dhc Working Group
Internet-Draft
Intended status: Informational
Expires: April 29, 2015

Y. Cui
Z. Liu
C. Liu
Tsinghua University

October 26, 2014

DHCP4o6 Active Leasequery
draft-cui-dhc-dhcp4o6-active-leasequery-00

Abstract

As networks migrate towards IPv6, some entities still have the requirement for IPv4 configuration. DHCPv4 over DHCPv6 [RFC7341] provides a mechanism for obtaining IPv4 configuration information dynamically in IPv6 networks. DHCPv4/DHCPv6 Active Leasequery allows a client to get real-time DHCP address binding information data via TCP. This document describes an extension of DHCPv6 Active Leasequery to provide an mechanism to getting real-time DHCPv4 over DHCPv6 lease information.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The DHCPv6 Leasequery [RFC5007] extends the basic DHCPv6 capability [RFC3315] to allow an external entity to query a DHCPv6 server to recover individual lease state information about a particular IPv6 address or client in near real-time. The DHCPv6 Bulk Leasequery [RFC5460] extends DHCPv6 Leasequery [RFC5007] that allows an external entity to query a DHCPv6 server for bulk transfer of lease information via TCP. The Active Leasequery allows an entity not directly participated in DHCPv6 client-server transactions and caches the current DHCPv6 lease state in real-time.And for DHCPv4, there are also similar protocols for DHCPv4 lease. [RFC4388] [RFC6926]

As networks migrate towards IPv6, hosts in some IPv6 network also need DHCPv4 configuration using DHCPv4 over DHCPv6[RFC7341]. The lease information in DHCPv4 over DHCPv6 (i.e. DHCP4o6 lease information) contains DHCPv4 lease information (including IPv4 address and other DHCPv4 options) in DHCPv4 messages, and stateless DHCPv6 options in DHCPV4-QUERY/DHCPV4-RESPONSE messages. The capability of additional DHCPv6 options makes it different from original DHCPv4 [RFC2131]. One example usage is in Lightweight 4over6 dynamic provisioning: A client (lwB4) chooses its IPv6 tunnel source address and puts it into a DHCPv6 option (OPTION_DHCP4O6_SADDR) [I-D.fsc-softwire-dhcp4o6-saddr-opt] to tell the provisioning system. The tuple of client (lease IPv4 address, port set, IPv6 tunnel source address)is then used to create a binding entry in lwAFTR.

In the case that a requestor wants to get both DHCPv4 lease information and DHCPv6 lease information of the same client, it can run DHCPv4 Active Leasequery and DHCPv6 Active Leasequery separately, using the same client identifier to associate them together. However, it doesn’t work for a requestor getting DHCP4o6 lease information because there’s no DUID or any other DHCPv6 identifiers in DHCPV4-QUERY/DHCPV4-RESPONSE messages, thus the DHCPv6 options can only be associated with the DHCPv4 lease. A requestor asking for DHCP4o6 lease must get the DHCPv6 options along with the DHCPv4 lease information.

However, the DHCPv4 Active Leasequery mechanism doesn’t support providing DHCPv6 options, and the DHCPv6 Active Leasequery mechanism doesn’t support providing DHCPv4 lease information. This document describes an extension of DHCPv6 Active Leasequery, naming the DHCP4o6 Active Leasequery, to allow an entity get the mixed DHCPv4 over DHCPv6 lease information in real-time in IPv6 network.

2. Terminology

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. Solution

The DHCP4o6 Active Leasequery mechanism is modeled on the existing DHCPv4 over DHCPv6 protocol in [RFC7341], which combine DHCPv4 Active Leasequery and DHCPv6 Active Leasequery to providing real-time DHCPv4 lease and related DHCPv6 lease information in IPv6 network. The DHCP4o6 Active Leasequery requestors and DHCP4o6 servers communicate with each other using DHCPv6 Active Leasequery which contains DHCPv4 Message Option defined in [RFC7341].

DHCPv4 Message Option defined in [RFC7341] contains the DHCPv4 message sent by the DHCP client or server. In DHCP4o6 Active Leasequery scenario, DHCPv4 Message Option contains the DHCPv4 Active Leasequery message sent by requestor and DHCP4o6 server.

DHCP4o6 Leasequery requestor SHOULD obtain necessary IPv6 configuration and have the DHCP4o6 server IPv6 address available via configuration or some other means, and that it has unicast IPv6 reachability to the DHCP4o6 server.

DHCP4o6 Leasequery requestor creates a TCP connection to DHCP4o6 server as defined [I-D.ietf-dhc-dhcpv6-active-leasequery]. After establishing a connection, requestor sends a DHCPv6 ACTIVELEASEQUERY message with a DHCPv4 Message Option in it to query for DHCPv4 over DHCPv6 lease information. The DHCPv4 Message Option encapsulates the DHCPv4 DHCPACTIVELEASEQUERY message to describe the query for a DHCPv4 lease. And the related DHCPv6 options will be queried in

DHCPv6 query-options. The requestor MUST NOT put more than one DHCPv4 Message Option into a single DHCPv6 ACTIVELEASEQUERY message.

When received the DHCPv6 ACTIVELEASEQUERY message, DHCP4o6 server SHOULD address the DHCPv4 DHCPACTIVELEASEQUERY message in the DHCPv4 Message Option and the related DHCPv6 query in DHCPv6 query-options. DHCP4o6 server will reply with DHCPv6 LEASEQUERY-REPLY message/ LEASEQUERY-DATA message. When the server update DHCPv4 lease or related DHCPv6 information, it will generate a response to requestors. In response, the server sends updates of DHCPv4 over DHCPv6 lease information in the DHCPv6 LEASEQUERY-DATA message. The DHCPv4 lease sent to the requestor using DHCPv4 DHCPLEASEACTIVE or DHCPLEASEUNASSIGNED message which will be encapsulated in DHCPv4 Message Option. The related DHCPv6 options will be carried in the DHCPv6 OPTION_CLIENT_DATA option.

4. Use Case

As the method described above, it will provide the requestor with related DHCPv4 lease and DHCPv6 information of a DHCP client in real-time. It MAY be used in many cases. We will describe the using for Lightweight 4over6 [I-D.ietf-softwire-lw4over6]as an example.

In Lightweight 4over6, lwAFTR need the binding IPv6 address for the mapping table. lwAFTR can work as a DHCP4o6 Active Leasequery requestor to get real-time DHCPv4 lease and related DHCPv6 information. lwAFTR need all lwB4's IPv4 address, PSID, IPv6 address to make the maaping table for the tunnel. So, lwAFTR will send the DHCPv6 ACTIVELEASEQUERY message with DHCPv4 Message Option to query for DHCPv4 lease and related DHCPv6 lease information of a DHCP client. DHCPv4 ACTIVELEASEQUERY message in the DHCPv4 Message Option SHOULD contains the primary query as Query for All Configured IP addresses. And the Parameter Request List option in DHCPv4 ACTIVELEASEQUERY message SHOULD contains the DHCPv4 Port Parameters option defined in [I-D.ietf-dhc-dynamic-shared-v4allocation]. And in the OPTION_LQ_QUERY option in DHCPv6 ACTIVELEASEQUERY message, the DHCPv6 OPTION_ORO option MUST contains the DHCPv4 over DHCPv6 Source address option defined in [I-D.fsc-softwire-dhcp4o6-saddr-opt].

DHCP4o6 server configure the lwB4s with DHCPv4 lease, and get the binding IPv6 address during the process. As defined in [RFC7341], DHCP4o6 client query for DHCP4o6 server's address during DHCPv6 interaction. After receiving DHCP4o6 server's address, DHCP4o6 client will query for IPv4 address from DHCP4o6 server. At the same time, DHCP4o6 client MAY negotiate the binding IPv6 address with DHCP4o6 server, and DHCP4o6 server can record the binding IPv6 address as defined in [I-D.fsc-softwire-dhcp4o6-saddr-opt]. When DHCP4o6 server received DHCPv6 ACTIVELEASEQUERY message from lwAFTR, it SHOULD reply with the DHCPv6 LEASEQUERY-REPLY/LEASEQUERY-DATA message which contains the CPEs’ IPv4 address, PSID, IPv6 address and other information in the following time if there is update of DHCPv4 lease or DHCPv6 lease.

In other cases, DHCP4o6 server MAY get more DHCPv6 information or even the whole DHCPv6 lease by some means, it can provide more information to the requestors.

5. Security Considerations

To be continue

6. References

6.1. Normative References

[I-D.ietf-dhc-dhcpv6-active-leasequery]
Dushyant, D., Kinnear, K., and D. Kukrety, "DHCPv6 Active Leasequery", draft-ietf-dhc-dhcpv6-active-leasequery-01 (work in progress), March 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.

[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.

[RFC4388] Woundy, R. and K. Kinnear, "Dynamic Host Configuration Protocol (DHCP) Leasequery", RFC 4388, February 2006.

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

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

[RFC6926] Kinnear, K., Stapp, M., Desetti, R., Joshi, B., Russell, N., Kurapati, P., and B. Volz, "DHCPv4 Bulk Leasequery", RFC 6926, April 2013.

[RFC7341] Sun, Q., Cui, Y., Siodelski, M., Krishnan, S., and I. Farrer, "DHCPv4-over-DHCPv6 (DHCP 4o6) Transport", RFC 7341, August 2014.

6.2. Informative References

[I-D.fsc-softwire-dhcp4o6-saddr-opt]
Farrer, I., Sun, Q., and Y. Cui, "DHCPv4 over DHCPv6 Source Address Option", draft-fsc-softwire-dhcp4o6-saddr-opt-01 (work in progress), September 2014.

[I-D.ietf-dhc-dhcpv4-active-leasequery]
Kinnear, K., Stapp, M., Volz, B., and N. Russell, "Active DHCPv4 Lease Query", draft-ietf-dhc-dhcpv4-active-leasequery-01 (work in progress), June 2014.

[I-D.ietf-dhc-dynamic-shared-v4allocation]
Cui, Y., Qiong, Q., Farrer, I., Lee, Y., Sun, Q., and M. Boucadair, "Dynamic Allocation of Shared IPv4 Addresses", draft-ietf-dhc-dynamic-shared-v4allocation-02 (work in progress), September 2014.

[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-lw4over6-11 (work in progress), October 2014.

