delegating router chooses prefix(es) for delegation, and responds with prefix(es) to the requesting router. The requesting router is then responsible for the delegated prefix(es). For example, the requesting router might assign a subnet from a delegated prefix to one of its interfaces, and begin sending router advertisements for the prefix on that link.

Each prefix has an associated valid and preferred lifetime, which constitutes an agreement about the length of time over which the requesting router is allowed to use the prefix. A requesting router can request an extension of the lifetimes on a delegated prefix and is required to terminate the use of a delegated prefix if the valid lifetime of the prefix expires.

This prefix delegation mechanism would be appropriate for use by an ISP to delegate a prefix to a subscriber, where the delegated prefix would possibly be subnetted and assigned to the links within the subscriber’s network.

Figure 1 illustrates a network architecture in which prefix delegation could be used.

```
                      ______________________        \
                     /                      \        \
                    |    ISP core network    |        \
                     \__________ ___________/          |
                                |                      |
                        +-------+-------+              |
                        |  Aggregation  |              | ISP
                        |    device     |              | network
                        |  (delegating  |              |
                        |    router)    |              |
                        +-------+-------+              |
                                |                     /
                                |DSL to subscriber   /
                                |premises           /
                                |
                         +------+------+            \
                         |     CPE     |             \
                         | (requesting |              \
                         |   router)   |               |
                         +----+---+----+               |
                              |   |                    | Subsciber
       ---+-------------+-----+   +-----+------        | Network
          |             |               |              |
     +----+-----+ +-----+----+     +----+-----+        |
     |Subscriber| |Subscriber|     |Subscriber|       /
     |    PC    | |    PC    |     |    PC    |      /
     +----------+ +----------+     +----------+     /
```

Figure 1: Prefix Delegation Newtork

In this example, the delegating router is configured with a set of prefixes to be used for assignment to customers at the time of each customer’s first connection to the ISP service. The prefix delegation process begins when the requesting router requests configuration information through DHCP. The DHCP messages from the requesting router are received by the delegating router in the aggregation device. When the delegating router receives the request, it selects an available prefix or prefixes for delegation to the requesting router. The delegating router then returns the prefix or prefixes to the requesting router.

The requesting router subnets the delegated prefix and assigns the longer prefixes to links in the subscriber’s network. In a typical scenario based on the network shown in Figure 1, the requesting router subnets a single delegated /48 prefix into /64 prefixes and assigns one /64 prefix to each of the links in the subscriber network.

The prefix delegation options can be used in conjunction with other DHCP options carrying other configuration information to the requesting router. The requesting router may, in turn, provide DHCP service to hosts attached to the internal network. For example, the requesting router may obtain the addresses of DNS and NTP servers from the ISP delegating router, and then pass that configuration information on to the subscriber hosts through a DHCP server in the requesting router.

5.4. DHCP for Customer Edge Routers

The DHCP requirements and network architecture for Customer Edge Routers are described in [RFC7084]. This model of operation combines address assignment (see Section 5.2) and prefix delegation (see Section 5.3). In general, this model assumes that a single set of transactions between the client and server will assign or extend the client’s non-temporary addresses and delegated prefixes.

5.5. DHCP for Temporary Addresses

Temporary addresses were originally introduced to avoid privacy concerns with stateless address autoconfiguration, which based 64-bits of the address on the EUI-64 (see [RFC3041] and [RFC4941]). They were added to DHCP to provide complementary support when stateful address assignment is used.

Temporary address assignment works mostly like non-temporary address assignment (see Section 5.2), however these addresses are generally intended to be used for a short period of time and not to have their lifetimes extended, though they can be if required.

6. DHCP Constants

This section describes various program and networking constants used by DHCP.

6.1. Multicast Addresses

DHCP makes use of the following multicast addresses:

All_DHCP_Relay_Agents_and_Servers (FF02::1:2) A link-scoped multicast address used by a client to communicate with neighboring (i.e., on-link) relay agents and servers. All servers and relay agents are members of this multicast group.

All_DHCP_Servers (FF05::1:3) A site-scoped multicast address used by a relay agent to communicate with servers, either because the relay agent wants to send messages to all servers or because it does not know the unicast addresses of the servers. Note that in order for a relay agent to use this address, it must have an address of sufficient scope to be reachable by the servers. All servers within the site are members of this multicast group.

6.2. UDP Ports

Clients listen for DHCP messages on UDP port 546. Servers and relay agents listen for DHCP messages on UDP port 547.

6.3. DHCP Message Types

DHCP defines the following message types. More detail on these message types can be found in Section 7 and Section 8. Message types not listed here are reserved for future use. The numeric encoding for each message type is shown in parentheses.

SOLICIT (1) A client sends a Solicit message to locate servers.

ADVERTISE (2) A server sends an Advertise message to indicate that it is available for DHCP service, in response to a Solicit message received from a client.

REQUEST (3) A client sends a Request message to request configuration parameters, including IP addresses, from a specific server.

CONFIRM (4) A client sends a Confirm message to any available server to determine whether the addresses it was assigned are still appropriate to the link to which the client is connected.

RENEW (5) A client sends a Renew message to the server that originally provided the client’s addresses and configuration parameters to extend the lifetimes on the addresses assigned to the client and to update other configuration parameters.

REBIND (6) A client sends a Rebind message to any available server to extend the lifetimes on the addresses assigned to the client and to update other configuration parameters; this message is sent after a client receives no response to a Renew message.

REPLY (7) A server sends a Reply message containing assigned addresses and configuration parameters in response to a Solicit, Request, Renew, Rebind message received from a client. A server sends a Reply message containing configuration parameters in response to an Information-request message. A server sends a Reply message in response to a Confirm message confirming or denying that the addresses assigned to the client are appropriate to the link to which the client is connected. A server sends a Reply message to acknowledge receipt of a Release or Decline message.

RELEASE (8) A client sends a Release message to the server that assigned addresses to the client to indicate that the client will no longer use one or more of the assigned addresses.

DECLINE (9) A client sends a Decline message to a server to indicate that the client has determined that one or more addresses assigned by the server are already in use on the link to which the client is connected.

RECONFIGURE (10) A server sends a Reconfigure message to a client to inform the client that the server has new or updated configuration parameters, and that the client is to initiate a Renew/Reply or Information-request/Reply transaction with the server in order to receive the updated information.

INFORMATION-REQUEST (11) A client sends an Information-request message to a server to request configuration parameters without the assignment of any IP addresses to the client.

RELAY-FORW (12) A relay agent sends a Relay-forward message to relay messages to servers, either directly or through another relay agent. The received message, either a client message or a Relay-forward message from another relay agent, is encapsulated in an option in the Relay-forward message.

RELAY-REPL (13) A server sends a Relay-reply message to a relay agent containing a message that the relay agent delivers to a client. The Relay-reply message may be relayed by other relay agents for delivery to the destination relay agent.

The server encapsulates the client message as an option in the Relay-reply message, which the relay agent extracts and relays to the client.

6.4. Status Codes

DHCPv6 uses status codes to communicate the success or failure of operations requested in messages from clients and servers, and to provide additional information about the specific cause of the failure of a message. The specific status codes are defined in Section 23.12.

If the Status Code option does not appear in a message in which the option could appear, the status of the message is assumed to be Success.

6.5. Transmission and Retransmission Parameters

This section presents a table of values used to describe the message transmission behavior of clients and servers.

Parameter	Default	Description
SOL_MAX_DELAY	1 sec	Max delay of first Solicit
SOL_TIMEOUT	1 sec	Initial Solicit timeout
SOL_MAX_RT	3600 secs	Max Solicit timeout value
REQ_TIMEOUT	1 sec	Initial Request timeout
REQ_MAX_RT	30 secs	Max Request timeout value
REQ_MAX_RC	10	Max Request retry attempts
CNF_MAX_DELAY	1 sec	Max delay of first Confirm
CNF_TIMEOUT	1 sec	Initial Confirm timeout
CNF_MAX_RT	4 secs	Max Confirm timeout
CNF_MAX_RD	10 secs	Max Confirm duration
REN_TIMEOUT	10 secs	Initial Renew timeout
REN_MAX_RT	600 secs	Max Renew timeout value
REB_TIMEOUT	10 secs	Initial Rebind timeout
REB_MAX_RT	600 secs	Max Rebind timeout value
INF_MAX_DELAY	1 sec	Max delay of first Information-request
INF_TIMEOUT	1 sec	Initial Information-request timeout
INF_MAX_RT	3600 secs	Max Information-request timeout value
REL_TIMEOUT	1 sec	Initial Release timeout
REL_MAX_RC	4	MAX Release retry attempts
DEC_TIMEOUT	1 sec	Initial Decline timeout
DEC_MAX_RC	4	Max Decline retry attempts
REC_TIMEOUT	2 secs	Initial Reconfigure timeout
REC_MAX_RC	8	Max Reconfigure attempts
HOP_COUNT_LIMIT	32	Max hop count in a Relay-forward message

6.6. Representation of time values and "Infinity" as a time value

All time values for lifetimes, T1 and T2 are unsigned integers. The value 0xffffffff is taken to mean "infinity" when used as a lifetime (as in [RFC4861]) or a value for T1 or T2.

7. Client/Server Message Formats

All DHCP messages sent between clients and servers share an identical fixed format header and a variable format area for options.

All values in the message header and in options are in network byte order.

Options are stored serially in the options field, with no padding between the options. Options are byte-aligned but are not aligned in any other way such as on 2 or 4 byte boundaries.

The following diagram illustrates the format of DHCP messages sent between clients and servers:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    msg-type   |               transaction-id                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.                            options                            .
.                           (variable)                          .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Client/Server message format

msg-type — Identifies the DHCP message type; the available message types are listed in Section 6.3.

transaction-id — The transaction ID for this message exchange.

options — Options carried in this message; options are described in Section 23.

8. Relay Agent/Server Message Formats

Relay agents exchange messages with servers to relay messages between clients and servers that are not connected to the same link.

All values in the message header and in options are in network byte order.

Options are stored serially in the options field, with no padding between the options. Options are byte-aligned but are not aligned in any other way such as on 2 or 4 byte boundaries.

There are two relay agent messages, which share the following format:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    msg-type   |   hop-count   |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
   |                                                               |
   |                         link-address                          |
   |                                                               |
   |                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|
   |                               |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
   |                                                               |
   |                         peer-address                          |
   |                                                               |
   |                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|
   |                               |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
   .                                                               .
   .            options (variable number and length)   ....        .
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Relay Agent/Server message format

The following sections describe the use of the Relay Agent message header.

8.1. Relay-forward Message

The following table defines the use of message fields in a Relay-forward message.

msg-type RELAY-FORW

hop-count Number of relay agents that have relayed this message.

link-address An address that will be used by the server to identify the link on which the client is located. This is typically global, site-scoped or ULA [RFC4193], but see discussion in Section 21.1.1.

peer-address The address of the client or relay agent from which the message to be relayed was received.

options	MUST include a "Relay Message option" (see Section 23.10); MAY include other options added by the relay agent.

8.2. Relay-reply Message

The following table defines the use of message fields in a Relay-reply message.

msg-type	RELAY-REPL
hop-count	Copied from the Relay-forward message
link-address	Copied from the Relay-forward message
peer-address	Copied from the Relay-forward message
options	MUST include a "Relay Message option"; see Section 23.10; MAY include other options

9. Representation and Use of Domain Names

So that domain names may be encoded uniformly, a domain name or a list of domain names is encoded using the technique described in section 3.1 of [RFC1035]. A domain name, or list of domain names, in DHCP MUST NOT be stored in compressed form, as described in section 4.1.4 of [RFC1035].

10. DHCP Unique Identifier (DUID)

Each DHCP client and server has a DUID. DHCP servers use DUIDs to identify clients for the selection of configuration parameters and in the association of IAs with clients. DHCP clients use DUIDs to identify a server in messages where a server needs to be identified. See Section 23.2 and Section 23.3 for the representation of a DUID in a DHCP message.

Clients and servers MUST treat DUIDs as opaque values and MUST only compare DUIDs for equality. Clients and servers MUST NOT in any other way interpret DUIDs. Clients and servers MUST NOT restrict DUIDs to the types defined in this document, as additional DUID types may be defined in the future.

The DUID is carried in an option because it may be variable length and because it is not required in all DHCP messages. The DUID is designed to be unique across all DHCP clients and servers, and stable for any specific client or server - that is, the DUID used by a client or server SHOULD NOT change over time if at all possible; for example, a device’s DUID should not change as a result of a change in the device’s network hardware.

The motivation for having more than one type of DUID is that the DUID must be globally unique, and must also be easy to generate. The sort of globally-unique identifier that is easy to generate for any given device can differ quite widely. Also, some devices may not contain any persistent storage. Retaining a generated DUID in such a device is not possible, so the DUID scheme must accommodate such devices.

10.1. DUID Contents

A DUID consists of a two-octet type code represented in network byte order, followed by a variable number of octets that make up the actual identifier. The length of the DUID (not including the type code) is at least 1 octet and at most 128 octets. The following types are currently defined:

Type	Description
1	Link-layer address plus time
2	Vendor-assigned unique ID based on Enterprise Number
3	Link-layer address
4	Universally Unique IDentifier (UUID) - see [RFC6355]

Formats for the variable field of the DUID for the first 3 of the above types are shown below. The fourth type, DUID-UUID [RFC6355], can be used in situations where there is a UUID stored in a device’s firmware settings.

10.2. DUID Based on Link-layer Address Plus Time, DUID-LLT

This type of DUID consists of a two octet type field containing the value 1, a two octet hardware type code, four octets containing a time value, followed by link-layer address of any one network interface that is connected to the DHCP device at the time that the DUID is generated. The time value is the time that the DUID is generated represented in seconds since midnight (UTC), January 1, 2000, modulo 2^32. The hardware type MUST be a valid hardware type assigned by the IANA as described in [RFC0826]. Both the time and the hardware type are stored in network byte order. The link-layer address is stored in canonical form, as described in [RFC2464].

The following diagram illustrates the format of a DUID-LLT:

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |               1               |    hardware type (16 bits)    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                        time (32 bits)                         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    .                                                               .
    .             link-layer address (variable length)              .
    .                                                               .
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: DUID-LLT format

The choice of network interface can be completely arbitrary, as long as that interface provides a globally unique link-layer address for the link type, and the same DUID-LLT SHOULD be used in configuring all network interfaces connected to the device, regardless of which interface’s link-layer address was used to generate the DUID-LLT.

Clients and servers using this type of DUID MUST store the DUID-LLT in stable storage, and MUST continue to use this DUID-LLT even if the network interface used to generate the DUID-LLT is removed. Clients and servers that do not have any stable storage MUST NOT use this type of DUID.

Clients and servers that use this DUID SHOULD attempt to configure the time prior to generating the DUID, if that is possible, and MUST use some sort of time source (for example, a real-time clock) in generating the DUID, even if that time source could not be configured prior to generating the DUID. The use of a time source makes it unlikely that two identical DUID-LLTs will be generated if the network interface is removed from the client and another client then uses the same network interface to generate a DUID-LLT. A collision between two DUID-LLTs is very unlikely even if the clocks have not been configured prior to generating the DUID.

This method of DUID generation is recommended for all general purpose computing devices such as desktop computers and laptop computers, and also for devices such as printers, routers, and so on, that contain some form of writable non-volatile storage.

Despite our best efforts, it is possible that this algorithm for generating a DUID could result in a client identifier collision. A DHCP client that generates a DUID-LLT using this mechanism MUST provide an administrative interface that replaces the existing DUID with a newly-generated DUID-LLT.

10.3. DUID Assigned by Vendor Based on Enterprise Number, DUID-EN

This form of DUID is assigned by the vendor to the device. It consists of the vendor’s registered Private Enterprise Number as maintained by IANA [IANA-PEN] followed by a unique identifier assigned by the vendor. The following diagram summarizes the structure of a DUID-EN:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |               2               |       enterprise-number       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   enterprise-number (contd)   |                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
   .                           identifier                          .
   .                       (variable length)                       .
   .                                                               .
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: DUID-EN format

The source of the identifier is left up to the vendor defining it, but each identifier part of each DUID-EN MUST be unique to the device that is using it, and MUST be assigned to the device no later than at the first usage and stored in some form of non-volatile storage. This typically means being assigned during manufacture process in case of physical devices or when the image is created or booted for the first time in case of virtual machines. The generated DUID SHOULD be recorded in non-erasable storage. The enterprise-number is the vendor’s registered Private Enterprise Number as maintained by IANA [IANA-PEN]. The enterprise-number is stored as an unsigned 32 bit number.

An example DUID of this type might look like this:

```
   +---+---+---+---+---+---+---+---+
   | 0 | 2 | 0 | 0 | 0 |  9| 12|192|
   +---+---+---+---+---+---+---+---+
   |132|211| 3 | 0 | 9 | 18|
   +---+---+---+---+---+---+
```

Figure 6: DUID-EN example

This example includes the two-octet type of 2, the Enterprise Number (9), followed by eight octets of identifier data (0x0CC084D303000912).

10.4. DUID Based on Link-layer Address, DUID-LL

This type of DUID consists of two octets containing the DUID type 3, a two octet network hardware type code, followed by the link-layer address of any one network interface that is permanently connected to the client or server device. For example, a host that has a network interface implemented in a chip that is unlikely to be removed and used elsewhere could use a DUID-LL. The hardware type MUST be a valid hardware type assigned by the IANA, as described in [RFC0826]. The hardware type is stored in network byte order. The link-layer address is stored in canonical form, as described in [RFC2464]. The following diagram illustrates the format of a DUID-LL:

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |               3               |    hardware type (16 bits)    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    .                                                               .
    .             link-layer address (variable length)              .
    .                                                               .
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: DUID-LL format

The choice of network interface can be completely arbitrary, as long as that interface provides a unique link-layer address and is permanently attached to the device on which the DUID-LL is being generated. The same DUID-LL SHOULD be used in configuring all network interfaces connected to the device, regardless of which interface’s link-layer address was used to generate the DUID.

DUID-LL is recommended for devices that have a permanently-connected network interface with a link-layer address, and do not have nonvolatile, writable stable storage. DUID-LL MUST NOT be used by DHCP clients or servers that cannot tell whether or not a network interface is permanently attached to the device on which the DHCP client is running.

11. Identity Association

An "identity-association" (IA) is a construct through which a server and a client can identify, group, and manage a set of related IPv6 addresses or delegated prefixes. Each IA consists of an IAID and associated configuration information.

The IAID uniquely identifies the IA and must be chosen to be unique among the IAIDs for that IA type on the client. The IAID is chosen by the client. For any given use of an IA by the client, the IAID for that IA MUST be consistent across restarts of the DHCP client. The client may maintain consistency either by storing the IAID in non-volatile storage or by using an algorithm that will consistently produce the same IAID as long as the configuration of the client has not changed. There may be no way for a client to maintain consistency of the IAIDs if it does not have non-volatile storage and the client’s hardware configuration changes. If the client uses only one IAID, it can use a well-known value, e.g., zero.

11.1. Identity Associations for Address Assignment

A client must associate at least one distinct IA with each of its network interfaces for which it is to request the assignment of IPv6 addresses from a DHCP server. The client uses the IAs assigned to an interface to obtain configuration information from a server for that interface. Each IA must be associated with exactly one interface.

The configuration information in an IA consists of one or more IPv6 addresses along with the times T1 and T2 for the IA. See Section 22.4 for the representation of an IA in a DHCP message.

Each address in an IA has a preferred lifetime and a valid lifetime, as defined in [RFC4862]. The lifetimes are transmitted from the DHCP server to the client in the IA option. The lifetimes apply to the use of IPv6 addresses, as described in section 5.5.4 of [RFC4862].

11.2. Identity Associations for Prefix Delegation

An IA_PD is different from an IA for address assignment, in that it does not need to be associated with exactly one interface. One IA_PD can be associated with the requesting router, with a set of interfaces or with exactly one interface. A requesting router must create at least one distinct IA_PD. It may associate a distinct IA_PD with each of its downstream network interfaces and use that IA_PD to obtain a prefix for that interface from the delegating router.

The configuration information in an IA_PD consists of one or more IPv6 prefixes along with the times T1 and T2 for the IA_PD. See Section 23.21 for the representation of an IA_PD in a DHCP message.

12. Selecting Addresses for Assignment to an IA

A server selects addresses to be assigned to an IA according to the address assignment policies determined by the server administrator and the specific information the server determines about the client from some combination of the following sources:

- The link to which the client is attached. The server determines the link as follows:

  * If the server receives the message directly from the client and the source address in the IP datagram in which the message was received is a link-local address, then the client is on the same link to which the interface over which the message was received is attached.

  * If the server receives the message from a forwarding relay agent, then the client is on the same link as the one to which the interface, identified by the link-address field in the message from the relay agent, is attached. According to [RFC6221], the server MUST ignore any link-address field whose value is zero. The link address field referes to the link-address field of the Relay-Forward message, and the link-address fields in any Relay-Forward messages that may be nested within the Relay-Forward message.

  * If the server receives the message directly from the client and the source address in the IP datagram in which the message was received is not a link-local address, then the client is on the link identified by the source address in the IP datagram (note that this situation can occur only if the server has enabled the use of unicast message delivery by the client and the client has sent a message for which unicast delivery is allowed).

- The DUID supplied by the client.

- Other information in options supplied by the client, e.g. IA Address options that include the client’s requests for specific addresses.

- Other information in options supplied by the relay agent.

Any address assigned by a server that is based on an EUI-64 identifier MUST include an interface identifier with the "u" (universal/local) and "g" (individual/group) bits of the interface identifier set appropriately, as indicated in section 2.5.1 of [RFC4291].

A server MUST NOT assign an address that is otherwise reserved for some other purpose. For example, a server MUST NOT assign reserved anycast addresses, as defined in [RFC2526], from any subnet.

13. Management of Temporary Addresses

A client may request the assignment of temporary addresses (see [RFC4941] for the definition of temporary addresses). DHCPv6 handling of address assignment is no different for temporary addresses.

Clients ask for temporary addresses and servers assign them. Temporary addresses are carried in the Identity Association for Temporary Addresses (IA_TA) option (see Section 23.5). Each IA_TA option contains at most one temporary address for each of the prefixes on the link to which the client is attached.

The lifetime of the assigned temporary address is set in the IA Address Option (see Section 23.6) with in the IA_TA option. It is RECOMMENDED to set short lifetimes, typically shorter than TEMP_VALID_LIFETIME and TEMP_PREFERRED_LIFETIME (see Section 5, [RFC4941].

The IAID number space for the IA_TA option IAID number space is separate from the IA_NA option IAID number space.

A DHCPv6 server implementation MAY generate temporary addresses referring to the algorithm defined in Section 3.2.1, [RFC4941], with additional condition that the new address is not duplicated with any assigned addresses.

The server MAY update the DNS for a temporary address, as described in section 4 of [RFC4941].

On the clients, by default, temporary addresses are preferred in source address selection, according to Rule 7, [RFC6724]. However, this policy is overridable.

One of the most important properties of temporary address is unlinkability of different actions over time. So, it is NOT RECOMMENDED for a client to renew expired temporary addresses, though DHCPv6 provides such possibility (see Section 23.5).

14. Transmission of Messages by a Client

Unless otherwise specified in this document, or in a document that describes how IPv6 is carried over a specific type of link (for link types that do not support multicast), a client sends DHCP messages to the All_DHCP_Relay_Agents_and_Servers.

A client uses multicast to reach all servers or an individual server. An individual server is indicated by specifying that server’s DUID in a Server Identifier option (see Section 23.3) in the client’s message (all servers will receive this message but only the indicated server will respond). All servers are indicated by not supplying this option.

A client may send some messages directly to a server using unicast, as described in Section 23.12.

14.1. Rate Limiting

In order to avoid prolonged message bursts that may be caused by possible logic loops, a DHCPv6 client MUST limit the rate of DHCPv6 messages it transmits. One example is that a client obtains an address, but does not like the response; it reverts back to Solicit procedure, discovers the same (sole) server, requests an address and gets the same address as before (the server still has the lease that was requested just previously). This loops can repeat infinitely if there is not a quit/stop mechanism. Therefore, a client must not initiate transmissions too frequently.

A recommended method for implementing the rate limiting function is a token bucket, limiting the average rate of transmission to a certain number in a certain time. This method of bounding burstiness also guarantees that the long-term transmission rate will not exceed.

TRT	Transmission Rate Limit

The Transmission Rate Limit parameter (TRT) SHOULD be configurable. A possible default could be 20 packets in 20 seconds.

For a device that has multiple interfaces, the limit MUST be enforced on a per interface basis.

Rate limiting of forwarded DHCPv6 messages and server-side messages are out of scope of this specification.

15. Reliability of Client Initiated Message Exchanges

DHCP clients are responsible for reliable delivery of messages in the client-initiated message exchanges described in Section 18 and Section 19. If a DHCP client fails to receive an expected response from a server, the client must retransmit its message. This section describes the retransmission strategy to be used by clients in client-initiated message exchanges.

Note that the procedure described in this section is slightly modified when used with the Solicit message. The modified procedure is described in Section 18.1.2.

The client begins the message exchange by transmitting a message to the server. The message exchange terminates when either the client successfully receives the appropriate response or responses from a server or servers, or when the message exchange is considered to have failed according to the retransmission mechanism described below.

The client retransmission behavior is controlled and described by the following variables:

RT	Retransmission timeout
IRT	Initial retransmission time
MRC	Maximum retransmission count
MRT	Maximum retransmission time
MRD	Maximum retransmission duration
RAND	Randomization factor

With each message transmission or retransmission, the client sets RT according to the rules given below. If RT expires before the message exchange terminates, the client recomputes RT and retransmits the message.

Each of the computations of a new RT include a randomization factor (RAND), which is a random number chosen with a uniform distribution between -0.1 and +0.1. The randomization factor is included to minimize synchronization of messages transmitted by DHCP clients.

The algorithm for choosing a random number does not need to be cryptographically sound. The algorithm SHOULD produce a different sequence of random numbers from each invocation of the DHCP client.

RT for the first message transmission is based on IRT:

```
   RT = IRT + RAND*IRT
```

RT for each subsequent message transmission is based on the previous value of RT:

```
   RT = 2*RTprev + RAND*RTprev
```

MRT specifies an upper bound on the value of RT (disregarding the randomization added by the use of RAND). If MRT has a value of 0, there is no upper limit on the value of RT. Otherwise:

```
   if (RT > MRT)
      RT = MRT + RAND*MRT
```

MRC specifies an upper bound on the number of times a client may retransmit a message. Unless MRC is zero, the message exchange fails once the client has transmitted the message MRC times.

MRD specifies an upper bound on the length of time a client may retransmit a message. Unless MRD is zero, the message exchange fails once MRD seconds have elapsed since the client first transmitted the message.

If both MRC and MRD are non-zero, the message exchange fails whenever either of the conditions specified in the previous two paragraphs are met.

If both MRC and MRD are zero, the client continues to transmit the message until it receives a response.

A client is not expected to listen for a response during the entire period between transmission of Solicit or Information-request messages.

16. Message Validation

Clients and servers might get messages that contain options not allowed to appear in the received message. For example, an IA option is not allowed to appear in an Information-request message. Clients and servers MAY choose either to extract information from such a message if the information is of use to the recipient, or to ignore such message completely and just drop it.

A server MUST discard any Solicit, Confirm, Rebind or Information-request messages it receives with a unicast destination address.

Message validation based on DHCP authentication is discussed in Section 22.4.2.

If a server receives a message that contains options it should not contain (such as an Information-request message with an IA option), is missing options that it should contain, or is otherwise not valid, it MAY send a Reply (or Advertise as appropriate) with a Server Identifier option, a Client Identifier option if one was included in the message and a Status Code option with status UnSpecFail.

A client or server MUST silently discary and yreceived DHCPv6 messages with an unknown message type.

16.1. Use of Transaction IDs

The "transaction-id" field holds a value used by clients and servers to synchronize server responses to client messages. A client SHOULD generate a random number that cannot easily be guessed or predicted to use as the transaction ID for each new message it sends. Note that if a client generates easily predictable transaction identifiers, it may become more vulnerable to certain kinds of attacks from off-path intruders. A client MUST leave the transaction ID unchanged in retransmissions of a message.

16.2. Solicit Message

Clients MUST discard any received Solicit messages.

Servers MUST discard any Solicit messages that do not include a Client Identifier option or that do include a Server Identifier option.

16.3. Advertise Message

Clients MUST discard any received Advertise message that meets any of the following conditions:

- the message does not include a Server Identifier option.

- the message does not include a Client Identifier option.

- the contents of the Client Identifier option does not match the client’s DUID.

- the "transaction-id" field value does not match the value the client used in its Solicit message.

Servers and relay agents MUST discard any received Advertise messages.

16.4. Request Message

Clients MUST discard any received Request messages.

Servers MUST discard any received Request message that meets any of the following conditions:

- the message does not include a Server Identifier option.

- the contents of the Server Identifier option do not match the server’s DUID.

- the message does not include a Client Identifier option.

16.5. Confirm Message

Clients MUST discard any received Confirm messages.

Servers MUST discard any received Confirm messages that do not include a Client Identifier option or that do include a Server Identifier option.

16.6. Renew Message

Clients MUST discard any received Renew messages.

Servers MUST discard any received Renew message that meets any of the following conditions:

- the message does not include a Server Identifier option.

- the contents of the Server Identifier option does not match the server’s identifier.

- the message does not include a Client Identifier option.

16.7. Rebind Message

Clients MUST discard any received Rebind messages.

Servers MUST discard any received Rebind messages that do not include a Client Identifier option or that do include a Server Identifier option.

16.8. Decline Messages

Clients MUST discard any received Decline messages.

Servers MUST discard any received Decline message that meets any of the following conditions:

- the message does not include a Server Identifier option.

- the contents of the Server Identifier option does not match the server’s identifier.

- the message does not include a Client Identifier option.

16.9. Release Message

Clients MUST discard any received Release messages.

Servers MUST discard any received Release message that meets any of the following conditions:

- the message does not include a Server Identifier option.

- the contents of the Server Identifier option does not match the server’s identifier.

- the message does not include a Client Identifier option.

16.10. Reply Message

Clients MUST discard any received Reply message that meets any of the following conditions:

- the message does not include a Server Identifier option.

- the "transaction-id" field in the message does not match the value used in the original message.

If the client included a Client Identifier option in the original message, the Reply message MUST include a Client Identifier option and the contents of the Client Identifier option MUST match the DUID of the client; OR, if the client did not include a Client Identifier option in the original message, the Reply message MUST NOT include a Client Identifier option.

Servers and relay agents MUST discard any received Reply messages.

16.11. Reconfigure Message

Servers and relay agents MUST discard any received Reconfigure messages.

Clients MUST discard any Reconfigure message that meets any of the following conditions:

- the message was not unicast to the client.

- the message does not include a Server Identifier option.

- the message does not include a Client Identifier option that contains the client’s DUID.

- the message does not contain a Reconfigure Message option.

- the Reconfigure Message option msg-type is not a valid value.

- the message includes any IA options and the msg-type in the Reconfigure Message option is INFORMATION-REQUEST.

- the message does not include DHCP authentication:

  * the message does not contain an authentication option.

  * the message does not pass the authentication validation performed by the client.

16.12. Information-request Message

Clients MUST discard any received Information-request messages.

Servers MUST discard any received Information-request message that meets any of the following conditions:

- The message includes a Server Identifier option and the DUID in the option does not match the server’s DUID.

- The message includes an IA option.

16.13. Relay-forward Message

Clients MUST discard any received Relay-forward messages.

16.14. Relay-reply Message

Clients and servers MUST discard any received Relay-reply messages.

17. Client Source Address and Interface Selection

Client’s behavior is different depending on the purpose of the configuration.

17.1. Address Assignment

When a client sends a DHCP message to the All_DHCP_Relay_Agents_and_Servers address, it SHOULD send the message through the interface for which configuration information is being requested. However, the client MAY send the message through another interface if the interface is a logical interface without direct link attachement or the client is certain that two interfaces are attached to the same link.

When a client sends a DHCP message directly to a server using unicast (after receiving the Server Unicast option from that server), the source address in the header of the IPv6 datagram MUST be an address assigned to the interface for which the client is interested in obtaining configuration and which is suitable for use by the server in responding to the client.

17.2. Prefix Delegation

Delegated prefixes are not associated with a particular interface in the same way as addresses are for address assignment, and mentioned above.

When a client (acting as requesting router) sends a DHCP message for the purpose of prefix delegation, it SHOULD be sent on the interface associated with the upstream router (ISP network). The upstream interface is typically determined by configuration. This rule applies even in the case where a separate IA_PD is used for each downstream interface.

When a requesting router sends a DHCP message directly to a delegating router using unicast (after receiving the Server Unicast option from that delegating router), the source address SHOULD be an address from the upstream interface and which is suitable for use by the delegating router in responding to the requesting router.

18. DHCP Server Solicitation

This section describes how a client locates servers that will assign addresses and delegated prefixes to IAs belonging to the client.

The client is responsible for creating IAs and requesting that a server assign IPv6 addresses and delegated prefixes to the IAs. The client first creates the IAs and assigns IAIDs to them. The client then transmits a Solicit message containing the IA options describing the IAs. The client MUST NOT be using any of the addresses or delegated prefixes for which it tries to obtain the bindings by sending the Solicit message. In particular, if the client had some valid bindings and has chosen to start the server solicitation process to obtain the bindings from a different server, the client MUST stop using the addresses and delegated prefixes for the bindings it had obtained from the previous server, and which it is now trying to obtain from a new server.

Servers that can assign addresses or delegated prefixes to the IAs respond to the client with an Advertise message. The client then initiates a configuration exchange as described in Section 19.

If the client will accept a Reply message with committed address assignments and other resources in response to the Solicit message, the client includes a Rapid Commit option (see Section 23.14) in the Solicit message.

18.1. Client Behavior

A client uses the Solicit message to discover DHCP servers configured to assign addresses or return other configuration parameters on the link to which the client is attached.

18.1.1. Creation of Solicit Messages

The client sets the "msg-type" field to SOLICIT. The client generates a transaction ID and inserts this value in the "transaction-id" field.

The client MUST include a Client Identifier option to identify itself to the server. The client includes IA options for any IAs to which it wants the server to assign addresses. The client MAY include addresses in the IAs as a hint to the server about addresses for which the client has a preference. The client MUST NOT include any other options in the Solicit message, except as specifically allowed in the definition of individual options.

The client uses IA_NA options to request the assignment of non-temporary addresses and uses IA_TA options to request the assignment of temporary addresses. Either IA_NA or IA_TA options, or a combination of both, can be included in DHCP messages.

The client MUST include an Option Request option (see Section 23.7) to request the SOL_MAX_RT option (see Section 23.23) and any other options the client is interested in receiving. The client MAY additionally include instances of those options that are identified in the Option Request option, with data values as hints to the server about parameter values the client would like to have returned.

The client includes a Reconfigure Accept option (see Section 23.20) if the client is willing to accept Reconfigure messages from the server.