Authors’ Addresses

Yong Cui
Tsinghua University
Beijing 100084
P.R.China

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

ZiLong Liu
Tsinghua University
Beijing 100084
P.R.China

Phone: +86-10-6278-5822
Email: liuzilong8266@126.com

Cong Liu
Tsinghua University
Beijing 100084
P.R.China

Phone: +86-10-6278-5822
Email: gnocuil@gmail.com

```
Dynamic Host Configuration (DHC)                          T. Mrugalski, Ed.
Internet-Draft                                                  M. Siodelski
Obsoletes: 3315,3633,3736,7083 (if                                       ISC
           approved)                                              B. Volz, Ed.
Intended status: Standards Track                               A. Yourtchenko
Expires: August 26, 2015                                                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:

o The client reboots.

o The client is physically connected to a wired connection.

o The client returns from sleep mode.

o 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 ID	Server ID	IA_NA IA_TA	IA_PD	Option Request	Pref	Elap. Time	Relay Msg.	Auth.	Server 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 Code	Rap. Comm.	User Class	Vendor Class	Vendor Spec.	Inter. ID	Recon. Msg.	Recon. Accept	SOL_MAX_RT 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
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

dhc
Internet-Draft
Intended status: Standards Track
Expires: April 30, 2015

S. Jiang
Huawei Technologies Co., Ltd
S. Krishnan
Ericsson
T. Mrugalski
ISC

October 27, 2014

Privacy considerations for DHCP
draft-jiang-dhc-dhcp-privacy-00

Abstract

DHCP is a protocol that is used to provide addressing and configuration information to IPv4 hosts. This document discusses the various identifiers used by DHCP and the potential privacy issues.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Dynamic Host Configuration Protocol (DHCP) [RFC2131] is a protocol that is used to provide addressing and configuration information to IPv4 hosts. The DHCP protocol uses several identifiers that could become a source for gleaning additional information about the IPv4 host. This information may include device type, operating system information, location(s) that the device may have previously visited, etc. This document discusses the various identifiers used by DHCP and the potential privacy issues [RFC6973].

Future works may propose protocol changes to fix the privacy issues that have been analyzed in this document. It is out of scope for this document.

Editor notes: for now, the document is mainly considering the privacy of DHCP client. The privacy of DHCP server and relay agent are considered less important because they are open for public services. However, this may be a subject to change if further study shows opposite result.

2. Terminology

This section clarifies the terminology used throughout this document.

Stable identifier - any property disclosed by a DHCP client that does not change over time or changes very infrequently and is unique for said client in a given context. Examples include MAC address, client-id that does not change or a hostname. Stable identifier may or may not be globally unique.

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]. 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. Identifiers in DHCP

There are several identifiers used in DHCP. This section provides an introduction to the various options that will be used further in the document.

3.1. Client ID Option

The Client Identifier Option [RFC2131] is used to pass an explicit client identifier to a DHCP server. There is an analogous Server Identifier Option but it is not as interesting in the privacy context (unless a host can be convinced to start acting as a server).

The client identifier is an opaque key, which must be unique to that client within the subnet to which the client is attached. It typically remains stable after it has been initially generated. It may contain a hardware address, identical to the contents of the 'chaddr' field, or another type of identifier, such as a DNS name. It is recommended that client identifiers be generated by using the permanent link-layer address of the network interface that the client is trying to configure. [RFC4361] updates the recommendation of Client Identifiers to be "consists of a type field whose value is normally 255, followed by a four-byte IA_ID field, followed by the DUID for the client as defined in RFC 3315, section 9". This does not change the lifecycle of the Client Identifiers. Clients are expected to generate their Client Identifiers once (during first operation) and store it in a non-volatile storage or use the same deterministic algorithm to generate the same Client Identifier values again.

3.2. Address Fields & Options

The 'yiaddr' field [RFC2131] in DHCP message is used to allocate address from the server to the client.

The DHCPv4 specification [RFC2131] provides a way to specify the client link-layer address in the DHCPv4 message header. A DHCPv4 message header has 'htype' and 'chaddr' fields to specify the client link-layer address type and the link-layer address, respectively. The 'chaddr' field is used both as a hardware address for transmission of reply messages and as a client identifier.

The 'requested IP address' option [RFC2131] is used by client to suggest that a particular IP address be assigned.

3.3. Subscriber-ID Option

A DHCP relay includes a Subscriber-ID option [RFC3993] to associate some provider-specific information with clients' DHCP messages that is independent of the physical network configuration through which the subscriber is connected.

The "subscriber-id" assigned by the provider is intended to be stable as customers connect through different paths, and as network changes occur. The Subscriber-ID is an ASCII string, which is assigned and configured by the network provider.

3.4. Relay Agent Information Option and Sub-options

A DHCP relay agent includes a Relay Agent Information [RFC3046] to identify the remote host end of the circuit. It contains a "circuit ID" sub-option for the incoming circuit, which is an agent-local identifier of the circuit from which a DHCP client-to-server packet was received, and a "remote ID" sub-option which provides a trusted identifier for the remote high-speed modem.

Possible encoding of "circuit ID" sub-option includes: router interface number, switching hub port number, remote access server port number, frame relay DLCI, ATM virtual circuit number, cable data virtual circuit number, etc.

Possible encoding of the "remote ID" sub-option includes: a "caller ID" telephone number for dial-up connection, a "user name" prompted for by a remote access server, a remote caller ATM address, a "modem ID" of a cable data modem, the remote IP address of a point-to-point link, a remote X.25 address for X.25 connections, etc.

The link-selection sub-option [RFC3527] is used by any DHCP relay agent that desires to specify a subnet/link for a DHCP client request that it is relaying but needs the subnet/link specification to be different from the IP address the DHCP server should use when communicating with the relay agent. It contains an IP address, which can identify the client’s subnet/link.

3.5. Client FQDN Option

The Client Fully Qualified Domain Name (FQDN) option [RFC4702] is used by DHCP clients and servers to exchange information about the client’s fully qualified domain name and about who has the responsibility for updating the DNS with the associated AAAA and PTR RRs.

A client can use this option to convey all or part of its domain name to a DHCP server for the IP-address-to-FQDN mapping. In most case a client sends its hostname as a hint for the server. The DHCP server MAY be configured to modify the supplied name or to substitute a different name. The server should send its notion of the complete FQDN for the client in the Domain Name field.

3.6. Parameter Request List Option

The Parameter Request List option [RFC2131] is used to inform the server about options the client wants the server to send to the client. The content of a Parameter Request List option are the option codes for an option requested by the client.

3.7. Vendor Class and Vendor-Identifying Vendor Class Options

The Vendor Class option [RFC2131] and the Vendor-Identifying Vendor Class option [RFC3925] is used by a DHCP 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 the details of the hardware configuration of the host on which the client is running, or of industry consortium compliance, for example, the version of the operating system the client is running or the amount of memory installed on the client.

3.8. Civic Location Option

DHCP servers use the Civic Location Option [RFC4776] to delivery of the location information (the civic and postal addresses) to the DHCP clients. It may refer to three locations: the location of the DHCP server, the location of the network element believed to be closest to the client, or the location of the client, identified by the "what" element within the option.

3.9. Coordinate-Based Location Option

The GeoConf and GeoLoc options [RFC6225] is used by DHCP server to provide the coordinate-based geographic location information to the DHCP clients. It enables a DHCP client to obtain its geographic location.

After the relevant DHCP exchanges have taken place, the location information is stored on the end device rather than somewhere else, where retrieving it might be difficult in practice.

3.10. Client System Architecture Type Option

The Client System Architecture Type Option [RFC4578] is used by DHCP client to send a list of supported architecture types to the DHCP server. It is used to provide configuration information for a node that must be booted using the network rather than from local storage.

4. Existing Mechanisms That Affect Privacy

This section describes available DHCP mechanisms that one can use to protect or enhance one’s privacy.

4.1. DNS Updates

DNS Updates [RFC4704] defines a mechanism that allows both clients and server to insert into DNS domain information about clients. Both forward (AAAA) and reverse (PTR) resource records can be updated. This allows other nodes to conveniently refer to a host, despite the fact that its IP address may be changing.

This mechanism exposes two important pieces of information: current address (which can be mapped to current location) and client’s hostname. The stable hostname can then by used to correlate the client across different network attachments even when its IP addresses keep changing.

4.2. Allocation strategies

A DHCP server running in typical, stateful mode is given a task of managing one or more pools of IP address resources. When a client requests a resource, server must pick a resource out of configured pool. Depending on the server’s implementation, various allocation strategies are possible. Choices in this regard may have privacy implications.

Iterative allocation - a server may choose to allocate addresses one by one. That strategy has the benefit of being very fast, thus can be favored in deployments that prefer performance. However, it makes the resources very predictable. Also, since the resources allocated tend to be clustered at the beginning of available pool, it makes scanning attacks much easier.

Identifier-based allocation - a server may choose to allocate an address that is based on one of available identifiers, e.g. client identifier or MAC address. It is also convenient, as returning client is very likely to get the same address. Those properties are convenient for system administrators, so DHCP server implementors are often requested to implement it. On the other hand, the downside of such allocation is that the client has a very stable IP address. That means that correlation of activities over time, location tracking, address scanning and OS/vendor discovery apply.

Hash allocation - it’s an extension of identifier based allocation. Instead of using the identifier directly, it is being hashed first. If the hash is implemented correctly, it removes the flaw of disclosing the identifier, a property that eliminates susceptibility to address scanning and OS/vendor discovery. If the hash is poorly implemented (e.g. can be reverted), it introduces no improvement over identifier-based allocation.

Random allocation - a server can pick a resource randomly out of available pool. That strategy works well in scenarios where pool utilization is small, as the likelihood of collision (resulting in the server needing to repeat randomization) is small. With the pool allocation increasing, the collision is disproportionally large, due to birthday paradox. With high pool utilization (e.g. when 90% of available resources being allocated already), the server will use most computational resources to repeatedly pick a random resource, which will degrade its performance. This allocation scheme essentially prevents returning clients from getting the same address again. On the other hand, it is beneficial from privacy perspective as addresses generated that way are not susceptible to correlation attacks, OS/vendor discovery attacks or identity discovery attacks. Note that even though the address itself may be resilient to a given attack, the client may still be susceptible if additional information is disclosed other way, e.g. client's address can be randomized, but it still can leak its MAC address in client-id option.

Other allocation strategies may be implemented.

However, giving the limited resource of IPv4 public address pool, allocation mechanism in IPv4 may not provide much protection, while in IPv6, the network has very large address space to distribute the address allocation.

5. Attacks

5.1. Device type discovery

The type of device used by the client can be guessed by the attacker using the Vendor Class Option, the 'chaddr' field, and by parsing the Client ID Option. All of those options may contain OUI (Organizationally Unique Identifier) that represents the device's vendor. That knowledge can be used for device-specific vulnerability exploitation attacks.

5.2. Operating system discovery

The operating system running on a client can be guessed using the Vendor Class option, the Client System Architecture Type option, or by using fingerprinting techniques on the combination of options requested using the Parameter Request List option.

5.3. Finding location information

The location information can be obtained by the attacker by many means. The most direct way to obtain this information is by looking into a server initiated message that contains the Civic Location, GeoConf, or GeoLoc options. It can also be indirectly inferred using the Relay Agent Information option, with the remote ID sub-option (e.g. using a telephone number), the circuit ID option (e.g. if an access circuit on an Access Node corresponds to a civic location), or the Subscriber ID Option (if the attacker has access to subscriber info).

5.4. Finding previously visited networks

When DHCP clients connect to a network, they attempt to obtain the same address they had used before they attached to the network. They do this by putting the previously assigned address in the requested IP address option. By observing these addresses, an attacker can identify the network the client had previously visited.

5.5. Finding a stable identity

An attacker might use a stable identity gleaned from DHCP messages to correlate activities of a given client on unrelated networks. The Client FQDN option, the Subscriber ID Option and the Client ID options can serve as long lived identifiers of DHCP clients. The Client FQDN option can also provide an identity that can easily be correlated with web server activity logs.

5.6. Pervasive monitoring

This is an enhancement, or a combination of most aforementioned mechanisms. Operator who controls non-trivial number of access points or network segments, may use obtained information about a single client and observer client’s habits.

5.7. Finding client’s IP address or hostname

Many DHCP deployments use DNS Updates [RFC4702] that put client’s information (current IP address, client’s hostname). Client ID is also disclosed, able it in not easily accessible form (SHA-256 digest of the client-id). Although SHA-256 is irreversible, so DHCID can’t be converted back to client-id. However, SHA-256 digest can be used as a unique identifier that is accessible by any host.

5.8. Correlation of activities over time

As with other identifiers, an IP address can be used to correlate the activities of a host for at least as long as the lifetime of the address. If that address was generated from some other, stable identifier and that generation scheme can be deducted by an attacker, the duration of correlation attack extends to that identifier. In many cases, its lifetime is equal to the lifetime of the device itself.

5.9. Location tracking

If a stable identifier is used for assigning an address and such mapping is discovered by an attacker. In particular both passive (a service that the client connects to can log client’s address and draw conclusions regarding its location and movement patterns based on address it is connecting from) and active (attacker can send ICMP echo requests or other probe packets to networks of suspected client locations).

5.10. Leasequery & bulk leasequery

Attackers may pretend as an access concentrator, either DHCP relay agent or DHCP client, to obtain location information directly from the DHCP server(s) using the DHCP Leasequery [RFC4388], [RFC6148] mechanism.

Location information is information needed by the access concentrator to forward traffic to a broadband-accessible host. This information includes knowledge of the host hardware address, the port or virtual circuit that leads to the host, and/or the hardware address of the intervening subscriber modem.

Furthermore, the attackers may use DHCP bulk leasequery [RFC6926] mechanism to obtain bulk information about DHCP bindings, even without knowing the target bindings.

6. Security Considerations

TBD

7. Privacy Considerations

This document at its entirety discusses privacy considerations in DHCP. As such, no separate section about this is needed.

8. IANA Considerations

This draft does not request any IANA action.

9. Acknowledgements

The authors would like to thanks the valuable comments made by Stephen Farrell, Ted Lemon, Ines Robles, Russ White, Christian Schaefer and other members of DHC WG.

This document was produced using the xml2rfc tool [RFC2629].

10. References

10.1. Normative References

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

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

[RFC3046] Patrick, M., "DHCP Relay Agent Information Option", RFC 3046, January 2001.

[RFC3527] Kinnear, K., Stapp, M., Johnson, R., and J. Kumarasamy, "Link Selection sub-option for the Relay Agent Information Option for DHCPv4", RFC 3527, April 2003.

[RFC3925] Littlefield, J., "Vendor-Identifying Vendor Options for Dynamic Host Configuration Protocol version 4 (DHCPv4)", RFC 3925, October 2004.

[RFC3993] Johnson, R., Palaniappan, T., and M. Stapp, "Subscriber-ID Suboption for the Dynamic Host Configuration Protocol (DHCP) Relay Agent Option", RFC 3993, March 2005.

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

[RFC4388] Woundy, R. and K. Kinnear, "Dynamic Host Configuration Protocol (DHCP) Leasequery", RFC 4388, February 2006.

[RFC4702] Stapp, M., Volz, B., and Y. Rekhter, "The Dynamic Host Configuration Protocol (DHCP) Client Fully Qualified Domain Name (FQDN) Option", RFC 4702, October 2006.

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

[RFC4776] Schulzrinne, H., "Dynamic Host Configuration Protocol (DHCPv4 and DHCPv6) Option for Civic Addresses Configuration Information", RFC 4776, November 2006.

[RFC6148] Kurapati, P., Desetti, R., and B. Joshi, "DHCPv4 Lease Query by Relay Agent Remote ID", RFC 6148, February 2011.

[RFC6225] Polk, J., Linsner, M., Thomson, M., and B. Aboba, "Dynamic Host Configuration Protocol Options for Coordinate-Based Location Configuration Information", RFC 6225, July 2011.

[RFC6926] Kinnear, K., Stapp, M., Desetti, R., Joshi, B., Russell, N., Kurapati, P., and B. Volz, "DHCPv4 Bulk Leasequery", RFC 6926, April 2013.

10.2. Informative References

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[RFC4578] Johnston, M. and S. Venaas, "Dynamic Host Configuration Protocol (DHCP) Options for the Intel Preboot eXecution Environment (PXE)", RFC 4578, November 2006.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, July 2013.

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

Suresh Krishnan
Ericsson
8400 Decarie Blvd.
Town of Mount Royal, QC
Canada

Phone: +1 514 345 7900 x42871
Email: suresh.krishnan@ericsson.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 S. Krishnan
Internet-Draft Ericsson
Intended status: Standards Track T. Mrugalski
Expires: April 30, 2015 ISC
S. Jiang
Huawei Technologies Co., Ltd
October 27, 2014

Privacy considerations for DHCPv6
draft-krishnan-dhc-dhcpv6-privacy-00

Abstract

DHCPv6 is a protocol that is used to provide addressing and configuration information to IPv6 hosts. This document discusses the various identifiers used by DHCPv6 and the potential privacy issues.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

DHCPv6 [RFC3315] is a protocol that is used to provide addressing and configuration information to IPv6 hosts. The DHCPv6 protocol uses several identifiers that could become a source for gleaning additional information about the IPv6 host. This information may include device type, operating system information, location(s) that the device may have previously visited, etc. This document discusses the various identifiers used by DHCPv6 and the potential privacy issues [RFC6973].

Future works may propose protocol changes to fix the privacy issues that have been analyzed in this document. It is out of scope for this document.

Editor notes: for now, the document is mainly considering the privacy of DHCPv6 client. The privacy of DHCPv6 server and relay agent are considered less important because they are open for public services. However, this may be a subject to change if further study shows opposite result.

2. Terminology

This section clarifies the terminology used throughout this document.

Stable identifier - any property disclosed by a DHCPv6 client that does not change over time or changes very infrequently and is unique for said client in a given context. Examples include MAC address, client-id that does not change or a hostname. Stable identifier may or may not be globally unique.

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]. 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. Identifiers in DHCPv6

There are several identifiers used in DHCPv6. This section provides an introduction to the various options that will be used further in the document.

3.1. DUID

Each DHCPv6 client and server has a DHCPv6 Unique Identifier (DUID) [RFC3315]. The DUID is designed to be unique across all DHCPv6 clients and servers, and to remain stable after it has been initially generated. The DUID can be of different forms. Commonly used forms are based on the link-layer address of one of the device's network interfaces (with or without a timestamp), on the Universally Unique IDentifier (UUID) [RFC6355]. The default type, recommended by [RFC3315], is DUID-LLT that is based on link-layer address, which is commonly implemented in most popular clients.

It is important to understand DUID lifecycle. Clients and servers are expected to generate their DUID once (during first operation) and store it in a non-volatile storage or use the same deterministic algorithm to generate the same DUID value again. This means that most implementations will use the available link-layer address during its first boot. Even if the administrator enables privacy extensions (see [RFC4941]) and its equivalent for link-layer address randomization, it is likely that those privacy mechanisms were disabled during the first device boot. Hence the original, unobfuscated link-layer address will likely end up being announced as client DUID, even if the link-layer address has changed (or even if being changed on a periodic basis).

3.2. Client ID Option

The Client Identifier Option (OPTION_CLIENTID) [RFC3315] is used to carry the DUID of a DHCPv6 client between a client and a server. There is an analogous Server Identifier Option but it is not as interesting in the privacy context (unless a host can be convinced to start acting as a server). Client ID is an example of DUID. See Section 3.1 for relevant discussion about DUIDs.

3.3. IA_NA, IA_TA, IA_PD, IA Address and IA Prefix Options

The Identity Association for Non-temporary Addresses (IA_NA) option [RFC3315] is used to carry the parameters and any non-temporary addresses associated with the given IA_NA. The Identity Association for Temporary Addresses (IA_TA) option [RFC3315] is analogous to the IA_NA option but for temporary addresses. The IA Address option [RFC3315] is used to specify IPv6 addresses associated with an IA_NA or an IA_TA and is encapsulated within the Options field of such an IA_NA or IA_TA option. The Identity Association for Prefix Delegation (IA_PD) [RFC3633] option is used to carry the prefixes that are assigned to the requesting router. IA Prefix option [RFC3633] is used to specify IPv6 prefixes associated with an IA_PD and is encapsulated within the Options field of such an IA_PD option.

To differentiate between instances of the same type of IA containers, each IA_NA, IA_TA and IA_PD options have an IAID field that is unique for each client/option type pair. It is up to the client to pick unique IAID values. At least one popular implementation uses last four octets of the link-layer address. In most cases, that means that merely two bytes are missing for a full link-layer address reconstruction. However, the first three octets in a typical link-layer address are vendor identifier. That can be determined with high level of certainty using other means, thus allowing full link-layer address discovery.

3.4. Interface ID

A DHCPv6 relay includes the Interface ID [RFC3315] option to identify the interface on which it received the client message that is being relayed.

Although in principle Interface ID can be arbitrarily long with completely random values, it is often a text string that includes the relay agent name followed by interface name. This can be used for fingerprinting the relay or determining client's point of attachment.

3.5. Subscriber ID

A DHCPv6 relay includes a Subscriber ID option [RFC4580] to associate some provider-specific information with clients' DHCPv6 messages that is independent of the physical network configuration.

In many deployments, the relay agent that inserts this option is configured to use client's link-layer address as Subscriber ID.

3.6. Remote ID

A DHCPv6 relay includes a Remote ID option [RFC4649] to identify the remote host end of the circuit.

The remote-id is vendor specific, for which the vendor is indicated in the enterprise-number field. The remote-id field may encode the information that identified the DHCPv6 clients:

o a "caller ID" telephone number for dial-up connection

o a "user name" prompted for by a Remote Access Server

o a remote caller ATM address o a "modem ID" of a cable data modem

o the remote IP address of a point-to-point link o an interface or port identifier

3.7. Client FQDN Option

The Client Fully Qualified Domain Name (FQDN) option [RFC4704] is used by DHCPv6 clients and servers to exchange information about the client’s fully qualified domain name and about who has the responsibility for updating the DNS with the associated AAAA and PTR RRs.

A client can use this option to convey all or part of its domain name to a DHCPv6 server for the IPv6-address-to-FQDN mapping. In most case a client sends its hostname as a hint for the server. The DHCPv6 server MAY be configured to modify the supplied name or to substitute a different name. The server should send its notion of the complete FQDN for the client in the Domain Name field.

3.8. Client Link-layer Address Option

The Client link-layer address option [RFC6939] is used by first-hop DHCPv6 relays to provide the client’s link-layer address towards the server.

DHCPv6 relay agents that receive messages originating from clients may include the link-layer source address of the received DHCPv6 message in the Client Link-Layer Address option, in relayed DHCPv6 Relay-Forward messages.

3.9. Option Request Option

DHCPv6 clients include an Option Request option [RFC3315] in DHCPv6 messages to inform the server about options the client wants the server to send to the client.

The content of an Option Request option are the option codes for an option requested by the client. 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.

3.10. Vendor Class Option

This Vendor Class option [RFC3315] is used by a DHCPv6 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, for example, the version of the operating system the client is running or the amount of memory installed on the client.

3.11. Civic Location Option

DHCPv6 servers use the Civic Location option [RFC4776] to delivery of location information (the civic and postal addresses) from the DHCPv6 server to the DHCPv6 clients. It may refer to three locations: the location of the DHCPv6 server, the location of the network element believed to be closest to the client, or the location of the client, identified by the "what" element within the option.

3.12. Coordinate-Based Location Option

The GeoLoc options [RFC6225] is used by DHCPv6 server to provide the coordinate- based geographic location information to the DHCPv6 clients. It enable a DHCPv6 client to obtain its location.

After the relevant DHCPv6 exchanges have taken place, the location information is stored on the end device rather than somewhere else, where retrieving it might be difficult in practice.

3.13. Client System Architecture Type Option

The Client System Architecture Type option [RFC5970] is used by DHCPv6 client to send a list of supported architecture types to the DHCPv6 server. It is used to provide configuration information for a node that must be booted using the network rather than from local storage.

4. Existing Mechanisms That Affect Privacy

This section describes available DHCPv6 mechanisms that one can use to protect or enhance one’s privacy.

4.1. Temporary addresses

[RFC3315] defines a mechanism for a client to request temporary addresses. The idea behind temporary addresses is that a client can request a temporary address for a specific purpose, use it, and then never renew it. i.e. let it expire.

There are number of serious issues, both protocolar and implementational, that make them nearly useless for their original goal. First, [RFC3315] does not include T1 and T2 renewal timers in IA_TA (a container for temporary addresses). However, it mentions that temporary addresses can be renewed. Many client implementations renew those addresses during a renewal procedure initiated by other resources (non-temporary addresses or prefixes), thus forfeiting shortliveness. Second, [RFC4704] allows servers to update DNS for assigned temporary addresses. Publishing client's IPv6 address in DNS that is publicly available is a major privacy breach.

4.2. DNS Updates

DNS Updates [RFC4704] defines a mechanism that allows both clients and server to insert into DNS domain information about clients. Both forward (AAAA) and reverse (PTR) resource records can be updated. This allows other nodes to conveniently refer to a host, despite the fact that its IPv6 address may be changing.

This mechanism exposes two important pieces of information: current address (which can be mapped to current location) and client's hostname. The stable hostname can then by used to correlate the client across different network attachments even when its IPv6 address keeps changing.

4.3. Allocation strategies

A DHCPv6 server running in typical, stateful mode is given a task of managing one or more pools of IPv6 resources (currently non-temporary addresses, temporary addresses and/or prefixes, but more resource types may be defined in the future). When a client requests a resource, server must pick a resource out of configured pool. Depending on the server's implementation, various allocation strategies are possible. Choices in this regard may have privacy implications.

Iterative allocation - a server may choose to allocate addresses one by one. That strategy has the benefit of being very fast, thus can be favored in deployments that prefer performance. However, it makes the resources very predictable. Also, since the resources allocated tend to be clustered at the beginning of available pool, it makes scanning attacks much easier.

Identifier-based allocation - a server may choose to allocate an address that is based on one of available identifiers, e.g. IID or MAC address. This has a property of being convenient for converting IP address to/from other identifiers, especially if the identifier is or contains MAC address. It is also convenient, as returning client is very likely to get the same address, even if the server does not store previous client's address. Those properties are convenient for system administrators, so DHCPv6 server implementors are sometimes requested to implement it. There is at least one implementation that supports it. On the other hand, the downside of such allocation is that the client now discloses its identifier in its IPv6 address to all services it connects to. That means that correlation of activities over time, location tracking, address scanning and OS/vendor discovery apply.

Hash allocation - it’s an extension of identifier based allocation. Instead of using the identifier directly, it is being hashed first. If the hash is implemented correctly, it removes the flaw of disclosing the identifier, a property that eliminates susceptibility to address scanning and OS/vendor discovery. If the hash is poorly implemented (e.g. can be reverted), it introduces no improvement over identifier-based allocation.

Random allocation - a server can pick a resource randomly out of available pool. That strategy works well in scenarios where pool utilization is small, as the likelihood of collision (resulting in the server needing to repeat randomization) is small. With the pool allocation increasing, the collision is disproportionally large, due to birthday paradox. With high pool utilization (e.g. when 90% of available resources being allocated already), the server will use most computational resources to repeatedly pick a random resource, which will degrade its performance. This allocation scheme essentially prevents returning clients from getting the same address or prefix again. On the other hand, it is beneficial from privacy perspective as addresses and prefixes generated that way are not susceptible to correlation attacks, OS/vendor discovery attacks or identity discovery attacks. Note that even though the address or prefix itself may be resilient to a given attack, the client may still be susceptible if additional information is disclosed other way, e.g. client’s address can be randomized, but it still can leak its MAC address in client-id option.

Other allocation strategies may be implemented.

5. Attacks

5.1. Device type discovery (fingerprinting)

The type of device used by the client can be guessed by the attacker using the Vendor Class option, the Client Link-layer Address option, and by parsing the Client ID option. All of those options may contain OUI (Organizationally Unique Identifier) that represents the device’s vendor. That knowledge can be used for device-specific vulnerability exploitation attacks. See Section 3.4 of [I-D.ietf-6man-ipv6-address-generation-privacy] for a discussion about this type of attack.

5.2. Operating system discovery (fingerprinting)

The operating system running on a client can be guessed using the Vendor Class option, the Client System Architecture Type option, or by using fingerprinting techniques on the combination of options requested using the Option Request option. See Section 3.4 of [I-D.ietf-6man-ipv6-address-generation-privacy] for a discussion about this type of attack.

5.3. Finding location information

The location information can be obtained by the attacker by many means. The most direct way to obtain this information is by looking into a server initiated message that contains the Civic Location or GeoLoc option. It can also be indirectly inferred using the Remote ID Option (e.g. using a telephone number), the Interface ID option (e.g. if an access circuit on an Access Node corresponds to a civic location), or the Subscriber ID Option (if the attacker has access to subscriber info).

5.4. Finding previously visited networks

When DHCPv6 clients connect to a network, they attempt to obtain the same address they had used before they attached to the network. They do this by putting the previously assigned address(es) in the IA Address Option(s) inside the IA_NA, IA_TA. By observing these addresses, an attacker can identify the network the client had previously visited.

5.5. Finding a stable identity

An attacker might use a stable identity gleaned from DHCPv6 messages to correlate activities of a given client on unrelated networks. The Client FQDN option, the Subscriber ID Option and the Client ID options can serve as long lived identifiers of DHCPv6 clients. The Client FQDN option can also provide an identity that can easily be correlated with web server activity logs.

5.6. Pervasive monitoring

This is an enhancement, or a combination of most aforementioned mechanisms. Operator, who controls non-trivial number of access points or network segments, may use obtained information about a single client and observer client’s habits.

5.7. Finding client’s IP address or hostname

Many DHCPv6 deployments use DNS Updates [RFC4704] that put client’s information (current IP address, client’s hostname). Client ID is also disclosed, able it in not easily accessible form (SHA-256 digest of the client-id). Although SHA-256 is irreversible, so DHCPv6 client ID can’t be converted back to client-id. However, SHA-256 digest can be used as a unique identifier that is accessible by any host.

5.8. Correlation of activities over time

As with other identifiers, an IPv6 address can be used to correlate the activities of a host for at least as long as the lifetime of the address. If that address was generated from some other, stable identifier and that generation scheme can be deducted by an attacker, the duration of correlation attack extends to that identifier. In many cases, its lifetime is equal to the lifetime of the device itself. See Section 3.1 of [I-D.ietf-6man-ipv6-address-generation-privacy] for detailed discussion.

5.9. Location tracking

If a stable identifier is used for assigning an address and such mapping is discovered by an attacker (e.g. a server that uses IEEE-identifier-based IID to generate IPv6 address), all scenarios discussed in Section 3.2 of [I-D.ietf-6man-ipv6-address-generation-privacy] apply. In particular both passive (a service that the client connects to can log client’s address and draw conclusions regarding its location and movement patterns based on prefix it is connecting from) and active (attacker can send ICMPv6 echo requests or other probe packets to networks of suspected client locations).

5.10. Leasequery & bulk leasequery

Attackers may pretend as an access concentrator, either DHCPv6 relay agent or DHCPv6 client, to obtain location information directly from the DHCP server(s) using the DHCPv6 Leasequery [RFC5007] mechanism.

Location information is information needed by the access concentrator to forward traffic to a broadband-accessible host. This information includes knowledge of the host hardware address, the port or virtual circuit that leads to the host, and/or the hardware address of the intervening subscriber modem.

Furthermore, the attackers may use DHCPv6 bulk leasequery [RFC5460] mechanism to obtain bulk information about DHCPv6 bindings, even without knowing the target bindings.

6. Security Considerations

TBD

7. Privacy Considerations

This document at its entirety discusses privacy considerations in DHCPv6. As such, no separate section about this is needed.

8. IANA Considerations

This draft does not request any IANA action.

9. Acknowledgements

The authors would like to thanks the valuable comments made by Stephen Farrell, Ted Lemon, Ines Robles, Russ White, Christian Schaefer and other members of DHC WG.

This document was produced using the xml2rfc tool [RFC2629].

10. References

10.1. 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.

10.2. Informative References

[I-D.ietf-6man-ipv6-address-generation-privacy]
Cooper, A., Gont, F., and D. Thaler, "Privacy Considerations for IPv6 Address Generation Mechanisms", draft-ietf-6man-ipv6-address-generation-privacy-02 (work in progress), October 2014.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

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

[RFC4580] Volz, B., "Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Relay Agent Subscriber-ID Option", RFC 4580, June 2006.

[RFC4649] Volz, B., "Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Relay Agent Remote-ID Option", RFC 4649, August 2006.

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

[RFC4776] Schulzrinne, H., "Dynamic Host Configuration Protocol (DHCPv4 and DHCPv6) Option for Civic Addresses Configuration Information", RFC 4776, November 2006.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, September 2007.

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

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

[RFC5970] Huth, T., Freimann, J., Zimmer, V., and D. Thaler, "DHCPv6 Options for Network Boot", RFC 5970, September 2010.

[RFC6225] Polk, J., Linsner, M., Thomson, M., and B. Aboba, "Dynamic Host Configuration Protocol Options for Coordinate-Based Location Configuration Information", RFC 6225, July 2011.

[RFC6355] Narten, T. and J. Johnson, "Definition of the UUID-Based DHCPv6 Unique Identifier (DUID-UUID)", RFC 6355, August 2011.

[RFC6939] Halwasia, G., Bhandari, S., and W. Dec, "Client Link-Layer Address Option in DHCPv6", RFC 6939, May 2013.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, July 2013.

Authors' Addresses

Suresh Krishnan
Ericsson
8400 Decarie Blvd.
Town of Mount Royal, QC
Canada

Phone: +1 514 345 7900 x42871
Email: suresh.krishnan@ericsson.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

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

Network Working Group   F. Templin, Ed.
Internet-Draft   Boeing Research & Technology
Obsoletes: rfc5320, rfc5558, rfc5720,   May 10, 2018
           rfc6179, rfc6706 (if
           approved)
Intended status: Standards Track
Expires: November 11, 2018

Asymmetric Extended Route Optimization (AERO)
draft-templin-aerolink-82.txt

Abstract

This document specifies the operation of IP over tunnel virtual links using Asymmetric Extended Route Optimization (AERO). Nodes attached to AERO links can exchange packets via trusted intermediate routers that provide forwarding services to reach off-link destinations and route optimization services for improved performance. AERO provides an IPv6 link-local address format that supports operation of the IPv6 Neighbor Discovery (ND) protocol and links ND to IP forwarding. Dynamic link selection, mobility management, quality of service (QoS) signaling and route optimization are naturally supported through dynamic neighbor cache updates, while IPv6 Prefix Delegation (PD) is supported by network services such as the Dynamic Host Configuration Protocol for IPv6 (DHCPv6). AERO is a widely-applicable tunneling solution especially well-suited to aviation services, mobile Virtual Private Networks (VPNs) and other applications as described in this document.

Status of This Memo

Table of Contents

1. Introduction

This document specifies the operation of IP over tunnel virtual links using Asymmetric Extended Route Optimization (AERO). The AERO link can be used for tunneling between neighboring nodes over either IPv6 or IPv4 networks, i.e., AERO views the IPv6 and IPv4 networks as equivalent links for tunneling. Nodes attached to AERO links can exchange packets via trusted intermediate routers that provide forwarding services to reach off-link destinations and route optimization services for improved performance [RFC5522].

AERO provides an IPv6 link-local address format that supports operation of the IPv6 Neighbor Discovery (ND) [RFC4861] protocol and links ND to IP forwarding. Dynamic link selection, mobility management, quality of service (QoS) signaling and route optimization are naturally supported through dynamic neighbor cache updates, while IPv6 Prefix Delegation (PD) is supported by network services such as the Dynamic Host Configuration Protocol for IPv6 (DHCPv6) [RFC3315][RFC3633].

A node’s AERO interface can be configured over multiple underlying interfaces. From the standpoint of ND, AERO interface neighbors therefore may appear to have multiple link-layer addresses (i.e., the addresses assigned to underlying interfaces). Each link-layer address is subject to change due to mobility and/or QoS fluctuations, and link-layer address changes are signaled by ND messaging the same as for any IPv6 link.

AERO is applicable to a wide variety of use cases. For example, it can be used to coordinate the Virtual Private Network (VPN) links of mobile nodes (e.g., cellphones, tablets, laptop computers, etc.) that connect into a home enterprise network via public access networks using services such as OpenVPN [OVPN]. AERO is also applicable to aviation services for both manned and unmanned aircraft where the aircraft is treated as a mobile node that can connect an Internet of Things (IoT). Other applicable use cases are also in scope.

The remainder of this document presents the AERO specification.

2. Terminology

The terminology in the normative references applies; the following terms are defined within the scope of this document:

IPv6 Neighbor Discovery (ND)
: an IPv6 control message service for coordinating neighbor relationships between nodes connected to a common link. The ND service used by AERO is specified in [RFC4861].

IPv6 Prefix Delegation (PD)
: a networking service for delegating IPv6 prefixes to nodes on the link. The nominal PD service is DHCPv6 [RFC3315] [RFC3633], however other services (e.g., alternate ND options, network management, static configuration, etc.) are also possible.

(native) Internetwork
: a connected IPv6 or IPv4 network topology over which the AERO link virtual overlay is configured and native peer-to-peer communications are supported. Example Internetworks include the global public Internet, private enterprise networks, aviation networks, etc.

AERO link
: a Non-Broadcast, Multiple Access (NBMA) tunnel virtual overlay configured over an underlying Internetwork. All nodes on the AERO link appear as single-hop neighbors from the perspective of the virtual overlay even though they may be separated by many underlying Internetwork hops. The AERO mechanisms can also operate over native link types (e.g., Ethernet, WiFi etc.) when a tunnel virtual overlay is not needed.

AERO interface
: a node’s attachment to an AERO link. Since the addresses assigned to an AERO interface are managed for uniqueness, AERO interfaces do not require Duplicate Address Detection (DAD) and therefore set the administrative variable DupAddrDetectTransmits to zero [RFC4862].

AERO address
: an IPv6 link-local address constructed as specified in Section 3.4.

AERO node
: a node that is connected to an AERO link.

AERO Client ("Client")
: a node that requests IP PDs from one or more AERO Servers. Following PD, the Client assigns an AERO address to the AERO interface for use in ND exchanges with other AERO nodes. A node that acts as an AERO Client on one AERO interface can also act as an AERO Server on a different AERO interface.

AERO Server ("Server")
: a node that configures an AERO interface to provide default forwarding services for AERO Clients. The Server assigns an administratively-provisioned IPv6 link-local address to the AERO interface to support the operation of the ND/PD services. An AERO Server can also act as an AERO Relay.

AERO Relay ("Relay")
: a node that configures an AERO interface to relay IP packets between nodes on the same AERO link and/or forward IP packets between the AERO link and the native Internetwork. The Relay

assigns an administratively-provisioned IPv6 link-local address to the AERO interface the same as for a Server. An AERO Relay can also act as an AERO Server.

AERO Proxy ("Proxy")
a node that provides proxying services for Clients that cannot associate directly with Servers, e.g., when the Client is located in a secured internal enclave and the Server is located in the external Internetwork. The AERO Proxy is a conduit between the secured enclave and the external Internetwork in the same manner as for common web proxies, and behaves in a similar fashion as for ND proxies [RFC4389].

ingress tunnel endpoint (ITE)
an AERO interface endpoint that injects encapsulated packets into an AERO link.

egress tunnel endpoint (ETE)
an AERO interface endpoint that receives encapsulated packets from an AERO link.

underlying network
the same as defined for Internetwork.

underlying link
a link that connects an AERO node to the underlying network.

underlying interface
an AERO node’s interface point of attachment to an underlying link.

link-layer address
an IP address assigned to an AERO node’s underlying interface. When UDP encapsulation is used, the UDP port number is also considered as part of the link-layer address. Packets transmitted over an AERO interface use link-layer addresses as encapsulation header source and destination addresses. Destination link-layer addresses can be either "reachable" or "unreachable" based on dynamically-changing network conditions.

network layer address
the source or destination address of an encapsulated IP packet.

end user network (EUN)
an internal virtual or external edge IP network that an AERO Client connects to the rest of the network via the AERO interface. The Client sees each EUN as a "downstream" network and sees the

AERO interface as its point of attachment to the "upstream" network.

AERO Service Prefix (ASP)
: an IP prefix associated with the AERO link and from which more-specific AERO Client Prefixes (ACPs) are derived.

AERO Client Prefix (ACP)
: an IP prefix derived from an ASP and delegated to a Client, where the ACP prefix length must be no shorter than the ASP prefix length and must be no longer than 64 for IPv6 or 32 for IPv4.

base AERO address
: the lowest-numbered AERO address from the first ACP delegated to the Client (see Section 3.4).

Throughout the document, the simple terms "Client", "Server", "Relay" and "Proxy" refer to "AERO Client", "AERO Server", "AERO Relay" and "AERO Proxy", respectively. Capitalization is used to distinguish these terms from DHCPv6 client/server/relay [RFC3315].

The terminology of DHCPv6 [RFC3315][RFC3633] and IPv6 ND [RFC4861] (including the names of node variables, messages and protocol constants) is used throughout this document. Also, the term "IP" is used to generically refer to either Internet Protocol version, i.e., IPv4 [RFC0791] or IPv6 [RFC8200].

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]. Lower case uses of these words are not to be interpreted as carrying RFC2119 significance.

3. Asymmetric Extended Route Optimization (AERO)

The following sections specify the operation of IP over Asymmetric Extended Route Optimization (AERO) links:

3.1. AERO Link Reference Model

```
                              .-(::::::::)
                           .-(::::::::::::)-.
                          (:: Internetwork ::)
                           '-(::::::::::::)-'
                              '-(::::::)-'
                                   |
       +--------------+   +--------+-------+   +--------------+
       |AERO Server S1|   | AERO Relay R1  |   |AERO Server S2|
       |  Nbr: C1, R1 |   |   Nbr: S1, S2  |   |  Nbr: C2, R1 |
       |  default->R1 |   |(X1->S1; X2->S2)|   |  default->R1 |
       |    X1->C1    |   |      ASP A1    |   |    X2->C2    |
       +-------+------+   +--------+-------+   +------+-------+
               |    AERO Link      |                  |
       X---+---+-------------------+-+----------------+---+---X
           |                         |                    |
     +-----+--------+     +----------+------+    +--------+-----+
     |AERO Client C1|     |  AERO Proxy P1  |    |AERO Client C2|
     |    Nbr: S1   |     |(Proxy Nbr Cache)|    |   Nbr: S2    |
     | default->S1  |     +--------+--------+    | default->S2  |
     |    ACP X1    |              |             |    ACP X2    |
     +------+-------+     .--------+------.      +-----+--------+
            |           (- Proxyed Clients -)          |
           .-.            '---------------'           .-.
        ,-(  _)-.                                  ,-(  _)-.
     .-(_  IP   )-.   +-------+     +-------+    .-(_  IP   )-.
   (__      EUN      )--|Host H1|     |Host H2|--(__      EUN      )
      '-(______)-'    +-------+     +-------+     '-(______)-'
```

Figure 1: AERO Link Reference Model

Figure 1 presents the AERO link reference model. In this model:

- o AERO Relay R1 aggregates AERO Service Prefix (ASP) A1, acts as a default router for its associated Servers (S1 and S2), and connects the AERO link to the rest of the Internetwork.

- o AERO Servers S1 and S2 associate with Relay R1 and also act as default routers for their associated Clients C1 and C2.

- o AERO Clients C1 and C2 associate with Servers S1 and S2, respectively. They receive AERO Client Prefix (ACP) delegations X1 and X2, and also act as default routers for their associated physical or internal virtual EUNs. Simple hosts H1 and H2 attach to the EUNs served by Clients C1 and C2, respectively.

- o AERO Proxy P1 provides proxy services for AERO Clients in secured enclaves that cannot associate directly with other AERO link neighbors.

Each node on the AERO link maintains an AERO interface neighbor cache and an IP forwarding table the same as for any link. Although the figure shows a limited deployment, in common operational practice there may be many additional Relays, Servers, Clients and Proxies.

3.2. AERO Node Types

AERO Relays provide default forwarding services to AERO Servers. Each Relay also peers with Servers and other Relays in a dynamic routing protocol instance to discover the list of active ACPs (see Section 3.3). Relays forward packets between neighbors connected to the same AERO link and also forward packets between the AERO link and the native Internetwork. Relays present the AERO link to the native Internetwork as a set of one or more AERO Service Prefixes (ASPs) and serve as a gateway between the AERO link and the Internetwork. Relays maintain AERO interface neighbor cache entries for Servers, and maintain an IP forwarding table entry for each AERO Client Prefix (ACP). AERO Relays can also be configured to act as AERO Servers.

AERO Servers provide default forwarding services to AERO Clients. Each Server also peers with Relays in a dynamic routing protocol instance to advertise its list of associated ACPs (see Section 3.3). Servers facilitate PD exchanges with Clients, where each delegated prefix becomes an ACP taken from an ASP. Servers forward packets between AERO interface neighbors, and maintain AERO interface neighbor cache entries for Relays. They also maintain both neighbor cache entries and IP forwarding table entries for each of their associated Clients. AERO Servers can also be configured to act as AERO Relays.

AERO Clients act as requesting routers to receive ACPs through PD exchanges with AERO Servers over the AERO link. Each Client can associate with a single Server or with multiple Servers, e.g., for fault tolerance, load balancing, etc. Each IPv6 Client receives at least a /64 IPv6 ACP, and may receive even shorter prefixes. Similarly, each IPv4 Client receives at least a /32 IPv4 ACP (i.e., a singleton IPv4 address), and may receive even shorter prefixes. Clients maintain an AERO interface neighbor cache entry for each of their associated Servers as well as for each of their correspondent Clients.

AERO Proxies provide a conduit for AERO Clients connected to secured enclaves to associate with AERO link Servers. The Proxy can either be explicit or transparent. In the explicit case, the Client sends all of its control plane messages addressed to the Server to the link-layer address of the Proxy. In the transparent case, the Client sends all of its control plane messages to the Server’s link-layer address and the Proxy intercepts them before they leave the secured enclave. In both cases, the Proxy forwards the Client's control and data plane messages to and from the Client's current Server(s). The Proxy may also discover a more direct route toward a target destination via AERO route optimization, in which case future outbound data packets would be forwarded via the more direct route. The Proxy function is specified in Section 4.

3.3. AERO Routing System

The AERO routing system comprises a private instance of the Border Gateway Protocol (BGP) [RFC4271] that is coordinated between Relays and Servers and does not interact with either the public Internet BGP routing system or the native Internetwork routing system. Relays advertise only a small and unchanging set of ASPs to the native Internetwork routing system instead of the full dynamically changing set of ACPs.

In a reference deployment, each AERO Server is configured as an Autonomous System Border Router (ASBR) for a stub Autonomous System (AS) using an AS Number (ASN) that is unique within the BGP instance, and each Server further uses eBGP to peer with one or more Relays but does not peer with other Servers. All Relays are members of the same hub AS using a common ASN, and use iBGP to maintain a consistent view of all active ACPs currently in service.

Each Server maintains a working set of associated ACPs, and dynamically announces new ACPs and withdraws departed ACPs in its eBGP updates to Relays. Clients are expected to remain associated with their current Servers for extended timeframes, however Servers SHOULD selectively suppress updates for impatient Clients that repeatedly associate and disassociate with them in order to dampen routing churn.

Each Relay configures a black-hole route for each of its ASPs. By black-holing the ASPs, the Relay will maintain forwarding table entries only for the ACPs that are currently active, and packets destined to all other ACPs will correctly incur Destination Unreachable messages due to the black hole route. Relays do not send eBGP updates for ACPs to Servers, but instead only originate a default route. In this way, Servers have only partial topology knowledge (i.e., they know only about the ACPs of their directly associated Clients) and they forward all other packets to Relays which have full topology knowledge.

Scaling properties of the AERO routing system are limited by the number of BGP routes that can be carried by Relays. At the time of this writing, the global public Internet BGP routing system manages more than 500K routes with linear growth and no signs of router resource exhaustion [BGP]. Network emulation studies have also shown that a single Relay can accommodate at least 1M dynamically changing BGP routes even on a lightweight virtual machine, i.e., and without requiring high-end dedicated router hardware.

Therefore, assuming each Relay can carry 1M or more routes, this means that at least 1M Clients can be serviced by a single set of Relays. A means of increasing scaling would be to assign a different set of Relays for each set of ASPs. In that case, each Server still peers with one or more Relays, but the Server institutes route filters so that it only sends BGP updates to the specific set of Relays that aggregate the ASP. For example, if the ASP for the AERO link is 2001:db8::/32, a first set of Relays could service the ASP segment 2001:db8::/40, a second set of Relays could service 2001:db8:0100::/40, a third set could service 2001:db8:0200::/40, etc.

Assuming up to 1K sets of Relays, the AERO routing system can then accommodate 1B or more ACPs with no additional overhead for Servers and Relays (for example, it should be possible to service 1B /64 ACPs taken from a /34 ASP and even more for shorter prefixes). In this way, each set of Relays services a specific set of ASPs that they advertise to the native Internetwork routing system, and each Server configures ASP-specific routes that list the correct set of Relays as next hops. This arrangement also allows for natural incremental deployment, and can support small scale initial deployments followed by dynamic deployment of additional Clients, Servers and Relays without disturbing the already-deployed base.

Note that in an alternate routing arrangement each set of Relays could advertise an aggregated ASP for the link into the native Internetwork routing system even though each Relay services only smaller segments of the ASP. In that case, a Relay upon receiving a packet with a destination address covered by the ASP segment of another Relay can simply tunnel the packet to the other Relay. The tradeoff then is the penalty for Relay-to-Relay tunneling compared with reduced routing information in the native routing system.

A full discussion of the BGP-based routing system used by AERO is found in [I-D.templin-atn-bgp].

3.4. AERO Interface Link-local Addresses

AERO interface link-local address types include administratively-provisioned addresses and AERO addresses.

Administratively-provisioned addresses are allocated from the range fe80::/96 and assigned to a Server or Relay’s AERO interface.

Administratively-provisioned addresses MUST be managed for uniqueness by the administrative authority for the AERO link. The address fe80:: is reserved as the IPv6 link-local subnet router anycast address, and the address fe80::ffff:ffff is reserved as the "prefix-solicitation" address used by Clients to bootstrap AERO address autoconfiguration. These reserved addresses are therefore not available for general assignment.

An AERO address is an IPv6 link-local address with an embedded prefix based on an ACP and associated with a Client's AERO interface. AERO addresses remain stable as the Client moves between topological locations, i.e., even if its link-layer addresses change.

For IPv6, AERO addresses begin with the prefix fe80::/64 and include in the interface identifier (i.e., the lower 64 bits) a 64-bit prefix taken from one of the Client's IPv6 ACPs. For example, if the AERO Client receives the IPv6 ACP:

```
   2001:db8:1000:2000::/56
```

it constructs its corresponding AERO addresses as:

```
   fe80::2001:db8:1000:2000

   fe80::2001:db8:1000:2001

   fe80::2001:db8:1000:2002

   ... etc. ...

   fe80::2001:db8:1000:20ff
```

For IPv4, AERO addresses are based on an IPv4-mapped IPv6 address [RFC4291] formed from an IPv4 ACP and with a Prefix Length of 96 plus the ACP prefix length. For example, for the IPv4 ACP 192.0.2.32/28 the IPv4-mapped IPv6 ACP is:

```
   0:0:0:0:0:FFFF:192.0.2.16/124
```

The Client then constructs its AERO addresses with the prefix fe80::/64 and with the lower 64 bits of the IPv4-mapped IPv6 address in the interface identifier as:

```
   fe80::FFFF:192.0.2.16

   fe80::FFFF:192.0.2.17

   fe80::FFFF:192.0.2.18
```

```
      ... etc. ...