18.1.2. Transmission of Solicit Messages

The first Solicit message from the client on the interface MUST be delayed by a random amount of time between 0 and SOL_MAX_DELAY. In the case of a Solicit message transmitted when DHCP is initiated by IPv6 Neighbor Discovery, the delay gives the amount of time to wait after IPv6 Neighbor Discovery causes the client to invoke the stateful address autoconfiguration protocol (see section 5.5.3 of [RFC4862]). This random delay desynchronizes clients which start at the same time (for example, after a power outage).

The client transmits the message according to Section 15, using the following parameters:

IRT	SOL_TIMEOUT
MRT	SOL_MAX_RT
MRC	0
MRD	0

If the client has included a Rapid Commit option in its Solicit message, the client terminates the waiting process as soon as a Reply message with a Rapid Commit option is received.

If the client is waiting for an Advertise message, the mechanism in Section 15 is modified as follows for use in the transmission of Solicit messages. The message exchange is not terminated by the receipt of an Advertise before the first RT has elapsed. Rather, the client collects Advertise messages until the first RT has elapsed.

Also, the first RT MUST be selected to be strictly greater than IRT by choosing RAND to be strictly greater than 0.

A client MUST collect Advertise messages for the first RT seconds, unless it receives an Advertise message with a preference value of 255. The preference value is carried in the Preference option (Section 23.8). Any Advertise that does not include a Preference option is considered to have a preference value of 0. If the client receives an Advertise message that includes a Preference option with a preference value of 255, the client immediately begins a client-initiated message exchange (as described in Section 19) by sending a Request message to the server from which the Advertise message was received. If the client receives an Advertise message that does not include a Preference option with a preference value of 255, the client continues to wait until the first RT elapses. If the first RT elapses and the client has received an Advertise message, the client SHOULD continue with a client-initiated message exchange by sending a Request message.

If the client does not receive any Advertise messages before the first RT has elapsed, it begins the retransmission mechanism described in Section 15. The client terminates the retransmission process as soon as it receives any Advertise message, and the client acts on the received Advertise message without waiting for any additional Advertise messages.

A DHCP client SHOULD choose MRC and MRD to be 0. If the DHCP client is configured with either MRC or MRD set to a value other than 0, it MUST stop trying to configure the interface if the message exchange fails. After the DHCP client stops trying to configure the interface, it SHOULD restart the reconfiguration process after some external event, such as user input, system restart, or when the client is attached to a new link.

18.1.3. Receipt of Advertise Messages

The client MUST process SOL_MAX_RT and INF_MAX_RT options in an Advertise message, even if the message contains a Status Code option indicating a failure, and the Advertise message will be discarded by the client.

The client MUST ignore any IAs in an Advertise message that include a Status Code option containing the value NoAddrsAvail, with the exception that the client MAY display the associated status message to the user.

Upon receipt of one or more valid Advertise messages, the client selects one or more Advertise messages based upon the following criteria.

- Those Advertise messages with the highest server preference value are preferred over all other Advertise messages.

- Within a group of Advertise messages with the same server preference value, a client MAY select those servers whose Advertise messages advertise information of interest to the client.

- The client MAY choose a less-preferred server if that server has a better set of advertised parameters, such as the available addresses advertised in IAs.

Once a client has selected Advertise message(s), the client will typically store information about each server, such as server preference value, addresses advertised, when the advertisement was received, and so on.

In practice, this means that the client will maintain independent per-IA state machines per each selected server.

If the client needs to select an alternate server in the case that a chosen server does not respond, the client chooses the next server according to the criteria given above.

18.1.4. Receipt of Reply Message

If the client includes a Rapid Commit option in the Solicit message, it will expect a Reply message that includes a Rapid Commit option in response. The client discards any Reply messages it receives that do not include a Rapid Commit option. If the client receives a valid Reply message that includes a Rapid Commit option, it processes the message as described in Section 19.1.8. If it does not receive such a Reply message and does receive a valid Advertise message, the client processes the Advertise message as described in Section 18.1.3.

If the client subsequently receives a valid Reply message that includes a Rapid Commit option, it either:

- processes the Reply message as described in Section 19.1.8, and discards any Reply messages received in response to the Request message, or

- processes any Reply messages received in response to the Request message and discards the Reply message that includes the Rapid Commit option.

18.2. Server Behavior

A server sends an Advertise message in response to valid Solicit messages it receives to announce the availability of the server to the client.

18.2.1. Receipt of Solicit Messages

The server determines the information about the client and its location as described in Section 12 and checks its administrative policy about responding to the client. If the server is not permitted to respond to the client, the server discards the Solicit message. For example, if the administrative policy for the server is that it may only respond to a client that is willing to accept a Reconfigure message, if the client does not include a Reconfigure Accept option (see Section 23.20) in the Solicit message, the servers discard the Solicit message.

If the client has included a Rapid Commit option in the Solicit message and the server has been configured to respond with committed address assignments and other resources, the server responds to the Solicit with a Reply message as described in Section 18.2.3. Otherwise, the server ignores the Rapid Commit option and processes the remainder of the message as if no Rapid Commit option were present.

18.2.2. Creation and Transmission of Advertise Messages

The server sets the "msg-type" field to ADVERTISE and copies the contents of the transaction-id field from the Solicit message received from the client to the Advertise message. The server includes its server identifier in a Server Identifier option and copies the Client Identifier from the Solicit message into the Advertise message.

The server MAY add a Preference option to carry the preference value for the Advertise message. The server implementation SHOULD allow the setting of a server preference value by the administrator. The server preference value MUST default to zero unless otherwise configured by the server administrator.

The server includes a Reconfigure Accept option if the server wants to require that the client accept Reconfigure messages.

The server includes options the server will return to the client in a subsequent Reply message. The information in these options may be used by the client in the selection of a server if the client receives more than one Advertise message. If the client has included an Option Request option in the Solicit message, the server includes options in the Advertise message containing configuration parameters for all of the options identified in the Option Request option that the server has been configured to return to the client. The server MAY return additional options to the client if it has been configured to do so. The server must be aware of the recommendations on packet sizes and the use of fragmentation in section 5 of [RFC2460].

If the Solicit message from the client included one or more IA options, the server MUST include IA options in the Advertise message containing any addresses that would be assigned to IAs contained in the Solicit message from the client. If the client has included addresses in the IAs in the Solicit message, the server uses those addresses as hints about the addresses the client would like to receive.

If the server will not assign any addresses to any IAs in a subsequent Request from the client, the server MUST send an Advertise message to the client that includes only a Status Code option with code NoAddrsAvail and a status message for the user, a Server Identifier option with the server’s DUID, a Client Identifier option with the client’s DUID, and (optionally) SOL_MAX_RT and/or INF_MAX_RT options. The server SHOULD include other stateful IA options (like IA_PD) and other configuration options in the Advertise message.

If the Solicit message was received directly by the server, the server unicasts the Advertise message directly to the client using the address in the source address field from the IP datagram in which the Solicit message was received. The Advertise message MUST be unicast on the link from which the Solicit message was received.

If the Solicit message was received in a Relay-forward message, the server constructs a Relay-reply message with the Advertise message in the payload of a "relay-message" option. If the Relay-forward messages included an Interface-id option, the server copies that option to the Relay-reply message. The server unicasts the Relay-reply message directly to the relay agent using the address in the source address field from the IP datagram in which the Relay-forward message was received.

18.2.3. Creation and Transmission of Reply Messages

The server MUST commit the assignment of any addresses or other configuration information message before sending a Reply message to a client in response to a Solicit message.

DISCUSSION:

When using the Solicit-Reply message exchange, the server commits the assignment of any addresses before sending the Reply message. The client can assume it has been assigned the addresses in the Reply message and does not need to send a Request message for those addresses.

Typically, servers that are configured to use the Solicit-Reply message exchange will be deployed so that only one server will respond to a Solicit message. If more than one server responds, the client will only use the addresses from one of the servers, while the addresses from the other servers will be committed to the client but not used by the client.

The server includes a Rapid Commit option in the Reply message to indicate that the Reply is in response to a Solicit message.

The server includes a Reconfigure Accept option if the server wants to require that the client accept Reconfigure messages.

The server produces the Reply message as though it had received a Request message, as described in Section 19.2.1. The server transmits the Reply message as described in Section 19.2.8.

18.3. Client behavior for Prefix Delegation

The requesting router creates and transmits a Solicit message as described in Section 18.1.1 and Section 18.1.2. The client creates an IA_PD and assigns it an IAID. The client MUST include the IA_PD option in the Solicit message.

The client processes any received Advertise messages as described in Section 18.1.3. The client MAY choose to consider the presence of advertised prefixes in its decision about which delegating router to respond to.

The client MUST ignore any IA_PDs in an Advertise message that include a Status Code option containing the value NoPrefixAvail, with the exception that the client MAY display the associated status message to the user and SHOULD process SOL_MAX_RT and INF_MAX_RT options.

18.4. Server Behavior for Prefix Delegation

The server sends an Advertise message to the requesting router in the same way as described in Section 18.2.2. If the message contains an IA_PD option and the delegating router is configured to delegate prefix(es) to the requesting router, the delegating router selects the prefix(es) to be delegated to the requesting router. The mechanism through which the delegating router selects prefix(es) for delegation is not specified in this document. Examples of ways in which the server might select prefix(es) for a client include: static assignment based on subscription to an ISP; dynamic assignment from a pool of available prefixes; selection based on an external authority such as a RADIUS server using the Framed-IPv6-Prefix option as described in [RFC3162].

If the client includes an IA_PD Prefix option in the IA_PD option in its Solicit message, the server MAY choose to use the information in that option to select the prefix(es) or prefix size to be delegated to the client.

The server sends an Advertise message to the requesting router in the same way as described in Section 18.2.2. The server MUST include an IA_PD option, identifying any prefix(es) that the server will delegate to the client.

If the server will not assign any prefixes to an IA_PD in a subsequent Request from the requesting router, the server MUST send an Advertise message to the client that includes the IA_PD with no prefixes in the IA_PD and a Status Code option in the IA_PD containing status code NoPrefixAvail and a status message for the user, a Server Identifier option with the server’s DUID and a Client Identifier option with the client’s DUID. The server SHOULD include other stateful IA options (like IA_NA) and other configuration options in the Advertise message.

19. DHCP Client-Initiated Configuration Exchange

A client initiates a message exchange with a server or servers to acquire or update configuration information of interest. The client may initiate the configuration exchange as part of the operating system configuration process, when requested to do so by the application layer, when required by Stateless Address Autoconfiguration or as required to extend the lifetime of address(es) or/and delegated prefix(es), using Renew and Rebind messages.

According to a terminology for the prefix delegation, a client requesting a delegation of a prefix is referred to as a requesting router and a server delegating the prefix is referred to as a delegating router. The requesting router and the delegating router use the IA_PD Prefix option to exchange information about prefix(es) in much the same way as IA Address options are used for assigned addresses. Typically, a single DHCP session is used to exchange information about addresses and prefixes, i.e. IA_NA and IA_PD options are carried in the same message.

19.1. Client Behavior

A client uses Request, Renew, Rebind, Release and Decline messages during the normal life cycle of addresses. It uses Confirm to validate addresses when it may have moved to a new link. It uses Information-Request messages when it needs configuration information but no addresses.

If the client has a source address of sufficient scope that can be used by the server as a return address, and the client has received a Server Unicast option (Section 23.12) from the server, the client SHOULD unicast any Request, Renew, Release and Decline messages to the server.

DISCUSSION:

> Use of unicast may avoid delays due to the relaying of messages by relay agents, as well as avoid overhead and duplicate responses by servers due to the delivery of client messages to multiple servers. Requiring the client to relay all DHCP messages through a relay agent enables the inclusion of relay agent options in all messages sent by the client. The server should enable the use of unicast only when relay agent options will not be used.

19.1.1. Creation and Transmission of Request Messages

The client uses a Request message to populate IAs with addresses and obtain other configuration information. The client includes one or more IA options in the Request message. The server then returns addresses and other information about the IAs to the client in IA options in a Reply message.

The client generates a transaction ID and inserts this value in the "transaction-id" field.

The client places the identifier of the destination server in a Server Identifier option.

The client MUST include a Client Identifier option to identify itself to the server. The client adds any other appropriate options, including one or more IA options (if the client is requesting that the server assign it some network addresses).

The client MUST include an Option Request option (see Section 23.7) to indicate the options the client is interested in receiving. The client MAY include options with data values as hints to the server about parameter values the client would like to have returned.

The client includes a Reconfigure Accept option (see Section 23.20) indicating whether or not the client is willing to accept Reconfigure messages from the server.

The client transmits the message according to Section 15, using the following parameters:

IRT	REQ_TIMEOUT
MRT	REQ_MAX_RT
MRC	REQ_MAX_RC
MRD	0

If the message exchange fails, the client takes an action based on the client’s local policy. Examples of actions the client might take include:

- Select another server from a list of servers known to the client; for example, servers that responded with an Advertise message.

- Initiate the server discovery process described in Section 18.

- Terminate the configuration process and report failure.

19.1.2. Creation and Transmission of Confirm Messages

Whenever a client may have moved to a new link, the prefixes/ addresses assigned to the interfaces on that link may no longer be appropriate for the link to which the client is attached. Examples of times when a client may have moved to a new link include:

- The client reboots.

- The client is physically connected to a wired connection.

- The client returns from sleep mode.

- The client using a wireless technology changes access points.

In any situation when a client may have moved to a new link, the client SHOULD initiate a Confirm/Reply message exchange. The client includes any IAs assigned to the interface that may have moved to a new link, along with the addresses associated with those IAs, in its Confirm message. Any responding servers will indicate whether those addresses are appropriate for the link to which the client is attached with the status in the Reply message it returns to the client.

One example when this rule may not be followed is when the client does not store its leases in stable storage and experiences a reboot. It may simply not retain any information, so it does not know what to confirm. In such case client MUST restart server discovery process as described in Section 18.1.1.

The client sets the "msg-type" field to CONFIRM. The client generates a transaction ID and inserts this value in the "transaction-id" field.

The client MUST include a Client Identifier option to identify itself to the server. The client includes IA options for all of the IAs assigned to the interface for which the Confirm message is being sent. The IA options include all of the addresses the client currently has associated with those IAs. The client SHOULD set the T1 and T2 fields in any IA_NA options, and the preferred-lifetime and valid-lifetime fields in the IA Address options to 0, as the server will ignore these fields.

The first Confirm message from the client on the interface MUST be delayed by a random amount of time between 0 and CNF_MAX_DELAY. The client transmits the message according to Section 15, using the following parameters:

IRT	CNF_TIMEOUT
MRT	CNF_MAX_RT
MRC	0
MRD	CNF_MAX_RD

If the client receives no responses before the message transmission process terminates, as described in Section 15, the client SHOULD continue to use any IP addresses, using the last known lifetimes for those addresses, and SHOULD continue to use any other previously obtained configuration parameters.

19.1.3. Creation and Transmission of Renew Messages

To extend the valid and preferred lifetimes for the addresses associated with an IA, the client sends a Renew message to the server from which the client obtained the addresses in the IA containing an IA option for the IA. The client includes IA Address options in the IA option for the addresses associated with the IA. The server determines new lifetimes for the addresses in the IA according to the administrative configuration of the server. The server may also add new addresses to the IA. The server may remove addresses from the IA by setting the preferred and valid lifetimes of those addresses to zero.

The server controls the time at which the client contacts the server to extend the lifetimes on assigned addresses through the T1 and T2 parameters assigned to an IA.

At time T1 for an IA, the client initiates a Renew/Reply message exchange to extend the lifetimes on any addresses in the IA. The client includes an IA option with all addresses currently assigned to the IA in its Renew message.

If T1 or T2 is set to 0 by the server (for an IA_NA) or there are no T1 or T2 times (for an IA_TA), the client may send a Renew or Rebind message, respectively, at the client’s discretion.

The client sets the "msg-type" field to RENEW. The client generates a transaction ID and inserts this value in the "transaction-id" field.

The client places the identifier of the destination server in a Server Identifier option.

The client MUST include a Client Identifier option to identify itself to the server. The client adds any appropriate options, including one or more IA options. The client MUST include the list of addresses the client currently has associated with the IAs in the Renew message.

The client MUST include an Option Request option (see Section 23.7) to indicate the options the client is interested in receiving. The client MAY include options with data values as hints to the server about parameter values the client would like to have returned.

The client transmits the message according to Section 15, using the following parameters:

```
   IRT     REN_TIMEOUT
```

MRT	REN_MAX_RT
MRC	0
MRD	Remaining time until T2

The message exchange is terminated when time T2 is reached (see Section 19.1.4), at which time the client begins a Rebind message exchange.

19.1.4. Creation and Transmission of Rebind Messages

At time T2 for an IA (which will only be reached if the server to which the Renew message was sent at time T1 has not responded), the client initiates a Rebind/Reply message exchange with any available server. The client includes an IA option with all addresses currently assigned to the IA in its Rebind message.

The client sets the "msg-type" field to REBIND. The client generates a transaction ID and inserts this value in the "transaction-id" field.

The client MUST include a Client Identifier option to identify itself to the server. The client adds any appropriate options, including one or more IA options. The client MUST include the list of addresses the client currently has associated with the IAs in the Rebind message.

The client MUST include an Option Request option (see Section 23.7) to indicate the options the client is interested in receiving. The client MAY include options with data values as hints to the server about parameter values the client would like to have returned.

The client transmits the message according to Section 15, using the following parameters:

IRT	REB_TIMEOUT
MRT	REB_MAX_RT
MRC	0
MRD	Remaining time until valid lifetimes of all addresses have expired

The message exchange is terminated when the valid lifetimes of all the addresses assigned to the IA expire (see Section 11), at which time the client has several alternative actions to choose from; for example:

- The client may choose to use a Solicit message to locate a new DHCP server and send a Request for the expired IA to the new server.

- The client may have other addresses in other IAs, so the client may choose to discard the expired IA and use the addresses in the other IAs.

19.1.5. Creation and Transmission of Information-request Messages

The client uses an Information-request message to obtain configuration information without having addresses assigned to it.

The client sets the "msg-type" field to INFORMATION-REQUEST. The client generates a transaction ID and inserts this value in the "transaction-id" field.

The client SHOULD include a Client Identifier option to identify itself to the server. If the client does not include a Client Identifier option, the server will not be able to return any client-specific options to the client, or the server may choose not to respond to the message at all. The client MUST include a Client Identifier option if the Information-Request message will be authenticated.

The client MUST include an Option Request option (see Section 23.7) to request the INF_MAX_RT option (see Section 23.24) and any other options the client is interested in receiving. The client MAY include options with data values as hints to the server about parameter values the client would like to have returned.

The first Information-request message from the client on the interface MUST be delayed by a random amount of time between 0 and INF_MAX_DELAY. The client transmits the message according to Section 15, using the following parameters:

IRT	INF_TIMEOUT
MRT	INF_MAX_RT
MRC	0
MRD	0

19.1.6. Creation and Transmission of Release Messages

To release one or more addresses, a client sends a Release message to the server.

The client sets the "msg-type" field to RELEASE. The client generates a transaction ID and places this value in the "transaction-id" field.

The client places the identifier of the server that allocated the address(es) in a Server Identifier option.

The client MUST include a Client Identifier option to identify itself to the server. The client includes options containing the IAs for the addresses it is releasing in the "options" field. The addresses to be released MUST be included in the IAs. Any addresses for the IAs the client wishes to continue to use MUST NOT be added to the IAs.

The client MUST NOT use any of the addresses it is releasing as the source address in the Release message or in any subsequently transmitted message.

Because Release messages may be lost, the client should retransmit the Release if no Reply is received. However, there are scenarios where the client may not wish to wait for the normal retransmission timeout before giving up (e.g., on power down). Implementations SHOULD retransmit one or more times, but MAY choose to terminate the retransmission procedure early.

The client transmits the message according to Section 15, using the following parameters:

IRT	REL_TIMEOUT
MRT	0
MRC	REL_MAX_RC
MRD	0

The client MUST stop using all of the addresses being released as soon as the client begins the Release message exchange process. If addresses are released but the Reply from a DHCP server is lost, the client will retransmit the Release message, and the server may respond with a Reply indicating a status of NoBinding. Therefore, the client does not treat a Reply message with a status of NoBinding in a Release message exchange as if it indicates an error.

Note that if the client fails to release the addresses, each address assigned to the IA will be reclaimed by the server when the valid lifetime of that address expires.

19.1.7. Creation and Transmission of Decline Messages

If a client detects that one or more addresses assigned to it by a server are already in use by another node, the client sends a Decline message to the server to inform it that the address is suspect.

The client sets the "msg-type" field to DECLINE. The client generates a transaction ID and places this value in the "transaction-id" field.

The client places the identifier of the server that allocated the address(es) in a Server Identifier option.

The client MUST include a Client Identifier option to identify itself to the server. The client includes options containing the IAs for the addresses it is declining in the "options" field. The addresses to be declined MUST be included in the IAs. Any addresses for the IAs the client wishes to continue to use should not be in added to the IAs.

The client MUST NOT use any of the addresses it is declining as the source address in the Decline message or in any subsequently transmitted message.

The client transmits the message according to Section 15, using the following parameters:

IRT	DEC_TIMEOUT
MRT	0
MRC	DEC_MAX_RC
MRD	0

If addresses are declined but the Reply from a DHCP server is lost, the client will retransmit the Decline message, and the server may respond with a Reply indicating a status of NoBinding. Therefore, the client does not treat a Reply message with a status of NoBinding in a Decline message exchange as if it indicates an error.

19.1.8. Receipt of Reply Messages

Upon the receipt of a valid Reply message in response to a Solicit (with a Rapid Commit option), Request, Confirm, Renew, Rebind or Information-request message, the client extracts the configuration information contained in the Reply. The client MAY choose to report any status code or message from the status code option in the Reply message.

The client SHOULD perform duplicate address detection [RFC4862] on each of the addresses in any IAs it receives in the Reply message before using that address for traffic. If any of the addresses are found to be in use on the link, the client sends a Decline message to the server as described in Section 19.1.7.

If the Reply was received in response to a Solicit (with a Rapid Commit option), Request, Renew or Rebind message, the client updates the information it has recorded about IAs from the IA options contained in the Reply message:

- Record T1 and T2 times.
- Add any new addresses in the IA option to the IA as recorded by the client.
- Update lifetimes for any addresses in the IA option that the client already has recorded in the IA.
- Discard any addresses from the IA, as recorded by the client, that have a valid lifetime of 0 in the IA Address option.
- Leave unchanged any information about addresses the client has recorded in the IA but that were not included in the IA from the server.

Management of the specific configuration information is detailed in the definition of each option in Section 23.

If the client receives a Reply message with a Status Code containing UnspecFail, the server is indicating that it was unable to process the message due to an unspecified failure condition. If the client retransmits the original message to the same server to retry the desired operation, the client MUST limit the rate at which it retransmits the message and limit the duration of the time during which it retransmits the message (see Section 14.1).

When the client receives a Reply message with a Status Code option with the value UseMulticast, the client records the receipt of the message and sends subsequent messages to the server through the interface on which the message was received using multicast. The client resends the original message using multicast.

When the client receives a NotOnLink status from the server in response to a Confirm message, the client performs DHCP server solicitation, as described in Section 18, and client-initiated configuration as described in Section 19. If the client receives any Reply messages that do not indicate a NotOnLink status, the client can use the addresses in the IA and ignore any messages that indicate a NotOnLink status.

When the client receives a NotOnLink status from the server in response to a Solicit (with a Rapid Commit option) or a Request, the client can either re-issue the Request without specifying any addresses or restart the DHCP server discovery process (see Section 18).

The client examines the status code in each IA individually. If the status code is NoAddrsAvail, the client has received no usable addresses in the IA and may choose to try obtaining addresses for the IA from another server. The client uses addresses and other information from any IAs that do not contain a Status Code option with the NoAddrsAvail code. If the client receives no addresses in any of the IAs, it may either try another server (perhaps restarting the DHCP server discovery process) or use the Information-request message to obtain other configuration information only.

Whenever a client restarts the DHCP server discovery process or selects an alternate server, as described in Section 18.1.3, the client SHOULD stop using all the addresses and delegated prefixes for which it has the bindings and try to obtain all required adresses and prefixes from the new server. This facilitates the client using a single state machine for all bindings.

When the client receives a Reply message in response to a Renew or Rebind message, the client examines each IA independently. For each IA in the original Renew or Rebind message, the client:

- sends a Request message if the IA contained a Status Code option with the NoBinding status (and does not send any additional Renew/Rebind messages)

- sends a Renew/Rebind if the IA is not in the Reply message

- otherwise accepts the information in the IA

When the client receives a valid Reply message in response to a Release message, the client considers the Release event completed, regardless of the Status Code option(s) returned by the server.

When the client receives a valid Reply message in response to a Decline message, the client considers the Decline event completed, regardless of the Status Code option(s) returned by the server.

19.2. Server Behavior

For this discussion, the Server is assumed to have been configured in an implementation specific manner with configuration of interest to clients.

In most instances, the server will send a Reply in response to a client message. This Reply message MUST always contain the Server Identifier option containing the server’s DUID and the Client Identifier option from the client message if one was present.

In most Reply messages, the server includes options containing configuration information for the client. The server must be aware of the recommendations on packet sizes and the use of fragmentation in section 5 of [RFC2460]. If the client included an Option Request option in its message, the server includes options in the Reply message containing configuration parameters for all of the options identified in the Option Request option that the server has been configured to return to the client. The server MAY return additional options to the client if it has been configured to do so.

19.2.1. Receipt of Request Messages

When the server receives a Request message via unicast from a client to which the server has not sent a unicast option, the server discards the Request message and responds with a Reply message containing a Status Code option with the value UseMulticast, a Server Identifier option containing the server’s DUID, the Client Identifier option from the client message, and no other options.

When the server receives a valid Request message, the server creates the bindings for that client according to the server’s policy and configuration information and records the IAs and other information requested by the client.

The server constructs a Reply message by setting the "msg-type" field to REPLY, and copying the transaction ID from the Request message into the transaction-id field.

The server MUST include a Server Identifier option containing the server’s DUID and the Client Identifier option from the Request message in the Reply message.

If the server finds that the prefix on one or more IP addresses in any IA in the message from the client is not appropriate for the link to which the client is connected, the server MUST return the IA to the client with a Status Code option with the value NotOnLink.

If the server cannot assign any addresses to an IA in the message from the client, the server MUST include the IA in the Reply message with no addresses in the IA and a Status Code option in the IA containing status code NoAddrsAvail.

For any IAs to which the server can assign addresses, the server includes the IA with addresses and other configuration parameters, and records the IA as a new client binding.

The server includes a Reconfigure Accept option if the server wants to require that the client accept Reconfigure messages.

The server includes other options containing configuration information to be returned to the client as described in Section 19.2.

If the server finds that the client has included an IA in the Request message for which the server already has a binding that associates the IA with the client, the client has resent a Request message for which it did not receive a Reply message. The server either resends a previously cached Reply message or sends a new Reply message.

19.2.2. Receipt of Confirm Messages

When the server receives a Confirm message, the server determines whether the addresses in the Confirm message are appropriate for the link to which the client is attached. If all of the addresses in the Confirm message pass this test, the server returns a status of Success. If any of the addresses do not pass this test, the server returns a status of NotOnLink. If the server is unable to perform this test (for example, the server does not have information about prefixes on the link to which the client is connected), or there were no addresses in any of the IAs sent by the client, the server MUST NOT send a reply to the client.

The server ignores the T1 and T2 fields in the IA options and the preferred-lifetime and valid-lifetime fields in the IA Address options.

The server constructs a Reply message by setting the "msg-type" field to REPLY, and copying the transaction ID from the Confirm message into the transaction-id field.

The server MUST include a Server Identifier option containing the server's DUID and the Client Identifier option from the Confirm message in the Reply message. The server includes a Status Code option indicating the status of the Confirm message.

19.2.3. Receipt of Renew Messages

When the server receives a Renew message via unicast from a client to which the server has not sent a unicast option, the server discards the Renew message and responds with a Reply message containing a Status Code option with the value UseMulticast, a Server Identifier option containing the server's DUID, the Client Identifier option from the client message, and no other options.

When the server receives a Renew message that contains an IA option from a client, it locates the client's binding and verifies that the information in the IA from the client matches the information stored for that client.

If the server cannot find a client entry for the IA the server returns the IA containing no addresses with a Status Code option set to NoBinding in the Reply message.

If the server finds that any of the addresses are not appropriate for the link to which the client is attached, the server returns the address to the client with lifetimes of 0.

If the server finds the addresses in the IA for the client then the server sends back the IA to the client with new lifetimes and T1/T2 times. The server may choose to change the list of addresses and the lifetimes of addresses in IAs that are returned to the client.

The server constructs a Reply message by setting the "msg-type" field to REPLY, and copying the transaction ID from the Renew message into the transaction-id field.

The server MUST include a Server Identifier option containing the server's DUID and the Client Identifier option from the Renew message in the Reply message.

The server includes other options containing configuration information to be returned to the client as described in Section 19.2.

19.2.4. Receipt of Rebind Messages

When the server receives a Rebind message that contains an IA option from a client, it locates the client's binding and verifies that the information in the IA from the client matches the information stored for that client.

If the server cannot find a client entry for the IA and the server determines that the addresses in the IA are not appropriate for the link to which the client's interface is attached according to the server's explicit configuration information, the server MAY send a Reply message to the client containing the client's IA, with the lifetimes for the addresses in the IA set to zero. This Reply constitutes an explicit notification to the client that the addresses in the IA are no longer valid. In this situation, if the server does not send a Reply message it discards the Rebind message.

If the server finds that any of the addresses are no longer appropriate for the link to which the client is attached, the server returns the address to the client with lifetimes of 0.

If the server finds the addresses in the IA for the client then the server SHOULD send back the IA to the client with new lifetimes and T1/T2 times.

The server constructs a Reply message by setting the "msg-type" field to REPLY, and copying the transaction ID from the Rebind message into the transaction-id field.

The server MUST include a Server Identifier option containing the server's DUID and the Client Identifier option from the Rebind message in the Reply message.

The server includes other options containing configuration information to be returned to the client as described in Section 19.2.

19.2.5. Receipt of Information-request Messages

When the server receives an Information-request message, the client is requesting configuration information that does not include the assignment of any addresses. The server determines all configuration parameters appropriate to the client, based on the server configuration policies known to the server.

The server constructs a Reply message by setting the "msg-type" field to REPLY, and copying the transaction ID from the Information-request message into the transaction-id field.

The server MUST include a Server Identifier option containing the server’s DUID in the Reply message. If the client included a Client Identification option in the Information-request message, the server copies that option to the Reply message.

The server includes options containing configuration information to be returned to the client as described in Section 19.2.

If the Information-request message received from the client did not include a Client Identifier option, the server SHOULD respond with a Reply message containing any configuration parameters that are not determined by the client’s identity. If the server chooses not to respond, the client may continue to retransmit the Information-request message indefinitely.

19.2.6. Receipt of Release Messages

When the server receives a Release message via unicast from a client to which the server has not sent a unicast option, the server discards the Release message and responds with a Reply message containing a Status Code option with value UseMulticast, a Server Identifier option containing the server’s DUID, the Client Identifier option from the client message, and no other options.

Upon the receipt of a valid Release message, the server examines the IAs and the addresses in the IAs for validity. If the IAs in the message are in a binding for the client, and the addresses in the IAs have been assigned by the server to those IAs, the server deletes the addresses from the IAs and makes the addresses available for assignment to other clients. The server ignores addresses not assigned to the IA, although it may choose to log an error.

After all the addresses have been processed, the server generates a Reply message and includes a Status Code option with value Success, a Server Identifier option with the server’s DUID, and a Client Identifier option with the client’s DUID. For each IA in the Release message for which the server has no binding information, the server adds an IA option using the IAID from the Release message, and includes a Status Code option with the value NoBinding in the IA option. No other options are included in the IA option.

A server may choose to retain a record of assigned addresses and IAs after the lifetimes on the addresses have expired to allow the server to reassign the previously assigned addresses to a client.

19.2.7. Receipt of Decline Messages

When the server receives a Decline message via unicast from a client to which the server has not sent a unicast option, the server discards the Decline message and responds with a Reply message containing a Status Code option with the value UseMulticast, a Server Identifier option containing the server’s DUID, the Client Identifier option from the client message, and no other options.

Upon the receipt of a valid Decline message, the server examines the IAs and the addresses in the IAs for validity. If the IAs in the message are in a binding for the client, and the addresses in the IAs have been assigned by the server to those IAs, the server deletes the addresses from the IAs. The server ignores addresses not assigned to the IA (though it may choose to log an error if it finds such an address).

The client has found any addresses in the Decline messages to be already in use on its link. Therefore, the server SHOULD mark the addresses declined by the client so that those addresses are not assigned to other clients, and MAY choose to make a notification that addresses were declined. Local policy on the server determines when the addresses identified in a Decline message may be made available for assignment.

After all the addresses have been processed, the server generates a Reply message and includes a Status Code option with the value Success, a Server Identifier option with the server’s DUID, and a Client Identifier option with the client’s DUID. For each IA in the Decline message for which the server has no binding information, the server adds an IA option using the IAID from the Decline message and includes a Status Code option with the value NoBinding in the IA option. No other options are included in the IA option.

19.2.8. Transmission of Reply Messages

If the original message was received directly by the server, the server unicasts the Reply message directly to the client using the address in the source address field from the IP datagram in which the original message was received. The Reply message MUST be unicast through the interface on which the original message was received.

If the original message was received in a Relay-forward message, the server constructs a Relay-reply message with the Reply message in the payload of a Relay Message option (see Section 23.10). If the Relay-forward messages included an Interface-id option, the server copies that option to the Relay-reply message. The server unicasts the Relay-reply message directly to the relay agent using the address in the source address field from the IP datagram in which the Relay-forward message was received.

19.3. Requesting Router Behavior for Prefix Delegation

The requesting router uses a Request message to populate IA_PDs with prefixes. The requesting router includes one or more IA_PD options in the Request message. The delegating router then returns the prefixes for the IA_PDs to the requesting router in IA_PD options in a Reply message.

The requesting router includes IA_PD options in any Renew, or Rebind messages sent by the requesting router. The IA_PD option includes all of the prefixes the requesting router currently has associated with that IA_PD.

In some circumstances the requesting router may need verification that the delegating router still has a valid binding for the requesting router. Examples of times when a requesting router may ask for such verification include:

- The requesting router reboots.

- The requesting router’s upstream link flaps.

- The requesting router is physically disconnected from a wired connection.

If such verification is needed the requesting router MUST initiate a Rebind/Reply message exchange as described in section Section 19.1.4, with the exception that the retransmission parameters should be set as for the Confirm message, described in Section 19.1.2. The requesting router includes any IA_PDs, along with prefixes associated with those IA_PDs in its Rebind message.

Each prefix has valid and preferred lifetimes whose durations are specified in the IA_PD Prefix option for that prefix. The requesting router uses Renew and Rebind messages to request the extension of the lifetimes of a delegated prefix.