      fe80:FFFF:192.0.2.31
```

When the Server delegates ACPs to the Client, both the Server and Client use the lowest-numbered AERO address from the first ACP delegation as the "base" AERO address (for example, for the ACP 2001:db8:1000:2000::/56 the base AERO address is fe80::2001:db8:1000:2000). The Client then assigns the base AERO address to the AERO interface and uses it for the purpose of maintaining the neighbor cache entry. The Server likewise uses the AERO address as its index into the neighbor cache for this Client.

If the Client has multiple AERO addresses (i.e., when there are multiple ACPs and/or ACPs with short prefix lengths), the Client originates ND messages using the base AERO address as the source address and accepts and responds to ND messages destined to any of its AERO addresses as equivalent to the base AERO address. In this way, the Client maintains a single neighbor cache entry that may be indexed by multiple AERO addresses.

3.5. AERO Interface Characteristics

AERO interfaces use encapsulation (see: Section 3.9) to exchange packets with neighbors attached to the AERO link.

AERO interfaces maintain a neighbor cache for tracking per-neighbor state the same as for any interface. AERO interfaces use ND messages including Neighbor Solicitation (NS), Neighbor Advertisement (NA), Router Solicitation (RS), Router Advertisement (RA) and Redirect for neighbor cache management. AERO interfaces use RS/RA messages with an embedded PD message (e.g., see: [I-D.templin-6man-dhcpv6-ndopt]). AERO interfaces include routing information in ND messages to support route optimization.

AERO interface ND messages include one or more Source/Target Link-Layer Address Options (S/TLLAOs) formatted as shown in Figure 2:

```
        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     |   Length = 5  |X|         Reserved            |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |          Interface ID         |        UDP Port Number        |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                                                               |
       +                                                               +
       |                                                               |
       +                          IP Address                           +
       |                                                               |
       +                                                               +
       |                                                               |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |P00|P01|P02|P03|P04|P05|P06|P07|P08|P09|P10|P11|P12|P13|P14|P15|
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |P16|P17|P18|P19|P20|P21|P22|P23|P24|P25|P26|P27|P28|P29|P30|P31|
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |P32|P33|P34|P35|P36|P37|P38|P39|P40|P41|P42|P43|P44|P45|P46|P47|
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |P48|P49|P50|P51|P52|P53|P54|P55|P56|P57|P58|P59|P60|P61|P62|P63|
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: AERO Source/Target Link-Layer Address Option (S/TLLAO) Format

In this format:

- o Type is set to ’1’ for SLLAO or ’2’ for TLLAO.
- o Length is set to the constant value ’5’ (i.e., 5 units of 8 octets).
- o X (proXy) is set to ’1’ in an S/TLLAO if the address corresponds to a Proxy; otherwise, X is set to ’0’.
- o Reserved is set to the value ’0’ on transmission and ignored on receipt.
- o Interface ID is set to a 16-bit integer value corresponding to an underlying interface of the AERO node. The value 255 is reserved for Server-based route optimization (see: Section 3.15.8).
- o UDP Port Number and IP Address are set to the addresses used by the AERO node when it sends encapsulated packets over the specified underlying interface (or to ’0’ when the addresses are left unspecified). When UDP is not used as part of the

encapsulation, UDP Port Number is set to ’0’. When the encapsulation IP address family is IPv4, IP Address is formed as an IPv4-mapped IPv6 address as specified in Section 3.4.

- o P(i) is a set of 64 Preference values that correspond to the 64 Differentiated Service Code Point (DSCP) values [RFC2474]. Each P(i) is set to the value ’0’ ("disabled"), ’1’ ("low"), ’2’ ("medium") or ’3’ ("high") to indicate a QoS preference level for packet forwarding purposes.

AERO interfaces may be configured over multiple underlying interface connections to underlying links. For example, common mobile handheld devices have both wireless local area network ("WLAN") and cellular wireless links. These links are typically used "one at a time" with low-cost WLAN preferred and highly-available cellular wireless as a standby. In a more complex example, aircraft frequently have many wireless data link types (e.g. satellite-based, cellular, terrestrial, air-to-air directional, etc.) with diverse performance and cost properties.

A Client’s underlying interfaces are classified as follows:

- o Native interfaces connect to the open Internetwork, and have a global IP address that is reachable from any open Internetwork correspondent.

- o NAT’ed interfaces connect to a closed network that is separated from the open Internetwork by a Network Address Translator (NAT). The NAT does not participate in any AERO control message signaling, but the AERO Server can issue AERO control messages on behalf of the Client.

- o VPN’ed interfaces use security encapsulation over the Internetwork to a Virtual Private Network (VPN) gateway that also acts as an AERO Server. As with NAT’ed links, the AERO Server can issue control messages on behalf of the Client.

- o Proxy’ed interfaces connect to a closed network that is separated from the open Internetwork by an AERO Proxy. Unlike NAT’ed and VPN’ed interfaces, the AERO Proxy (rather than the Server) can issue control message on behalf of the Client.

- o Direct interfaces connect the Client directly to a peer without crossing any networked paths. An example is a line-of-sight link between a remote pilot and an unmanned aircraft.

If a Client’s multiple underlying interfaces are used "one at a time" (i.e., all other interfaces are in standby mode while one interface is active), then ND messages include only a single S/TLLAO with Interface ID set to a constant value. In that case, the Client would appear to have a single underlying interface but with a dynamically changing link-layer address.

If the Client has multiple active underlying interfaces, then from the perspective of ND it would appear to have multiple link-layer addresses. In that case, ND messages MAY include multiple S/TLLAOs -- each with an Interface ID that corresponds to a specific underlying interface of the AERO node.

When the Client includes an S/TLLAO for an underlying interface for which it is aware that there is a NAT or Proxy on the path to the Server, or when a node includes an S/TLLAO solely for the purpose of announcing new QoS preferences, the node sets both UDP Port Number and IP Address to 0 to indicate that the addresses are unspecified.

When an ND message includes multiple S/TLLAOs, the first S/TLLAO MUST correspond to the AERO node’s underlying interface used to transmit the message.

3.6. AERO Interface Initialization

3.6.1. AERO Relay Behavior

When a Relay enables an AERO interface, it first assigns an administratively-provisioned link-local address fe80::ID to the interface. Each fe80::ID address MUST be unique among all AERO nodes on the link. The Relay then engages in a dynamic routing protocol session with one or more Servers and all other Relays on the link (see: Section 3.3), and advertises its assigned ASPs into the native Internetwork.

Each Relay subsequently maintains an IP forwarding table entry for each active ACP covered by its ASP(s), and maintains neighbor cache entries for all Servers on the link. Relays exchange NS/NA messages with AERO link neighbors the same as for any AERO node. However, Neighbor Unreachability Detection (NUD) (see: Section 3.16) is optional since the dynamic routing protocol already provides reachability confirmation.

3.6.2. AERO Server Behavior

When a Server enables an AERO interface, it assigns an administratively-provisioned link-local address fe80::ID the same as for Relays. The Server further configures a service to facilitate PD exchanges with AERO Clients. The Server maintains neighbor cache entries for one or more Relays on the link, and manages per-Client neighbor cache entries and IP forwarding table entries based on control message exchanges. Each Server also engages in a dynamic routing protocol with their neighboring Relays (see: Section 3.3).

When the Server receives an NS/RS message from a Client on the AERO interface it authenticates the message and returns an NA/RA message. The Server further provides a simple link-layer conduit between AERO interface neighbors. In particular, when a packet sent by a source Client arrives on the Server’s AERO interface and is destined to another AERO node, the Server forwards the packet from within the AERO interface driver at the link layer without ever disturbing the network layer.

3.6.3. AERO Client Behavior

When a Client enables an AERO interface, it sends RS messages with PD "Solicit" options over an underlying interface using the prefix-solicitation address as the source network layer address and all-routers [RFC4861] as the destination network layer address to obtain ACPs from one or more AERO Servers. Each Server processes the message and returns an RA message with a PD "Reply" option with the Server’s link-layer address as the source and the base AERO address as the destination network layer addresses. In this way, the ND/PD control messages securely perform all autoconfiguration operations in a single request/response exchange.

After the initial ND/PD message exchange, the Client can register additional underlying interfaces with the Server by sending an RS message over each underlying interface using its base AERO address as the source network layer address and without including a PD option. The Server will update its neighbor cache entry for the Client and return an RA message.

The Client maintains a neighbor cache entry for each of its Servers and each of its active correspondent Clients. When the Client receives ND messages on the AERO interface it updates or creates neighbor cache entries, including link-layer address and QoS preferences.

3.6.4. AERO Proxy Behavior

When a Proxy enables an AERO interface, it maintains per-Client proxy neighbor cache entries based on control message exchanges. Proxies forward packets between their associated Clients and the Clients’ associated Servers.

When the Proxy receives an RS message from a Client in the secured enclave, it creates an incomplete proxy neighbor cache entry and forwards the message to a Server selected by the Client while using its own link-layer address as the source address. When the Server returns an RA message, the Proxy completes the proxy neighbor cache entry based on autoconfiguration information in the RA and forwards the RA to the Client while using its own link-layer address as the source address. The Client, Server and Proxy will then have the necessary state for managing the proxyed neighbor association.

3.7. AERO Interface Neighbor Cache Maintenance

Each AERO interface maintains a conceptual neighbor cache that includes an entry for each neighbor it communicates with on the AERO link, the same as for any IPv6 interface [RFC4861]. AERO interface neighbor cache entries are said to be one of "permanent", "static", "proxy" or "dynamic".

Permanent neighbor cache entries are created through explicit administrative action; they have no timeout values and remain in place until explicitly deleted. AERO Relays maintain permanent neighbor cache entries for Servers on the link, and AERO Servers maintain permanent neighbor cache entries for Relays. Each entry maintains the mapping between the neighbor’s fe80::ID network-layer address and corresponding link-layer address.

Static neighbor cache entries are created and maintained through ND/PD exchanges as specified in Section 3.14, and remain in place for durations bounded by ND/PD lifetimes. AERO Servers maintain static neighbor cache entries for each of their associated Clients, and AERO Clients maintain static neighbor cache entries for each of their associated Servers.

Proxy neighbor cache entries are created and maintained by AERO Proxies by gleaning information from Client/Server ND/PD exchanges, and remain in place for durations bounded by ND/PD lifetimes. AERO Proxies maintain proxy neighbor cache entries for each of their associated Clients, and include pointers to the Client’s current set of Servers.

Dynamic neighbor cache entries are created or updated based on receipt of route optimization messages as specified in Section 3.15, and are garbage-collected when keepalive timers expire. AERO nodes maintain dynamic neighbor cache entries for each of their active correspondents with lifetimes based on ND messaging constants.

When a target AERO node receives a valid NS message with an AERO source address, it returns an NA message and also creates or updates a dynamic neighbor cache entry for the source network-layer and link-layer addresses. The node then sets an "AcceptTime" variable in the neighbor cache entry to ACCEPT_TIME seconds and uses this value to determine whether packets received from the correspondent can be accepted. The node resets AcceptTime when it receives a new ND message, and otherwise decrements AcceptTime while no ND messages have been received. It is RECOMMENDED that ACCEPT_TIME be set to the default constant value 40 seconds to allow a 10 second window so that the AERO route optimization procedure can converge before AcceptTime decrements below FORWARD_TIME (see below).

When a source AERO node receives a valid NA message with an AERO source address that matches its NS message, it creates or updates a dynamic neighbor cache entry for the target network-layer and link-layer addresses. The node then sets a "ForwardTime" variable in the neighbor cache entry to FORWARD_TIME seconds and uses this value to determine whether packets can be forwarded directly to the correspondent, i.e., instead of via a default route. The node resets ForwardTime when it receives a new NA, and otherwise decrements ForwardTime while no further NA messages have been received. It is RECOMMENDED that FORWARD_TIME be set to the default constant value 30 seconds to match the default REACHABLE_TIME value specified in [RFC4861].

The node also sets a "MaxRetry" variable to MAX_RETRY to limit the number of keepalives sent when a correspondent may have gone unreachable. It is RECOMMENDED that MAX_RETRY be set to 3 the same as described for address resolution in Section 7.3.3 of [RFC4861].

Different values for ACCEPT_TIME, FORWARD_TIME and MAX_RETRY MAY be administratively set, if necessary, to better match the AERO link’s performance characteristics; however, if different values are chosen, all nodes on the link MUST consistently configure the same values. Most importantly, ACCEPT_TIME SHOULD be set to a value that is sufficiently longer than FORWARD_TIME to allow the AERO route optimization procedure to converge.

When there may be a NAT between the Client and the Server, or if the path from the Client to the Server should be tested for reachability, the Client can send periodic RS messages to the Server without a PD option to receive RA replies. The RS/RA messaging will keep NAT state alive and test Server reachability without disturbing the PD service.

3.8. AERO Interface Forwarding Algorithm

IP packets enter a node’s AERO interface either from the network layer (i.e., from a local application or the IP forwarding system) or from the link layer (i.e., from the AERO tunnel virtual link). Packets that enter the AERO interface from the network layer are encapsulated and forwarded into the AERO link, i.e., they are tunneled to an AERO interface neighbor. Packets that enter the AERO interface from the link layer are either re-admitted into the AERO link or forwarded to the network layer where they are subject to either local delivery or IP forwarding. In all cases, the AERO interface itself MUST NOT decrement the network layer TTL/Hop-count since its forwarding actions occur below the network layer.

AERO interfaces may have multiple underlying interfaces and/or neighbor cache entries for neighbors with multiple Interface ID registrations (see Section 3.5). The AERO node uses each packet’s DSCP value to select an outgoing underlying interface based on the node’s own QoS preferences, and also to select a destination link-layer address based on the neighbor’s underlying interface with the highest preference. If multiple outgoing interfaces and/or neighbor interfaces have a preference of "high", the AERO node sends one copy of the packet via each of the (outgoing / neighbor) interface pairs; otherwise, the node sends a single copy of the packet via the interface with the highest preference. AERO nodes keep track of which underlying interfaces are currently "reachable" or "unreachable", and only use "reachable" interfaces for forwarding purposes.

The following sections discuss the AERO interface forwarding algorithms for Clients, Proxies, Servers and Relays. In the following discussion, a packet’s destination address is said to "match" if it is a non-link-local address with a prefix covered by an ASP/ACP, or if it is an AERO address that embeds an ACP, or if it is the same as an administratively-provisioned link-local address.

3.8.1. Client Forwarding Algorithm

When an IP packet enters a Client’s AERO interface from the network layer the Client searches for a dynamic neighbor cache entry that matches the destination. If there is a match, the Client uses one or more "reachable" link-layer addresses in the entry as the link-layer addresses for encapsulation and admits the packet into the AERO link. Otherwise, the Client uses the link-layer address in a static neighbor cache entry for a Server as the encapsulation address (noting that there may be a Proxy on the path to the real Server).

When an IP packet enters a Client’s AERO interface from the link-layer, if the destination matches one of the Client’s ACPs or link-local addresses the Client decapsulates the packet and delivers it to the network layer. Otherwise, the Client drops the packet and MAY return a network-layer ICMP Destination Unreachable message subject to rate limiting (see: Section 3.13).

3.8.2. Proxy Forwarding Algorithm

When the Proxy receives a packet from a Client within the secured enclave, the Proxy searches for a dynamic neighbor cache entry that matches the destination. If there is a match, the Proxy uses one or more "reachable" link-layer addresses in the entry as the link-layer addresses for encapsulation and admits the packet into the AERO link. Otherwise, the Proxy uses the link-layer address for one of the Client’s Servers as the encapsulation address.

When the Proxy receives a packet from an AERO interface neighbor, it searches for a proxy neighbor cache entry for a Client within the secured enclave that matches the destination. If there is a match, the Proxy forwards the packet to the Client. Otherwise, the Proxy returns the packet to the neighbor, i.e., by reversing the source and destination link-layer addresses.

3.8.3. Server Forwarding Algorithm

When an IP packet enters a Server’s AERO interface from the network layer, the Server searches for a static neighbor cache entry for a Client that matches the destination. If there is a match, the Server uses one or more link-layer addresses in the entry as the link-layer addresses for encapsulation and admits the packet into the AERO link. Otherwise, the Server uses the link-layer address in a permanent neighbor cache entry for a Relay (selected through longest-prefix match) as the link-layer address for encapsulation.

When an IP packet enters a Server’s AERO interface from the link layer, the Server processes the packet according to the network-layer destination address as follows:

- o if the destination matches one of the Server’s own addresses the Server decapsulates the packet and forwards it to the network layer for local delivery.

- o else, if the destination matches a static neighbor cache entry for a Client the Server first determines whether the neighbor is the same as the one it received the packet from. If so, the Server drops the packet silently to avoid looping; otherwise, the Server uses the neighbor’s link-layer address(es) as the destination for encapsulation and re-admits the packet into the AERO link.

- o else, the Server uses the link-layer address in a neighbor cache entry for a Relay (selected through longest-prefix match) as the link-layer address for encapsulation.

3.8.4. Relay Forwarding Algorithm

When an IP packet enters a Relay’s AERO interface from the network layer, the Relay searches its IP forwarding table for an ACP entry that matches the destination and otherwise searches for a neighbor cache entry that matches the destination (e.g., for administratively-provisioned link-local addresses). If there is a match, the Relay uses the link-layer address in the corresponding neighbor cache entry as the link-layer address for encapsulation and forwards the packet into the AERO link. Otherwise, the Relay drops the packet and (for non-link-local addresses) returns a network-layer ICMP Destination Unreachable message subject to rate limiting (see: Section 3.13).

When an IP packet enters a Relay’s AERO interface from the link-layer, the Relay processes the packet as follows:

- o if the destination does not match an ASP, or if the destination matches one of the Relay’s own addresses, the Relay decapsulates the packet and forwards it to the network layer where it will be subject to either IP forwarding or local delivery.

- o else, if the destination matches an ACP entry in the IP forwarding table, or if the destination matches the link-local address in a permanent neighbor cache entry, the Relay first determines whether the neighbor is the same as the one it received the packet from. If so the Relay MUST drop the packet silently to avoid looping; otherwise, the Relay uses the neighbor’s link-layer address as the destination for encapsulation and re-admits the packet into the AERO link.

- o else, the Relay drops the packet and (for non-link-local addresses) returns an ICMP Destination Unreachable message subject to rate limiting (see: Section 3.13).

3.8.5. Processing Return Packets

When an AERO node receives a return packet such as generated by an AERO Proxy (see Section 3.8.2), it proceeds according to the AERO link trust basis. Namely, the return packets have the same trust profile as for link-layer Destination Unreachable messages. If the node has sufficient trust basis to accept link-layer Destination Unreachable messages, it can then process the return packet as described in the following paragraph. Otherwise, the node SHOULD drop the packet and treat it as an indication that a path may be failing, and MAY use NUD to test the path for reachability.

If the node has sufficient trust basis to accept return packets, it searches for a dynamic neighbor cache entry that matches the destination. If there is a match, the neighbor marks the corresponding link-layer address as "unreachable", selects the next-highest priority "reachable" link-layer address in the entry as the link-layer address for encapsulation then (re)admits the packet into the AERO link. If there are no "reachable" link-layer addresses, the neighbor instead sets FowardTime in the dynamic neighbor cache entry to 0. If the source address corresponds to one of the neighbor’s own addresses, the neighbor also forwards the packet to the corresponding Server; otherwise, it drops the packet.

3.9. AERO Interface Encapsulation and Re-encapsulation

AERO interfaces encapsulate IP packets according to whether they are entering the AERO interface from the network layer or if they are being re-admitted into the same AERO link they arrived on. This latter form of encapsulation is known as "re-encapsulation".

The AERO interface encapsulates packets per the Generic UDP Encapsulation (GUE) procedures in [I-D.ietf-intarea-gue][I-D.ietf-intarea-gue-extensions], or through an alternate encapsulation format (see: Appendix A). For packets entering the AERO interface from the network layer, the AERO interface copies the "TTL/Hop Limit", "Type of Service/Traffic Class" [RFC2983], "Flow Label"[RFC6438] (for IPv6) and "Congestion Experienced" [RFC3168] values in the packet’s IP header into the corresponding fields in the encapsulation IP header. For packets undergoing re-encapsulation, the AERO interface instead copies these values from the original encapsulation IP header into the new encapsulation header, i.e., the values are transferred between encapsulation headers and *not* copied from the encapsulated packet’s network-layer header. (Note especially that by copying the TTL/Hop Limit between encapsulation headers the value will eventually decrement to 0 if there is a (temporary) routing loop.) For IPv4 encapsulation/re-encapsulation, the AERO interface sets the DF bit as discussed in Section 3.12.