The requesting router uses a Release message to return a delegated prefix to a delegating router. The prefixes to be released MUST be included in the IA_PDs.

The Confirm and Decline message types are not used with Prefix Delegation.

Upon the receipt of a valid Reply message, for each IA_PD the requesting router assigns a subnet from each of the delegated prefixes to each of the links to which the associated interfaces are attached.

When the Delegating Router delegates prefixes to a Requesting Router, the Requesting Router has sole authority for assignment of those prefixes, and the Delegating Router MUST NOT assign any prefixes from that delegated prefix to any of its own links.

When a requesting router subnets a delegated prefix, it must assign additional bits to the prefix to generate unique, longer prefixes. For example, if the requesting router in Figure 1 were delegated 3FFE:FFFF:0::/48, it might generate 3FFE:FFFF:0:1::/64 and 3FFE:FFFF:0:2::/64 for assignment to the two links in the subscriber network. If the requesting router were delegated 3FFE:FFFF:0::/48 and 3FFE:FFFF:5::/48, it might assign 3FFE:FFFF:0:1::/64 and 3FFE:FFFF:5:1::/64 to one of the links, and 3FFE:FFFF:0:2::/64 and 3FFE:FFFF:5:2::/64 for assignment to the other link.

If the requesting router assigns a delegated prefix to a link to which the router is attached, and begins to send router advertisements for the prefix on the link, the requesting router MUST set the valid lifetime in those advertisements to be no later than the valid lifetime specified in the IA_PD Prefix option. A requesting router MAY use the preferred lifetime specified in the IA_PD Prefix option.

Handling of Status Codes options in received Reply messages is described in section Section 19.1.8. The NoPrefixAvail Status Code is handled in the same manner as the NoAddrsAvail Status Code.

19.4. Delegating Router Behavior for Prefix Delegation

When a delegating router receives a Request message from a requesting router that contains an IA_PD option, and the delegating router is authorized to delegate prefix(es) to the requesting router, the delegating router selects the prefix(es) to be delegated to the requesting router. The mechanism through which the delegating router selects prefix(es) for delegation is not specified in this document. Section 18.4 gives examples of ways in which a delegating router might select the prefix(es) to be delegated to a requesting router.

A delegating router examines the prefix(es) identified in IA_PD Prefix options (in an IA_PD option) in Renew and Rebind messages and responds according to the current status of the prefix(es). The delegating router returns IA_PD Prefix options (within an IA_PD option) with updated lifetimes for each valid prefix in the message from the requesting router. If the delegating router finds that any of the prefixes are not in the requesting router’s binding entry, the delegating router returns the prefix to the requesting router with lifetimes of 0.

The delegating router behaves as follows when it cannot find a binding for the requesting router’s IA_PD:

Renew message: If the delegating router cannot find a binding for the requesting router’s IA_PD the delegating router returns the IA_PD containing no prefixes with a Status Code option set to NoBinding in the Reply message.

Rebind message: If the delegating router cannot find a binding for the requesting router’s IA_PD and the delegating router determines that the prefixes in the IA_PD are not appropriate for the link to which the requesting router’s interface is attached according to the delegating routers explicit configuration, the delegating router MAY send a Reply message to the requesting router containing the IA_PD with the lifetimes of the prefixes in the IA_PD set to zero. This Reply constitutes an explicit notification to the requesting router that the prefixes in the IA_PD are no longer valid. If the delegating router is unable to determine if the prefix is not appropriate for the link, the Rebind message is discarded.

A delegating router may mark any prefix(es) in IA_PD Prefix options in a Release message from a requesting router as "available", dependent on the mechanism used to acquire the prefix, e.g., in the case of a dynamic pool.

The delegating router MUST include an IA_PD Prefix option or options (in an IA_PD option) in Reply messages sent to a requesting router.

20. DHCP Server-Initiated Configuration Exchange

A server initiates a configuration exchange to cause DHCP clients to obtain new addresses and other configuration information. For example, an administrator may use a server-initiated configuration exchange when links in the DHCP domain are to be renumbered. Other examples include changes in the location of directory servers, addition of new services such as printing, and availability of new software.

20.1. Server Behavior

A server sends a Reconfigure message to cause a client to initiate immediately a Renew/Reply or Information-request/Reply message exchange with the server.

20.1.1. Creation and Transmission of Reconfigure Messages

The server sets the "msg-type" field to RECONFIGURE. The server sets the transaction-id field to 0. The server includes a Server Identifier option containing its DUID and a Client Identifier option containing the client’s DUID in the Reconfigure message.

The server MAY include an Option Request option to inform the client of what information has been changed or new information that has been added. In particular, the server specifies the IA option in the Option Request option if the server wants the client to obtain new address information. If the server identifies the IA option in the Option Request option, the server MUST include an IA option to identify each IA that is to be reconfigured on the client. The IA options included by the server MUST NOT contain any options.

Because of the risk of denial of service attacks against DHCP clients, the use of a security mechanism is mandated in Reconfigure messages. The server MUST use DHCP authentication in the Reconfigure message.

The server MUST include a Reconfigure Message option (defined in Section 23.19) to select whether the client responds with a Renew message, a Rebind message, or an Information-Request message.

The server MUST NOT include any other options in the Reconfigure except as specifically allowed in the definition of individual options.

A server sends each Reconfigure message to a single DHCP client, using an IPv6 unicast address of sufficient scope belonging to the DHCP client. If the server does not have an address to which it can send the Reconfigure message directly to the client, the server uses a Relay-reply message (as described in Section 21.3) to send the Reconfigure message to a relay agent that will relay the message to the client. The server may obtain the address of the client (and the appropriate relay agent, if required) through the information the server has about clients that have been in contact with the server, or through some external agent.

To reconfigure more than one client, the server unicasts a separate message to each client. The server may initiate the reconfiguration of multiple clients concurrently; for example, a server may send a Reconfigure message to additional clients while previous reconfiguration message exchanges are still in progress.

The Reconfigure message causes the client to initiate a Renew/Reply, a Rebind/Reply, or Information-request/Reply message exchange with the server. The server interprets the receipt of a Renew, a Rebind, or Information-request message (whichever was specified in the original Reconfigure message) from the client as satisfying the Reconfigure message request.

20.1.2. Time Out and Retransmission of Reconfigure Messages

If the server does not receive a Renew, Rebind, or Information-request message from the client in REC_TIMEOUT milliseconds, the server retransmits the Reconfigure message, doubles the REC_TIMEOUT value and waits again. The server continues this process until REC_MAX_RC unsuccessful attempts have been made, at which point the server SHOULD abort the reconfigure process for that client.

Default and initial values for REC_TIMEOUT and REC_MAX_RC are documented in Section 6.5.

20.2. Receipt of Renew or Rebind Messages

In response to a Renew message, the server generates and sends a Reply message to the client as described in Section 19.2.3 and Section 19.2.8, including options for configuration parameters.

In response to a Rebind message, the server generates and sends a Reply message to the client as described in Section 19.2.4 and Section 19.2.8, including options for configuration parameters.

The server MAY include options containing the IAs and new values for other configuration parameters in the Reply message, even if those IAs and parameters were not requested in the Renew or Rebind message from the client.

20.3. Receipt of Information-request Messages

The server generates and sends a Reply message to the client as described in Section 19.2.5 and Section 19.2.8, including options for configuration parameters.

The server MAY include options containing new values for other configuration parameters in the Reply message, even if those parameters were not requested in the Information-request message from the client.

20.4. Client Behavior

A client receives Reconfigure messages sent to the UDP port 546 on interfaces for which it has acquired configuration information through DHCP. These messages may be sent at any time. Since the results of a reconfiguration event may affect application layer programs, the client SHOULD log these events, and MAY notify these programs of the change through an implementation-specific interface.

20.4.1. Receipt of Reconfigure Messages

Upon receipt of a valid Reconfigure message, the client responds with either a Renew message, a Rebind message, or an Information-request message as indicated by the Reconfigure Message option (as defined in Section 23.19). The client ignores the transaction-id field in the received Reconfigure message. While the transaction is in progress, the client discards any Reconfigure messages it receives.

DISCUSSION:

The Reconfigure message acts as a trigger that signals the client to complete a successful message exchange. Once the client has received a Reconfigure, the client proceeds with the message exchange (retransmitting the Renew or Information-request message if necessary); the client ignores any additional Reconfigure messages until the exchange is complete. Subsequent Reconfigure messages cause the client to initiate a new exchange.

How does this mechanism work in the face of duplicated or retransmitted Reconfigure messages? Duplicate messages will be ignored because the client will begin the exchange after the receipt of the first Reconfigure. Retransmitted messages will either trigger the exchange (if the first Reconfigure was not received by the client) or will be ignored. The server can discontinue retransmission of Reconfigure messages to the client once the server receives the Renew or Information-request message from the client.

It might be possible for a duplicate or retransmitted Reconfigure to be sufficiently delayed (and delivered out of order) to arrive at the client after the exchange (initiated by the original Reconfigure) has been completed. In this case, the client would initiate a redundant exchange. The likelihood of delayed and out of order delivery is small enough to be ignored. The consequence of the redundant exchange is inefficiency rather than incorrect operation.

20.4.2. Creation and Transmission of Renew or Rebind Messages

When responding to a Reconfigure, the client creates and sends the Renew message in exactly the same manner as outlined in Section 19.1.3, with the exception that the client copies the Option Request option and any IA options from the Reconfigure message into the Renew message. The client MUST include a Server Identifier option in the Renew message, identifying the server with which the client most recently communicated.

When responding to a Reconfigure, the client creates and sends the Rebind message in exactly the same manner as outlined in Section 19.1.4, with the exception that the client copies the Option Request option and any IA options from the Reconfigure message into the Rebind message.

If a client is currently sending Rebind messages, as described in Section 19.1.3, the client ignores any received Reconfigure messages.

20.4.3. Creation and Transmission of Information-request Messages

When responding to a Reconfigure, the client creates and sends the Information-request message in exactly the same manner as outlined in Section 19.1.5, with the exception that the client includes a Server Identifier option with the identifier from the Reconfigure message to which the client is responding.

20.4.4. Time Out and Retransmission of Renew, Rebind or Information-request Messages

The client uses the same variables and retransmission algorithm as it does with Renew, Rebind, or Information-request messages generated as part of a client-initiated configuration exchange. See Section 19.1.3, Section 19.1.4, and Section 19.1.5 for details. If the client does not receive a response from the server by the end of the retransmission process, the client ignores and discards the Reconfigure message.

20.4.5. Receipt of Reply Messages

Upon the receipt of a valid Reply message, the client processes the options and sets (or resets) configuration parameters appropriately. The client records and updates the lifetimes for any addresses specified in IAs in the Reply message.

20.5. Prefix Delegation Reconfiguration

This section describes prefix delegation in Reconfigure message exchanges.

20.5.1. Delegating Router Behavior

The delegating router initiates a configuration message exchange with a requesting router, as described in Section 20, by sending a Reconfigure message (acting as a DHCP server) to the requesting router, as described in Section 20.1. The delegating router specifies the IA_PD option in the Option Request option to cause the requesting router to include an IA_PD option to obtain new information about delegated prefix(es).

20.5.2. Requesting Router Behavior

The requesting router responds to a Reconfigure message, acting as a DHCP client, received from a delegating router as described in Section 20.4 The requesting router MUST include the IA_PD Prefix option(s) (in an IA_PD option) for prefix(es) that have been delegated to the requesting router by the delegating router from which the Reconfigure message was received.

21. Relay Agent Behavior

The relay agent MAY be configured to use a list of destination addresses, which MAY include unicast addresses, the All_DHCP_Servers multicast address, or other addresses selected by the network administrator. If the relay agent has not been explicitly configured, it MUST use the All_DHCP_Servers multicast address as the default.

If the relay agent relays messages to the All_DHCP_Servers multicast address or other multicast addresses, it sets the Hop Limit field to 32.

If the relay agent receives a message other than Relay-forward and Relay-reply and the relay agent does not recognize its message type, it MUST forward them as described in Section 21.1.1.

21.1. Relaying a Client Message or a Relay-forward Message

A relay agent relays both messages from clients and Relay-forward messages from other relay agents. When a relay agent receives a valid message (for a definition of a valid message, see Section 4.1 of [RFC7283]) to be relayed, it constructs a new Relay-forward message. The relay agent copies the source address from the header of the IP datagram in which the message was received to the peer-address field of the Relay-forward message. The relay agent copies the received DHCP message (excluding any IP or UDP headers) into a Relay Message option in the new message. The relay agent adds to the Relay-forward message any other options it is configured to include.

[RFC6221] defines a Lightweight DHCPv6 Relay Agent (LDRA) that allows Relay Agent Information to be inserted by an access node that performs a link- layer bridging (i.e., non-routing) function.

21.1.1. Relaying a Message from a Client

If the relay agent received the message to be relayed from a client, the relay agent places a global, ULA [RFC4193] or site-scoped address with a prefix assigned to the link on which the client should be assigned an address in the link-address field. (It is possible for the relay to use link local address instead, but that is not recommended as it would require additional information to be provided in the server configuration. See Section 3.2 of [I-D.ietf-dhc-topo-conf] for detailed discussion.) This address will be used by the server to determine the link from which the client should be assigned an address and other configuration information. The hop-count in the Relay-forward message is set to 0.

If the relay agent cannot use the address in the link-address field to identify the interface through which the response to the client will be relayed, the relay agent MUST include an Interface-id option (see Section 23.18) in the Relay-forward message. The server will include the Interface-id option in its Relay-reply message. The relay agent fills in the link-address field as described in the previous paragraph regardless of whether the relay agent includes an Interface-id option in the Relay-forward message.

21.1.2. Relaying a Message from a Relay Agent

If the message received by the relay agent is a Relay-forward message and the hop-count in the message is greater than or equal to HOP_COUNT_LIMIT, the relay agent discards the received message.

The relay agent copies the source address from the IP datagram in which the message was received from the relay agent into the peer-address field in the Relay-forward message and sets the hop-count field to the value of the hop-count field in the received message incremented by 1.

If the source address from the IP datagram header of the received message is a global or site-scoped address (and the device on which the relay agent is running belongs to only one site), the relay agent sets the link-address field to 0; otherwise the relay agent sets the link-address field to a global or site-scoped address assigned to the interface on which the message was received, or includes an Interface-ID option to identify the interface on which the message was received.

21.1.3. Relay Agent Behavior with Prefix Delegation

A relay agent forwards messages containing Prefix Delegation options in the same way as described earlier in this section.

If a delegating router communicates with a requesting router through a relay agent, the delegating router may need a protocol or other out-of-band communication to configure routing information for delegated prefixes on any router through which the requesting router may forward traffic.

21.2. Relaying a Relay-reply Message

The relay agent processes any options included in the Relay-reply message in addition to the Relay Message option, and then discards those options.

The relay agent extracts the message from the Relay Message option and relays it to the address contained in the peer-address field of the Relay-reply message. Relay agents MUST NOT modify the message.

If the Relay-reply message includes an Interface-id option, the relay agent relays the message from the server to the client on the link identified by the Interface-id option. Otherwise, if the link-address field is not set to zero, the relay agent relays the message on the link identified by the link-address field.

If the relay agent receives a Relay-reply message, it MUST process the message as defined above, regardless of the type of message encapsulated in the Relay Message option.

21.3. Construction of Relay-reply Messages

A server uses a Relay-reply message to return a response to a client if the original message from the client was relayed to the server in a Relay-forward message or to send a Reconfigure message to a client if the server does not have an address it can use to send the message directly to the client.

A response to the client MUST be relayed through the same relay agents as the original client message. The server causes this to happen by creating a Relay-reply message that includes a Relay

Message option containing the message for the next relay agent in the return path to the client. The contained Relay-reply message contains another Relay Message option to be sent to the next relay agent, and so on. The server must record the contents of the peer-address fields in the received message so it can construct the appropriate Relay-reply message carrying the response from the server.

For example, if client C sent a message that was relayed by relay agent A to relay agent B and then to the server, the server would send the following Relay-Reply message to relay agent B:

```
msg-type:       RELAY-REPLY
hop-count:      1
link-address:   0
peer-address:   A
Relay Message option, containing:
  msg-type:     RELAY-REPLY
  hop-count:    0
  link-address: address from link to which C is attached
  peer-address: C
  Relay Message option: <response from server>
```

Figure 8: Relay-reply Example

When sending a Reconfigure message to a client through a relay agent, the server creates a Relay-reply message that includes a Relay Message option containing the Reconfigure message for the next relay agent in the return path to the client. The server sets the peer-address field in the Relay-reply message header to the address of the client, and sets the link-address field as required by the relay agent to relay the Reconfigure message to the client. The server obtains the addresses of the client and the relay agent through prior interaction with the client or through some external mechanism.

22. Authentication of DHCP Messages

Some network administrators may wish to provide authentication of the source and contents of DHCP messages. For example, clients may be subject to denial of service attacks through the use of bogus DHCP servers, or may simply be misconfigured due to unintentionally instantiated DHCP servers. Network administrators may wish to constrain the allocation of addresses to authorized hosts to avoid denial of service attacks in "hostile" environments where the network medium is not physically secured, such as wireless networks or college residence halls.

The DHCP authentication mechanism is based on the design of authentication for DHCPv4 [RFC3118].

22.1. Security of Messages Sent Between Servers and Relay Agents

Relay agents and servers that exchange messages securely use the IPsec mechanisms for IPv6 [RFC4301]. If a client message is relayed through multiple relay agents, each of the relay agents must have established independent, pairwise trust relationships. That is, if messages from client C will be relayed by relay agent A to relay agent B and then to the server, relay agents A and B must be configured to use IPsec for the messages they exchange, and relay agent B and the server must be configured to use IPsec for the messages they exchange.

Relay agents and servers that support secure relay agent to server or relay agent to relay agent communication use IPsec under the following conditions:

Selectors

Relay agents are manually configured with the addresses of the relay agent or server to which DHCP messages are to be forwarded. Each relay agent and server that will be using IPsec for securing DHCP messages must also be configured with a list of the relay agents to which messages will be returned. The selectors for the relay agents and servers will be the pairs of addresses defining relay agents and servers that exchange DHCP messages on DHCPv6 UDP port 547.

Mode

Relay agents and servers use transport mode and ESP. The information in DHCP messages is not generally considered confidential, so encryption need not be used (i.e., NULL encryption can be used).

Key management

Because the relay agents and servers are used within an organization, public key schemes are not necessary. Because the relay agents and servers must be manually configured, manually configured key management may suffice, but does not provide defense against replayed messages. Accordingly, IKE with preshared secrets SHOULD be supported. IKE with public keys MAY be supported.

Security policy DHCP messages between relay agents and servers should only be accepted from DHCP peers as identified in the local configuration.

Authentication Shared keys, indexed to the source IP address of the received DHCP message, are adequate in this application.

Availability Appropriate IPsec implementations are likely to be available for servers and for relay agents in more featureful devices used in enterprise and core ISP networks. IPsec is less likely to be available for relay agents in low end devices primarily used in the home or small office markets.

22.2. Summary of DHCP Authentication

Authentication of DHCP messages is accomplished through the use of the Authentication option (see Section 23.11). The authentication information carried in the Authentication option can be used to reliably identify the source of a DHCP message and to confirm that the contents of the DHCP message have not been tampered with.

The Authentication option provides a framework for multiple authentication protocols. Two such protocols are defined here. Other protocols defined in the future will be specified in separate documents.

Any DHCP message MUST NOT include more than one Authentication option.

The protocol field in the Authentication option identifies the specific protocol used to generate the authentication information carried in the option. The algorithm field identifies a specific algorithm within the authentication protocol; for example, the algorithm field specifies the hash algorithm used to generate the message authentication code (MAC) in the authentication option. The replay detection method (RDM) field specifies the type of replay detection used in the replay detection field.

22.3. Replay Detection

The Replay Detection Method (RDM) field determines the type of replay detection used in the Replay Detection field.

If the RDM field contains 0x00, the replay detection field MUST be set to the value of a strictly monotonically increasing counter. Using a counter value, such as the current time of day (for example, an NTP-format timestamp [RFC5905]), can reduce the danger of replay attacks. This method MUST be supported by all protocols.

22.4. Delayed Authentication Protocol

If the protocol field is 2, the message is using the "delayed authentication" mechanism. In delayed authentication, the client requests authentication in its Solicit message, and the server replies with an Advertise message that includes authentication information. This authentication information contains a nonce value generated by the source as a message authentication code (MAC) to provide message authentication and entity authentication.

Note that the delayed authentication protocol cannot work with 2-message exchange model. This protocol uses Solicit/Advertise exchange as the key and server selection process. So, real DHCPv6 procedures can only be made in the follow-up messages.

The use of a particular technique based on the HMAC protocol [RFC2104] using the MD5 hash [RFC1321] is defined here.

22.4.1. Use of the Authentication Option in the Delayed Authentication Protocol

In a Solicit message, the client fills in the protocol, algorithm and RDM fields in the Authentication option with the client’s preferences. The client sets the replay detection field to zero and omits the authentication information field. The client sets the option-len field to 11.

In all other messages, the protocol and algorithm fields identify the method used to construct the contents of the authentication information field. The RDM field identifies the method used to construct the contents of the replay detection field.

The format of the Authentication information is:

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                          DHCP realm                           |
    |                      (variable length)                        |
    .                                                               .
    .                                                               .
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                      key ID (32 bits)                         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    |                           HMAC-MD5                            |
    |                          (128 bits)                           |
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: Authentication information format

DHCP realm — The DHCP realm that identifies the key used to generate the HMAC-MD5 value. This is a domain name encoded as described in Section 9.

key ID — The key identifier that identified the key used to generate the HMAC-MD5 value.

HMAC-MD5 — The message authentication code generated by applying MD5 to the DHCP message using the key identified by the DHCP realm, client DUID, and key ID.

The sender computes the MAC using the HMAC generation algorithm [RFC2104] and the MD5 hash function [RFC1321]. The entire DHCP message (setting the MAC field of the authentication option to zero), including the DHCP message header and the options field, is used as input to the HMAC-MD5 computation function.

DISCUSSION:

Algorithm 1 specifies the use of HMAC-MD5. Use of a different technique, such as HMAC-SHA, will be specified as a separate protocol.

The DHCP realm used to identify authentication keys is chosen to be unique among administrative domains. Use of the DHCP realm allows DHCP administrators to avoid conflict in the use of key identifiers, and allows a host using DHCP to use authenticated DHCP while roaming among DHCP administrative domains.

22.4.2. Message Validation

Any DHCP message that includes more than one authentication option MUST be discarded.

To validate an incoming message, the receiver first checks that the value in the replay detection field is acceptable according to the replay detection method specified by the RDM field. If no replay is detected, then the receiver computes the MAC as described in [RFC2104]. The entire DHCP message (setting the MAC field of the authentication option to 0) is used as input to the HMAC-MD5 computation function. If the MAC computed by the receiver does not match the MAC contained in the authentication option, the receiver MUST discard the DHCP message.

22.4.3. Key Utilization

Each DHCP client has a set of keys. Each key is identified by <DHCP realm, client DUID, key id>. Each key also has a lifetime. The key may not be used past the end of its lifetime. The client’s keys are initially distributed to the client through some out-of-band mechanism. The lifetime for each key is distributed with the key. Mechanisms for key distribution and lifetime specification are beyond the scope of this document.

The client and server use one of the client’s keys to authenticate DHCP messages during a session (until the next Solicit message sent by the client).

22.4.4. Client Considerations for Delayed Authentication Protocol

The client announces its intention to use DHCP authentication by including an Authentication option in its Solicit message. The server selects a key for the client based on the client’s DUID. The client and server use that key to authenticate all DHCP messages exchanged during the session.

22.4.4.1. Sending Solicit Messages

When the client sends a Solicit message and wishes to use authentication, it includes an Authentication option with the desired protocol, algorithm and RDM as described in Section 22.4. The client does not include any replay detection or authentication information in the Authentication option.

22.4.4.2. Receiving Advertise Messages

The client validates any Advertise messages containing an Authentication option specifying the delayed authentication protocol using the validation test described in Section 22.4.2.

The Client behavior is defined by local policy, as detailed below.

If the client requires that Advertise messages be authenticated, then it MUST ignore Advertise messages that do not include authentication information, or for which the client has no matching key, or that do not pass the validation test.

Local policy MAY also prefer authenticated Advertise messages, in which case the client SHOULD attempt to validate all Advertise messages for which the client has a matching key. Messages for which the client has a key, but which do not pass the validation test MUST be rejected, even if the client would otherwise accept the same message without the Authentication option.

In all cases, messages for which the client does not have a matching key should be treated as if they have no Authentication option.

When the decision to accept unauthenticated message is made, it should be made with care. Accepting an unauthenticated Advertise message can make the client vulnerable to spoofing and other attacks. Policies and actions which were depending upon Authentication MUST NOT be executed. Local users SHOULD be informed that the client has accepted an unauthenticated Advertise message.

A client MUST be configurable to discard unauthenticated messages, and SHOULD be configured by default to discard unauthenticated messages if the client has been configured with an authentication key or other authentication information.

A client MAY choose to differentiate between Advertise messages with no authentication information and Advertise messages that do not pass the validation test; for example, a client might accept the former and discard the latter. If a client does accept an unauthenticated message, the client SHOULD inform any local users and SHOULD log the event.

22.4.4.3. Sending Request, Confirm, Renew, Rebind, Decline or Release Messages

If the client authenticated the Advertise message through which the client selected the server, the client MUST generate authentication information for subsequent Request, Confirm, Renew, Rebind or Release messages sent to the server, as described in Section 22.4. When the client sends a subsequent message, it MUST use the same key used by the server to generate the authentication information.

22.4.4.4. Sending Information-request Messages

If the server has selected a key for the client in a previous message exchange (see Section 22.4.5.1), the client MUST use the same key to generate the authentication information throughout the session.

22.4.4.5. Receiving Reply Messages

If the client authenticated the Advertise it accepted, the client MUST validate the associated Reply message from the server. The client MUST ignore and discard the Reply if the message fails to pass the validation test and MAY log the validation failure.

If the client accepted an Advertise message that did not include authentication information or did not pass the validation test, the client MAY accept an unauthenticated Reply message from the server.

22.4.4.6. Receiving Reconfigure Messages

The client MUST discard the Reconfigure if the message fails to pass the validation test and MAY log the validation failure.

22.4.5. Server Considerations for Delayed Authentication Protocol

After receiving a Solicit message that contains an Authentication option, the server selects a key for the client, based on the client’s DUID and key selection policies with which the server has been configured. The server identifies the selected key in the Advertise message and uses the key to validate subsequent messages between the client and the server.

22.4.5.1. Receiving Solicit Messages and Sending Advertise Messages

The server selects a key for the client and includes authentication information in the Advertise message returned to the client as specified in Section 22.4. The server MUST record the identifier of the key selected for the client and use that same key for validating subsequent messages with the client.

22.4.5.2. Receiving Request, Confirm, Renew, Rebind or Release Messages and Sending Reply Messages

The server uses the key identified in the message and validates the message as specified in Section 22.4.2. If the message fails to pass the validation test or the server does not know the key identified by the ’key ID’ field, the server MUST discard the message and MAY choose to log the validation failure. If the server receives a client message without an authentication option while the server has previously sent authentication information in the same session, it MUST discard the message and MAY choose to log the validation failure, because the client violates the definition in Section 22.4.4.3.

If the message passes the validation test, the server responds to the specific message as described in Section 19.2. The server MUST include authentication information generated using the key identified in the received message, as specified in Section 22.4.

22.5. Reconfigure Key Authentication Protocol

The Reconfigure key authentication protocol provides protection against misconfiguration of a client caused by a Reconfigure message sent by a malicious DHCP server. In this protocol, a DHCP server sends a Reconfigure Key to the client in the initial exchange of DHCP messages. The client records the Reconfigure Key for use in authenticating subsequent Reconfigure messages from that server. The server then includes an HMAC computed from the Reconfigure Key in subsequent Reconfigure messages.

Both the Reconfigure Key sent from the server to the client and the HMAC in subsequent Reconfigure messages are carried as the Authentication information in an Authentication option. The format of the Authentication information is defined in the following section.

The Reconfigure Key protocol is used (initiated by the server) only if the client and server are not using any other authentication protocol and the client and server have negotiated to use Reconfigure messages.

22.5.1. Use of the Authentication Option in the Reconfigure Key Authentication Protocol

The following fields are set in an Authentication option for the Reconfigure Key Authentication Protocol:

protocol 3

algorithm 1

RDM 0

The format of the Authentication information for the Reconfigure Key Authentication Protocol is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |                 Value (128 bits)              |
+-+-+-+-+-+-+-+-+                                               |
.                                                               .
.                                                               .
.                                               +-+-+-+-+-+-+-+-+
|                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10: RKAP Authentication Information

Type Type of data in Value field carried in this option:

1 Reconfigure Key value (used in Reply message).

2 HMAC-MD5 digest of the message (used in Reconfigure message).

Value Data as defined by the Type field.

22.5.2. Server considerations for Reconfigure Key protocol

The server selects a Reconfigure Key for a client during the Request/Reply, Solicit/Reply or Information-request/Reply message exchange. The server records the Reconfigure Key and transmits that key to the client in an Authentication option in the Reply message.

The Reconfigure Key is 128 bits long, and MUST be a cryptographically strong random or pseudo-random number that cannot easily be predicted.

To provide authentication for a Reconfigure message, the server selects a replay detection value according to the RDM selected by the server, and computes an HMAC-MD5 of the Reconfigure message using the Reconfigure Key for the client. The server computes the HMAC-MD5 over the entire DHCP Reconfigure message, including the

Authentication option; the HMAC-MD5 field in the Authentication option is set to zero for the HMAC-MD5 computation. The server includes the HMAC-MD5 in the authentication information field in an Authentication option included in the Reconfigure message sent to the client.

22.5.3. Client considerations for Reconfigure Key protocol

The client will receive a Reconfigure Key from the server in the initial Reply message from the server. The client records the Reconfigure Key for use in authenticating subsequent Reconfigure messages.

To authenticate a Reconfigure message, the client computes an HMAC-MD5 over the DHCP Reconfigure message, using the Reconfigure Key received from the server. If this computed HMAC-MD5 matches the value in the Authentication option, the client accepts the Reconfigure message.

23. DHCP Options

Options are used to carry additional information and parameters in DHCP messages. Every option shares a common base format, as described in Section 23.1. All values in options are represented in network byte order.

This document describes the DHCP options defined as part of the base DHCP specification. Other options may be defined in the future in separate documents. See [RFC7227] for guidelines regarding new options definition.

Unless otherwise noted, each option may appear only in the options area of a DHCP message and may appear only once. If an option does appear multiple times, each instance is considered separate and the data areas of the options MUST NOT be concatenated or otherwise combined.

Options that are allowed to appear only once are called singleton options. The only non-singleton options defined in this document are IA_NA (see Section 23.4), IA_TA (see Section 23.5), and IA_PD (see Section 23.21) options. Also, IAAddress (see Section 23.6) and IAPrefix (see Section 23.22) may appear in their respective IA options more than once.

23.1. Format of DHCP Options

The format of DHCP options is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          option-code          |           option-len          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          option-data                          |
|                      (option-len octets)                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: Option Format

option-code An unsigned integer identifying the specific option type carried in this option.

option-len An unsigned integer giving the length of the option-data field in this option in octets.

option-data The data for the option; the format of this data depends on the definition of the option.

DHCPv6 options are scoped by using encapsulation. Some options apply generally to the client, some are specific to an IA, and some are specific to the addresses within an IA. These latter two cases are discussed in Section 23.4 and Section 23.6.

23.2. Client Identifier Option

The Client Identifier option is used to carry a DUID (see Section 10) identifying a client between a client and a server. The format of the Client Identifier option is:

```
      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     |        OPTION_CLIENTID        |          option-len           |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
     .                                                               .
     .                              DUID                             .
     .                        (variable length)                      .
     .                                                               .
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 12: Client Identifier Option Format

option-code   OPTION_CLIENTID (1).

option-len   Length of DUID in octets.

DUID   The DUID for the client.

23.3. Server Identifier Option

The Server Identifier option is used to carry a DUID (see Section 10) identifying a server between a client and a server. The format of the Server Identifier option is:

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |        OPTION_SERVERID        |          option-len           |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      .                                                               .
      .                              DUID                             .
      .                        (variable length)                      .
      .                                                               .
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 13: Server Identifier Option Format

option-code   OPTION_SERVERID (2).

option-len   Length of DUID in octets.

DUID   The DUID for the server.

23.4. Identity Association for Non-temporary Addresses Option

The Identity Association for Non-temporary Addresses option (IA_NA option) is used to carry an IA_NA, the parameters associated with the IA_NA, and the non-temporary addresses associated with the IA_NA.

Addresses appearing in an IA_NA option are not temporary addresses (see Section 23.5).

The format of the IA_NA option is:

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |          OPTION_IA_NA         |          option-len           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                        IAID (4 octets)                        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                              T1                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                              T2                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    .                         IA_NA-options                         .
    .                                                               .
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 14: Identity Association for Non-temporary Addresses Option Format

option-code   OPTION_IA_NA (3).

option-len   12 + length of IA_NA-options field.

IAID   The unique identifier for this IA_NA; the IAID must be unique among the identifiers for all of this client’s IA_NAs. The number space for IA_NA IAIDs is separate from the number space for IA_TA IAIDs.

T1   The time at which the client contacts the server from which the addresses in the IA_NA were obtained to extend the lifetimes of the addresses assigned to the IA_NA; T1 is a time duration relative to the current time expressed in units of seconds.

T2 The time at which the client contacts any available server to extend the lifetimes of the addresses assigned to the IA_NA; T2 is a time duration relative to the current time expressed in units of seconds.

IA_NA-options Options associated with this IA_NA.

The IA_NA-options field encapsulates those options that are specific to this IA_NA. For example, all of the IA Address Options carrying the addresses associated with this IA_NA are in the IA_NA-options field.

Each IA_NA carries one "set" of non-temporary addresses; that is, at most one address from each prefix assigned to the link to which the client is attached.

An IA_NA option may only appear in the options area of a DHCP message. A DHCP message may contain multiple IA_NA options.

The status of any operations involving this IA_NA is indicated in a Status Code option in the IA_NA-options field.

Note that an IA_NA has no explicit "lifetime" or "lease length" of its own. When the valid lifetimes of all of the addresses in an IA_NA have expired, the IA_NA can be considered as having expired. T1 and T2 are included to give servers explicit control over when a client recontacts the server about a specific IA_NA.

In a message sent by a client to a server, values in the T1 and T2 fields indicate the client’s preference for those parameters. The client sets T1 and T2 to 0 if it has no preference for those values. In a message sent by a server to a client, the client MUST use the values in the T1 and T2 fields for the T1 and T2 parameters, unless those values in those fields are 0. The values in the T1 and T2 fields are the number of seconds until T1 and T2.

The server selects the T1 and T2 times to allow the client to extend the lifetimes of any addresses in the IA_NA before the lifetimes expire, even if the server is unavailable for some short period of time. Recommended values for T1 and T2 are .5 and .8 times the shortest preferred lifetime of the addresses in the IA that the server is willing to extend, respectively. If the "shortest" preferred lifetime is 0xffffffff ("infinity"), the recommended T1 and T2 values are also 0xffffffff. If the time at which the addresses in an IA_NA are to be renewed is to be left to the discretion of the client, the server sets T1 and T2 to 0.

If a server receives an IA_NA with T1 greater than T2, and both T1 and T2 are greater than 0, the server ignores the invalid values of T1 and T2 and processes the IA_NA as though the client had set T1 and T2 to 0.

If a client receives an IA_NA with T1 greater than T2, and both T1 and T2 are greater than 0, the client discards the IA_NA option and processes the remainder of the message as though the server had not included the invalid IA_NA option.

Care should be taken in setting T1 or T2 to 0xffffffff ("infinity"). A client will never attempt to extend the lifetimes of any addresses in an IA with T1 set to 0xffffffff. A client will never attempt to use a Rebind message to locate a different server to extend the lifetimes of any addresses in an IA with T2 set to 0xffffffff.

This option MAY appear in a Confirm message if the lifetimes on the non-temporary addresses in the associated IA have not expired.

23.5. Identity Association for Temporary Addresses Option

The Identity Association for the Temporary Addresses (IA_TA) option is used to carry an IA_TA, the parameters associated with the IA_TA and the addresses associated with the IA_TA. All of the addresses in this option are used by the client as temporary addresses, as defined in [RFC4941]. The format of the IA_TA option is:

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |          OPTION_IA_TA         |          option-len           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                         IAID (4 octets)                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    .                         IA_TA-options                         .
    .                                                               .
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 15: Identity Association for Temporary Addresses Option Format

option-code	OPTION_IA_TA (4).
option-len	4 + length of IA_TA-options field.
IAID	The unique identifier for this IA_TA; the IAID must be unique among the identifiers for

```
                              all of this client's IA_TAs.  The number
                              space for IA_TA IAIDs is separate from the
                              number space for IA_NA IAIDs.