When GUE encapsulation is used, the AERO interface next sets the UDP source port to a constant value that it will use in each successive packet it sends, and sets the UDP length field to the length of the encapsulated packet plus 8 bytes for the UDP header itself plus the length of the GUE header (or 0 if GUE direct IP encapsulation is used). For packets sent to a Server or Relay, the AERO interface sets the UDP destination port to 8060, i.e., the IANA-registered port number for AERO. For packets sent to a Client, the AERO interface sets the UDP destination port to the port value stored in the neighbor cache entry for this Client. The AERO interface then either includes or omits the UDP checksum according to the GUE specification.

Clients normally use the IP address of the underlying interface as the encapsulation source address. If the underlying interface does not have an IP address, however, the Client uses an IP address taken from an ACP as the encapsulation source address (assuming the node has some way of injecting the ACP into the underlying network routing system). For IPv6 addresses, the Client normally uses the ACP Subnet Router Anycast address [RFC4291].

3.10. AERO Interface Decapsulation

AERO interfaces decapsulate packets destined either to the AERO node itself or to a destination reached via an interface other than the AERO interface the packet was received on. Decapsulation is per the procedures specified for the appropriate encapsulation format.

3.11. AERO Interface Data Origin Authentication

AERO nodes employ simple data origin authentication procedures for encapsulated packets they receive from other nodes on the AERO link. In particular:

o AERO Relays and Servers accept encapsulated packets with a link-layer source address that matches a permanent neighbor cache entry.

o AERO Servers accept authentic encapsulated ND messages from Clients, and create or update a static neighbor cache entry for the Client based on the specific message type.

o AERO Clients and Servers accept encapsulated packets if there is a static neighbor cache entry with a link-layer address that matches the packet’s link-layer source address.

o AERO Clients and Servers accept encapsulated packets if there is a dynamic neighbor cache entry with an AERO address that matches the packet’s network-layer source address, with a link-layer address that matches the packet’s link-layer source address, and with a non-zero AcceptTime.

o AERO Proxies accept encapsulated packets if there is a proxy neighbor cache entry that matches the packet’s network-layer destination address (i.e., the address of the Client) and link-layer source address (i.e., the address of one of the Client’s Servers). When the proxy is configured to accept packets originating from any address in the open Internetwork however (e.g., from another Proxy), it omits the source address check.

Note that this simple data origin authentication is effective in environments in which link-layer addresses cannot be spoofed. In other environments, each AERO message must include a signature that the recipient can use to authenticate the message origin, e.g., as for common VPN systems such as OpenVPN [OVPN]. In environments where end systems use end-to-end security, however, it may be sufficient to require signatures only for ND and ICMP control plane messages and omit signatures for data plane messages.

3.12. AERO Interface Packet Size Issues

The AERO interface is the node’s attachment to the AERO link. The AERO interface acts as a tunnel ingress when it sends a packet to an AERO link neighbor and as a tunnel egress when it receives a packet from an AERO link neighbor. AERO interfaces observe the packet sizing considerations for tunnels discussed in [I-D.ietf-intarea-tunnels] and as specified below.

The Internet Protocol expects that IP packets will either be delivered to the destination or a suitable Packet Too Big (PTB) message returned to support the process known as IP Path MTU Discovery (PMTUD) [RFC1191][RFC1981]. However, PTB messages may be crafted for malicious purposes such as denial of service, or lost in the network [RFC2923]. This can be especially problematic for tunnels, where a condition known as a PMTUD "black hole" can result. For these reasons, AERO interfaces employ operational procedures that avoid interactions with PMTUD, including the use of fragmentation when necessary.

AERO interfaces observe two different types of fragmentation. Source fragmentation occurs when the AERO interface (acting as a tunnel ingress) fragments the encapsulated packet into multiple fragments before admitting each fragment into the tunnel. Network fragmentation occurs when an encapsulated packet admitted into the tunnel by the ingress is fragmented by an IPv4 router on the path to the egress. Note that a packet that incurs source fragmentation may also incur network fragmentation.

IPv6 specifies a minimum link Maximum Transmission Unit (MTU) of 1280 bytes [RFC8200]. Although IPv4 specifies a smaller minimum link MTU of 68 bytes [RFC0791], AERO interfaces also observe the IPv6 minimum for IPv4 even if encapsulated packets may incur network fragmentation.

IPv6 specifies a minimum Maximum Reassembly Unit (MRU) of 1500 bytes [RFC8200], while the minimum MRU for IPv4 is only 576 bytes [RFC1122] (note that common IPv6 over IPv4 tunnels already assume a larger MRU than the IPv4 minimum).

AERO interfaces therefore configure an MTU that MUST NOT be smaller than 1280 bytes, MUST NOT be larger than the minimum MRU among all nodes on the AERO link minus the encapsulation overhead ("ENCAPS"), and SHOULD NOT be smaller than 1500 bytes. AERO interfaces also configure a Maximum Segment Unit (MSU) as the maximum-sized encapsulated packet that the ingress can inject into the tunnel without source fragmentation. The MSU value MUST NOT be larger than (MTU+ENCAPS) and MUST NOT be larger than 1280 bytes unless there is operational assurance that a larger size can traverse the link along all paths.

All AERO nodes MUST configure the same MTU/MSU values for reasons cited in [RFC3819][RFC4861]; in particular, multicast support requires a common MTU value among all nodes on the link. All AERO nodes MUST configure an MRU large enough to reassemble packets up to (MTU+ENCAPS) bytes in length; nodes that cannot configure a large-enough MRU MUST NOT enable an AERO interface.

The network layer proceeds as follow when it presents an IP packet to the AERO interface. For each IPv4 packet that is larger than the AERO interface MTU and with the DF bit set to 0, the network layer uses IPv4 fragmentation to break the packet into a minimum number of non-overlapping fragments where the first fragment is no larger than the MTU and the remaining fragments are no larger than the first. For all other IP packets, if the packet is larger than the AERO interface MTU, the network layer drops the packet and returns a PTB message to the original source. Otherwise, the network layer admits each IP packet or fragment into the AERO interface.

For each IP packet admitted into the AERO interface, the interface (acting as a tunnel ingress) encapsulates the packet. If the encapsulated packet is larger than the AERO interface MSU the ingress source-fragments the encapsulated packet into a minimum number of non-overlapping fragments where the first fragment is no larger than the MSU and the remaining fragments are no larger than the first. The ingress then admits each encapsulated packet or fragment into the tunnel, and for IPv4 sets the DF bit to 0 in the IP encapsulation header in case any network fragmentation is necessary. The encapsulated packets will be delivered to the egress, which reassembles them into a whole packet if necessary.

Several factors must be considered when fragmentation is needed. For AERO links over IPv4, the IP ID field is only 16 bits in length, meaning that fragmentation at high data rates could result in data corruption due to reassembly misassociations [RFC6864][RFC4963]. For AERO links over both IPv4 and IPv6, studies have also shown that IP fragments are dropped unconditionally over some network paths [I-D.taylor-v6ops-fragdrop]. In environments where IP fragmentation issues could result in operational problems, the ingress SHOULD employ intermediate-layer source fragmentation (see: [RFC2764] and [I-D.ietf-intarea-gue-extensions]) before appending the outer encapsulation headers to each fragment. Since the encapsulation fragment header reduces the room available for packet data, but the original source has no way to control its insertion, the ingress MUST include the fragment header length in the ENCAPS length even for packets in which the header is absent.

3.13. AERO Interface Error Handling

When an AERO node admits encapsulated packets into the AERO interface, it may receive link-layer or network-layer error indications.

A link-layer error indication is an ICMP error message generated by a router in the underlying network on the path to the neighbor or by the neighbor itself. The message includes an IP header with the address of the node that generated the error as the source address and with the link-layer address of the AERO node as the destination address.

The IP header is followed by an ICMP header that includes an error Type, Code and Checksum. Valid type values include "Destination Unreachable", "Time Exceeded" and "Parameter Problem" [RFC0792][RFC4443]. (AERO interfaces ignore all link-layer IPv4 "Fragmentation Needed" and IPv6 "Packet Too Big" messages since they only emit packets that are guaranteed to be no larger than the IP minimum link MTU as discussed in Section 3.12.)

The ICMP header is followed by the leading portion of the packet that generated the error, also known as the "packet-in-error". For ICMPv6, [RFC4443] specifies that the packet-in-error includes: "As much of invoking packet as possible without the ICMPv6 packet exceeding the minimum IPv6 MTU" (i.e., no more than 1280 bytes). For ICMPv4, [RFC0792] specifies that the packet-in-error includes: "Internet Header + 64 bits of Original Data Datagram", however [RFC1812] Section 4.3.2.3 updates this specification by stating: "the ICMP datagram SHOULD contain as much of the original datagram as possible without the length of the ICMP datagram exceeding 576 bytes".

The link-layer error message format is shown in Figure 3 (where, "L2" and "L3" refer to link-layer and network-layer, respectively):

```
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        ~                               ~
        |        L2 IP Header of        |
        |         error message         |
        ~                               ~
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |         L2 ICMP Header        |
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ---
        ~                               ~   P
        |   IP and other encapsulation  |   a
        | headers of original L3 packet |   c
        ~                               ~   k
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+   e
        ~                               ~   t
        |        IP header of           |
        |      original L3 packet       |   i
        ~                               ~   n
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        ~                               ~   e
        |    Upper layer headers and    |   r
        |    leading portion of body    |   r
        |   of the original L3 packet   |   o
        ~                               ~   r
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ---
```

Figure 3: AERO Interface Link-Layer Error Message Format

The AERO node rules for processing these link-layer error messages are as follows:

o When an AERO node receives a link-layer Parameter Problem message, it processes the message the same as described as for ordinary ICMP errors in the normative references [RFC0792][RFC4443].

o When an AERO node receives persistent link-layer Time Exceeded messages, the IP ID field may be wrapping before earlier fragments awaiting reassembly have been processed. In that case, the node SHOULD begin including integrity checks and/or institute rate limits for subsequent packets.

o When an AERO node receives persistent link-layer Destination Unreachable messages in response to encapsulated packets that it sends to one of its dynamic neighbor correspondents, the node SHOULD process the message as an indication that a path may be failing, and MAY initiate NUD over that path. If it receives Destination Unreachable messages on many or all paths, the node SHOULD set ForwardTime for the corresponding dynamic neighbor cache entry to 0 and allow future packets destined to the correspondent to flow through a default route.

- When an AERO Client receives persistent link-layer Destination Unreachable messages in response to encapsulated packets that it sends to one of its static neighbor Servers, the Client SHOULD math the path as unusable and use another path. If it receives Destination Unreachable messages on many or all paths, the Client SHOULD associate with a new Server and send a PD "Release" message to the old Server as specified in Section 3.17.6.

- When an AERO Server receives persistent link-layer Destination Unreachable messages in response to encapsulated packets that it sends to one of its static neighbor Clients, the Server SHOULD mark the path as unusable and use another path. If it receives Destination Unreachable messages on multiple paths, the Server should take no further actions unless it receives a PD "Release" message or if the PD lifetime expires. In that case, the Server MUST release the Client’s delegated ACP, withdraw the ACP from the AERO routing system and delete the neighbor cache entry.

- When an AERO Relay or Server receives link-layer Destination Unreachable messages in response to an encapsulated packet that it sends to one of its permanent neighbors, it treats the messages as an indication that the path to the neighbor may be failing. However, the dynamic routing protocol should soon reconverge and correct the temporary outage.

When an AERO Relay receives a packet for which the network-layer destination address is covered by an ASP, if there is no more-specific routing information for the destination the Relay drops the packet and returns a network-layer Destination Unreachable message subject to rate limiting. The Relay first writes the network-layer source address of the original packet as the destination address of the message and determines the next hop to the destination. If the next hop is reached via the AERO interface, the Relay uses the IPv6 address "::" or the IPv4 address "0.0.0.0" as the source address of the message, then encapsulates the message and forwards it to the next hop within the AERO interface. Otherwise, the Relay uses one of its non link-local addresses as the source address of the message and forwards it via a link outside the AERO interface.

When an AERO node receives an encapsulated packet for which the reassembly buffer it too small, it drops the packet and returns a network-layer Packet Too Big (PTB) message. The node first writes the MRU value into the PTB message MTU field, writes the network-layer source address of the original packet as the destination address of the message and determines the next hop to the destination. If the next hop is reached via the AERO interface, the node uses the IPv6 address "::" or the IPv4 address "0.0.0.0" as the source address of the message, then encapsulates the message and forwards it to the next hop within the AERO interface. Otherwise, the node uses one of its non link-local addresses as the source address of the message and forwards it via a link outside the AERO interface.

When an AERO node receives any network-layer error message via the AERO interface, it examines the network-layer destination address. If the next hop toward the destination is via the AERO interface, the node re-encapsulates and forwards the message to the next hop within the AERO interface. Otherwise, if the network-layer source address is the IPv6 address "::" or the IPv4 address "0.0.0.0", the node writes one of its non link-local addresses as the source address, recalculates the IP and/or ICMP checksums then forwards the message via a link outside the AERO interface.

3.14. AERO Router Discovery, Prefix Delegation and Autoconfiguration

AERO Router Discovery, Prefix Delegation and Autoconfiguration are coordinated as discussed in the following Sections.

3.14.1. AERO ND/PD Service Model

Each AERO Server configures a PD service to facilitate Client requests. Each Server is provisioned with a database of ACP-to-Client ID mappings for all Clients enrolled in the AERO system, as well as any information necessary to authenticate each Client. The Client database is maintained by a central administrative authority for the AERO link and securely distributed to all Servers, e.g., via the Lightweight Directory Access Protocol (LDAP) [RFC4511], via static configuration, etc. Therefore, no Server-to-Server PD state synchronization is necessary, and Clients can optionally hold separate PDs for the same ACPs from multiple Servers. In this way, Clients can associate with multiple Servers, and can receive new PDs from new Servers before releasing PDs received from existing Servers. This provides the Client with a natural fault-tolerance and/or load balancing profile.

AERO Clients and Servers use ND messages to maintain neighbor cache entries. AERO Servers configure their AERO interfaces as advertising interfaces, and therefore send unicast RA messages with configuration information in response to a Client's RS message. The RS/RA messaging is conducted in the same fashion as specified in [RFC5214].

AERO Clients and Servers include PD messages as options in the RS/RA messages they exchange (see: [I-D.templin-6man-dhcpv6-ndopt]).

Client-initiated PD options are included in RS messages, and Server-initiated PD options are included in RA messages. The unified ND/PD messages are exchanged between Client and Server according to the prefix management schedule determined by the PD service. The unified messages can be protected using SEcure Neighbor Discovery (SEND) [RFC3971].

On Some AERO links, PD arrangements may be through some out-of-band service such as network management, static configuration, etc. In those cases, AERO nodes can use simple RS/RA message exchanges with no explicit PD options. Instead, the RS/RA messages use AERO addresses as a means of representing the delegated prefixes, e.g., if a message includes a source address of "fe80::2001:db8:1:2" then the recipient can infer that the sender holds the prefix delegation "2001:db8:1:2::/64".

The following sections specify the Client and Server behavior.

3.14.2. AERO Client Behavior

AERO Clients discover the link-layer addresses of AERO Servers via static configuration (e.g., from a flat-file map of Server addresses and locations), or through an automated means such as Domain Name System (DNS) name resolution [RFC1035]. In the absence of other information, the Client resolves the DNS Fully-Qualfied Domain Name (FQDN) "linkupnetworks.[domainname]" where "linkupnetworks" is a constant text string and "[domainname]" is a DNS suffix for the Client's underlying interface (e.g., "example.com"). After discovering the link-layer addresses, the Client associates with one or more of the corresponding Servers.

To associate with a Server, the Client acts as a requesting router to request ACPs through a combined ND/PD message exchange. The Client includes a PD "Solicit" message as an ND option in an RS message with the prefix-solicitation address as the IPv6 source address, all-routers multicast as the IPv6 destination address, the address of the Client's underlying interface as the link-layer source address and the link-layer address of the Server as the link-layer destination address. (If the Client's underlying interface does not have an IP address, the Client can use the ACP Subnet Router Anycast address as the link-layer source address.)

The Client next includes a "Client Identifier" and an "IA_PD" (i.e., prefix request) code in the PD "Solicit" message. If the Client is pre-provisioned with ACPs associated with the AERO service, it MAY also include the ACPs in the "IA_PD" option to indicate its preferences to the Server. The Client finally includes any additional PD codes (e.g., "Rapid Commit").

The Client next includes one or more SLLAOs in the RS message formatted as described in Section 3.5 to register its link-layer address(es) with the Server. The first SLLAO MUST correspond to the underlying interface over which the Client will send the RS message. The Client MAY include additional SLLAOs specific to other underlying interfaces, but if so it MUST have assurance that there will be no NATs or Proxies on the paths to the Server via those interfaces. (Otherwise, the Client can register additional link-layer addresses with the Server by sending subsequent NS/RS messages via different underlying interfaces after the initial RS/RA exchange).

The Client then sends the RS message to the AERO Server and waits for an RA message reply (see Section 3.14.3) while retrying MAX_RETRY times until an RA is received. If no RA is received, or if it receives an RA with Router Lifetime set to 0 and/or a "Reply" with no ACPs, the Client SHOULD discontinue autoconfiguration attempts through this Server and try another Server. Otherwise, the Client processes the ACPs in the embedded "Reply" message.

Next, the Client creates a static neighbor cache entry with the Server's link-local address as the network-layer address and the Server's encapsulation source address as the link-layer address. The Client then autoconfigures AERO addresses for each of the delegated ACPs and assigns them to the AERO interface.

The Client next examines the P bit in the RA message flags field [RFC5175]. If the P bit value was 1, the Client assumes that there is a NAT or Proxy on the path to the Server via the interface over which it sent the RS message. In that case, the Client sets UDP Port Number and IP Address to 0 in the S/TLLAOs of any subsequent ND messages it sends to the Server over that link.

The Client also caches any ASPs included in Route Information Options (RIOs) [RFC4191] as ASPs to associate with the AERO link, and assigns the MTU/MSU values in the MTU options to its AERO interface while configuring an appropriate MRU. This configuration information applies to the AERO link as a whole, and all AERO nodes will receive the same values.

Following autoconfiguration, the Client sub-delegates the ACPs to its attached EUNs and/or the Client's own internal virtual interfaces as described in [I-D.templin-v6ops-pdhost]. The Client subsequently maintains its ACP delegations through each of its Servers by sending RS "Renew", "Rebind", and/or "Release" messages. The Server will in turn send RA "Reply" messages.

After the Client registers its Interface IDs and their associated UDP/IP addresses and 'P(i)' values, it may wish to change one or more

Interface ID registrations, e.g., if an underlying interface changes address or becomes unavailable, if QoS preferences change, etc. To do so, the Client prepares an unsolicited NA message to send over any available underlying interface. The source and target address of the NA message are set to the Client's AERO address, and the destination address is set to all-nodes multicast. The NA MUST include a TLLAO specific to the selected available underlying interface as the first TLLAO and MAY include any additional TLLAOs specific to other underlying interfaces. The Client includes fresh 'P(i)' values in each TLLAO to update the Server's neighbor cache entry. If the Client wishes to update 'P(i)' values without updating the link-layer address, it sets the UDP Port Number and IP Address fields to 0. If the Client wishes to disable the interface, it sets all 'P(i)' values to '0' ("disabled").

If the Client wishes to discontinue use of a Server it issues an RS "Release" message. When the Server processes the message, it releases the ACP, deletes its neighbor cache entry for the Client, withdraws the IP route from the routing system and returns an RA "Reply".

3.14.3. AERO Server Behavior

AERO Servers act as IPv6 routers and support a PD service on their AERO links. AERO Servers arrange to add their encapsulation layer IP addresses (i.e., their link-layer addresses) to a static map of Server addresses for the link and/or the DNS resource records for the FQDN "linkupnetworks.[domainname]" before entering service.

When an AERO Server receives a prospective Client's RS "Solicit" message on its AERO interface, and the Server is too busy, it SHOULD return an immediate RA "Reply" message with no ACPs and with Router Lifetime set to 0. Otherwise, the Server authenticates the RS message and processes the embedded "Solicit" option. The Server first determines the correct ACPs to delegate to the Client by searching the Client database. When the Server delegates the ACPs, it also creates an IP forwarding table entry for each ACP so that the AERO BGP-based routing system will propagate the ACPs to the Relays that aggregate the corresponding ASP (see: Section 3.3).