      IA_TA-options           Options associated with this IA_TA.
```

The IA_TA-Options field encapsulates those options that are specific to this IA_TA. For example, all of the IA Address Options carrying the addresses associated with this IA_TA are in the IA_TA-options field.

Each IA_TA carries one "set" of temporary addresses.

An IA_TA option may only appear in the options area of a DHCP message. A DHCP message may contain multiple IA_TA options.

The status of any operations involving this IA_TA is indicated in a Status Code option in the IA_TA-options field.

Note that an IA has no explicit "lifetime" or "lease length" of its own. When the valid lifetimes of all of the addresses in an IA_TA have expired, the IA can be considered as having expired.

An IA_TA option does not include values for T1 and T2. A client MAY request that the lifetimes on temporary addresses be extended by including the addresses in a IA_TA option sent in a Renew or Rebind message to a server. For example, a client would request an extension on the lifetime of a temporary address to allow an application to continue to use an established TCP connection.

The client obtains new temporary addresses by sending an IA_TA option with a new IAID to a server. Requesting new temporary addresses from the server is the equivalent of generating new temporary addresses as described in [RFC4941]. The server will generate new temporary addresses and return them to the client. The client should request new temporary addresses before the lifetimes on the previously assigned addresses expire.

A server MUST return the same set of temporary address for the same IA_TA (as identified by the IAID) as long as those addresses are still valid. After the lifetimes of the addresses in an IA_TA have expired, the IAID may be reused to identify a new IA_TA with new temporary addresses.

This option MAY appear in a Confirm message if the lifetimes on the temporary addresses in the associated IA have not expired.

23.6. IA Address Option

The IA Address option is used to specify IPv6 addresses associated with an IA_NA or an IA_TA. The IA Address option must be encapsulated in the Options field of an IA_NA or IA_TA option. The Options fields of the IA_NA or IA_TA option encapsulates those options that are specific to this address.

The format of the IA Address option is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          OPTION_IAADDR        |          option-len           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                         IPv6 address                          |
|                                                               |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      preferred-lifetime                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        valid-lifetime                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                        IAaddr-options                         .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 16: IA Address Option Format

option-code OPTION_IAADDR (5).

option-len 24 + length of IAaddr-options field.

IPv6 address An IPv6 address.

preferred-lifetime The preferred lifetime for the IPv6 address in the option, expressed in units of seconds.

valid-lifetime The valid lifetime for the IPv6 address in the option, expressed in units of seconds.

IAaddr-options Options associated with this address.

In a message sent by a client to a server, values in the preferred and valid lifetime fields indicate the client’s preference for those parameters. The client may send 0 if it has no preference for the preferred and valid lifetimes. If a client wishes to express its lifetimes preferences and does not have the knowledge to populate the IPv6 address field, it can use unspecified address (::). It is up to a server to honor or ignore these preferences.

In a message sent by a server to a client, the client MUST use the values in the preferred and valid lifetime fields for the preferred and valid lifetimes. The values in the preferred and valid lifetimes are the number of seconds remaining in each lifetime.

A client discards any addresses for which the preferred lifetime is greater than the valid lifetime. A server ignores the lifetimes set by the client if the preferred lifetime is greater than the valid lifetime and ignores the values for T1 and T2 set by the client if those values are greater than the preferred lifetime.

Care should be taken in setting the valid lifetime of an address to 0xffffffff ("infinity"), which amounts to a permanent assignment of an address to a client.

More than one IA Address Option can appear in an IA_NA option or an IA_TA option.

The status of any operations involving this IA Address is indicated in a Status Code option in the IAaddr-options field, as specified in Section 23.13.

23.7. Option Request Option

The Option Request option is used to identify a list of options in a message between a client and a server. The format of the Option Request option is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           OPTION_ORO          |           option-len          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    requested-option-code-1    |    requested-option-code-2    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 17: Option Request Option Format

```
option-code          OPTION_ORO (6).
```

```
      option-len           2 * number of requested options.

      requested-option-code-n  The option code for an option requested
                           by the client.
```

A client MAY include an Option Request option in a Solicit, Request, Renew, Rebind, Confirm or Information-request message to inform the server about options the client wants the server to send to the client. A server MAY include an Option Request option in a Reconfigure message to indicate which options the client should request from the server. If there is a need to request encapsulated options, top-level Option Request option MUST be used for that purpose. There is no need request IAADDR or IAPREFIX.

23.8. Preference Option

The Preference option is sent by a server to a client to affect the selection of a server by the client.

The format of the Preference option is:

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |       OPTION_PREFERENCE       |          option-len           |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |  pref-value   |
      +-+-+-+-+-+-+-+-+
```

Figure 18: Preference Option Format

```
      option-code          OPTION_PREFERENCE (7).

      option-len           1.

      pref-value           The preference value for the server in this
                           message.
```

A server MAY include a Preference option in an Advertise message to control the selection of a server by the client. See Section 18.1.3 for the use of the Preference option by the client and the interpretation of Preference option data value.

23.9. Elapsed Time Option

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |      OPTION_ELAPSED_TIME      |           option-len          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |          elapsed-time         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 19: Elapsed Time Option Format

option-code OPTION_ELAPSED_TIME (8).

option-len 2.

elapsed-time The amount of time since the client began its current DHCP transaction. This time is expressed in hundredths of a second (10^-2 seconds).

A client MUST include an Elapsed Time option in messages to indicate how long the client has been trying to complete a DHCP message exchange. The elapsed time is measured from the time at which the client sent the first message in the message exchange, and the elapsed-time field is set to 0 in the first message in the message exchange. Servers and Relay Agents use the data value in this option as input to policy controlling how a server responds to a client message. For example, the elapsed time option allows a secondary DHCP server to respond to a request when a primary server has not answered in a reasonable time. The elapsed time value is an unsigned, 16 bit integer. The client uses the value 0xffff to represent any elapsed time values greater than the largest time value that can be represented in the Elapsed Time option.

23.10. Relay Message Option

The Relay Message option carries a DHCP message in a Relay-forward or Relay-reply message.

The format of the Relay Message option is:

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |        OPTION_RELAY_MSG       |           option-len          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    .                       DHCP-relay-message                      .
    .                                                               .
    .                                                               .
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 20: Relay Message Option Format

option-code   OPTION_RELAY_MSG (9)

option-len   Length of DHCP-relay-message

DHCP-relay-message   In a Relay-forward message, the received message, relayed verbatim to the next relay agent or server; in a Relay-reply message, the message to be copied and relayed to the relay agent or client whose address is in the peer-address field of the Relay-reply message

23.11. Authentication Option

The Authentication option carries authentication information to authenticate the identity and contents of DHCP messages. The use of the Authentication option is described in Section 22. The format of the Authentication option is:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          OPTION_AUTH          |          option-len           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   protocol    |   algorithm   |      RDM      |               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+               |
   |                                                               |
   |          replay detection (64 bits)           +-+-+-+-+-+-+-+-+
   |                                               |   auth-info   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+               |
   .                   authentication information                  .
   .                       (variable length)                       .
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 21: Authentication Option Format

option-code OPTION_AUTH (11).

option-len 11 + length of authentication information field.

protocol The authentication protocol used in this authentication option.

algorithm The algorithm used in the authentication protocol.

RDM The replay detection method used in this authentication option.

Replay detection The replay detection information for the RDM.

authentication information The authentication information, as specified by the protocol and algorithm used in this authentication option.

23.12. Server Unicast Option

The server sends this option to a client to indicate to the client that it is allowed to unicast messages to the server. The format of the Server Unicast option is:

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |          OPTION_UNICAST       |        option-len             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    |                       server-address                          |
    |                                                               |
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 22: Server Unicast Option Format

option-code OPTION_UNICAST (12).

option-len 16.

server-address The IP address to which the client should send messages delivered using unicast.

The server specifies the IPv6 address to which the client is to send unicast messages in the server-address field. When a client receives this option, where permissible and appropriate, the client sends messages directly to the server using the IPv6 address specified in the server-address field of the option.

When the server sends a Unicast option to the client, some messages from the client will not be relayed by Relay Agents, and will not include Relay Agent options from the Relay Agents. Therefore, a server should only send a Unicast option to a client when Relay Agents are not sending Relay Agent options. A DHCP server rejects any messages sent inappropriately using unicast to ensure that messages are relayed by Relay Agents when Relay Agent options are in use.

Details about when the client may send messages to the server using unicast are in Section 19.

23.13. Status Code Option

This option returns a status indication related to the DHCP message or option in which it appears. The format of the Status Code option is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       OPTION_STATUS_CODE      |         option-len            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          status-code          |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
.                                                               .
.                        status-message                         .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 23: Status Code Option Format

option-code OPTION_STATUS_CODE (13).

option-len 2 + length of status-message.

status-code The numeric code for the status encoded in this option.

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

A Status Code option may appear in the options field of a DHCP message and/or in the options field of another option. If the Status Code option does not appear in a message in which the option could appear, the status of the message is assumed to be Success.

The status-codes values previously defined by [RFC3315] and [RFC3633] are:

Name	Code	Description
Success	0	Success.
UnspecFail	1	Failure, reason unspecified; this status code is sent by either a client or a server to indicate a failure not explicitly specified in this document.
NoAddrsAvail	2	Server has no addresses available to assign to the IA(s).
NoBinding	3	Client record (binding) unavailable.
NotOnLink	4	The prefix for the address is not appropriate for the link to which the client is attached.
UseMulticast	5	Sent by a server to a client to force the client to send messages to the server using the All_DHCP_Relay_Agents_and_Servers address.
NoPrefixAvail	6	Delegating router has no prefixes available to assign to the IAPD(s).

23.14. Rapid Commit Option

The Rapid Commit option is used to signal the use of the two message exchange for address assignment. The format of the Rapid Commit option is:

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |      OPTION_RAPID_COMMIT      |               0               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 24: Rapid Commit Option Format

```
   option-code          OPTION_RAPID_COMMIT (14).

   option-len           0.
```

A client MAY include this option in a Solicit message if the client is prepared to perform the Solicit-Reply message exchange described in Section 18.1.1.

A server MUST include this option in a Reply message sent in response to a Solicit message when completing the Solicit-Reply message exchange.

DISCUSSION:

Each server that responds with a Reply to a Solicit that includes a Rapid Commit option will commit the assigned addresses in the Reply message to the client, and will not receive any confirmation that the client has received the Reply message. Therefore, if more than one server responds to a Solicit that includes a Rapid Commit option, some servers will commit addresses that are not actually used by the client.

The problem of unused addresses can be minimized, for example, by designing the DHCP service so that only one server responds to the Solicit or by using relatively short lifetimes for assigned addresses, or the DHCP client initiatively releases unused addresses using the Release message.

23.15. User Class Option

The User Class option is used by a client to identify the type or category of user or applications it represents.

The format of the User Class option is:

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |       OPTION_USER_CLASS       |          option-len           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    .                                                               .
    .                          user-class-data                      .
    .                                                               .
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 25: User Class Option Format

option-code   OPTION_USER_CLASS (15).

option-len   Length of user class data field.

user-class-data   The user classes carried by the client.

The information contained in the data area of this option is contained in one or more opaque fields that represent the user class or classes of which the client is a member. A server selects configuration information for the client based on the classes identified in this option. For example, the User Class option can be used to configure all clients of people in the accounting department with a different printer than clients of people in the marketing department. The user class information carried in this option MUST be configurable on the client.

The data area of the user class option MUST contain one or more instances of user class data. Each instance of the user class data is formatted as follows:

```
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-...-+-+-+-+-+-+-+
      |        user-class-len         |          opaque-data          |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-...-+-+-+-+-+-+-+
```

Figure 26: User Class Data Format

The user-class-len is two octets long and specifies the length of the opaque user class data in network byte order.

A server interprets the classes identified in this option according to its configuration to select the appropriate configuration information for the client. A server may use only those user classes that it is configured to interpret in selecting configuration information for a client and ignore any other user classes. In response to a message containing a User Class option, a server includes a User Class option containing those classes that were successfully interpreted by the server, so that the client can be informed of the classes interpreted by the server.

23.16. Vendor Class Option

This option is used by a client to identify the vendor that manufactured the hardware on which the client is running. The information contained in the data area of this option is contained in one or more opaque fields that identify details of the hardware configuration. The format of the Vendor Class option is:

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |      OPTION_VENDOR_CLASS      |           option-len          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                       enterprise-number                       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    .                                                               .
    .                       vendor-class-data                       .
    .                             . . .                             .
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 27: Vendor Class Option Format

option-code OPTION_VENDOR_CLASS (16).

option-len 4 + length of vendor class data field.

enterprise-number The vendor’s registered Enterprise Number as registered with IANA [IANA-PEN].

vendor-class-data The hardware configuration of the host on which the client is running.

The vendor-class-data is composed of a series of separate items, each of which describes some characteristic of the client’s hardware configuration. Examples of vendor-class-data instances might include the version of the operating system the client is running or the amount of memory installed on the client.

Each instance of the vendor-class-data is formatted as follows:

```
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+...-+-+-+-+-+-+-+
    |       vendor-class-len        |          opaque-data          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+...-+-+-+-+-+-+-+
```

Figure 28: Vendor Class Data Format

The vendor-class-len is two octets long and specifies the length of the opaque vendor class data in network byte order.

Servers and clients MUST NOT include more than one instance of OPTION_VENDOR_CLASS with the same Enterprise Number. Each instance of OPTION_VENDOR_CLASS can carry multiple sub-options.

23.17. Vendor-specific Information Option

This option is used by clients and servers to exchange vendor-specific information.

The format of the Vendor-specific Information option is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      OPTION_VENDOR_OPTS       |           option-len          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       enterprise-number                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                          option-data                          .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 29: Vendor-specific Information Option Format

option-code   OPTION_VENDOR_OPTS (17).

option-len   4 + length of option-data field.

enterprise-number   The vendor’s registered Enterprise Number as registered with IANA [IANA-PEN].

option-data   An opaque object, interpreted by vendor-specific code on the clients and servers.

The definition of the information carried in this option is vendor specific. The vendor is indicated in the enterprise-number field. Use of vendor-specific information allows enhanced operation, utilizing additional features in a vendor’s DHCP implementation. A DHCP client that does not receive requested vendor-specific information will still configure the host device’s IPv6 stack to be functional.

The encapsulated vendor-specific options field MUST be encoded as a sequence of code/length/value fields of identical format to the DHCP options field. The option codes are defined by the vendor identified in the enterprise-number field and are not managed by IANA. Each of the encapsulated options is formatted as follows:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          opt-code             |             option-len        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   .                                                               .
   .                          option-data                          .
   .                                                               .
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 30: Vendor-specific Options Format

opt-code The code for the encapsulated option.

option-len An unsigned integer giving the length of the option-data field in this encapsulated option in octets.

option-data The data area for the encapsulated option.

Multiple instances of the Vendor-specific Information option may appear in a DHCP message. Each instance of the option is interpreted according to the option codes defined by the vendor identified by the Enterprise Number in that option. Servers and clients MUST NOT send more than one instance of Vendor-specific Information option with the same Enterprise Number. Each instanf of Vendor-specific Information option MAY contain multiple encapsulated options.

A client that is interested in receiving a Vendor-specific Information Option:

- MUST specify the Vendor-specific Information Option in an Option Request Option.

- MAY specify an associated Vendor Class Option.

- MAY specify the Vendor-specific Information Option with any data.

Severs only return the Vendor-specific Information Options if specified in Option Request Options from clients and:

- MAY use the Enterprise Numbers in the associated Vendor Class Options to restrict the set of Enterprise Numbers in the Vendor-specific Information Options returned.

- MAY return all configured Vendor-specific Information Options.

- MAY use other information in the packet or in its configuration to determine which set of Enterprise Numbers in the Vendor-specific Information Options to return.

23.18. Interface-Id Option

The relay agent MAY send the Interface-id option to identify the interface on which the client message was received. If a relay agent receives a Relay-reply message with an Interface-id option, the relay agent relays the message to the client through the interface identified by the option.

The format of the Interface ID option is:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      OPTION_INTERFACE_ID      |         option-len            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   .                                                               .
   .                         interface-id                          .
   .                                                               .
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 31: Interface-ID Option Format

option-code   OPTION_INTERFACE_ID (18).

option-len   Length of interface-id field.

interface-id   An opaque value of arbitrary length generated by the relay agent to identify one of the relay agent’s interfaces.

The server MUST copy the Interface-Id option from the Relay-forward message into the Relay-reply message the server sends to the relay agent in response to the Relay-forward message. This option MUST NOT appear in any message except a Relay-forward or Relay-reply message.

Servers MAY use the Interface-ID for parameter assignment policies. The Interface-ID SHOULD be considered an opaque value, with policies based on exact match only; that is, the Interface-ID SHOULD NOT be internally parsed by the server. The Interface-ID value for an interface SHOULD be stable and remain unchanged, for example, after the relay agent is restarted; if the Interface-ID changes, a server will not be able to use it reliably in parameter assignment policies.

23.19. Reconfigure Message Option

A server includes a Reconfigure Message option in a Reconfigure message to indicate to the client whether the client responds with a Renew message, a Rebind message, or an Information-request message. The format of this option is:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      OPTION_RECONF_MSG        |         option-len            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |    msg-type   |
   +-+-+-+-+-+-+-+-+
```

Figure 32: Reconfigure Message Option Format

option-code   OPTION_RECONF_MSG (19).

option-len   1.

msg-type   5 for Renew message, 6 for Rebind, 11 for Information-request message.

The Reconfigure Message option can only appear in a Reconfigure message.

23.20. Reconfigure Accept Option

A client uses the Reconfigure Accept option to announce to the server whether the client is willing to accept Reconfigure messages, and a server uses this option to tell the client whether or not to accept Reconfigure messages. The default behavior, in the absence of this option, means unwillingness to accept Reconfigure messages, or instruction not to accept Reconfigure messages, for the client and server messages, respectively. The following figure gives the format of the Reconfigure Accept option:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     OPTION_RECONF_ACCEPT      |               0               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 33: Reconfigure Accept Option Format

option-code OPTION_RECONF_ACCEPT (20).

option-len 0.

23.21. Identity Association for Prefix Delegation Option

The IA_PD option is used to carry a prefix delegation identity association, the parameters associated with the IA_PD and the prefixes associated with it.

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |         OPTION_IA_PD          |         option-length         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                         IAID (4 octets)                       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                              T1                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                              T2                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 .                                                               .
 .                          IA_PD-options                        .
 .                                                               .
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 34: Identity Association for Prefix Delegation Option Format

option-code OPTION_IA_PD (25).

option-length 12 + length of IA_PD-options field.

IAID The unique identifier for this IA_PD; the IAID must be unique among the identifiers for all of this requesting router’s IA_PDs.

T1 The time at which the requesting router should contact the delegating router from which the prefixes in the IA_PD were obtained to extend the lifetimes of the prefixes delegated to the IA_PD; T1 is a time duration relative to the current time expressed in units of seconds.

T2 The time at which the requesting router should contact any available delegating router to extend the lifetimes of the

```
                              prefixes assigned to the IA_PD; T2 is a time
                              duration relative to the current time
                              expressed in units of seconds.

      IA_PD-options           Options associated with this IA_PD.
```

The IA_PD-options field encapsulates those options that are specific to this IA_PD. For example, all of the IA_PD Prefix Options carrying the prefixes associated with this IA_PD are in the IA_PD-options field.

An IA_PD option may only appear in the options area of a DHCP message. A DHCP message may contain multiple IA_PD options.

The status of any operations involving this IA_PD is indicated in a Status Code option in the IA_PD-options field.

Note that an IA_PD has no explicit "lifetime" or "lease length" of its own. When the valid lifetimes of all of the prefixes in a IA_PD have expired, the IA_PD can be considered as having expired. T1 and T2 are included to give delegating routers explicit control over when a requesting router should contact the delegating router about a specific IA_PD.

In a message sent by a requesting router to a delegating router, values in the T1 and T2 fields indicate the requesting router’s preference for those parameters. The requesting router sets T1 and T2 to zero if it has no preference for those values. In a message sent by a delegating router to a requesting router, the requesting router MUST use the values in the T1 and T2 fields for the T1 and T2 parameters. The values in the T1 and T2 fields are the number of seconds until T1 and T2.

The delegating router selects the T1 and T2 times to allow the requesting router to extend the lifetimes of any prefixes in the IA_PD before the lifetimes expire, even if the delegating router is unavailable for some short period of time. Recommended values for T1 and T2 are .5 and .8 times the shortest preferred lifetime of the prefixes in the IA_PD that the delegating router is willing to extend, respectively. If the time at which the prefixes in an IA_PD are to be renewed is to be left to the discretion of the requesting router, the delegating router sets T1 and T2 to 0.

If a delegating router receives an IA_PD with T1 greater than T2, and both T1 and T2 are greater than 0, the delegating router ignores the invalid values of T1 and T2 and processes the IA_PD as though the requesting router had set T1 and T2 to 0.

If a requesting router receives an IA_PD with T1 greater than T2, and both T1 and T2 are greater than 0, the requesting router discards the IA_PD option and processes the remainder of the message as though the requesting router had not included the IA_PD option.

23.22. IA Prefix Option

The IA_PD Prefix option is used to specify IPv6 address prefixes associated with an IA_PD. The IA_PD Prefix option must be encapsulated in the IA_PD-options field of an IA_PD option.

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |        OPTION_IAPREFIX        |         option-length         |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                      preferred-lifetime                       |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                        valid-lifetime                         |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      | prefix-length |                                               |
      +-+-+-+-+-+-+-+-+          IPv6 prefix                          |
      |                           (16 octets)                         |
      |                                                               |
      |                                                               |
      |                                                               |
      |               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |               |                                               .
      +-+-+-+-+-+-+-+-+                                               .
      .                       IAprefix-options                        .
      .                                                               .
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 35: IA Prefix Option Format

option-code   OPTION_IAPREFIX (26).

option-length   25 + length of IAprefix-options field.

preferred-lifetime   The recommended preferred lifetime for the IPv6 prefix in the option, expressed in units of seconds. A value of 0xFFFFFFFF represents infinity.

valid-lifetime   The valid lifetime for the IPv6 prefix in the option, expressed in units of seconds. A value of 0xFFFFFFFF represents infinity.

prefix-length	Length for this prefix in bits.
IPv6-prefix	An IPv6 prefix.
IAprefix-options	Options associated with this prefix.

In a message sent by a requesting router to a delegating router, the values in the fields can be used to indicate the requesting router’s preference for those values. The requesting router may send a value of zero to indicate no preference. A requesting router may set the IPv6 prefix field to zero and a given value in the prefix-length field to indicate a preference for the size of the prefix to be delegated.

In a message sent by a delegating router the preferred and valid lifetimes should be set to the values of AdvPreferredLifetime and AdvValidLifetime as specified in section 6.2.1, "Router Configuration Variables" of [RFC2461], unless administratively configured.

A requesting router discards any prefixes for which the preferred lifetime is greater than the valid lifetime. A delegating router ignores the lifetimes set by the requesting router if the preferred lifetime is greater than the valid lifetime and ignores the values for T1 and T2 set by the requesting router if those values are greater than the preferred lifetime.

The values in the preferred and valid lifetimes are the number of seconds remaining for each lifetime.

An IA_PD Prefix option may appear only in an IA_PD option. More than one IA_PD Prefix Option can appear in a single IA_PD option.

The status of any operations involving this IA_PD Prefix option is indicated in a Status Code option in the IAprefix-options field.

23.23. SOL_MAX_RT Option

A DHCP server sends the SOL_MAX_RT option to a client to override the default value of SOL_MAX_RT. The value of SOL_MAX_RT in the option replaces the default value defined in Section 6.5. One use for the SOL_MAX_RT option is to set a longer value for SOL_MAX_RT, which reduces the Solicit traffic from a client that has not received a response to its Solicit messages.

The format of the SOL_MAX_RT option is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          option-code          |         option-len            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       SOL_MAX_RT value                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 36: SOL_MAX_RT Option Format

option-code   OPTION_SOL_MAX_RT (82).

option-len   4.

SOL_MAX_RT value   Overriding value for SOL_MAX_RT in seconds; MUST be in range: 60 <= "value" <= 86400 (1 day).

A DHCP client MUST include the SOL_MAX_RT option code in any Option Request option (see Section 23.7) it sends.

The DHCP server MAY include the SOL_MAX_RT option in any response it sends to a client that has included the SOL_MAX_RT option code in an Option Request option. The SOL_MAX_RT option is sent in the main body of the message to client, not as an encapsulated option in, e.g., an IA_NA, IA_TA, or IA_PD option.

A DHCP client MUST ignore any SOL_MAX_RT option values that are less than 60 or more than 86400.

If a DHCP client receives a message containing a SOL_MAX_RT option that has a valid value for SOL_MAX_RT, the client MUST set its internal SOL_MAX_RT parameter to the value contained in the SOL_MAX_RT option. This value of SOL_MAX_RT is then used by the retransmission mechanism defined in Section 15 and Section 18.1.2.

Updated SOL_MAX_RT value applies only to the network interface on which the client received SOL_MAX_RT option.

23.24. INF_MAX_RT Option

A DHCP server sends the INF_MAX_RT option to a client to override the default value of INF_MAX_RT. The value of INF_MAX_RT in the option replaces the default value defined in Section 6.5. One use for the INF_MAX_RT option is to set a longer value for INF_MAX_RT, which reduces the Information-request traffic from a client that has not received a response to its Information-request messages.

The format of the INF_MAX_RT option is:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          option-code          |         option-len            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       INF_MAX_RT value                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 37: INF_MAX_RT Option Format

option-code OPTION_INF_MAX_RT (83).

option-len 4.

SOL_MAX_RT value Overriding value for INF_MAX_RT in seconds; MUST be in range: 60 <= "value" <= 86400 (1 day).

A DHCP client MUST include the INF_MAX_RT option code in any Option Request option (see Section 23.7) it sends.

The DHCP server MAY include the INF_MAX_RT option in any response it sends to a client that has included the INF_MAX_RT option code in an Option Request option. The INF_MAX_RT option is sent in the main body of the message to client, not as an encapsulated option in, e.g., an IA_NA, IA_TA, or IA_PD option.

A DHCP client MUST ignore any INF_MAX_RT option values that are less than 60 or more than 86400.

If a DHCP client receives a message containing an INF_MAX_RT option that has a valid value for INF_MAX_RT, the client MUST set its internal INF_MAX_RT parameter to the value contained in the INF_MAX_RT option. This value of INF_MAX_RT is then used by the retransmission mechanism defined in Section 15 and Section 19.1.5.

Updated INF_MAX_RT value applies only to the network interface on which the client received INF_MAX_RT option.

24. Security Considerations

The threat to DHCP is inherently an insider threat (assuming a properly configured network where DHCPv6 ports are blocked on the perimeter gateways of the enterprise). Regardless of the gateway configuration, however, the potential attacks by insiders and outsiders are the same.

Use of manually configured preshared keys for IPsec between relay agents and servers does not defend against replayed DHCP messages. Replayed messages can represent a DOS attack through exhaustion of processing resources, but not through mis-configuration or exhaustion of other resources such as assignable addresses.

One attack specific to a DHCP client is the establishment of a malicious server with the intent of providing incorrect configuration information to the client. The motivation for doing so may be to mount a "man in the middle" attack that causes the client to communicate with a malicious server instead of a valid server for some service such as DNS or NTP. The malicious server may also mount a denial of service attack through misconfiguration of the client that causes all network communication from the client to fail.

A malicious DHCP server might cause a client to set its SOL_MAX_RT and INF_MAX_RT parameters to an unreasonably high value with the SOL_MAX_RT and INF_MAX_RT options, which may cause an undue delay in a client completing its DHCP protocol transaction in the case no other valid response is received. Assuming the client also receives a response from a valid DHCP server, large values for SOL_MAX_RT and INF_MAX_RT will not have any effect.

There is another threat to DHCP clients from mistakenly or accidentally configured DHCP servers that answer DHCP client requests with unintentionally incorrect configuration parameters.

A DHCP client may also be subject to attack through the receipt of a Reconfigure message from a malicious server that causes the client to obtain incorrect configuration information from that server. Note that although a client sends its response (Renew or Information-request message) through a relay agent and, therefore, that response will only be received by servers to which DHCP messages are relayed, a malicious server could send a Reconfigure message to a client, followed (after an appropriate delay) by a Reply message that would be accepted by the client. Thus, a malicious server that is not on the network path between the client and the server may still be able to mount a Reconfigure attack on a client. The use of transaction IDs that are cryptographically sound and cannot easily be predicted will also reduce the probability that such an attack will be successful.

The threat specific to a DHCP server is an invalid client masquerading as a valid client. The motivation for this may be for theft of service, or to circumvent auditing for any number of nefarious purposes.

The threat common to both the client and the server is the resource "denial of service" (DoS) attack. These attacks typically involve the exhaustion of available addresses, or the exhaustion of CPU or network bandwidth, and are present anytime there is a shared resource.

In the case where relay agents add additional options to Relay Forward messages, the messages exchanged between relay agents and servers may be used to mount a "man in the middle" or denial of service attack.

This threat model does not consider the privacy of the contents of DHCP messages to be important. DHCP is not used to exchange authentication or configuration information that must be kept secret from other networks nodes.

DHCP authentication provides for authentication of the identity of DHCP clients and servers, and for the integrity of messages delivered between DHCP clients and servers. DHCP authentication does not provide any privacy for the contents of DHCP messages.

The Delayed Authentication protocol described in Section 22.4 uses a secret key that is shared between a client and a server. The use of a "DHCP realm" in the shared key allows identification of administrative domains so that a client can select the appropriate key or keys when roaming between administrative domains. However, the Delayed Authentication protocol does not define any mechanism for sharing of keys, so a client may require separate keys for each administrative domain it encounters. The use of shared keys may not scale well and does not provide for repudiation of compromised keys. This protocol is focused on solving the intradomain problem where the out-of-band exchange of a shared key is feasible.

Because of the opportunity for attack through the Reconfigure message, a DHCP client MUST discard any Reconfigure message that does not include authentication or that does not pass the validation process for the authentication protocol.

The Reconfigure Key protocol described in Section 22.5 provides protection against the use of a Reconfigure message by a malicious DHCP server to mount a denial of service or man-in-the-middle attack on a client. This protocol can be compromised by an attacker that can intercept the initial message in which the DHCP server sends the key to the client.

Communication between a server and a relay agent, and communication between relay agents, can be secured through the use of IPsec, as described in Section 22.1. The use of manual configuration and installation of static keys are acceptable in this instance because relay agents and the server will belong to the same administrative domain and the relay agents will require other specific configuration (for example, configuration of the DHCP server address) as well as the IPsec configuration.

A rogue delegating router can issue bogus prefixes to a requesting router. This may cause denial of service due to unreachability.

A malicious requesting router may be able to mount a denial of service attack by repeated requests for delegated prefixes that exhaust the delegating router’s available prefixes.

To guard against attacks through prefix delegation, requesting routers and delegating routers SHOULD use DHCP authentication as described in Section 22. For point to point links, where one trusts that there is no man in the middle, or one trusts layer two authentication, DHCP authentication or IPsec may not be necessary. Because a requesting router and delegating routers must each have at least one assigned IPv6 address, the routers may be able to use IPsec for authentication of DHCPv6 messages. The details of using IPsec for DHCPv6 are under development.

Networks configured with delegated prefixes should be configured to preclude intentional or inadvertent inappropriate advertisement of these prefixes.

25. IANA Considerations

This document does not define any new DHCPv6 name spaces or definitions.

IANA is requested to update the http://www.iana.org/assignments/dhcpv6-parameters/dhcpv6-parameters.xhtml page to add a reference to this document for definitions previously created by [RFC3315], [RFC3633], and [RFC7083].

26. Acknowledgments

The following people are authors of the original RFC 3315: Ralph Droms, Jim Bound, Bernie Volz, Ted Lemon, Charles Perkins, and Mike Carney. The following people are authors of the original RFC 3633: Ole Troan and Ralph Droms. This document is merely a refinement of their work and would not be possible without their original work.

A number of additional people have contributed to identifying issues with RFC 3315 and RFC 3633 and proposed resolutions to these issues as reflected in this document (in no particular order): Ole Troan, Robert Marks, Leaf Yeh, Tim Winters, Michelle Cotton, Pablo Armando, John Brzozowski, Suresh Krishnan, Hideshi Enokihara, Alexandru Petrescu, Yukiyo Akisada, Tatuya Jinmei, Fred Templin. With special thanks to Ralph Droms for answering many questions related to the original RFC 3315 work.

The following acknowledgements are from the original RFC 3315 and RFC 3633:

Thanks to the DHC Working Group and the members of the IETF for their time and input into the specification. In particular, thanks also for the consistent input, ideas, and review by (in alphabetical order) Bernard Aboba, Bill Arbaugh, Thirumalesh Bhat, Steve Bellovin, A. K. Vijayabhaskar, Brian Carpenter, Matt Crawford, Steve Deering, Francis Dupont, Dave Forster, Brian Haberman, Richard Hussong, Tatuya Jinmei, Kim Kinnear, Fredrik Lindholm, Tony Lindstrom, Josh Littlefield, Gerald Maguire, Jack McCann, Shin Miyakawa, Thomas Narten, Erik Nordmark, Jarno Rajahalme, Yakov Rekhter, Pekka Savola, Mark Stapp, Matt Thomas, Sue Thomson, Tatuya Jinmei, Bernie Volz, Trevor Warwick, Phil Wells and Toshi Yamasaki.

Thanks to Steve Deering and Bob Hinden, who have consistently taken the time to discuss the more complex parts of the IPv6 specifications.

And, thanks to Steve Deering for pointing out at IETF 51 in London that the DHCPv6 specification has the highest revision number of any Internet Draft.

27. References

27.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC0826] Plummer, D., "Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware", STD 37, RFC 826, November 1982.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[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.

[RFC2132] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor Extensions", RFC 2132, March 1997.

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

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2461] Narten, T., Nordmark, E., and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461, December 1998.

[RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, December 1998.

[RFC2526] Johnson, D. and S. Deering, "Reserved IPv6 Subnet Anycast Addresses", RFC 2526, March 1999.

[RFC3118] Droms, R. and W. Arbaugh, "Authentication for DHCP Messages", RFC 3118, June 2001.

[RFC3646] Droms, R., "DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3646, December 2003.

[RFC4075] Kalusivalingam, V., "Simple Network Time Protocol (SNTP) Configuration Option for DHCPv6", RFC 4075, May 2005.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, September 2007.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

[RFC6221] Miles, D., Ooghe, S., Dec, W., Krishnan, S., and A. Kavanagh, "Lightweight DHCPv6 Relay Agent", RFC 6221, May 2011.

[RFC6355] Narten, T. and J. Johnson, "Definition of the UUID-Based DHCPv6 Unique Identifier (DUID-UUID)", RFC 6355, August 2011.

[RFC6724] Thaler, D., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, September 2012.

[RFC7083] Droms, R., "Modification to Default Values of SOL_MAX_RT and INF_MAX_RT", RFC 7083, November 2013.

[RFC7227] Hankins, D., Mrugalski, T., Siodelski, M., Jiang, S., and S. Krishnan, "Guidelines for Creating New DHCPv6 Options", BCP 187, RFC 7227, May 2014.

[RFC7283] Cui, Y., Sun, Q., and T. Lemon, "Handling Unknown DHCPv6 Messages", RFC 7283, July 2014.

27.2. Informative References

[I-D.ietf-dhc-topo-conf]
Lemon, T. and T. Mrugalski, "Customizing DHCP Configuration on the Basis of Network Topology", draft-ietf-dhc-topo-conf-04 (work in progress), January 2015.

[IANA-PEN]
IANA, "Private Enterprise Numbers registry http://www.iana.org/assignments/enterprise-numbers", .

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC2462] Thomson, S. and T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 2462, December 1998.

[RFC3041] Narten, T. and R. Draves, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 3041, January 2001.

[RFC3162] Aboba, B., Zorn, G., and D. Mitton, "RADIUS and IPv6", RFC 3162, August 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.

[RFC3736] Droms, R., "Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6", RFC 3736, April 2004.

[RFC3769] Miyakawa, S. and R. Droms, "Requirements for IPv6 Prefix Delegation", RFC 3769, June 2004.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005.

[RFC7084] Singh, H., Beebee, W., Donley, C., and B. Stark, "Basic Requirements for IPv6 Customer Edge Routers", RFC 7084, November 2013.

[RFC7341] Sun, Q., Cui, Y., Siodelski, M., Krishnan, S., and I. Farrer, "DHCPv4-over-DHCPv6 (DHCP 4o6) Transport", RFC 7341, August 2014.

Appendix A. Changes since RFC3315

1. Incorporated RFC3315 errata (ids: 294, 1373, 2928, 1815, 3577, 2509, 295).

2. Partially incorporated RFC3315 errata id 2472 (place other IA options if NoAddrsAvail is sent in Advertise).

3. Clarified section 21.4.1 of RFC3315 by defining length of "key ID" field and specifying that ’DHCP realm’ is Domain Name encoded as per section 8 of RFC3315. Ticket #43.

4. Added DUID-UUID and reference to RFC6355. Ticket #54.

5. Specified a minimum length for the DUID in section "9.1. DUID Contents". Ticket #39.

6. Removed the use of term "sub-options" from section "19.1.1. Creation and Transmission of Reconfigure Messages". Ticket #40.

7. Added text to section 22.6 "IA Address Option" about the usage of unspecified address to express the client hints for Preferred and Valid lifetimes. Ticket #45.

8. Updated text in 21.4.2 of RFC3315 ("Message Validation") as suggested in section 3.1 of draft-ietf-dhc-dhcpv6-clarify-auth-01. Ticket #87.

9. Merged RFC7083, "Modification to Default Values of SOL_MAX_RT and INF_MAX_RT", into this document. Ticket #51.

10. Incorporated RFC3315 errata (id 2471), into section 17.1.3. Ticket #25.

11. Added text that relay agents MUST NOT modify the relayed message to section 20.1.2. Ticket #57.

12. Modified the text in section 21.4.4.5, Receiving Reply Messages, to remove special treatment of a Reply validation failure (client ignores message). Ticket #89.

13. Appendix C updated: Authentication option is no longer allowed in Relay-forward and Relay-reply messages, ORO is no longer allowed in Confirm, Release and Decline messages; Preference option is no longer allowed in Reply messages (only in Advertise). Ticket #10.

14. Removed "silently" from several instances of "silently ignores" or "silently" discards. It is up to software vendor if and how to log such events (debug log message, event log, message pop-up etc.). Ticket #50.

15. Clarified that: there should be no more that one instance of Vendor Class option with a given Enterprise Number; that one instance of Vendor Class can contain multiple encapsulated

options; the same applies to Vendor Specific Information option. Ticket #22.

16. Clarified relay agent definition. Ticket #12.

17. Changed REL_MAX_RC and DEC_MAX_RC defaults from 5 to 4 and added retry to parameter description. Ticket #84.

18. Clarify handling process for Vendor-specific Information Option and Vendor Class Option. Ticket #20.

19. Replace "monotonic" with "strictly monotonic" in Section 21.3. Ticket #11.

20. Incorporate everything of RFC 6644, except for Security Considerations Section, which has already covered in a more abstracted way. Ticket #55 & #56.

21. Clarify the server behavior process when a client violates Delayed Authentication Protocol, in Section 21.4. Ticket #90.

22. Updated titles of sections 19.4.2. and 19.4.4. to include Rebind messages.

23. Applied many of the review comments from a review done by Fred Templin in August 2006. Ticket #14.

24. Reworded the first paragraph of Section 15 to relax the "SHOULD" requirement to drop the messages which contain the options not expected in the current message. Ticket #17.

25. Changed WG to DHC, added keywords

26. Loosened requirements for DUID-EN, so that DUID type can be used for virtual machines. Ticket #16.

27. Clarified that IA may contain other resources than just address. Ticket #93.

28. Clarified that most options are singletons (i.e. can appear only once). Ticket #83.

29. Merged sections 1 (Ticket #96), 2 (Ticket #97), 3 (Ticket #98), 4 (Ticket #99), 6 (Ticket #101), 8 (Ticket #103), 9 (Ticket #104), 10 (Ticket #105), 11 (Ticket #106), 13 (Ticket #108), 14 (Ticket #109), 15 (Ticket #110), 16 (Ticket #111), 17 (Ticket #112) and 19 (Ticket #113) from RFC3633 (Prefix Delegation).

30. Clarified that encapsulated options must be requested using top level ORO (ticket #38).

31. Clarified that configuration for interface X should be requested over interface X (ticket #48).

32. CONFIRM is now an optional message (MUST send Confirm eased to SHOULD) (ticket #120).

33. Added reference to RFC7227: DHCPv6 Option Guidelines (ticket #121).

34. Added new section 5 providing an overview of DHCPv6 operational modes and removed two prefix delegation sections from section 1. See tickets #53, #100, and #102.

35. Addressed ticket #115 - don’t use DHCPv6 for DHCPv4 configuration.

36. Revised IANA Considerations based on ticket #117.

37. Updated IAID description in the terminology with the clarification that the IAID is unique among IAs of a specific type, rather than globally unique among all IAs (ticket #94).

38. Merged Section 12 from RFC3633 (ticket #107)

39. Clarified behavior for unknown messages (RFC7283), ticket #58.

40. Addressed tickets #123 and #126, and clarified that the client SHOULD abandon its bindings when restarts the server solicitation.

41. Clarified link-address field usage, ticket #73.

Appendix B. Changes since RFC3633

1. Incorporated RFC3633 errata (ids: 248, 1880, 2468, 2469, 2470, 3736)

2. ...

Appendix C. Appearance of Options in Message Types

The following table indicates with a "*" the options are allowed in each DHCP message type:

	Client	Server	IA_NA	IA_PD	Option	Pref	Elap.	Relay	Auth.	Server
	ID	ID	IA_TA		Request		Time	Msg.		Unicast
Solicit	*		*	*	*		*		*	
Advert.	*	*	*	*		*			*	
Request	*	*	*	*	*		*		*	
Confirm	*		*				*		*	
Renew	*	*	*	*	*		*		*	
Rebind	*		*	*	*		*		*	
Decline	*	*	*	*			*		*	
Release	*	*	*	*			*		*	
Reply	*	*	*	*					*	*
Reconf.	*	*			*				*	
Inform.	* (see note)				*		*		*	
R-forw.								*		
R-repl.								*		

NOTE:

Only included in Information-request messages that are sent in response to a Reconfigure (see Section 20.4.3).

	Status	Rap.	User	Vendor	Vendor	Inter.	Recon.	Recon.	SOL_MAX_RT
	Code	Comm.	Class	Class	Spec.	ID	Msg.	Accept	INF_MAX_RT
Solicit		*	*	*	*			*	
Advert.	*		*	*	*			*	*
Request			*	*	*			*	
Confirm			*	*	*				
Renew			*	*	*			*	
Rebind			*	*	*			*	
Decline			*	*	*				
Release			*	*	*				
Reply	*	*	*	*	*			*	*
Reconf.							*		
Inform.			*	*	*			*	
R-forw.			*	*	*	*			
R-repl.			*	*	*	*			

Appendix D. Appearance of Options in the Options Field of DHCP Options

The following table indicates with a "*" where options can appear in the options field of other options:

	Option Field	IA_NA/ IA_TA	IAADDR	IA_PD	IAPREFIX	Relay Forw.	Relay Reply
Client ID	*						
Server ID	*						
IA_NA/IA_TA	*						
IAADDR		*					
IA_PD	*						
IAPREFIX				*			
ORO	*						
Preference	*						
Elapsed Time	*						
Relay Message						*	*
Authentic.	*						
Server Uni.	*						
Status Code	*	*		*			
Rapid Comm.	*						
User Class	*						
Vendor Class	*						
Vendor Info.	*					*	*
Interf. ID						*	*
Reconf. MSG.	*						
Reconf. Accept	*						

Note: "Relay Forw" / "Relay Reply" options appear in the options field of the message but may only appear in these messages.

Authors’ Addresses

Tomek Mrugalski (editor)
Internet Systems Consortium, Inc.
950 Charter Street
Redwood City, CA 94063
USA

Email: tomasz.mrugalski@gmail.com

Marcin Siodelski
Internet Systems Consortium, Inc.
950 Charter St.
Redwood City, CA 94063
USA

Email: msiodelski@gmail.com

Bernie Volz (editor)
Cisco Systems, Inc.
1414 Massachusetts Ave
Boxborough, MA 01719
USA

Email: volz@cisco.com

Andrew Yourtchenko
Cisco Systems, Inc.
De Kleetlaan, 7
Diegem B-1831
Belgium

Email: ayourtch@cisco.com

Michael C. Richardson
Sandelman Software Works
470 Dawson Avenue
Ottawa, ON K1Z 5V7
CA

Email: mcr+ietf@sandelman.ca
URI: http://www.sandelman.ca/

Sheng Jiang
Huawei Technologies Co., Ltd
Q14, Huawei Campus, No.156 Beiqing Road
Hai-Dian District, Beijing, 100095
P.R. China

Email: jiangsheng@huawei.com

Ted Lemon
Nominum, Inc.
950 Charter St.
Redwood City, CA 94043
USA

Email: Ted.Lemon@nominum.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: August 16, 2015

C. Donley
M. Kloberdans
CableLabs
J. Brzozowski
Comcast
C. Grundemann
ISOC

February 12, 2015

Customer Edge Router Identification Option
draft-donley-dhc-cer-id-option-05

Abstract

Addressing mechanisms supporting DHCPv6 Prefix Delegation in home networks such as those described in CableLabs' eRouter specification and the HIPnet Internet-Draft require identification of the customer edge router (CER) as the demarcation between the customer network and the service provider network. This document reserves a DHCPv6 option to identify the CER.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Some addressing mechanisms supporting DHCPv6 Prefix Delegation in home networks such as those described in [I-D.grundemann-homenet-hipnet] and [EROUTER] require identification of the customer edge router as the demarcation between the customer network and the service provider network. For prefix delegation purposes, it is desirable for other routers within the home to know which device is the CER so that the customer home network only requests a single prefix from the ISP DHCPv6 server, and efficiently distributes this prefix within the home. CER-ID is a 128-bit string that optionally represents an IPV6 address, or another arbitrary number. The CER-ID maybe treated as a hint to be used with border detection methods. This document reserves a DHCPv6 option to be used to identify the CER.

1.1. 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. CER Identification Option

A Customer Edge Router (CER) sets the CER_ID to the IPv6 address of its LAN interface. If it has more than one LAN IPv6 address, it selects one of its LAN or other non-WAN IPv6 addresses to be used as the CER_ID. An ISP server does not respond with the CER_ID or sets the CER_ID to ::. Such a response or lack of response indicates to the DHCPv6 client that it is the CER.

The format of the CER Identification option is:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      option-code              |      option-len               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   |                           CER_ID                              |
   |                                                               |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       option-code          OPTION_CER_ID (TBD).
       option-len           36
       CER_ID value         IPv6 address of CER or ::
```