Next, the Server prepares an RA "Reply" message that includes the delegated ACPs. For IPv4 ACPs, the ACP is in IPv4-mapped IPv6 address format and with prefix length set as specified in Section 3.4. The Server then prepares an RA "Reply" message using its link-local address (i.e., fe80::ID) as the network-layer source address, the Client's base AERO address from the first ACP as the network-layer destination address, the Server's link-layer address as the source link-layer address, and the source link-layer address of the RS message as the destination link-layer address. The Server next sets the P flag in the RA message flags field [RFC5175] to 1 if the source link-layer address in the RS message was different than the address in the first SLLAO to indicate that there is a NAT or Proxy on the path; otherwise it sets P to 0. The Server then includes one or more RIOs that encode the ASPs for the AERO link. The Server also includes two MTU options - the first MTU option includes the MTU for the link and the second MTU option includes the MSU for the link (see Section 3.12). The Server finally sends the RA "Reply" message to the Client.

The Server next creates a static neighbor cache entry for the Client using the base AERO address as the network-layer address and with lifetime set to no more than the smallest PD lifetime. Next, the Server updates the neighbor cache entry link-layer address(es) by recording the information in each SLLAO option indexed by the Interface ID and including the UDP port number, IP address and P(i) values. For the first SLLAO in the list, however, the Server records the actual encapsulation source UDP and IP addresses instead of those that appear in the SLLAO in case there was a NAT or Proxy in the path.

After the initial RS/RA exchange, the AERO Server maintains the neighbor cache entry for the Client until the PD lifetimes expire. If the Client issues an RS "Renew", the Server extends the PD lifetimes. If the Client issues an RS "Release", or if the Client does not issue a "Renew" before the lifetime expires, the Server deletes the neighbor cache entry for the Client and withdraws the IP routes from the AERO routing system. The Server processes these and any other Client PD messages, and returns an RA "Reply". The Server may also issue an unsolicited RA "Reconfigure" message to inform the Client that it needs to renegotiate its PDs.

3.14.3.1. Lightweight DHCPv6 Relay Agent (LDRA)

When DHCPv6 is used as the PD service, AERO Clients and Servers are always on the same link (i.e., the AERO link) from the perspective of DHCPv6. However, in some implementations the DHCPv6 server and ND function may be located in separate modules. In that case, the Server’s AERO interface driver module can act as a Lightweight DHCPv6 Relay Agent (LDRA)[RFC6221] to relay PD messages to and from the DHCPv6 server module.

When the LDRA receives an authentic RS message, it extracts the PD message option and wraps it in IPv6/UDP headers. It sets the IPv6 source address to the source address of the RS message, sets the IPv6 destination address to ’All_DHCP_Relay_Agents_and_Servers’ and sets the UDP fields to values that will be understood by the DHCPv6 server.

The LDRA then wraps the message in a Relay-Forward message header and includes an Interface-ID option that includes enough information to allow the LDRA to forward the resulting Reply message back to the Client (e.g., the Client's link-layer addresses, a security association identifier, etc.). The LDRA also wraps the information in all of the SLLAO options from the RS message into the Interface-ID option, then forwards the message to the DHCPv6 server.

When the DHCPv6 server prepares a Reply message, it wraps the message in a Relay-Reply message and echoes the Interface-ID option. The DHCPv6 server then delivers the Relay-Reply message to the LDRA, which discards the Relay-Reply wrapper and IPv6/UDP headers, then delivers the DHCPv6 message to be wrapped into an RA response to the Client. The Server uses the information in the Interface ID option to prepare the RA message and to cache the link-layer addresses taken from the SLLAOs echoed in the Interface-ID option.

3.15. AERO Interface Route Optimization

When a source Client forwards packets to a prospective correspondent Client within the same AERO link domain (i.e., one for which the packet's destination address is covered by an ASP), the source Client MAY initiate an AERO link route optimization procedure on behalf of any of its native underlying interfaces. The procedure is based on an exchange of IPv6 ND messages using a chain of AERO Servers and Relays as a trust basis.

Although the Client is responsible for initiating route optimization, the Server is the policy enforcement point that determines whether route optimization is permitted. For example, on some AERO links route optimization would allow traffic to circumvent critical network-based traffic inspection points. In those cases, the Server can simply discard any route optimization messages instead of forwarding them.

The following sections specify the AERO link route optimization procedure.

3.15.1. Reference Operational Scenario

Figure 4 depicts the AERO link route optimization reference operational scenario, using IPv6 addressing as the example (while not shown, a corresponding example for IPv4 addressing can be easily constructed). The figure shows an AERO Relay ('R1'), two AERO

Servers (’S1’, ’S2’), two AERO Clients (’C1’, ’C2’) and two ordinary IPv6 hosts (’H1’, ’H2’):

```
            +--------------+  +--------------+  +--------------+
            |   Server S1  |  |    Relay R1  |  |   Server S2  |
            +--------------+  +--------------+  +--------------+
                fe80::2            fe80::1           fe80::3
                 L2(S1)             L2(R1)            L2(S2)
                   |                  |                 |
       X-----+-----+------------------+-----------------+----+----X
             |       AERO Link                               |
           L2(C1)                                          L2(C2)
    fe80::2001:db8:0:0                               fe80::2001:db8:1:0
     +--------------+                                 +--------------+
     |AERO Client C1|                                 |AERO Client C2|
     +--------------+                                 +--------------+
     2001:DB8:0::/48                                  2001:DB8:1::/48
             |                                                |
            .-.                                              .-.
         ,-(  _)-.   2001:db8:0::1      2001:db8:1::1     ,-(  _)-.
      .-(_  IP   )-.   +-------+          +-------+    .-(_  IP   )-.
    (__    EUN     )--|Host H1|          |Host H2|--(__    EUN     )
       `-(______)-'    +-------+          +-------+     `-(______)-'
```

Figure 4: AERO Reference Operational Scenario

In Figure 4, Relay (’R1’) assigns the administratively-provisioned link-local address fe80::1 to its AERO interface with link-layer address L2(R1), Server (’S1’) assigns the address fe80::2 with link-layer address L2(S1),and Server (’S2’) assigns the address fe80::3 with link-layer address L2(S2). Servers (’S1’) and (’S2’) next arrange to add their link-layer addresses to a published list of valid Servers for the AERO link.

AERO Client (’C1’) receives the ACP 2001:db8:0::/48 in an ND/PD exchange via AERO Server (’S1’) then assigns the address fe80::2001:db8:0:0 to its AERO interface with link-layer address L2(C1). Client (’C1’) configures a default route and neighbor cache entry via the AERO interface with next-hop address fe80::2 and link-layer address L2(S1), then sub-delegates the ACP to its attached EUNs. IPv6 host (’H1’) connects to the EUN, and configures the address 2001:db8:0::1.

AERO Client (’C2’) receives the ACP 2001:db8:1::/48 in an ND/PD exchange via AERO Server (’S2’) then assigns the address fe80::2001:db8:1:0 to its AERO interface with link-layer address L2(C2). Client (’C2’) configures a default route and neighbor cache entry via the AERO interface with next-hop address fe80::3 and link-layer address L2(S2), then sub-delegates the ACP to its attached EUNs. IPv6 host ('H2') connects to the EUN, and configures the address 2001:db8:1::1.

3.15.2. Concept of Operations

Again, with reference to Figure 4, when source host ('H1') sends a packet to destination host ('H2'), the packet is first forwarded over the source host's attached EUN to Client ('C1'). Client ('C1') then forwards the packet via its AERO interface to Server ('S1') and also sends an NS message toward Client ('C2') via Server ('S1').

Server ('S1') then re-encapsulates and forwards both the packet and the NS message out the same AERO interface toward Client ('C2') via Relay ('R1'). When Relay ('R1') receives the packet and NS message, it consults its forwarding table to discover Server ('S2') as the next hop toward Client ('C2'). Relay ('R1') then forwards both the packet and the NS message to Server ('S2'), which then forwards them to Client ('C2').

After Client ('C2') receives the NS message, it process the message and creates or updates a dynamic neighbor cache entry for Client ('C1'), then sends the NA response to the link-layer address of Server ('S2'). When Server ('S2') receives the NA message it re-encapsulates the message and forwards it on to Relay ('R1'), which re-encapsulates and forwards the message on to Server ('S1') which re-encapsulates and forwards the message on to Client ('C1').

After Client ('C1') receives the NA message, it processes the message and creates or updates a dynamic neighbor cache entry for Client ('C2'). Thereafter, forwarding of packets from Client ('C1') to Client ('C2') without involving any intermediate nodes is enabled. The mechanisms that support this exchange are specified in the following sections.

3.15.3. Sending NS Messages

When a Client forwards a packet with a source address from one of its ACPs toward a destination address covered by an ASP (i.e., toward another AERO Client connected to the same AERO link), the source Client MAY send an NS message forward toward the destination Client via the Server.

In the reference operational scenario, when Client ('C1') forwards a packet toward Client ('C2'), it MAY also send an NS message forward toward Client ('C2'), subject to rate limiting (see Section 8.2 of [RFC4861]). Client ('C1') prepares the NS message as follows:

- o the link-layer source address is set to 'L2(C1)' (i.e., the link-layer address of Client ('C1')).
- o the link-layer destination address is set to 'L2(S1)' (i.e., the link-layer address of Server ('S1')).
- o the network-layer source address is set to fe80::2001:db8:0:0 (i.e., the base AERO address of Client ('C1')).
- o the network-layer destination address is set to the AERO address corresponding to the destination address of Client ('C2').
- o the Type is set to 135.
- o the Target Address is set to the destination address of the packet that triggered route optimization.
- o the message includes one or more SLLAOs set to appropriate values for Client ('C1')'s native underlying interfaces.
- o the message includes one or more RIOs that include Client ('C1')'s ACPs [I-D.templin-6man-rio-redirect].
- o the message SHOULD include a Timestamp option and a Nonce option.

Note that the act of sending NS messages is cited as "MAY", since Client ('C1') may have advanced knowledge that the direct path to Client ('C2') would be unusable or otherwise undesirable. If the direct path later becomes unusable after the initial route optimization, Client ('C1') simply allows packets to again flow through Server ('S1').

3.15.4. Re-encapsulating and Relaying the NS

When Server ('S1') receives an NS message from Client ('C1'), it first verifies that the SLLAOs in the NS are a proper subset of the link-layer addresses in Client ('C1')'s neighbor cache entry. If the Client's SLLAOs are not acceptable, Server ('S1') discards the message. Otherwise, Server ('S1') verifies that Client ('C1') is authorized to use the ACPs encoded in the RIOs of the NS and discards the NS if verification fails.

Server ('S1') then examines the network-layer destination address of the NS to determine the next hop toward Client ('C2') by searching for the AERO address in the neighbor cache. Since Client ('C2') is not one of its neighbors, Server ('S1') re-encapsulates the NS and relays it via Relay ('R1') by changing the link-layer source address of the message to 'L2(S1)' and changing the link-layer destination address to 'L2(R1)'. Server ('S1') finally forwards the re-encapsulated message to Relay ('R1') without decrementing the network-layer TTL/Hop Limit field.

When Relay ('R1') receives the NS message from Server ('S1') it determines that Server ('S2') is the next hop toward Client ('C2') by consulting its forwarding table. Relay ('R1') then re-encapsulates the NS while changing the link-layer source address to 'L2(R1)' and changing the link-layer destination address to 'L2(S2)'. Relay ('R1') then relays the NS via Server ('S2').

When Server ('S2') receives the NS message from Relay ('R1') it determines that Client ('C2') is a neighbor by consulting its neighbor cache. Server ('S2') then re-encapsulates the NS while changing the link-layer source address to 'L2(S2)' and changing the link-layer destination address to 'L2(C2)'. Server ('S2') then forwards the message to Client ('C2').

3.15.5. Processing NSs and Sending NAs

When Client ('C2') receives the NS message, it accepts the NS only if the message has a link-layer source address of one of its Servers (e.g., L2(S2)). Client ('C2') further accepts the message only if it is willing to serve as a route optimization target.

In the reference operational scenario, when Client ('C2') receives a valid NS message, it either creates or updates a dynamic neighbor cache entry that stores the source address of the message as the network-layer address of Client ('C1') , stores the link-layer addresses found in the SLLAOs as the link-layer addresses of Client ('C1'), and stores the ACPs encoded in the RIOs of the NS as the ACPs for Client ('C1'). Client ('C2') then sets AcceptTime for the neighbor cache entry to ACCEPT_TIME.

After processing the message, Client ('C2') prepares an NA message response as follows:

o the link-layer source address is set to 'L2(C2)' (i.e., the link-layer address of Client ('C2')).

o the link-layer destination address is set to 'L2(S2)' (i.e., the link-layer address of Server ('S2')).

o the network-layer source address is set to fe80::2001:db8:1:0 (i.e., the base AERO address of Client ('C2')).

o the network-layer destination address is set to fe80::2001:db8:0:0 (i.e., the base AERO address of Client ('C1')).

- o the Type is set to 136.
- o The Target Address is set to the Target Address field in the NS message.
- o the message includes one or more TLLAOs set to appropriate values for Client ('C2')'s native underlying interfaces.
- o the message includes one or more RIOs that include Client ('C2')'s ACPs [I-D.templin-6man-rio-redirect].
- o the message SHOULD include a Timestamp option and MUST echo the Nonce option received in the NS (i.e., if a Nonce option is included).

Client ('C2') then sends the NA message to Server ('S2').

3.15.6. Re-encapsulating and Relaying NAs

When Server ('S2') receives an NA message from Client ('C2'), it first verifies that the TLLAOs in the NA are a proper subset of the Interface IDs in Client ('C2')'s neighbor cache entry. If the Client's TLLAOs are not acceptable, Server ('S2') discards the message. Otherwise, Server ('S2') verifies that Client ('C2') is authorized to use the ACPs encoded in the RIOs of the NA message. If validation fails, Server ('S2') discards the NA.

Server ('S2') then examines the network-layer destination address of the NA to determine the next hop toward Client ('C1') by searching for the AERO address in the neighbor cache. Since Client ('C1') is not a neighbor, Server ('S2') re-encapsulates the NA and relays it via Relay ('R1') by changing the link-layer source address of the message to 'L2(S2)' and changing the link-layer destination address to 'L2(R1)'. Server ('S2') finally forwards the re-encapsulated message to Relay ('R1') without decrementing the network-layer TTL/ Hop Limit field.

When Relay ('R1') receives the NA message from Server ('S2') it determines that Server ('S1') is the next hop toward Client ('C1') by consulting its forwarding table. Relay ('R1') then re-encapsulates the NA while changing the link-layer source address to 'L2(R1)' and changing the link-layer destination address to 'L2(S1)'. Relay ('R1') then relays the NA via Server ('S1').

When Server ('S1') receives the NA message from Relay ('R1') it determines that Client ('C1') is a neighbor by consulting its neighbor cache. Server ('S1') then re-encapsulates the NA while changing the link-layer source address to 'L2(S1)' and changing the link-layer destination address to 'L2(C1)'. Server ('S1') then forwards the message to Client ('C1').

3.15.7. Processing NAs

When Client ('C1') receives the NA message, it first verifies the Nonce value matches the value that it included in its NS message (if any). If the Nonce values match, Client ('C1') then processes the message as follows.

In the reference operational scenario, when Client ('C1') receives the NA message, it either creates or updates a dynamic neighbor cache entry that stores the source address of the message as the network-layer address of Client ('C2'), stores the link-layer addresses found in the TLLAOs as the link-layer addresses of Client ('C2') and stores the ACPs encoded in the RIOs of the NA as the ACPs for Client ('C2'). Client ('C1') then sets ForwardTime for the neighbor cache entry to FORWARD_TIME.

Now, Client ('C1') has a neighbor cache entry with a valid ForwardTime value, while Client ('C2') has a neighbor cache entry with a valid AcceptTime value. Thereafter, Client ('C1') may forward ordinary network-layer data packets directly to Client ('C2') without involving any intermediate nodes, and Client ('C2') can verify that the packets came from an acceptable source. (In order for Client ('C2') to forward packets to Client ('C1'), a corresponding NS/NA message exchange is required in the reverse direction; hence, the mechanism is asymmetric.)

3.15.8. Server and Proxy Extended Route Optimization

Route optimization may be initiated by the source Client by sending NS messages with SLLAOs corresponding to its native underlying interfaces. Route optimization for the source Client's other interfaces may be initiated by Servers and/or Proxies. Each node initiates route optimization by sending NS messages with SLLAOs only for those underlying interfaces they are authoritative for. Each node MUST consistently use the same Interface ID values to denote the same interfaces. The Interface IDs are established and maintained by the source Client's RS/RA exchanges.

The target Client's Server serves as a route optimization target if some or all of the target Client's underlying interfaces connect via NATs, Proxies and/or VPNs. In that case, when the source sends an NS message the target Server both forwards the NS toward a native underlying interface of the target Client (if any) and prepares an NA response the same as if it were the target Client (see: Section 3.15.5). (This means that the source may receive two separate NA messages - one from the target Server and one from the target Client. The source must accept the union of the information from both messages.)

For non-native underlying interfaces, the target Server includes a first TLLAO option in the NA with Interface ID set to 255 and includes any additional TLLAOs corresponding to the Client's NATed, Proxyed and/or VPNed underlying interfaces. The Server writes its own link-layer address in TLLAOs corresponding to NATed and VPNed underlying interfaces, and writes the link-layer address of the Proxy in TLLAOs corresponding to Proxyed underlying interfaces (while also setting the X flag). The Interface ID and QoS Preference values in the TLLAOs are those supplied by the Client during the initial RS/RA exchange and updated by any ensuing unsolicited NA messages. The target Server must then maintain a dynamic neighbor cache entry for the Client, but MUST NOT send BGP updates for Clients discovered through dynamic route optimization.

Thereafter, if the target Client moves to a new Server, the old Server sends unsolicited NA messages with no TLLAOs (subject to rate limiting) back to the source in response to data packets received from a correspondent node while forwarding the packets themselves to a Relay. The Relay will then either forward the packets to the new Server if the target Client has moved, or drop the packets if the target Client is no longer in the network. The source then allows future packets destined to the target Client to again flow through its own Server (or Relay). Note however that the old Server retains the neighbor cache entry with its associated AcceptTime since there may be many packets in flight. AcceptTime will then eventually decrement to 0 once the correspondent node processes and acts on the unsolicited NAs.

When the target Client (or Proxy) sends unsolicited NA messages to the target Server to update link-layer address and/or QoS preferences, the target Server repeats the messages to any of its dynamic neighbors while using its own link-layer and link-local addresses as the source addresses. In this way, the target Server acts as a link-scoped multicast repeater on behalf of the target Client (or Proxy).

(Note that instead of serving as the route optimization target for Proxy interfaces, the target Server could instead forward the source's NS messages and allow the Proxies to return NA messages, i.e., the same as for Clients on native interfaces. That would mean that the source could receive multiple NA messages from multiple Proxies and, if some or all NA messages are lost, the source would not be able to determine the full picture of the Client's Proxy affiliations. If this alternate architecture is deemed appropriate in some use cases, then the AERO Proxies could be employed to serve as route optimization targets instead of depending on the Servers to do so.)

3.16. Neighbor Unreachability Detection (NUD)

AERO nodes perform Neighbor Unreachability Detection (NUD) by sending NS messages to elicit solicited NA messages from neighbors the same as described in [RFC4861]. NUD is performed either reactively in response to persistent link-layer errors (see Section 3.13) or proactively to update neighbor cache entry timers and/or link-layer address information.

When an AERO node sends an NS/NA message, it uses one of its link-local addresses as the IPv6 source address and a link-local address of the neighbor as the IPv6 destination address. When route optimization directs a source AERO node to a target AERO node, the source node SHOULD proactively test the direct path by sending an initial NS message to elicit a solicited NA response. While testing the path, the source node can optionally continue sending packets via its default router, maintain a small queue of packets until target reachability is confirmed, or (optimistically) allow packets to flow directly to the target.

While data packets are still flowing, the source node thereafter periodically tests the direct path to the target node (see Section 7.3 of [RFC4861]) in order to keep dynamic neighbor cache entries alive. When the target node receives a valid NS message, it resets AcceptTime to ACCEPT_TIME and updates its cached link-layer addresses (if necessary). When the source node receives a solicited NA message, it resets ForwardTime to FORWARD_TIME and updates its cached link-layer addresses (if necessary). If the source node is unable to elicit a solicited NA response from the target node after MaxRetry attempts, it SHOULD set ForwardTime to 0. Otherwise, the source node considers the path usable and SHOULD thereafter process any link-layer errors as an indication that the direct path to the target node has either failed or has become intermittent.

When ForwardTime for a dynamic neighbor cache entry expires, the source node resumes sending any subsequent packets via a Server (or Relay) and may (eventually) attempt to re-initiate the AERO route optimization process. When AcceptTime for a dynamic neighbor cache entry expires, the target node discards any subsequent packets received directly from the source node. When both ForwardTime and AcceptTime for a dynamic neighbor cache entry expire, the node deletes the neighbor cache entry.

Note that an AERO node may have multiple underlying interface paths toward the target neighbor. In that case, the node SHOULD perform NUD over each underlying interface and only consider the neighbor unreachable if NUD fails over multiple underlying interface paths.

3.17. Mobility Management and Quality of Service (QoS)

AERO is an example of a Distributed Mobility Management (DMM) service. Each AERO Server is responsible for only a subset of the Clients on the AERO link, as opposed to a Centralized Mobility Management (CMM) service where there is a single network service for all Clients. AERO Clients coordinate with their regional Servers via RS/RA exchanges to maintain the DMM profile, and the AERO routing system tracks the current AERO Client/Server peering relationships.

Mobility management for AERO interfaces is accommodated by sending unsolicited NA messages the same as for announcing link-layer address changes for any interface that implements IPv6 ND [RFC4861]. When a node sends an unsolicited NA message, it sets the IPv6 source to its own link-local address, sets the IPv6 destination address to all-nodes multicast, sets the link-layer source address to its own address and sets the link-layer destination address to either a multicast address or the unicast link-layer address of a neighbor. If the unsolicited NA message must be received by multiple neighbors, the node sends multiple copies of the NA using a different unicast link-layer destination address for each neighbor. Mobility management considerations are specified in the following sections.

3.17.1. Forwarding Packets on Behalf of Departed Clients

When a Server receives packets with destination addresses that do not match one of its static neighbor cache Clients, it forwards the packets to a Relay and also returns an unsolicited NA message to the sender with no TLLAOs. The packets will be delivered to the target Client's new location, and the sender will realize that it needs to deprecate its routing information that associated the target with this Server.

3.17.2. Announcing Link-Layer Address and QoS Preference Changes

When a Client needs to change its link-layer addresses, e.g., due to a mobility event, it sends unsolicited NAs to its neighbors using the new link-layer address as the source address and with TLLAOs that include the new Client UDP Port Number, IP Address and P(i) values. If the Client sends the NA solely for the purpose of updating QoS preferences without updating the link-layer address, the Client sets the UDP Port Number and IP Address to 0.

The Client MAY send up to MaxRetry unsolicited NA messages in parallel with sending actual data packets in case one or more NAs are lost. If all NAs are lost, the neighbor will eventually invoke NUD by sending NS messages that include SLLAOs.

3.17.3. Bringing New Links Into Service

When a Client needs to bring new underlying interfaces into service (e.g., when it activates a new data link), it sends unsolicited NAs to its neighbors using the new link-layer address as the source address and with TLLAOs that include the new Client link-layer information.

3.17.4. Removing Existing Links from Service

When a Client needs to remove existing underlying interfaces from service (e.g., when it de-activates an existing data link), it sends unsolicited NAs to its neighbors with TLLAOs with all P(i) values set to 0.

If the Client needs to send the unsolicited NAs over an underlying interface other than the one being removed from service, it MUST include a current TLLAO for the sending interface as the first TLLAO and include TLLAOs for any underlying interface being removed from service as additional TLLAOs.

3.17.5. Implicit Mobility Management

AERO interface neighbors MAY provide a configuration option that allows them to perform implicit mobility management in which no ND messaging is used. In that case, the Client only transmits packets over a single interface at a time, and the neighbor always observes packets arriving from the Client from the same link-layer source address.

If the Client’s underlying interface address changes (either due to a readdressing of the original interface or switching to a new interface) the neighbor immediately updates the neighbor cache entry for the Client and begins accepting and sending packets to the Client’s new link-layer address. This implicit mobility method applies to use cases such as cellphones with both WiFi and Cellular interfaces where only one of the interfaces is active at a given time, and the Client automatically switches over to the backup interface if the primary interface fails.

3.17.6. Moving to a New Server

When a Client associates with a new Server, it performs the Client procedures specified in Section 3.14.2.

When a Client disassociates with an existing Server, it sends an RS "Release" message via a new Server with its base AERO address as the network-layer source address and the (administratively-provisioned) link-local address of the old Server as the network-layer destination address. The new Server then caches the Client’s AERO address and "Release" message parameters (e.g., "transaction ID") and writes its own administratively-provisioned link-local address as the network-layer source address. The new Server then forwards the message to a Relay, which forwards the message to the old Server.

When the old Server receives the "Release", it releases the Client’s ACP prefix delegations and routes. The old Server then deletes the Client’s neighbor cache entry so that any in-flight packets will be forwarded via a Relay to the new Server, which will forward them to the Client. The old Server finally returns a "Reply" message via a Relay to the new Server, which will decapsulate the "Reply" message and forward it as an RA "Reply" to the Client.

When the new Server forwards the "Reply" message, the Client can delete both the default route and the neighbor cache entry for the old Server. (Note that since messages may be lost in the network the Client SHOULD retry until it gets an RA "Reply" indicating that the RS "Release" was successful. If the Client does not receive a "Reply" after MaxRetry attempts, the old Server may have failed and the Client should discontinue its "Release" attempts.)

Finally, Clients SHOULD NOT move rapidly between Servers in order to avoid causing excessive oscillations in the AERO routing system. Such oscillations could result in intermittent reachability for the Client itself, while causing little harm to the network. Examples of when a Client might wish to change to a different Server include a Server that has gone unreachable, topological movements of significant distance, etc.

3.18. Multicast Considerations

When the underlying network does not support multicast, AERO Clients map link-scoped multicast addresses to the link-layer address of a Server, which acts as a multicast forwarding agent. The AERO Client also serves as an IGMP/MLD Proxy for its EUNs and/or hosted applications per [RFC4605] while using the link-layer address of the Server as the link-layer address for all multicast packets.

When the underlying network supports multicast, AERO nodes use the multicast address mapping specification found in [RFC2529] for IPv4 underlying networks and use a TBD site-scoped multicast mapping for IPv6 underlying networks. In that case, border routers must ensure that the encapsulated site-scoped multicast packets do not leak outside of the site spanned by the AERO link.

4. The AERO Proxy

In some deployments, AERO Clients may be located in secured enclaves (e.g., a corporate enterprise network, a radio access network, etc.) that do not allow direct communications from the Client to a Server in the outside Internetwork. In that case, the secured enclave can employ an AERO Proxy.

The AERO Proxy is located at the secured enclave perimeter and listens for RS messages originating from or RA messages destined to AERO Clients located within the enclave. The Proxy acts on these control messages as follows:

- o when the Proxy receives an RS message from a Client within the secured enclave, it first authenticates the message then creates a proxy neighbor cache entry for the Client in the INCOMPLETE State and caches the Client and Server link-layer address along with any identifying information including PD "transaction IDs", "Client Identifiers", etc. and/or ND Nonce values. The Proxy then re-encapsulates the message and forwards it to the Server indicated by the destination link-layer address in the packet while substituting its own external address as the source link-layer address.

- o when the Proxy receives an RA message from the Server, it matches the message with the (INCOMPLETE) proxy neighbor cache entry. The Proxy then caches the route information in the message as a mapping from the Client’s ACPs to the Client’s address within the secured enclave, and sets the neighbor cache entry state to REACHABLE. The Proxy then re-encapsulates the message and forwards it to the Client. At the same time, the Proxy sends an unsolicited NA message including a TLLAO with the X flag set back to the Server to assert that it is indeed a Proxy as opposed to an ordinary NAT. (In environments where spoofing is a threat, the Proxy signs the NA using SEND.)

After the initial RS/RA handshake, the Proxy can send unsolicited NA messages to the Client’s Server(s) to update Server neighbor cache entries on behalf of the Client. (For example, the Proxy can send NA messages with a TLLAO with UDP Port Number and IP Address set to 0 and with valid P(i) values to update the Server(s) with the Client’s new QoS preferences for that link). The Proxy also forwards any unsolicited NA messages originating from the Client to the Client’s Server(s) (e.g. if the Client needs to announce new QoS preferences on its own behalf), and forwards any data packets originating from the Client to the Client’s primary Server.

At the same time, for data packets originating from a Client within the enclave with destination addresses that match an ASP, the Proxy can initiate route optimization by sending an NS message via the Server to solicit an NA message from a target node on the path to the destination Client the same as discussed in Section 3.15. The target must deliver the NA message directly to the Proxy, i.e., instead of relaying through the backward chain of Relays and Servers, since the backward chain could deliver the NA to a different Proxy besides the one that produced the NS. For this reason, the Proxy prepares an NS message as specified in Section 3.15.3, but with its own link-layer address as the link-layer source address and with a single SLLAO containing its link-layer address and with the X flag set to indicate that direct delivery is required.

When the target receives the NS message, it creates a dynamic neighbor cache entry in the ACCEPT state and returns an NA message directly to the Proxy. When the target is a Client, it includes TLLAOs in the NA message with link-layer addresses corresponding to its native underling interfaces. When the target is a Server, it includes a first TLLAO in the NA message with Interface ID set to 255 and with its own link-layer address information, and also includes additional TLLAOs corresponding to the destination Client’s Proxyed, NATed or VPNed underlying interfaces. (For NATed or VPNed underlying interfaces the server writes its own link-layer address in the TLLAO, and for Proxyed interfaces it writes the link-layer address of the Proxy.) When the source Proxy receives the NA message, it creates a dynamic neighbor cache entry in the FORWARD state that associates the TLLAOs of the NA message as the next-hop toward the routes advertised in the NA RIOs.

When a source Proxy sends route optimization NS messages toward the target, it can include RIOs to assert specific routes, and the target will only accept packets from the source Proxy with matching source addresses. If the source Proxy wishes to assert a "wildcard" route, it includes an RIO in the NS message with Prefix and Prefix Length set to 0. In that case, the target will either accept or ignore the NS based on its configured trust policy. If the target accepts the NS, it will accept all packets originating from the source Proxy regardless of their source address.

After the initial NS/NA exchange, the target may need to update the neighbor cache entries for any source Proxies for which it holds a dynamic neighbor cache entry in the ACCEPT state. The target therefore sends unsolicited NA messages to announce any link layer changes. As a result:

- o the source Proxy may receive unsolicited NA messages with TLLAOs with new UDP Port Number, IP Address and/or QoS preferences from the target. In that case, the Proxy updates its neighbor cache entry and forwards future outbound packets based on the new link layer information.
- o the source Proxy may receive reflected packets destined to the link-layer address of a departed Client. In that case, the Proxy proceeds as discussed in Section 3.8.5.
- o the source Proxy may receive link-layer Destination Unreachable messages in response to data packets it sends to one of the target link-layer addresses. In that case, the Proxy processes the link-layer error messages as an indication that the path may be failing and proceeds as discussed in Section 3.13.

After the NS/NA exchange, while data packets are still flowing the source Proxy sends additional NS messages to the target using the address in the target's first TLLAO as the destination. The NS message will update the target's AcceptTime timer, and the resulting NA reply will update the source Proxy's ForwardTime timer in their respective neighbor cache entries.

If at some later time the target Client departs from its secured enclave, the Proxy sends unsolicited NAs to the Client's Servers to announce the departure.

5. Direct Underlying Interfaces

When a Client's AERO interface is configured over a direct underlying interface, the neighbor at the other end of the direct link can receive packets without any encapsulation. In that case, the Client sends packets over the direct link according to the QoS preferences associated with its underling interfaces. If the direct underlying interface has the highest QoS preference, then the Client's IP packets are transmitted directly to the peer without going through an underlying network. If other underlying interfaces have higher QoS preferences, then the Client's IP packets are transmitted via a different underlying interface, which may result in the inclusion of AERO Proxies, Servers and Relays in the communications path. Direct underlying interfaces must be tested periodically for reachability, e.g., via NUD, via periodic unsolicited NAs, etc.

6. Operation on AERO Links with /64 ASPs

IPv6 AERO links typically have ASPs that cover many candidate ACPs of length /64 or shorter. However, in some cases it may be desirable to use AERO over links that have only a /64 ASP. This can be accommodated by treating all Clients on the AERO link as simple hosts that receive /128 prefix delegations.

In that case, the Client sends an RS message to the Server the same as for ordinary AERO links. The Server responds with an RA message that includes one or more /128 prefixes (i.e., singleton addresses) that include the /64 ASP prefix along with an interface identifier portion to be assigned to the Client. The Client and Server then configure their AERO addresses based on the interface identifier portions of the /128s (i.e., the lower 64 bits) and not based on the /64 prefix (i.e., the upper 64 bits).

For example, if the ASP for the host-only IPv6 AERO link is 2001:db8:1000:2000::/64, each Client will receive one or more /128 IPv6 prefix delegations such as 2001:db8:1000:2000::1/128, 2001:db8:1000:2000::2/128, etc. When the Client receives the prefix delegations, it assigns the AERO addresses fe80::1, fe80::2, etc. to the AERO interface, and assigns the global IPv6 addresses (i.e., the /128s) to either the AERO interface or an internal virtual interface such as a loopback. In this arrangement, the Client conducts route optimization in the same sense as discussed in Section 3.15.

This specification has applicability for nodes that act as a Client on an "upstream" AERO link, but also act as a Server on "downstream" AERO links. More specifically, if the node acts as a Client to receive a /64 prefix from the upstream AERO link it can then act as a Server to provision /128s to Clients on downstream AERO links.

7. Implementation Status

An AERO implementation based on OpenVPN (https://openvpn.net/) was announced on the v6ops mailing list on January 10, 2018. The latest version is available at: http://linkupnetworks.net/aero/AERO-OpenVPN-1.0.tgz.

An initial public release of the AERO proof-of-concept source code was announced on the intarea mailing list on August 21, 2015. The latest version is available at: http://linkupnetworks.net/aero/aero-3.0.3a.tgz.

8. IANA Considerations

The IANA has assigned a 4-octet Private Enterprise Number "45282" for AERO in the "enterprise-numbers" registry.

The IANA has assigned the UDP port number "8060" for an earlier experimental version of AERO [RFC6706]. This document obsoletes [RFC6706] and claims the UDP port number "8060" for all future use.

No further IANA actions are required.

9. Security Considerations

AERO link security considerations are the same as for standard IPv6 Neighbor Discovery [RFC4861] except that AERO improves on some aspects. In particular, AERO uses a trust basis between Clients and Servers, where the Clients only engage in the AERO mechanism when it is facilitated by a trusted Server.

NS and NA messages SHOULD include a Timestamp option (see Section 5.3 of [RFC3971]) that other AERO nodes can use to verify the message time of origin. NS and RS messages SHOULD include a Nonce option (see Section 5.3 of [RFC3971]) that recipients echo back in corresponding responses. In cases where spoofing cannot be mitigated through other means, however, all AERO IPv6 ND messages should employ SEND [RFC3971], which also protects the PD information embedded in RS/RA message options.

AERO links must be protected against link-layer address spoofing attacks in which an attacker on the link pretends to be a trusted neighbor. Links that provide link-layer securing mechanisms (e.g., IEEE 802.1X WLANs) and links that provide physical security (e.g., enterprise network wired LANs) provide a first line of defense, however AERO nodes SHOULD also use securing services such as SEND for Client authentication and network admission control. Following authenticated Client admission and prefix delegation procedures, AERO nodes MUST ensure that the source of data packets corresponds to the node to which the prefixes were delegated.

AERO Clients MUST ensure that their connectivity is not used by unauthorized nodes on their EUNs to gain access to a protected network, i.e., AERO Clients that act as routers MUST NOT provide routing services for unauthorized nodes. (This concern is no different than for ordinary hosts that receive an IP address delegation but then "share" the address with other nodes via some form of Internet connection sharing such as tethering.)

AERO Clients, Servers and Relays on the open Internet are susceptible to the same attack profiles as for any Internet nodes. For this reason, IP security SHOULD be used when AERO is employed over unmanaged/unsecured links using securing mechanisms such as IPsec [RFC4301], IKE [RFC5996] and/or TLS [RFC5246]. In some environments, however, the use of end-to-end security from Clients to correspondent nodes (i.e., other Clients and/or Internet nodes) could obviate the need for IP security between AERO Clients, Servers and Relays.

AERO Servers and Relays present targets for traffic amplification DoS attacks. This concern is no different than for widely-deployed VPN security gateways in the Internet, where attackers could send spoofed packets to the gateways at high data rates. This can be mitigated by connecting Relays and Servers over dedicated links with no connections to the Internet and/or when connections to the Internet are only permitted through well-managed firewalls.

Traffic amplification DoS attacks can also target an AERO Client’s low data rate links. This is a concern not only for Clients located on the open Internet but also for Clients in secured enclaves. AERO Servers can institute rate limits that protect Clients from receiving packet floods that could DoS low data rate links.

Security considerations for accepting link-layer ICMP messages and reflected packets are discussed throughout the document.

10. Acknowledgements

Discussions in the IETF, aviation standards communities and private exchanges helped shape some of the concepts in this work. Individuals who contributed insights include Mikael Abrahamsson, Mark Andrews, Fred Baker, Bob Braden, Stewart Bryant, Brian Carpenter, Wojciech Dec, Ralph Droms, Adrian Farrel, Sri Gundavelli, Brian Haberman, Bernhard Haindl, Joel Halpern, Tom Herbert, Sascha Hlusiak, Lee Howard, Andre Kostur, Ted Lemon, Andy Malis, Satoru Matsushima, Tomek Mrugalski, Alexandru Petrescu, Behcet Saikaya, Michal Skorepa, Joe Touch, Bernie Volz, Ryuji Wakikawa, Lloyd Wood and James Woodyatt. Members of the IESG also provided valuable input during their review process that greatly improved the document. Special thanks go to Stewart Bryant, Joel Halpern and Brian Haberman for their shepherding guidance during the publication of the AERO first edition.

This work has further been encouraged and supported by Boeing colleagues including Kyle Bae, M. Wayne Benson, Dave Bernhardt, Cam Brodie, Balaguruna Chidambaram, Irene Chin, Bruce Cornish, Claudiu Danilov, Wen Fang, Anthony Gregory, Jeff Holland, Ed King, Gene MacLean III, Rob Muszkiewicz, Sean O’Sullivan, Kent Shuey, Brian

Skeen, Mike Slane, Carrie Spiker, Brendan Williams, Julie Wulff, Yueli Yang, Eric Yeh and other members of the BR&T and BIT mobile networking teams. Wayne Benson, Kyle Bae and Eric Yeh are especially acknowledged for implementing the AERO functions as extensions to the public domain OpenVPN distribution.

Earlier works on NBMA tunneling approaches are found in [RFC2529][RFC5214][RFC5569].

Many of the constructs presented in this second edition of AERO are based on the author's earlier works, including:

- o The Internet Routing Overlay Network (IRON) [RFC6179][I-D.templin-ironbis]
- o Virtual Enterprise Traversal (VET) [RFC5558][I-D.templin-intarea-vet]
- o The Subnetwork Encapsulation and Adaptation Layer (SEAL) [RFC5320][I-D.templin-intarea-seal]
- o AERO, First Edition [RFC6706]

Note that these works cite numerous earlier efforts that are not also cited here due to space limitations. The authors of those earlier works are acknowledged for their insights.

This work is aligned with the NASA Safe Autonomous Systems Operation (SASO) program under NASA contract number NNA16BD84C.

This work is aligned with the FAA as per the SE2025 contract number DTFAWA-15-D-00030.

This work is aligned with the Boeing Information Technology (BIT) MobileNet program.

This work is aligned with the Boeing Research and Technology (BR&T) autonomous systems networking program.

11. References

11.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <https://www.rfc-editor.org/info/rfc792>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <https://www.rfc-editor.org/info/rfc2474>.

[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, <https://www.rfc-editor.org/info/rfc3315>.

[RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, DOI 10.17487/RFC3633, December 2003, <https://www.rfc-editor.org/info/rfc3633>.

[RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <https://www.rfc-editor.org/info/rfc3971>.

[RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <https://www.rfc-editor.org/info/rfc4191>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <https://www.rfc-editor.org/info/rfc4861>.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <https://www.rfc-editor.org/info/rfc4862>.

[RFC5175] Haberman, B., Ed. and R. Hinden, "IPv6 Router Advertisement Flags Option", RFC 5175, DOI 10.17487/RFC5175, March 2008, <https://www.rfc-editor.org/info/rfc5175>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

11.2. Informative References

[BGP] Huston, G., "BGP in 2015, http://potaroo.net", January 2016.

[I-D.ietf-intarea-gue]
Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", draft-ietf-intarea-gue-05 (work in progress), December 2017.

[I-D.ietf-intarea-gue-extensions]
Herbert, T., Yong, L., and F. Templin, "Extensions for Generic UDP Encapsulation", draft-ietf-intarea-gue-extensions-04 (work in progress), March 2018.

[I-D.ietf-intarea-tunnels]
Touch, J. and M. Townsley, "IP Tunnels in the Internet Architecture", draft-ietf-intarea-tunnels-08 (work in progress), January 2018.

[I-D.templin-6man-dhcpv6-ndopt]
Templin, F., "IPv6 Neighbor Discovery Extensions for Prefix Delegation", draft-templin-6man-dhcpv6-ndopt-04 (work in progress), March 2018.