Figure 1.

A DHCPv6 client SHOULD include the CER Identification option code in an Option Request option [RFC3315] in its DHCP Solicit messages.

The DHCPv6 server MAY include the CER Identification option in any response it sends to a client that has included the CER Identification option code in an Option Request option. The CER Identification option is sent in the main body of the message to client, not as a sub-option in, e.g., an IA_NA, IA_TA [RFC3315]option.

When sending the CER Identification option, the DHCPv6 server MUST set the CER_ID value to either one of its IPv6 addresses, another identifier, or ::. If a device does not receive the CER Identification Option or receives a CER ID of :: from the DHCPv6 server, it MUST include one of its Globally Unique IPv6 addresses (unless another identifier is used), in the CER_ID value in response to DHCPv6 messages received by its DHCPv6 server that contains the CER Identification option code in an Option Request option. If the device has only one LAN interface, it SHOULD use its LAN IPv6 address as the CER_ID value. If the device has more than one LAN interface, it SHOULD use the lowest Globally Unique address.

3. CER-ID Compatibility

CER-ID explicitly indicates that a gateway is, or is not, the demarcation point between public and private networks by containing a reachable IPv6 address, other identifier or a double colon '::'

(double colon indicates that the CER-ID sender is NOT the edge router), and as a complement, can be applied to various border definitions and detection methods such as:

- I.D. Draft-IETF-Homenet-Arch-16 [I-D.ietf-homenet-arch]

- I.D. Draft-Grundemann-homenet-HIPnet-01 [I-D.grundemann-homenet-hipnet]

- I.D. Draft-IETF-Kline-Homenet-Default-Perimeter-01 [I-D.kline-default-perimeter]

- Others, including manual configuration

4. IANA Considerations

IANA is requested to assign an option code from the "DHCP Option Codes" Registry for OPTION_CER_ID. IANA is also requested to maintain a list of authentication options.

5. Security Considerations

The security of a home network is an important consideration. Both the HIPNet [I-D.grundemann-homenet-hipnet] and Homenet [I-D.ietf-homenet-arch] approaches change the operational model of the home network vs. today’s IPv4-only paradigm. Specifically, these networks eliminate NAT inside the home network (and only enable it for IPv4 at the edge router, if required), support global addressability of devices, and thus need to consider firewall and/or filter support in various home routers. As the security profile of these home routers can shift based on their position in the network (e.g., edge vs. internal), security can be severely compromised if routers misidentify their border and mistakenly reduce or eliminate firewall rules. If the CER-ID option is used as part of the border detection algorithm, it becomes a natural, but not the only place to enact firewall, NAT, Prefix Delegation and other functions in the home network. Further security is provided using the mechanisms defined in RFC 3315, DHCP for IPv6.

6. Acknowledgements

7. References

7.1. 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.

7.2. Informative References

[EROUTER] CableLabs, "CableLabs IPv4 and IPv6 eRouter Specification (CM-SP-eRouter-I12-131120)", April 2014.

[I-D.grundemann-homenet-hipnet]
Grundemann, C., Donley, C., Brzozowski, J., Howard, L., and V. Kuarsingh, "A Near Term Solution for Home IP Networking (HIPnet)", draft-grundemann-homenet-hipnet-01 (work in progress), February 2013.

[I-D.ietf-homenet-arch]
Chown, T., Arkko, J., Brandt, A., Troan, O., and J. Weil, "IPv6 Home Networking Architecture Principles", draft-ietf-homenet-arch-16 (work in progress), June 2014.

[I-D.kline-default-perimeter]
Kline, E., "Default Border Definition", draft-kline-default-perimeter-01 (work in progress), November 2012.

Authors' Addresses

Chris Donley
CableLabs
858 Coal Creek Cir.
Louisville, CO 80027
US

Email: c.donley@cablelabs.com

Michael Kloberdans
CableLabs
858 Coal Creek Cir
Louisville, CO 80027
US

Email: m.kloberdans@cablelabs.com

John Brzozowski
Comcast
1306 Goshen Parkway
West Chester, PA 19380
US

Email: john_brzozowski@cable.comcast.com

Chris Grundemann
ISOC
Denver CO

Email: cgrundemann@gmail.com

DHC WG I. Farrer
Internet-Draft Deutsche Telekom AG
Intended status: Standards Track Q. Sun
Expires: August 16, 2014 Y. Cui
Tsinghua University
February 12, 2014

DHCPv4 over DHCPv6 Source Address Option
draft-fsc-dhc-dhcp4o6-saddr-opt-00

Abstract

DHCPv4 over DHCPv6 [I-D.ietf-dhc-dhcpv4-over-dhcpv6] describes one possible mechanism for dynamically configuring IPv4 over an IPv6 only network. For DHCPv4 over DHCPv6 to function with some softwire mechanisms, the operator must obtain information about the DHCP 4o6 client’s allocated IPv4 address and PSID, as well as the /128 IPv6 prefix that the client will use as the source of IPv4-in-IPv6 tunnel. This memo defines a DHCPv6 option to convey this IPv6 prefix between the DHCP 4o6 client and server. It is designed to work in conjunction with the DHCPv4 IPv4 address allocation process message flow.

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].

Status of This Memo

This Internet-Draft will expire on August 16, 2014.

Copyright Notice

Table of Contents

1. Introduction

DHCPv4 over DHCPv6 [I-D.ietf-dhc-dhcpv4-over-dhcpv6] describes a mechanism for dynamically configuring IPv4 over an IPv6 only network by transporting the complete set of DHCPv4 messages within a specific pair of DHCPv6 messages. The IPv4 configuration provisioned to the DHCP 4o6 clients is then generally used for configuring IPv4 over IPv6 services. IPv4 addresses can be dynamically leased to DHCP 4o6 clients in the same manner as IPv4 addresses are leased to DHCPv4 clients in IPv4 networks. The main advantages to this approach is a greater efficiency in the use of limited IPv4 address resources over IPv6 networks.

Currently defined IPv4 over IPv6 transition technologies are, by design, quite prescriptive in the location of the tunnel endpoint within the home network. The tunnel endpoint must usually be configured on either the home gateway device, or sourced from a very specific IPv6 tunnel prefix allocated to the home network (and in some cases, both). This is necessary to enable the end-to-end provisioning chain between the IPv4-over-IPv6 client in the home network, the border router (the egress point from the IPv4 over IPv6 domain to the IPv4-only domain) and the provisioning systems responsible for configuring the functional elements.

The dynamic leasing of IPv4 addresses to clients alters this end-to-end provisioning chain. It can no longer be assumed that a Softwire Initiator sourcing from a specific IPv6 prefix have to use a certain IPv6 address as the source for encapsulating its IPv4 packets. Therefore, a mechanism is necessary to inform the service provider of the binding between the allocated IPv4 address (learnt through DHCPv4) and the IPv6 address that the IPv4 over IPv6 client will use for accessing IPv4-over-IPv6 services. The service provider can then use this binding information to provision other functional elements it their network such as the border router accordingly.

A second benefit of such a mechanism is that it allows much more flexibility in the location of the IPv4 over IPv6 tunnel endpoint as this will be dynamically signalled back to the service provider. The DHCP 4o6 client and tunnel client could be run on end devices attached to any valid IPv6 prefix allocated to an end-user, located anywhere within an arbitrary home network topology.

As The DHCP 4o6 server manages the leasing of IPv4 addresses to the DHCP 4o6 clients, which runs on the Softwire Initiators, it holds the most accurate IPv4 lease information available across the IPv6 network between the server and the client. It follows that the DHCP 4o6 server should also hold information about the /128 IPv6 prefixes that active clients are using, so that the server contains a single, comprehensive and up to date dynamic IPv4/IPv6 binding table.

This memo describes a DHCPv6 option so that the server can indicate to the client a preferred IPv6 prefix to use (if necessary) and for the client to signal back the /128 IPv6 prefix that they will bind the allocated IPv4 configuration to. The DHCP 4o6 server then stores this information alongside the IPv4 lease information.

Current mechanisms suitable for extending to incorporate DHCPv4 over DHCPv6 with dynamic IPv4 address leasing include [I-D.ietf-softwire-map] and [I-D.ietf-softwire-lw4over6]. For these mechanisms to function, the operator needs the information about the DHCP 4o6 client’s allocated IPv4 address, PSID and also the /128 IPv6 prefix that the client will use to source the IPv4-in-IPv6 tunnel endpoint.

2. Applicabiliity

Although DHCPv4 over DHCPv6 is used as the configuration protocol throughout this document, the DHCPv6 option and provisioning process which is described here may also be used with other DHCP based IPv4 over IPv6 configuration mechanisms, such as DHCPv4 over IPv6 [I-D.ietf-dhc-dhcpv4-over-ipv6].

3. Solution Overview

The DHCPv6 option (OPTION_DHCPV4OV6_SADDR) described by this memo is intended to be used alongside the normal DHCPv4 IPv4 address allocation message flow as described in [RFC2131], in the context of DHCPv4 over DHCPv6 [I-D.ietf-dhc-dhcpv4-over-dhcpv6] . It is a two-way communication process, allowing the service provider to (optionally) indicate to the client a preferred prefix alongside the DHCPOFFER message, which can be used for binding the received IPv4 configuration and sourcing tunnel traffic. When the client has selected the IPv6 prefix to bind the IPv4 configuration to, it passes this back to the DHCP 4o6 server along with the DHCPREQUEST message. This may be necessary if there are multiple IPv6 prefixes in use in the customer network, or if the specific IPv4 over IPv6 transition mechanism requires the use of a particular prefix for any reason.

The following diagram shows the client/server message flow and how the different fields of OPTION_DHCPV4OV6_SADDR are used. In each step, the relevant DHCPv4 message is given above the arrow and the relevant paramaters used in OPTION_DHCPV4OV6_SADDR’s fields below the arrow.

```
          DHCP 4o6                                     DHCP 4o6
           Client                                       Server
             |                 DHCPDISCOVER                |
      Step 1 |-------------------------------------------->|
             |    OPTION_DHCPV4OV6_SADDR (blank fields)    |
             |                                             |
             |                  DHCPOFFER                  |
      Step 2 |<--------------------------------------------|
             |  OPTION_DHCPV4OV6_SADDR (cipv6-prefix-hint  |
             |  with service provider's preferred prefix)  |
             |                                             |
             |                 DHCPREQUEST                 |
      Step 3 |-------------------------------------------->|
             |  OPTION_DHCPV4OV6_SADDR (cipv6-bound-prefix |
             |   with client's bound /128 IPv6 prefix)     |
             |                                             |
             |                   DHCPACK                   |
      Step 4 |<--------------------------------------------|
             |  OPTION_DHCPV4OV6_SADDR (cipv6-bound-prefix |
             |   with client's bound /128 IPv6 prefix)     |
             |                                             |
```

IPv6/IPv4 Binding Message Flow

The OPTION_DHCPV4OV6_SADDR (defined below) option is included by the DHCP 4o6 client within DHCPv4-query messages. The DHCP 4o6 server MAY reply with this option within DHCPv4-response messages.

The DHCP 4o6 Server and Client MAY implement the OPTION_DHCPV4OV6_SADDR option. If used, this option MUST be present within all future DHCPv4 over DHCPv6 transactions.

The option comprises of two prefixes (with associated prefix length fields):

cipv6-prefix-hint Used by the server to indicate a preferred prefix that the client should use to bind IPv4 configuration to. If this field contains a prefix, the client MUST perform a longest prefix match between cipv6-prefix-hint and all prefixes configured on the device. The selected prefix MUST then be used to bind the received IPv4 configuration to. If this field is left blank, then the client can select any valid IPv6 prefix.

cipv6-bound-prefix Used by the client to inform the DHCP 4o6 Server of the IPv6 prefix that it has bound the IPv4 configuration to. This MUST be a /128 prefix configured on the client.

4. DHCPv4 over DHCPv6 Source Address Option

The format of DHCPv4 over DHCPv6 Source address option is defined as follows:

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |    OPTION_DHCPV4OV6_SADDR     |         option-length         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | cipv6-hintlen |                                               |
    +-+-+-+-+-+-+-+-+          cipv6-prefix-hint                    .
    .                          (variable length)                    .
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |cipv6-boundlen |                                               |
    +-+-+-+-+-+-+-+-+          cipv6-bound-prefix                   .
    .                          (variable length)                    .
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- o option-code: OPTION_DHCPV4OV6_SADDR (TBA)
- o option-length: 2 + length of cipv6-prefix-hint + length of cipv6 -bound-prefix, specified in bytes.
- o cipv6-hintlen: 8-bit field expressing the bit mask length of the IPv6 prefix specified in cipv6-prefix-hint.
- o cipv6-prefix-hint: The IPv6 prefix that the server uses to indicate the preferred prefix that the client should use to bind IPv4 configuration to.
- o cipv6-boundlen: 8-bit field expressing the bit mask length of the IPv6 prefix specified in cipv6-bound-prefix. Default: 128.
- o cipv6-bound-prefix: The IPv6 prefix that the client is using to bind the allocated IPv4 configuration to.

5. Security Considerations

TBD

6. IANA Considerations

IANA is requested to allocate the DHCPv6 option code: OPTION_DHCPV4OV6_SADDR.

7. Acknowledgements

8. References

8.1. Normative References

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

8.2. Informative References

[I-D.ietf-dhc-dhcpv4-over-dhcpv6]
Sun, Q., Cui, Y., Siodelski, M., Krishnan, S., and I. Farrer, "DHCPv4 over DHCPv6 Transport", draft-ietf-dhc-dhcpv4-over-dhcpv6-04 (work in progress), January 2014.

[I-D.ietf-dhc-dhcpv4-over-ipv6]
Cui, Y., Wu, P., Wu, J., Lemon, T., and Q. Sun, "DHCPv4 over IPv6 Transport", draft-ietf-dhc-dhcpv4-over-ipv6-08 (work in progress), October 2013.

[I-D.ietf-softwire-lw4over6]
Cui, Y., Qiong, Q., Boucadair, M., Tsou, T., Lee, Y., and I. Farrer, "Lightweight 4over6: An Extension to the DS-Lite Architecture", draft-ietf-softwire-lw4over6-06 (work in progress), February 2014.

[I-D.ietf-softwire-map-dhcp]
Mrugalski, T., Troan, O., Dec, W., Bao, C., leaf.yeh.sdo@gmail.com, l., and X. Deng, "DHCPv6 Options for configuration of Softwire Address and Port Mapped Clients", draft-ietf-softwire-map-dhcp-06 (work in progress), November 2013.

[I-D.ietf-softwire-map]
Troan, O., Dec, W., Li, X., Bao, C., Matsushima, S., Murakami, T., and T. Taylor, "Mapping of Address and Port with Encapsulation (MAP)", draft-ietf-softwire-map-10 (work in progress), January 2014.

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

[RFC3927] Cheshire, S., Aboba, B., and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses", RFC 3927, May 2005.

[RFC4361] Lemon, T. and B. Sommerfeld, "Node-specific Client Identifiers for Dynamic Host Configuration Protocol Version Four (DHCPv4)", RFC 4361, February 2006.

[RFC6148] Kurapati, P., Desetti, R., and B. Joshi, "DHCPv4 Lease Query by Relay Agent Remote ID", RFC 6148, February 2011.

[RFC6346] Bush, R., "The Address plus Port (A+P) Approach to the IPv4 Address Shortage", RFC 6346, August 2011.

Authors' Addresses

Ian Farrer
Deutsche Telekom AG
CTO-ATI, Landgrabenweg 151
Bonn, NRW 53227
Germany

Email: ian.farrer@telekom.de

Qi Sun
Tsinghua University
Beijing 100084
P.R.China

Phone: +86-10-6278-5822
Email: sunqi@csnet1.cs.tsinghua.edu.cn

Yong Cui
Tsinghua University
Beijing 100084
P.R.China

Phone: +86-10-6260-3059
Email: yong@csnet1.cs.tsinghua.edu.cn

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

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. 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:

o When a backup server contacts its failover partner for the first time.

o 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.

o 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.

```
                               +---------+
                /------------->|  ACTIVE |<--------------\
                |              +---------+               |
                |                |  |  |                 |
                |       /--(8)--/  (3)  \--(9)-\         |
                |      |            |           |        |
                |      V            V           V        |
                |  +-------+   +--------+   +---------+  |
                |  |EXPIRED|   |RELEASED|   |ABANDONED|  |
                |  +-------+   +--------+   +---------+  |
                |      |            |            |       |
                |      |            |           (10)     |
                |      |            |            V       |
                |      |            |       +---------+  |
                |      |            |       |  RESET  |  |
                |      |            |       +---------+  |
                |      |            |            |       |
                |       \--(4)--\  (4)  /--(4)--/        |
                |                |  |  |                 |
               (1)               V  V  V                (2)
                |              /---------\               |
                |              |  FREE*  |               |
                |              \---------/               |
                |                 |   |                  |
                |         /-(5)--/     \-(6)-\          |
                |        |                    |         |
                |        V                    V         |
                |    +-------+        +-----------+     |
                \----|  FREE |<--(7)-->|FREE_BACKUP|-----/
                     +-------+        +-----------+

                             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.

\Resource owner \----------\ Algorithm \	Primary	Secondary
Proportional	FREE	FREE
Independent	FREE	FREE_BACKUP

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.

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

Figure 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)

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

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

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

9.5. RECOVER State

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

9.5.1. Operation in RECOVER State

The server MUST NOT be responsive in RECOVER state.

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

9.5.2. Transition Out of RECOVER State

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

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

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

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

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

```
                   A                                        B
                 Server                                  Server

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

Figure 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:

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

- RECOVER: Stay in COMMUNICATIONS-INTERRUPTED state.

- RECOVER-DONE: Transition into NORMAL state.

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

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

```
      Primary                                Secondary
       Server                                  Server

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

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                                Secondary
       Server                                  Server
```

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

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

Network Working Group
Internet-Draft
Updates: 3315,3633 (if approved)
Intended status: Standards Track
Expires: September 21, 2015

O. Troan
B. Volz
Cisco Systems, Inc.
M. Siodelski
ISC

March 20, 2015

Issues and Recommendations with Multiple Stateful DHCPv6 Options
draft-ietf-dhc-dhcpv6-stateful-issues-12.txt

Abstract

The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) specification defined two stateful options, IA_NA and IA_TA, but did not anticipate the development of additional stateful options. DHCPv6 Prefix Delegation added the IA_PD option, which is stateful. Applications that use IA_NA and IA_PD together have revealed issues that need to be addressed. This document updates RFC 3315 and RFC 3633 to address these issues.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

DHCPv6 [RFC3315] was written without the expectation that additional stateful DHCPv6 options would be developed. DHCPv6 Prefix Delegation [RFC3633] since added a new stateful option for Prefix Delegation to DHCPv6. Implementation experience of the Customer Edge Router (CER) model described in [RFC7084] has shown issues with the DHCPv6 protocol in supporting multiple stateful option types, in particular IA_NA (non-temporary addresses) and IA_PD (delegated prefixes).

This document describes a number of problems encountered with coexistence of the IA_NA and IA_PD option types and specifies changes to the DHCPv6 protocol to address these problems.

The intention of this work is to clarify and, where needed, modify the DHCPv6 protocol specification to support IA_NA and IA_PD option types within a single DHCPv6 session.

Note that while IA_TA (temporary addresses) options may be included with other IA option type requests, these generally are not renewed (there are no T1/T2 times) and have a separate life cycle from IA_NA and IA_PD option types. Therefore, the IA_TA option type is mostly out of scope for this document.

The changes described in this document are intended to be incorporated in a new revision of the DHCPv6 protocol specification ([I-D.dhcwg-dhc-rfc3315bis]).

2. Conventions

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 [RFC2119].

3. Terminology

In addition to the terminology defined in [RFC3315], [RFC3633], and [RFC7227], the following terminology is used in this document:

Identity association (IA): Throughout this document, "IA" is used to refer to the Identity Association containing addresses or prefixes assigned to a client and carried in the IA_NA or IA_PD options respectively.

IA option types: This is used to generally mean an IA_NA and/or IA_PD option.

Stateful options: Options that require dynamic binding state per client on the server.

Top-level options: Top-level options are DHCPv6 options that are not encapsulated within other options, excluding the Relay-Message option. Options encapsulated by Relay-message options, but not by any other option, are still top-level options, whether they appear in a relay agent message or a server message. See [RFC7227].

4. Handling of Multiple IA Option Types

The DHCPv6 specification [RFC3315] was written with the assumption that the only stateful options were for assigning addresses. DHCPv6 Prefix Delegation [RFC3633] describes how to extend the DHCPv6 protocol to handle prefix delegation, but does not clearly specify how the DHCP address assignment and prefix delegation co-exist.

If a client requests multiple IA option types, but the server is configured to only offer a subset of them, the client could react in several ways:

1. Reset the state machine and continue to send Solicit messages,

2. Create separate DHCP sessions for each IA option type and continue to Solicit for the unfulfilled IA options, or

3. The client could continue with the single session, and include the unfulfilled IA options in subsequent messages to the server.

Resetting the state machine and continuing to send Solicit messages may result in the client never completing DHCP and is generally not considered a good solution. It can also result in a packet storm if the client does not appropriately rate limit its sending of Solicit messages or there are many clients on the network. Client implementors that follow this approach, SHOULD implement the updates to RFC-3315 specified in [RFC7083].