[I-D.templin-6man-rio-redirect]
Templin, F. and j. woodyatt, "Route Information Options in IPv6 Neighbor Discovery", draft-templin-6man-rio-redirect-06 (work in progress), May 2018.

[I-D.templin-atn-bgp]
Templin, F., Saccone, G., Dawra, G., and A. Lindem, "A Simple BGP-based Mobile Routing System for the Aeronautical Telecommunications Network", draft-templin-atn-bgp-06 (work in progress), March 2018.

[I-D.templin-intarea-grefrag]
Templin, F., "GRE Tunnel Level Fragmentation", draft-templin-intarea-grefrag-04 (work in progress), July 2016.

[I-D.templin-intarea-seal]
Templin, F., "The Subnetwork Encapsulation and Adaptation Layer (SEAL)", draft-templin-intarea-seal-68 (work in progress), January 2014.

[I-D.templin-intarea-vet]
Templin, F., "Virtual Enterprise Traversal (VET)", draft-templin-intarea-vet-40 (work in progress), May 2013.

[I-D.templin-ironbis]
Templin, F., "The Interior Routing Overlay Network (IRON)", draft-templin-ironbis-16 (work in progress), March 2014.

[I-D.templin-v6ops-pdhost]
Templin, F., "IPv6 Prefix Delegation Models", draft-templin-v6ops-pdhost-19 (work in progress), March 2018.

[OVPN] OpenVPN, O., "http://openvpn.net", October 2016.

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://www.rfc-editor.org/info/rfc1122>.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <https://www.rfc-editor.org/info/rfc1191>.

[RFC1812] Baker, F., Ed., "Requirements for IP Version 4 Routers", RFC 1812, DOI 10.17487/RFC1812, June 1995, <https://www.rfc-editor.org/info/rfc1812>.

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, DOI 10.17487/RFC1981, August 1996, <https://www.rfc-editor.org/info/rfc1981>.

[RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, DOI 10.17487/RFC2003, October 1996, <https://www.rfc-editor.org/info/rfc2003>.

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <https://www.rfc-editor.org/info/rfc2131>.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <https://www.rfc-editor.org/info/rfc2473>.

[RFC2529] Carpenter, B. and C. Jung, "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels", RFC 2529, DOI 10.17487/RFC2529, March 1999, <https://www.rfc-editor.org/info/rfc2529>.

[RFC2764] Gleeson, B., Lin, A., Heinanen, J., Armitage, G., and A. Malis, "A Framework for IP Based Virtual Private Networks", RFC 2764, DOI 10.17487/RFC2764, February 2000, <https://www.rfc-editor.org/info/rfc2764>.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <https://www.rfc-editor.org/info/rfc2784>.

[RFC2890] Dommety, G., "Key and Sequence Number Extensions to GRE", RFC 2890, DOI 10.17487/RFC2890, September 2000, <https://www.rfc-editor.org/info/rfc2890>.

[RFC2923] Lahey, K., "TCP Problems with Path MTU Discovery", RFC 2923, DOI 10.17487/RFC2923, September 2000, <https://www.rfc-editor.org/info/rfc2923>.

[RFC2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, DOI 10.17487/RFC2983, October 2000, <https://www.rfc-editor.org/info/rfc2983>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <https://www.rfc-editor.org/info/rfc3168>.

[RFC3819] Karn, P., Ed., Bormann, C., Fairhurst, G., Grossman, D., Ludwig, R., Mahdavi, J., Montenegro, G., Touch, J., and L. Wood, "Advice for Internet Subnetwork Designers", BCP 89, RFC 3819, DOI 10.17487/RFC3819, July 2004, <https://www.rfc-editor.org/info/rfc3819>.

[RFC4213] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, DOI 10.17487/RFC4213, October 2005, <https://www.rfc-editor.org/info/rfc4213>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/info/rfc4271>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <https://www.rfc-editor.org/info/rfc4291>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

[RFC4389] Thaler, D., Talwar, M., and C. Patel, "Neighbor Discovery Proxies (ND Proxy)", RFC 4389, DOI 10.17487/RFC4389, April 2006, <https://www.rfc-editor.org/info/rfc4389>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[RFC4511] Sermersheim, J., Ed., "Lightweight Directory Access Protocol (LDAP): The Protocol", RFC 4511, DOI 10.17487/RFC4511, June 2006, <https://www.rfc-editor.org/info/rfc4511>.

[RFC4605] Fenner, B., He, H., Haberman, B., and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, DOI 10.17487/RFC4605, August 2006, <https://www.rfc-editor.org/info/rfc4605>.

[RFC4963] Heffner, J., Mathis, M., and B. Chandler, "IPv4 Reassembly Errors at High Data Rates", RFC 4963, DOI 10.17487/RFC4963, July 2007, <https://www.rfc-editor.org/info/rfc4963>.

[RFC5214] Templin, F., Gleeson, T., and D. Thaler, "Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)", RFC 5214, DOI 10.17487/RFC5214, March 2008, <https://www.rfc-editor.org/info/rfc5214>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC5320] Templin, F., Ed., "The Subnetwork Encapsulation and Adaptation Layer (SEAL)", RFC 5320, DOI 10.17487/RFC5320, February 2010, <https://www.rfc-editor.org/info/rfc5320>.

[RFC5522] Eddy, W., Ivancic, W., and T. Davis, "Network Mobility Route Optimization Requirements for Operational Use in Aeronautics and Space Exploration Mobile Networks", RFC 5522, DOI 10.17487/RFC5522, October 2009, <https://www.rfc-editor.org/info/rfc5522>.

[RFC5558] Templin, F., Ed., "Virtual Enterprise Traversal (VET)", RFC 5558, DOI 10.17487/RFC5558, February 2010, <https://www.rfc-editor.org/info/rfc5558>.

[RFC5569] Despres, R., "IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)", RFC 5569, DOI 10.17487/RFC5569, January 2010, <https://www.rfc-editor.org/info/rfc5569>.

[RFC5720] Templin, F., "Routing and Addressing in Networks with Global Enterprise Recursion (RANGER)", RFC 5720, DOI 10.17487/RFC5720, February 2010, <https://www.rfc-editor.org/info/rfc5720>.

[RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, DOI 10.17487/RFC5996, September 2010, <https://www.rfc-editor.org/info/rfc5996>.

[RFC6179] Templin, F., Ed., "The Internet Routing Overlay Network (IRON)", RFC 6179, DOI 10.17487/RFC6179, March 2011, <https://www.rfc-editor.org/info/rfc6179>.

[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, <https://www.rfc-editor.org/info/rfc6221>.

[RFC6422] Lemon, T. and Q. Wu, "Relay-Supplied DHCP Options", RFC 6422, DOI 10.17487/RFC6422, December 2011, <https://www.rfc-editor.org/info/rfc6422>.

[RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <https://www.rfc-editor.org/info/rfc6438>.

[RFC6706] Templin, F., Ed., "Asymmetric Extended Route Optimization (AERO)", RFC 6706, DOI 10.17487/RFC6706, August 2012, <https://www.rfc-editor.org/info/rfc6706>.

[RFC6864] Touch, J., "Updated Specification of the IPv4 ID Field", RFC 6864, DOI 10.17487/RFC6864, February 2013, <https://www.rfc-editor.org/info/rfc6864>.

[TUNTAP] Wikipedia, W., "http://en.wikipedia.org/wiki/TUN/TAP", October 2014.

Appendix A. AERO Alternate Encapsulations

When GUE encapsulation is not needed, AERO can use common encapsulations such as IP-in-IP [RFC2003][RFC2473][RFC4213], Generic Routing Encapsulation (GRE) [RFC2784][RFC2890] and others. The encapsulation is therefore only differentiated from non-AERO tunnels through the application of AERO control messaging and not through, e.g., a well-known UDP port number.

As for GUE encapsulation, alternate AERO encapsulation formats may require encapsulation layer fragmentation. For simple IP-in-IP encapsulation, an IPv6 fragment header is inserted directly between the inner and outer IP headers when needed, i.e., even if the outer header is IPv4. The IPv6 Fragment Header is identified to the outer IP layer by its IP protocol number, and the Next Header field in the IPv6 Fragment Header identifies the inner IP header version. For GRE encapsulation, a GRE fragment header is inserted within the GRE header [I-D.templin-intarea-grefrag].

Figure 5 shows the AERO IP-in-IP encapsulation format before any fragmentation is applied:

```
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |     Outer IPv4 Header     |     |    Outer IPv6 Header      |
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |IPv6 Frag Header (optional)|     |IPv6 Frag Header (optional)|
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |      Inner IP Header      |     |      Inner IP Header      |
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |                           |     |                           |
        ~                           ~     ~                           ~
        ~    Inner Packet Body      ~     ~     Inner Packet Body     ~
        ~                           ~     ~                           ~
        |                           |     |                           |
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

        Minimal Encapsulation in IPv4      Minimal Encapsulation in IPv6
```

Figure 5: Minimal Encapsulation Format using IP-in-IP

Figure 6 shows the AERO GRE encapsulation format before any fragmentation is applied:

```
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |        Outer IP Header        |
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |          GRE Header           |
        | (with checksum, key, etc..)   |
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        | GRE Fragment Header (optional)|
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |        Inner IP Header        |
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |                               |
        ~                               ~
        ~      Inner Packet Body        ~
        ~                               ~
        |                               |
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 6: Minimal Encapsulation Using GRE

Alternate encapsulation may be preferred in environments where GUE encapsulation would add unnecessary overhead. For example, certain low-bandwidth wireless data links may benefit from a reduced encapsulation overhead.

GUE encapsulation can traverse network paths that are inaccessible to non-UDP encapsulations, e.g., for crossing Network Address Translators (NATs). More and more, network middleboxes are also being configured to discard packets that include anything other than a well-known IP protocol such as UDP and TCP. It may therefore be necessary to determine the potential for middlebox filtering before enabling alternate encapsulation in a given environment.

In addition to IP-in-IP, GRE and GUE, AERO can also use security encapsulations such as IPsec and SSL/TLS. In that case, AERO control messaging and route determination occur before security encapsulation is applied for outgoing packets and after security decapsulation is applied for incoming packets.

AERO is especially well suited for use with VPN system encapsulations such as OpenVPN [OVPN].

Appendix B. When to Insert an Encapsulation Fragment Header

An encapsulation fragment header is inserted when the AERO tunnel ingress needs to apply fragmentation to accommodate packets that must be delivered without loss due to a size restriction. Fragmentation is performed on the inner packet while encapsulating each inner packet fragment in outer IP and encapsulation layer headers that differ only in the fragment header fields.

The fragment header can also be inserted in order to include a coherent Identification value with each packet, e.g., to aid in Duplicate Packet Detection (DPD). In this way, network nodes can cache the Identification values of recently-seen packets and use the cached values to determine whether a newly-arrived packet is in fact a duplicate. The Identification value within each packet could further provide a rough indicator of packet reordering, e.g., in cases when the tunnel egress wishes to discard packets that are grossly out of order.

In some use cases, there may be operational assurance that no fragmentation of any kind will be necessary, or that only occasional large control messages will require fragmentation. In that case, the encapsulation fragment header can be omitted and ordinary fragmentation of the outer IP protocol version can be applied when necessary.

Appendix C. Autoconfiguration for Constrained Platforms

On some platforms (e.g., popular cell phone operating systems), the act of assigning a default IPv6 route and/or assigning an address to an interface may not be permitted from a user application due to security policy. Typically, those platforms include a TUN/TAP interface [TUNTAP] that acts as a point-to-point conduit between user applications and the AERO interface. In that case, the Client can instead generate a "synthesized RA" message. The message conforms to [RFC4861] and is prepared as follows:

o the IPv6 source address is the Client's AERO address

o the IPv6 destination address is all-nodes multicast

o the Router Lifetime is set to a time that is no longer than the ACP DHCPv6 lifetime

o the message does not include a Source Link Layer Address Option (SLLAO)

o the message includes a Prefix Information Option (PIO) with a /64 prefix taken from the ACP as the prefix for autoconfiguration

The Client then sends the synthesized RA message via the TUN/TAP interface, where the operating system kernel will interpret it as though it were generated by an actual router. The operating system will then install a default route and use StateLess Address AutoConfiguration (SLAAC) to configure an IPv6 address on the TUN/TAP interface. Methods for similarly installing an IPv4 default route and IPv4 address on the TUN/TAP interface are based on synthesized DHCPv4 messages [RFC2131].

Appendix D. Operational Deployment Alternatives

AERO can be used in many different variations based on the specific use case. The following sections discuss variations that adhere to the AERO principles while allowing selective application of AERO components.

D.1. Operation on AERO Links Without DHCPv6 Services

When Servers on the AERO link do not provide DHCPv6 services, operation can still be accommodated through administrative configuration of ACPs on AERO Clients. In that case, administrative configurations of AERO interface neighbor cache entries on both the Server and Client are also necessary. However, this may interfere with the ability for Clients to dynamically change to new Servers, and can expose the AERO link to misconfigurations unless the administrative configurations are carefully coordinated.

D.2. Operation on Server-less AERO Links

In some AERO link scenarios, there may be no Servers on the link and/or no need for Clients to use a Server as an intermediary trust anchor. In that case, each Client acts as a Server unto itself to establish neighbor cache entries by performing direct Client-to-Client IPv6 ND message exchanges, and some other form of trust basis must be applied so that each Client can verify that the prospective neighbor is authorized to use its claimed ACP.

When there is no Server on the link, Clients must arrange to receive ACPs and publish them via a secure alternate PD authority through some means outside the scope of this document.

D.3. Operation on Client-less AERO Links

In some environments, the AERO service may be useful for mobile nodes that do not implement the AERO Client function and do not perform encapsulation. For example, if the mobile node has a way of injecting its ACP into the access subnetwork routing system an AERO Server connected to the same access network can accept the ACP prefix injection as an indication that a new mobile node has come onto the subnetwork. The Server can then inject the ACP into the BGP routing system the same as if an AERO Client/Server DHCPv6 PD exchange had occurred. If the mobile node subsequently withdraws the ACP from the access network routing system, the Server can then withdraw the ACP from the BGP routing system.

In this arrangement, AERO Servers and Relays are used in exactly the same ways as for environments where DHCPv6 Client/Server exchanges are supported. However, the access subnetwork routing systems must be capable of accommodating rapid ACP injections and withdrawals from mobile nodes with the understanding that the information must be propagated to all routers in the system. Operational experience has shown that this kind of routing system "churn" can lead to overall instability and routing system inconsistency.

D.4. Manually-Configured AERO Tunnels

In addition to the dynamic neighbor discovery procedures for AERO link neighbors described above, AERO encapsulation can be applied to manually-configured tunnels. In that case, the tunnel endpoints use an administratively-provisioned link-local address and exchange NS/NA messages the same as for dynamically-established tunnels.

D.5. Encapsulation Avoidance on Relay-Server Dedicated Links

In some environments, AERO Servers and Relays may be connected by dedicated point-to-point links, e.g., high speed fiberoptic leased lines. In that case, the Servers and Relays can participate in the AERO link the same as specified above but can avoid encapsulation over the dedicated links. In that case, however, the links would be dedicated for AERO and could not be multiplexed for both AERO and non-AERO communications.

D.6. Encapsulation Protocol Version Considerations

A source Client may connect only to an IPvX underlying network, while the target Client connects only to an IPvY underlying network. In that case, the target and source Clients have no means for reaching each other directly (since they connect to underlying networks of different IP protocol versions) and so must ignore any route optimization messages and continue to send packets via their Servers.

D.7. Extending AERO Links Through Security Gateways

When an enterprise mobile node moves from a campus LAN connection to a public Internet link, it must re-enter the enterprise via a security gateway that has both a physical interface connection to the Internet and a physical interface connection to the enterprise internetwork. This most often entails the establishment of a Virtual Private Network (VPN) link over the public Internet from the mobile node to the security gateway. During this process, the mobile node supplies the security gateway with its public Internet address as the link-layer address for the VPN. The mobile node then acts as an AERO Client to negotiate with the security gateway to obtain its ACP.

In order to satisfy this need, the security gateway also operates as an AERO Server with support for AERO Client proxying. In particular, when a mobile node (i.e., the Client) connects via the security gateway (i.e., the Server), the Server provides the Client with an ACP in a DHCPv6 PD exchange the same as if it were attached to an enterprise campus access link. The Server then replaces the Client's link-layer source address with the Server's enterprise-facing link-layer address in all AERO messages the Client sends toward neighbors on the AERO link. The AERO messages are then delivered to other nodes on the AERO link as if they were originated by the security gateway instead of by the AERO Client. In the reverse direction, the AERO messages sourced by nodes within the enterprise network can be forwarded to the security gateway, which then replaces the link-layer destination address with the Client's link-layer address and replaces the link-layer source address with its own (Internet-facing) link-layer address.

After receiving the ACP, the Client can send IP packets that use an address taken from the ACP as the network layer source address, the Client's link-layer address as the link-layer source address, and the Server's Internet-facing link-layer address as the link-layer destination address. The Server will then rewrite the link-layer source address with the Server's own enterprise-facing link-layer address and rewrite the link-layer destination address with the target AERO node's link-layer address, and the packets will enter the enterprise network as though they were sourced from a node located within the enterprise. In the reverse direction, when a packet sourced by a node within the enterprise network uses a destination address from the Client's ACP, the packet will be delivered to the security gateway which then rewrites the link-layer destination address to the Client's link-layer address and rewrites the link-layer source address to the Server's Internet-facing link-layer address. The Server then delivers the packet across the VPN to the AERO Client. In this way, the AERO virtual link is essentially extended *through* the security gateway to the point at which the VPN link and AERO link are effectively grafted together by the link-layer address rewriting performed by the security gateway. All AERO messaging services (including route optimization and mobility signaling) are therefore extended to the Client.

In order to support this virtual link grafting, the security gateway (acting as an AERO Server) must keep static neighbor cache entries for all of its associated Clients located on the public Internet. The neighbor cache entry is keyed by the AERO Client's AERO address the same as if the Client were located within the enterprise internetwork. The neighbor cache is then managed in all ways as though the Client were an ordinary AERO Client. This includes the AERO IPv6 ND messaging signaling for Route Optimization and Neighbor Unreachability Detection.

Note that the main difference between a security gateway acting as an AERO Server and an enterprise-internal AERO Server is that the security gateway has at least one enterprise-internal physical interface and at least one public Internet physical interface. Conversely, the enterprise-internal AERO Server has only enterprise-internal physical interfaces. For this reason security gateway proxying is needed to ensure that the public Internet link-layer addressing space is kept separate from the enterprise-internal link-layer addressing space. This is afforded through a natural extension of the security association caching already performed for each VPN client by the security gateway.

Appendix E. Change Log

Changes from -81 to -82:

- o Make DHCPv6 the default (but not exclusive) PD service
- o Support operation with no PD services nor ND Route Information Options
- o Updates to AERO Proxy function

Changes from -80 to -81:

- o Updates to Server and Proxy Extended Route Optimization
- o Updates to AERO Proxy section
- o Cleanups and clarifications

Changes from -79 to -80:

- o Substantial updates to AERO Proxy function
- o Removed ’V’ bit from SLLAO and replaced with ’X’ bit
- o Added concept of Direct, Proxyed, NATed, VPNed and Native underlying interfaces
- o Adjusted route optimization text according to underrlying interface types

Changes from -78 to -79:

- o Neighbors now set UDP Port Number and IP Address in S/TLLAOs to 0 if the node is behind a NAT or otherwise does not wish to update its link-layer address for this underlying interface
- o Introduced "proxy" as a new neighbor cache entry type
- o updated GUE references
- o multipath considerations for error message handling and NUD

Changes from -77 to -78:

- o Added "V" bit to SLLAO flags field for NS messages. V=1 indicates that the NA response must go through the reverse chain of Servers and Relays

- Now including DHCPv6 PD messages as IPv6 ND message options
- Clarified the use of the "P" bit in the RA flags field
- Use of SEND to protect the combined DHCPv6/IPv6ND messages
- Proxy now treats a Client's Servers as the default routers (i.e., instead of using a Relay as the default).

Changes from -76 to -77:

- Now using IPv6 ND NS/NA messaging for route optimization (no longer using Predirect/Redirect)
- Now using combined IPv6 ND/DHCPv6 messaging so autoconfiguration can be conducted in a single message exchange
- Introduced the AERO Proxy construct. Critical for applications such as ATN/IPS

Changes from -75 to -76:

- Bumped version number ahead of expiration deadline

Changes from -74 to -75:

- Bumped version number ahead of expiration deadline

Author's Address

Fred L. Templin (editor)
Boeing Research & Technology
P.O. Box 3707
Seattle, WA 98124
USA

Email: fltemplin@acm.org