Creating a separate DHCP session (separate instances of the client state machine) per IA option type, while conceptually simple, causes a number of issues: additional host resources required to create and maintain multiple instances of the state machine in clients, additional DHCP protocol traffic, unnecessary duplication of other configuration options and the potential for conflict, divergence in that each IA option type specification specifies its 'own' version of the DHCP protocol.

The single session and state machine allows the client to use the best configuration it is able to obtain from a single DHCP server during the configuration exchange. Note, however, that the server may not be configured to deliver the entire configuration requested by the client. In that case the client could continue to operate only using the configuration received, even if other servers can provide the missing configuration. In practice, especially in the case of handling IA_NA and IA_PD, this situation should be rare or a temporary operational error. So, it is more likely for the client to get all configuration if it continues, in each subsequent configuration exchange, to request all the configuration information it is programmed to try to obtain, including any stateful configuration options for which no results were returned in previous exchanges.

One major issue of this last approach is that it is difficult to allow it with the current DHCPv6 specifications; in some cases they are not clear enough, and in other cases existing restrictions can make it impossible. This document introduces some clarifications and small modifications to the current specifications to address these concerns.

While all approaches have their own pros and cons, approach 3 SHOULD be used and is the focus of this document because it is deemed to work best for common cases of the mixed use of IA_NA and IA_PD. But this document does not exclude other approaches. Also, in some corner cases it may not be feasible to maintain a single DHCPv6 session for both IA_NA and IA_PD. These corner cases are beyond the scope of this document and may depend on the network in which the client (CER) is designed to operate and on the functions the client is required to perform.

The sections which follow update RFC 3315 and RFC 3633 to accommodate the recommendation, though many of the changes are also applicable even if other approaches are used.

4.1. Placement of Status Codes in an Advertise Message

In Reply messages IA specific status codes (i.e., NoAddrsAvail, NotOnLink, NoBinding, NoPrefixAvail) are encapsulated in the IA option. In Advertise messages though, the NoAddrsAvail code is returned at in the top level. This makes sense if the client is only interested in the assignment of the addresses and the failure case is fatal. However, if the client sends both IA_NA and IA_PD options in a Solicit message, it is possible that the server offers no addresses but it offers some prefixes, and the client may choose to send a Request message to obtain the offered prefixes. In this case, it is better if the Status Code option for IA specific status codes is encapsulated in the IA option to indicate that the failure occurred for the specific IA. This also makes the NoAddrsAvail and NoPrefixAvail Status Code option placement for Advertise messages identical to Reply messages.

In addition, how a server formats the Advertise message when addresses are not available has been a point of some confusion and implementations seem to vary (some strictly follow RFC 3315 while others assumed it was encapsulated in the IA option as for Reply messages).

We have chosen the following solution:

Clients MUST handle each of the following Advertise messages formats when there are no addresses available (even when no other IA option types were in the Solicit):

1. Advertise containing the IA_NAs and/or IA_TAs with encapsulated Status Code option of NoAddrsAvail and no top-level Status Code option.

2. Advertise containing just a top-level Status Code option of NoAddrsAvail and no IA_NAs/IA_TAs.

3. Advertise containing a top-level Status Code option of NoAddrsAvail and IA_NAs and/or IA_TAs with a Status Code option of NoAddrsAvail.

Note: Clients MUST handle the last two formats listed above to facilitate backward compatibility with the servers which have not been updated to this specification.

See Section 4.2 for updated text for Section 17.1.3 of RFC 3315 and Section 11.1 of RFC 3633.

Servers MUST return the Status Code option of NoAddrsAvail encapsulated in IA_NA/IA_TA options and MUST NOT return a top-level Status Code option of NoAddrsAvail when no addresses will be assigned (1 in the above list). This means that the Advertise response matches the Reply response with respect to the handling of the NoAddrsAvail status.

Replace the following paragraph in RFC 3315, section 17.2.2:

> If the server will not assign any addresses to any IAs in a subsequent Request from the client, the server MUST send an Advertise message to the client that includes only a Status Code option with code NoAddrsAvail and a status message for the user, a Server Identifier option with the server's DUID, and a Client Identifier option with the client's DUID.

With:

> If the server will not assign any addresses to an IA in a subsequent Request from the client, the server MUST include the IA in the Advertise message with no addresses in the IA and a Status Code option encapsulated in the IA containing status code NoAddrsAvail.

4.2. Advertise Message Processing by a Client

[RFC3315] specifies that a client must ignore an Advertise message if a server will not assign any addresses to a client, and [RFC3633] specifies that a client must ignore an Advertise message if a server returns the NoPrefixAvail status to a requesting router. Thus, a client requesting both IA_NA and IA_PD, with a server that only offers either addresses or delegated prefixes, is not supported by the current protocol specifications.

Solution: a client SHOULD accept Advertise messages, even when not all IA option types are being offered. And, in this case, the client SHOULD include the not offered IA option types in its Request. A client SHOULD only ignore an Advertise message when none of the requested IA options include offered addresses or delegated prefixes. Note that ignored messages MUST still be processed for SOL_MAX_RT and INF_MAX_RT options as specified in [RFC7083].

Replace Section 17.1.3 of RFC 3315: (existing errata)

> The client MUST ignore any Advertise message that includes a Status Code option containing the value NoAddrsAvail, with the exception that the client MAY display the associated status message(s) to the user.

With (this includes the changes made by [RFC7083]):

> The client MUST ignore any Advertise message that contains no addresses (IAADDR options encapsulated in IA_NA or IA_TA options) and no delegated prefixes (IAPREFIX options encapsulated in IA_PD options, see RFC 3633) with the exception that the client:
>
> - MUST process an included SOL_MAX_RT option (RFC 7083) and
> - MUST process an included INF_MAX_RT option (RFC 7083).
>
> A client can display any associated status message(s) to the user or activity log.
>
> The client ignoring this Advertise message MUST NOT restart the Solicit retransmission timer.

And, replace:

- The client MAY choose a less-preferred server if that server has a better set of advertised parameters, such as the available addresses advertised in IAs.

With:

- The client MAY choose a less-preferred server if that server has a better set of advertised parameters, such as the available set of IAs, as well as the set of other configuration options advertised.

And, replace the last paragraph of Section 11.1 of RFC 3633 with:

> The requesting router MUST ignore any Advertise message that contains no addresses (IAADDR options encapsulated in IA_NA or IA_TA options) and no delegated prefixes (IAPREFIX options encapsulated in IA_PD options, see RFC 3633) with the exception that the requesting router:
>
> - MUST process an included SOL_MAX_RT option (RFC 7083) and
> - MUST process an included INF_MAX_RT option (RFC 7083).
>
> A client can display any associated status message(s) to the user or activity log.
>
> The requesting router ignoring this Advertise message MUST NOT restart the Solicit retransmission timer.

4.3. T1/T2 Timers

The T1 and T2 times determine when the client will contact the server to extend lifetimes of information received in an IA. How should a client handle the case where multiple IA options have different T1 and T2 times?

In a multiple IA option type model, the T1/T2 times are protocol timers, that should be independent of the IA options themselves. If we were to redo the DHCP protocol from scratch the T1/T2 times should be carried in a separate DHCP option.

Solution: The server MUST set the T1/T2 times in all IA options in a Reply or Advertise message to the same value. To deal with the case where servers have not yet been updated to do that, the client MUST select a T1 and T2 time from all IA options which will guarantee that the client will send Renew/Rebind messages not later than at the T1/T2 times associated with any of the client's bindings.

As an example, if the client receives a Reply with T1_NA of 3600 / T2_NA of 5760 and T1_PD of 0 / T2_PD of 1800, the client SHOULD use the T1_PD of 0 / T2_PD of 1800. The reason for this is that a T1 of 0 means that the Renew time is at the client's discretion, but this value cannot be greater than the T2 value (1800).

The following paragraph should be added to Sections 18.2.1, 18.2.3, and 18.2.4 of RFC 3315:

> The T1/T2 times set in each applicable IA option for a Reply MUST be the same values across all IAs. The server MUST determine the T1/T2 times across all of the applicable client's bindings in the Reply. This facilitates the client being able to renew all of the bindings at the same time.

Note: This additional paragraph has also been included in the revised text later for Sections 18.2.3 and 18.2.4 of RFC 3315.

Changes for client T1/T2 handling are included in Section 4.4.3 and Section 4.4.4.

4.4. Renew and Rebind Messages

This section presents issues with handling multiple IA option types in the context of creation and processing the Renew and Rebind messages. It also introduces relevant updates to the [RFC3315] and [RFC3633].

4.4.1. Renew Message

In multiple IA option type model, the client may include multiple IA options in the Request message, and the server may create bindings only for a subset of the IA options included by the client. For the IA options in the Request message for which the server does not create the bindings, the server sends the IA options in the Reply message with the NoAddrsAvail or NoPrefixAvail status codes.

The client may accept the bindings created by the server, but may desire the other bindings to be created once they become available, e.g. when the server configuration is changed. The client which accepted the bindings created by the server will periodically send a Renew message to extend their lifetimes. However, the Renew message, as described in the [RFC3315], does not support the ability for the client to extend the lifetimes of the bindings for some IAs, while requesting bindings for other IAs.

Solution: The client, which sends a Renew message to extend the lifetimes of the bindings assigned to the client, SHOULD include IA options for these bindings as well as IA options for all other bindings that the client desires but has been unable to obtain. The client and server processing need to be modified. Note that this change makes the server’s IA processing of Renew similar to the Request processing.

4.4.2. Rebind Message

According to the Section 4.4.1, the client includes IA options in a Renew message for the bindings it desires but has been unable to obtain by sending a Request message, apart from the IA options for the existing bindings.

At time T2, the client stops sending Renew messages to the server and initiates the Rebind/Reply message exchange with any available server. In this case, it should be possible to continue trying to obtain new bindings using the Rebind message if the client failed to get the response from the server to the Renew message.

Solution: The client SHOULD continue to include the IA options received from the server and it MAY include additional IA options to request creation of the additional bindings.

4.4.3. Updates to section 18.1.3 of RFC 3315

Replace Section 18.1.3 of RFC 3315 with the following text:

> To extend the valid and preferred lifetimes for the addresses assigned to an IA, the client sends a Renew message to the server from which the addresses were obtained, which includes an IA option for the IA whose address lifetimes are to be extended. The client includes IA Address options within the IA option for the addresses assigned to the IA. The server determines new lifetimes for these addresses according to the administrative configuration of the server. The server may also add new addresses to the IA. The server can remove addresses from the IA by returning IA Address

options for such addresses with preferred and valid lifetimes set to zero.

The server controls the time at which the client contacts the server to extend the lifetimes on assigned addresses through the T1 and T2 parameters assigned to an IA. However, as the client Renews/Rebinds all IAs from the server at the same time, the client MUST select a T1 and T2 time from all IA options which will guarantee that the client will send Renew/Rebind messages not later than at the T1/T2 times associated with any of the client’s bindings.

At time T1, the client initiates a Renew/Reply message exchange to extend the lifetimes on any addresses in the IA.

If T1 or T2 had been set to 0 by the server (for an IA_NA) or there are no T1 or T2 times (for an IA_TA) in a previous Reply, the client may send a Renew or Rebind message, respectively, at the client’s discretion.

The client sets the "msg-type" field to RENEW. The client generates a transaction ID and inserts this value in the "transaction-id" field.

The client places the identifier of the destination server in a Server Identifier option.

The client MUST include a Client Identifier option to identify itself to the server. The client adds any appropriate options, including one or more IA options.

For IAs to which addresses have been assigned, the client includes a corresponding IA option containing an IA Address option for each address assigned to the IA. The client MUST NOT include addresses in any IA option that the client did not obtain from the server or that are no longer valid (that have a zero valid lifetime).

The client MAY include an IA option for each binding it desires but has been unable to obtain. This IA option MUST NOT contain any addresses. However, it MAY contain the IA Address option with IPv6 address field set to 0 to indicate the client’s preference for the preferred and valid lifetimes for any newly assigned addresses.

The client MUST include an Option Request option (see section 22.7) to indicate the options the client is interested in receiving. The client MAY include options with data values as hints to the server about parameter values the client would like to have returned.

The client transmits the message according to section 14, using the following parameters:

IRT	REN_TIMEOUT
MRT	REN_MAX_RT
MRC	0
MRD	Remaining time until T2

The message exchange is terminated when time T2 is reached (see section 18.1.4), at which time the client begins a Rebind message exchange.

4.4.4. Updates to Section 18.1.4 of RFC 3315

Replace Section 18.1.4 of RFC 3315 with the following text:

At time T2 (which will only be reached if the server to which the Renew message was sent at time T1 has not responded), the client initiates a Rebind/Reply message exchange with any available server.

The client constructs the Rebind message as described in 18.1.3 with the following differences:

- The client sets the "msg-type" field to REBIND.
- The client does not include the Server Identifier option in the Rebind message.

The client transmits the message according to section 14, using the following parameters:

IRT	REB_TIMEOUT
MRT	REB_MAX_RT
MRC	0
MRD	Remaining time until valid lifetimes of all addresses in all IAs have expired

If all addresses for an IA have expired the client may choose to include this IA without any addresses (or with only a hint for lifetimes) in subsequent Rebind messages to indicate that the client is interested in assignment of the addresses to this IA.

The message exchange is terminated when the valid lifetimes of all addresses across all IAs have expired, at which time the client uses Solicit message to locate a new DHCP server and sends a Request for the expired IAs to the new server.

4.4.5. Updates to Section 18.1.8 of RFC 3315

Replace Section 18.1.8 of RFC 3315 with the following text:

> Upon the receipt of a valid Reply message in response to a Solicit (with a Rapid Commit option), Request, Confirm, Renew, Rebind or Information-request message, the client extracts the configuration information contained in the Reply. The client MAY choose to report any status code or message from the status code option in the Reply message.
>
> If the client receives a Reply message with a Status Code containing UnspecFail, the server is indicating that it was unable to process the message due to an unspecified failure condition. If the client retransmits the original message to the same server to retry the desired operation, the client MUST limit the rate at which it retransmits the message and limit the duration of the time during which it retransmits the message.
>
> When the client receives a Reply message with a Status Code option with the value UseMulticast, the client records the receipt of the message and sends subsequent messages to the server through the interface on which the message was received using multicast. The client resends the original message using multicast.
>
> When the client receives a NotOnLink status from the server in response to a Confirm message, the client performs DHCP server solicitation, as described in section 17, and client-initiated configuration as described in section 18. If the client receives any Reply messages that do not indicate a NotOnLink status, the client can use the addresses in the IA and ignore any messages that indicate a NotOnLink status.
>
> When the client receives a NotOnLink status from the server in response to a Request, the client can either re-issue the Request without specifying any addresses or restart the DHCP server discovery process (see section 17).
>
> The client SHOULD perform duplicate address detection [17] on each of the received addresses in any IAs, on which it has not performed duplicate address detection during processing of any of the previous Reply messages from the server. The client performs the duplicate address detection before using the received addresses for

the traffic. If any of the addresses are found to be in use on the link, the client sends a Decline message to the server for those addresses as described in section 18.1.7.

If the Reply was received in response to a Solicit (with a Rapid Commit option), Request, Renew or Rebind message, the client updates the information it has recorded about IAs from the IA options contained in the Reply message:

- Record T1 and T2 times.

- Add any new addresses in the IA option to the IA as recorded by the client.

- Update lifetimes for any addresses in the IA option that the client already has recorded in the IA.

- Discard any addresses from the IA, as recorded by the client, that have a valid lifetime of 0 in the IA Address option.

- Leave unchanged any information about addresses the client has recorded in the IA but that were not included in the IA from the server.

Management of the specific configuration information is detailed in the definition of each option in section 22.

The client examines the status code in each IA individually. If the client receives a NoAddrsAvail status code, the client has received no usable addresses in the IA.

If the client can operate with the addresses obtained from the server the client uses addresses and other information from any IAs that do not contain a Status Code option with the NoAddrsAvail status code. The client MAY include the IAs for which it received the NoAddrsAvail status code, with no addresses, in subsequent Renew and Rebind messages sent to the server, to retry obtaining the addresses for these IAs.

If the client cannot operate without the addresses for the IAs for which it received the NoAddrsAvail status code, the client may try another server (perhaps by restarting the DHCP server discovery process).

If the client finds no usable addresses in any of the IAs, it may either try another server (perhaps restarting the DHCP server discovery process) or use the Information-request message to obtain other configuration information only.

When the client receives a Reply message in response to a Renew or Rebind message, the client:

- sends a Request message if any of the IAs in the Reply message contains the NoBinding status code. The client places IA options in this message for only those IAs for which the server returned the NoBinding status code in the Reply message. The client continues to use other bindings for which the server did not return an error

- sends a Renew/Rebind if any of the IAs is not in the Reply message, but in this case the client MUST limit the rate at which it sends these messages, to avoid the Renew/Rebind storm

- otherwise accepts the information in the IA.

When the client receives a valid Reply message in response to a Release message, the client considers the Release event completed, regardless of the Status Code option(s) returned by the server.

When the client receives a valid Reply message in response to a Decline message, the client considers the Decline event completed, regardless of the Status Code option(s) returned by the server.

4.4.6. Updates to Section 18.2.3 of RFC 3315

Replace Section 18.2.3 of RFC 3315 with the following text:

When the server receives a Renew message via unicast from a client to which the server has not sent a unicast option, the server discards the Renew message and responds with a Reply message containing a Status Code option with the value UseMulticast, a Server Identifier option containing the server’s DUID, the Client Identifier option from the client message, and no other options.

For each IA in the Renew message from a client, the server locates the client’s binding and verifies that the information in the IA from the client matches the information stored for that client.

If the server finds the client entry for the IA the server sends back the IA to the client with new lifetimes and, if applicable, T1/T2 times. If the server is unable to extend the lifetimes of an address in the IA, the server MAY choose not to include the IA Address option for this address.

The server may choose to change the list of addresses and the lifetimes of addresses in IAs that are returned to the client.

If the server finds that any of the addresses in the IA are not appropriate for the link to which the client is attached, the server returns the address to the client with lifetimes of 0.

For each IA for which the server cannot find a client entry, the server has the following choices depending on the server's policy and configuration information:

- If the server is configured to create new bindings as a result of processing Renew messages, the server SHOULD create a binding and return the IA with allocated addresses with lifetimes and, if applicable, T1/T2 times and other information requested by the client. The server MAY use values in the IA Address option (if included) as a hint.

- If the server is configured to create new bindings as a result of processing Renew messages, but the server will not assign any addresses to an IA, the server returns the IA option containing a Status Code option with the NoAddrsAvail status code and a status message for a user.

- If the server does not support creation of new bindings for the client sending a Renew message, or if this behavior is disabled according to the server's policy or configuration information, the server returns the IA option containing a Status code option with the NoBinding status code and a status message for a user.

The server constructs a Reply message by setting the "msg-type" field to REPLY, and copying the transaction ID from the Renew message into the transaction-id field.

The server MUST include a Server Identifier option containing the server's DUID and the Client Identifier option from the Renew message in the Reply message.

The server includes other options containing configuration information to be returned to the client as described in section 18.2.

The T1/T2 times set in each applicable IA option for a Reply MUST be the same values across all IAs. The server MUST determine the T1/T2 times across all of the applicable client's bindings in the Reply. This facilitates the client being able to renew all of the bindings at the same time.

4.4.7. Updates to Section 18.2.4 of RFC 3315

Replace Section 18.2.4 of RFC 3315 with the following text:

> When the server receives a Rebind message that contains an IA option from a client, it locates the client's binding and verifies that the information in the IA from the client matches the information stored for that client.
>
> If the server finds the client entry for the IA and the server determines that the addresses in the IA are appropriate for the link to which the client's interface is attached according to the server's explicit configuration information, the server SHOULD send back the IA to the client with new lifetimes and, if applicable, T1/T2 times. If the server is unable to extend the lifetimes of an address in the IA, the server MAY choose not to include the IA Address option for this address.
>
> If the server finds the client entry for the IA and any of the addresses are no longer appropriate for the link to which the client's interface is attached according to the server's explicit configuration information, the server returns the address to the client with lifetimes of 0.
>
> If the server cannot find a client entry for the IA, the IA contains addresses and the server determines that the addresses in the IA are not appropriate for the link to which the client's interface is attached according to the server's explicit configuration information, the server MAY send a Reply message to the client containing the client's IA, with the lifetimes for the addresses in the IA set to 0. This Reply constitutes an explicit notification to the client that the addresses in the IA are no longer valid. In this situation, if the server does not send a Reply message it silently discards the Rebind message.
>
> Otherwise, for each IA for which the server cannot find a client entry, the server has the following choices depending on the server's policy and configuration information:
>
> - If the server is configured to create new bindings as a result of processing Rebind messages (also see the note about the Rapid Commit option below), the server SHOULD create a binding and return the IA with allocated addresses with lifetimes and, if applicable, T1/T2 times and other information requested by the client. The server MAY use values in the IA Address option (if included) as a hint.

- If the server is configured to create new bindings as a result of processing Rebind messages, but the server will not assign any addresses to an IA, the server returns the IA option containing a Status Code option with the NoAddrsAvail status code and a status message for a user.

- If the server does not support creation of new bindings for the client sending a Rebind message, or if this behavior is disabled according to the server’s policy or configuration information, the server returns the IA option containing a Status Code option with the NoBinding status code and a status message for a user.

When the server creates new bindings for the IA it is possible that other servers also create bindings as a result of receiving the same Rebind message. This is the same issue as in the Discussion under the Rapid Commit option, see section 22.14. Therefore, the server SHOULD only create new bindings during processing of a Rebind message if the server is configured to respond with a Reply message to a Solicit message containing the Rapid Commit option.

The server constructs a Reply message by setting the "msg-type" field to REPLY, and copying the transaction ID from the Rebind message into the transaction-id field.

The server MUST include a Server Identifier option containing the server’s DUID and the Client Identifier option from the Rebind message in the Reply message.

The server includes other options containing configuration information to be returned to the client as described in section 18.2.

The T1/T2 times set in each applicable IA option for a Reply MUST be the same values across all IAs. The server MUST determine the T1/T2 times across all of the applicable client’s bindings in the Reply. This facilitates the client being able to renew all of the bindings at the same time.

4.4.8. Updates to RFC 3633

Replace the following text in Section 12.1 of RFC 3633:

> Each prefix has valid and preferred lifetimes whose durations are specified in the IA_PD Prefix option for that prefix. The requesting router uses Renew and Rebind messages to request the extension of the lifetimes of a delegated prefix.

With:

> Each prefix has valid and preferred lifetimes whose durations are specified in the IA_PD Prefix option for that prefix. The requesting router uses Renew and Rebind messages to request the extension of the lifetimes of a delegated prefix.
>
> The requesting router MAY include IA_PD options without any prefixes, i.e. without IA Prefix option or with IPv6 prefix field of IA Prefix option set to 0, in a Renew or Rebind message to obtain bindings it desires but has been unable to obtain. The requesting router MAY set the prefix-length field of the IA Prefix option as a hint to the server. As in [RFC3315], the requesting router MAY also provide lifetime hints in the IA Prefix option.

Replace the following text in Section 12.2 of RFC 3633:

> The delegating router behaves as follows when it cannot find a binding for the requesting router’s IA_PD:

With:

> For the Renew or Rebind, if the IA_PD contains no IA Prefix option or it contains an IA Prefix option with the IPv6 prefix field set to 0, the delegating router SHOULD assign prefixes to the IA_PD according to the delegating router’s explicit configuration information. In this case, if the IA_PD contains an IA Prefix option with the IPv6 prefix field set to 0, the delegating router MAY use the value in the prefix-length field of the IA Prefix option as a hint for the length of the prefixes to be assigned. The delegating router MAY also respect lifetime hints provided by the requesting router in the IA Prefix option.
>
> The delegating router behaves as follows when it cannot find a binding for the requesting router’s IA_PD containing prefixes:

4.5. Confirm Message

The Confirm message, as described in [RFC3315], is specific to address assignment. It allows a server without a binding to reply to the message, under the assumption that the server only needs knowledge about the prefix(es) on the link, to inform the client that the address is likely valid or not. This message is sent when e.g. the client has moved and needs to validate its addresses. Not all bindings can be validated by servers and the Confirm message provides for this by specifying that a server that is unable to determine the on-link status MUST NOT send a Reply.

Note: Confirm has a specific meaning and does not overload Renew/ Rebind. It also is lower processing cost as the server does NOT need to extend lease times or otherwise send back other configuration options.

The Confirm message is used by the client to verify that it has not moved to a different link. For IAs with addresses, the mechanism used to verify if a client has moved or not, is by matching the link’s on-link prefix(es) (typically a /64) against the prefix-length first bits of the addresses provided by the client in the IA_NA or IA_TA IA-types. As a consequence Confirm can only be used when the client has an IA with address(es) (IA_NA or IA_TA).

A client MUST have a binding including an IA with addresses to use the Confirm message. A client with IAs with addresses as well as other IA-types MAY, depending on the IA-type, use the Confirm message to detect if the client has moved to a different link. A client that does not have a binding with an IA with addresses MUST use the Rebind message instead.

IA_PD requires verification that the delegating router (server) has the binding for the IAs. In that case a requesting router (client) MUST use the Rebind message in place of the Confirm message and it MUST include all of its bindings, even address IAs.

Note that Section 18.1.2 of RFC 3315 states that a client MUST initiate a Confirm when it may have moved to a new link. This is relaxed to a SHOULD as a client may have determined whether it has or has not moved using other techniques, such as described in [RFC6059]. And, as stated above, a client with delegated prefixes, MUST send a Rebind instead of a Confirm.

4.6. Decline Should Not Necessarily Trigger a Release

Some client implementations have been found to send a Release message for other bindings they may have received after they determine a conflict and have correctly sent a Decline message for the conflicting address(es).

A client SHOULD NOT send a Release message for other bindings it may have received just because it sent a Decline message. The client SHOULD retain the non-conflicting bindings. The client SHOULD treat the failure to acquire a binding as a result of the conflict, to be equivalent to not having received the binding, insofar as it behaves when sending Renew and Rebind messages.

4.7. Multiple Provisioning Domains

This document has assumed that all DHCP servers on a network are in a single provisioning domain and thus should be "equal" in the service that they offer. This was also assumed by [RFC3315] and [RFC3633].

One could envision a network where the DHCP servers are in multiple provisioning domains, and it may be desirable to have the DHCP client obtain different IA types from different provisioning domains. How a client detects the multiple provisioning domains and how it would interact with the multiple servers in these different domains is outside the scope of this document (see [I-D.ietf-mif-mpvd-arch] and [I-D.ietf-mif-mpvd-dhcp-support]).

5. IANA Considerations

This specification does not require any IANA actions.

6. Security Considerations

There are no new security considerations pertaining to this document.

7. Acknowledgements

Thanks to many people that contributed to identify the stateful issues addressed by this document and for reviewing drafts of the document, including Ralph Droms, John Brzozowski, Ted Lemon, Hemant Singh, Wes Beebee, Gaurau Halwasia, Bud Millword, Tim Winters, Rob Shakir, Jinmei Tatuya, Andrew Yourtchenko, Fred Templin, Tomek Mrugalski, Suresh Krishnan, and Ian Farrer.

8. References

8.1. 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.

[RFC7083] Droms, R., "Modification to Default Values of SOL_MAX_RT and INF_MAX_RT", RFC 7083, November 2013.

8.2. Informative References

[I-D.dhcwg-dhc-rfc3315bis]
Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., and T. Lemon, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6) bis", draft-dhcwg-dhc-rfc3315bis-04 (work in progress), February 2015.

[I-D.ietf-mif-mpvd-arch]
Anipko, D., "Multiple Provisioning Domain Architecture", draft-ietf-mif-mpvd-arch-11 (work in progress), March 2015.

[I-D.ietf-mif-mpvd-dhcp-support]
Krishnan, S., Korhonen, J., and S. Bhandari, "Support for multiple provisioning domains in DHCPv6", draft-ietf-mif-mpvd-dhcp-support-01 (work in progress), March 2015.

[RFC6059] Krishnan, S. and G. Daley, "Simple Procedures for Detecting Network Attachment in IPv6", RFC 6059, November 2010.

[RFC7084] Singh, H., Beebee, W., Donley, C., and B. Stark, "Basic Requirements for IPv6 Customer Edge Routers", RFC 7084, November 2013.

[RFC7227] Hankins, D., Mrugalski, T., Siodelski, M., Jiang, S., and S. Krishnan, "Guidelines for Creating New DHCPv6 Options", BCP 187, RFC 7227, May 2014.

Authors’ Addresses

Ole Troan
Cisco Systems, Inc.
Philip Pedersens vei 20
N-1324 Lysaker
Norway

Email: ot@cisco.com

Bernie Volz
Cisco Systems, Inc.
1414 Massachusetts Ave
Boxborough, MA 01719
USA

Email: volz@cisco.com

Marcin Siodelski
ISC
950 Charter Street
Redwood City, CA 94063
USA

Email: msiodelski@gmail.com

Network Working Group T. Lemon
Internet-Draft Nominum, Inc.
Intended status: Informational T. Mrugalski
Expires: January 9, 2017 ISC
July 8, 2016

Customizing DHCP Configuration on the Basis of Network Topology
draft-ietf-dhc-topo-conf-09

Abstract

DHCP servers have evolved over the years to provide significant functionality beyond that which is described in the DHCP base specifications. One aspect of this functionality is support for context-specific configuration information. This memo describes some such features and explains their operation.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The DHCPv4 [RFC2131] and DHCPv6 [RFC3315] protocol specifications describe how addresses can be allocated to clients based on network topology information provided by the DHCP relay infrastructure. Address allocation decisions are integral to the allocation of addresses and prefixes in DHCP.

The DHCP protocol also describes mechanisms for provisioning devices with additional configuration information; for example, DNS [RFC1034] server addresses, default DNS search domains, and similar information.

Although it was the intent of the authors of these specifications that DHCP servers would provision devices with configuration information appropriate to each device’s location on the network, this practice was never documented, much less described in detail.

Existing DHCP server implementations do in fact provide such capabilities; the goal of this document is to describe those capabilities for the benefit both of operators and of protocol designers who may wish to use DHCP as a means for configuring their own services, but may not be aware of the capabilities provided by most modern DHCP servers.

2. Terminology

- CPE device: Customer Premise Equipment device. Typically a router belonging to the customer that connects directly to the provider link.

- DHCP, DHCPv4, and DHCPv6. DHCP refers to the Dynamic Host Configuration Protocol in general and applies to both DHCPv4 and DHCPv6. The terms DHCPv4 and DHCPv6 are used in contexts where it is necessary to avoid ambiguity and explain differences.

- PE router: Provider Edge router. The provider router closest to the customer.

- Routable IP address: an IP address with a scope of use wider than the local link.

- Shared subnet: a case where two or more subnets of the same protocol family are available on the same link. ’Shared subnet’ terminology is typically used in Unix environments. It is typically called ’multinet’ in Windows environment. The administrative configuration inside a Microsoft DHCP server is called ’DHCP Superscope’.

3. Identifying Client’s Location by DHCP Servers

Figure 1 illustrates a small hierarchy of network links with Link D serving as a backbone to which the DHCP server is attached.

Figure 2 illustrates a more complex case. Although some of its aspects are unlikely to be seen in actual production networks, they are beneficial for explaining finer aspects of the DHCP protocols. Note that some nodes act as routers (which forward all IPv6 traffic) and some are relay agents (i.e. run DHCPv6 specific software that forwards only DHCPv6 traffic).

```
              Link A                   Link B
           |===+===========|    |===========+======|
               |                            |
               |                            |
           +---+---+                    +---+---+
           | relay |                    | relay |
           |   A   |                    |   B   |
           +---+---+                    +---+---+
               |                            |
               |       Link C               |
           |===+==========+=================+======|
                          |
                          |
                     +----+---+        +--------+
                     | router |        |  DHCP  |
                     |    A   |        | Server |
                     +----+---+        +----+---+
                          |                 |
                          |                 |
                          |   Link D        |
           |==============+=================+======|
                          |
                          |
                     +----+---+
                     | router |
                     |    B   |
                     +----+---+
                          |
                          |
           |===+==========+=================+======|
               |       Link E               |
               |                            |
           +---+---+                    +---+---+
           | relay |                    | relay |
           |   C   |                    |   D   |
           +---+---+                    +---+---+
               |                            |
               |                            |
           |===+===========|    |===========+======|
              Link F                   Link G
```

Figure 1: A simple network with a small hierarchy of links

```
              Link A                   Link B            Link H
           |===+===========|   |=========+======|  |======+======|
               |                         |                |
               |                         |                |
           +---+---+                 +---+---+        +---+---+
           | relay |                 | relay |        | relay |
           |   A   |                 |   B   |        |   G   |
           +---+---+                 +---+---+        +---+---+
               |                         |                |
               |       Link C            |                | Link J
           |===+==========+==============+======|  |======+======|
                          |                               |
                          |                               |
                     +----+---+        +--------+     +---+---+
                     | router |        |  DHCP  |     | relay |
                     |    A   |        | Server |     |   F   |
                     +----+---+        +----+---+     +---+---+
                          |                 |             |
                          |                 |             |
                          |   Link D        |             |
           |==============+=========+=======+=============+======|
                          |         |
                          |         |
                     +----+---+ +---+---+
                     | router | | relay |
                     |    B   | |   E   |
                     +----+---+ +---+---+
                          |         |
                          |         |
           |===+==========+=========+=======+======|
               |       Link E               |
               |                            |
           +---+---+                    +---+---+
           | relay |                    | relay |
           |   C   |                    |   D   |
           +---+---+                    +---+---+
               |                            |
               |                            |
           |===+===========|    |===========+======|
              Link F                   Link G
```

Figure 2: Complex network

Those diagrams allow us to represent a variety of different network configurations and illustrate how existing DHCP servers can provide configuration information customized to the particular location from which a client is making its request.

It is important to understand the background of how DHCP works when considering those diagrams. It is assumed that the DHCP clients may not have routable IP addresses when they are attempting to obtain configuration information.

The reason for making this assumption is that one of the functions of DHCP is to bootstrap the DHCP client's IP address configuration; if the client does not yet have an IP address configured, it cannot route packets to an off-link DHCP server, therefore some kind of relay mechanism is required.

The details of how packet delivery between clients and servers works are different between DHCPv4 and DHCPv6, but the essence is the same: whether or not the client actually has an IP configuration, it generally communicates with the DHCP server by sending its requests to a DHCP relay agent on the local link; this relay agent, which has a routable IP address, then forwards the DHCP requests to the DHCP server (directly or via other relays). In later stages of the configuration when the client has acquired an address and certain conditions are met, it is possible for the client to send packets directly to the server, thus bypassing the relays. The conditions for such behavior are different for DHCPv4 and DHCPv6 and are discussed in sections Section 3.1 and Section 3.2.

To determine the client's point of attachment and link specific configuration, the server typically uses the client facing IP address of the relay agent. In some cases the server may use the routable IP address of the client, if the client has the routable IP address assigned to its interface and it is transmitted in the DHCP message. The server is then able to determine the client's point of attachment and select appropriate subnet- or link-specific configuration.

Sometimes it is useful for the relay agents to provide additional information about the topology. A number of extensions have been defined for this purpose. The specifics are different, but the core principle remains the same: the relay agent knows exactly where the original request came from, so it provides an identifier that will help the server to choose appropriate address pool and configuration parameters. Examples of such options are mentioned in the following sections.

Finally, clients may be connected to the same link as the server, so no relay agents are required. In such cases, the DHCPv4 server typically uses the IPv4 address assigned to the network interface over which the transmission was received to select an appropriate subnet. This is more complicated for DHCPv6, as the DHCPv6 server is not required to have any globally unique addresses. In such cases, additional configuration information may need to be required. Some servers allow indicating that a given subnet is directly reachable over a specific local network interface.

3.1. DHCPv4 Specific Behavior

In some cases in DHCPv4, when a DHCPv4 client has a routable IPv4 address, the message is unicast to the DHCPv4 server rather than going through a relay agent. Examples of such transmissions are renewal (DHCPREQUEST) and address release (DHCPRELEASE).

The relay agent that receives client's message sets giaddr field to the address of the network interface the message was received on. The relay agent may insert a relay agent option [RFC3046].

There are several options defined that are useful for subnet selection in DHCPv4. [RFC3527] defines the Link Selection sub-option that is inserted by a relay agent. This option is particularly useful when the relay agent needs to specify the subnet/link on which a DHCPv4 client resides, which is different from an IP address that can be used to communicate with the relay agent. The Virtual Subnet Selection sub-option, specified in [RFC6607], can also be added by a relay agent to specify information in a VPN environment. In certain cases, it is useful for the client itself to specify the Virtual Subnet Selection option, e.g. when there are no relay agents involved during the VPN set up process.

Another option that may influence the subnet selection is the IPv4 Subnet Selection Option, defined in [RFC3011], which allows the client to explicitly request allocation from a given subnet.

3.2. DHCPv6 Specific Behavior

In DHCPv6 unicast communication is possible in case where the server is configured with a Server Unicast option (see Section 22.12 in [RFC3315]) and clients are able to take advantage of it. In such cases, once a client is assigned a, presumably global, address, it is able to contact the server directly, bypassing any relays. It should be noted that such a mode is completely controllable by administrators in DHCPv6. (They may simply choose to not configure server unicast option, thus forcing clients to send their messages always via relay agents in every case).

In the DHCPv6 protocol, there are two core mechanisms defined in [RFC3315] that allow a server to distinguish which link the relay agent is connected to. The first mechanism is the link-address field in the Relay-forward and Relay-reply messages. Somewhat contrary to its name, relay agents insert in the link-address field an address that is typically global and can be used to uniquely identify the link on which the client is located. In normal circumstances this is the solution that is easiest to maintain, as existing address assignments can be used and no additional administrative actions (like assigning dedicated identifiers for each relay agent, making sure they are unique and maintaining a list of such identifiers) are needed. It requires, however, for the relay agent to have an address with a scope larger than link-local configured on its client-facing interface.

The second mechanism uses Interface-Id option (see Section 22.18 of [RFC3315]) inserted by the relay agent, which identifies the link that the client is connected to. This mechanism may be used when the relay agent does not have a globally unique address or ULA [RFC4193] configured on its client-facing interface, thus making the first mechanism not feasible. If the interface-id is unique within an administrative domain, the interface-id value may be used to select the appropriate subnet. As there is no guarantee for the uniqueness ([RFC3315] only mandates the interface-id to be unique within a single relay agent context), it is up to the administrator to check whether the relay agents deployed use unique interface-id values. If the interface-id values are not unique, the Interface-id option cannot be used to determine the client’s point of attachment.

It should be noted that Relay-forward and Relay-reply messages are exchanged between relays and servers only. Clients are never exposed to those messages. Also, servers never receive Relay-reply messages. Relay agents must be able to process both Relay-forward (sending already relayed message further towards the server, when there is more than one relay agent in a chain) and Relay-reply (when sending back the response towards the client, when there is more than one relay agent in a chain).

For completeness, we also mention an uncommon, but valid case, where relay agents use a link-local address in the link-address field in relayed Relay-forward messages. This may happen if the relay agent doesn’t have any address with a larger scope on the interface connected to that specific link. Even though link-local addresses cannot be automatically used to associate relay agent with a given link, with additional configuration information the server may still be able to select the proper link. That requires the DHCP server software to be able to specify relay agent link-address associated with each link or a feature similar to ’shared subnets’ (see Section 8). Both may or may not be supported by the server software. Network administrator has to manually configure additional information that a given subnet uses a relay agent with link-address X. Alternatively, if the relay agent uses link address X and relays messages from a subnet A, an administrator can configure that subnet A is a shared subnet with a very small X/128 subnet. That is not a recommended configuration, but in cases where it is impossible for relay agents to get an address from the subnet they are relaying from, it may be a viable solution.

DHCPv6 also has support for more finely grained link identification, using Lightweight DHCPv6 Relay Agents [RFC6221] (LDRA). In this case, the link-address field is set to unspecified address (::), but the DHCPv6 server also receives an Interface-Id option from the relay agent that can be used to more precisely identify the client’s location on the network. It is possible to mix LDRA and regular relay agents in the same network. See Sections 7.2 and 7.3 in [RFC6221] for detailed examples.

What this means in practice is that the DHCP server in all cases has sufficient information to pinpoint, at the very least, the layer 3 link to which the client is connected, and in some cases which layer 2 link the client is connected to, when the layer 3 link is aggregated out of multiple layer 2 links.

In all cases, then, the DHCPv6 server will have a link-identifying IP address, and in some cases it may also have a link-specific identifier (e.g. Interface-Id Option or Link Address Option defined in Section 5 of [RFC6977]). It should be noted that the link-specific identifier is unique only within the scope of the link-identifying IP address. For example, link-specific identifier of "eth0" assigned to a relay agent using IPv6 address 2001:db8::1 is distinct from a "eth0" identifier used by a different relay agent with address 2001:db8::2.

It is also possible for link-specific identifiers to be nested, so that the actual identifier that identifies the link is an aggregate of two or more link-specific identifiers sent by a set of LDRAs in a chain; in general this functions exactly as if a single identifier were received from a single LDRA, so we do not treat it specially in the discussion below, but sites that use chained LDRA configurations will need to be aware of this when configuring their DHCPv6 servers.

The Virtual Subnet Selection Options, present in DHCPv4, are also defined for DHCPv6. The use case is the same as in DHCPv4: the relay agent inserts VSS options that can help the server to select the appropriate subnet with its address pool and associated configuration options. See [RFC6607] for details.

4. Simple Subnetted Network

Consider Figure 1 in the context of a simple subnetted network. In this network, there are four leaf subnets: links A, B, F and G, on which DHCP clients will be configured. Relays A, B, C and D in this example are represented in the diagram as IP routers with an embedded relay function, because this is a very typical configuration, but the relay function can also be provided in a separate node on each link.

In a simple network like this, there may be no need for link-specific configuration in DHCPv6, since local routing information is delivered through router advertisements. However, in IPv4, it is very typical to configure the default route using DHCP; in this case, the default route will be different on each link. In order to accomplish this, the DHCP server will need link-specific configuration for the default route.

To illustrate, we will use an example from a hypothetical DHCP server that uses a simple JSON notation [RFC7159] for configuration. Although we know of no DHCP server that uses this specific syntax, most modern DHCP server provides similar functionality.

```
{
    "prefixes": {
        "192.0.2.0/26": {
            "options": {
                "routers": ["192.0.2.1"]
            },
            "on-link": ["A"]
        },
        "192.0.2.64/26": {
            "options": {
                "routers": ["192.0.2.65"]
            },
            "on-link": ["B"]
        },
        "192.0.2.128/26": {
            "options": {
                "routers": ["192.0.2.129"]
            },
            "on-link": ["F"]
        },
        "192.0.2.192/26": {
            "options": {
                "routers": ["192.0.2.193"]
            },
            "on-link": ["G"]
        }
    }
}
```

Figure 3: Configuration Example

In Figure 3, we see a configuration example for this scenario: a set of prefixes, each of which has a set of options and a list of links for which it is on-link. We have defined one option for each prefix: a routers option. This option contains a list of values; each list only has one value, and that value is the IP address of the router specific to the prefix.

When the DHCP server receives a request, it searches the list of prefixes for one that encloses the link-identifying IP address provided by the client or relay agent. The DHCP server then examines the options list associated with that prefix and returns those options to the client.

So for example a client connected to link A in the example would have a link-identifying IP address within the 192.0.2.0/26 prefix, so the DHCP server would match it to that prefix. Based on the configuration, the DHCP server would then return a routers option containing a single IP address: 192.0.2.1. A client on link F would have a link-identifying address in the 192.0.2.128/26 prefix, and would receive a routers option containing the IP address 192.0.2.129.

5. Relay Agent Running on a Host

A relay agent is DHCP software that may be run on any IP node. Although it is typically run on a router, this is by no means required by the DHCP protocol. The relay agent is simply a service that operates on a link, receiving link-local multicasts (IPv6) or broadcasts (IPv4) and relaying them, using IP routing, to a DHCP server. As long as the relay has an IP address on the link, and a default route or more specific route through which it can reach a DHCP server, it need not be a router, or even have multiple interfaces.

A relay agent can be run on a host connected to two links. That case is presented in Figure 2. There is router B that is connected to links D and E. At the same time there is also a host that is connected to the same links. The relay agent software is running on that host. That is uncommon, but a valid configuration.

6. Cascaded Relays

Let’s observe another case, shown in Figure 2. Note that in this configuration, the clients connected to link G will send their requests to relay D which will forward its packets directly to the DHCP server. That is typical, but not the only possible configuration. It is possible to configure relay agent D to forward client messages to relay E which in turn will send it to the DHCP server. This configuration is sometimes referred to as cascaded relay agents.

Note that the relaying mechanism works differently in DHCPv4 and in DHCPv6. In DHCPv4 only the first relay is able to set the giaddr field in the DHCPv4 packet. Any following relays that receive that packet will not change it as the server needs giaddr information from the first relay (i.e. the closest to the client). The server will send the response back to the giaddr address, which is the address of the first relay agent that saw the client’s message. That means that the client messages travel on a different path than the server’s responses. A message from client connected to link G will travel via relay D, relay E and to the server. A response message will be sent from the server to relay D via router B, and relay D will send it to the client on link G.

Relaying in DHCPv6 is more structured. Each relay agent encapsulates a packet that is destined to the server and sends it towards the server. Depending on the configuration, that can be a server’s unicast address, a multicast address or next relay agent address. The next relay repeats the encapsulation process. Although the resulting packet is more complex (may have up to 32 levels of encapsulation if the packet traveled through 32 relays), every relay may insert its own options and it is clear which relay agent inserted which option.

7. Regional Configuration Example

In the Figure 2 example, link C is a regional backbone for an ISP. Link E is also a regional backbone for that ISP. Relays A, B, C and D are PE routers, and Links A, B, F and G are actually link aggregators with individual layer 2 circuits to each customer--for example, the relays might be DSLAMs or cable head-end systems. At each customer site we assume there is a single CPE device attached to the link.

We further assume that links A, B, F and G are each addressed by a single prefix, although it would be equally valid for each CPE device to be numbered on a separate prefix.

In a real-world deployment, there would likely be many more than two PE routers connected to each regional backbone; we have kept the number small for simplicity.

In the example presented in Figure 4, the goal is to configure all the devices within a region with server addresses local to that region, so that service traffic does not have to be routed between regions unnecessarily.

```
   {
       "prefixes": {
           "2001:db8::/40": {
               "on-link": ["A"]
           },
           "2001:db8:100::/40": {
               "on-link": ["B"]
           },
           "2001:db8:200::/40": {
               "on-link": ["F"]
           },
           "2001:db8:300::/40": {
               "on-link": ["G"]
           }
       },
       "links": {
           "A": {"region": "omashu"},
           "B": {"region": "omashu"},
           "F": {"region": "gaoling"},
           "G": {"region": "gaoling"}
       },
      "regions": {
          "omashu": {
              "options": {
                  "sip-servers": ["sip.omashu.example.org"],
                  "dns-servers": ["dns1.omashu.example.org",
                                  "dns2.omashu.example.org"]
              }
          },
          "gaoling": {
              "options": {
                  "sip-servers": ["sip.gaoling.example.org"],
                  "dns-servers": ["dns1.gaoling.example.org",
                                  "dns2.gaoling.example.org"]
              }
           }
       }
   }
```

Figure 4: Regional Configuration Example

In this example, when a request comes in to the DHCPv6 server with a link-identifying IP address in the 2001:db8::/40 prefix, it is identified as being on link A. The DHCPv6 server then looks on the list of links to see what region the client is in. Link A is identified as being in omashu. The DHCPv6 server then looks up omashu in the set of regions, and discovers a list of region-specific options.

The DHCPv6 server then resolves the domain names listed in the options and sends a sip-server option containing the IP addresses that the resolver returned for sip.omashu.example.org, and a dns-server option containing the IP addresses returned by the resolver for dns1.omashu.example.org and dns2.omashu.example.org. Depending on the server capability and configuration, it may cache resolved responses for specific period of time, repeat queries every time or even keep the response until reconfiguration or shutdown. For more detailed discussion see Section 7 of [RFC7227].

Similarly, if the DHCPv6 server receives a request from a DHCPv6 client where the link-identifying IP address is contained by the prefix 2001:db8:300::/40, then the DHCPv6 server identifies the client as being connected to link G. The DHCPv6 server then identifies link G as being in the gaoling region, and returns the sip-servers and dns-servers options specific to that region.

As with the previous example, the exact configuration syntax and structure shown above does not precisely match what existing DHCPv6 servers do, but the behavior illustrated in this example can be accomplished with most existing modern DHCPv6 servers.

8. Multiple subnets on the same link

There are scenarios where there is more than one subnet from the same protocol family (i.e. two or more IPv4 subnets or two or more IPv6 subnets) configured on the same link. Such a configuration is often referred to as ’shared subnets’ in Unix environments or ’multinet’ in Microsoft terminology.

The most frequently mentioned use case is a network renumbering where some services are migrated to the new addressing scheme, but some aren’t yet.

Second example is expanding the allocation space. In DHCPv4 and for DHCPv6 Prefix Delegation, there could be cases where multiple subnets are needed, because a single subnet may be too small to accommodate the client population.

The third use case covers allocating addresses (or delegation prefixes) that are not the same as topological information. For example, the link-address is on prefix X and the addresses to be assigned are on prefix Y. This could be based on differentiating information (i.e., whether device is CPE or CM in DOCSIS) or just because the link-address/giaddr is different from the actual allocation space.

The fourth use case is a cable network, where cable modems and the devices connected behind them are connected to the same layer 2 link. However, operators want the cable modems and user devices to get addresses from distinct address spaces, so users couldn’t easily access their modems management interfaces.

To support such a configuration, additional differentiating information is required. Many DHCP server implementations offer a feature that is typically called client classification. The server segregates incoming packets into one or more classes based on certain packet characteristics, e.g. presence or value of certain options or even a match between existing options. Servers require additional information to handle such configuration, as they cannot use the topographical property of the relay addresses alone to properly choose a subnet. Exact details of such operation is not part of the DHCPv4 or DHCPv6 protocols and is implementation dependent.

9. Acknowledgments

Thanks to Dave Thaler for suggesting that even though "everybody knows" how DHCP servers are deployed in the real world, it might be worthwhile to have an IETF document that explains what everybody knows, because in reality not everybody is an expert in how DHCP servers are administered. Thanks to Andre Kostur, Carsten Strotmann, Simon Perreault, Jinmei Tatuya, Suresh Krishnan, Qi Sun, Jean-Francois Tremblay, Marcin Siodelski, Bernie Volz and Yaron Sheffer for their reviews, comments and feedback.

10. Security Considerations

This document explains existing practice with respect to the use of Dynamic Host Configuration Protocol [RFC2131] and Dynamic Host Configuration Protocol Version 6 [RFC3315]. The security considerations for these protocols are described in their specifications and in related documents that extend these protocols.

The mechanisms described in this document could possibly be exploited by an attacker to misrepresent its point of attachment in the network. This would cause the server to assign addresses, prefixes and other configuration options, which can be considered a leak of information. In particular, this could be used a preliminary stage of an attack, when the DHCP server leaks information about available services in parts of the network the attacker does not have access to.

There are several ways how such an attack can be prevented. First, it seems to be a common practice to filter out DHCP traffic coming in from outside of the network and one that is directed to clients outside of the network. Second, the DHCP servers can be configured to not respond to traffic that is coming from unknown (i.e. those subnets the server is not configured to serve) subnets. Third, some relays provide the ability to reject messages that do not fit expected characteristics. For example CMTS (Cable Modem Termination System) acting as a DHCP relay detects if the MAC address specified in chaddr in incoming DHCP messages matches the MAC address of the cable modem it came from and can alter its behavior accordingly. Also, relay agents and servers that are connected to clients directly can reject traffic that looks as if it has passed a relay (this could indicate the client is attempting to spoof a relay, possibly to inject forged relay options).

There are a number of general DHCP recommendations that should be considered in all DHCP deployments. While not strictly related to the mechanisms described in this document, they may be useful in certain deployment scenarios. [RFC7819] and [RFC7824] provide an analysis of privacy problems in DHCPv4 and DHCPv6, respectively. If those are of concern, [RFC7844] offers mitigation steps.

Current DHCPv4 and DHCPv6 standards lack strong cryptographic protection. There is an ongoing effort in DHC working group to address this. [I-D.ietf-dhc-sedhcpv6] attempts to provide mechanism for strong authentication and encryption between DHCPv6 clients and servers. [I-D.volz-dhc-relay-server-security] attempts to improve security of exchanges between DHCP relay agents and servers.

Another possible attack vector is to set up a rogue DHCP server and provide clients with false information, either as a denial of service or to execute man in the middle type of attack. This can be mitigated by deplyoing DHCPv6-shield [RFC7610].

Finally, there is an ongoing effort to update DHCPv6 specification, that is currently 13 years old. Sections 23 (Security Considerations) and 24 (Privacy Considerations) of [I-D.ietf-dhc-rfc3315bis] contain more recent analysis of the security and privacy considerations.

11. IANA Considerations

The IANA is hereby absolved of any requirement to take any action in relation to this document.

12. References

12.1. Normative References

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <http://www.rfc-editor.org/info/rfc2131>.

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <http://www.rfc-editor.org/info/rfc3315>.

12.2. Informative References

[I-D.ietf-dhc-rfc3315bis] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6) bis", draft-ietf-dhc-rfc3315bis-05 (work in progress), June 2016.

[I-D.ietf-dhc-sedhcpv6] Jiang, S., Li, L., Cui, Y., Jinmei, T., Lemon, T., and D. Zhang, "Secure DHCPv6", draft-ietf-dhc-sedhcpv6-12 (work in progress), April 2016.

[I-D.volz-dhc-relay-server-security] Volz, B. and Y. Pal, "Security of Messages Exchanged Between Servers and Relay Agents", draft-volz-dhc-relay-server-security-01 (work in progress), June 2016.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <http://www.rfc-editor.org/info/rfc1034>.

[RFC3011] Waters, G., "The IPv4 Subnet Selection Option for DHCP", RFC 3011, DOI 10.17487/RFC3011, November 2000, <http://www.rfc-editor.org/info/rfc3011>.

[RFC3046] Patrick, M., "DHCP Relay Agent Information Option", RFC 3046, DOI 10.17487/RFC3046, January 2001, <http://www.rfc-editor.org/info/rfc3046>.

[RFC3527] Kinnear, K., Stapp, M., Johnson, R., and J. Kumarasamy, "Link Selection sub-option for the Relay Agent Information Option for DHCPv4", RFC 3527, DOI 10.17487/RFC3527, April 2003, <http://www.rfc-editor.org/info/rfc3527>.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <http://www.rfc-editor.org/info/rfc4193>.

[RFC6221] Miles, D., Ed., Ooghe, S., Dec, W., Krishnan, S., and A. Kavanagh, "Lightweight DHCPv6 Relay Agent", RFC 6221, DOI 10.17487/RFC6221, May 2011, <http://www.rfc-editor.org/info/rfc6221>.

[RFC6607] Kinnear, K., Johnson, R., and M. Stapp, "Virtual Subnet Selection Options for DHCPv4 and DHCPv6", RFC 6607, DOI 10.17487/RFC6607, April 2012, <http://www.rfc-editor.org/info/rfc6607>.

[RFC6977] Boucadair, M. and X. Pougnard, "Triggering DHCPv6 Reconfiguration from Relay Agents", RFC 6977, DOI 10.17487/RFC6977, July 2013, <http://www.rfc-editor.org/info/rfc6977>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7227] Hankins, D., Mrugalski, T., Siodelski, M., Jiang, S., and S. Krishnan, "Guidelines for Creating New DHCPv6 Options", BCP 187, RFC 7227, DOI 10.17487/RFC7227, May 2014, <http://www.rfc-editor.org/info/rfc7227>.

[RFC7610] Gont, F., Liu, W., and G. Van de Velde, "DHCPv6-Shield: Protecting against Rogue DHCPv6 Servers", BCP 199, RFC 7610, DOI 10.17487/RFC7610, August 2015, <http://www.rfc-editor.org/info/rfc7610>.

[RFC7819] Jiang, S., Krishnan, S., and T. Mrugalski, "Privacy Considerations for DHCP", RFC 7819, DOI 10.17487/RFC7819, April 2016, <http://www.rfc-editor.org/info/rfc7819>.

[RFC7824] Krishnan, S., Mrugalski, T., and S. Jiang, "Privacy Considerations for DHCPv6", RFC 7824, DOI 10.17487/RFC7824, May 2016, <http://www.rfc-editor.org/info/rfc7824>.

[RFC7844] Huitema, C., Mrugalski, T., and S. Krishnan, "Anonymity Profiles for DHCP Clients", RFC 7844, DOI 10.17487/RFC7844, May 2016, <http://www.rfc-editor.org/info/rfc7844>.

Authors’ Addresses

Ted Lemon
Nominum, Inc.
2000 Seaport Blvd
Redwood City, CA 94063
USA

Phone: +1-650-381-6000
Email: Ted.Lemon@nominum.com

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

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

DHC WG S. Jiang
Internet-Draft B. Liu
Intended status: Standards Track Huawei Technologies Co., Ltd
Expires: August 18, 2014 February 14, 2014

Stateless Reconfiguration in Dynamic Host Configuration Protocol for IPv6 (DHCPv6)
draft-jiang-dhc-stateless-reconfiguration-01

Abstract

This document defines a mechanism to propagate reconfigure messages towards stateless configured DHCPv6 clients.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

[RFC3736] defines a stateless configuration procedure using DHCPv6. With it, the network configure information, such as the addresses of DNS recursive name servers, can be propagated to nodes, which have obtained their IPv6 addresses through some other mechanism. The basic scenario is that a newly online client initiates an information request to DHCPv6 server, then the server responses with requested configuration information. This mechanism is called the Stateless DHCPv6 services, because DHCPv6 servers do not maintain any dynamic state for individual clients, including the unicast addresses of clients.

However, the specification of stateless DHCPv6 service lacks a mechanism to inform these configured clients if some configuration information is changed. Transplanting Reconfigure message of [RFC3315] into stateless DHCPv6 services does not solve the issue, because in stateful DHCPv6, servers send Reconfigure messages to clients using their unicast addresses.

The lifetime option for DHCPv6 [RFC4242] assigns a lifetime to configuration information obtained through DHCPv6. At the expiration of the lifetime, the host contacts the DHCPv6 server to obtain updated configuration information. This lifetime gives the network administrator another mechanism to configure hosts with new configuration by controlling the time at which the host refreshes the list. However, such mechanism is not flexible enough: one aspect is the minimum of refresh time is 10 minutes, which is so long that it might not be suitable for unplanned configuration changes; the other aspect is, in order to update the configuration quickly, the short time setting would cause un-necessary refresh all the time.

This document defines a mechanism to propagate a newly defined Stateless-Reconfigure message towards stateless configured DHCPv6 clients. It requests a mechanism for the DHCPv6 server to be aware of all relay agent destinations.

{Question to WG No.1} There are three potential mechanisms to create relay agent destinations on the DHCPv6 server:

a) Static configuration

Network administrators manually configure static unicast addresses of all relay agents on the DHCPv6 server.

Pros: no need to update any protocol/function implementation in relays; allows fast deployment in current network.

Cons: cost significant human management burden; error-prone, mistakenly configuring the relay addresses or leaving out some relays are expected.

b) Define a new ALL_RELAY_AGENT multicast address

The DHCPv6 server could send the stateless reconfiguration messages directly to the new multicast address.

Pros: a solid coverage of all relays.

Cons: network administrators need to maintain an all-relay-agent multicast group; all relays and DHCPv6 servers need to be updated to know the new multicast address.

c) DHCPv6 server dynamic learning

the DHCPv6 server dynamically records unicast addresses of all relay agents from client Information-request messages and maintains the relay addresses list. A keepalive mechanism is needed between relay agents and servers to track the availability of the list entries.

Pros: automatic processing without human intervene.

Cons: requires more function update to the DHCPv6 server; the keepalive mechanism requires more function/protocol burden to the whole DHCP system.

[Editor Notes] the current form of this document is based on only the first mechanism of above three. If the WG decided to change or include other mechanism, the document would be updated accordingly.

The document newly defines a new link-scope well-known all-client multicast address and a new DHCPv6 message type for stateless reconfiguration. Correspondent server behavior, agent behavior and client behavior are specificed in details.

The design of new DHCPv6 elements and precedures obey the recommendations and guidance of [I-D.ietf-dhc-option-guidelines].

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] when they appear in ALL CAPS. When these words are not in ALL CAPS (such as "should" or "Should"), they have their usual English meanings, and are not to be interpreted as [RFC2119] key words.

3. Stateless Reconfiguration Use Cases

This section described scenarisos where stateless reconfiguration are expected.

- Configuration error

Configuration errors, eithor caused by human or program, are hard to be immune in networks. Especially, human errors is identified as one of the top reasons of network failure. In stateless DHCPv6, if the administrators/program accidentally mis-configure the parameters (e.g. DNS), then significant network failure would be expected. Current protocols just lack the ability to eliminate the configuration errors when such accident happens. The hosts configured with wrong parameters can only wait until the wrong parameters lifetime expired then to refresh them. This would not be acceptable especially when the lifetime was long. The stateless reconfiguration mechanism could be highly expected in this scenario.

- Emergent event

The network needs to initially update the already configured parameters within a short period due to some emergent events; and waiting the clients to refresh the parameters according to the lifetime is just un-acceptable. These scenarios would also require statless reconfiguration.

4. New DHCPv6 Specifications

This section define new DHCPv6 elements requested by the stateless configuration mechanism, including multicast address constant, and message type.

4.1. Multicast Address

ALL_CLIENT_MULTICAST_ADDRESS (FF02::xxxx, TBD1) A link-scoped multicast address used by a DHCPv6 server or relay agent to communicate with neighboring (i.e., on-link) clients. All clients are members of this multicast group

4.2. Stateless Reconfigure Message

A new Stateless-Reconfigure message, which is mainly based on server to clients advertise model, is defined in order to distinguish from the existing Reconfigure message, which is mainly based on server/ client one-to-one model.

[Editor Notes] According to the results of Qeston 2 and Question 4 (in Section 5 & 7 below), there might be two new messages needed. Current document uses the alternative of one new message.

STATELESS-RECONFIGURE-TRIGGER Message type value is TBD2. It follows the message format specification, defined in Section 6 of [RFC3315]. A server sends a Stateless-Reconfigure message to a client to inform the client that the server has new or updated configuration parameters, and that the client is to initiate an Information-Request transaction with the server in order to receive the updated information.

5. Stateless Reconfiguration Procedure

{Question to WG No.2} There could be two kind of stateless DHCPv6 reconfiguration modes as the following, which one is proper? Or we should support both?

- Trigger mode

The server sends out a multicast Stateless-Reconfiguration message to ALL_CLIENT_MULTICAST_ADDRESS. As response, every client is requested to initiate an Information-Request message back to the server. The server can then inform the changed configuration information to clients.

This mode is similar with stateful DHCPv6 reconfiguration and also provide the potential possibility that the server response to information-request differently according to various user policies.

- Push mode

The server sends out a multicast Stateless-Reconfiguration message to ALL_CLIENT_MULTICAST_ADDRESS to directly advertise new configuration to the clients. The clients then update the parameters accordingly.

Trigger mode requires every client to initial individual request to servers. This is reasonable for the stateful information that need to be maintaind and tracked in the servers, e.g. clients’ IP addresses. But for the stateless information shared among clients (such as DNS), it might not necessary. Some resource constrained networks (e.g. a 802.15.4e/g based mesh network) might have efficency problem with the trigger mode. These scenarios might significantly benifit from the push mode stateless reconfiguration mechanism. However, push mode seems breaking the triditional behavior model of DHCP. Whether it is a good break needs further discussion.

[Editor Notes] the current form of this document is based on triggering client information-request model, which complies the traditional behavior model of DHCPv6. If the WG chooses to directly advertise new configuration, the document would be updated accordingly.

5.1. Server Behavior

When the network configuration information on a stateless DHCPv6 server changes, the server creates and transmit a new Stateless-Reconfigure message towards all clients following the below steps:

- o The server sets the "msg-type" field to STATELESS-RECONFIGURE. The server sets the transaction-id field to 0. The server MUST include a Server Identifier option containing its DUID in the Reconfigure message.

- o The server MAY include an Option Request option to inform the client of what information has been changed or new information that has been added.

- o The server MUST NOT include a Reconfigure Message option (defined in section 22.19 of [RFC3315]), which is mandated in Reconfigure message to indicate the client to respond a Renew or an Information-Request message. It is because there is only one possible response on the client follow a Stateless-Reconfigure message - an Information-request message.

- The server MUST NOT include any other options in the Reconfigure except as specifically allowed in the definition of individual options.

- The server sends Stateless-Reconfigure message to its direct local link using ALL_CLIENT_MULTICAST_ADDRESS.

- Simultaneously, the server uses a Relay-Reply message (as described in Section 20.3 of [RFC3315]) to send the Stateless-Reconfigure message to all relay agents in its static relay-agent-destination record using the unicast address of these relay agents. The peer-address of the Relay-Reply message MUST be filled by Relay-Reply message ALL_CLIENT_MULTICAST_ADDRESS.

Notes: since there is no previous Relay-Forward message that went through multiple relay agents and the server has to send the Relay-Reply message through the return same path, the server should be able to send the Relay-Reply message to the relay agent that direct connects with clients. Consequently, the Relay-Reply message SHOULD NOT contain another Relay-Reply message.

The below is an example of a typical Relay-Reply message that contains a Stateless-Reconfigure message:

```
msg-type: RELAY-REPLY
hop-count: 0
link-address: 0
peer-address: ALL_CLIENT_MULTICAST_ADDRESS
Relay Message option: <Stateless-Reconfigure message>
```

Servers MUST discard any received Stateless-Reconfigure messages.

5.2. Relay Agent Behavior

The relay agent extracts the Stateless-Reconfigure message from the Relay Message option and relays it to all clients. If the relay agent is attached to multiple links, it MUST broadcast the Stateless-Reconfigure message on every links. This behavior is compliance with normal behavior of relaying a Relay-reply message, defined in Section 20.2 of [RFC3315].

Relay agents MUST discard any received Stateless-Reconfigure messages. By design, relay agents do not process any directly received Stateless-Reconfigure messages.

The result is that the relay agent sends out a Stateless-Reconfigure message towards all client on the local link using ALL_CLIENT_MULTICAST_ADDRESS.

5.3. Client Behavior

Clients MUST discard any Stateless-Reconfigure messages that meets any of the following conditions:

- the message was not multicast to the client using ALL_CLIENT_MULTICAST_ADDRESS.

- the message does not include a Server Identifier option.

- the message contains a Reconfigure Message Option, defined in Section 22.19 of [RFC3315].

Upon receipt of a valid Stateless-Reconfigure message, after a random delay time, the client responds with an Information-request message. The random delay time is designed to avoid congested Information-request on the server. While the transaction is in progress, the client silently discards any Stateless-Reconfigure messages it receives.

{Question to WG No.3} Should we define a maximum time of random delay time? If yes, should it come from server by a new option?

6. Security Considerations

Malicious server sends Stateless Reconfigure message to cause all clients response. There is the risk of denial of service attacks against DHCP clients and server. {Current auth mechanism cannot work in this broadcast model, server public key model maybe work.}

Since the clients response to Information-Request using the standard mechanism, defined in [RFC3315], the chance that receive wrong configuration information from malicious attackers does not raise.

7. IANA Considerations

Per this document, IANA has assigned one new well-known Multicast Address in the "IPv6 Multicast Address Space Registry" registry (currently located at http://www.iana.org/assignments/ipv6-multicast-addresses) for the following attribute:

ALL_CLIENT_MULTICAST_ADDRESS: (FF02::xxxx, TBD1).

Per this document, IANA has assigned one new DHCPv6 message type in the "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)" registry (currently located at http://www.iana.org/assignments/dhcpv6-parameters) for the following attribute:

STATELESS-RECONFIGURE Message Type, TBD2.

{Question to WG No.4} As raised in Question 2, if we support both Trigger mode and Push mode, then there should be two kind of corresponding messages. We could use two message types to distinguish them; or use a flag in one message type. Which is better?

8. Acknowledgements

The authors would like to thanks the valuable comments made by Suresh Krishnan, Ted Lemon, Bernie Voltz and other members of DHC WG.

This document was produced using the xml2rfc tool [RFC2629].

9. References

9.1. Normative References

[I-D.ietf-dhc-option-guidelines] Hankins, D., Mrugalski, T., Siodelski, M., Jiang, S., and S. Krishnan, "Guidelines for Creating New DHCPv6 Options", draft-ietf-dhc-option-guidelines-17 (work in progress), January 2014.

[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.

[RFC3736] Droms, R., "Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6", RFC 3736, April 2004.

9.2. Informative References

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[RFC4242] Venaas, S., Chown, T., and B. Volz, "Information Refresh Time Option for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 4242, November 2005.

Authors’ Addresses

Sheng Jiang
Huawei Technologies Co., Ltd
Q14, Huawei Campus, No.156 Beiqing Road
Hai-Dian District, Beijing, 100095
P.R. China

Email: jiangsheng@huawei.com

Bing Liu
Huawei Technologies Co., Ltd
Q14, Huawei Campus, No.156 Beiqing Road
Hai-Dian District, Beijing, 100095
P.R. China

Email: leo.liubing@huawei.com

HOMENET
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2015

D. Migault (Ed)
Orange
W. Cloetens
SoftAtHome
C. Griffiths
Dyn
R. Weber
Nominum

July 2, 2014

DHCP Options for Homenet Naming Architecture
draft-mglt-homenet-naming-architecture-dhc-options-02.txt

Abstract

CPEs are usually constraint devices with reduced network and CPU capacities. As such, a CPE hosting on the Internet the authoritative naming service for its home network may become vulnerable to resource exhaustion attacks. One way to avoid exposing CPE is to outsource the authoritative service to a third party. This third party can be the ISP or any other independent third party.

Outsourcing the authoritative naming service to a third party requires setting up an architecture which may be unappropriated for most end users. To leverage this issue, this document proposes DHCP Options so any agnostic CPE can automatically proceed to the appropriated configuration and outsource the authoritative naming service for the home network. This document shows that in most cases, these DHCP Options make outsourcing to a third party (be it the ISP or any ISP independent service provider) transparent for the end user.

Status of This Memo

This Internet-Draft will expire on January 3, 2015.

Copyright Notice

Table of Contents

1. Requirements notation

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 [RFC2119].

2. Terminology

- Customer Premises Equipment: (CPE) is the router providing connectivity to the home network. It is configured and managed by the end user. In this document, the CPE might also hosts services such as DHCPv6. This device might be provided by the ISP.

- Public Key: designates a public Key generated by the CPE. This key is used as an authentication credential for the CPE.

- Registered Homenet Domain: is the Domain Name associated to the home network.

- DNS Homenet Zone: is the DNS zone associated to the home network. This zone is set by the CPE and essentially contains the bindings between names and IP addresses of the nodes of the home network. In this document, the CPE does neither perform any DNSSEC management operations such as zone signing nor provide an authoritative service for the zone. Both are delegated to the Public Authoritative Server. The CPE synchronizes the DNS Homenet Zone with the Public Authoritative Server via a hidden master / slave architecture. The Public Authoritative Server might use specific servers for the synchronization of the DNS Homenet Zone: the Public Authoritative Name Server Set.

- DNS Homenet Zone Template: The template used as a basis to generate the DNS Homenet Zone.

- DNS Template Server: The DNS server that hosts the DNS Homenet Zone Template.

- DNS Homenet Reverse Zone: The reverse zone file associated to the DNS Homenet Zone.

- Public Authoritative Master(s): are the visible name server hosting the DNS Homenet Zone. End users' resolutions for the Homenet Domain are sent to this server, and this server is a master for the zone.

- Public Authoritative Name Server Set: is the server the CPE synchronizes the DNS Homenet Zone. It is configured as a slave and the CPE acts as master. The CPE sends information so the DNSSEC zone can be set and served.

- Reverse Public Authoritative Master(s): are the visible name server hosting the DNS Homenet Reverse Zone. End users' resolutions for the Homenet Domain are sent to this server, and this server is a master for the zone.

- Reverse Public Authoritative Name Server Set: is the server the CPE synchronizes the DNS Homenet Reverse Zone. It is configured as a slave and the CPE acts as master. The CPE sends information so the DNSSEC zone can be set and served.

3. Introduction

CPEs are usually constraint devices with reduced network and CPU capacities. As such, a CPE hosting on the Internet the authoritative naming service for its home network may become vulnerable to resource exhaustion attacks. One way to avoid exposing CPE is to outsource the authoritative service to a third party. This third party can be the ISP or any other independent third party.

Outsourcing the authoritative naming service to a third party requires setting up an architecture which may be unappropriated for most end users. To leverage this issue, this document proposes DHCP Options so any agnostic CPE can automatically proceed to the appropriated configuration and outsource the authoritative naming service for the home network. This document shows that in most cases, these DHCP Options make outsourcing to a third party (be it the ISP or any ISP independent service provider) transparent for the end user.

When the CPE is plugged, the DHCP Options described in the document enable the CPE:

- 1. To build the DNS Homenet Zone: Building the DNS Homenet Zone requires filling the zone with appropriated bindings likes name / IP addresses of the different devices in the home networks. Such information can be provided for example by the DHCP Server hosted on the CPE. On the other hand, it also requires configuration parameters like the name of the Registered Domain Name associated to the home network or the Public Authoritative Master(s) the DNS Homenet Zone is outsourced to. These configuration parameters are stored in the DNS Homenet Zone Template. This document describes the DHCP Zone Template Option. This option carries a DNS Homenet Zone Template FQDN. In order to retrieve the DNS Homenet Zone Template, the CPE sends a query of type AXFR [RFC1034] [RFC5936]for the DNS Homenet Zone Template FQDN.

- 2. To upload the DNS(SEC) Homenet Zone to the appropriated server: This server is designated as the Public Authoritative Name Server Set. It is in charge of publishing the DNS(SEC) Homenet Zone on the Public Authoritative Master(s). This document describes the DHCP Public Authoritative Name Server Set Option that provides the FQDN of the appropriated server. Note that, in the document we do not consider whether the DNS(SEC) Homenet Zone is signed or not and if signed who signs it. Such questions are out of the scope of the current document.

- 3. To upload the DNS Homenet Reverse Zone to the appropriated server: This server is designated as the Reverse Public Authoritative Name Server Set. It is in charge of publishing the DNS Homenet Reverse Zone on the Reverse Public Authoritative Master(s). This document describes the DHCP Reverse Public Authoritative Name Server Set Option that provides the FQDN of the appropriated server. Similarly to item 2., we do not consider in this document if the DNS Homenet Reverse Zone is signed or not, and if signed who signs it.

- 4. To provide authentication credential (a public key) to the DHCP Server: Information stored in the DNS Homenet Zone Template, the DNS(SEC) Homenet Zone and DNS Homenet Reverse Zone belongs to the CPE, and only the CPE should be able to update or upload these zones. To authenticate the CPE, this document defines the DHCP Public Key Option. This option is sent by the CPE to the DHCP Server and provides the Public Key the CPE uses to authenticate itself. The DHCP Server is then responsible to provide the Public Key to the various DNS servers.

As a result, the DHCP Options described in this document enable an agnostic CPE to outsource its naming infrastructure without any configuration from the end user. The main reason no configuration is required by the end user is that there are privilege links first between the CPE and the DHCP Server and then between the DHCP Server and the various DNS servers (DNS Homenet Zone Server, the Reverse Public Authoritative Name Server Set, Public Authoritative Name Server Set). This enables the CPE to send its authentication credentials (a Public Key) to the DHCP Server that in turn forward it to the various DNS servers. With the authentication credential on the DNS servers set, the CPE is able to update the various zones in a secure way.

If the DHCP Server cannot provide the public key to one of these servers (most likely the Public Authoritative Name Server Set) and the CPE needs to interact with the server, then, the end user is expected to provide it manually or using other mechanisms. Such mechanisms are outside the scope of this document. In that case, the authentication credentials need to be provided every time the key is modified. Appendix A provides more details on how different scenarios impact the end users.

4. Protocol Overview

This section illustrates how a CPE configures its naming infrastructure to outsource its authoritative naming service. All configurations and settings are performed using DHCP Options. In this section, for the sake of simplicity, we consider that the DHCP Server is able to communicate to the various DNS servers and provide them them the public key associated to the CPE. Once each server got the credentials, the CPE can proceed to updates in a authenticated and secure way.

This scenario has been chosen as it is believed to be the most popular scenario. This document does not ignore that scenarios where the DHCP Server does not have privilege relations with the Public Authoritative Name Server Set must be considered. These cases are discussed latter in Appendix A. Such scenario does not necessarily require configuration for the end user and can also be Zero Config.

The scenario is represented in Figure 1.

- 1: The CPE provides its Public Key to the DHCP Server using a DHCP Public Key Option (OPTION_PUBLIC_KEY) and sends a DHCP Option Request Option (ORO) for the DHCP Zone Template Option (OPTION_DNS_ZONE_TEMPLATE), the DHCP Public Authoritative Name Server Set Option (OPTION_NAME_SERVER_SET) and the DHCP Reverse

Public Authoritative Name Server Set Option (OPTION_REVERSE_NAME_SERVER_SET).

- 2: The DHCP Server makes the Public Key available to the DNS servers, so the CPE can secure its DNS transactions. Note that the Public Key alone is not sufficient to perform the authentication and the key should be, for example, associated with an identifier, or the concerned domain name. How the binding is performed is out of scope of the document. It can be a centralized database or various bindings may be sent to the different servers. Figure 1 represents the specific case were the DHCP Server forwards the set (Public Key, Zone Template FQDN) to the DNS Template Server, the set (Public Key, IPv6 subnet) to the Reverse Public Authoritative Name Server Set and the set (Public Key, Registered Homenet Domain) to the Public Authoritative Name Server Set.

- 3.: The DHCP Server responds to the CPE with the requested DHCP Options, i.e. the DHCP Public Key Option (OPTION_PUBLIC_KEY), DHCP Zone Template Option OPTION_DNS_ZONE_TEMPLATE, DHCP Public Authoritative Name Server Set Option (OPTION_NAME_SERVER_SET), DHCP Reverse Public Authoritative Name Server Set Option (OPTION_REVERSE_NAME_SERVER_SET).

- 4.: Upon receiving the DHCP Zone Template Option (OPTION_DNS_ZONE_TEMPLATE), the CPE performs an AXFR DNS query for the Zone Template FQDN. The exchange is secured according to the security protocols defined in the Security field of the DHCP option. Once the CPE has retrieved the DNS Zone Template, the CPE can build the DNS Homenet Zone and the DNS Homenet Reverse Zone. Eventually the CPE signs these zones.

- 5.: Once the DNS(SEC) Homenet Reverse Zone has been set, the CPE uploads the zone to the Reverse Public Authoritative Name Server Set. The DHCP Reverse Public Authoritative Name Server Set Option (OPTION_REVERSE_NAME_SERVER_SET) provides the Reverse Public Authoritative Name Server Set FQDN as well as the upload method, and the security protocol to secure the upload.

- 6.: Once the DNS(SEC) Homenet Zone has been set, the CPE uploads the zone to the Public Authoritative Name Server Set. The DHCP Public Authoritative Name Server Set Option (OPTION_NAME_SERVER_SET) provides the Public Authoritative Name Server Set FQDN as well as the upload method and the security protocol to secure the upload.

```
      +----------------------+
      |     DHCP Server      |
      +----------------------+
            ^   ^         ^
            |   |         |2.
          1.|   |3.       |
            v   v         |
           +------+       |    +--------------------------------+
           |      |  4.   +--->|  DNS Template Server           |
           |      |<---------->|                                |
           |      |       |    +--------------------------------+
           | CPE  |       |
           |      |       |    +--------------------------------+
           |      |  5.   +--->|  Reverse Public Authoritative  |
           |      |<---------->|  Name Server Set               |
           |      |       |    +--------------------------------+
    -------|      |-------|--------------------------------------
           |      |       |    +--------------------------------+
           +------+       +--->|  Public Authoritative          |
                               |  Name Server Set               |
                               +--------------------------------+
```

Figure 1: Protocol Overview

5. Securing the exchanges

Multiple protocols like IPsec [RFC4301] or TLS / DTLS [RFC5246] / [RFC6347] may be used to secure DNS transactions between the CPE and the DNS servers. This document restricts the scope of security protocols to those that have been designed specifically for DNS. This includes DNSSEC [RFC4033], [RFC4034], [RFC4035] that authenticates and provides integrity protection of DNS data, TSIG [RFC2845], [RFC2930] that use a shared secret to secure a transaction between two end points and SIG(0) [RFC2931] authenticates the DNS packet exchanged.

The key issue with TSIG is that a shared secret must be negotiated between the CPE and the server. On the other end, TSIG performs symmetric cryptography which is light in comparison with asymmetric cryptography used by SIG(0). As a result, over large zone transfer, TSIG may be preferred to SIG(0).

This document does not provides means to distribute shared secret for example using a specific DHCP Option. The only assumption made is that the CPE generates or is assigned a public key.

As a result, when the document specifies the transaction is secured with TSIG, it means that either the CPE and the DNS Server have been manually configured with a shared secret, or the shared secret has been negotiated using TKEY [RFC2930], and the TKEY exchanged are secured with SIG(0).

Exchange with the DNS Template Server to retrieve the DNS Homenet Zone Template may be protected by SIG(0), TSIG or DNSSEC. When DNSSEC is used, it means the DNS Template Server only provides integrity protection, and does not necessarily prevents someone else to query the DNS Homenet Zone Template. In addition, DNSSEC is only a way to protect the communication of AXFR queries, in other words, DNSSEC cannot be used to secure updates. If DNSSEC is used to provide integrity protection for the AXFR response, the CPE should proceed to the DNSSEC signature checks. If signature check fails, it MUST reject the response. If the signature check succeeds, the CPE removes all DNSSEC related RRsets (DNSKEY, RRSIG, NSEC* ...) before building the DNS Homenet Zone. In fact, these DNSSEC related fields as associated to the DNS Homenet Zone Template and not the DNS Homenet Zone.

Any update exchange should use SIG(0) or TSIG to authenticate the exchange.

6. DNS Zones Update Mechanisms

6.1. Data subjected to update

The CPE is likely to update various DNS contents:

- DNS Homenet Zone Template: may be updated by the CPE if the configuration of the zone may be changed. This can include additional Public Authoritative Master(s), a different Registered Homenet Domain as the one initially proposed, or a redirection to another domain.

- DNS Homenet Reverse Zone: may be updated every time a new device is connected or dis-connected.

- DNS Homenet Zone: may be updated every time a new device is connected, dis-connected.

6.2. Master / Slave Synchronization versus DNS Update

As updates only concern DNS zones, this document only considers DNS update mechanisms such as DNS update [RFC2136] [RFC3007] or a master / slave synchronization.

The DNS Homenet Zone Template can only be updated with DNS update. The reason is that the DNS Homenet Zone Template contains static configuration data that is not expected to evolve over time.

The DNS Homenet Reverse Zone and the DNS Homenet Zone can be updated either with DNS update or using a master / slave synchronization. As these zones may be large, with frequent updates, we recommend to use the master / slave architecture as described in [I-D.mglt-homenet-front-end-naming-delegation]. The master / slave mechanism is preferred as it better scales and avoids DoS attacks: First the master notifies the slave the zone must be updated, and leaves the slave to proceed to the update when possible. Then, the NOTIFY message sent by the master is a small packet that is less likely to load the slave. At last, the AXFR query performed by the slave is a small packet sent over TCP (section 4.2 [RFC5936]) which makes unlikely the slave to perform reflection attacks with a forged NOTIFY. On the other hand, DNS updates can use UDP, packets require more processing then a NOTIFY, and they do not provide the server the opportunity to post-pone the update.

6.3. Setting Master / Slave Synchronization

The master / slave architecture is described in [I-D.mglt-homenet-front-end-naming-delegation]. The CPE is configured as a master whereas the DNS Server is configured as a slave. The DNS Server represents the Public Authoritative Name Server Set or the Reverse Public Authoritative Name Server Set.

When the CPE is plugged its IP address may be unknown to the slave. The section details how the CPE or master communicate the necessary information to set up the slave.

In order to set the master / slave configuration, both master and slaves must agree on 1) the zone to be synchronized, 2) the IP address used by both master and slave. In this document we assume that synchronization is performed on both side on port 53.

[QUESTION Do we have to consider different port of port 53 is fine. I guess it is fine.]

6.4. Master / Slave Synchronization: CPE / Public Authoritative Name Server Set

The CPE knows the zone to be synchronized by reading the Registered Homenet Domain in the DNS Homenet Zone Template provided by the DHCP Zone Template Option (OPTION_DNS_ZONE_TEMPLATE). The IP address of the slave is provided by the DHCP Public Authoritative Name Server Set Option (OPTION_NAME_SERVER_SET).

The Public Authoritative Name Server Set has been configured with the Registered Homenet Domain and the Public Key that identifies the CPE. The only thing missing is the IP address of the CPE. This IP address is provided by the CPE by sending a NOTIFY [RFC1996].

When the CPE has built its DNS Homenet Zone, it sends a NOTIFY message to the Public Authoritative Name Server Sets. Upon receiving the NOTIFY message, the slave reads the Registered Homenet Domain and checks the NOTIFY is sent by the authorized master. This can be done using the shared secret (TSIG) or the public key (SIG(0)). Once the NOTIFY has been authenticated, the Public Authoritative Name Server Sets might consider the source IP address of the NOTIFY query to configure the masters attributes.

6.5. Master / Slave Synchronization: CPE / Reverse Public Authoritative Name Server Set

The CPE knows the zone to be synchronized by looking at its assigned prefix. The IP address of the slave is provided by the DHCP Reverse Public Authoritative Name Server Set Option (OPTION_REVERSE_NAME_SERVER_SET).

Configuration of the slave is performed as illustrated in Section 6.4.

7. DNS Zone Update Data

7.1. DNS Homenet Zone Template

The DNS Homenet Zone Template contains at least the related fields of the Public Authoritative Master(s) as well as the Homenet Registered Domain, that is SOA, and NS fields. This template might be generated automatically by the owner of the DHCP Server. For example, an ISP might provide a default Homenet Registered Domain as well as default Public Authoritative Master(s). This default settings should provide the CPE the necessary pieces of information to set the homenet naming architecture.

If the DNS Homenet Zone Template is not subject to modifications or updates, the owner of the template might only use DNSSEC to enable integrity check.

The DNS Homenet Zone Template might be subject to modification by the CPE. The advantage of using the standard DNS zone format is that standard DNS update mechanism can be used to perform updates. These updates might be accepted or rejected by the owner of the DNS Homenet Zone Template. Policies that defines what is accepted or rejected is out of scope of this document. However, in this document we assume the Registered Homenet Domain is used as an index by the Public Authoritative Name Server Set, and SIG(0), TSIG are used to authenticate the CPE. As a result, the Registered Homenet Domain should not be modified unless the Public Authoritative Name Server Set can handle with it.

7.2. DNS Homenet Zone

The DNS Homenet Zone might be generated from the DNS Homenet Zone Template. How the DNS Homenet Zone is generated is out of scope of this document. In some cases, the DNS Homenet Zone might be the exact copy of the DNS Homenet Zone Template. In other cases, it might be generated from the DNS Homenet Zone Template with additional RRsets. In some other cases, the DNS Homenet Zone might be generated without considering the DNS Homenet Zone Template, but only considering specific configuration rules.

In the current document the CPE only sets a single zone that is associated with one single Homenet Registered Domain. The domain might be assigned by the owner of the DNS Homenet Zone Template. This constrain does not prevent the CPE to use multiple domain names. How additional domains are considered is out of scope of this document. One way to handle these additional zones is to configure static redirections to the DNS Homenet Zone using CNAME [RFC2181], [RFC1034], DNAME [RFC6672] or CNAME+DNAME [I-D.sury-dnsext-cname-dname].

8. Payload Description

8.1. Security Field

The Security Field of the DHCP Option is represented in Figure 2. It indicates the security mechanism supported by the DNS Server. One of these mechanism MUST be chosen by the CPE in order to perform a transaction with the DNS server. See Section 5 for more details.

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Security           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Security Field

- DNS (Bit 0): indicates, when set to 1, that DNS without any security extension is supported.

- DNSSEC (Bit 1): indicates, when set to 1, that DNSSEC provides integrity protection. This can only be used for read operations like retrieving the DNS Homenet Zone Template.

- SIG(0) (Bit 2): indicates, when set to 1, that transaction protected by SIG(0) are supported.

- TSIG (Bit 3): indicates, when set to 1, that transaction using TSIG is supported. Note that if a shared secret has not been previously negotiated between the two party, it should be negotiated using TKEY. The TKEY exchanges MUST be protected with SIG(0) even though SIG(0) is not supported.

- Remaining Bits (Bit 4-15): MUST be set to 0 by the DHCP Server and ignored by the DHCP Client.

A Security field with all bits set to zero indicates the operation is not permitted. The Security field may be set to zero when updates operations are not permitted for the DNS Homenet Template. In any other case this is an error.

8.2. Update Field

The Update Field of the DHCP Option is represented in Figure 3. It indicates the update mechanism supported by the DNS server. See Section 6 for more details.

```
 0
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|    Update     |
+-+-+-+-+-+-+-+-+
```

Figure 3: Update Field

- Master / Slave (Bit 0): indicates, when set to 1, that DNS Server supports data synchronization using a Master / Slave mechanism.

- DNS Update (Bit 1): indicates, when set to 1, that DNS Server supports data synchronization using DNS Updates.

- Remaining Bits (Bit 2-7): MUST be set to 0 by the DHCP Server and ignored by the DHCP Client.

8.3. DHCP Public Key Option

The DHCP Public Key Option (OPTION_PUBLIC_KEY) indicates the Public Key that is used to authenticate the CPE.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       OPTION_PUBLIC_KEY       |          option-len           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
/                       Public Key Data                         /
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: DHCP Public Key Option

- OPTION_PUBLIC_KEY (variable): the option code for the DHCP Public Key Option.

- option-len (16 bits): length in octets of the option-data field as described in [RFC3315].

- Public Key Data: contains the Public Key. The format is the DNSKEY RDATA format as defined in [RFC4034].

8.4. DHCP Zone Template Option

The DHCP Zone Template Option (OPTION_DNS_ZONE_TEMPLATE) Option indicates the CPE how to retrieve the DNS Homenet Zone Template. It provides a FQDN the CPE SHOULD query with a DNS query of type AXFR. The option also specifies which security protocols are available on the authoritative server. DNS Homenet Zone Template update, if permitted MUST use the DNS Update mechanism.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    OPTION_DNS_ZONE_TEMPLATE   |          option-len           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       Security  (axfr)        |            Security           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
/                     Zone Template FQDN                        /
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: DHCP Zone Template Option

- OPTION_DNS_ZONE_TEMPLATE (variable): the option code for the DHCP Zone Template Option.

- option-len (16 bits): length in octets of the option-data field as described in [RFC3315].

- Security (axfr) (16 bits): defines which security protocols are supported by the DNS server. This field concerns the AXFR and consultation queries, not the update queries. See Section 8.1 for more details.

- Security (16 bits): defines which security protocols are supported by the DNS server. This field concerns the update. See Section 8.1 for more details.

- Zone Template FQDN FQDN (variable): the FQDN of the DNS server hosting the DNS Homenet Zone Template.

8.5. DHCP Public Authoritative Name Server Set Option

The DHCP Public Authoritative Name Server Set Option (OPTION_NAME_SERVER_SET) provides information so the CPE can upload the DNS Homenet Zone to the Public Authoritative Name Server Set. Finally, the option provides the security mechanisms that are available to perform the upload. The upload is performed via a DNS master / slave architecture or DNS updates.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    OPTION_NAME_SERVER_SET     |          option-len           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Security           |    Update    |                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                |
|                                                               |
/           Public Authoritative Name Server Set FQDN           /
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: DHCP Public Authoritative Name Server Set Option

- OPTION_NAME_SERVER_SET (16 bits): the option code for the DHCP Public Authoritative Name Server Set Option.

- option-len (16 bits): length in octets of the option-data field as described in [RFC3315].

- Security (16 bits): defines which security protocols are supported by the DNS server. See Section 8.1 for more details.

- Update (8 bits): defines which update mechanisms are supported by the DNS server. See Section 6 for more details.

- Public Authoritative Name Server Set FQDN (variable): the FQDN of the Public Authoritative Name Server Set.

8.6. DHCP Reverse Public Authoritative Name Server Set Option

The DHCP Reverse Public Authoritative Name Server Set Option (OPTION_REVERSE_NAME_SERVER_SET) provides information so the CPE can upload the DNS Homenet Zone to the Public Authoritative Name Server Set. The option provides the security mechanisms that are available to perform the upload. The upload is performed via a DNS master / slave architecture or DNS updates.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | OPTION_REVERSE_NAME_SERVER_SET|          option-len           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            Security           |    Update   |                 |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                 |
   |                                                               |
   /      Reverse Public Authoritative Name Server Set FQDN        /
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 7: DHCP Reverse Public Authoritative Name Server Set Option

- OPTION_REVERSE_NAME_SERVER_SET (16 bits): the option code for the DHCP Reverse Public Authoritative Name Server Set Option.

- option-len (16 bits): length in octets of the option-data field as described in [RFC3315].

- Security (16 bits): defines which security protocols are supported by the DNS server. See Section 8.1 for more details.

- Update (8 bits): defines which update mechanisms are supported by the DNS server. See Section 6 for more details.

- Reverse Public Authoritative Name Server Set FQDN (variable): The FQDN of the Reverse Public Authoritative Name Server Set.

9. DHCP Behavior

9.1. DHCPv6 Server Behavior

The DHCP Server sends the DHCP Zone Template Option (OPTION_DNS_ZONE_TEMPLATE), DHCP Public Authoritative Name Server Set Option (OPTION_NAME_SERVER_SET), DHCP Reverse Public Authoritative Name Server Set Option (OPTION_REVERSE_NAME_SERVER_SET) upon request by the DHCP Client.

The DHCP Server MAY receive a DHCP Public Key Option (OPTION_PUBLIC_KEY) from the CPE. Upon receipt of this DHCP Option, the DHCP Sever is expect to communicate this credential to the available DNS Servers like the DNS Template Server, the Public Authoritative Name Server Set and the Reverse Public Authoritative Name Server Set.

9.2. DHCPv6 Client Behavior

The DHCP Client MAY send a DHCP Public Key Option (OPTION_PUBLIC_KEY) to the DHCP Server. This Public Key authenticates the CPE.

The DHCP Client sends a DHCP Option Request Option (ORO) with the necessary DHCP options.

A CPE SHOULD only send the an ORO request for DHCP Options it needs or for information that needs to be up-to-date.

Upon receiving a DHCP option described in this document, the CPE SHOULD retrieve or update DNS zones using the associated security and update protocols.

9.3. DHCPv6 Relay Behavior

DHCP Relay behavior are not modified by this document.

10. IANA Considerations

The DHCP options detailed in this document is:

- OPTION_DNS_ZONE_TEMPLATE: TBD

- OPTION_NAME_SERVER_SET: TBD

- OPTION_REVERSE_NAME_SERVER_SET: TBD

- OPTION_PUBLIC_KEY: TBD

11. Security Considerations

11.1. DNSSEC is recommended to authenticate DNS hosted data

It is recommended that the (Reverse) DNS Homenet Zone is signed with DNSSEC. The zone may be signed by the CPE or by a third party. We recommend the zone to be signed by the CPE, and that the signed zone is uploaded.

11.2. Channel between the CPE and ISP DHCP Server MUST be secured

The document considers that the channel between the CPE and the ISP DHCP Server is trusted. More specifically, the CPE is authenticated and the exchanged messages are protected. The current document does not specify how to secure the channel. [RFC3315] proposes a DHCP authentication and message exchange protection, [RFC4301], [RFC5996] propose to secure the channel at the IP layer.

In fact, the channel MUST be secured because the CPE provides authentication credentials. Unsecured channel may result in CPE impersonation attacks.

11.3. CPEs are sensitive to DoS

CPE have not been designed for handling heavy load. The CPE are exposed on the Internet, and their IP address is publicly published on the Internet via the DNS. This makes the Home Network sensitive to Deny of Service Attacks. The resulting outsourcing architecture is described in [I-D.mglt-homenet-front-end-naming-delegation]. This document shows how the outsourcing architecture can be automatically set.

12. Acknowledgment

We would like to thank Tomasz Mrugalski, Marcin Siodelski and Bernie Volz for their comments on the design of the DHCP Options. We would also like to thank Mark Andrews, Andrew Sullivan and Lorenzo Colliti for their remarks on the architecture design. The designed solution has been largely been inspired by Mark Andrews's document [I-D.andrews-dnsop-pd-reverse] as well as discussions with Mark.

13. References

13.1. Normative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1996] Vixie, P., "A Mechanism for Prompt Notification of Zone Changes (DNS NOTIFY)", RFC 1996, August 1996.

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

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

[RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, July 1997.

[RFC2845] Vixie, P., Gudmundsson, O., Eastlake, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, May 2000.

[RFC2930] Eastlake, D., "Secret Key Establishment for DNS (TKEY RR)", RFC 2930, September 2000.

[RFC2931] Eastlake, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, September 2000.

[RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, November 2000.

[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.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, March 2005.

[RFC4035] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Protocol Modifications for the DNS Security Extensions", RFC 4035, March 2005.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5936] Lewis, E. and A. Hoenes, "DNS Zone Transfer Protocol (AXFR)", RFC 5936, June 2010.

[RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, January 2012.

[RFC6672] Rose, S. and W. Wijngaards, "DNAME Redirection in the DNS", RFC 6672, June 2012.

13.2. Informational References

[I-D.andrews-dnsop-pd-reverse]
Andrews, M., "Automated Delegation of IP6.ARPA reverse zones with Prefix Delegation", draft-andrews-dnsop-pd-reverse-02 (work in progress), November 2013.

[I-D.mglt-homenet-front-end-naming-delegation]
Migault, D., Cloetens, W., Griffiths, C., and R. Weber, "IPv6 Home Network Naming Delegation", draft-mglt-homenet-front-end-naming-delegation-03 (work in progress), October 2013.

[I-D.sury-dnsext-cname-dname]
Sury, O., "CNAME+DNAME Name Redirection", draft-sury-dnsext-cname-dname-00 (work in progress), April 2010.

Appendix A. Scenarios and impact on the End User

This section details various scenarios and discuss their impact on the end user.

A.1. Base Scenario

The base scenario is the one described in Section 4. It is typically the one of an ISP that manages the DHCP Server, and all DNS servers.

The end user subscribes to the ISP (foo), and at subscription time registers for example.foo as its Registered Homenet Domain example.foo. Since the ISP knows the Registered Homenet Domain and the Public Authoritative Master(s) the ISP is able to build the DNS Homenet Zone Template.

The ISP manages the DNS Template Server, so it is able to load the DNS Homenet Zone Template on the DNS Template Server.

When the CPE is plugged (at least the first time), it provides its Public Key to the DHCP Server. In this scenario, the DHCP Server and the DNS Servers are managed by the ISP so the DHCP Server can provide authentication credentials of the CPE to enable secure authenticated transaction between the CPE and these DNS servers. More specifically, credentials are provided to:

- Public Authoritative Name Server Set

- Reverse Public Authoritative Name Server Set

- DNS Template Server

The CPE can update the zone using DNS update or a master / slave configuration in a secure way.

The main advantage of this scenario is that the naming architecture is configured automatically and transparently for the end user.

The drawbacks are that the end user uses a Registered Homenet Domain managed by the ISP and that it relies on the ISP naming infrastructure.

A.2. Third Party Registered Homenet Domain

This section considers the case when the end user wants its home network to use example.com as a Registered Homenet Domain instead of example.foo that has been assigned by the ISP. We also suppose that example.com is not managed by the ISP.

This can also be achieved without any configuration. When the end user buys the domain name example.com, it may request to redirect the name example.com to example.foo using static redirection with CNAME [RFC2181], [RFC1034], DNAME [RFC6672] or CNAME+DNAME [I-D.sury-dnsext-cname-dname].

This configuration is performed once when the domain name example.com is registered. The only information the end user needs to know is the domain name assigned by the ISP. Once this configuration is done no additional configuration is needed anymore. More specifically, the CPE may be changed, the zone can be updated as in Appendix A.1 without any additional configuration from the end user.

The main advantage of this scenario is that the end user benefits from the Zero Configuration of the Base Scenario Appendix A.1. Then, the end user is able to register for its home network an unlimited number of domain names provided by an unlimited number of different third party providers.

The drawback of this scenario may be that the end user still rely on the ISP naming infrastructure. Note that the only case this may be inconvenient is when the DNS Servers provided by the ISPs results in high latency.

A.3. Third Party DNS Infrastructure

This scenario considers that the end user uses example.com as a Registered Homenet Domain, and does not want to rely on the authoritative servers provided by the ISP.

In this section we limit the outsourcing to the Public Authoritative Name Server Set and Public Authoritative Master(s) to a third party. All other DNS Servers DNS Template Server, Reverse Public Authoritative Master(s) and Reverse Public Authoritative Name Server Set remain managed by the ISP. The reason we consider that Reverse Public Authoritative Masters(s) and Reverse Public Authoritative Name Server Set remains managed by the ISP are that the prefix is managed by the ISP, so outsourcing these resources requires some redirection agreement with the ISP. More specifically the ISP will need to configure the redirection on one of its Reverse DNS Servers. That said, outsourcing these resources is similar as outsourcing Public Authoritative Name Server Set and Public Authoritative Master(s) to a third party. Similarly, the DNS Template Server can be easily outsourced as detailed in this section

Outsourcing Public Authoritative Name Server Set and Public Authoritative Master(s) requires:

- 1) Updating the DNS Homenet Zone Template: this can be easily done as detailed in Section 6 as the DNS Template Server is still managed by the ISP. Such modification can be performed once by any CPE. Once this modification has been performed, the CPE can be changed, the Public Key of the CPE may be changed, this does not need to be done another time. One can imagine a GUI on the CPE asking the end user to fill the field with Registered Homenet Domain, optionally Public Authoritative Master(s), with a button "Configure DNS Homenet Zone Template".

- 2) Updating the DHCP Server Information. In fact the Reverse Public Authoritative Name Server Set returned by the ISP is modified. One can imagine a GUI interface that enables the end user to modify its profile parameters. Again, this configuration update is done once-for-ever.

- 3) Upload the authentication credential of the CPE, that is the Public Key of the CPE, to the third party. Unless we use specific mechanisms, like communication between the DHCP Server and the third party, or a specific token that is plugged into the CPE, this operation is likely to be performed every time the CPE is changed, and every time the Public Key generated by the CPE is changed.

The main advantage of this scenario is that the DNS infrastructure is completely outsourced to the third party. Most likely the Public Key that authenticate the CPE need to be configured for every CPE. Configuration is expected to be CPE live-long.

Appendix B. Document Change Log

[RFC Editor: This section is to be removed before publication]

-03: Working Version Major modifications are:

- Redesigning options/scope: according to feed backs received from the IETF89 presentation in the dhc WG.

- Redesigning architecture: according to feed backs received from the IETF89 presentation in the homenet WG, discussion with Mark and Lorenzo.

-02: Working Version Major modifications are:

- Redesigning options/scope: As suggested by Bernie Volz

-01: Working Version Major modifications are:

- Remove the DNS Zone file construction: As suggested by Bernie Volz

- DHCPv6 Client behavior: Following options guide lines

- DHCPv6 Server behavior: Following options guide lines

-00: version published in the homenet WG. Major modifications are:

- Reformatting of DHCP Options: Following options guide lines

- DHCPv6 Client behavior: Following options guide lines

- DHCPv6 Server behavior: Following options guide lines

-00: First version published in dhc WG.

Authors' Addresses

Daniel Migault
Orange
38 rue du General Leclerc
92794 Issy-les-Moulineaux Cedex 9
France

Phone: +33 1 45 29 60 52
Email: daniel.migault@orange.com

Wouter Cloetens
SoftAtHome
vaartdijk 3 701
3018 Wijgmaal
Belgium

Email: wouter.cloetens@softathome.com

Chris Griffiths
Dyn
150 Dow Street
Manchester, NH 03101
US

Email: cgriffiths@dyn.com
URI: http://dyn.com

Ralf Weber
Nominum
2000 Seaport Blvd #400
Redwood City, CA 94063
US

Email: ralf.weber@nominum.com
URI: http://www.nominum.com